USTL

LABORATOIRE D'ANALYSE NUMERIQUE ET D'OPTIMISATION

n° d'ordre : 1252

THESE

Nouveau régime

présentée à
l'Université des Sciences et Technologies de Lille

pour obtenir le titre de

DOCTEUR en MATHEMATIQUES
par

Maria Zélia RAMOS ALVES DA ROCHA

APPLICATIONS DE LA THEORIE DE LA BIORTHOGONALITE

soutenue le 3 février 1994 devant la commission d'examen

Membres du jury

Président :	J. Van Iseghem
Rapporteurs :	A.C. Matos
	M. Prévost
Membres :	M.R. Da Silva
	C. Brezinski

Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3, USTL Flandres-Artois,
59655 Villeneuve d'Ascq - Cedex - FRANCE.

UNIVERSITE DES SCIENCES
ET TECHNOLOGIES DE LILLE

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M. H. LEFEBVRE, M. PARREAU

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PARREAU, J. LOMBARD, M. MIGEON, J. CORTOIS, A.DUBRULLE

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

M. P. LOUIS

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CHAMLEY Hervé	Géotechnique
M. CONSTANT Eugène	Electronique
M. ESCAIG Bertrand	Physique du solide
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. LABLACHE COMBIER Alain	Chimie
M. LOMBARD Jacques	Sociologie
M. MACKE Bruno	Physique moléculaire et rayonnements atmosphériques

M. MIGEON Michel	EUDIL
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du Solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean Pierre	Chimie-Physique
M. BOSCQ Denis	Probabilités
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie Végétale
M. BRASSELET Jean Paul	Géométrie et topologie
M. BREZINSKI Claude	Analyse numérique
M. BRIDOUX Michel	Chimie Physique
M. BRUYELLE Pierre	Géographie
M. CARREZ Christian	Informatique
M. CELET Paul	Géologie générale
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. CROSNIER Yves	Electronique
Mme DACHARRY Monique	Géographie
M. DAUCHET Max	Informatique
M. DEBOURSE Jean Pierre	Gestion des entreprises
M. DEBRABANT Pierre	Géologie appliquée
M. DECLERCQ Roger	Sciences de gestion
M. DEGAUQUE Pierre	Electronique
M. DESCHEPPER Joseph	Sciences de gestion
Mme DESSAUX Odile	Spectroscopie de la réactivité chimique
M. DHAINAUT André	Biologie animale
Mme DHAINAUT Nicole	Biologie animale
M. DJAFARI Rouhani	Physique
M. DORMARD Serge	Sciences Economiques
M. DOUKHAN Jean Claude	Physique du solide
M. DUBRULLE Alain	Spectroscopie hertzienne
M. DUPOUY Jean Paul	Biologie
M. DYMENT Arthur	Mécanique
M. FOCT Jacques Jacques	Métallurgie
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie structurale
M. FRONTIER Serge	Ecologie numérique
M. GLORIEUX Pierre	Physique moléculaire et rayonnements atmosphériques
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie-Physique
M. GRANELLE Jean Jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILBAULT Pierre	Physiologie animale
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. HENRY Jean Pierre	Génie mécanique
M. HERMAN Maurice	Physique spatiale
M. LACOSTE Louis	Biologie Végétale
M. LANGRAND Claude	Probabilités et statistiques

M. LATTEUX Michel	Informatique
M. LAVEINE Jean Pierre	Paléontologie
Mme LECLERCQ Ginette	Catalyse
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique atomique et moléculaire
M. LEROY Jean Marie	Spectrochimie
M. LHENAFF René	Géographie
M. LHOMME Jean	Chimie organique biologique
M. LOUAGE Françis	Electronique
M. LOUCHEUX Claude	Chimie-Physique
M. LUCQUIN Michel	Chimie physique
M. MAILLET Pierre	Sciences Economiques
M. MAROUF Nadir	Sociologie
M. MICHEAU Pierre	Mécanique des fluides
M. PAQUET Jacques	Géologie générale
M. PASZKOWSKI Stéfan	Mathématiques
M. PETIT Francis	Chimie organique
M. PORCHET Maurice	Biologie animale
M. POUZET Pierre	Modélisation - calcul scientifique
M. POVY Lucien	Automatique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. RAMAN Jean Pierre	Sciences de gestion
M. SALMER Georges	Electronique
M. SCHAMPS Joël	Spectroscopie moléculaire
Mme SCHWARZBACH Yvette	Géométrie
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
M. SLIWA Henri	Chimie organique
M. SOMME Jean	Géographie
Melle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. THIEBAULT François	Sciences de la Terre
M. THOMAS Jean Claude	Géométrie - Topologie
M. THUMERELLE Pierre	Démographie - Géographie humaine
M. TILLIEU Jacques	Physique théorique
M. TOULOTTE Jean Marc	Automatique
M. TREANTON Jean René	Sociologie du travail
M. TURRELL Georges	Spectrochimie infrarouge et raman
M. VANEECLOO Nicolas	Sciences Economiques
M. VAST Pierre	Chimie inorganique
M. VERBERT André	Biochimie
M. VERNET Philippe	Génétique
M. VIDAL Pierre	Automatique
M. WALLART Françis	Spectrochimie infrarouge et raman
M. WEINSTEIN Olivier	Analyse économique de la recherche et développement
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ABRAHAM Francis	
M. ALLAMANDO Etienne	Composants électroniques
M. ANDRIES Jean Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BALL Steven	Génétique
M. BART André	Biologie animale
M. BASSERY Louis	Génie des procédés et réactions chimiques
Mme BATTIAU Yvonne	Géographie
M. BAUSIERE Robert	Systèmes électroniques
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique atomique et moléculaire
M. BERNAGE Pascal	Physique atomique, moléculaire et du rayonnement
M. BERTHOUD Arnaud	Sciences Economiques
M. BERTRAND Hugues	Sciences Economiques
M. BERZIN Robert	Analyse
M. BISKUPSKI Gérard	Physique de l'état condensé et cristallographie
M. BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie végétale
M. BOHIN Jean Pierre	Biochimie métabolique et cellulaire
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie civil
M. BOIVIN Jean Claude	Spectrochimie
M. BOUCHER Daniel	Physique
M. BOUQUELET Stéphane	Biologie appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BROCARD Jacques	Chimie
Mme BROUSMICHE Claudine	Paléontologie
M. BUISINE Daniel	Mécanique
M. CAPURON Alfred	Biologie animale
M. CARRE François	Géographie humaine
M. CATTEAU Jean Pierre	Chimie organique
M. CAYATTE Jean Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie structurale
M. CHIVE Maurice	Composants électroniques optiques
M. COMYN Gérard	Informatique théorique
Mme CONSTANT Monique	Composants électroniques et optiques
M. COQUERY Jean Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique nucléaire et corpusculaire
M. COUTURIER Daniel	Chimie organique
M. CRAMPON Norbert	Tectolique géodynamique
M. CURGY Jean Jacques	Biologie
M. DANGOISSE Didier	Physique théorique
M. DE PARIS Jean Claude	Analyse
M. DECOSTER Didier	Composants électroniques et optiques
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELAHAYE Jean Paul	Informatique
M. DELORME Pierre	Physiologie animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
Mme DEMUYNCK Claire	Physique atomique, moléculaire et du rayonnement
M. DENEL Jacques	Informatique
M. DEPREZ Gilbert	Physique du solide - christallographie

M. DERIEUX Jean Claude	Microbiologie
M. DERYCKE Alain	Informatique
M. DESCAMPS Marc	Physique de l'état condensé et cristallographie
M. DEVRAINNE Pierre	Chimie minérale
M. DEWAILLY Jean Michel	Géographie humaine
M. DHAMELINCOURT Paul	Chimie physique
M. DI PERSIO Jean	Physique de l'état condensé et cristallographie
M. DUBAR Claude	Sociologie démographique
M. DUBOIS Henri	Spectroscopie hertzienne
M. DUBOIS Jean Jacques	Géographie
M. DUBUS Jean Paul	Spectrométrie des solides
M. DUPONT Christophe	Vie de la firme
M. DUTHOIT Bruno	Génie civil
Mme DUVAL Anne	Algèbre
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FARVACQUE Jean Louis	Physique de l'état condensé et cristallographie
M. FAUQUEMBERGUE Renaud	Composants électroniques
Mi. FELIX Yves	Mathématiques
M. FERRIERE Jacky	Tectonique - Géodynamique
M. FISCHER Jean Claude	Chimie organique, minérale et analytique
M. FONTAINE Hubert	Dynamique des cristaux
M. FORSE Michel	Sociologie
M. GADREY Jean	Sciences économiques
M. GAMBLIN André	Géographie urbaine, industrielle et démographie
M. GOBLOT Rémi	Algèbre
M. GOURIEROUX Christian	Probabilités et statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie organique
M. GUELTON Michel	Chimie physique
M. GUICHAOUA André	Sociologie
M. HAIMAN Georges	Modélisation,calcul scientifique, statistiques
M. HOUDART René	Physique atomique
M. HUEBSCHMANN Johannes	Mathématiques
M. HUTTNER Marc	Algèbre
M. ISAERT Noël	Physique de l'état condensé et cristallographie
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et statistiques
M. JEAN Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KOENIG Gérard	Sciences de gestion
M. KOSTRUBIEC Benjamin	Géographie
M. KREMBEL Jean	Biochimie
Mme KRIFA Hadjila	Sciences Economiques
M. LANGEVIN Michel	Algèbre
M. LASSALLE Bernard	Embryologie et biologie de la différenciation
M. LE MEHAUTE Alain	Modélisation,calcul scientifique,statistiques
M. LEBFEVRE Yannic	Physique atomique,moléculaire et du rayonnement
M. LECLERCQ Lucien	Chimie physique
M. LEFEBVRE Jacques	Physique
M. LEFEBVRE Marc	Composants électroniques et optiques
M. LEFEVRE Christian	Pétrologie
Melle LEGRAND Denise	Algèbre
M. LEGRAND Michel	Astronomie - Météorologie
M. LEGRAND Pierre	Chimie
Mme LEGRAND Solange	Algèbre
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne

M. LE MAROIS Henri	Vie de la firme
M. LEMOINE Yves	Biologie et physiologie végétales
M. LESCURE François	Algèbre
M. LESENNE Jacques	Systèmes électroniques
M. LOCQUENEUX Robert	Physique théorique
Mme LOPES Maria	Mathématiques
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU François	Sciences économiques
M. MAHIEU Jean Marie	Optique - Physique atomique
M. MAIZIERES Christian	Automatique
M. MANSY Jean Louis	Géologie
M. MAURISSON Patrick	Sciences Economiques
M. MERIAUX Michel	EUDIL
M. MERLIN Jean Claude	Chimie
M. MESMACQUE Gérard	Génie mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MOCHE Raymond	Modélisation,calcul scientifique,statistiques
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie organique
M. MORE Marcel	Physique de l'état condensé et cristallographie
M. MORTREUX André	Chimie organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
M. NIAY Pierre	Physique atomique,moléculaire et du rayonnement
M. NICOLE Jacques	Spectrochimie
M. NOTELET Francis	Systèmes électroniques
M. PALAVIT Gérard	Génie chimique
M. PARSY Fernand	Mécanique
M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée
M. PERTUZON Emile	Physiologie animale
M. PETIT Daniel	Biologie des populations et écosystèmes
M. PLIHON Dominique	Sciences Economiques
M. PONSOLLE Louis	Chimie physique
M. POSTAIRE Jack	Informatique industrielle
M. RAMBOUR Serge	Biologie
M. RENARD Jean Pierre	Géographie humaine
M. RENARD Philippe	Sciences de gestion
M. RICHARD Alain	Biologie animale
M. RIETSCH François	Physique des polymères
M. ROBINET Jean Claude	EUDIL
M. ROGALSKI Marc	Analyse
M. ROLLAND Paul	Composants électroniques et optiques
M. ROLLET Philippe	Sciences Economiques
Mme ROUSSEL Isabelle	Géographie physique
M. ROUSSIGNOL Michel	Modélisation,calcul scientifique,statistiques
M. ROY Jean Claude	Psychophysiologie
M. SALERNO Françis	Sciences de gestion
M. SANCHOLLE Michel	Biologie et physiologie végétales
Mme SANDIG Anna Margarette	
M. SAWERYSYN Jean Pierre	Chimie physique
M. STAROSWIECKI Marcel	Informatique
M. STEEN Jean Pierre	Informatique
Mme STELLMACHER Irène	Astronomie - Météorologie
M. STERBOUL François	Informatique
M. TAILLIEZ Roger	Génie alimentaire
M. TANRE Daniel	Géométrie - Topologie
M. THERY Pierre	Systèmes électroniques
Mme TJOTTA Jacqueline	Mathématiques
M. TOURSEL Bernard	Informatique
M. TREANTON Jean René	Sociologie du travail

M. TURREL Georges	Spectrochimie infrarouge et raman
M. VANDIJK Hendrik	
Mme VAN ISEGHEM Jeanine	Modélisation,calcul scientifique,statistiques
M. VANDORPE Bernard	Chimie minérale
M. VASSEUR Christian	Automatique
M. VASSEUR Jacques	Biologie
Mme VIANO Marie Claude	
M. WACRENIER Jean Marie	Electronique
M. WARTEL Michel	Chimie inorganique
M. WATERLOT Michel	géologie générale
M. WEICHERT Dieter	Génie mécanique
M. WERNER Georges	Informatique théorique
M. WIGNACOURT Jean Pierre	
M. WOZNIAK Michel	Spectrochimie
Mme ZINN JUSTIN Nicole	Algèbre

A mon père
Dr. Mário Alves da Rocha

REMERCIEMENTS

Je tiens tout d'abord à exprimer toute ma gratitude à Mr. Claude Brezinski, Professeur à l'Université des Sciences et Technologies de Lille (U.S.T.L.) - France, pour avoir bien voulu me suivre dans mes travaux de recherche et pour m'avoir si bien accueilli et intégré au sein du Laboratoire d'Analyse Numérique et d'Optimisation (A.N.O.).

Je tiens particulièrement à remercier Mr. Manuel Rogério de Jesus da Silva, Professeur à l'Université de Porto - Portugal, pour m'avoir introduit et poussé à la recherche et pour m'avoir conseillé le Laboratoire A.N.O. à Lille. Je le remercie aussi d'avoir accepté de prendre part au Jury.

Je remercie vivement Madame Jeannette Van Iseghem, Professeur à l'U.S.T.L., pour l'honneur qu'elle me fait en présidant ce Jury. Je la remercie également pour les commentaires suggestifs portés sur ce travail.

Mes remerciements vont de même à Mademoiselle Ana Cristina Mendes Mena de Matos, Professeur à l'Université de Porto - Portugal, pour avoir bien voulu accepter de juger cette thése et pour ses pertinentes suggestions.

Je remercie également Mr. Marc Prévost, Professeur à l'Université du Littoral - France, qui a bien accepté de prendre part au Jury et de juger ce travail.

Je remercie tous ceux qui, à un moment donné, m'ont été d'un certain apport dans la réalisation de cette thése, notamment Mademoiselle Gabriela Sansigre, Professeur à l'Université de Madrid, et Mr. Hassan Sadok, Maître de Conférences à l'U.S.T.L., pour leur collaboration amicale.

Je tiens à remercier tous les membres du Laboratoire A.N.O. et la Sécrétaire Madame Françoise Tailly pour leur accueil et leur gentillesse dont ils ont fait montre pendant mon séjour à Lille.

Je remercie le "Grupo de Matemática Aplicada da Faculdade de Ciências" de l'Université de Porto - Portugal, qui m'a permis de rester toutes ces années à Lille.

Je remercie encore le "Programa Ciência da Junta Nacional de Investigação Científica e Tecnológica" pour le soutien financier.

INTRODUCTION

Claude Brezinski dans ses travaux sur les applications de la théorie de la biorthogonalité à l'analyse numérique [3], a défini, entre autres, une généralisation des approximants de type Padé pour les séries de fonctions et la notion des polynômes biorthogonaux adjacents. Il a aussi déduit un ensemble de relations de récurrence qui permettent le calcul de ces éléments.

Il était donc important de faire l'implémentation de ces relations avec comme objectif le calcul effectif des approximants de type Padé généralisés et des polynômes biorthogonaux adjacents.

Dans les travaux de cette thèse, nous distinguerons deux parties essentielles.

Dans la première, nous nous intéresserons au problème du calcul d'une suite d'approximants de type Padé généralisés. Ces approximants sont définis par rapport aux séries de fonctions de la même manière que les approximants de type Padé sont définis par rapport aux séries de puissances. Cependant, il existe une différence cruciale; à savoir que dans les approximants généralisés, nous considérons des fonctionnelles quelconques, tandis que dans les approximants de type Padé, les fonctionnelles correspondent toujours à l'interpolation de Lagrange ou de Hermite.

Nous présenterons deux méthodes pour calculer les approximants généralisés. La première utilise la méthode de bordage par blocs développée par Brezinski et Redivo Zaglia [6], et la deuxième utilise les relations de récurrence de la biorthogonalité déduites par Brezinski [3].

L'implémentation de la deuxième méthode pose des problèmes d'encombrement de mémoire et de volume de calcul, que nous réussirons à minimiser.

Nous développerons des logiciels pour implémenter les deux méthodes de calcul. La première sera implémentée en *Fortran* (voir annexe A) et la deuxième sera implémentée en *Mathematica* (voir da Rocha [7] et annexe C).

Le logiciel *Mathematica* permet de faire aussi bien du calcul symbolique que du calcul numérique. Dans les deux cas nous pouvons faire du calcul exact, du calcul approché de basse précision ou du calcul approché de haute précision.

Faire du calcul exact est très souhaitable, mais cela prend en général trop de temps. Par contre, le calcul approché de haute précision est beaucoup plus rapide et permet de maîtriser l'effet cumulatif des erreurs d'arrondi de telle sorte que ces erreurs ne viennent pas entâcher les chiffres exacts des approximants.

Le véritable problème qui se pose par rapport aux approximants de type Padé généralisés est de savoir comment choisir les fonctionnelles qui leur sont associées de sorte que les approximants convergent vers la série plus vite que la suite des sommes partielles. Nous nous demandons aussi sous quelles conditions ces fonctionnelles existent. Est-ce qu'elles sont définies de façon unique ou non ? Saurions nous choisir les fonctionnelles qui produisent la convergence la plus rapide ?

Le premier pas pour aborder toutes ces questions est de trouver des exemples.

Nous le ferons à travers plusieurs séries de puissances et séries de fonctions par rapport auxquelles nous trouverons des suites de fonctionnelles qui produisent des approximants qui convergent vers la série et plus vite que la suite des sommes partielles.

La deuxième partie de ce travail est consacrée au problème du calcul des polynômes biorthogonaux adjacents.

Ces polynômes admettent comme cas particuliers importants, les polynômes orthogonaux adjacents [2, 9], les polynômes orthogonaux vectoriels de dimension d définis et étudiés par Van Iseghem [10, 11], et les polynômes orthogonaux vectoriels de dimension -1 définis et étudiés par Brezinski [4].

Nous commencerons par déduire toutes les relations de récurrence d'ordre deux d'un certain type. Nous trouverons douze relations parmi lesquelles quatre avaient déjà été déduites par Brezinski [3]. Nous écrivons ces relations dans les cas particuliers cités.

Dans le cas général, l'utilisation simultanée des douze relations de récurrence s'avére sans intérêt. En effet, nous ne voyons pas comment on pourrait utiliser l'ensemble des relations pour calculer la table des polynômes biorthogonaux ou des suites particulières des polynômes biorthogonaux, en ne gardant qu'un nombre fixe de polynômes en mémoire. Ceci est possible dans les cas particuliers des polynômes orthogonaux adjacents et des polynômes orthogonaux vectoriels de dimension -1 en ne gardant que deux polynômes à la fois.

Nous nous intéresserons aussi au problème du calcul des coefficients des relations de récurrence. Et, nous déduirons des algorithmes du type QD, qui permettent leur calcul. Enfin, nous écrirons ces algorithmes dans les cas particuliers cités.

Table des matières

1

Approximants de type Padé généralisés

1.1 Définition

Considérons la série de fonctions

$$f(t) = \sum_{i=0}^{\infty} c_i \, g_i(t) \tag{1.1}$$

où $c_i \in +\mathbf{C}$ et $g_i = g_i(t)$ est une fonction complexe de la variable complexe.

Il existe une et une seule fonctionnelle linéaire c sur l'espace vectoriel des polynômes complexes, qui satisfait:

$$c(u_i(x)) = c_i \qquad , i = 0, 1, \ldots \tag{1.2}$$

où $u_i(x)$ est un polynôme de degré exactement i.

En effet, si on écrit $u_i(x)$ dans la base canonique

$$u_i(x) = a_i^{(i)} x^i + a_{i-1}^{(i)} x^{i-1} + \cdots + a_0^{(i)}$$

les moments $d_i = c(x^i)$, de la fonctionnelle c, sont les solutions du système 1.2, qui s'écrit sous la forme

$$a_i^{(i)} d_i + a_{i-1}^{(i)} d_{i-1} + \cdots + a_0^{(i)} d_0 = c_i \qquad , i = 0, 1, \ldots$$

Comme $a_i^{(i)} \neq 0$, ce système infini triangulaire inférieur admet une et une seule solution.

Souvent on appelle c_i, les moments modifiés de la fonctionnelle c.

Avec cette définition de c, on peut écrire

$$f(t) = c(G(x,t))$$

où c agit sur la variable x, t est interprété comme un paramètre et

$$G(x,t) = \sum_{i=0}^{\infty} u_i(x) \, g_i(t) \tag{1.3}$$

est la fonction génératrice de la famille $\{g_i(t)\}_i$ dans la base $\{u_i(x)\}_i$.

Ainsi, le calcul de $f(t)$ pour t fixé, se réduit au calcul de $c(G(x,t))$, qui peut être interprété comme une intégration formelle de la fonction $G(x) = G(x,t)$, obtenue en remplaçant la fonctionnelle d'intégration habituelle, par la fonctionnelle plus générale c.

En analyse numérique, le calcul approché d'une intégrale définie s'effectue classiquement en remplaçant la fonction à intégrer par son polynôme d'interpolation d'Hermite, et en intégrant celui-ci.

Mais, nous pouvons généraliser cette idée, comme il a été fait par exemple par Brezinski [3], et considérer un polynôme d'interpolation P_n, de degré au plus n, pas nécessairement d'Hermite, mais qui satisfait les conditions d'interpolation générales suivantes:

$$L_i(P_n(x)) = L_i(G(x,t)) \qquad ,i = 0,1,\ldots,n$$

où $\{L_i\}_{i=0,1,\ldots}$ est une famille de fonctionnelles linéaires quelconques [1]. Ensuite, $f(t)$ est approché par $c(P_n)$.

Cherchons une expression pour l'approximant $c(P_n)$.

Écrivons le polynôme P_n dans la base $\{u_i\}$

$$P_n(x) = b_0^{(n)}u_0(x) + \cdots + b_n^{(n)}u_n(x). \tag{1.4}$$

Les conditions générales d'interpolation s'écrivent sous la forme

$$b_0^{(n)}L_i(u_0(x)) + \cdots + b_n^{(n)}L_i(u_n(x)) = L_i(G(x,t)) \quad ,i = 0,1,\ldots,n. \tag{1.5}$$

Ces équations constituent un système de $n+1$ équations linéaires à $n+1$ inconnues $b_i^{(n)}, i = 0,1,\ldots,n$, les coefficients de P_n.

P_n existe et est unique ssi

$$D_{n+1}^{(0,0)} = \begin{vmatrix} L_0(u_0) & \cdots & L_0(u_n) \\ \cdots & & \cdots \\ L_n(u_0) & \cdots & L_n(u_n) \end{vmatrix} \neq 0.$$

A partir de 1.4 et 1.5, nous pouvons écrire P_n comme rapport de deux déterminants:

$$P_n(x) = -\begin{vmatrix} 0 & u_0(x) & \cdots & u_n(x) \\ L_0(G(x,t)) & L_0(u_0(x)) & \cdots & L_0(u_n(x)) \\ \cdots & \cdots & & \cdots \\ L_n(G(x,t)) & L_n(u_0(x)) & \cdots & L_n(u_n(x)) \end{vmatrix} / D_{n+1}^{(0,0)}.$$

Finalement, en appliquant c aux deux membres de l'égalité précédente, nous obtenons l'expression recherchée:

$$c(P_n) = -\begin{vmatrix} 0 & c(u_0(x)) & \cdots & c(u_n(x)) \\ L_0(G(x,t)) & L_0(u_0(x)) & \cdots & L_0(u_n(x)) \\ \cdots & \cdots & & \cdots \\ L_n(G(x,t)) & L_n(u_0(x)) & \cdots & L_n(u_n(x)) \end{vmatrix} / D_{n+1}^{(0,0)}. \tag{1.6}$$

Et nous remarquons que:

$$c(P_n) \text{ existe} \quad \text{ssi} \quad P_n \text{ existe} \quad \text{ssi} \quad D_{n+1}^{(0,0)} \neq 0.$$

[1] Les fonctionnelles L_i agissent sur la variable x; t est interprété comme un paramètre.

La question qui se pose maintenant est de savoir quel est l'ordre de la différence $c(P_n) - f(t)$.

Dans l'égalité (1.6), remplaçons $G(x,t)$ par son développement en série (1.3). Comme les fonctionnelles L_i sont linéaires et agissent sur la variable x, et un déterminant est linéaire par rapport à chaque colonne, nous obtenons:

$$c(P_n) = \sum_{i=0}^{\infty} e_i\, g_i(t)$$

où

$$e_i = -\begin{vmatrix} 0 & c_0 & \cdots & c_n \\ L_0(u_i(x)) & L_0(u_0(x)) & \cdots & L_0(u_n(x)) \\ \cdots & \cdots & & \cdots \\ L_n(u_i(x)) & L_n(u_0(x)) & \cdots & L_n(u_n(x)) \end{vmatrix} / D_{n+1}^{(0,0)}.$$

Il est facile de vérifier que $e_i = c_i$, $i = 0, 1, \ldots, n$; donc on peut écrire:

$$c(P_n) = f(t) + O(g_{n+1}(t))$$

où $O(g_{n+1}(t))$ dénote la série $\sum_{i=n+1}^{\infty}(e_i - c_i)\, g_i(t)$.

Si $G(x,t) = 1/(1-xt)$, fonction génératrice de la famille $\{t^i\}$ dans la base canonique, et si

$$L_i(f) = f^{(j)}(z_k)$$

où $\{z_k\}$ est une suite de nombres complexes distincts, $\{n_k\}$ est une suite d'entiers non négatifs, $k = 0, 1, \ldots$; $j = 0, 1, \ldots, n_k$ et

$$i = j + \sum_{l=0}^{k-1}(n_l + 1),$$

alors $c(P_n)$ est l'approximant de type Padé $(n/(n+1))$ de la série de puissances $f(t) = \sum_{i=0}^{\infty} c_i\, t^i$, avec polynôme générateur

$$v_{n+1}(x) = (x - z_{p+1})^{l+1} \prod_{k=0}^{p} (x - z_k)^{n_k+1},$$

où $p = -1, 0, 1, \ldots$ dépend de n et $0 \leq l \leq n_{p+1}$. Remarquons que le degré de v_{n+1} est égal à $n+1 = 1 + l + \sum_{k=0}^{p}(n_k + 1)$, et que $n_k + 1$ est la multiplicité du point z_k.

Alors, nous appellerons $c(P_n)$ l'approximant de type Padé généralisé d'ordre n de la série de fonctions $f(t)$ et on le dénotera par $(n)_f(t)$. Nous remarquons que dans le cas général $(n)_f(t)$ n'est pas une fonction rationnelle.

Les approximants de Padé [2] ne sont pas un cas particulier de cette généralisation. En effet, on a retrouvé les approximants de type Padé, à condition que $1/(1-xt)$ soit la fonction génératrice et les zéros de chaque polynôme générateur soient aussi des zéros du polynôme générateur suivant. Or ceci est impossible quand les polynômes générateurs sont des polynômes orthogonaux par rapport à une fonctionnelle c.

Ces approximants généralisés peuvent être introduits d'une autre façon. En effet, $(n)_f(t)$ peut être défini comme la combinaison linéaire des fonctions $L_i(G(x,t))$, $i = 0, 1, \ldots, n$, dont le développement en série, par rapport à la famille $\{g_i(t)\}_i$, coïncide avec celui de f aussi loin que possible, c'est-à-dire compte tenu du nombre de coefficients à déterminer.
Ainsi,

$$(n)_f(t) = a_0^{(n)} L_0(G(x,t)) + \cdots + a_n^{(n)} L_n(G(x,t)) \tag{1.7}$$

où les coefficients $\{a_i^{(n)}\}_{i=0,1,\ldots,n}$ sont déterminés de façon à ce que

$$(n)_f(t) - f(t) = O(g_{n+1}(t)).$$

Si nous remplaçons dans l'égalité précédente, $(n)_f(t)$ par son expression (1.7) et ensuite $f(t)$ et $G(x,t)$ par leurs développements en série (1.1) et (1.3), nous obtenons:

$$\sum_{i=0}^{\infty} (a_0^{(n)} L_0(u_i(x)) + \ldots + a_n^{(n)} L_n(u_i(x)) - c_i) g_i(t) = O(g_{n+1}(t)),$$

qui est équivalent aux équations:

$$a_0^{(n)} L_0(u_i(x)) + \cdots + a_n^{(n)} L_n(u_i(x)) = c_i \quad , i = 0, 1, \ldots, n.$$

Donc, $\{a_i^{(n)}\}_{i=0,1,\ldots,n}$ sont les solutions du système linéaire suivant:

$$\begin{pmatrix} L_0(u_0) & \cdots & L_n(u_0) \\ \cdots & & \cdots \\ L_0(u_n) & \cdots & L_n(u_n) \end{pmatrix} \begin{pmatrix} a_0^{(n)} \\ \vdots \\ a_n^{(n)} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix}. \tag{1.8}$$

Ce système admet une et une seule solution ssi son déterminant $D_{n+1}^{(0,0)} \neq 0$, condition nécessaire et suffisante de l'existence et de l'unicité de l'approximant $(n)_f(t)$.
Remarquons que (1.7) et (1.8) peuvent être déduits directement de l'expression de $(n)_f(t)$ comme rapport de deux déterminants et vice-versa.

1.2 Calcul en utilisant la méthode de bordage par blocs

Commençons par rappeler brièvement la méthode de bordage par blocks développée par Brezinski et Redivo Zaglia [5, 6], pour résoudre récursivement des systèmes d'équations linéaires de dimensions croissantes.
Soit M_n une matrice de dimensions $k_n \times k_n$. Considérons la matrice M_{n+1}, $(k_n + k_m) \times (k_n + k_m)$, donnée par

$$M_{n+1} = \begin{pmatrix} M_n & Q_n \\ P_n & R_n \end{pmatrix},$$

où Q_n, P_n et R_n sont des matrices de dimensions $k_n \times k_m$, $k_m \times k_n$ et $k_m \times k_m$ respectivement.

Supposons que M_n et M_{n+1} sont inversibles et considérons les systèmes d'équations linéaires :

$$M_n a_n = d_n$$
$$M_{n+1} a_{n+1} = d_{n+1} = \begin{pmatrix} d_n \\ f_n \end{pmatrix},$$

où f_n est un vecteur de dimension k_m.

Alors, a_{n+1} peut être calculé récursivement à partir de a_n, en utilisant l'égalité suivante:

$$a_{n+1} = \begin{pmatrix} a_n \\ 0 \end{pmatrix} + \begin{pmatrix} -M_n^{-1}Q_n \\ I \end{pmatrix} B_n^{-1}(f_n - P_n a_n), \tag{1.9}$$

où $B_n = R_n - P_n M_n^{-1} Q_n$ est une matrice $k_m \times k_m$.

Si on suppose que M_n est inversible, on peut démontrer que M_{n+1} est inversible ssi B_n est inversible.

Nous voulons calculer une suite d'approximants de type Padé généralisés en utilisant cette méthode. Pour calculer $(n)_f(t)$, il suffit de résoudre le système (1.8), dont les solutions sont les coefficients de la combinaison linéaire (1.7).

Supposons que $D_{n+1}^{(0,0)} \neq 0$, c'est-à-dire $(n)_f(t)$ existe et est unique. Si $D_{n+1+l}^{(0,0)} = 0$ pour $l = 1, \ldots, k-1$, où $k > 1$ est fixé, alors les approximants $(n+l)_f(t)$ n'existent pas. Mais, si $D_{n+k+1}^{(0,0)} \neq 0$, l'approximant $(n+k)_f(t)$ existe, est unique et ses coefficients sont les solutions d'un système qui s'obtient du système (1.8) en ajoutant k lignes et k colonnes à la matrice des coefficients et k éléments au terme indépendant:

$$\begin{pmatrix} L_0(u_0) & \cdots & L_n(u_0) & L_{n+1}(u_0) & \cdots & L_{n+k}(u_0) \\ \vdots & & \vdots & \vdots & & \vdots \\ L_0(u_n) & \cdots & L_n(u_n) & L_{n+1}(u_n) & \cdots & L_{n+k}(u_n) \\ L_0(u_{n+1}) & \cdots & L_n(u_{n+1}) & L_{n+1}(u_{n+1}) & \cdots & L_{n+k}(u_{n+1}) \\ \vdots & & \vdots & \vdots & & \vdots \\ L_0(u_{n+k}) & \cdots & L_n(u_{n+k}) & L_{n+1}(u_{n+k}) & \cdots & L_{n+k}(u_{n+k}) \end{pmatrix} \begin{pmatrix} a_0^{(n+k)} \\ \vdots \\ a_n^{(n+k)} \\ a_{n+1}^{(n+k)} \\ \vdots \\ a_{n+k}^{(n+k)} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_n \\ c_{n+1} \\ \vdots \\ c_{n+k} \end{pmatrix}.$$

La méthode de bordage par blocs calcule les solutions de ce dernier système à partir des solutions du système (1.8), d'une façon récursive.

En effet, l'égalité (1.9) nous permet d'écrire

$$\begin{pmatrix} a_0^{(n+k)} \\ \vdots \\ a_n^{(n+k)} \\ a_{n+1}^{(n+k)} \\ \vdots \\ a_{n+k}^{(n+k)} \end{pmatrix} = \begin{pmatrix} a_0^{(n)} \\ \vdots \\ a_n^{(n)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + S_{n+k}, \tag{1.10}$$

où S_{n+k} est un vecteur de dimensions $n+k+1$ donné par

$$S_{n+k} = \begin{pmatrix} -(M_{n+1,n+1}^{(0,0)})^{-1} M_{n+1,k}^{(n+1,0)} \\ I \end{pmatrix} B_n^{-1} \left(\begin{pmatrix} c_{n+1} \\ \vdots \\ c_{n+k} \end{pmatrix} - M_{k,n+1}^{(0,n+1)} \begin{pmatrix} a_0^{(n)} \\ \vdots \\ a_n^{(n)} \end{pmatrix} \right),$$

où B_n est une matrice de dimensions $k \times k$ donnée par

$$B_n = M_{k,k}^{(n+1,n+1)} - M_{k,n+1}^{(0,n+1)} (M_{n+1,n+1}^{(0,0)})^{-1} M_{n+1,k}^{(n+1,0)}$$

et

$$M_{n,m}^{(i,j)} = \begin{pmatrix} L_i(u_j) & \cdots & L_{i+m-1}(u_j) \\ \cdots & & \cdots \\ L_i(u_{j+n-1}) & \cdots & L_{i+m-1}(u_{j+n-1}) \end{pmatrix}.$$

Si nous multiplions scalairement les deux membres de l'égalité (1.10) par

$$F_{n+k}(t) = (L_0(G(x,t)), \ldots, L_n(G(x,t)), L_{n+1}(G(x,t)), \ldots, L_{n+k}(G(x,t)))^T,$$

nous obtenons :

$$(n+k)_f(t) = (n)_f(t) + S_{n+k} F_{n+k}(t).$$

Nous remarquons que les conditions d'applicabilité de cette méthode coïncident avec les conditions d'existence et d'unicité des approximants à calculer.

1.2.1 Implémentation et algorithme

Nous voulons implémenter en *Fortran* le calcul d'une suite d'approximants de type Padé généralisés en utilisant la méthode de bordage par blocs. Cette dernière méthode a été implementée par Brezinski et Redivo Zaglia [5] dans la subroutine BLBORD dont nous nous en servons.

Algorithme de calcul de $((n)_f(t))_{n=0,1,\ldots}$

- nm est la valeur maximum de n.
- $n \leftarrow 0$.
- $f_0 \leftarrow$ vecteur vide.
- $M_0 \leftarrow$ matrice vide.
- $d_0 \leftarrow$ vecteur vide.
- lire t

1. • Définir L_n.
   • Calculer $L_n(G(x,t))$.
   • $f_{n+1} \leftarrow \begin{pmatrix} f_n \\ L_n(G(x,t)) \end{pmatrix}$.
   • Calculer u_n.
   • Calculer c_n.
   • Calculer $p_n \leftarrow (L_0(u_n), \ldots, L_{n-1}(u_n))$.
   • Calculer $q_n \leftarrow (L_n(u_0), \ldots, L_n(u_{n-1}))^T$.
   • Calculer $r_n \leftarrow L_n(u_n)$.
   • $M_{n+1} \leftarrow \begin{pmatrix} M_n & q_n \\ p_n & r_n \end{pmatrix}$, $d_{n+1} \leftarrow \begin{pmatrix} d_n \\ c_n \end{pmatrix}$.
   $S_{n+1} \leftarrow M_{n+1} a_{n+1} = d_{n+1}$.
   • **Si** S_{n+1} est singulier, **alors aller à** 2.
   • **Si** S_{n+1} est régulier, **alors** sa solution $a_{n+1} = (a_0^{(n)}, \ldots, a_n^{(n)})$ est calculé en utilisant la subroutine BLBORD.
   • Calculer $(n)_f(t) = (a_{n+1}, f_{n+1})$.
   • **écrire** $(n)_f(t)$.
2. • $f_n \leftarrow f_{n+1}$.
   • $M_n \leftarrow M_{n+1}$.
   • $d_n \leftarrow d_{n+1}$.
   • $n \leftarrow n+1$.
   • **Si** $n <= nm$ et nous voulons calculer l'approximant suivant, **alors aller à** 1, **si non stop.**

Le logiciel *Fortran* correspondant à cet algorithme est donné dans l'annexe A. Il est constitué par :

- Le programme principal GENPADETYPE.
- Les subroutines BASISPOLY, BETABOR, BLBORD, DERIVEPOLY, FUNGENERATING, GAUSS, INVERS, MODIFIEDMOMENTS, PRIMITIVEPOLY, READPOLY, RCPROD, SYSTEM, WRITEPOLY et WRITING.
- Les fonctions PFACT, ELFUNGEN, F, FUNPOLY, GPADETYPE, HORNER et INNERG.

Les subroutines BETABOR, GAUSS, INVERS et RCPROD, et la fonction INNERG sont utilisées uniquement par la subroutine BLBORD. Ces modules appartiennent au software de [5].
Le programme principal établit un dialogue entre l'utilisateur et le logiciel, en commençant par proposer les:

- Menu des séries.
- Menu des fonctions génératrices.
- Menu des bases de polynômes.
- Menu des fonctionnelles L_k.

disponibles.

A chaque menu correspondent plusieurs options au choix de l'utilisateur.

Le code des modules BASISPOLY, FUNGENERATING, ELFUNGEN, F, FUNPOLY et MODIFIEDMOMENTS est écrit en accord avec les options des différents menus.

Quand l'utilisateur veut tester de nouveaux exemples, il doit ajouter les options nécessaires aux menus, et ajouter le code correspondant à ces options aux modules cités ci-dessus.

Les calculs sont faits en double précision.

1.2.2 Exemples

Commençons par l'exemple utilisé par Brezinski [1, 2] pour montrer que les approximants de Padé ne sont pas toujours meilleurs que les approximants de type Padé, à condition qu'un bon choix des polynômes générateurs soit fait.

Il s'agit de la série de puissances

$$\frac{\log(1+t)}{t} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i+1} t^i,$$

pour laquelle la suite

$$(((k-1)/k)_f(t))_{k=1,2,\ldots}$$

d'approximants de type Padé [2], avec polynôme générateur

$$v_k(t) = \prod_{i=1}^{k} (t + \frac{1}{i})$$

converge pour tout $t \in [0,1]$.

Le choix des zéros

$$-1/i\ ,\ i = 1, \ldots, k$$

des polynômes générateurs correspond au choix des fonctionnelles initialles

$$L_i(f) = f(-\frac{1}{i+1})\ ,i = 0, \ldots, k-1.$$

A partir de cette série de puissances, nous considérons les exemples 1, 2, 3 et 4 qui suivent :

[2] La suite d'approximants de Padé $([(k-1)/k]_f(t))_{k=1,2,\ldots}$ converge pour tout $t \in C -]-\infty, -1]$.

Tableau 1.1: $f(t) = \log(1+t)/t, G(x,t) = 1/(1-xt), c_i = (-1)^i/(i+1), u_i(x) = x^i$.

	$t = 1/10$			$t = 1/2$	
n	Exemple 1 Choix A	Exemple 2 Choix B	n	Exemple 1 Choix A	Exemple 2 Choix B
0	$4.6\ 10^{-2}$	$2.4\ 10^{-2}$	0	$1.8\ 10^{-1}$	$1.0\ 10^{-1}$
1	$7.6\ 10^{-4}$	$5.1\ 10^{-4}$	1	$1.3\ 10^{-2}$	$9.5\ 10^{-3}$
2	$2.4\ 10^{-5}$	$9.3\ 10^{-6}$	2	$1.7\ 10^{-3}$	$7.8\ 10^{-4}$
3	$9.2\ 10^{-8}$	$1.5\ 10^{-7}$	3	$5.6\ 10^{-6}$	$5.6\ 10^{-5}$
4	$2.5\ 10^{-8}$	$2.0\ 10^{-9}$	4	$3.4\ 10^{-5}$	$3.5\ 10^{-6}$
5	$1.3\ 10^{-9}$	$2.4\ 10^{-11}$	5	$8.0\ 10^{-6}$	$2.0\ 10^{-7}$
6	$1.1\ 10^{-10}$	$1.2\ 10^{-12}$	6	$2.7\ 10^{-6}$	$1.0\ 10^{-8}$
7	$1.3\ 10^{-9}$	$3.1\ 10^{-12}$	7	$9.1\ 10^{-7}$	$4.7\ 10^{-10}$
8	$7.4\ 10^{-8}$	$7.2\ 10^{-11}$	8	$4.0\ 10^{-7}$	$2.3\ 10^{-11}$
9	$2.3\ 10^{-6}$	$7.5\ 10^{-10}$	9	$2.4\ 10^{-6}$	$1.1\ 10^{-10}$
10	$5.1\ 10^{-4}$	$9.3\ 10^{-9}$	10	$5.6\ 10^{-4}$	$2.1\ 10^{-9}$
11	$7.1\ 10^{-2}$	$4.6\ 10^{-8}$	11	$1.9\ 10^{-2}$	$5.2\ 10^{-9}$
12	$2.3\ 10^{+0}$	$2.5\ 10^{-8}$	12	$2.5\ 10^{+0}$	$2.7\ 10^{-7}$

Exemple 1
$f(t) = \log(1+t)/t$
$G(x,t) = 1/(1-xt)$
$c_i = (-1)^i/(i+1)$
$u_i(x) = x^i$
$L_i(f) = f(-\frac{1}{i+1})$ (Choix A)
Les approximants sont de type Padé.

Exemple 2
$f(t) = \log(1+t)/t$
$G(x,t) = 1/(1-xt)$
$c_i = (-1)^i/(i+1)$
$u_i(x) = x^i$
$L_i(f) = \int_{-\frac{1}{i+1}}^{-\frac{1}{i+2}} f(x)dx$ (Choix B)
Les approximants sont de type Padé généralisés.

Nous avons calculé, pour les exemples 1 et 2, les suites des approximants correspondants à plusieurs valeurs de t, et nous observons que le choix B des fonctionnelles produit des erreurs relatives plus petites et des résultats plus stables que le choix A, comme nous pouvons le constater dans le tableau 1.1, où nous montrons seulement les résultats correspondants à $t = 1/10$ et $t = 1/2$.

Tableau 1.2: $f(t) = \log(1+t)/t, G(x,t) = \log(1+xt)/t, c_i = 1, u_i(x) = x^i$.

	$t = 1/10$	
n	Exemple 3 Choix A	Exemple 4 Choix B
0	$2.1\ 10^{+0}$	$1.8\ 10^{+0}$
1	$1.6\ 10^{-1}$	$1.4\ 10^{-1}$
2	$1.5\ 10^{-2}$	$1.2\ 10^{-2}$
3	$1.4\ 10^{-3}$	$1.2\ 10^{-3}$
4	$1.4\ 10^{-4}$	$1.2\ 10^{-4}$
5	$1.4\ 10^{-5}$	$1.2\ 10^{-5}$
6	$1.4\ 10^{-6}$	$4.1\ 10^{-6}$
7	$1.9\ 10^{-6}$	$1.7\ 10^{-4}$
8	$1.2\ 10^{-4}$	$5.3\ 10^{-3}$
9	$4.4\ 10^{-3}$	$3.8\ 10^{-2}$
10	$1.6\ 10^{+0}$	$5.9\ 10^{+0}$

Le même phénomène ne se vérifie pas si nous assoçions à cette série une fonction génératrice et une suite de moments différents (voir exemples 3 et 4 et le tableau 1.2).

Exemple 3
$f(t) = \log(1+t)/t$
$G(x,t) = \log(1+xt)/t$
$c_i = 1$
$u_i(x) = x^i$
$L_i(f) = f(-\frac{1}{i+1})$ (Choix A)
Les approximants sont de type Padé généralisés.

Exemple 4
$f(t) = \log(1+t)/t$
$G(x,t) = \log(1+xt)/t$
$c_i = 1$
$u_i(x) = x^i$
$L_i(f) = \int_{-\frac{1}{i+1}}^{-\frac{1}{i+2}} f(x)dx$ (Choix B)
Les approximants sont de type Padé généralisés.

Les mêmes essais avec la série $f(t) = \exp(-t)$ (voir exemples 5, 6, 7 et 8 et les tableaux 1.3 et 1.4), n'ont pas révélé de différences entre les résultats correspondants aux choix A et B.

Il est étonnant de remarquer comment le choix A des fonctionnelles, adapté à la série $f(t) = \log(1+t)/t$, produit aussi des bons résultats dans le cas de la série $f(t) = \exp(-t)$.

Le choix B des fonctionnelles a été complètement fait au hasard.

Tableau 1.3: $f(t) = \exp(-t), G(x,t) = 1/(1-xt), c_i = (-1)^i/i!, u_i(x) = x^i$.

	$t = 1/10$			$t = 1/2$	
n	Exemple 5 Choix A	Exemple 6 Choix B	n	Exemple 5 Choix A	Exemple 6 Choix B
0	$4.7\ 10^{-3}$	$2.8\ 10^{-2}$	0	$9.9\ 10^{-2}$	$2.0\ 10^{-1}$
1	$4.7\ 10^{-3}$	$3.7\ 10^{-3}$	1	$9.9\ 10^{-2}$	$8.0\ 10^{-2}$
2	$7.1\ 10^{-5}$	$5.8\ 10^{-6}$	2	$4.9\ 10^{-3}$	$8.5\ 10^{-4}$
3	$4.4\ 10^{-6}$	$3.8\ 10^{-6}$	3	$2.0\ 10^{-3}$	$1.8\ 10^{-3}$
4	$1.0\ 10^{-7}$	$4.5\ 10^{-8}$	4	$1.9\ 10^{-4}$	$8.4\ 10^{-5}$
5	$1.3\ 10^{-9}$	$1.4\ 10^{-9}$	5	$1.4\ 10^{-5}$	$1.6\ 10^{-5}$
6	$4.6\ 10^{-11}$	$6.1\ 10^{-11}$	6	$2.2\ 10^{-6}$	$1.3\ 10^{-6}$
7	$4.0\ 10^{-10}$	$3.3\ 10^{-10}$	7	$1.1\ 10^{-8}$	$5.4\ 10^{-8}$
8	$3.8\ 10^{-10}$	$5.7\ 10^{-10}$	8	$1.2\ 10^{-8}$	$1.6\ 10^{-8}$
9	$4.7\ 10^{-8}$	$2.0\ 10^{-7}$	9	$6.7\ 10^{-8}$	$3.9\ 10^{-7}$
10	$5.6\ 10^{-7}$	$3.1\ 10^{-7}$	10	$7.8\ 10^{-7}$	$1.9\ 10^{-8}$
11	$4.2\ 10^{-7}$	$1.8\ 10^{-5}$	11	$4.6\ 10^{-5}$	$4.7\ 10^{-5}$
12	$1.4\ 10^{-5}$	$2.3\ 10^{-4}$	12	$3.8\ 10^{-4}$	$1.0\ 10^{-3}$
13	$9.9\ 10^{-5}$	$1.3\ 10^{-3}$	13	$4.9\ 10^{-3}$	$3.7\ 10^{-2}$
14	$8.5\ 10^{-4}$	$1.2\ 10^{-1}$	14	$1.9\ 10^{-1}$	$8.4\ 10^{-1}$
15	$5.6\ 10^{-2}$	$5.5\ 10^{-1}$	15	$9.9\ 10^{-1}$	$7.6\ 10^{-1}$

Exemple 5
$f(t) = \exp(-t)$
$G(x,t) = 1/(1-xt)$
$c_i = (-1)^i/i!$
$u_i(x) = x^i$
$L_i(f) = f(-\frac{1}{i+1})$ (Choix A)
Les approximants sont de type Padé.

Exemple 6
$f(t) = \exp(-t)$
$G(x,t) = 1/(1-xt)$
$c_i = (-1)^i/i!$
$u_i(x) = x^i$
$L_i(f) = \int_{-\frac{1}{i+1}}^{-\frac{1}{i+2}} f(x)dx$ (Choix B)
Les approximants sont de type Padé généralisés.

Tableau 1.4: $f(t) = \exp(-t), G(x,t) = \exp(-xt), c_i = 1, u_i(x) = x^i$.

	$t = 1/10$	
n	Exemple 7 Choix A	Exemple 8 Choix B
0	$2.2\ 10^{-1}$	$1.9\ 10^{-1}$
1	$1.7\ 10^{-2}$	$1.4\ 10^{-2}$
2	$7.5\ 10^{-4}$	$6.3\ 10^{-4}$
3	$2.4\ 10^{-5}$	$2.0\ 10^{-5}$
4	$5.6\ 10^{-7}$	$4.8\ 10^{-7}$
5	$1.1\ 10^{-8}$	$1.2\ 10^{-8}$
6	$7.3\ 10^{-8}$	$9.7\ 10^{-8}$
7	$8.7\ 10^{-6}$	$5.2\ 10^{-6}$
8	$7.9\ 10^{-4}$	$2.6\ 10^{-3}$
9	$3.4\ 10^{-2}$	$3.0\ 10^{-1}$
10	$1.8\ 10^{-1}$	$4.0\ 10^{+1}$

Exemple 7
$f(t) = \exp(-t)$
$G(x,t) = \exp(-xt)$
$c_i = 1$
$u_i(x) = x^i$
$L_i(f) = f(-\frac{1}{i+1})$ (Choix A)
Les approximants sont de type Padé généralisés.

Exemple 8
$f(t) = \exp(-t)$
$G(x,t) = \exp(-xt)$
$c_i = 1$
$u_i(x) = x^i$
$L_i(f) = \int_{-\frac{1}{i+1}}^{-\frac{1}{i+2}} f(x)dx$ (Choix B)
Les approximants sont de type Padé généralisés.

Ces résultats ont été obtenus avec le programme GENPADETYPE, donné dans l'annexe A, et le logiciel *MacFortran/020*, dans un *Macintosh II*.

Nous rappelons que les calculs sont faits en double précision, c'est-à-dire avec 16 chiffres. Cependant, les tableaux précédents montrent que l'effet cumulatif des erreurs d'arrondi entache rapidement les résultats. En effet, les erreurs commençent par décroître, mais à partir d'un certain moment elles croissent. Or, nous savons qu'au moins les approximants de l'exemple 1 convergent vers la série.

1.3 Calcul en utilisant les relations de récurrence de la biorthogonalité

Nous commençons par introduire quelques notations.

Soit E un espace vectoriel de dimension infinie.

Soit $\{x_i\}_i$ une suite d'éléments de E, telle que $\forall j, n, \{x_j, \ldots, x_{j+n}\}$ sont linéairement indépendants.

Soit $\{L_i\}_i$ une suite de fonctionnelles linéaires de E^*, l'espace dual de E, telle que $\forall i, j, n$:

$$D_{n+1}^{(i,j)} = \begin{vmatrix} L_i(x_j) & \cdots & L_i(x_{j+n}) \\ \cdots & & \cdots \\ L_{i+n}(x_j) & \cdots & L_{i+n}(x_{j+n}) \end{vmatrix} \neq 0.$$

Soit $f \in E$ et $L \in E^*$. Définissons les déterminants suivants:

$$N_{n+1}^{(i,j)} = \begin{vmatrix} L_i(x_j) & \cdots & L_i(x_{j+n}) \\ \cdots & & \cdots \\ L_{i+n-1}(x_j) & \cdots & L_{i+n-1}(x_{j+n}) \\ x_j & \cdots & x_{j+n} \end{vmatrix}$$

$$\bar{N}_{n+1}^{(i,j)} = \begin{vmatrix} L_i(x_j) & \cdots & L_{i+n}(x_j) \\ \cdots & & \cdots \\ L_i(x_{j+n-1}) & \cdots & L_{i+n}(x_{j+n-1}) \\ L_i & \cdots & L_{i+n} \end{vmatrix}$$

$$N_{n+2}^{(i,j)}(L, f) = \begin{vmatrix} 0 & L(x_j) & \cdots & L(x_{j+n}) \\ L_i(f) & L_i(x_j) & \cdots & L_i(x_{j+n}) \\ \cdots & \cdots & & \cdots \\ L_{i+n}(f) & L_{i+n}(x_j) & \cdots & L_{i+n}(x_{j+n}) \end{vmatrix}.$$

A partir de ces déterminants, définissons:

$$x_n^{(i,j)} = N_{n+1}^{(i,j)} / D_n^{(i,j)} \tag{1.11}$$

$$L_n^{(i,j)} = \bar{N}_{n+1}^{(i,j)} / D_{n+1}^{(i,j)} \tag{1.12}$$

$$K_n^{(i,j)}(L, f) = -N_{n+2}^{(i,j)}(L, f) / D_{n+1}^{(i,j)} \tag{1.13}$$

avec les conditions initialles

$$x_0^{(i,j)} = x_j \tag{1.14}$$

$$L_0^{(i,j)} = L_i / L_i(x_j) \tag{1.15}$$

$$K_{-1}^{(i,j)}(L, f) = 0 \tag{1.16}$$

Si nous comparons les expressions de $(n)_f(t)$ et de $K_n^{(i,j)}(L,f)$ comme rapport de deux déterminants, nous obtenons:

$$(n)_f(t) = K_n^{(0,0)}(c,G)$$

en supposant par exemple, que E est l'espace vectoriel des fonctions complexes de la variable complexe, et que $x_i = u_i(x)$.

En appliquant les identités de Sylvester et de Schweins aux déterminants définis ci-dessus, et en utilisant quelques résultats de la théorie de la biorthogonalité, Brezinski [3] a obtenu plusieurs relations de récurrence entre des éléments adjacents. Nous utiliserons les relations suivantes avec les conditions initialles 1.14, 1.15 et 1.16, pour calculer $(n)_f(t)$:

$$\begin{aligned} K_n^{(i,j)}(L,f) &= K_{n-1}^{(i,j)}(L,f) + L_n^{(i,j)}(f)L(x_n^{(i,j)}) && (K)\\ L_n^{(i,j)}(f) &= \frac{L_{n-1}^{(i+1,j)}(f)-L_{n-1}^{(i,j)}(f)}{L_{n-1}^{(i+1,j)}(x_{j+n})-L_{n-1}^{(i,j)}(x_{j+n})} && (L)\\ L(x_n^{(i,j)}) &= L(x_{n-1}^{(i,j+1)}) - \frac{L_{i+n-1}(x_{n-1}^{(i,j+1)})}{L_{i+n-1}(x_{n-1}^{(i,j)})}L(x_{n-1}^{(i,j)}). && (X) \end{aligned}$$

1.3.1 Implémentation

Nous voulons calculer une suite

$$((k)_f(t))_{k=0,1,\ldots}$$

d'approximants de type Padé généralisés, à partir des relations de récurrence (X), (L) et (K) et des conditions initialles (1.14), (1.15) et (1.16) de la section précédente.

Considérons le problème plus général consistant à calculer la suite

$$(K_k^{(i,j)}(c,G(x,t)))_{k=0,1,\ldots}$$

avec les indices i et j fixés et $x_j = u_j(x)$, puisque les raisonnements à appliquer sont exactement les mêmes.

Ce problème se réduit au problème consistant à calculer les suites

$$(L_k^{(i,j)}(G))_{k=0,1,\ldots} \text{ et } (c(x_k^{(i,j)}))_{k=0,1,\ldots}$$

avec les indices i et j fixés et $x_j = u_j(x)$.

Commençons par traiter la première suite.

Le calcul de

$$(L_k^{(i,j)}(G))_{k=0,1,\ldots,n}$$

à partir de la relation (L), exige le calcul des éléments de la figure 1.1, que nous disposons dans $n+1$ tableaux correspondants aux différentes valeurs de k.

Les éléments du premier tableau sont des conditions initialles de la relation (L), puisque k vaut 0, et sont calculés à partir de l'égalité (1.15). Chaque élément de chacun des tableaux restants est calculé à partir de quatre éléments du tableau précédent, en appliquant la relation (L) (voir figure 1.2).

Figure 1.1: Tableaux L

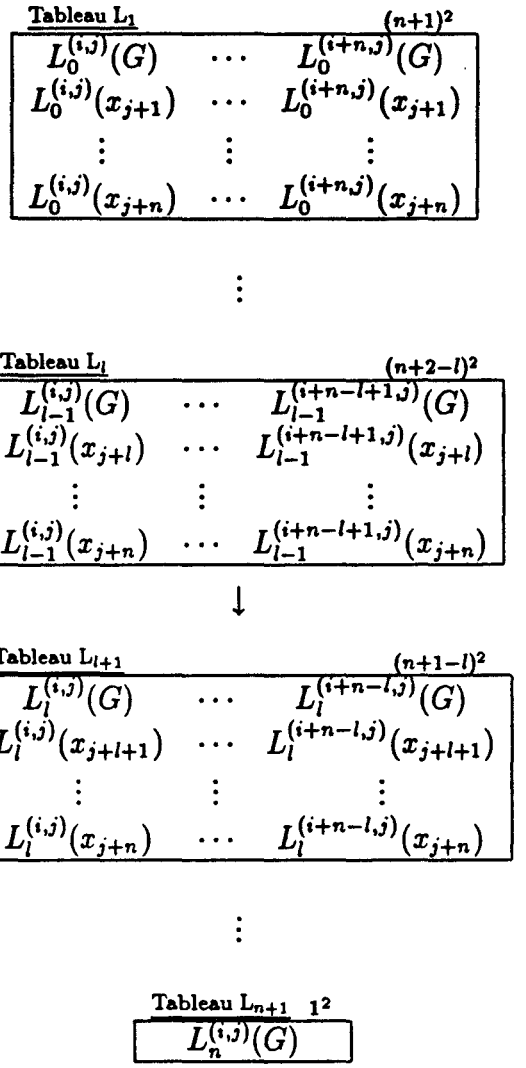


Figure 1.2: Calcul du tableau L_{l+1} à partir du tableau L_l

$$\begin{array}{|cc|} L_{l-1}^{(i+m,j)}(G) & L_{l-1}^{(i+m+1,j)}(G) \\ L_{l-1}^{(i+m,j)}(x_{j+l}) & L_{l-1}^{(i+m+1,j)}(x_{j+l}) \end{array} \rightarrow \boxed{L_l^{(i+m,j)}(G)}$$

$$\begin{array}{|cc|} L_{l-1}^{(i+m,j)}(x_{j+p}) & L_{l-1}^{(i+m+1,j)}(x_{j+p}) \\ L_{l-1}^{(i+m,j)}(x_{j+l}) & L_{l-1}^{(i+m+1,j)}(x_{j+l}) \end{array} \rightarrow \boxed{L_l^{(i+m,j)}(x_{j+p})}$$

$l = 1, \ldots, n$, $m = 0, 1, \ldots, n-l$, $p = l+1, \ldots, n$, i et j fixés.

Nous remarquons que chaque tableau a une ligne et une colonne de moins que le tableau précédent.

Supposons maintenant que l'on veuille poursuivre les calculs, c'est-à-dire qu'après avoir calculé $(L_k^{(i,j)}(G))_{k=0,1,\ldots,n}$, nous voulions calculer $L_{n+1}^{(i,j)}(G)$. Ceci correspond à ajouter une nouvelle ligne et une nouvelle colonne à chacun des $n+1$ tableaux de la figure 1.1, plus le tableau L_{n+2} constitué par l'élément $L_{n+1}^{(i,j)}(G)$. Nous commençons par calculer la nouvelle ligne et la nouvelle colonne du tableau L_1, ce que nous appelons les nouvelles conditions initialles. La nouvelle ligne et la nouvelle colonne de chacun des tableaux restants sont calculées à partir de la deuxième et la dernière lignes et des deux dernières colonnes du tableau précédent, auquel nous avions déjà ajouté les nouveaux éléments. Donc, pour calculer $L_{n+1}^{(i,j)}(G)$ nous n'avons besoin que des éléments situés dans la deuxième ligne et la dernière colonne des tableaux $\mathrm{L}_1,\ldots,\mathrm{L}_{n+1}$.

La méthode la plus simple de programmer la relation (L) est de garder dans la mémoire les éléments $L_k^{(i,j)}(-)$, au fur et à mesure qu'ils sont calculés. Ceci correspond dans le calcul de $L_n^{(i,j)}(G)$ à garder les

$$\sum_{i=1}^{n+1} i^2 = \frac{(n+1)(n+2)(2n+3)}{6} \approx n^3/3$$

éléments de la figure 1.1. Ce calcul peut être fait en *Mathematica* (voir Wolfram [13]) à partir des fonctions suivantes:

$l[i\_, j\_, 0, body\_, var\_] := l[i, j, 0, body, var] =$
$Expand[initialf[i, body, var]/initialf[i, u[j, var], var]]$

```
l[i_,j_,0,body_,var_]:=l[i,j,0,body,var]=
Expand[initialf[i,body,var]/initialf[i,u[j,var],var]]
```

```
l[i_,j_,n_,body_,var_]:=l[i,j,n,body,var]=
Expand[(l[i+1,j,n-1,body,var]-
        l[i,j,n-1,body,var])/
       (l[i+1,j,n-1,u[j+n,var],var]-
        l[i,j,n-1,u[j+n,var],var])]
```

en appelant $l[i, j, n, g[x, t], x]$.

Nous pouvons prendre, par exemple

$$u[j\_, var\_] := u[j, var] = var \wedge j$$

et

$$initialf[i\_, body\_, var\_] := initialf[i, body, var] =$$
$$Function[var, body][-1/(i+1)]$$

comme définitions de u_j et de L_i.

Ce type de définitions *Mathematica* garde automatiquement dans la mémoire les éléments qu'elle calcule, ce que évite de devoir recalculer des éléments intermédiaires (voir Wolfram

Figure 1.3: Tableau L de la méthode 1

Tableau L

$(n+1)^2$

$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_1^{(i+n-1,j)}(G)$	$L_0^{(i+n,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n-1,j)}(x_{j+1})$	$L_0^{(i+n,j)}(x_{j+1})$
$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$\cdots$	$L_1^{(i+n-1,j)}(x_{j+2})$	$L_0^{(i+n,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{n-2}^{(i,j)}(x_{j+n-1})$	$L_{n-2}^{(i+1,j)}(x_{j+n-1})$	$\cdots$	$L_1^{(i+n-1,j)}(x_{j+n-1})$	$L_0^{(i+n,j)}(x_{j+n-1})$
$L_{n-1}^{(i,j)}(x_{j+n})$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_1^{(i+n-1,j)}(x_{j+n})$	$L_0^{(i+n,j)}(x_{j+n})$

[13, pages 202–203]). Ainsi, quand nous appelons $l[i,j,n,g[x,t],x]$ tous les éléments de la figure 1.1 sont calculés et gardés dans la mémoire. Si après, nous appelons $l[i,j,n+1,g[x,t],x]$ ces mêmes éléments-là demeurent disponibles et ne sont pas recalculés, alors que la ligne et colonne d'éléments à ajouter à chaque tableau, plus le nouveau tableau vont être calculés et gardés dans la mémoire.

1.3.1.1 Economie de l'espace mémoire

Supposons maintenant que l'on désire réduire l'encombrement mémoire.

Dans l'application de la relation (L) (voir figure 1.2), lorsque l'élément $L_l^{(i+m,j)}(G)$ $(L_l^{(i+m,j)}(x_{j+p}))$ est calculé, l'élément $L_{l-1}^{(i+m,j)}(G)$ $(L_{l-1}^{(i+m,j)}(x_{j+p}))$ n'est plus nécessaire dans la suite des calculs. Donc, $L_l^{(i+m,j)}(G)$ $(L_l^{(i+m,j)}(x_{j+p}))$ peut, après avoir été calculé, être gardé dans la place de $L_{l-1}^{(i+m,j)}(G)$ $(L_{l-1}^{(i+m,j)}(x_{j+p}))$.

En appliquant ce procédé, c'est-à-dire en faisant ce changement de place après le calcul de chaque élément, nous pouvons calculer $(L_k^{(i,j)}(G))_{k=0,1,\ldots,n}$, en ne gardant que

$$(n+1)^2 \approx n^2$$

éléments, plus un élément temporaire.

Nous considérons deux méthodes distinctes qui utilisent cette technique de remplacement.

Méthode 1: Nous commençons par calculer les $(n+1)^2$ conditions initialles du tableau L_1. Ensuite, chaque élément du tableau L_2, après avoir été calculé, remplace un élément du tableau L_1 et ainsi de suite pour les autres tableaux. En réalité, nous utilisons un seul et unique tableau de dimension fixe $n+1$, que nous appelerons tableau L, où vont être superposés les tableaux $L_1,\ldots,L_{n+1}$. A la fin du calcul, le tableau L contient les éléments de la figure 1.3.

Malheureusement cette méthode simple souffre d'un inconvénient majeur: si après avoir calculé $L_n^{(i,j)}(G)$, nous voulons calculer $L_{n+1}^{(i,j)}(G)$ nous devons recommencer tous les calculs dès le début, en considérant un tableau L de dimension $n+2$.

Méthode 2: Au lieu de partir d'un tableau de taille fixée à priori, nous considérons un seul tableau L, qui a initialement comme dimension 1×1, et qui augmente d'une ligne et d'une colonne à chaque étape.

Cette méthode est constituée par deux procédés itératifs emboîtés. A la fin de l'itération d'ordre l, $(l = 1, 2, \ldots)$, nous aboutissons au calcul de l'élément $L_{l-1}^{(i,j)}(G)$. Dans la sous-itération d'ordre m de l'itération l, $(m = 1, 2, \ldots, l)$, nous calculons certains éléments du tableau L_m, qui vont être nécessaires au calcul de $L_{l-1}^{(i,j)}(G)$.

Voyons en détail les trois premières itérations, pour pouvoir mieux comprendre la procédure employée.

Dans les figures qui suivent les tableaux reliés par des flèches sont destinés uniquement à aider le lecteur à mieux suivre les explications relatives à chaque itération. Dans la pratique, nous travaillons uniquement avec le tableau L.

Itération 1

Sous-itération 1

Calcul de la condition initialle $L_0^{(i,j)}(G)$, qui initie le tableau L.

Tableau L (sous-iter 1)

$L_0^{(i,j)}(G)$

Itération 2

Nous voulons calculer $L_1^{(i,j)}(G)$. Or, selon la relation (L), $L_1^{(i,j)}(G)$ est calculé à partir des éléments $L_0^{(i,j)}(G)$, $L_0^{(i+1,j)}(G)$, $L_0^{(i,j)}(x_{j+1})$ et $L_0^{(i+1,j)}(x_{j+1})$, dont le premier à déjà été calculé et gardé dans l'itération 1.

$L_0^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$

$\longmapsto$ $\boxed{L_1^{(i,j)}(G)}$

Sous-itération 1

Les trois nouvelles conditions initialles sont calculées et placées dans le tableau L, occupant une nouvelle ligne et une nouvelle colonne du tableau.

Tableau L (sous-iter 1)

$L_0^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$

Sous-itération 2

$L_1^{(i,j)}(G)$ est calculé à partir des quatre éléments du tableau L, ensuite il est gardé dans la place de l'élément $L_0^{(i,j)}(G)$.

Tableau L (sous-iter 2)

$L_1^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$

Itération 3

Nous voulons calculer $L_2^{(i,j)}(G)$. Or, selon la relation (L), $L_2^{(i,j)}(G)$ est calculé à partir des éléments $L_1^{(i,j)}(G), L_1^{(i+1,j)}(G), L_1^{(i,j)}(x_{j+2})$ et $L_1^{(i+1,j)}(x_{j+2})$, dont le premier à déjà été calculé et gardé dans l'itération 2. Il reste à calculer les autres.
$L_1^{(i+1,j)}(G)$ est calculé à partir des éléments $L_0^{(i+1,j)}(G)$, $L_0^{(i+1,j)}(x_{j+1})$, $L_0^{(i+2,j)}(G)$ et $L_0^{(i+2,j)}(x_{j+1})$, dont les deux premiers ont déjà été calculés et gardés dans l'itération 2.
$L_1^{(i,j)}(x_{j+2})$ est calculé à partir des éléments $L_0^{(i,j)}(x_{j+1}), L_0^{(i+1,j)}(x_{j+1}), L_0^{(i,j)}(x_{j+2})$ et $L_0^{(i+1,j)}(x_{j+2})$, dont les deux premiers ont déjà été calculés et gardés dans l'itération 2.
Et finalement $L_1^{(i+1,j)}(x_{j+2})$ est calculé à partir des éléments $L_0^{(i+1,j)}(x_{j+1})$, $L_0^{(i+2,j)}(x_{j+1})$, $L_0^{(i+1,j)}(x_{j+2})$ et $L_0^{(i+2,j)}(x_{j+2})$, dont le premier à déjà été calculé et gardé dans l'itération 2.

$\times$	$L_0^{(i+1,j)}(G)$	$L_0^{(i+2,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$L_0^{(i+2,j)}(x_{j+1})$
$L_0^{(i,j)}(x_{j+2})$	$L_0^{(i+1,j)}(x_{j+2})$	$L_0^{(i+2,j)}(x_{j+2})$

$\downarrow$

$L_1^{(i,j)}(G)$	$L_1^{(i+1,j)}(G)$
$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$

$\downarrow$

$L_2^{(i,j)}(G)$

Remarque : Dans le premier tableau une croix a été mise à la place de l'élément $L_0^{(i,j)}(G)$, qui n'est plus nécéssaire.

Sous-itération 1

Les cinq nouvelles conditions initialles sont calculées et placées dans le tableau L, occupant une nouvelle ligne et une nouvelle colonne du tableau.

Tableau L (sous-iter 1)

$L_1^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$	$L_0^{(i+2,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$L_0^{(i+2,j)}(x_{j+1})$
$L_0^{(i,j)}(x_{j+2})$	$L_0^{(i+1,j)}(x_{j+2})$	$L_0^{(i+2,j)}(x_{j+2})$

Sous-itération 2

$L_1^{(i,j)}(x_{j+2}), L_1^{(i+1,j)}(G)$ et $L_1^{(i+1,j)}(x_{j+2})$ sont calculés à partir des éléments déjà cités et

qui se trouvent tous dans le tableau L. Chacun des éléments $L_1^{(i,j)}(x_{j+2}), L_1^{(i+1,j)}(G)$ et $L_1^{(i+1,j)}(x_{j+2})$ après avoir été calculé, est gardé dans la place de $L_0^{(i,j)}(x_{j+2}), L_0^{(i+1,j)}(G)$ et $L_0^{(i+1,j)}(x_{j+2})$ respectivement.

Tableau L (sous-iter 2)

$L_1^{(i,j)}(G)$	$L_1^{(i+1,j)}(G)$	$L_0^{(i+2,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$L_0^{(i+2,j)}(x_{j+1})$
$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$L_0^{(i+2,j)}(x_{j+2})$

Sous-itération 3

$L_2^{(i,j)}(G)$ est calculé à partir des quatre éléments déjà cités et qui se trouvent tous dans le tableau L, ensuite il est gardé dans la place de $L_1^{(i,j)}(G)$.

Tableau L (sous-iter 3)

$L_2^{(i,j)}(G)$	$L_1^{(i+1,j)}(G)$	$L_0^{(i+2,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$L_0^{(i+2,j)}(x_{j+1})$
$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$L_0^{(i+2,j)}(x_{j+2})$

$\vdots$

Si nous continuons jusqu'à l'itération $n+1$, nous aboutirons au calcul de $L_n^{(i,j)}(G)$, et le tableau L, qui en ce moment aura comme dimension $(n+1)^2$, contiendra les mêmes éléments que le tableau L de la méthode 1 à la fin de la même itération (voir figure 1.3). Mais, avec la méthode 2, il est possible de poursuivre les calculs sans revenir en arrière. Il suffit d'ajouter au tableau L les conditions initialles correspondantes à l'itération $n+2$, et d'effectuer les $n+1$ sous-itérations restantes comme nous venons de l'expliquer. Voyons en termes plus précis le déroulement du calcul de $L_{n+1}^{(i,j)}(G)$.

Itération $n+2$

Sous-itération 1

Calcul des nouvelles conditions initialles qui vont occuper une nouvelle ligne et une nouvelle colonne du tableau L.

Tableau L (sous-iter 1) $(n+2)^2$

$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_0^{(i+n,j)}(G)$	$L_0^{(i+n+1,j)}(G)$
$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n,j)}(x_{j+1})$	$L_0^{(i+n+1,j)}(x_{j+1})$
$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$\cdots$	$L_0^{(i+n,j)}(x_{j+2})$	$L_0^{(i+n+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{n-2}^{(i,j)}(x_{j+n-1})$	$L_{n-2}^{(i+1,j)}(x_{j+n-1})$	$\cdots$	$L_0^{(i+n,j)}(x_{j+n-1})$	$L_0^{(i+n+1,j)}(x_{j+n-1})$
$L_{n-1}^{(i,j)}(x_{j+n})$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_0^{(i+n,j)}(x_{j+n})$	$L_0^{(i+n+1,j)}(x_{j+n})$
$L_0^{(i,j)}(x_{j+n+1})$	$L_0^{(i+1,j)}(x_{j+n+1})$	$\cdots$	$L_0^{(i+n,j)}(x_{j+n+1})$	$L_0^{(i+n+1,j)}(x_{j+n+1})$

$\vdots$

Sous-itération $m+1$ $(m=1,\cdots,n+1)$

A la fin de la sous-itération m, le tableau L contient les éléments de la figure 1.4.

Dans la sous-itération $m+1$ nous faisons les calculs suivants, effectués avec les éléments du tableau L.

Avec les éléments qui occupent les lignes d'ordres $m+1$ et $n+2$, de la colonne 1 jusqu'à la colonne $n-m+2$, nous calculons les éléments $L_m^{(i,j)}(x_{j+n+1})$, $L_m^{(i+1,j)}(x_{j+n+1}),\ldots,$ $L_m^{(i+n-m,j)}(x_{j+n+1})$, qui vont être gardés à la place des éléments $L_{m-1}^{(i,j)}(x_{j+n+1})$, $L_{m-1}^{(i+1,j)}(x_{j+n+1}),\ldots,$ $L_{m-1}^{(i+n-m,j)}(x_{j+n+1})$, respectivement.

Avec les éléments qui occupent les colonnes d'ordres $n-m+2$ et $n-m+3$ de la ligne 1 et des lignes $m+1$ jusqu'à $n+2$, nous calculons les éléments $L_m^{(i+n-m+1,j)}(G)$, $L_m^{(i+n-m+1,j)}(x_{j+m+1}),\ldots,$ $L_m^{(i+n-m+1,j)}(x_{j+n+1})$, qui vont être gardés dans la place des éléments $L_{m-1}^{(i+n-m+1,j)}(G)$, $L_{m-1}^{(i+n-m+1,j)}(x_{j+m+1}),\ldots,L_{m-1}^{(i+n-m+1,j)}(x_{j+n+1})$, respectivement.

Après ceci le tableau L contient les éléments de la figure 1.5.

Figure 1.4: Tableau L (itération $n+2-$ sous-itération m)

	1	2	$\cdots$	$n-m+2$	$n-m+3$
1	$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_{m-1}^{(i+n-m+1,j)}(G)$	$L_{m-1}^{(i+n-m+2,j)}(G)$
2	$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n-m+1,j)}(x_{j+1})$	$L_0^{(i+n-m+2,j)}(x_{j+1})$
3	$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$\cdots$	$L_1^{(i+n-m+1,j)}(x_{j+2})$	$L_1^{(i+n-m+2,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m+1$	$L_{m-1}^{(i,j)}(x_{j+m})$	$L_{m-1}^{(i+1,j)}(x_{j+m})$	$\cdots$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+m})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m})$
$m+2$	$L_m^{(i,j)}(x_{j+m+1})$	$L_m^{(i+1,j)}(x_{j+m+1})$	$\cdots$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+m+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n+1$	$L_{n-1}^{(i,j)}(x_{j+n})$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+n})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n})$
$n+2$	$L_{m-1}^{(i,j)}(x_{j+n+1})$	$L_{m-1}^{(i+1,j)}(x_{j+n+1})$	$\cdots$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+n+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n+1})$

$n-m+4$	$\cdots$	n	$n+1$	$n+2$
$L_{m-2}^{(i+n-m+3,j)}(G)$	$\cdots$	$L_2^{(i+n-1,j)}(G)$	$L_1^{(i+n,j)}(G)$	$L_0^{(i+n+1,j)}(G)$
$L_0^{(i+n-m+3,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n-1,j)}(x_{j+1})$	$L_0^{(i+n,j)}(x_{j+1})$	$L_0^{(i+n+1,j)}(x_{j+1})$
$L_1^{(i+n-m+3,j)}(x_{j+2})$	$\cdots$	$L_1^{(i+n-1,j)}(x_{j+2})$	$L_1^{(i+n,j)}(x_{j+2})$	$L_0^{(i+n+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{m-2}^{(i+n-m+3,j)}(x_{j+m})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+m})$	$L_1^{(i+n,j)}(x_{j+m})$	$L_0^{(i+n+1,j)}(x_{j+m})$
$L_{m-2}^{(i+n-m+3,j)}(x_{j+m+1})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+m+1})$	$L_1^{(i+n,j)}(x_{j+m+1})$	$L_0^{(i+n+1,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{m-2}^{(i+n-m+3,j)}(x_{j+n})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+n})$	$L_1^{(i+n,j)}(x_{j+n})$	$L_0^{(i+n+1,j)}(x_{j+n})$
$L_{m-2}^{(i+n-m+3,j)}(x_{j+n+1})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+n+1})$	$L_1^{(i+n,j)}(x_{j+n+1})$	$L_0^{(i+n+1,j)}(x_{j+n+1})$

Figure 1.5: Tableau L (itération $n+2$ – sous-itération $m+1$)

	1	2	$\cdots$	$n-m+1$	$n-m+2$
1	$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_m^{(i+n-m,j)}(G)$	$L_{m-1}^{(i+n-m+1,j)}(G)$
2	$L_0^{(i,j)}(x_{j+1})$	$L_0^{(i+1,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n-m,j)}(x_{j+1})$	$L_0^{(i+n-m+1,j)}(x_{j+1})$
3	$L_1^{(i,j)}(x_{j+2})$	$L_1^{(i+1,j)}(x_{j+2})$	$\cdots$	$L_1^{(i+n-m,j)}(x_{j+2})$	$L_0^{(i+n-m+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m+1$	$L_{m-1}^{(i,j)}(x_{j+m})$	$L_{m-1}^{(i+1,j)}(x_{j+m})$	$\cdots$	$L_{m-1}^{(i+n-m,j)}(x_{j+m})$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+m})$
$m+2$	$L_m^{(i,j)}(x_{j+m+1})$	$L_m^{(i+1,j)}(x_{j+m+1})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+m+1})$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n+1$	$L_{n-1}^{(i,j)}(x_{j+n})$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+n})$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+n})$
$n+2$	$L_m^{(i,j)}(x_{j+n+1})$	$L_m^{(i+1,j)}(x_{j+n+1})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+n+1})$	$L_{m-1}^{(i+n-m+1,j)}(x_{j+n+1})$

$n-m+3$	$\cdots$	n	$n+1$	$n+2$
$L_{m-2}^{(i+n-m+2,j)}(G)$	$\cdots$	$L_2^{(i+n-1,j)}(G)$	$L_1^{(i+n,j)}(G)$	$L_0^{(i+n+1,j)}(G)$
$L_0^{(i+n-m+2,j)}(x_{j+1})$	$\cdots$	$L_0^{(i+n-1,j)}(x_{j+1})$	$L_0^{(i+n,j)}(x_{j+1})$	$L_0^{(i+n+1,j)}(x_{j+1})$
$L_1^{i+n-m+2,j)}(x_{j+2})$	$\cdots$	$L_1^{(i+n-1,j)}(x_{j+2})$	$L_1^{(i+n,j)}(x_{j+2})$	$L_0^{(i+n+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{m-2}^{(i+n-m+2,j)}(x_{j+m})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+m})$	$L_1^{(i+n,j)}(x_{j+m})$	$L_0^{(i+n+1,j)}(x_{j+m})$
$L_{m-2}^{(i+n-m+2,j)}(x_{j+m+1})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+m+1})$	$L_1^{(i+n,j)}(x_{j+m+1})$	$L_0^{(i+n+1,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{m-2}^{(i+n-m+2,j)}(x_{j+n})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+n})$	$L_1^{(i+n,j)}(x_{j+n})$	$L_0^{(i+n+1,j)}(x_{j+n})$
$L_{m-2}^{(i+n-m+2,j)}(x_{j+n+1})$	$\cdots$	$L_2^{(i+n-1,j)}(x_{j+n+1})$	$L_1^{(i+n,j)}(x_{j+n+1})$	$L_0^{(i+n+1,j)}(x_{j+n+1})$

Le calcul de la suite

$$(c(x_k^{(i,j)}))_{k=0,1,\ldots}$$

est fait en appliquant les mêmes techniques employées dans le calcul de la suite $(L_k^{(i,j)}(G))_{k=0,1,\ldots}$, bien que les éléments des tableaux et la relation de récurrence à appliquer ne soient pas les mêmes dans les deux cas.

Le calcul de

$$(c(x_k^{(i,j)}))_{k=0,1,\ldots,n}$$

à partir de la relation (X) exige le calcul des éléments de la figure 1.6, que nous disposons dans $n+1$ tableaux correspondants aux différentes valeurs de k.

Voyons comment ces tableaux sont calculés.

Nous commençons par le calcul, à partir de l'égalité (1.14), des conditions initialles de la relation (X) qui sont présentes dans le premier tableau. C'est-à-dire, nous faisons:

$$x_0^{(i,j+m)} = x_{j+m} \quad , m = 0, 1, \ldots, n.$$

Après cela, les lignes du tableau X_1 s'obtiennent en appliquant à ces éléments les fonctionnelles $c, L_i, \ldots, L_{i+n-1}$, respectivement.

Le calcul de chaque élément de chacun des tableaux restants est fait à partir de quatre éléments du tableau précédent en utilisant la relation (X) (voir figure 1.7).

Avec les figures 1.6 et 1.7 il ne sera pas difficile au lecteur de reprendre ce qui a été écrit pour la suite $(L_k^{(i,j)}(G))_{k=0,1,\ldots}$ et de l'appliquer à la suite $(c(x_k^{(i,j)}))_{k=0,1,\ldots}$. Il suffit de remplacer dans le texte et dans les tableaux les éléments d'une suite par les éléments correspondants de l'autre suite. Tous les problèmes posés et les raisonnements à appliquer demeurent valables.

1.3.1.2 Economie du nombre de calculs

Après avoir résolu le problème d'encombrement de mémoire posé par le calcul de ces suites, il est conseillé d'économiser le nombre de calculs à effectuer.

Remarquons que c'est le calcul des tableaux X_1 et L_1 qui risque de coûter plus cher parce qu'il s'agit d'évaluer des fonctionnelles qui peuvent avoir une expression trés compliquée. Les tableaux restants sont obtenus en appliquant les relations (L) ou (X); on n'évalue plus de fonctionnelles les calculs sont réduits aux quatre opérations arithmétiques.

Reprenons ici les tableaux X_1 et L_1 des figures 1.1 et 1.6, après avoir remplacé les conditions initialles des relations (X) et (L), par leurs expressions données par les égalités (1.14) et (1.15).

Tableau X_1 $(n+1)^2$

$c(x_j)$	$c(x_{j+1})$	$c(x_{j+2})$	$\cdots$	$c(x_{j+n})$
$L_i(x_j)$	$L_i(x_{j+1})$	$L_i(x_{j+2})$	$\cdots$	$L_i(x_{j+n})$
$L_{i+1}(x_j)$	$L_{i+1}(x_{j+1})$	$L_{i+1}(x_{j+2})$	$\cdots$	$L_{i+1}(x_{j+n})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{i+n-1}(x_j)$	$L_{i+n-1}(x_{j+1})$	$L_{i+n-1}(x_{j+2})$	$\cdots$	$L_{i+n-1}(x_{j+n})$

Figure 1.6: Tableaux X

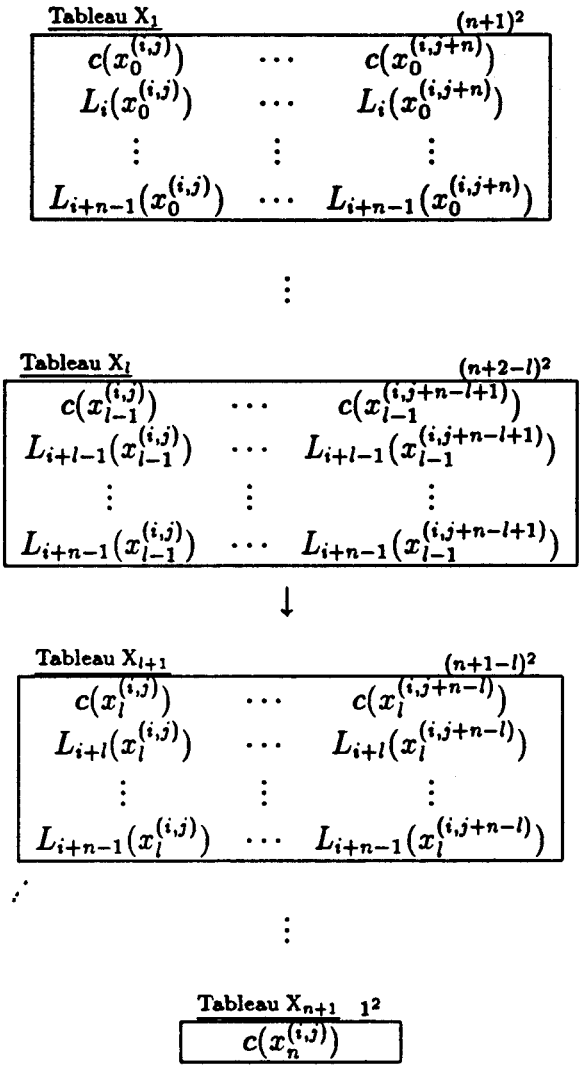
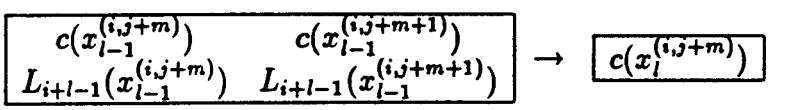
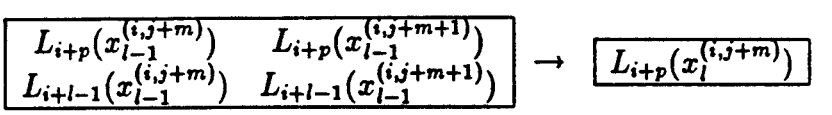


Figure 1.7: Calcul du tableau X_{l+1} à partir du tableau X_l





$l = 1, \ldots, n$, $m = 0, 1, \ldots, n-l$, $p = l, \ldots, n-1$, i et j fixés.

Tableau L_1 ($(n+1)^2$)

$\frac{L_i(G)}{L_i(x_j)}$	$\frac{L_{i+1}(G)}{L_{i+1}(x_j)}$	$\frac{L_{i+2}(G)}{L_{i+2}(x_j)}$	...	$\frac{L_{i+n}(G)}{L_{i+n}(x_j)}$
$\frac{L_i(x_{j+1})}{L_i(x_j)}$	$\frac{L_{i+1}(x_{j+1})}{L_{i+1}(x_j)}$	$\frac{L_{i+2}(x_{j+1})}{L_{i+2}(x_j)}$	...	$\frac{L_{i+n}(x_{j+1})}{L_{i+n}(x_j)}$
$\frac{L_i(x_{j+2})}{L_i(x_j)}$	$\frac{L_{i+1}(x_{j+2})}{L_{i+1}(x_j)}$	$\frac{L_{i+2}(x_{j+2})}{L_{i+2}(x_j)}$	...	$\frac{L_{i+n}(x_{j+2})}{L_{i+n}(x_j)}$
⋮	⋮	⋮	⋮	⋮
$\frac{L_i(x_{j+n})}{L_i(x_j)}$	$\frac{L_{i+1}(x_{j+n})}{L_{i+1}(x_j)}$	$\frac{L_{i+2}(x_{j+n})}{L_{i+2}(x_j)}$	...	$\frac{L_{i+n}(x_{j+n})}{L_{i+n}(x_j)}$

Dans chaque étape de la méthode 2, nous calculons une ligne et une colonne de chacun de ces tableaux. Faisons spécialement attention à la façon dont ces éléments sont calculés parce que, d'une part, nous voulons garder un minimum de calculs intermédiaires et, d'autre part, nous ne devons jamais répéter le calcul d'un élément x_{j+p}, ou de $c(x_{j+p})$, ou de $L_{i+m}(G)$, ou de $L_{i+m}(x_{j+p})$.

Reprenons les tableaux X et L de la méthode 2, où nous n'allons représenter que les éléments correspondants à la sous-itération 1 de chaque itération , les autres éléments étant représentés par une croix.

Commençons par suivre les deux premières itérations pour que nous puissions mieux comprendre la procédure employée.

Itération 1

Sous-itération 1

x_j est calculé et gardé dans la mémoire. $c(x_j)$ est calculé et placé dans le tableau X.

Tableau X

$c(x_j)$

$L_i(x_j)$ est calculé, gardé dans la mémoire et placé dans le tableau X. Nous commençons déjà à construire les conditions initialles de l'itération 2, nous verrons plus tard la raison de cela.

Tableau X

$c(x_j)$
$L_i(x_j)$

$L_i(G)/L_i(x_j)$ est calculé et placé dans le tableau L.

Tableau L

$\frac{L_i(G)}{L_i(x_j)}$

Itération 2

Sous-itération 1

x_{j+1} est calculé et gardé dans la mémoire. $c(x_{j+1})$ est calculé et placé dans le tableau X.

Tableau X

×	$c(x_{j+1})$
$L_i(x_j)$	

$L_i(x_{j+1})$ est calculé, gardé temporairement dans la mémoire et placé dans le tableau X. Les conditions initialles correspondantes à l'itération 2 sont complètes.

Tableau X

×	$c(x_{j+1})$
$L_i(x_j)$	$L_i(x_{j+1})$

$L_i(x_{j+1})/L_i(x_j)$ est calculé et placé dans le tableau L.

Tableau L

×
$\frac{L_i(x_{j+1})}{L_i(x_j)}$

$L_{i+1}(x_j)$ est calculé et gardé dans la mémoire. $L_{i+1}(G)/L_{i+1}(x_j)$ est calculé et placé dans le tableau L.

Tableau L

×	$\frac{L_{i+1}(G)}{L_{i+1}(x_j)}$
$\frac{L_i(x_{j+1})}{L_i(x_j)}$	

$L_{i+1}(x_j)$ est placé dans le tableau X. Nous commençons déjà à construire les conditions initialles de l'itération 3.

Tableau X

×	$c(x_{j+1})$
$L_i(x_j)$	$L_i(x_{j+1})$
$L_{i+1}(x_j)$	

$L_{i+1}(x_{j+1})$ est calculé et gardé temporairement dans la mémoire. $L_{i+1}(x_{j+1})/L_{i+1}(x_j)$ est calculé et placé dans le tableau L.

Tableau L

×	$\frac{L_{i+1}(G)}{L_{i+1}(x_j)}$
$\frac{L_i(x_{j+1})}{L_i(x_j)}$	$\frac{L_{i+1}(x_{j+1})}{L_{i+1}(x_j)}$

$L_{i+1}(x_{j+1})$ est placé dans le tableau X.

Tableau X

×	$c(x_{j+1})$
$L_i(x_j)$	$L_i(x_{j+1})$
$L_{i+1}(x_j)$	$L_{i+1}(x_{j+1})$

$\vdots$

Supposons que nous ayons terminé l'itération $n+1$. Les tableaux X et L ont l'aspect suivant:

Tableau X $(n+2)\times(n+1)$

×	$\cdots$	×	$c(x_{j+n})$
×	$\cdots$	×	$L_i(x_{j+n})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
×	$\cdots$	×	$L_{i+n-1}(x_{j+n})$
$L_{i+n}(x_j)$	$\cdots$	$L_{i+n}(x_{j+n-1})$	$L_{i+n}(x_{j+n})$

Tableau L $(n+1)^2$

×	$\cdots$	×	$\frac{L_{i+n}(G)}{L_{i+n}(x_j)}$
×	$\cdots$	×	$\frac{L_{i+n}(x_{j+1})}{L_{i+n}(x_j)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
×	$\cdots$	×	$\frac{L_{i+n}(x_{j+n})}{L_{i+n}(x_j)}$

En ce moment, nous avons gardé dans la mémoire les éléments $x_j, x_{j+1}, \ldots, x_{j+n}$, $L_i(x_j), L_{i+1}(x_j), \ldots, L_{i+n}(x_j)$, qui vont être tous nécessaires dans les itérations suivantes.

Itération $n+2$

Sous-itération 1

1ère Partie: Une nouvelle colonne du tableau X et une nouvelle ligne du tableau L sont calculées simultanément.

- x_{j+n+1} est calculé et gardé dans la mémoire.
- $c(x_{j+n+1})$ est calculé et placé dans le tableau X.
- Pour $m = 0$ jusqu'à n
  - $L_{i+m}(x_{j+n+1})$ est calculé, gardé temporairement dans la mémoire, et placé dans le tableau X.
  - $L_{i+m}(x_{j+n+1})/L_{i+m}(x_j)$ est calculé et placé dans le tableau L.

En ce moment les tableaux X et L ont comme dimension $(n+2)^2$ et $(n+2)\times(n+1)$ respectivement. Nous remarquons que, pour les besoins de la méthode 2, le tableau X est déjà complet.

2ème Partie : Une nouvelle colonne du tableau L et une nouvelle ligne du tableau X sont calculées simultanément.

- $L_{i+n+1}(x_j)$ est calculé et gardé dans la mémoire.
- $L_{i+n+1}(G)/L_{i+n+1}(x_j)$ est calculé et placé dans le tableau L.
- $L_{i+n+1}(x_j)$ est placé dans le tableau X.
- Pour $m = 1$ jusqu'à $n+1$
  - $L_{i+n+1}(x_{j+m})$ est calculé et gardé temporairement dans la mémoire.
  - $L_{i+n+1}(x_{j+m})/L_{i+n+1}(x_j)$ est calculé et placé dans le tableau L.
  - $L_{i+n+1}(x_{j+m})$ est placé dans le tableau X.

Maintenant nous comprenons la raison pour laquelle nous avons calculé une ligne du tableau X qui correspond à l'itération suivante. Si nous n'avions pas gardé ces éléments dans le tableau X, nous aurions été obligés soit de les retenir temporairement dans la mémoire jusqu'à la prochaine itération, soit de les recalculer. Comme cette dernière hypothèse est hors de question, il vaut mieux garder ces éléments, tout de suite après leur calcul, directement dans le tableau X.
Montrons finalement les tableaux X et L après la sous-itération 1 de l'itération $n+2$

Tableau X $(n+3)\times(n+2)$

$\times$	$\cdots$	$\times$	$c(x_{j+n})$	$c(x_{j+n+1})$
$\times$	$\cdots$	$\times$	$L_i(x_{j+n})$	$L_i(x_{j+n+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\times$	$\cdots$	$\times$	$L_{i+n-1}(x_{j+n})$	$L_{i+n-1}(x_{j+n+1})$
$L_{i+n}(x_j)$	$\cdots$	$L_{i+n}(x_{j+n-1})$	$L_{i+n}(x_{j+n})$	$L_{i+n}(x_{j+n+1})$
$L_{i+n+1}(x_j)$	$\cdots$	$L_{i+n+1}(x_{j+n-1})$	$L_{i+n+1}(x_{j+n})$	$L_{i+n+1}(x_{j+n+1})$

Tableau L $(n+2)^2$

$\times$	$\cdots$	$\times$	$\frac{L_{i+n}(G)}{L_{i+n}(x_j)}$	$\frac{L_{i+n+1}(G)}{L_{i+n+1}(x_j)}$
$\times$	$\cdots$	$\times$	$\frac{L_{i+n}(x_{j+1})}{L_{i+n}(x_j)}$	$\frac{L_{i+n+1}(x_{j+1})}{L_{i+n+1}(x_j)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\times$	$\cdots$	$\times$	$\frac{L_{i+n}(x_{j+n})}{L_{i+n}(x_j)}$	$\frac{L_{i+n+1}(x_{j+n})}{L_{i+n+1}(x_j)}$
$\frac{L_i(x_{j+n+1})}{L_i(x_j)}$	$\cdots$	$\frac{L_{i+n-1}(x_{j+n+1})}{L_{i+n-1}(x_j)}$	$\frac{L_{i+n}(x_{j+n+1})}{L_{i+n}(x_j)}$	$\frac{L_{i+n+1}(x_{j+n+1})}{L_{i+n+1}(x_j)}$

Essayons encore d'économiser des calculs.

Revenons aux tableaux $L_1,\ldots,L_{n+1}$ de la figure 1.1, et remplaçons, dans la figure 1.2, les schémas par les relations correspondantes:

Figure 1.8: Calcul du tableau L_{l+1} à partir du tableau L_l

$$L_l^{(i+m,j)}(G) = \frac{L_{l-1}^{(i+m+1,j)}(G) - L_{l-1}^{(i+m,j)}(G)}{L_{l-1}^{(i+m+1,j)}(x_{j+l}) - L_{l-1}^{(i+m,j)}(x_{j+l})}$$

$$L_l^{(i+m,j)}(x_{j+p}) = \frac{L_{l-1}^{(i+m+1,j)}(x_{j+p}) - L_{l-1}^{(i+m,j)}(x_{j+p})}{L_{l-1}^{(i+m+1,j)}(x_{j+l}) - L_{l-1}^{(i+m,j)}(x_{j+l})}$$

$l = 1,\ldots,n$, $m = 0,\ldots,n-l$, $p = l+1,\ldots,n$, i et j fixés.

Nous devons faire en sorte que les différences

$$(L_{l-1}^{(i+m+1,j)}(x_{j+l}) - L_{l-1}^{(i+m,j)}(x_{j+l}))$$

ne soient calculées qu'une seule fois.

Si nous remplaçons dans le tableau L_l ($l = 1,\ldots,n$ fixé) la deuxième ligne par la ligne suivante:

$(L_{l-1}^{(i+1,j)}(x_{j+l})-$ $L_{l-1}^{(i,j)}(x_{j+l}))$	$\ldots$	$(L_{l-1}^{(i+n-l+1,j)}(x_{j+l})-$ $L_{l-1}^{(i+n-l,j)}(x_{j+l}))$	$L_{l-1}^{(i+n-l+1,j)}(x_{j+l})$

le calcul de chaque élément du tableau L_{l+1} se fera à partir, cette fois-ci, de trois éléments du tableau L_l, comme l'indiquent les schémas:

$L_{l-1}^{(i+m,j)}(G)$	$L_{l-1}^{(i+m+1,j)}(G)$
$(L_{l-1}^{(i+m+1,j)}(x_{j+l})-$ $L_{l-1}^{(i+m,j)}(x_{j+l}))$	

$\rightarrow$ $\boxed{L_l^{(i+m,j)}(G)}$

$L_{l-1}^{(i+m,j)}(x_{j+p})$	$L_{l-1}^{(i+m+1,j)}(x_{j+p})$
$(L_{l-1}^{(i+m+1,j)}(x_{j+l})-$ $L_{l-1}^{(i+m,j)}(x_{j+l}))$	

$\rightarrow$ $\boxed{L_l^{(i+m,j)}(x_{j+p})}$

Ensuite, la deuxième ligne du tableau L_{l+1} est remplacée par la ligne suivante:

$(L_l^{(i+1,j)}(x_{j+l})-$ $L_l^{(i,j)}(x_{j+l}))$	$\ldots$	$(L_l^{(i+n-l+1,j)}(x_{j+l})-$ $L_l^{(i+n-l,j)}(x_{j+l}))$	$L_l^{(i+n-l+1,j)}(x_{j+l})$

Voyons comment ceci se traduit dans la méthode 2.

Itération 1

Sous-itération 1

La première itération reste inchangée.

Tableau L (sous-iter 1)

$L_0^{(i,j)}(G)$

Itération 2

Sous-itération 1

$L_0^{(i,j)}(x_{j+1})$, $L_0^{(i+1,j)}(G)$ et $L_0^{(i+1,j)}(x_{j+1})$ sont calculés et placés dans le tableau L, ensuite $L_0^{(i,j)}(x_{j+1})$ est remplacé par la différence $(L_0^{(i+1,j)}(x_{j+1}) - L_0^{(i,j)}(x_{j+1}))$.

Tableau L (sous-iter 1)

$L_0^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$
$(L_0^{(i+1,j)}(x_{j+1}) - L_0^{(i,j)}(x_{j+1}))$	$L_0^{(i+1,j)}(x_{j+1})$

Sous-itération 2

$L_1^{(i,j)}(G)$ est calculé à partir des trois éléments $L_0^{(i,j)}(G)$, $L_0^{(i+1,j)}(G)$ et $(L_0^{(i+1,j)}(x_{j+1}) - L_0^{(i,j)}(x_{j+1}))$, ensuite il est placé dans le tableau L.

Tableau L (sous-iter 1)

$L_1^{(i,j)}(G)$	$L_0^{(i+1,j)}(G)$
$(L_0^{(i+1,j)}(x_{j+1}) - L_0^{(i,j)}(x_{j+1}))$	$L_0^{(i+1,j)}(x_{j+1})$

$\vdots$

Itération $n+2$

Sous-itération $m+1$ $(m = 1, \cdots, n+1)$

A la fin de la sous-itération m, le tableau L contient les éléments de la figure 1.9.

Avec cette altération de la méthode 2, les différences qui apparaîssent comme dénominateur dans la relation (L), et qui doivent être utilisées plus d'une fois dans une même sous-itération ou qui sont nécessaires dans la suite des calculs, sont gardées dans le tableau L

Figure 1.9: Tableau L (itération $n+2$ / sous-itération m)

	1	2	$\cdots$	$n-m+1$
1	$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_m^{(i+n-m,j)}(G)$
2	$(L_0^{(i+1,j)}(x_{j+1})-$ $L_0^{(i,j)}(x_{j+1}))$	$(L_0^{(i+2,j)}(x_{j+1})-$ $L_0^{(i+1,j)}(x_{j+1}))$	$\cdots$ $\cdots$	$(L_0^{(i+n-m+1,j)}(x_{j+1})-$ $L_0^{(i+n-m,j)}(x_{j+1}))$
3	$(L_1^{(i+1,j)}(x_{j+2})-$ $L_1^{(i,j)}(x_{j+2}))$	$(L_1^{(i+2,j)}(x_{j+2})-$ $L_1^{(i+1,j)}(x_{j+2}))$	$\cdots$ $\cdots$	$(L_1^{(i+n-m+1,j)}(x_{j+2})-$ $L_1^{(i+n-m,j)}(x_{j+2}))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m+1$	$(L_{m-1}^{(i+1,j)}(x_{j+m})-$ $L_{m-1}^{(i,j)}(x_{j+m}))$	$(L_{m-1}^{(i+2,j)}(x_{j+m})-$ $L_{m-1}^{(i+1,j)}(x_{j+m}))$	$\cdots$ $\cdots$	$(L_{m-1}^{(i+n-m+1,j)}(x_{j+m})-$ $L_{m-1}^{(i+n-m,j)}(x_{j+m}))$
$m+2$	$(L_m^{(i+1,j)}(x_{j+m+1})-$ $L_m^{(i,j)}(x_{j+m+1}))$	$(L_m^{(i+2,j)}(x_{j+m+1})-$ $L_m^{(i+1,j)}(x_{j+m+1}))$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n+1$	$(L_{n-1}^{(i+1,j)}(x_{j+n})-$ $L_{n-1}^{(i,j)}(x_{j+n}))$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+n})$
$n+2$	$L_{m-1}^{(i,j)}(x_{j+n+1})$	$L_{m-1}^{(i+1,j)}(x_{j+n+1})$	$\cdots$	$L_{m-1}^{(i+n-m,j)}(x_{j+n+1})$

$n-m+2$	$n-m+3$	$\cdots$	$n+1$	$n+2$
$L_{m-1}^{(i+n-m+1,j)}(G)$	$L_{m-1}^{(i+n-m+2,j)}(G)$	$\cdots$	$L_1^{(i+n,j)}(G)$	$L_0^{(i+n+1,j)}(G)$
$(L_0^{(i+n-m+2,j)}(x_{j+1})-$ $L_0^{(i+n-m+1,j)}(x_{j+1}))$	$(L_0^{(i+n-m+3,j)}(x_{j+1})-$ $L_0^{(i+n-m+2,j)}(x_{j+1}))$	$\cdots$	$(L_0^{(i+n+1,j)}(x_{j+1})-$ $L_0^{(i+n+1,j)}(x_{j+1}))$	$L_0^{(i+n+1,j)}(x_{j+1})$
$(L_1^{(i+n-m+2,j)}(x_{j+2})-$ $L_1^{(i+n-m+1,j)}(x_{j+2}))$	$(L_1^{(i+n-m+3,j)}(x_{j+2})-$ $L_1^{(i+n-m+2,j)}(x_{j+2}))$	$\cdots$	$L_1^{(i+n,j)}(x_{j+2})$	$L_0^{(i+n+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(L_{m-1}^{(i+n-m+2,j)}(x_{j+m})-$ $L_{m-1}^{(i+n-m+1,j)}(x_{j+m}))$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+m})$	$L_0^{(i+n+1,j)}(x_{j+m})$
$L_{m-1}^{(i+n-m+1,j)}(x_{j+m+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m+1})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+m+1})$	$L_0^{(i+n+1,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_{m-1}^{(i+n-m+1,j)}(x_{j+n})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+n})$	$L_0^{(i+n+1,j)}(x_{j+n})$
$L_{m-1}^{(i+n-m+1,j)}(x_{j+n+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n+1})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+n+1})$	$L_0^{(i+n+1,j)}(x_{j+n+1})$

tout de suite après avoir été calculées la première fois qu'elles ont été nécessaires.

Ainsi, le calcul d'un nouvel élément est fait à partir de trois éléments du tableau L, et pas quatre comme avant, puisque l'un de ces éléments contient déjà la différence qui figure comme dénominateur dans la relation (L).

Calculons les éléments correspondants à l'itération $m+1$. Après le calcul de l'élément $L_m^{(i+n-m+1,j)}(x_{j+m+1})$, nous remplaçons l'élément $L_m^{(i+n-m,j)}(x_{j+m+1})$ par la différence

$$(L_m^{(i+n-m+1,j)}(x_{j+m+1}) - L_m^{(i+n-m,j)}(x_{j+m+1}))$$

puisque celle-ci va être nécessaire dans la suite des calculs.

A la fin de la sous-itération $m+1$, le tableau L contient les éléments de la figure 1.10.

Dans le cas du calcul du tableau X, cette technique permet de calculer une seule fois les rapports qui apparaissent dans la relation (X).

Plus précisément, si nous reprenons la figure 1.7 avec les schémas remplacés par les relations correspondantes, (voir figure 1.11), ce sont les rapports

$$\frac{L_{i+l-1}(x_{l-1}^{(i,j+m+1)})}{L_{i+l-1}(x_{l-1}^{(i,j+m)})}$$

qui ne vont être calculés qu'une seule fois.

Et, nous finissons ici notre démarche d'économie des calculs.

Le coût du calcul de $(K_k^{(i,j)}(c,G))_{k=0,\ldots,n}$ dépend toujours du choix des fonctionnelles initialles L_i et si nous faisons du calcul symbolique exact, symbolique non exact, numérique exact ou numérique non exact.

Supposons que nous fassions du calcul numérique non exact. Nous cherchons le coût du calcul des tableaux X et L de l'itération 1 jusqu'à $n+1$. Il est plus facile de nous guider par les tableaux $X_1,\ldots,X_{n+1}$ et $L_1,\ldots,L_{n+1}$, tout en raisonnant selon la méthode de calcul 2.

Pour le calcul des tableaux X_1 et L_1, nous devons calculer:

- x_{j+m} , $m = 0,\ldots,n$
- $c(x_{j+m})$, $m = 0,\ldots,n$
- $L_{i+p}(x_{j+m})$, $m,p = 0,\ldots,n$
- $L_{i+p}(G)$, $p = 0,\ldots,n$
- $(n+1)^2$ divisions
- n différences et n divisions

Pour le calcul de chaque élément des tableaux X_{l+1} et L_{l+1}, $l = 1,\ldots,n$, nous effectuons respectivement une multiplication et une différence, et une différence et une division. Ceci fait un total de $(n+1-l)^2$ multiplications et $(n+1-l)^2$ différences pour le tableau X_{l+1}, et $(n+1-l)^2$ différences et $(n+1-l)^2$ divisions pour le tableau L_{l+1}. Nous devons encore

Figure 1.10: Tableau L (itération $n+2$ / sous-itération $m+1$)

	1	2	$\cdots$	$n-m+1$
1	$L_n^{(i,j)}(G)$	$L_{n-1}^{(i+1,j)}(G)$	$\cdots$	$L_m^{(i+n-m,j)}(G)$
2	$(L_0^{(i+1,j)}(x_{j+1})-$ $L_0^{(i,j)}(x_{j+1}))$	$(L_0^{(i+2,j)}(x_{j+1})-$ $L_0^{(i+1,j)}(x_{j+1}))$	$\cdots$ $\cdots$	$(L_0^{(i+n-m+1,j)}(x_{j+1})-$ $L_0^{(i+n-m,j)}(x_{j+1}))$
3	$(L_1^{(i+1,j)}(x_{j+2})-$ $L_1^{(i,j)}(x_{j+2}))$	$(L_1^{(i+2,j)}(x_{j+2})-$ $L_1^{(i+1,j)}(x_{j+2}))$	$\cdots$ $\cdots$	$(L_1^{(i+n-m+1,j)}(x_{j+2})-$ $L_1^{(i+n-m,j)}(x_{j+2}))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m+1$	$(L_{m-1}^{(i+1,j)}(x_{j+m})-$ $L_{m-1}^{(i,j)}(x_{j+m}))$	$(L_{m-1}^{(i+2,j)}(x_{j+m})-$ $L_{m-1}^{(i+1,j)}(x_{j+m}))$	$\cdots$ $\cdots$	$(L_{m-1}^{(i+n-m+1,j)}(x_{j+m})-$ $L_{m-1}^{(i+n-m,j)}(x_{j+m}))$
$m+2$	$(L_m^{(i+1,j)}(x_{j+m+1})-$ $L_m^{(i,j)}(x_{j+m+1}))$	$(L_m^{(i+2,j)}(x_{j+m+1})-$ $L_m^{(i+1,j)}(x_{j+m+1}))$	$\cdots$	$(L_m^{(i+n-m+1,j)}(x_{j+m+1})-$ $L_m^{(i+n-m,j)}(x_{j+m+1}))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n+1$	$(L_{n-1}^{(i+1,j)}(x_{j+n})-$ $L_{n-1}^{(i,j)}(x_{j+n}))$	$L_{n-1}^{(i+1,j)}(x_{j+n})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+n})$
$n+2$	$L_m^{(i,j)}(x_{j+n+1})$	$L_m^{(i+1,j)}(x_{j+n+1})$	$\cdots$	$L_m^{(i+n-m,j)}(x_{j+n+1})$

$n-m+2$	$n-m+3$	$\cdots$	$n+1$	$n+2$
$L_m^{(i+n-m+1,j)}(G)$	$L_{m-1}^{(i+n-m+2,j)}(G)$	$\cdots$	$L_1^{(i+n,j)}(G)$	$L_0^{(i+n+1,j)}(G)$
$(L_0^{(i+n-m+2,j)}(x_{j+1})-$ $L_0^{(i+n-m+1,j)}(x_{j+1}))$	$(L_0^{(i+n-m+3,j)}(x_{j+1})-$ $L_0^{(i+n-m+2,j)}(x_{j+1}))$	$\cdots$	$(L_0^{(i+n+1,j)}(x_{j+1})-$ $L_0^{(i+n+1,j)}(x_{j+1}))$	$L_0^{(i+n+1,j)}(x_{j+1})$
$(L_1^{(i+n-m+2,j)}(x_{j+2})-$ $L_1^{(i+n-m+1,j)}(x_{j+2}))$	$(L_1^{(i+n-m+3,j)}(x_{j+2})-$ $L_1^{(i+n-m+2,j)}(x_{j+2}))$	$\cdots$	$L_1^{(i+n,j)}(x_{j+2})$	$L_0^{(i+n+1,j)}(x_{j+2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(L_{m-1}^{(i+n-m+2,j)}(x_{j+m})-$ $L_{m-1}^{(i+n-m+1,j)}(x_{j+m}))$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+m})$	$L_0^{(i+n+1,j)}(x_{j+m})$
$L_m^{(i+n-m+1,j)}(x_{j+m+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+m+1})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+m+1})$	$L_0^{(i+n+1,j)}(x_{j+m+1})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L_m^{(i+n-m+1,j)}(x_{j+n})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+n})$	$L_0^{(i+n+1,j)}(x_{j+n})$
$L_m^{(i+n-m+1,j)}(x_{j+n+1})$	$L_{m-1}^{(i+n-m+2,j)}(x_{j+n+1})$	$\cdots$	$L_1^{(i+n,j)}(x_{j+n+1})$	$L_0^{(i+n+1,j)}(x_{j+n+1})$

Figure 1.11: Calcul du tableau X_{l+1} à partir du tableau X_l

$$c(x_l^{(i,j+m)}) = c(x_{l-1}^{(i,j+m+1)}) - \frac{L_{i+l-1}(x_{l-1}^{(i,j+m+1)})}{L_{i+l-1}(x_{l-1}^{(i,j+m)})}\, c(x_{l-1}^{(i,j+m)})$$

$$L_{i+p}(x_l^{(i,j+m)}) = L_{i+p}(x_{l-1}^{(i,j+m+1)}) - \frac{L_{i+l-1}(x_{l-1}^{(i,j+m+1)})}{L_{i+l-1}(x_{l-1}^{(i,j+m)})}\, L_{i+p}(x_{l-1}^{(i,j+m)})$$

$l = 1, \ldots, n$, $m = 0, 1, \ldots, n-l$, $p = l, \ldots, n-1$, i et j fixés.

effectuer $(n-l)$ divisions et $(n-l)$ différences pour préparer les deuxièmes lignes des tableaux X_{l+1} et L_{l+1}.

N'oublions pas qu'après le calcul de $(L_k^{(i,j)}(c,G))_{k=0,\ldots,n}$ et de $(c(x_k^{(i,j)}))_{k=0,\ldots,n}$ le calcul de $(K_k^{(i,j)}(c,G))_{k=0,\ldots,n}$ coûte $n+1$ multiplications et n additions.

Si nous comptabilisons toutes ces opérations, nous obtenons:

- $\approx n^3/3$ divisions
- $\approx 2n^3/3$ différences
- $\approx n^3/3$ multiplications
- n additions

1.3.2 Algorithme

Après avoir discuté en détail tous les problèmes liés au calcul de la suite $(K_k^{(i,j)}(c,G))_{k=0,1,\ldots}$, nous présentons l'algorithme correspondant, qui a été écrit en envisageant son implémentation en *Mathematica* (voir Wolfram [13]). En effet, nous nous servons déjà de la sctructure de *liste* et des fonctions *Append*, et *AppendTo* de ce langage. Le lecteur qui n'est pas familiarisé avec *Mathematica*, est invité à consulter l'annexe B où nous donnons toutes les explications nécéssaires pour comprendre l'algorithme.

Algorithme de calcul de $(K_k^{(i,j)}(c,G))_{k=0,\ldots,n}$

- **Calcul de $\mathbf{K_0^{(iJ)}(c,G)}$**

  $cm \leftarrow \{c_0, c_1, \ldots, c_j\}$

  $wx \leftarrow \{x_j\}$

  $wc \leftarrow \{c(wx[[1]])\}$

  $wl \leftarrow \{L_i(wx[[1]])\}$

  $larray \leftarrow \{\{L_i(g(x,t))/wl[[1]]\}\}$

  $xarray \leftarrow \{AppendTo[wc, wl[[1]]]\}$

  $w1 \leftarrow larray[[1,1]] * xarray[[1,1]]$

  $\mathbf{larray[[1,1]] = L_0^{(iJ)}(G)}$, $\mathbf{xarray[[1,1]] = c(x_0^{(iJ)})}$
  $\mathbf{w1 = K_0^{(iJ)}(c,G)}$

  write $w1$

- **For** $l = 2, \ldots, n+1$ **do:**

  Calcul de $\mathbf{K_{l-1}^{(iJ)}(c,G)}$

  $AppendTo[cm, c_{j+l-1}]$

  $AppendTo[wx, x_{j+l-1}]$

  $AppendTo[wc, c(wx[[l]])]$

  $AppendTo[wl, L_{i+l-1}(wx[[1]])]$

  Calcul des nouveaux éléments des tableaux $\mathbf{X_1}$ et $\mathbf{L_1}$

  - **1ère Partie: Calcul simultané de la nouvelle colonne du tableau X et de la nouvelle ligne du tableau L**

    $AppendTo[xarray, \{wc[[l]]\}]$

    For $n = 2, \ldots, l$ **do:**

    $w \leftarrow L_{i+n-2}(wx[[l]])$

    If $n = 2$ **then**

    $xarray[[l-1,2]] \leftarrow w/xarray[[l-1,2]]$

    end if

    $AppendTo[xarray[[l]], w]$

    $AppendTo[larray[[n-1]], w/wl[[n-1]]]$

    end for

  - **2ème Partie: Calcul simultané de la nouvelle colonne du tableau L et de la nouvelle ligne du tableau X**

    $AppendTo[larray, L_{i+l-1}(g(x,t))/wl[[l]]]$

    $AppendTo[xarray[[1]], wl[[l]]]$

    For $n = 2, \ldots, l$ **do:**

$w \leftarrow L_{i+l-1}(wx[[n]])$
$AppenTo[larray[[l]], w/wl[[l]]]$
If $n = 2$ **then**
$larray[[l-1,2]] = larray[[l,2]] - larray[[l-1,2]]$
end if
$AppendTo[xarray[[n]], w]$
end for

For $m = 2, \ldots, l$ **do:**

Calcul des nouveaux éléments des tableaux X_m et L_m

- **1ère Partie: Calcul des nouveaux éléments situés dans une même ligne des tableaux X et L**

For $n = 1, \ldots, l-m$ **do:**
$n1 \leftarrow n+1$
$xarray[[n,l]] \leftarrow xarray[[n1,l]] - xarray[[n,m]] * xarray[[n,l]]$
$larray[[n,l]] \leftarrow (larray[[n1,l]] - larray[[n,l]])/larray[[n,m]]$
end for

- **2ème Partie: Calcul des nouveaux éléments situés dans une même colonne des tableaux X et L**

$n1 \leftarrow l-m+1$
$n2 \leftarrow n1+1$
$n3 \leftarrow n1-1$
$m1 \leftarrow m+1$
$xarray[[n1,1]] \leftarrow xarray[[n2,1]] - xarray[[n1,m]] * xarray[[n1,1]]$
$larray[[n1,1]] \leftarrow (larray[[n2,1]] - larray[[n1,1]])/larray[[n1,m]]$
For $n = m1, \ldots, l$ **do:**
$xarray[[n1,n]] \leftarrow xarray[[n2,n]] - xarray[[n1,m]] * xarray[[n1,n]]$
$larray[[n1,n]] \leftarrow (larray[[n2,n]] - larray[[n1,n]])/larray[[n1,m]]$
If $n = m1$ **then**
$xarray[[n3,m1]] = xarray[[n1,m1]]/xarray[[n3,m1]]$
$larray[[n3,m1]] = larray[[n1,m1]] - larray[[n3,m1]]$
end if
end for
$w2 \leftarrow w1 + larray[[1,1]] * xarray[[1,1]]$

$\mathbf{larray[[1,1]] = L_{l-1}^{(iJ)}(G)}$, $\mathbf{xarray[[1,1]] = c(x_{l-1}^{(iJ)})}$
$\mathbf{w_2 = K_{l-1}^{(iJ)}(c, G)}$

write $w2$
$w1 \leftarrow w2$

end for

1.3.3 Conditions d'applicabilité

Nous avons déjà vu que le calcul de

$$\{K_k^{(i,j)}(c,G)\}_{k=0,\ldots,n} \text{ , avec les indices i et j fixés}$$

à partir des relations (K), (L) et (X) exige le calcul des éléments des tableaux $L_1,\ldots,L_{n+1}$ et $X_1,\ldots,X_{n+1}$ des figures 1.1 et 1.6.

Le problème que nous posons maintenant est de savoir sous quelles conditions le calcul de ces éléments est possible.

Commençons par le tableau L_1, qui contient les conditions initialles

$$L_0^{(i+p,j)}(G)\ ,\ L_0^{(i+p,j)}(x_{j+q})\ \ ,p=0,\ldots,n\ ,q=1,\ldots,n.$$

De (1.15) nous écrivons:

$$L_0^{(i+p,j)}(G) = \frac{L_{i+p}(G)}{L_{i+p}(x_j)}\ \ ,p=0,\ldots,n$$
$$L_0^{(i+p,j)}(x_{j+q}) = \frac{L_{i+p}(x_{j+q})}{L_{i+p}(x_j)}\ \ ,p=0,\ldots,n\ ,q=1,\ldots,n.$$

Le calcul de ces éléments est possible ssi

$$L_{i+p}(x_j) = D_1^{(i+p,j)} \neq 0\ \ ,p=0,\ldots,n. \tag{1.17}$$

Le calcul des tableaux restants $(L_{q+1})_{q=1,\ldots,n}$ est fait en appliquant la relation (L). Ce calcul est possible ssi les différences qui apparaissent comme dénominateur dans la relation (L) sont non nulles, c'est-à-dire:

$$L_{q-1}^{(i+p+1,j)}(x_{j+q}) - L_{q-1}^{(i+p,j)}(x_{j+q}) \neq 0\ \ ,q=1,\ldots,n\ ,p=0,\ldots,n-q.$$

D'après la définition (1.12), ces conditions sont équivalentes aux conditions:

$$\frac{\bar{N}_q^{(i+p+1,j)}(x_{j+q})D_q^{(i+p,j)} - D_q^{(i+p+1,j)}\bar{N}_q^{(i+p,j)}(x_{j+q})}{D_q^{(i+p+1,j)}D_q^{(i+p,j)}} \neq 0 \tag{1.18}$$
$$,q=1,\ldots,n\ ,p=0,\ldots,n-q$$

En appliquant l'identité de Sylvester (voir annexe E) au déterminant $(-1)^q D_{q+1}^{(i+p,j)}$, nous obtenons:

$$D_{q+1}^{(i+p,j)}D_{q-1}^{(i+p+1,j)} = \bar{N}_q^{(i+p+1,j)}(x_{j+q})D_q^{(i+p,j)} - D_q^{(i+p+1,j)}\bar{N}_q^{(i+p,j)}(x_{j+q}).$$

Les conditions (1.18) sont équivalentes aux conditions:

$$D_{q+1}^{(i+p,j)} \neq 0 \text{ et } D_{q-1}^{(i+p+1,j)} \neq 0 \text{ et } D_q^{(i+p+1,j)} \neq 0 \text{ et } D_q^{(i+p,j)} \neq 0,$$
$$q=1,\ldots,n\ ,p=0,\ldots,n-q,$$

qui s'écrivent sous la forme[3]

$$D_q^{(i+p,j)} \neq 0 \ , q = 1, \ldots, n+1 \ , p = 0, \ldots, n-q+1,$$

et englobent les conditions (1.17).

Le calcul du tableau X_1 n'impose pas de conditions.

Le calcul des tableaux restants $(X_{q+1})_{q=1,\ldots,n}$ est fait en appliquant la relation (X). Ce calcul est possible ssi les dénominateurs des rapports qui apparaissent dans la relation (X) sont non nuls, c'est-à-dire:

$$L_{i+q-1}(x_{q-1}^{(i,j+p)}) \neq 0 \ , q = 1, \ldots, n \ , p = 0, \ldots, n-q.$$

D'après la définition (1.11), nous écrivons:

$$\frac{L_{i+q-1}(N_q^{(i,j+p)})}{D_{q-1}^{(i,j+p)}} \neq 0 \ \ , q = 1, \ldots, n \ , p = 0, \ldots, n-q. \tag{1.19}$$

Mais,

$$L_{i+q-1}(N_q^{(i,j+p)}) = D_q^{(i,j+p)},$$

donc les conditions (1.19) sont équivalentes aux conditions:

$$D_q^{(i,j+p)} \neq 0 \ et \ D_{q-1}^{(i,j+p)} \neq 0 \ \ , q = 1, \ldots, n \ , p = 0, \ldots, n-q,$$

qui s'écrivent sous la forme:

$$D_q^{(i,j+p)} \neq 0 \ \ , q = 1, \ldots, n \ , p = 0, \ldots, n-q.$$

Voyons que les conditions que nous avons obtenues coïncident avec les conditions d'existence des éléments des tableaux $L_1, \ldots, L_{n+1}$, $X_1, \ldots, X_{n+1}$.

D'après la définition (1.12), les éléments des tableaux $(L_{q+1})_{q=0,\ldots,n}$, s'écrivent de la façon suivante:

$$L_q^{(i+p,j)}(G) = \frac{\bar{N}_{q+1}^{(i+p,j)}(G)}{D_{q+1}^{(i+p,j)}} \ \ , q = 0, \ldots, n \ , p = 0, \ldots, n-q$$

$$L_q^{(i+p,j)}(x_{j+r}) = \frac{\bar{N}_{q+1}^{(i+p,j)}(x_{j+r})}{D_{q+1}^{(i+p,j)}} \ \ , q = 0, \ldots, n-1 \ , p = 0, \ldots, n-q \ , r = q+1, \ldots, n.$$

Ces éléments existent ssi

$$D_{q+1}^{(i+p,j)} \neq 0 \ , q = 0, \ldots, n \ , p = 0, \ldots, n-q.$$

D'après la définition (1.11) les éléments des tableaux $(X_{q+1})_{q=0,\ldots,n}$, s'écrivent de la façon suivante:

[3] $\forall i, j > 0, \ D_0^{(i,j)} = 1$

$$c(x_q^{(i,j+p)}) = \frac{c(N_{q+1}^{(i,j+p)})}{D_q^{(i,j+p)}} \ , \ L_{i+r}(x_q^{(i,j+p)}) = \frac{L_{i+r}(N_{q+1}^{(i,j+p)})}{D_q^{(i,j+p)}}$$

$$q = 0,\ldots,n \ , p = 0,\ldots,n-q \ , r = q,\ldots,n-1.$$

Ces éléments existent ssi

$$D_q^{(i,j+p)} \neq 0 \ , q = 1,\ldots,n \ , p = 0,\ldots,n-q.$$

Le calcul de $((k)_f(t))_{k=0,\ldots,n}$ à partir de la méthode de bordage par blocs est possible ssi ces approximants existent et sont uniques, c'est-à-dire ssi

$$D_k^{(0,0)} \neq 0 \ , k = 1,\ldots,n+1.$$

Nous constatons que la méthode de calcul qui utilise les relations (K), (L) et (X) exige des conditions d'applicabilité plus restrictives que celles correspondantes à l'existence et l'unicité des approximants calculés.

Par exemple, si $L_i(f) = f^{(i)}(z)$, où $i > 0$ et $z \in C$ est fixé, alors $D_k^{(0,0)} \neq 0 \ , \forall k > 0$, les approximants $((k)_f(t))_{k=0,\ldots}$ existent, sont uniques et la méthode de bordage permet leur calcul. Mais $D_q^{(p,0)} = 0 \ , \forall p > 0$ et nous ne pouvons pas calculer ces approximants à partir des relations de récurrence.

Nous devons remarquer que l'évaluation de la fonctionnelle c dans les éléments x_{j_s} $(c(x_{j_s}))$, ou l'évaluation des fonctionnelles L_{i_s} dans la fonction G $(L_{i_s}(G))$, ou dans les éléments x_{j_s} $(L_{i_s}(x_{j_s}))$, peut ne pas être possible et par conséquence empêcher le calcul des approximants $(k)_f(t), k = 0,\ldots,n$. Nous avons exclu cette possibilité de l'étude précécente.

1.3.4 Commentaires au programme

Le lecteur qui n'est pas familiarisé avec *Mathematica* (voir Wolfram [13]) est vivement invité à consulter l'annexe B avant d'aborder cette section.

Nous avons développé un logiciel, écrit en *Mathematica*, qui calcule une suite d'approximants de type Padé généralisés, en utilisant l'algorithme de la section 1.3.2. Ce logiciel est constitué par trois *packages*: GPADETYPE.M, BE.M et BIF.M, que nous donnons dans l'annexe C, avec une *Session Mathematica* qui permet de mieux comprendre le fonctionnement du logiciel.

GPADETYPE.M contient la définition de la fonction *tablegpt*, qui calcule une suite d'approximants de type Padé généralisés. Il appelle les *packages* BE.M et BIF.M.

BE.M contient les définitions de la série de fonctions *series*, de la fonction génératrice g, des moments modifiés $c_i = c(u_i(x))$, des polynômes u_i et des fonctions complexes $f_i = f_i(t)$ de l'exemple que l'utilisateur veut tester.

BIF.M contient la définition des fonctionnelles initialles L_i (*initialf*) que l'utilisateur a choisit.

Donnons quelques explications sur la fonction *tablegpt*.

tablegpt[k , t , *toption* , *coption* , *psoption* ,
resultprec , *preoption* , *preprec* , *cprec* , *aroption* , *compoption* , *norm*]
calcule les $(k+1)$ premiers approximants de type Padé généralisés au point t.

L'argument t peut être une expression symbolique (*toption* == *"VAR"*), ou une expression numérique (*toption* == *"SCL"*). Si t est une expression symbolique, les approximants seront eux aussi des expressions symboliques et nous faisons du calcul symbolique. Si t est une expression numérique, les approximants seront eux aussi des expressions numériques et nous faisons du calcul numérique.

Si *coption* == *"EC"*, nous faisons du calcul exact. Si *coption* == *"AC"*, nous faisons du calcul approché avec précision *cprec*. Plus précisément, nous faisons tous les calculs intermédiaires en utilisant la fonction *N[- , cprec]*. Si *cprec* <= *precision de la machine* , nous faisons du calcul approché de basse précision. Si *cprec* > *precision de la machine*, nous faisons du calcul approché de haute précision.

Les résultats numériques ou la partie numérique des résultats symboliques sont écrits avec *resultprec* chiffres significatifs. Nous devons choisir *resultprec* <= *cprec*.

Si *psoption* == 1, les sommes partielles de la série sont aussi calculées en accord avec les valeurs de *toption* et de *coption*. Si *psoption* ! = 1, les sommes partielles ne sont pas calculées. Nous remarquons qu'il est intéressant de calculer les sommes partielles surtout quand *toption* == *"SCL"*, c'est-à-dire quand nous obtenons des résultats numériques. Nous comparons la suite des approximants de type Padé généralisés avec la suite des sommes partielles de la série.

Si *toption* == *"SCL"* et *preoption* == 1, les erreurs rélatives des approximants par rapport à la série $(|((k)_f(t) - f(t))/f(t)|_{k=0,1,...})$ sont calculées avec précision *cprec* et sont écrites avec *preprec* chiffres significatifs. En outre, si *psoption* == 1, les erreurs rélatives des sommes partielles par rapport à la série $(|(ps(k)(t) - f(t))/f(t)|_{k=0,1,...})$ sont calculées avec précision *cprec* et sont écrites avec *preprec* chiffres significatifs.

Si *toption* == *"SCL"* et *preoption* == 2, le nombre de chiffres exacts des approximants par rapport à la série $(-\log_{10}|(k)_f(t) - f(t)|_{k=0,1,...})$ sont calculés avec précision *cprec*, et sont écrits avec *preprec* chiffres significatifs. En outre, si *psoption* == 1, le nombre de chiffres exacts des sommes partielles par rapport à la série $(-\log_{10}|ps(k)(t) - f(t)|_{k=0,1,...})$ est calculé avec précision *cprec* et est écrit avec *preprec* chiffres significatifs.

Si *compoption* == 0, nous faisons du calcul réel. Si *compoption* == 1, nous faisons du calcul complex.

Si *compoption* == 1, *toption* == *"SCL"* et *preoption* == 1, les erreurs rélatives des parties réelles et des parties imaginaires des approximants, par rapport aux parties réelles et imaginaires de la série sont calculées avec précision *cprec* et sont écrites avec *preprec* chiffres significatives. Si *psoption* == 1, nous faisons de même pour les sommes partielles.

Si *compoption* == 1, *toption* == *"SCL"* mais *preoption* == 2, nous calculons le nombre de chiffres exacts à la place des erreurs rélatives.

Si *compoption* == 1 and *toption* == *"SCL"*, nous calculons aussi la norme p ($p >= 1$) des vecteurs des erreurs rélatives (si *preoption* == 1), ou la norme p des vecteurs du nombre de chiffres exacts (si *preoption* == 2). La valeur de p est donnée par l'argument *norm* de la fonction *tablegpt*.

Si *norm* == −1, nous calculons la norme infinie des vecteurs des erreurs rélatives (si *preoption* == 1), ou la composante de valeur absolue minimalle des vecteurs du nombre de chiffres exacts (si *preoption* == 2).

Les normes sont calculées avec précision *cprec* et sont écrites avec *preprec* chiffres significatives.

Si *toption* == *"VAR"* ou *preoption* est différent de 1 et de 2, ni les erreurs rélatives, ni le nombre de chiffres exacts ne sont calculés.

Si *toption* == *"SCL"* et *coption* == *"EC"*, nous obtenons les expressions numériques exactes des approximants au point t. En outre, si *psoption* == 1, nous obtenons aussi les expressions numériques exactes des sommes partielles au point t. Si *preoption* == 1, les erreurs rélatives de ces expressions exactes sont calculées avec précision *cprec*. Si *preoption* == 2, le nombre de chiffres exacts est calculé avec précision *cprec*. Nous devons choisir une valeur suffisamment élevée de *cprec* de sorte que les erreurs ou le nombre de chiffres exacts puissent être correctement calculés. En outre, si *aroption* == 1, les expressions exactes sont écrites avec *resultprec* chiffres significatifs. Ce n'est pas le cas, si *aroption* ! = 1.

La valeur de la série au point t est écrite avec *resultprec* chiffres significatifs, si *toption* == *"SCL"*, *coption* == *"EC"* et *aroption* == 1, ou *toption* == *"SCL"* et *coption* == *"AC"*.

Les quatres premiers arguments de la fonction *tablegpt* sont obligatoires et les huit derniers arguments sont facultatifs. Si l'utilisateur ne donne pas de valeurs à *psoption, resultprec, preoption, preprec, cprec, aroption, compoption* et *norm*, ces arguments prennent automatiquement les valeurs *0, Precision[1.0], 0, 2, Precision[1.0], 0, 0* et −1 respectivement.

Nous rappelons que la correspondance des arguments est faite par ordre de la gauche vers la droite. Si, par exemple, l'utilisateur sait que les arguments *preoption, preprec, aroption, compoption* et *norm* ne sont pas nécessaires, mais que *cprec* est nécessaire, il est obligé de donner des valeurs fictives à *preoption* et *preprec*, parce que s'il ne donnait que sept arguments, le dernier serait interprété comme la valeur de *preoption* et pas de *cprec*. Cependant l'utilisateur n'a pas besoin de donner des valeures à *aroption, compoption* et *norm*.

Voyons à quel genre de calcul correspond toutes les combinaisons possibles des différentes valeurs de *toption, coption* et *cprec*. Pour chaque cas nous donnons un exemple. Tous les exemples sont basés sur l'exemple 1 de la section 1.2.2.

In[1] :=<<GPADETYPE.M

Lecture du *package* GPADETYPE.M, qui appelle les *packages* BE.M et BIF.M. Les définitions de la série de fonctions, de la fonction génératrice, des moments modifiés, des polynômes correspondants, des fonctions élémentaires et des fonctionnelles initialles sont connues:

In[2] := *series*[t]
Out[2] = *Log*[1 + t]/t
In[3] := $g[x,t]$

$Out[3] = 1/(1 - xt)$
$In[4] := c[i]$
$Out[4] = (-1)^i/(1 + i)$
$In[5] := u[i, x]$
$Out[5] = x^i$
$In[6] := f[i, t]$
$Out[6] = t^i$
$In[7] := initialf[i, f[x], x]$
$Out[7] = f[-(1/(1 + i))]$

Nous remarquons que toutes ces définitions sont écrites d'une façon exacte.

- **Calcul symbolique exact.**
  And[toption == "VAR", coption == "EC"]

  $In[8] :=$ *tablegpt*[3, t, *"VAR"*, *"EC"*]

  $Out[8] =$

  $fgpt[0] = 1/(1 + t)$
  $fgpt[1] = 2/(2 + t)$
  $fgpt[2] = (6 + 5t)/(2(1 + t)(3 + t))$
  $fgpt[3] = (144 + 228t + 108t^2 + 19t^3)/(6(1 + t)(2 + t)(3 + t)(4 + t))$

  t est un symbole, donc nous devons prendre *toption == "VAR"*. En prennant *coption == "EC"*, nous obtenons les expressions formelles des approximants dans la variable t.

- **Calcul symbolique approché de basse précision.**
  And[toption == "VAR", coption == "AC", cprec <= Precision[1.]]

  $In[9] :=$ *tablegpt*[3, t, *"VAR"*, *"AC"*, 0, 19, 0, 0, 19]

  $Out[9] =$

  $ngpt[0] = 1./(1. + t)$
  $ngpt[1] = (2. + 2.t)/((1. + t)(2. + 1.t))$
  $ngpt[2] = (6. + 8.t + 2.5t^2)/((1. + t)(2. + 1.t)(3. + 1.t))$
  $ngpt[3] = (24. + 38.t + 18.t^2 + 3.1666666666666666t^3)/$
  $((1. + t)(2. + 1.t)(3. + 1.t)(4. + 1.t))$

  coption == "AC" et *cprec == 19*, donc tous les calculs intermédiaires sont faits avec 19 chiffres significatifs, en utilisant la fonction *N[- , 19]*, c'est-à-dire nous faisons les calculs dans la précision de la machine. *resultprec == 19*, donc les approximants sont écrits avec 19 chiffres significatifs. Comme *toption == "VAR"*, les erreurs relatives ne sont pas calculées, et les septième et huitième arguments de la fonction *tablegpt* ne sont pas utilisés.

- **Calcul symbolique approché de haute précision.**
  And[toption == "VAR", coption == "AC", cprec > Precision[1.]]

  $In[10] :=$ *tablegpt*[3, *t*, *"VAR"*, *"AC"*, 0, 30, 0, 0, 40]

  $Out[10] =$

  $ngpt[0] = 1./(1. + t)$

  $ngpt[1] = (2. + 2.t)/((1. + t)(2. + 1.t))$

  $ngpt[2] = (6. + 8.t + 2.5t^2)/((1. + t)(3. + 1.t)(2. + 1.t))$

  $ngpt[3] = (24. + 38.t + 18.t^2 + 3.16666666666666666666666666666t^3)/$
  $((1. + t)(3. + 1.t)(2. + 1.t)(4. + 1.t))$

  Maintenant *cprec == 40*, donc tous les calculs sont faits avec 40 chiffres significatifs en utilisant la fonction *N[- , 40]*. *resultprec == 30*, donc les résultats sont écrits avec 30 chiffres significatifs.

- **Calcul numérique exact.**
  And[toption == "SCL", coption == "EC"]

  $In[11] :=$ *tablegpt*[3, 1/10, *"SCL"*, *"EC"*, 1, 19, 1 , 2, 19, 1]

  $Out[11] =$

  $fgpt[0] = 10/11$

  $fps[0] = 1$

  $f[1/10] = 0.9531017980432485999$

  $ngpt[0] = 0.9090909090909090909$

  $nps[0] = 1.$

  $regpt[0] = 0.046$ ($edgpt[0] = 1.4$)

  $reps[0] = 0.049$ ($edps[0] = 1.3$)

  $fgpt[1] = 20/21$

  $fps[1] = 19/20$

  $f[1/10] = 0.9531017980432485999$

  $ngpt[1] = 0.9523809523809523809$

  $nps[1] = 0.95$

  $regpt[1] = 0.00076$ ($edgpt[1] = 3.1$)

  $reps[1] = 0.0033$ ($edps[1] = 2.5$)

  $fgpt[2] = 325/341$

  $fps[2] = 143/150$

$f[1/10] = 0.9531017980432485999$

$ngpt[2] = 0.9530791788856304985$

$nps[2] = 0.9533333333333333333$

$regpt[2] = 0.000024$ ($edgpt[2] = 4.6$)

$reps[2] = 0.00024$ ($edps[2] = 3.6$)

$fgpt[3] = 839495/880803$

$fps[3] = 11437/12000$

$f[1/10] = 0.9531017980432485999$

$ngpt[3] = 0.9531018854386281609$

$nps[3] = 0.9530833333333333334$

$regpt[3] = 9.2\ 10^{-8}$ ($edgpt[3] = 7.1$)

$reps[3] = 0.000019$ ($edps[3] = 4.7$)

1/10 est un nombre, donc nous devons prendre *toption == "SCL"*. 1/10 est exact, comme les définitions de g, c_i, u_i et $initialf$ sont aussi exactes, nous pouvons obtenir les expressions exactes des approximants au point 1/10 ($fgpt[i]$).

Comme *psoption == 1* et la définition de f_i est exacte, nous obtenons aussi l'expression exact des sommes partielles de la série au point 1/10 ($fps[i]$).

aroption == 1, donc nous écrivons ces expressions exactes avec *resultprec == 19* chiffres significatives ($ngpt[i] = N[\ fgpt[i]\ ,\ 19\ ]$, $nps[i] = N[\ fps[i]\ ,\ 19\ ]$).

preoption == 1, donc les erreurs relatives des expressions exactes par rapport à la série sont calculées avec *cprec == 19* chiffres significatifs et sont écrites avec *preprec == 2* chiffres significatifs :

$$regpt[i] = N[\ N[\ Abs[\ (fgpt[i] - series[t])/series[t]\ ]\ ,\ 19\ ],\ 2\ ],$$

$$regps[i] = N[\ N[\ Abs[\ (fps[i] - series[t])/series[t]\ ]\ ,\ 19\ ],\ 2\ ].$$

Si nous avions pris *preoption == 2*, nous aurions calculé le nombre de chiffres exacts à la place des erreurs rélatives (voir résultats entre parenthèses) :

$$edgpt[i] = N[\ N[\ -Log[\ 10\ ,\ Abs[\ fgpt[i] - series[t]\ ]\ ]\ ,\ 19\ ],\ 2\ ],$$

$$edps[i] = N[\ N[\ -Log[\ 10\ ,\ Abs[\ fps[i] - series[t]\ ]\ ]\ ,\ 19\ ],\ 2\ ].$$

- **Calcul numérique approché de basse précision.**
  And[toption == "SCL", coption == "AC", cprec <= Precision[1.]]

  $In[12]$:= *tablegpt[3, 0.1, "SCL", "AC", 1, 19, 2, 2]*

  $Out[12]$ =

$f[0.1] = 0.9531017980432485999$

$ngpt[0] = 0.9090909090909090909$

$nps[0] = 1.$

$edgpt[0] = 1.4$

$edps[0] = 1.3$

$f[0.1] = 0.9531017980432485999$

$ngpt[1] = 0.952380952380952381$

$nps[1] = 0.95$

$edgpt[1] = 3.1$

$edps[1] = 2.5$

$f[0.1] = 0.9531017980432485999$

$ngpt[2] = 0.9530791788856304986$

$nps[2] = 0.9533333333333333333$

$edgpt[2] = 4.6$

$edps[2] = 3.6$

$f[0.1] = 0.9531017980432485999$

$ngpt[3] = 0.953101885438628161$

$nps[3] = 0.9530833333333333334$

$edgpt[3] = 7.1$

$edps[3] = 4.7$

0.1 est un nombre réel de basse précision, nous devons prendre *toption == "SCL"*. Nous n'avons pas donné de valeurs aux arguments *cprec* et *aroption*, donc ils valent automatiquement *Precision[1.0] == 19*, c'est-à-dire la précision de la machine, et 0.

psoption == 1, donc nous calculons aussi les sommes partielles de la série.

Tous les calculs intermédiaires sont faits en utilisant la fonction *N[- , 19]*. Les approximants et les sommes partielles sont écrits avec *resultprec == 19* chiffres significatifs.

preoption == 2, donc nous calculons le nombre de chiffres exacts avec précision *cprec == 19*, et nous les écrivons avec *preprec == 2* chiffres significatifs.

Même si nous faisons les calculs avec 40 chiffres significatifs (*cprec* == 40), la précision des résultats ne sera pas supérieure à 19, parce que t vaut 0.1 et *Precision*[0.1] == 19.

In[13] := *tablegpt*[3, 0.1, "*SCL*", "*AC*",1 , 30, 2, 2 ,40]

Out[13] =

f[0.1] = 0.9531017980432485999
ngpt[0] = 0.9090909090909090909
nps[0] = 1.
edgpt[0] = 1.4
edps[0] = 1.3

f[0.1] = 0.9531017980432485999
ngpt[1] = 0.9523809523809523810
nps[1] = 0.95
edgpt[1] = 3.1
edps[1] = 2.5

f[0.1] = 0.9531017980432485999
ngpt[2] = 0.9530791788856304986
nps[2] = 0.9533333333333333333
edgpt[2] = 4.6
edps[2] = 3.6

f[0.1] = 0.9531017980432485999
ngpt[3] = 0.9531018854386281610
nps[3] = 0.9530833333333333334
edgpt[3] = 7.1
edps[3] = 4.7

- **Calcul numérique approché de haute précision.**
  And[toption == "SCL", coption == "AC", cprec > Precision[1.]]

In[14] := *tablegpt*[3, 1/10, "*SCL*", "*AC*", 0, 30, 1, 2, 40]

Out[14] =

f[1/10] = 0.953101798043248600439521232800
ngpt[0] = 0.909090909090909090909090909091
regpt[0] = 0.046

f[1/10] = 0.953101798043248600439521232800
ngpt[1] = 0.952380952380952380952380952381

$regpt[1] = 0.00076$

$f[1/10] = 0.95310179804324860043952123 28$

$ngpt[2] = 0.95307917888563049853372434 0176$

$regpt[2] = 0.000024$

$f[1/10] = 0.95310179804324860043952123 28$

$ngpt[3] = 0.95310188543862816089409323 0836$

$regpt[3] = 9.2\ 10^{-8}$

cprec == *40*, donc tous les calculs intermédiaires sont faits avec 40 chiffres significatifs en utilisant la fonction *N[- , 40]*. *resultprec* == *30*, donc les approximants sont écrits avec 30 chiffres significatifs.

Si nous avions utilisé 0.1 à la place de 1/10 les résultats n'auraient pas une précision supérieure à 19. Remarquons que (*Precision[0.1]* == *19* et *Precision[1/10]* == *Infinity*). *preoption* == *1*, donc les erreurs relatives sont calculées avec *cprec* == *40* chiffres significatifs, et sont écrites avec *preprec* == *2* chiffres significatifs.

Le logiciel que nous avons écrit exploite les capacités de *Mathematica*, qui n'existent pas en *Fortran*; c'est-à-dire les capacités de faire du calcul symbolique, du calcul exact et du calcul approché de haute précision.

Le calcul exact, s'il est possible, est très souhaitable, mais prend en général énormément de temps. On est souvent obligé de faire du calcul approché. Le calcul approché produit des erreurs d'arrondi qui vont entacher les résultats. Quand nous augmentons la précision des calculs intermédiaires ces erreurs diminuent. Une bonne manière de savoir si les résultats sont affectés par des erreurs d'arrondi est de répéter leur calcul en utilisant des précisions croissantes. Tant que les résultats obtenus avec deux précisions consécutives sont différents, les erreurs d'arrondi sont encore présentes et nous devons augmenter la précision des calculs intermédiaires.

Supposons que les définitions de *series*, g, c_i, u_i et f_i soient exactes et que t soit une expression numérique exacte. Dans ce cas, si nous faisons du calcul exact, nous obtenons les expressions numériques exactes des approximants au point t. Si nous faisons du calcul approché avec une précision *cprec*, nous obtenons des approximations numériques des valeurs des approximants au point t. Si nous utilisons plusieurs valeurs de *cprec*, en commençant par la précision de la machine et en augmentant *cprec* à chaque étape, nous observons que les erreurs relatives par rapport à la série diminuent jusqu'à égaler les erreurs relatives correspondantes aux expressions numériques exactes. Les erreurs d'arrondi diminuent au fur et à mesure que *cprec* augmente jusqu'à ne plus affecter les chiffres corrects des résultats approchés.

Les tables 1.5, 1.6, 1.7 et 1.8 sont destinées à illustrer ce commentaire. Nous utilisons les exemples 1 et 2 de la section 1.2.2. La première colonne de chaque table contient les erreurs relatives des expressions exactes des approximants par rapport à la série et constitue notre

référence. Les autres colonnes contiennent les erreurs relatives des résultats obtenus en faisant du calcul approché avec plusieurs précisions. Dans la dernière ligne des tables, nous écrivons les secondes CPU nécessaires au calcul de chaque colonne. Ces valeurs ont été obtenues en utilisant la fonction *Mathematica Timing* (voir Wolfram [13, page 317]).

Tableau 1.5: Exemple 1 ($t = 1/10$)

n	exact cprec=30	N[-,19]	N[-,30]	N[-,40]	N[-,50]
0	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$
1	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$
2	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$
3	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$
4	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$
5	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$
6	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$
7	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$
8	$4.9\ 10^{-13}$	$5.0\ 10^{-13}$	$4.9\ 10^{-13}$	$4.9\ 10^{-13}$	$4.9\ 10^{-13}$
9	$3.7\ 10^{-14}$	$8.5\ 10^{-14}$	$3.7\ 10^{-14}$	$3.7\ 10^{-14}$	$3.7\ 10^{-14}$
10	$2.9\ 10^{-15}$	$4.2\ 10^{-12}$	$2.9\ 10^{-15}$	$2.9\ 10^{-15}$	$2.9\ 10^{-15}$
11	$2.3\ 10^{-16}$	$1.4\ 10^{-10}$	$2.3\ 10^{-16}$	$2.3\ 10^{-16}$	$2.3\ 10^{-16}$
12	$1.9\ 10^{-17}$	$5.3\ 10^{-9}$	$1.9\ 10^{-17}$	$1.9\ 10^{-17}$	$1.9\ 10^{-17}$
13	$1.5\ 10^{-18}$	$2.1\ 10^{-7}$	$1.5\ 10^{-18}$	$1.5\ 10^{-18}$	$1.5\ 10^{-18}$
14	$1.3\ 10^{-19}$	$8.1\ 10^{-6}$	0	$1.3\ 10^{-19}$	$1.3\ 10^{-19}$
15	$1.1\ 10^{-20}$	$3.1\ 10^{-4}$	0	$1.1\ 10^{-20}$	$1.1\ 10^{-20}$
16	$8.9\ 10^{-22}$	$1.1\ 10^{-2}$		$8.9\ 10^{-22}$	$8.9\ 10^{-22}$
17	$7.6\ 10^{-23}$	$4.4\ 10^{-1}$		0	$7.6\ 10^{-23}$
18	$6.5\ 10^{-24}$	$1.8\ 10^{+1}$		0	$6.5\ 10^{-24}$
19	$5.7\ 10^{-25}$				$5.7\ 10^{-25}$
sec CPU	297	105	79	149	159

Note: Quand un zéro apparaît dans les tables, ceci ne signifie pas que l'erreur relative correspondant soit nulle, mais qu'il n'y a plus assez de précision pour la calculer.

Tableau 1.6: Exemple 1 ($t = 5/10$)

n	exact cprec=19	N[-,19]	N[-,30]	N[-,40]	N[-,50]	N[-,60]
0	$1.8\ 10^{-1}$	$1.8\ 10^{-1}$	$1.8\ 10^{-1}$	$1.8\ 10^{-1}$	$1.8\ 10^{-1}$	$1.8\ 10^{-1}$
1	$1.3\ 10^{-2}$	$1.3\ 10^{-2}$	$1.3\ 10^{-2}$	$1.3\ 10^{-2}$	$1.3\ 10^{-2}$	$1.3\ 10^{-2}$
2	$1.7\ 10^{-3}$	$1.7\ 10^{-3}$	$1.7\ 10^{-3}$	$1.7\ 10^{-3}$	$1.7\ 10^{-3}$	$1.7\ 10^{-3}$
3	$5.6\ 10^{-6}$	$5.6\ 10^{-6}$	$5.6\ 10^{-6}$	$5.6\ 10^{-6}$	$5.6\ 10^{-6}$	$5.6\ 10^{-6}$
4	$3.4\ 10^{-5}$	$3.4\ 10^{-5}$	$3.4\ 10^{-5}$	$3.4\ 10^{-5}$	$3.4\ 10^{-5}$	$3.4\ 10^{-5}$
5	$8.0\ 10^{-6}$	$8.0\ 10^{-6}$	$8.0\ 10^{-6}$	$8.0\ 10^{-6}$	$8.0\ 10^{-6}$	$8.0\ 10^{-6}$
6	$2.7\ 10^{-6}$	$2.7\ 10^{-6}$	$2.7\ 10^{-6}$	$2.7\ 10^{-6}$	$2.7\ 10^{-6}$	$2.7\ 10^{-6}$
7	$9.1\ 10^{-7}$	$9.1\ 10^{-7}$	$9.1\ 10^{-7}$	$9.1\ 10^{-7}$	$9.1\ 10^{-7}$	$9.1\ 10^{-7}$
8	$3.2\ 10^{-7}$	$3.2\ 10^{-7}$	$3.2\ 10^{-7}$	$3.2\ 10^{-7}$	$3.2\ 10^{-7}$	$3.2\ 10^{-7}$
9	$1.2\ 10^{-7}$	$1.2\ 10^{-7}$	$1.2\ 10^{-7}$	$1.2\ 10^{-7}$	$1.2\ 10^{-7}$	$1.2\ 10^{-7}$
10	$4.4\ 10^{-8}$	$4.4\ 10^{-8}$	$4.4\ 10^{-8}$	$4.4\ 10^{-8}$	$4.4\ 10^{-8}$	$4.4\ 10^{-8}$
11	$1.7\ 10^{-8}$	$1.7\ 10^{-8}$	$1.7\ 10^{-8}$	$1.7\ 10^{-8}$	$1.7\ 10^{-8}$	$1.7\ 10^{-8}$
12	$6.6\ 10^{-9}$	$9.7\ 10^{-9}$	$6.6\ 10^{-9}$	$6.6\ 10^{-9}$	$6.6\ 10^{-9}$	$6.6\ 10^{-9}$
13	$2.6\ 10^{-9}$	$1.5\ 10^{-7}$	$2.6\ 10^{-9}$	$2.6\ 10^{-9}$	$2.6\ 10^{-9}$	$2.6\ 10^{-9}$
14	$1.0\ 10^{-9}$	$5.8\ 10^{-6}$	$1.0\ 10^{-9}$	$1.0\ 10^{-9}$	$1.0\ 10^{-9}$	$1.0\ 10^{-9}$
15	$4.3\ 10^{-10}$	$1.8\ 10^{-4}$	$4.3\ 10^{-10}$	$4.3\ 10^{-10}$	$4.3\ 10^{-10}$	$4.3\ 10^{-10}$
16	$1.8\ 10^{-10}$	$2.4\ 10^{-3}$	$1.8\ 10^{-10}$	$1.8\ 10^{-10}$	$1.8\ 10^{-10}$	$1.8\ 10^{-10}$
17	$7.4\ 10^{-11}$	$1.8\ 10^{-1}$	$7.4\ 10^{-11}$	$7.4\ 10^{-11}$	$7.4\ 10^{-11}$	$7.4\ 10^{-11}$
18	$3.1\ 10^{-11}$	$2.0\ 10^{+1}$	0	$3.1\ 10^{-11}$	$3.1\ 10^{-11}$	$3.1\ 10^{-11}$
19	$1.3\ 10^{-11}$		0	$1.3\ 10^{-11}$	$1.3\ 10^{-11}$	$1.3\ 10^{-11}$
20	$5.6\ 10^{-12}$			$5.6\ 10^{-12}$	$5.6\ 10^{-12}$	$5.6\ 10^{-12}$
21	$2.4\ 10^{-12}$			$2.4\ 10^{-12}$	$2.4\ 10^{-12}$	$2.4\ 10^{-12}$
22	$1.0\ 10^{-12}$			$1.0\ 10^{-12}$	$1.0\ 10^{-12}$	$1.0\ 10^{-12}$
23	$4.6\ 10^{-13}$			0	$4.6\ 10^{-13}$	$4.6\ 10^{-13}$
24	$2.0\ 10^{-13}$			0	$2.0\ 10^{-13}$	$2.0\ 10^{-13}$
25	$8.8\ 10^{-14}$				$8.8\ 10^{-14}$	$8.8\ 10^{-14}$
26	$3.9\ 10^{-14}$				$3.9\ 10^{-14}$	$3.9\ 10^{-14}$
27	$1.7\ 10^{-14}$				$1.7\ 10^{-14}$	$1.7\ 10^{-14}$
28	$7.7\ 10^{-15}$				0	$7.7\ 10^{-15}$
29	$3.4\ 10^{-15}$				0	$3.4\ 10^{-15}$
30	$1.5\ 10^{-15}$					$1.5\ 10^{-15}$
sec CPU	2415	106	154	284	500	561

Nous observons que jusqu'à un certain point les valeurs de toutes les colonnes des tables coïncident. L'effet cumulatif des erreurs d'arrondi, présent dans le calcul approché, n'a pas encore affecté les chiffres corrects des résultats. Mais ceci va se produire d'autant plus vite que *cprec* est petit. Dans la dernière colonne des tables, en faisant du calcul approché avec précision suffisamment élevée, nous avons réussi à maîtriser l'effet accumulatif des erreurs d'arrondi, en obtenant les mêmes erreurs relatives que celles obtenues en faisant du calcul exact, mais en prennant beaucoup moins de temps CPU.

Les résultats de ces tableaux semblent montrer qu'étant donné les temps CPU, le calcul exact n'est intéressant que si l'on veut l'expression symbolique des approximants.

Si nous voulons placer dans les tables 1.5, 1.6, 1.7 et 1.8 les résultats obtenus en *Fortran* dans la section 1.2.2 correspondantes aux exemples 1 et 2, nous devons les situer immédiatement à gauche des colonnes étiquetées par "$N[-,19]$".

Dans les quatre tables que nous venons de citer, dans la colonne correspondante au calcul exact, nous avons mentionné la valeur de *cprec* utilisée. Quand *toption* == *"SCL"* et *coption* == *"EC"*, *cprec* est employée uniquement pour calculer le nombre de chiffres exacts ou les erreurs relatives des expressions numériques exactes des approximants par rapport à la série. Or, si *cprec* n'est pas suffisamment grand, ces valeurs risquent d'être mal calculées. D'un autre côté, une valeur trop grande de *cprec* augmente le temps de calcul du nombre de chiffres exacts ou des erreurs relatives. Pour trouver la valeur idéale de *cprec* adaptée au nombre d'approximants que nous voulons obtenir, nous avons calculé les erreurs relatives avec plusieurs valeurs de *cprec*, en commençant par la précision de la machine, et en augmentant *cprec* de cinq ou dix unités à chaque fois. Quand nous obtenons deux colonnes consécutives égales, nous prenons comme valeur de *cprec* celle correspondante à la colonne de gauche.

Nous montrons la table 1.9, qui a permit de choisir la valeur de *cprec* utilisée dans la table 1.5.

A titre comparatif, nous avons écrit un logiciel en *Mathematica* qui calcule une suite d'approximants de type Padé généralisés en utilisant l'implémentation récursive des relations (X), (L) et (K), comme nous l'avons expliqué dans la sous-section 1.3.1. A chaque relation correspond une définition *Mathematica* récursive qui garde automatiquement dans la mémoire les éléments qu'elle calcule.

Ce logiciel est constitué par trois packages: BRR.M, BRE.M et BRIF.M, que nous donnons dans l'annexe D.

BRR.M contient la définition de la fonction *tabgptrr* qui calcule une suite d'approximants de type Padé généralisés. Il appelle les packages BRE.M et BRIF.M. qui contiennent les mêmes définitions que les packages BE.M et BIF.M, mais avec la différence que les définitions des moments modifiés et des fonctionnelles initialles gardent automatiquement dans la mémoire les éléments qu'elles calculent.

Donnons quelques explications sur la fonction *tabgptrr*.

tabgptrr[k, t, toption, roption, psoption, resultprec, preoption,
preprec, preresprec, aroption ,compoption ,norm]

calcule les $k+1$ premiers approximants de type Padé généralisés au point t.

Si t est une expression symbolique, *toption* == "VAR"; si t est une expression numérique,

Tableau 1.7: Exemple 2 ($t = 1/10$)

n	exact cprec=60	N[-,19]	N[-,30]	N[-,40]	N[-,50]
0	$2.4\ 10^{-2}$	$2.4\ 10^{-2}$	$2.4\ 10^{-2}$	$2.4\ 10^{-2}$	$2.4\ 10^{-2}$
1	$5.1\ 10^{-4}$	$5.1\ 10^{-4}$	$5.1\ 10^{-4}$	$5.1\ 10^{-4}$	$5.1\ 10^{-4}$
2	$9.3\ 10^{-6}$	$9.3\ 10^{-6}$	$9.3\ 10^{-6}$	$9.3\ 10^{-6}$	$9.3\ 10^{-6}$
3	$1.5\ 10^{-7}$	$1.5\ 10^{-7}$	$1.5\ 10^{-7}$	$1.5\ 10^{-7}$	$1.5\ 10^{-7}$
4	$2.0\ 10^{-9}$	$2.0\ 10^{-9}$	$2.0\ 10^{-9}$	$2.0\ 10^{-9}$	$2.0\ 10^{-9}$
5	$2.4\ 10^{-11}$	$2.4\ 10^{-11}$	$2.4\ 10^{-11}$	$2.4\ 10^{-11}$	$2.4\ 10^{-11}$
6	$2.6\ 10^{-13}$	$2.5\ 10^{-13}$	$2.6\ 10^{-13}$	$2.6\ 10^{-13}$	$2.6\ 10^{-13}$
7	$2.5\ 10^{-15}$	$1.4\ 10^{-14}$	$2.5\ 10^{-15}$	$2.5\ 10^{-15}$	$2.5\ 10^{-15}$
8	$2.2\ 10^{-17}$	$3.8\ 10^{-14}$	$2.2\ 10^{-17}$	$2.2\ 10^{-17}$	$2.2\ 10^{-17}$
9	$1.8\ 10^{-19}$	$1.4\ 10^{-13}$	$1.8\ 10^{-19}$	$1.8\ 10^{-19}$	$1.8\ 10^{-19}$
10	$1.4\ 10^{-21}$	$3.2\ 10^{-13}$	$1.8\ 10^{-21}$	$1.4\ 10^{-21}$	$1.4\ 10^{-21}$
11	$9.7\ 10^{-24}$	$1.4\ 10^{-13}$	0	$9.7\ 10^{-24}$	$9.7\ 10^{-24}$
12	$6.4\ 10^{-26}$	$2.6\ 10^{-12}$	0	$6.4\ 10^{-26}$	$6.4\ 10^{-26}$
13	$3.9\ 10^{-28}$	$1.9\ 10^{-12}$	0	$3.9\ 10^{-28}$	$3.9\ 10^{-28}$
14	$2.3\ 10^{-30}$	$8.0\ 10^{-11}$	0	$7.1\ 10^{-29}$	$2.3\ 10^{-30}$
15	$1.3\ 10^{-32}$	$5.0\ 10^{-10}$	0	0	$1.3\ 10^{-32}$
sec CPU	46118	117	131	136	158

Tableau 1.8: Exemple 2 ($t = 5/10$)

n	exact cprec=40	N[-,19]	N[-,30]	N[-,40]
0	$1.0\ 10^{-1}$	$1.0\ 10^{-1}$	$1.0\ 10^{-1}$	$1.0\ 10^{-1}$
1	$9.5\ 10^{-3}$	$9.5\ 10^{-3}$	$9.5\ 10^{-3}$	$9.5\ 10^{-3}$
2	$7.8\ 10^{-4}$	$7.8\ 10^{-4}$	$7.8\ 10^{-4}$	$7.8\ 10^{-4}$
3	$5.6\ 10^{-5}$	$5.6\ 10^{-5}$	$5.6\ 10^{-5}$	$5.6\ 10^{-5}$
4	$3.5\ 10^{-6}$	$3.5\ 10^{-6}$	$3.5\ 10^{-6}$	$3.5\ 10^{-6}$
5	$2.0\ 10^{-7}$	$2.0\ 10^{-7}$	$2.0\ 10^{-7}$	$2.0\ 10^{-7}$
6	$1.0\ 10^{-8}$	$1.0\ 10^{-8}$	$1.0\ 10^{-8}$	$1.0\ 10^{-8}$
7	$4.7\ 10^{-10}$	$4.7\ 10^{-10}$	$4.7\ 10^{-10}$	$4.7\ 10^{-10}$
8	$2.0\ 10^{-11}$	$2.0\ 10^{-11}$	$2.0\ 10^{-11}$	$2.0\ 10^{-11}$
9	$7.9\ 10^{-13}$	$9.3\ 10^{-13}$	$7.9\ 10^{-13}$	$7.9\ 10^{-13}$
10	$2.9\ 10^{-14}$	$3.5\ 10^{-13}$	$2.9\ 10^{-14}$	$2.9\ 10^{-14}$
11	$9.8\ 10^{-16}$	$3.2\ 10^{-13}$	$9.8\ 10^{-16}$	$9.8\ 10^{-16}$
12	$3.1\ 10^{-17}$	$2.6\ 10^{-12}$	$3.1\ 10^{-17}$	$3.1\ 10^{-17}$
13	$9.4\ 10^{-19}$	$1.7\ 10^{-11}$	$9.4\ 10^{-19}$	$9.4\ 10^{-19}$
14	$2.7\ 10^{-20}$	$7.1\ 10^{-11}$	$3.1\ 10^{-19}$	$2.7\ 10^{-20}$
15	$7.1\ 10^{-22}$	$2.5\ 10^{-10}$	$6.1\ 10^{-19}$	$7.1\ 10^{-20}$
sec CPU	34301	131	150	155

Tableau 1.9: Exemple 1 (t=1/10). Choix de la valeur de *cprec*.

n	exact cprec=19	exact cprec=25	exact cprec=30	exact cprec=40
0	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$	$4.6\ 10^{-2}$
1	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$	$7.6\ 10^{-4}$
2	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$	$2.4\ 10^{-5}$
3	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$	$9.2\ 10^{-8}$
4	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$	$2.5\ 10^{-8}$
5	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$	$1.3\ 10^{-9}$
6	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$	$9.4\ 10^{-11}$
7	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$	$6.7\ 10^{-12}$
8	$4.9\ 10^{-13}$	$4.9\ 10^{-13}$	$4.9\ 10^{-13}$	$4.9\ 10^{-13}$
9	$3.7\ 10^{-14}$	$3.7\ 10^{-14}$	$3.7\ 10^{-14}$	$3.7\ 10^{-14}$
10	$2.9\ 10^{-15}$	$2.9\ 10^{-15}$	$2.9\ 10^{-15}$	$2.9\ 10^{-15}$
11	$2.3\ 10^{-16}$	$2.3\ 10^{-16}$	$2.3\ 10^{-16}$	$2.3\ 10^{-16}$
12	$1.8\ 10^{-17}$	$1.9\ 10^{-17}$	$1.9\ 10^{-17}$	$1.9\ 10^{-17}$
13	$2.2\ 10^{-18}$	$1.5\ 10^{-18}$	$1.5\ 10^{-18}$	$1.5\ 10^{-18}$
14	$5.1\ 10^{-19}$	$1.3\ 10^{-19}$	$1.3\ 10^{-19}$	$1.3\ 10^{-19}$
15	$6.3\ 10^{-19}$	$1.1\ 10^{-20}$	$1.1\ 10^{-20}$	$1.1\ 10^{-20}$
16	$5.7\ 10^{-19}$	$8.9\ 10^{-22}$	$8.9\ 10^{-22}$	$8.9\ 10^{-22}$
17	$6.3\ 10^{-19}$	$7.6\ 10^{-23}$	$7.6\ 10^{-23}$	$7.6\ 10^{-23}$
18	$6.3\ 10^{-19}$	$6.5\ 10^{-24}$	$6.5\ 10^{-24}$	$6.5\ 10^{-24}$
19	$6.3\ 10^{-19}$	0	$5.7\ 10^{-25}$	$5.7\ 10^{-25}$
sec CPU	294	297	297	299

toption == "SCL". Si nous attendons des résultats exacts, *coption* =="ER"; si nous attendons des résultats approchés, *coption* == "AR".

Si *roption* == "AR", les résultats numériques ou la partie numérique des résultats symboliques sont écrits avec *resultprec* chiffres significatifs.

Si *psoption* == 1, les sommes partielles de la série sont aussi calculées. Si *psoption* est différent de 1, les sommes partielles ne sont pas calculées.

Si *toption* == "SCL" et *preoption* == 1, les erreurs rélatives des approximants par rapport à la série sont calculées avec précision *preprec* et sont écrites avec *preresprec* chiffres significatifs. En outre, si *psoption* == 1, les erreurs rélatives des sommes partielles par rapport à la série sont calculées avec précision *preprec* et sont écrites avec *preresprec* chiffres significatifs. Si *toption* == "SCL" et *preoption* == 2, nous calculons le nombre de chiffres exacts à la place des erreurs rélatives.

Si *compoption* == 0, nous faisons du calcul réel. Si *compoption* == 1, nous faisons du calcul complex.

Si *compoption* == 1, *toption* == *"SCL"* et *preoption* == 1, les erreurs rélatives des parties réelles et des parties imaginaires des approximants, par rapport aux parties réelles et imaginaires de la série sont calculées avec précision *preprec* et sont écrites avec *preresprec* chiffres significatives. Si *psoption* == 1, nous faisons de même pour les sommes partielles.

Si *compoption* == 1, *toption* == *"SCL"*, mais *preoption* == 2, nous calculons le nombre de chiffres exacts à la place des erreurs rélatives.

Si *compoption* == 1 et *toption* == *"SCL"* , nous calculons aussi la norme p ($p >= 1$) des vecteurs des erreurs rélatives (si *preoption* == 1), ou la norme p des vecteurs du nombre de chiffres exacts (si *preoption* == 2). La valeur de p est donnée par l'argument *norm* de la fonction *tabgptrr*.

Si *norm* == -1, nous calculons la norme infinie des vecteurs des erreurs rélatives (si *preoption* == 1), ou la composante de valeur absolue minimalle des vecteurs du nombre de chiffres exacts (si *preoption* == 2).

Les normes sont calculées avec précision *preprec* et sont écrites avec *preresprec* chiffres significatives.

Si *toption* == "VAR" ou *preoption* est différent de 1 et de 2, ni les erreurs rélatives, ni le nombre de chiffres exacts sont calculés.

Si *toption* == "SCL" et *roption* == "ER", nous devons obtenir les expressions numériques exactes des approximants au point t; si *aroption* == 1, nous écrivons ces expressions exactes avec *resultprec* chiffres significatifs. Si *aroption* est différente de 1, ces valeurs ne sont pas écrites.

Les quatre premiers arguments de la fonction *tabgptrr* sont obligatoires et les huit derniers arguments sont facultatifs. Si l'utilisateur ne donne pas de valeurs à *psoption*, *resultprec*, *preoption*, *preprec*, *preresprec*, *aroption*, *compoption* et *norm* ces arguments prennent automatiquement les valeurs 0, *Precision*[1.], 0, *Precision*[1.], 2, 0, 0 et -1 respectivement.

Le code de la fonction *tabgptrr* (voir package BRR.M dans l'annexe D) peut être schématisé de la façon suivante:

```
tabgptrr[ k, t, toption, roption, psoption, resultprec, preoption,
        preprec, preresprec, aroption, compoption, norm ]
If psoption == 1 then sp ← 0
For i= 0,...,k do
   w ← key[ 0, 0, i, functionalc, g[x,t], x, t ]
   If psoption == 1 then sp ← sp + c[i]f[i,t]
   write w
   If psoption == 1 then write sp
   If preoption == 1 then
        compute and write relative error of w
        If psoption == 1 then
        compute and write relative error of sp
   If preoption == 2 then
        compute and write number of exact digits of w
        If psoption == 1 then
        compute and write number of exact digits of sp
```

Les arguments *preprec*, *preresprec*, *compoption* et *norm* sont utilisés uniquement dans l'éventuel calcul des erreurs relatives ou du nombre de chiffres exacts.

Les arguments *toption*, *roption*, *resultprec* et *aroption* sont utilisés uniquement dans l'écriture de w et de sp pour choisir sous quelle forme les approximants et les sommes partielles doivent être écrits, et n'interviennent pas dans le calcul de ces objects. Contrairement à ce qui se passe avec la fonction *tablegpt*, avec *tabgptrr* nous ne pouvons pas contrôler la précision des calculs intermédiaires. Le type de calcul effectué est choisi automatiquement par *Mathematica* et dépend de la nature des définitions de *series*, g, c_i, u_i, f_i et *initialf* et de la valeur de l'argument t, c'est-à-dire le type de calcul varie selon que ces définitions et la valeur de t sont exactes, approchées de haute ou de basse précision.

Si nous appelons

tabgptrr[n,t,toption,roption,psoption,resultprec,preoption,
preprec,preresprec,aroption,compoption,norm],

sont calculés et gardés dans la mémoire:

- les éléments des tableaux $L_1,\ldots,L_{n+1}$ et $X_1,\ldots,X_{n+1}$ des figures 1.1 et 1.6 avec $i=j=0$.
- les éléments $x_l^{(m,n)}$ présents dans ces tableaux.
- $c_0,\ldots,c_n$.
- $(K_i^{(0,0)}(c,g(x,t)))_{i=0,\ldots,n}$.

Pour qu'on puisse évaluer en termes pratiques la différence d'efficacité existante entre les fonctions *tablegpt* et *tabgptrr*, nous avons mesuré le temps d'exécution, et l'espace

de mémoire dépensé dans les appels suivants, correspondants à l'exemple 1 de la section 1.2.2.

tablegpt[15, 1/10, "*SCL*", "*EC*", 0, 19, 1, 2, 30]	114 sec CPU	44 K bytes.
tabgptrr[15, 1/10, "*SCL*", "*ER*", 0, 19, 1, 30, 2]	8660 sec CPU	1540 K bytes.
tablegpt[15, 1/10, "*SCL*", "*AC*", 0, 19, 1, 2, 19]	64 sec CPU	28 K bytes.
tabgptrr[15, 0.1, "*SCL*", "*AR*", 0, 19, 1, 19, 2]	8661 sec CPU	1531 K bytes.

Quand $k = 19$, la fonction *tabgptrr* avant d'arriver à la fin des calculs épuise toute la mémoire disponible (environ 4100 K bytes), et provoque une erreur de système. Cependant, *tablegpt* termine rapidement les calculs.

tablegpt[19, 1/10, "*SCL*", "*EC*", 0, 19, 1, 2, 30]	296 sec CPU	119 K bytes.
tablegpt[19, 1/10, "*SCL*", "*AC*", 0, 19, 1, 2, 19]	120 sec CPU	37 K bytes.

1.3.5 Exemples

Commençons par donner l'expression formelle des quatre premiers approximants correspondants aux exemples de la section 1.2.2.

Exemple 1

$$fgpt[0] = 1/(1+t)$$
$$fgpt[1] = 2/(2+t)$$
$$fgpt[2] = (6+5t)/((2(1+t)(3+t))$$
$$fgpt[3] = (144+228t+108t^2+19t^3)/(6(1+t)(2+t)(3+t)(4+t))$$

Exemple 2

$$fgpt[0] = (2(Log[-1-t] - Log[-1-t/2]))/t$$
$$fgpt[1] = (Log[-1-t] + 8Log[-1-t/2] - 9Log[-1-t/3])/(2t)$$
$$fgpt[2] = (7Log[-1-t] - 24Log[-1-t/2] - 81Log[-1-t/3] - 64Log[-1-t/4])/(6t)$$
$$fgpt[3] = (23Log[-1-t] + 64Log[-1-t/2] - 486Log[-1-t/3] + 1024Log[-1-t/4] - 625Log[-1-t/5])/(24t)$$

Exemple 3

$$fgpt[0] = Log[1-t]/t$$
$$fgpt[1] = (-3Log[1-t] + 4Log[1-t/2])/(t)$$
$$fgpt[2] = (6Log[1-t] - 32Log[1-t/2] - 27Log[1-t/3])/t$$
$$fgpt[3] = (-10Log[1-t] + 160Log[1-t/2] - 405Log[1-t/3] + 256Log[1-t/4])/t$$

Exemple 4

$$fgpt[0] = (-t - 2Log[1-t] + 2tLog[1-t] + 2Log[1-t/2] - tLog[1-t/2])/t^2$$
$$fgpt[1] = (-2t + 17Log[1-t] - 17tLog[1-t] - 80Log[1-t/2] +$$
$$40tLog[1-t/2] + 63Log[1-t/3] - 21tLog[1-t/3]))/(2t^2)$$
$$fgpt[2] = (-6t - 133Log[1-t] + 133tLog[1-t] + 1968Log[1-t/2] -$$
$$-984tLog[1-t/2] - 4779Log[1-t/3] + 1593tLog[1-t/3] +$$
$$2944Log[1-t/4] - 736tLog[1-t/4])/(6t^2)$$
$$fgpt[3] = (-12t + 549Log[1-t] - 549tLog[1-t] - 22144Log[1-t/2] +$$
$$11072tLog[1-t/2] + 122472Log[1-t/3] - 40824tLog[1-t/3] -$$
$$202752Log[1-t/4] + 50688tLog[1-t/4] + 101875Log[1-t/5] -$$
$$20375tLog[1-t/5])/(12t^2)$$

Exemple 5

$$fgpt[0] = 1/(1+t)$$
$$fgpt[1] = 1/(1+t)$$
$$fgpt[2] = (6 + 5t - 2t^2)/((1+t)(2+t)(3+t))$$
$$fgpt[3] = (24 + 26t - 3t^2 - 4t^3)/((1+t)(2+t)(3+t)(4+t))$$

Exemple 6

$$fgpt[0] = (2(Log[-1-t] - Log[-1-t/2]))/t$$
$$fgpt[1] = (7Log[-1-t] - 16Log[-1-t/2] + 9Log[-1-t/3])/(2t)$$
$$fgpt[2] = (-7Log[-1-t] + 288Log[-1-t/2] - 729Log[-1-t/3] +$$
$$+448Log[-1-t/4])/(6t)$$
$$fgpt[3] = (-23Log[-1-t] + 992Log[-1-t/2] - 2106Log[-1-t/3] +$$
$$512Log[-1-t/4] + Log[-1-t/5])/(24t)$$

Exemple 7

$$fgpt[0] = E^t$$
$$fgpt[1] = 4E^{t/2} - 3E^t$$
$$fgpt[2] = 27E^{t/3} - 32E^{t/2} + 6E^t$$
$$fgpt[3] = 256E^{t/4} - 405E^{t/3} + 160E^{t/2} - 10E^t$$

Exemple 8

$$fgpt[0] = (2(-E^{t/2} + E^t))/t$$
$$fgpt[1] = (-63E^{t/3} + 80E^{t/2} - 17E^t)/(2t)$$
$$fgpt[2] = (-2944E^{t/4} + 4779E^{t/3} - 1968E^{t/2} + 133E^t)/(6t)$$
$$fgpt[3] = (-101875E^{t/5} + 202752E^{t/4} - 122472E^{t/3} +$$
$$22144E^{t/2} - 549E^t)/(12t).$$

Nous observons que ce n'est pas raisonable d'approcher une série par une fonction qui a une expression plus compliquée que la série elle même. Or, ceci est le cas de tous les exemples montrés sauf le premier et le cinquième, où les approximants sont de type Padé.

Vu que

$$(n)_f(t) = a_0^{(n)} L_0(G(x,t)) + \cdots + a_n^{(n)} L_n(G(x,t)),$$

nous devons choisir la suite des fonctionnelles $(L_i)_{i=0,1,\ldots}$ de sorte que les fonctions $F_i(t) = L_i(G(x,t)), i = 0,1,\ldots$ soient plus simples que la série que nous voulons approcher.

Le problème qui est derrière ces exemples est le suivant: étant donnés une série de fonctions, la fonction génératrice et la suite des moments correspondants, comment choisir des formes L_i qui produisent des approximants de type Padé généralisés qui convergent vers la série, dans un certain domaine du plan complexe, au moins plus vite que la suite des sommes partielles. Nous nous demandons aussi sous quelles conditions ces fonctionnelles existent. Est-ce qu'elles sont définies de façon unique ou non ? Saurions nous choisir les fonctionnelles qui produisent la convergence la plus rapide ?

Une autre question consiste à savoir dans quel domaine du plan complexe les approximants convergent: le domaine de convergence de la série, le domaine d'analiticité de la fonction, ou autre ?

Le premier pas pour aborder toutes ces questions est de trouver des exemples supplémentaires [4].

Si nous comparons les graphes de $f(t) = \log(1+t)/t$ et de $f(t) = \exp(-t)$ (voir figures 1.12 et 1.13), avec les graphes de $F_i(t) = L_i(1/(1-xt)) = 1/(1+\frac{t}{i+1})$, $i = 0,1,\ldots$ (voir figure 1.14), pour $t > -1$, nous constatons qu'ils ont tous la même forme. Ce fait peut peut-être nous aider à trouver d'autres exemples.

Remarquons que $F_i = F_i(t)$ admet un pôle pour $t = -(i+1)$, $i = 0,1,\ldots$. Donc, on ne doit pas envisager d'approcher une série par $(n)_f(t)$, pour $t \in]-\infty,-1]$.

La fonction $f(t) = 1/\sqrt{1+t}$, dont le graphe a la même forme que les graphes précédents, pour $t > -1$ (voir figure 1.15), est développable en série de puissances.

Considérons donc l'exemple 9:

Exemple 9
$f(t) = 1/\sqrt{1+t}$
$G(x,t) = 1/(1-xt)$
$c_0 = 1$
$c_i = \frac{(-1)^i(2i-1)!!}{(2i)!!}$
$u_i(x) = x^i$
$f_i(t) = t^i$
$L_i(f) = f(-\frac{1}{i+1})$
Les approximants sont de type Padé.

Ecrivons l'expression formelle des quatre premiers approximants :

$fgpt[0] = 1/(1+t)$
$fgpt[1] = 2/(2+t)$
$fgpt[2] = (24+32t-11t^2)/(4(1+t)(2+t)(3+t))$
$fgpt[3] = (96+152t+76t^2-15t^3)/(4(1+t)(2+t)(3+t)(4+t))$.

[4] Voir figures et tableaux à la fin de cette section.

D'après les tableaux 1.10 et 1.11, nous constatons que, dans ce cas, la suite des fonctionnelles $L_i(f) = f(-\frac{1}{i+1})$, $i = 0, 1, \ldots$ produit aussi des approximants qui convergent vers la série, pour des valeurs de $t \in]-1, 1]$. Nous remarquons que le nombre de chiffres exacts des approximants est toujours supérieur au nombre de chiffres exacts des sommes partielles. La vitesse de convergence diminue au fur et à mesure qu'on s'éloigne de l'origine.

Considérons maintenant une autre suite de fonctionnelles

$$L_i(f) = f(\frac{1}{i+1}), \; i = 0, 1, \ldots.$$

En appliquant ces fonctionnelles à la fonction génératrice des puissances de la variable x, nous obtenons la suite des fonctions

$$F_i(t) = L_i(\frac{1}{1-xt}) = \frac{1}{1-\frac{t}{i+1}}, \; i = 0, 1, \ldots,$$

dont les graphes sont dessinés dans la figure 1.16, pour $t < 1$.

Remarquons que $F_i = F_i(t)$ admet un pôle pour $t = i + 1$, $i = 0, 1, \ldots$. Donc, on ne doit pas envisager d'approcher une série par $(n)_f(t)$, pour des valeurs de $t \in [1, +\infty[$.

Considérons la fonction $f(t) = \exp(7t)$, dont le graphe (voir figure 1.17) a la même forme que les graphes de $F_i = F_i(t), i = 0, 1, \ldots$, pour $t < 1$. Cette fonction est développable en série de puissances.

Considérons donc l'exemple 10:

Exemple 10
$f(t) = \exp(7t)$
$G(x,t) = 1/(1-xt)$
$c_i = 7^i/i!$
$u_i(x) = x^i$
$f_i(t) = t^i$
$L_i(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé.

Ecrivons l'expression formelle des quatre premiers approximants :

$fgpt[0] = -(1/(-1+t))$
$fgpt[1] = (2+11t)/((-2+t)(-1+t))$
$fgpt[2] = (-6-31t-76t^2)/((-3+t)(-2+t)(-1+t))$
$fgpt[3] = (24+118t+273t^2-382t^3)/((-4+t)(-3+t)(-2+t)(-1+t))$.

D'après les tableaux 1.12 et 1.13, nous constatons que la suite des fonctionnelles $L_i(f) = f(\frac{1}{i+1}), i = 0, 1, \ldots$ produit, dans ce cas, des approximants qui convergent vers la série et plus rapidement que la suite des sommes partielles, pour des valeurs de $t \in [-1, 1[$.. La vitesse de convergence diminue au fur et à mesure qu'on s'éloigne de l'origine.

La fonction $f(t) = \exp(7t)$ est aussi développable en série de polynômes de Tchebichev (voir Paszkowski [14]). Ecrivons la fonction génératrice de ces polynômes

$$G(x,t) = \frac{1-xt}{1-2xt+x^2} = \sum_{i=0}^{\infty} x^i T_i(t) \; , \; |t| < 1.$$

En appliquant à $G = G(x,t)$ la suite des fonctionnelles $L_i(f) = f(1/(i+1)), i = 0,1,\ldots,$ nous obtenons la suite des fonctions

$$F_i(t) = L_i(\frac{1-xt}{1-2xt+x^2}) = \frac{1-\frac{t}{i+1}}{1-\frac{2t}{i+1}+\frac{1}{(i+1)^2}}, i = 0,1,\ldots,$$

dont les graphes sont dessinés dans la figure 1.18, pour $t < 1$. Nous voyons qu'ils ont la même forme que le graphe de $f(t) = exp(7t)$, pour $t < 1$.

Remarquons que $F_i = F_i(t)$ admet un pôle pour $t = \frac{i+1}{2} + \frac{1}{2(i+1)}, i = 0,1,\ldots.$ Donc, on ne doit pas envisager d'approcher une série par $(n)_f(t)$ pour $t \in [1,+\infty[$.

Considérons donc l'exemple 11:

Exemple 11
$f(t) = \exp(7t)$
$G(x,t) = (1-xt)/(1-2xt+x^2)$
$c_i = BesselI(0,7)$
$c_i = 2BesselI(i,7)$
$u_i(x) = x^i$
$f_i(t) = T_i(t)$
$L_i(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé généralisé.

Ecrivons l'expression formelle des trois premiers approximants :

$$fgpt[0] = BesselI[0,7]/2$$
$$fgpt[1] = (-11BesselI[0,7] + 4tBesselI[0,7] + 12BesselI[1,7])/(2(-5+4t))$$
$$fgpt[2] = (55BesselI[0,7] - 71tBesselI[0,7] + 12t^2BesselI[0,7] - 60BesselI[1,7] + \\ +144tBesselI[1,7] - 72tBesselI[2,7])/(2(-5+3t)(-5+4t))$$

D'après les tableaux 1.14 et 1.15, nous constatons que la suite des fonctionnelles produit, dans ce cas, des approximants qui convergent vers la série et plus rapidement que la suite des sommes partielles, pour des valeurs de $t \in [-1,1[$. La vitesse de convergence est pratiquement la même tout au long de l'intervalle $[-1,1]$.

Il y a une différence fondamentale entre les approximants de type Padé généralisés construits d'après une série de puissances et ceux construits d'après un développement en polynômes orthogonaux. Le dèveloppement en série de puissances, à partir de laquelle les approximants de la fonction sont construits, dépend de la valeur de la fonction et de ses dérivées en un seul point et converge plus vite quand on est proche de ce point. Tandis que le développement polynomial dépend des moyennes pondérées de la fonction le long de l'intervalle d'orthogonalité, et la vitesse de convergence est pratiquement la même tout au long de cet intervalle.

Ces propriétés des séries de puissances et des développements orthogonaux sont transmises aux approximants de type Padé généralisés construits d'après eux.

Si nous représentons les valeurs des tableaux 1.12, 1.13, 1.14 et 1.15 dans des graphes (voir figures 1.19, 1.20, 1.21 et 1.22), nous constatons dans ce cas particulier ce que nous venons de dire.

Dans chaque exemple, les approximants et les sommes partielles correspondantes ont le même type de comportement du point de vue de la précision des résultats, c'est-à-dire les graphes du nombre de chiffres exacts ont la même forme. Dans le cas de la série de puissances (exemple 10), le nombre de chiffres exacts croît des extrêmes de l'intervalle $[-1,1]$ vers l'origine, où les approximants aussi bien que la série donnent la valeur exacte de la fonction. Tandis que dans le cas du développement de Tchebichev (exemple 11), le nombre de chiffres exacts est pratiquement constant tout au long de l'intervalle.

Les résultats de l'exemple 10 sont plus précis dans les alentours de l'origine que les résultats de l'exemple 11. Par contre, dans la proximité des extrêmes de l'intervalle, la situation est inverse. Ce sont les résultats de l'exemple 11 qui sont plus précis que les résultats de l'exemple 10.

Considérons maintenant la série de Tchebichev suivante (voir Paszkowski [14])

$$\exp(qt)\cos(q\sqrt{1-t^2}) = \sum_{k=0}^{\infty} \frac{q^k}{k!} T_k(t), \tag{1.20}$$

qui converge pour tout $t \in [-1,1]$ et $q \in R$.

Faisons $q = 1$, et observons que le graphe de $f(t) = \exp(t)\cos(\sqrt{1-t^2})$ (voir figure 1.23) a la même forme que les graphes de $F_i = F_i(t)$, pour $t \in [-1,1[$ (revoir figure 1.18).

Considérons donc l'exemple 12:

Exemple 12
$f(t) = \exp(t)\cos(\sqrt{1-t^2})$
$G(x,t) = (1-xt)/(1-2xt+x^2)$
$c_i = 1/i!$
$u_i(x) = x^i$
$f_i(t) = T_i(t)$
$L_i(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé généralisé.

Nous omettons les résultats de cet exemple, parce que les approximants convergent vers la série, mais pas plus vite que les sommes partielles.

Notre idée est de ralentir la convergence de la suite des sommes partielles. Pour cela, il suffit de choisir, dans (1.20), une valeur de q de sorte que les coefficients $q^k/k!$ convergent moins vite vers 0. Prenons, par exemple $q = 4$, et considérons l'exemple suivant :

Exemple 13
$f(t) = \exp(4t)\cos(4\sqrt{1-t^2})$
$G(x,t) = (1-xt)/(1-2xt+x^2)$
$c_i = 4^i/i!$
$u_i(x) = x^i$
$f_i(t) = T_i(t)$
$L_i(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé généralisé.

Ecrivons l'expression formelle des quatre premiers approximants :

$fgpt[0] = 1/2$
$fgpt[1] = (13 + 4t)/(2(-5 + 4t))$
$fgpt[2] = (-65 - 71t + 12t^2)/(2(-5 + 3t)(-5 + 4t))$
$fgpt[3] = (1185 + 695t - 964t^2 + 96t^3)/(2(-5 + 3t)(-5 + 4t)(-17 + 8t))$.

D'après les tableaux 1.16 et 1.17, nous constatons que la suite des approximants converge vers la série et plus vite que la suite des sommes partielles, pour des valeurs de $t \in [-1, 1[$. Observons aussi les graphes des figures 1.24 et 1.25, où nous représentons les valeurs des tableaux précédents.

La surprise que renferme cet exemple est que le graphe de $f(t) = \exp(4t)\cos(4\sqrt{1-t^2})$ (voir figure 1.26) n'a pas la même forme que les graphes de $F_i = F_i(t)$ (revoir figure 1.18).

Considérons finalement la série de Laguerre suivante (voir Lebedev [12])

$$\exp(a)J_0(2\sqrt{at}) = \sum_{i=0}^{\infty} \frac{a^i}{i!} L_i^0(t), \tag{1.21}$$

qui converge pour tout $t > 0$ et $a > 0$. [5]

Ecrivons la fonction génératrice des polynômes de Laguerre:

$$G(x,t) = \frac{1}{1-x}\exp(-\frac{x}{1-x}t) = \sum_{i=0}^{\infty} x^i L_i^0(t).$$

En appliquant à $G = G(x,t)$ la suite des fonctionnelles $L_i(f) = f(1/(i+2))$, $i = 0, 1, \ldots$, nous obtenons la suite des fonctions:

$$F_i(t) = L_i(\frac{1}{1-x}\exp(-\frac{x}{1-x}t)) = \frac{i+2}{i+1}\exp(-\frac{1}{i+1}t)\,,\ i = 0, 1, \ldots,$$

dont les graphes sont dessinés dans la figure 1.27.

Remarquons que les fonctions $F_i = F_i(t)$ sont entières.

Dans (1.21), faisons $a = 1$, et considérons la fonction $f(t) = \exp(1)J_0(2\sqrt{t})$, dont le graphe est dessiné dans la figure 1.28. Les graphes de f et de F_i ne se ressemblent pas. De toute façon essayons l'exemple 14:

Exemple 14
$f(t) = \exp(1)J_0(2\sqrt{t})$
$G(x,t) = \frac{1}{1-x}\exp(-\frac{x}{1-x}t)$
$c_i = 1/i!$
$u_i(x) = x^i$
$f_i(t) = L_i(t)$
$L_i^0(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé généralisé.

Nous omettons les résultats de cet exemple, parce que les approximants convergent vers la série, mais pas plus vite que les sommes partielles. Nous sommes dans la même situation

[5] J_0 représente la fonction de Bessel d'ordre 0 et $\{L_i^0\}_{i=0,1,\ldots}$ la famille des polynômes de Laguerre.

que l'exemple 12. Augmentons dans (1.21) la valeur de a. Prenons, par exemple $a = 5$, et considérons l'exemple 15:

Exemple 15
$f(t) = \exp(5)J_0(2\sqrt{5t})$ $G(x,t) = \frac{1}{1-x}\exp(-\frac{x}{1-x}t)$ $c_i = 5^i/i!$ $u_i(x) = x^i$ $f_i(t) = L_i^0(t)$ $L_i(f) = f(\frac{1}{i+1})$
Les approximants sont de type Padé généralisé.

Ecrivons l'expression formelle des quatre premiers approximants :

$fgpt[0] = 2/Exp[t]$
$fgpt[1] = (112 - 81Exp[t/2])/(2Exp[t])$
$fgpt[2] = (928 - 1917Exp[t/2] + 1088Exp[2t/3])/(2Exp[t])$
$fgpt[3] = (15392 - 66798Exp[t/2] + 97792Exp[2t/3] - 45625Exp[3t/4])/(8Exp[t])$.

D'après le tableau 1.18, nous constatons que les approximants convergent vers la série et plus vite que les sommes partielles, pour des valeurs de $t > 0$. Observons aussi les graphes des figures 1.29, 1.30, 1.31, 1.32, 1.33, 1.34, 1.35, et 1.36, où nous représentons les résultats de cette exemple. Au fur et à mesure qu'on s'éloigne de l'origine, la vitesse de convergence diminue progressivement. Nous remarquons que, quand $t = 200$, au bout de 38 itérations, ni l'approximant, ni la somme partielle n'exhibe un seul chiffre exact. Pour terminer, nous dessinons dans la figure 1.37 le graphe de $f(t) = \exp(5)J_0(2\sqrt{5t})$, qui n'a pas du tout la même forme que les graphes de $F_i = F_i(t)$ (revoir la figure 1.27.)

Pour chaque exemple montré, nous avons aussi testé des valeurs de t complexes. Les résultats ont le même type de comportement que ceux obtenus pour les valeurs réelles de t. Plus précisément, pour t complexe appartenant au domaine de convergence de la série, les approximants correspondants convergent. Dans l'ensemble des exemples testés, nous n'avons jamais observé le cas, à savoir que la série diverge et la suite des approximants converge. Cette situation a lieu pour la série

$$\frac{\log(1+t)}{t} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i+1} t^i,$$

convergente dans le disque unité, tandis que les approximants de Padé convergent dans le plan coupé par la semi-droite réelle $]-\infty, -1]$.

Les résultats des deux dernières sections ont été obtenus avec le package GPADE-TYPE.M, donné dans l'annexe C, et le logiciel *Mathematica 2.0* dans un *Macintosh SE/30*.

Les résultats symboliques ont été obtenus en faisant du calcul exact.

Les résultats numériques ont été obtenus en faisant du calcul approché de haute précision avec une valeur suffisamment élevée de *cprec* de sorte que les erreurs d'arrondi n'entachent pas les chiffres exacts des résultats ($cprec == 65$).

Les résultats numériques ont été obtenus en faisant des appels comme suit :

$$tablegpt[40, 1/10, "SCL", "AC", 1, 35, 2, 2, 65],$$

où le premier argument varie selon le nombre maximum d'approximants à calculer et le deuxième est la valeur de t.

Nous rappelons que les définitions de la série, de la fonction génératrice, de la base des polynômes, des fonctions élémentaires, des fonctionnelles et de la valeur de t doivent être exactes, pour qu'il soit possible de faire du calcul exact et du calcul approché de précision arbitraire (voir les packages BE.M et BIF.M donnés dans l'annexe C) .

Figure 1.12: $f(t) = \log(1+t)/t$.

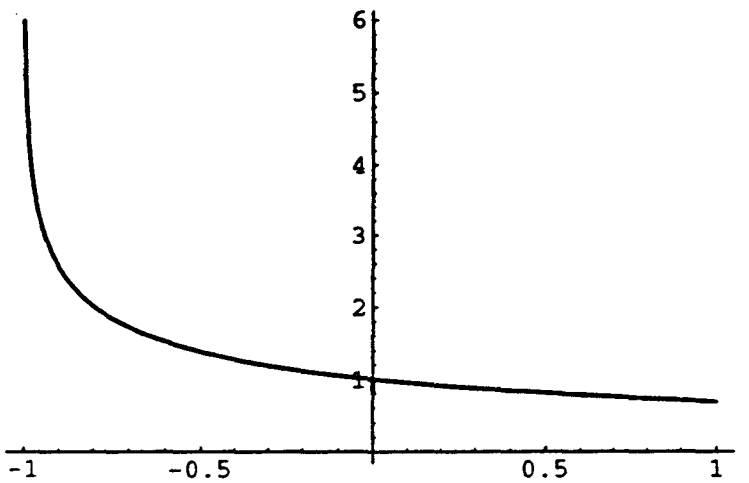


Figure 1.13: $f(t) = \exp(-t)$.

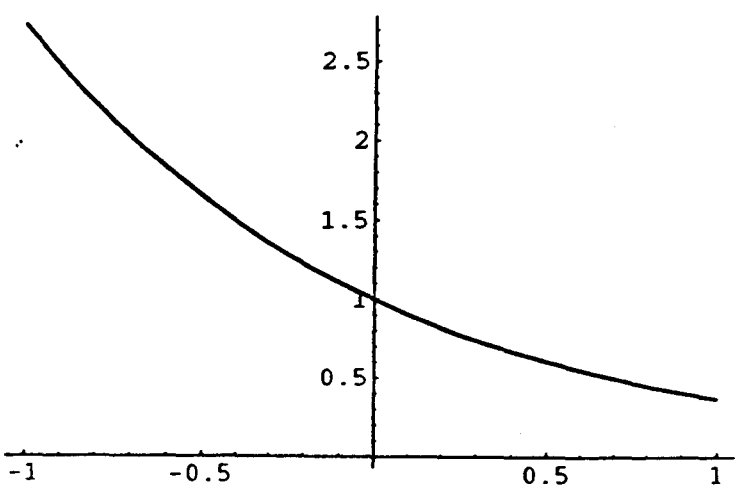


Figure 1.14: $F_i(t) = 1/(1 + \frac{t}{i+1}), i = 0, 1, 2, 3, 4, 5.$

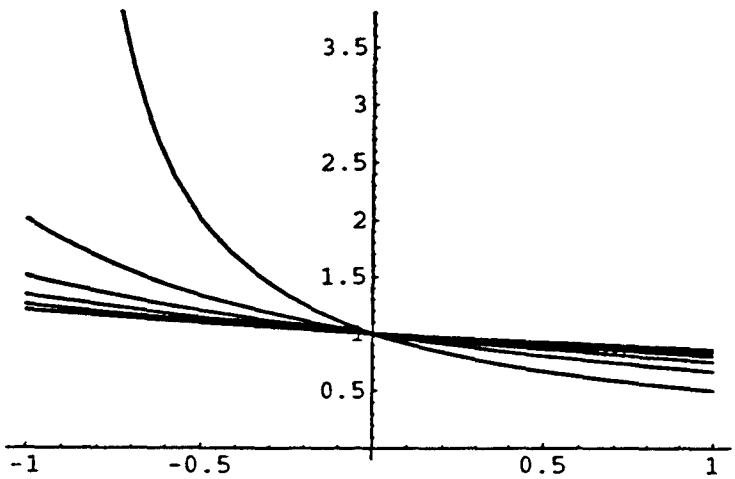


Figure 1.15: $f(t) = 1/\mathrm{sqrt}(1 + t)$.

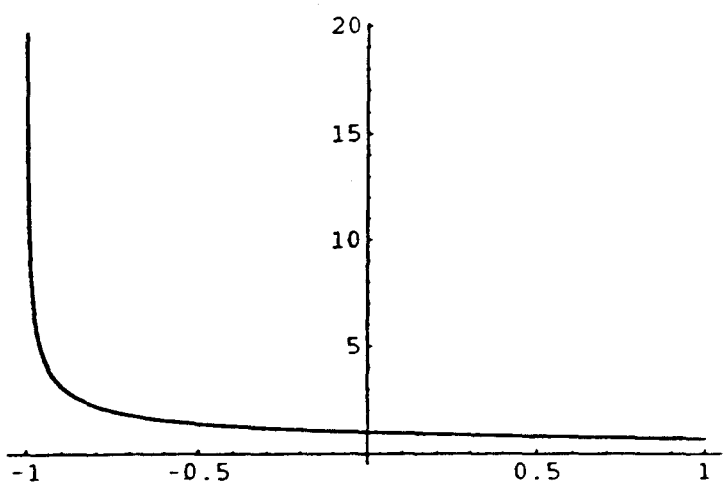


Figure 1.16: $F_i(t) = \frac{1}{1-\frac{t}{i+1}}, i = 0, 1, 2, 3, 4, 5$

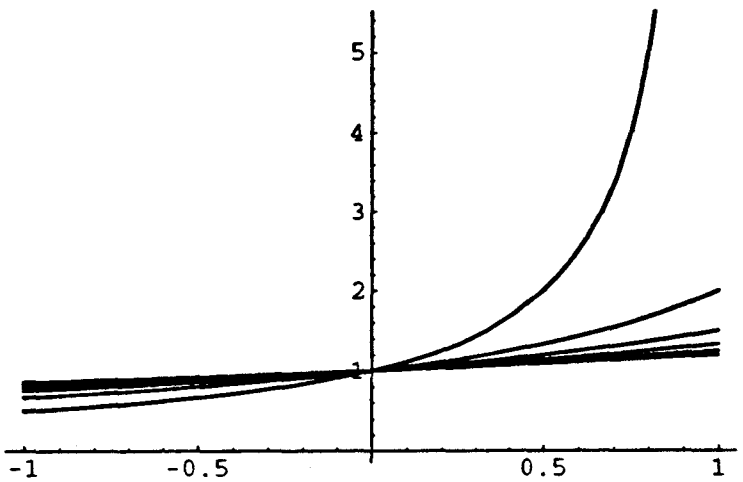


Figure 1.17: $f(t) = \exp(7t)$

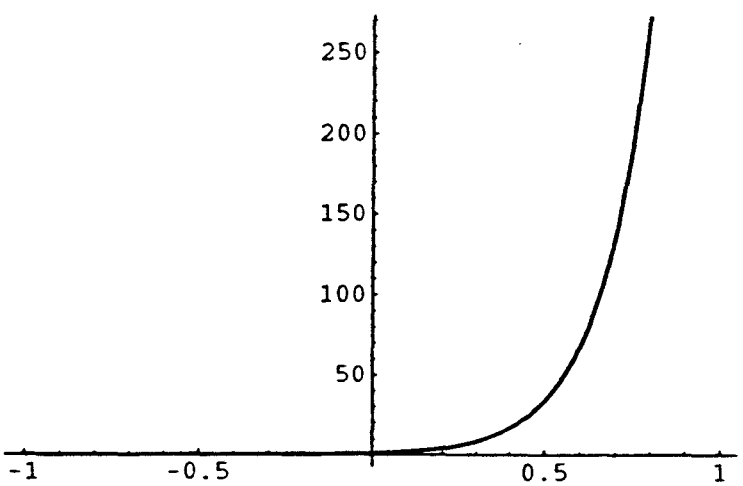


Figure 1.18: $F_i(t) = \frac{1-\frac{t}{i+1}}{1-\frac{2t}{i+1}+\frac{t}{(i+1)^2}}, i = 0, 1, 2, 3, 4, 5$

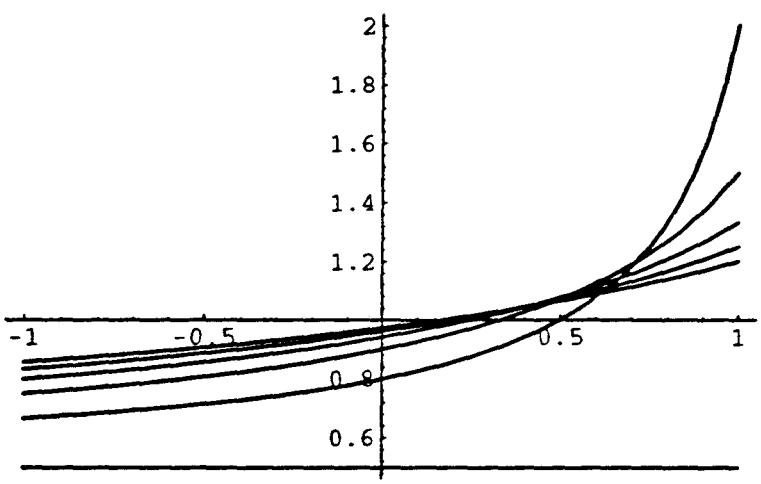


Figure 1.19 :

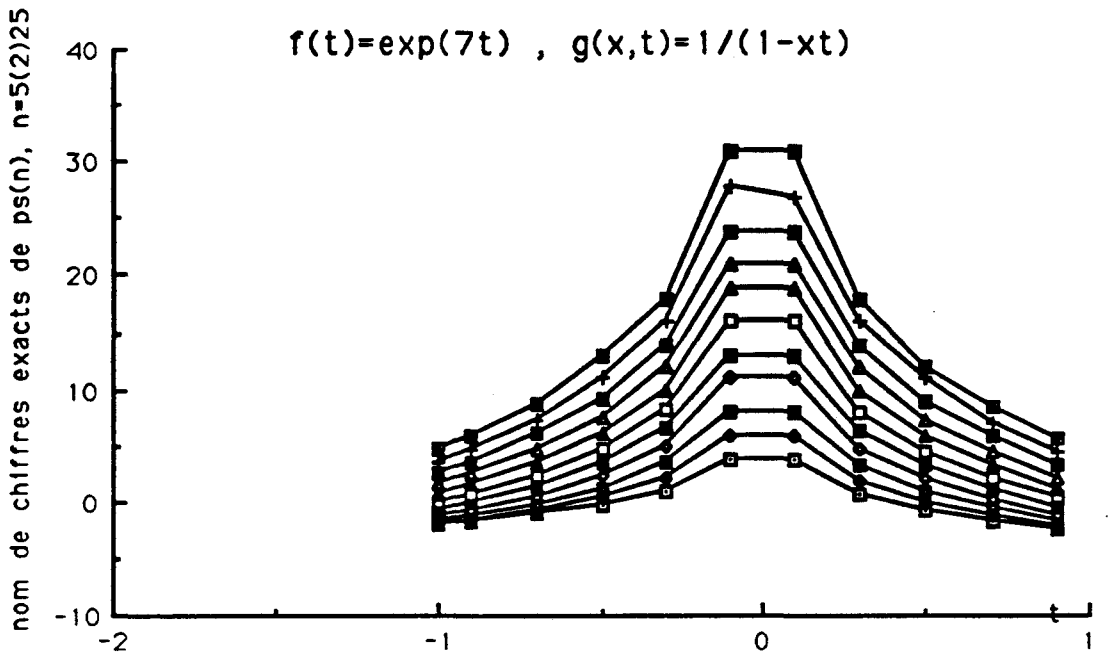


Figure 1.20 :

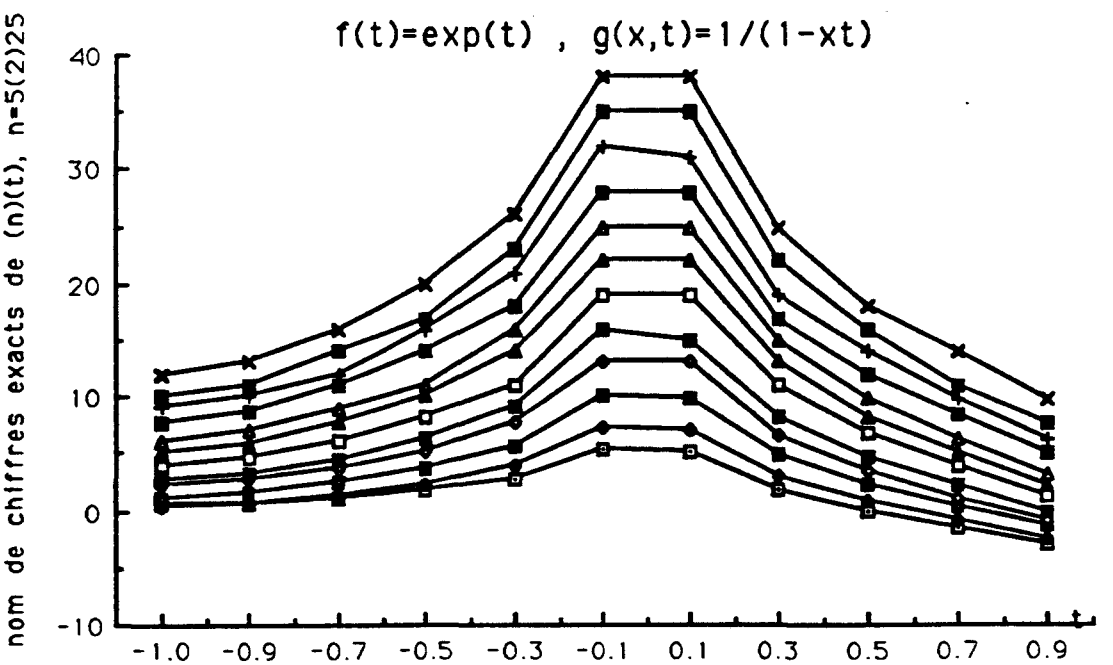


Figure 1.21 :

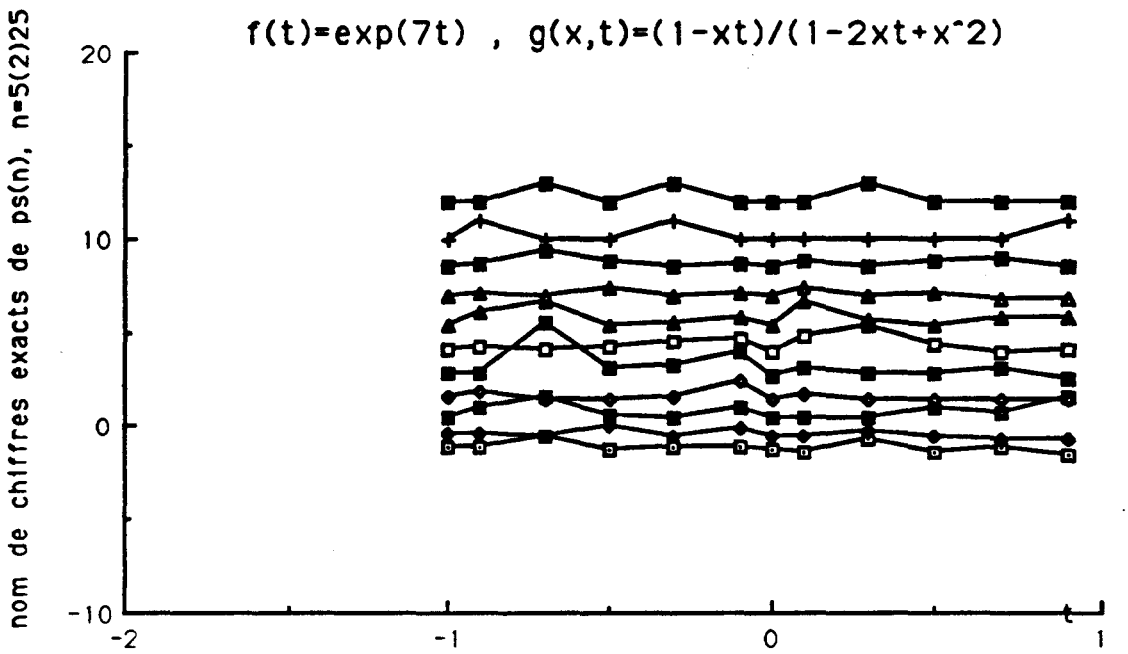


Figure 1.22 :

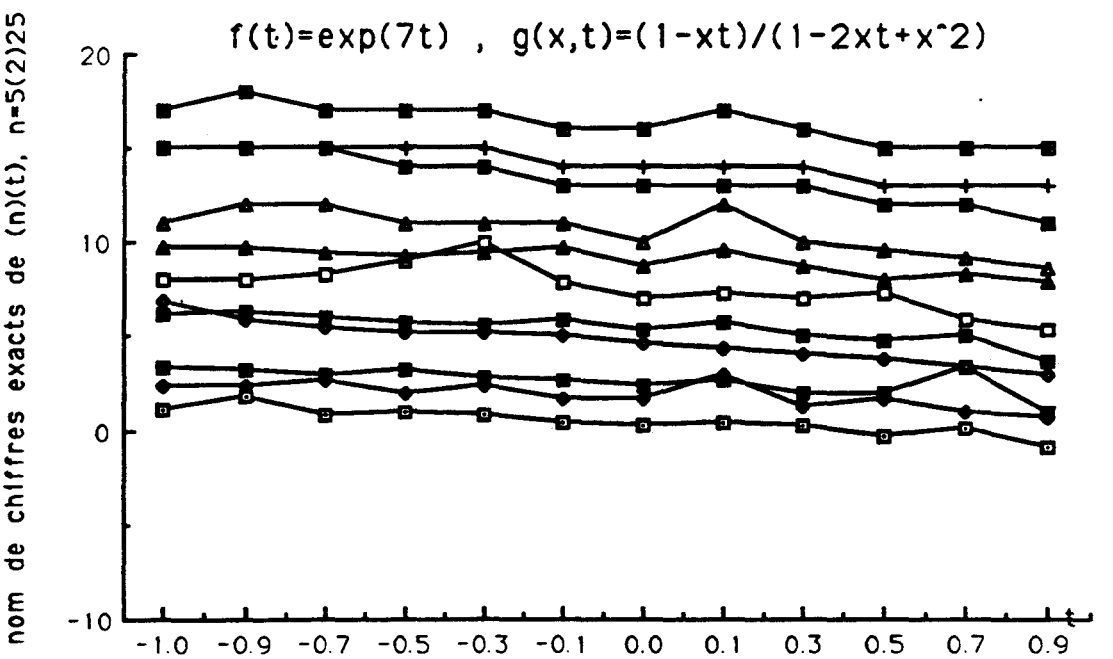


Figure 1.23: $f(t) = \exp(t)\cos(\text{sqrt}(1 - t^2))$

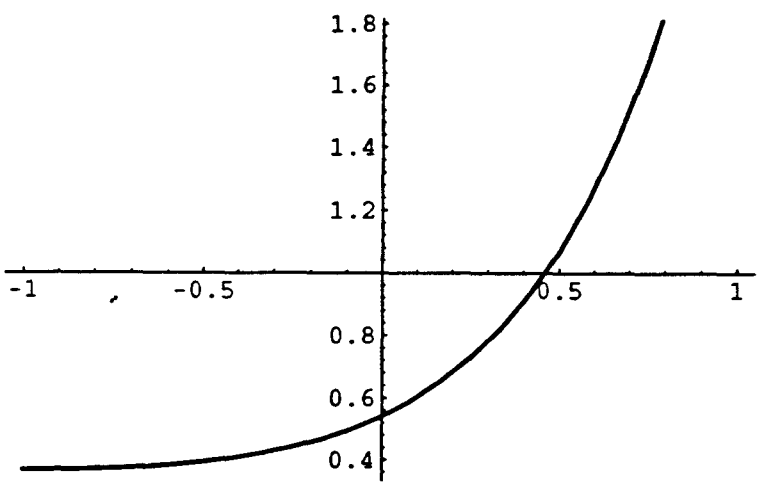


Figure 1.24 :

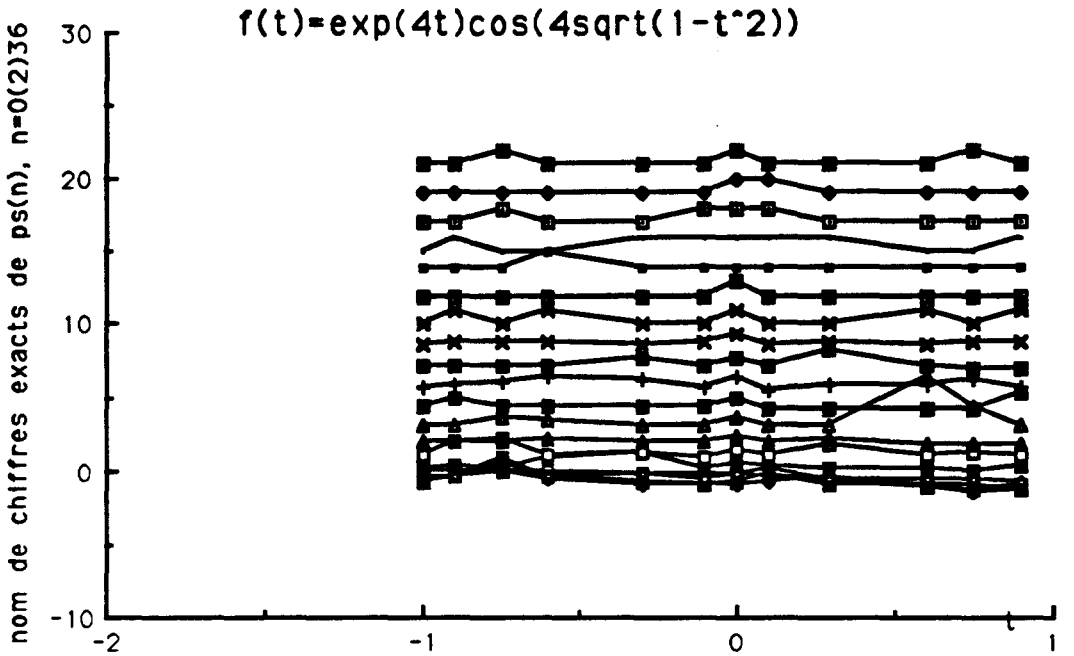


Figure 1.25 :

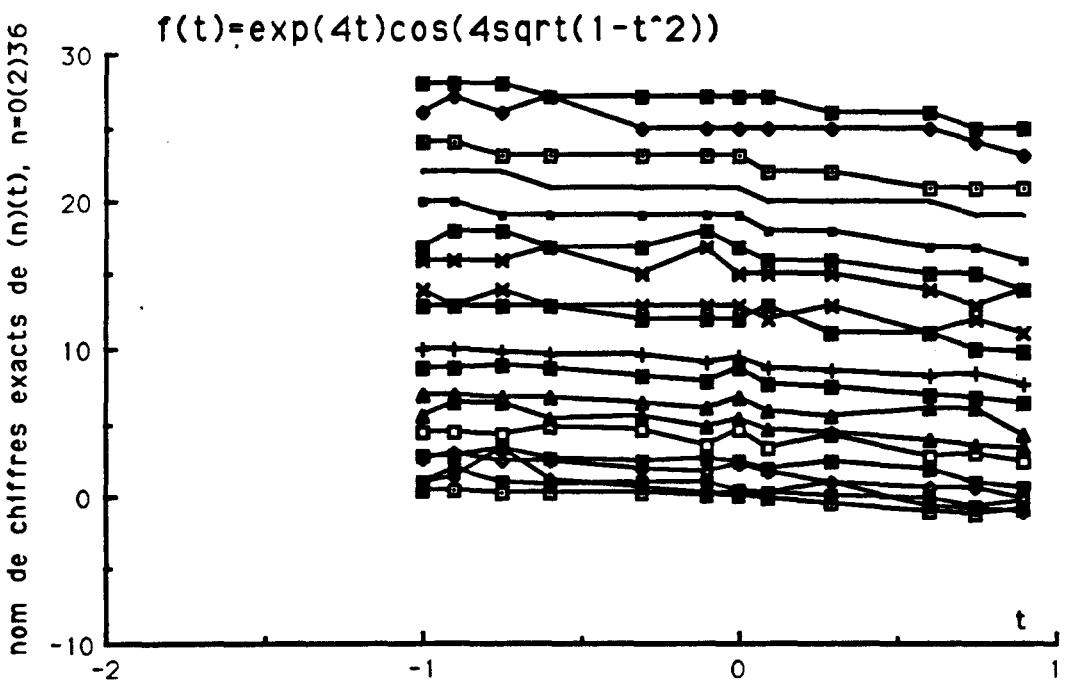


Figure 1.26: $f(t) = \exp(4t)\cos(4\text{sqrt}(1 - t^2))$

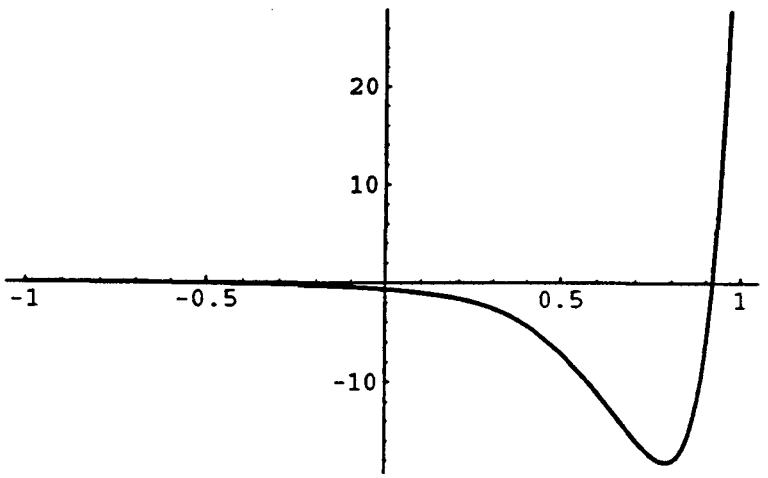


Figure 1.27: $F_i(t) = \frac{1}{1-\frac{1}{i+2}} \exp(-\frac{\frac{1}{i+2}}{1-\frac{1}{i+2}} t), i = 0, 1, 2, 3, 4, 5.$

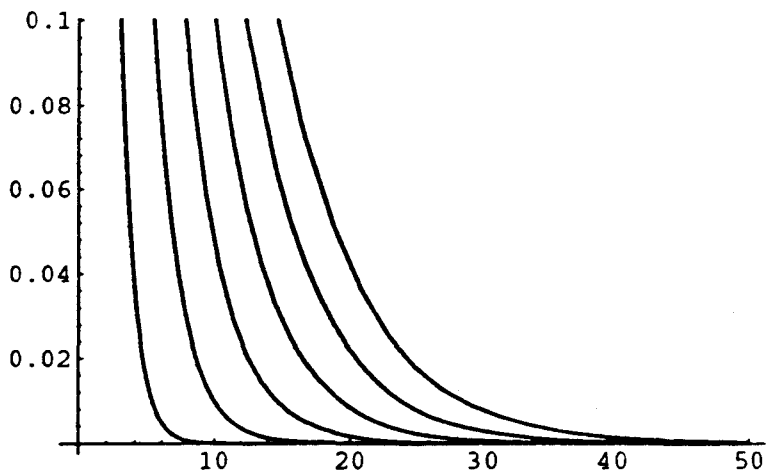


Figure 1.28: $f(t) = \exp(1)J_0(2\mathrm{sqrt}(t))$

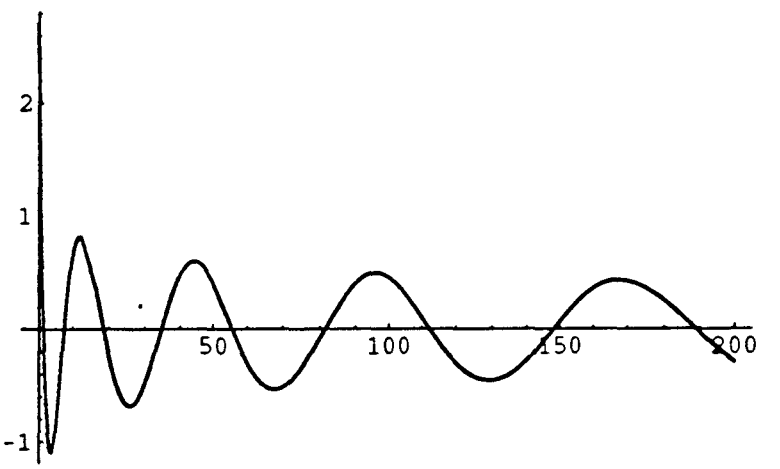


Figure 1.29 :

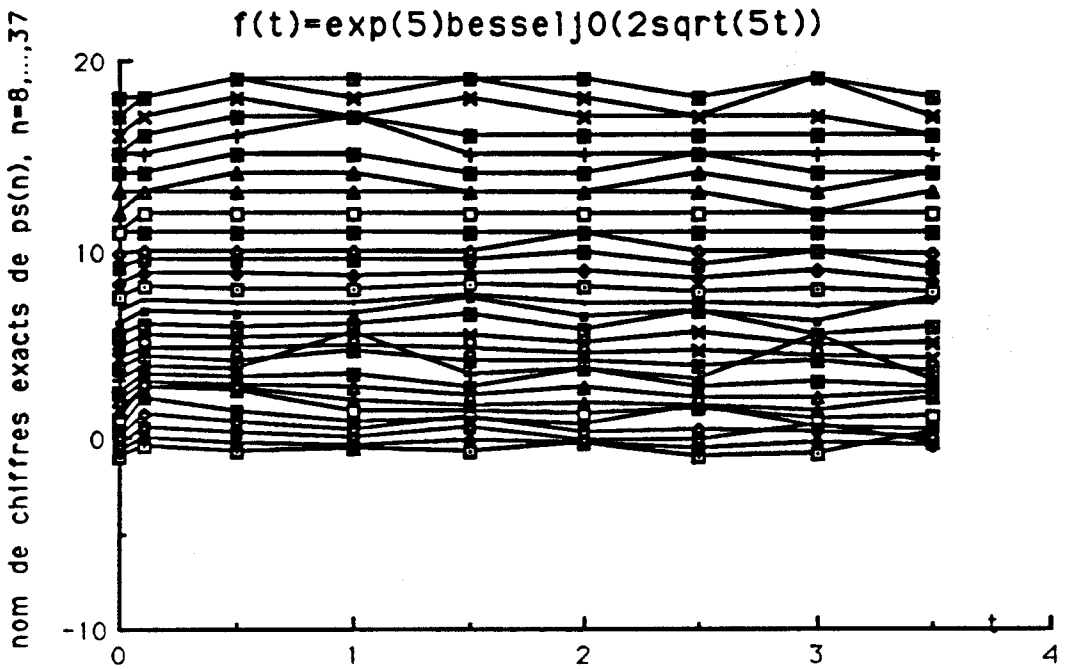


Figure 1.30 :

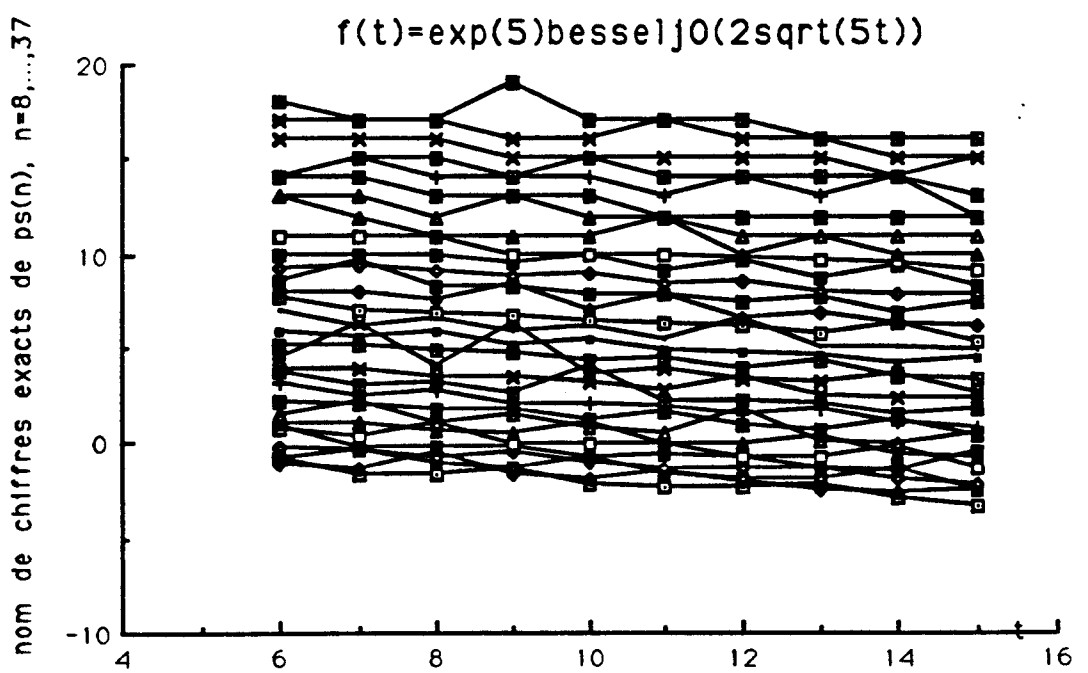


Figure 1.31 :

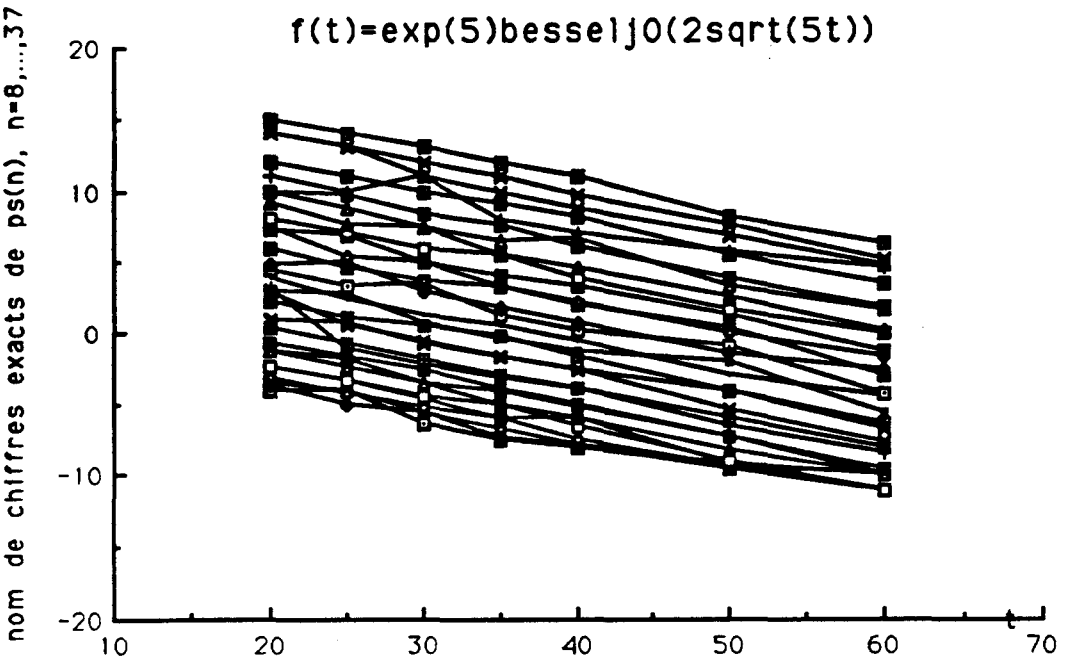


Figure 1.32 :

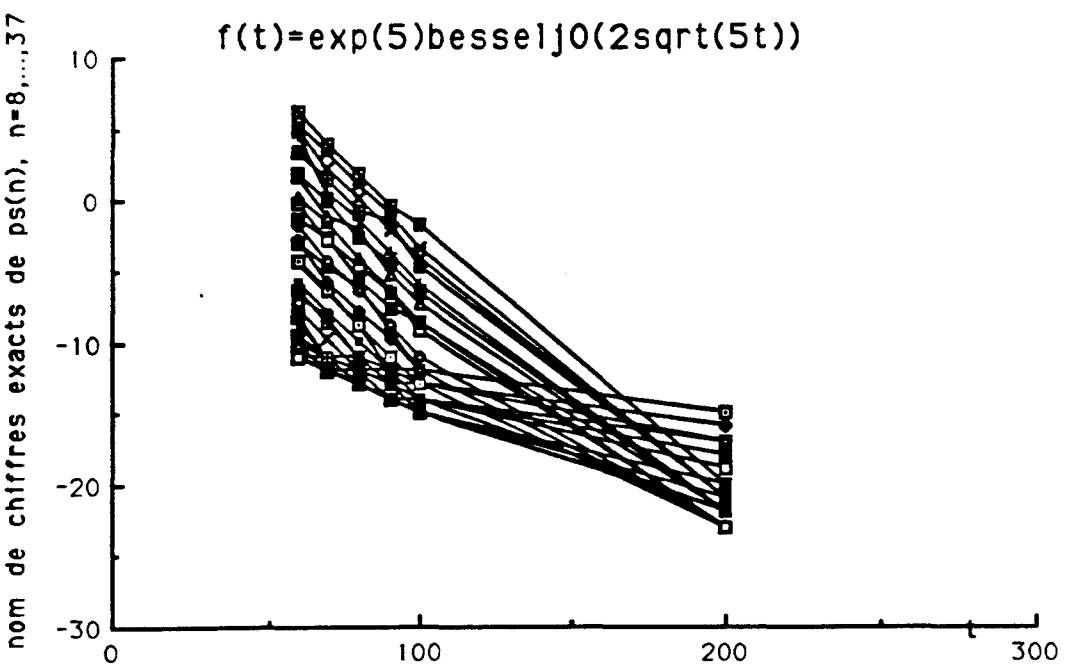


Figure 1.33 :

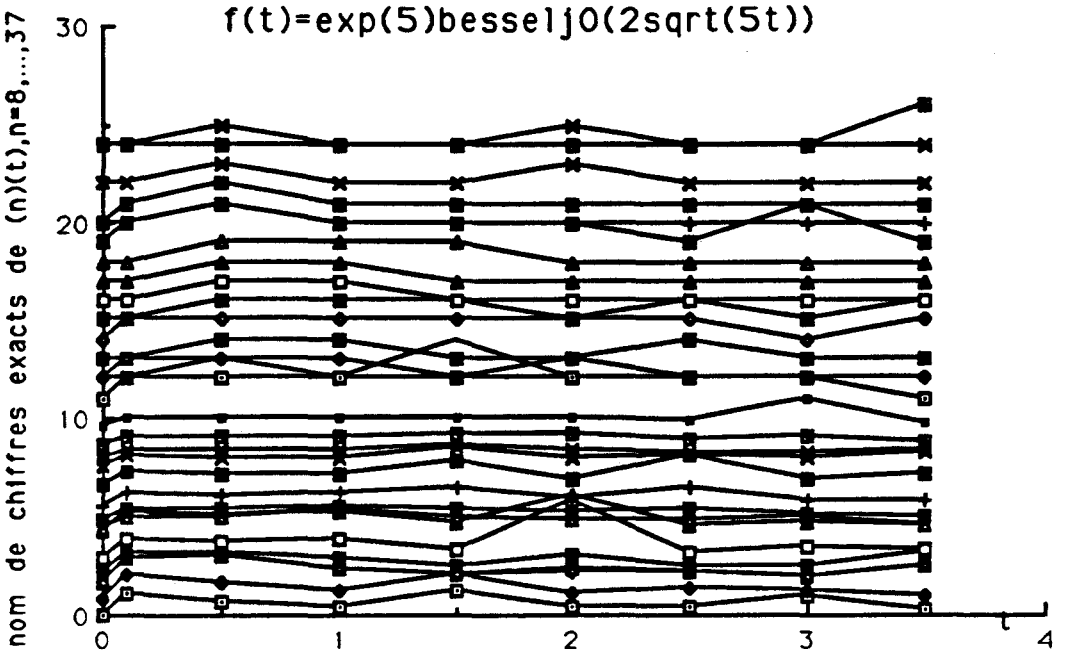


Figure 1.34 :

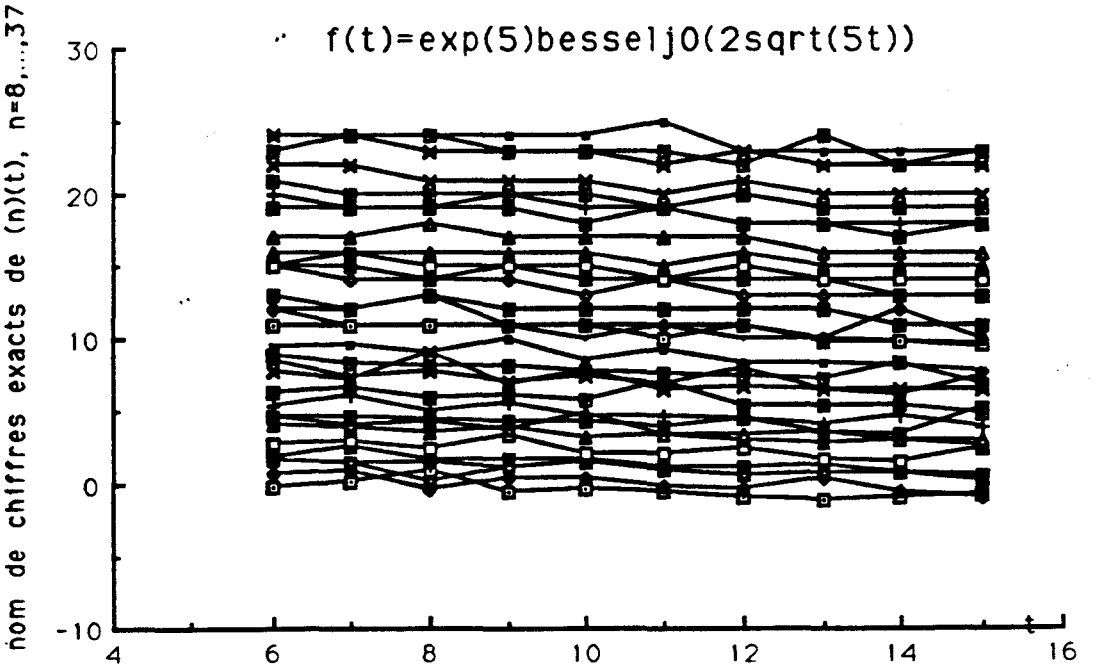


Figure 1.35 :

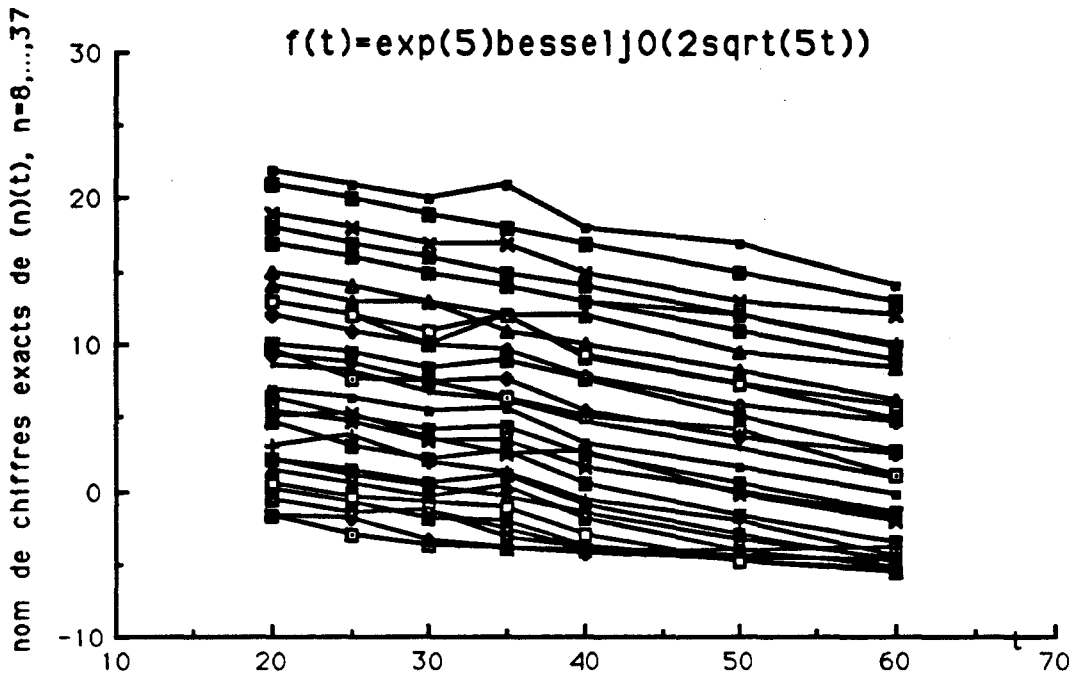


Figure 1.36 :

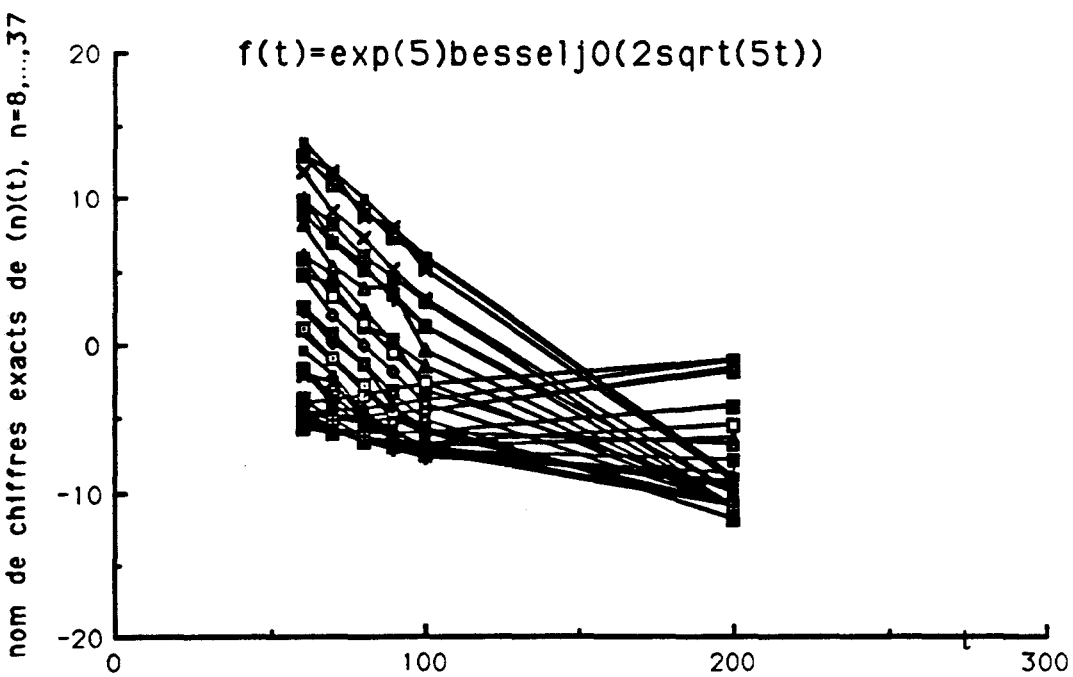


Figure 1.37: $f(t) = \exp(5)J_0(2\mathrm{sqrt}(5t))$

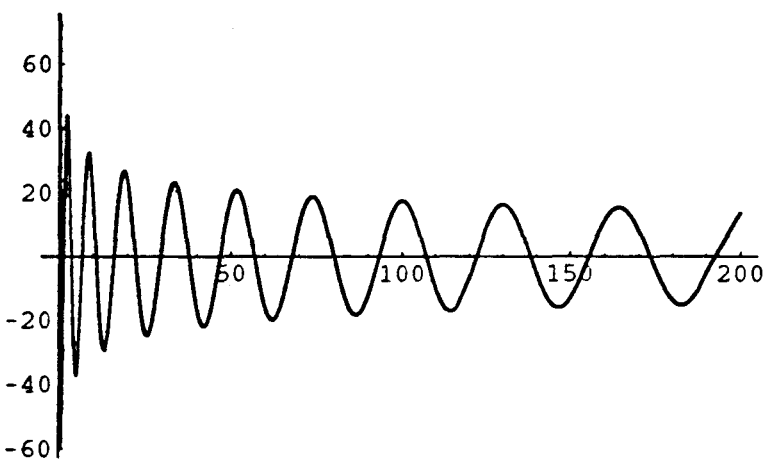


Tableau 1.10: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 20$.

$f(t) = 1/\sqrt{1+t}$

$G(x,t) = 1/(1-xt)$, $c_i = (-1)^i(2i-1)!!/(2i)!!$

$u_i(x) = x^i$, $f_i(t) = t^i$, $L_i(f) = f(-1/(i+1))$

	$t = -\frac{9}{10}$		$t = -\frac{7}{10}$		$t = -\frac{1}{2}$		$t = -\frac{3}{10}$		$t = -\frac{1}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	−.83	−.33	−.18	.083	.23	.38	.63	.71	1.2	1.3
1	−.13	−.23	.54	.32	1.1	.78	1.7	1.3	2.8	2.4
2	−.11	−.15	.91	.53	1.7	1.2	2.6	1.9	4.3	3.5
3	.57	−.072	2.2	.73	4.1	1.5	4.7	2.5	6.6	4.5
4	.47	−.001	1.9	.92	3.1	1.8	4.6	3.1	7.3	5.6
5	.64	.068	2.2	1.1	3.7	2.2	5.4	3.6	8.6	6.6
6	.74	.13	2.5	1.3	4.1	2.5	6.0	4.2	9.7	7.6
7	.85	.2	2.7	1.5	4.5	2.8	6.7	4.7	11.	8.7
8	.94	.26	2.9	1.6	4.8	3.2	7.3	5.3	12.	9.7
9	1.0	.32	3.2	1.8	5.2	3.5	7.9	5.8	13.	11.
10	1.1	.38	3.4	2.0	5.6	3.8	8.5	6.4	14.	12.
11	1.2	.44	3.6	2.2	6.0	4.1	9.1	6.9	15.	13.
12	1.3	.5	3.8	2.3	6.3	4.4	9.7	7.5	16.	14.
13	1.4	.55	4.0	2.5	6.7	4.8	10.	8.0	17.	15.
14	1.5	.61	4.2	2.7	7.0	5.1	11.	8.5	18.	16.
15	1.5	.66	4.4	2.8	7.4	5.4	11.	9.1	19.	17.
16	1.6	.72	4.6	3.0	7.7	5.7	12.	9.6	21.	18.
17	1.7	.77	4.8	3.2	8.1	6.0	13.	10.	22.	19.
18	1.8	.83	5.0	3.3	8.4	6.3	13.	11.	23.	20.
19	1.8	.88	5.1	3.5	8.8	6.6	14.	11.	24.	21.
20	1.9	.94	5.3	3.7	9.1	6.9	14.	12.	25.	22.

Tableau 1.11: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 20$.

$f(t) = 1/\sqrt{1+t}$

$G(x,t) = 1/(1-xt)$, $c_i = (-1)^i(2i-1)!!/(2i)!!$

$u_i(x) = x^i$, $f_i(t) = t^i$, $L_i(f) = f(-1/(i+1))$

	$t=\frac{1}{10}$		$t=\frac{3}{10}$		$t=\frac{1}{2}$		$t=\frac{7}{10}$		$t=\frac{9}{10}$		$t=1$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	1.4	1.3	.97	.91	.82	.74	.75	.63	.7	.56	.68	.53
1	3.	2.5	2.1	1.6	1.8	1.2	1.6	.93	1.4	.76	1.4	.68
2	4.5	3.5	3.2	2.2	2.7	1.6	2.3	1.2	2.1	.89	2.	.77
3	6.6	4.6	4.8	2.8	4.0	1.9	3.5	1.4	3.2	1.0	3.1	.84
4	7.5	5.6	5.4	3.3	4.5	2.3	3.9	1.6	3.5	1.1	3.4	.89
5	8.9	6.7	6.3	3.9	5.2	2.6	4.6	1.8	4.1	1.2	3.9	.93
6	10.	7.7	7.	4.4	5.6	3.0	4.8	2.	4.3	1.3	4.	.97
7	11.	8.7	7.6	5.0	6.1	3.3	5.2	2.2	4.5	1.3	4.2	1.
8	12.	9.8	8.3	5.5	6.5	3.6	5.5	2.3	4.7	1.4	4.4	1.
9	13.	11.	8.9	6.1	7.	3.9	5.8	2.5	4.9	1.5	4.5	1.
10	14.	12.	9.5	6.6	7.4	4.3	6.	2.7	5.1	1.5	4.7	1.1
11	16.	13.	10.	7.2	7.8	4.6	6.3	2.9	5.2	1.6	4.8	1.1
12	17.	14.	11.	7.7	8.2	4.9	6.5	3.	5.4	1.7	4.9	1.1
13	18.	15.	11.	8.3	8.6	5.2	6.8	3.2	5.5	1.7	5.	1.1
14	19.	16.	12.	8.8	9.	5.5	7.	3.4	5.7	1.8	5.1	1.1
15	20.	17.	13.	9.3	9.3	5.8	7.3	3.6	5.8	1.9	5.2	1.1
16	21.	18.	13.	9.9	9.7	6.2	7.5	3.7	5.9	1.9	5.3	1.2
17	22.	19.	14.	10.	10.	6.5	7.7	3.9	6.1	2.0	5.4	1.2
18	23.	20.	14.	11.	10.	6.8	8.	4.1	6.2	2.0	5.4	1.2
19	24.	21.	15.	11.	11.	7.1	8.2	4.2	6.3	2.1	5.5	1.2
20	25.	22.	15.	12.	11.	7.4	8.4	4.4	6.4	2.1	5.6	1.2

Tableau 1.12: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 26$.

$f(t) = \exp(7t)$

$G(x,t) = 1/(1-xt)$, $c_i = 7^i/i!$

$u_i(x) = x^i$, $f_i(t) = t^i$, $L_i(f) = f(1/(i+1))$

	$t=-1$		$t=-\frac{9}{10}$		$t=-\frac{7}{10}$		$t=-\frac{1}{2}$		$t=-\frac{3}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	.3	.0004	.28	.0008	.24	.0032	.2	.013	.19	.057
1	−.18	−.78	−.16	−.72	−.097	−.59	.016	−.4	.25	−.087
2	−.33	−1.3	−.27	−1.2	−.1	−.91	.16	−.56	.63	.0076
3	−.23	−1.6	−.12	−1.4	.15	−1.1	.56	−.55	1.3	.25
4	.22	−1.8	.38	−1.6	.78	−1.1	1.4	−.43	2.3	.6
5	.56	−1.9	.72	−1.6	1.1	−1.	1.7	−.22	2.8	1.
6	.25	−1.9	.47	−1.6	1.	−.91	1.8	.057	3.1	1.6
7	.36	−1.9	.63	−1.6	1.3	−.72	2.2	.4	3.8	2.1
8	.66	−1.8	.98	−1.4	1.7	−.47	2.8	.8	4.6	2.7
9	1.2	−1.7	1.5	−1.2	2.4	−.18	3.6	1.2	5.6	3.4
10	2.1	−1.5	2.5	−1.	3.5	.16	5.	1.7	7.3	4.1
11	2.2	−1.3	2.7	−.74	3.7	.54	5.2	2.3	7.6	4.9
12	2.3	−1.	2.8	−.43	4.	.95	5.6	2.8	8.3	5.7
13	2.7	−.72	3.2	−.096	4.5	1.4	6.3	3.4	9.2	6.5
14	3.3	−.4	3.8	.27	5.2	1.9	7.2	4.	10.	7.3
15	4.	−.049	4.6	.67	6.1	2.4	8.2	4.7	11.	8.2
16	5.4	.33	6.1	1.1	7.9	2.9	10.	5.4	14.	9.1
17	5.2	.73	5.9	1.5	7.6	3.5	10.	6.1	14.	10.
18	5.6	1.2	6.3	2.	8.1	4.1	11.	6.8	15.	11.
19	6.1	1.6	6.9	2.5	8.8	4.7	11.	7.6	16.	12.
20	6.8	2.1	7.6	3.	9.7	5.3	12.	8.3	17.	13.
21	7.7	2.6	8.6	3.6	11.	6.	14.	9.1	18.	14.
22	9.8	3.1	11.	4.1	13.	6.6	16.	10.	20.	15.
23	9.1	3.6	10.	4.7	12.	7.3	16.	11.	21.	16.
24	9.5	4.2	11.	5.3	13.	8.	16.	12.	22.	17.
25	10.	4.7	11.	5.9	14	8.7	17.	13.	23.	18.
26	11.	5.3	12.	6.5	15.	9.5	18.	13.	24.	19.

Tableau 1.13: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 26$.

$f(t) = \exp(7t)$ $G(x,t) = 1/(1-xt)$, $c_i = 7^i/i!$ $u_i(x) = x^i$, $f_i(t) = t^i$, $L_i(f) = f(1/(i+1))$												
	$t = -\frac{1}{10}$		$t = \frac{1}{10}$		$t = \frac{3}{10}$		$t = \frac{1}{2}$		$t = \frac{7}{10}$		$t = \frac{9}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	.38	.3	.044	−.0059	−.83	−.86	−1.5	−1.5	−2.1	−2.1	−2.7	−2.7
1	.97	.71	.7	.5	−.57	−.7	−1.4	−1.5	−2.	−2.1	−2.6	−2.7
2	1.8	1.3	1.6	1.2	−.11	−.46	−1.1	−1.4	−1.8	−2.1	−2.1	−2.7
3	3.	2.1	2.8	1.9	.63	−.12	−.45	−1.2	−.95	−2.	−2.5	−2.7
4	4.6	2.9	4.7	2.8	2.3	.29	.15	−.96	−1.4	−1.9	−2.8	−2.6
5	5.3	3.8	5.	3.7	1.8	.78	−.015	−.67	−1.4	−1.7	−2.9	−2.5
6	6.2	4.8	5.9	4.7	2.2	1.3	.27	−.33	−1.2	−1.5	−2.8	−2.4
7	7.3	5.9	7.	5.8	2.9	1.9	.75	.056	−.88	−1.2	−2.5	−2.2
8	8.6	7.	8.4	6.9	3.8	2.6	1.4	.49	−.36	−.92	−2.1	−2.
9	10.	8.1	9.9	8.1	4.8	3.2	2.3	.96	.43	−.58	−1.3	−1.8
10	12.	9.3	13.	9.3	7.	4.	3.7	1.5	1.3	−.21	−.83	−1.5
11	13.	11.	13.	11.	6.6	4.7	3.5	2.	1.2	.0	−.89	−1.2
12	14.	12.	14.	12.	7.3	5.5	4.	2.6	1.6	.64	−.59	−.84
13	16	13.	15.	13.	8.2	6.4	4.7	3.2	2.2	1.1	−.12	−.48
14	17.	14.	17.	14.	9.3	7.2	5.6	3.9	2.9	1.6	.51	−.082
15	19.	16.	19.	16.	11.	8.1	6.7	4.5	3.9	2.1	1.4	.34
16	22.	17.	21.	17.	12.	9.	8.1	5.2	5.	2.7	2.2	.78
17	22.	19	22.	19.	13.	10.	8.2	5.9	5.	3.3	2.3	1.2
18	23.	20.	23.	20.	14.	11.	8.9	6.7	5.6	3.9	2.7	1.7
19	25.	21.	25.	21.	15.	12.	9.7	7.4	6.3	4.5	3.3	2.2
20	27.	23.	26.	23.	16.	13.	11.	8.2	7.2	5.1	4.1	2.8
21	28.	24.	28.	24.	17.	14.	12.	9.	8.4	5.8	5.2	3.3
22	31.	26.	30.	26.	19.	15.	13.	9.8	9.4	6.4	6.	3.9
23	32.	28.	31.	27.	19.	16.	14.	11.	9.7	7.1	6.3	4.5
24	33.	29.	33.	29.	21.	17.	15.	12.	10.	7.8	6.8	5.1
25	35.	31.	35.	31.	22.	18.	16.	12.	11.	8.6	7.6	5.7
26	37.	32.	36.	32.	23.	19.	17.	13.	12.	9.3	8.5	6.3

Tableau 1.14: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 26$.

$f(t) = \exp(7t)$

$G(x,t) = (1-xt)/(1-2xt+x^2)$, $c_i = 2BesselI[i,7]$

$u_i(x) = x^i$, $f_i(t) = T_i(t)$, $L_i(f) = f(1/(i+1))$

	$t=-1$		$t=-\frac{9}{10}$		$t=-\frac{7}{10}$		$t=-\frac{1}{2}$		$t=-\frac{3}{10}$		$t=-\frac{1}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	−1.9	−2.2	−1.9	−2.2	−1.9	−2.2	−1.9	−2.2	−1.9	−2.2	−1.9	−2.2
1	−1.6	−2.2	−1.5	−2.1	−1.5	−1.7	−1.4	−1.1	−1.2	−1.9	−.61	−2.1
2	−.4	−2.2	−.009	−1.6	−.25	−1.7	−.61	−2.	−.72	−2.1	−.65	−2.
3	−.41	−1.8	−.033	−.67	−.25	−1.8	−.61	−1.8	−.73	−.81	−.67	−1.7
4	.64	−1.6	1.6	−1.3	.57	−1.5	.4	−.9	.51	−1.6	1.	−1.6
5	1.1	−1.2	1.8	−1.2	.85	−.52	1.	−1.3	.87	−1.1	.34	−1.1
6	1.3	−.88	1.4	−.89	1.4	−.69	1.2	−.79	.87	−.76	2.7	−.93
7	2.3	−.47	2.3	−.45	2.7	−.5	2.	.058	2.4	−.51	1.6	−.2
8	3.	−.022	3.	.13	3.	.063	2.7	−.056	2.5	.82	2.3	−.083
9	3.3	.47	3.2	.98	3.	1.6	3.2	.59	2.8	.41	2.6	.93
10	4.	.99	4.1	1.8	4.	1.1	3.7	1.5	3.8	1.3	3.5	.92
11	6.9	1.6	5.9	1.8	5.5	1.5	5.2	1.5	5.2	1.6	5.1	2.4
12	5.5	2.1	6.8	2.2	5.5	2.3	5.7	2.3	5.2	2.2	4.7	2.1
13	6.2	2.8	6.4	2.8	6.	5.6	5.8	3.2	5.7	3.3	5.9	4.
14	7.5	3.4	7.5	3.4	7.4	3.5	7.4	3.4	7.1	3.4	6.8	3.4
15	8.1	4.1	8.	4.3	8.3	4.1	9.1	4.3	10.	4.5	7.9	4.7
16	8.7	4.8	8.6	5.4	8.7	4.9	8.3	5.2	8.	4.8	7.8	4.7
17	9.8	5.5	9.8	6.2	9.5	6.7	9.4	5.5	9.5	5.6	9.7	5.9
18	11.	6.3	11.	6.5	11.	6.4	11.	6.4	11.	6.7	10.	6.3
19	11.	7.	12.	7.1	12.	7.	11.	7.4	11.	7.	11.	7.2
20	12.	7.8	13.	7.8	12.	8.	12.	7.8	13.	8.4	12.	7.9
21	15.	8.6	15.	8.7	15.	9.5	14.	8.8	14.	8.6	13.	8.7
22	14.	9.4	14.	9.6	14.	9.5	14.	9.8	14.	9.5	14.	9.6
23	15.	10.	15.	11.	15.	10.	15.	10.	15.	11.	14.	10.
24	17.	11.	17.	12.	17.	11.	17.	11.	16.	11.	16.	11.
25	17.	12.	18.	12.	17.	13.	17.	12.	17.	13.	16.	12.
26	18.	13.	19.	13.	18.	13.	18.	13.	18.	13.	20.	13.

Tableau 1.15: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 0, \ldots, 26$.

$f(t) = \exp(7t)$

$G(x,t) = (1-xt)/(1-2xt+x^2)$, $c_i = 2BesselI[i,7]$

$u_i(x) = x^i$, $f_i(t) = T_i(t)$, $L_i(f) = f(1/(i+1))$

	$t=0$		$t=\frac{1}{10}$		$t=\frac{3}{10}$		$t=\frac{1}{2}$		$t=\frac{7}{10}$		$t=\frac{9}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
0	−1.9	−2.2	−1.9	−2.2	−1.9	−2.2	−1.7	−2.1	−1.7	−1.5	−2.7	−2.6
1	−.45	−2.2	−1.1	−2.3	−1.6	−2.4	−2.	−2.5	−2.4	−2.4	−2.9	−2.
2	−.45	−1.9	1.9	−1.7	−1.	−1.7	−1.4	−2.2	−1.4	−2.4	−1.9	−1.8
3	−.48	−1.9	.61	−1.7	−.98	−1.9	−1.4	−.45	−1.4	−2.1	−1.9	−2.
4	.34	−1.3	.012	−.25	−.35	−1.7	−.29	−1.7	−.9	−1.3	−.61	−1.9
5	.26	−1.3	.32	−1.4	.28	−.67	−.38	−1.4	.037	−1.2	−.91	−1.6
6	.83	−.56	.51	−.51	.49	−1.	.13	−.25	.067	−1.1	−.66	−1.2
7	1.7	−.56	3.	−.55	1.3	−.32	1.6	−.55	.97	−.67	.67	−.65
8	4.	.39	2.1	.12	1.9	.012	1.4	−.18	1.2	.15	.48	.061
9	2.4	.39	2.7	.48	2.	.37	2.	.93	3.3	.73	1.	1.6
10	3.7	1.5	3.1	1.	3.3	2.1	3.	.99	2.4	.85	2.1	1.2
11	4.6	1.5	4.4	1.7	4.1	1.4	3.8	1.4	3.4	1.4	3.	1.5
12	6.2	2.7	4.4	2.1	4.4	2.3	4.4	2.5	3.7	2.3	3.1	2.
13	5.3	2.7	5.8	3.1	5.1	2.9	4.8	2.8	5.	3.1	3.7	2.6
14	7.	4.	6.5	3.3	6.2	3.3	6.	3.3	6.1	3.3	5.3	3.3
15	7.	4.	7.4	4.8	7.1	5.4	7.4	4.4	5.9	4.	5.3	4.2
16	8.5	5.4	7.5	4.7	7.2	4.7	7.	4.9	6.9	5.	6.2	5.7
17	8.8	5.4	9.6	6.7	8.7	5.7	8.1	5.4	8.3	5.9	7.9	5.8
18	10.	7.	9.7	6.2	9.4	6.4	9.8	6.6	8.6	6.2	8.3	6.3
19	10.	7.	12.	7.5	10.	7.	9.6	7.1	9.2	6.9	8.6	6.9
20	12.	8.6	11.	7.8	11.	9.1	11.	7.7	11.	8.1	9.6	7.7
21	13.	8.6	13.	8.9	13.	8.6	12.	8.9	12.	9.	11.	8.6
22	13.	10.	13.	9.5	14.	9.6	12.	9.6	12.	9.4	11.	9.6
23	14.	10.	14.	10.	14.	10.	13.	10.	13.	10.	13.	11.
24	17.	12.	16.	11.	15.	11.	16.	11.	15.	11.	15.	12.
25	16.	12.	17.	12.	16.	13.	15.	12.	15.	12.	15.	12.
26	17.	14.	17.	13.	17.	13.	16.	13.	16.	13.	15.	13.

Tableau 1.16: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 5, \ldots, 36$.

$f(t) = \exp(4t)\cos(4\sqrt{1-t^2})$

$G(x,t) = (1-xt)/(1-2xt+x^2)$, $c_i = 4^i/i!$

$u_i(x) = x^i$, $f_i(t) = T_i(t)$, $L_i(f) = f(1/(i+1))$

	$t=-1$		$t=-\frac{9}{10}$		$t=-\frac{3}{4}$		$t=-\frac{6}{10}$		$t=-\frac{3}{10}$		$t=-\frac{1}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
5	1.5	−.55	1.9	−.48	1.4	−.36	1.5	−.28	1.1	−.47	.93	−.32
6	2.5	−.55	2.9	−.34	2.3	.65	2.4	−.37	1.8	−.17	1.7	−.42
7	2.3	−.047	2.3	−.034	2.5	.054	2.2	.089	2.5	−.11	1.6	.3
8	2.8	.29	2.8	.43	3.2	.27	2.6	.89	2.3	1.4	2.5	.2
9	3.9	.68	3.9	1.2	3.7	.79	4.4	.67	3.9	.61	3.2	1.2
10	4.3	1.1	4.3	2.	4.1	2.2	4.7	1.1	4.5	1.4	3.5	1.
11	4.7	1.6	5.	1.8	4.6	1.8	4.4	2.6	4.1	1.7	4.1	2.6
12	5.5	2.1	6.4	2.1	6.4	2.1	5.2	2.2	5.5	2.1	4.7	2.
13	7.3	2.6	7.9	2.6	7.4	2.7	6.7	2.6	8.	3.2	6.2	3.6
14	6.9	3.2	6.9	3.2	6.7	3.7	6.8	3.6	6.3	3.1	6.	3.1
15	7.6	3.8	7.5	4.	7.4	4.1	7.2	4.	7.8	4.2	8.6	4.3
16	8.7	4.4	8.7	5.	8.9	4.4	8.7	4.4	8.1	4.5	7.9	4.4
17	9.6	5.1	9.5	5.8	11.	5.1	10.	5.2	9.	5.1	8.8	5.4
18	10.	5.7	10.	5.9	9.9	6.	9.7	6.4	9.6	6.2	9.1	5.7
19	11.	6.4	12.	6.5	11.	7.1	11.	6.4	10.	6.4	10.	6.6
20	13.	7.1	13.	7.1	13.	7.2	13.	7.2	12.	7.7	12.	7.2
21	13.	7.9	13.	7.9	13.	7.9	13.	8.7	12.	7.9	12.	8.
22	14.	8.6	13.	8.8	14.	8.8	13.	8.8	13.	8.7	13.	8.8
23	15.	9.4	15.	10.	15.	11.	15	9.4	14.	9.8	14.	9.4
24	16.	10.	16.	11.	16.	10.	17.	11.	15.	10.	17.	10.
25	16.	11.	17.	11.	16.	11.	16.	11.	16.	12.	15.	11.
26	17.	12.	18.	12.	18.	12.	17.	12.	17.	12.	18.	12.
27	19.	13.	19.	13.	19.	13.	21.	13.	18.	13.	18.	13.
28	20.	14.	20.	14.	19.	14.	19.	15.	19.	14.	19.	14.
29	20.	14.	20.	15.	20.	14.	20.	14.	20.	14.	19.	14.
30	22.	15.	22.	16.	22.	15.	21.	15.	21.	16.	21.	16.
31	24.	16.	23.	17.	24.	17.	24.	17.	23.	16.	23.	16.
32	24.	17.	24.	17.	23.	18.	23.	17.	23.	17.	23.	18.
33	25.	18.	25.	18.	24.	18.	24.	18.	24.	18.	24.	18.
34	26.	19.	27.	19.	26.	19.	27.	19.	25.	19.	25.	19.
35	27.	20.	27.	20.	28.	20.	27.	20.	27.	21.	27.	20.
36	28.	21.	28.	21.	28.	22.	27.	21.	27.	21.	27.	21.

Tableau 1.17: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$, $n = 5, \ldots, 36$.

$f(t) = \exp(4t)\cos(4\sqrt{1-t^2})$

$G(x,t) = (1-xt)/(1-2xt+x^2)$, $c_i = 4^i/i!$

$u_i(x) = x^i$, $f_i(t) = T_i(t)$, $L_i(f) = f(1/(i+1))$

	$t=0$		$t=\frac{1}{10}$		$t=\frac{3}{10}$		$t=\frac{6}{10}$		$t=\frac{3}{4}$		$t=\frac{9}{10}$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
5	1.	−.64	1.9	−.73	.56	−.3	1.	−.87	−.027	−.1	.038	−1.
6	2.1	−.14	1.7	.2	1.	−.54	.55	−.5	.58	−.53	−.23	−.71
7	1.5	−.14	1.6	−.17	1.2	.16	.98	3.	.51	−.35	−.025	−.28
8	2.4	.59	1.9	.47	2.4	.26	1.8	.16	.98	.083	.6	.36
9	3.3	.59	3.4	.64	3.	.56	3.3	.46	2.3	.92	2.2	2.4
10	4.6	1.5	3.3	1.1	4.1	1.8	2.7	1.2	2.9	1.3	2.4	1.2
11	3.7	1.5	4.1	1.7	3.4	1.4	3.5	1.8	2.7	1.4	2.8	1.4
12	5.3	2.5	4.5	2.	4.3	2.2	3.8	1.9	3.5	1.9	3.3	1.9
13	5.9	2.5	5.7	2.9	5.3	2.7	5.1	2.6	4.7	2.6	3.9	2.4
14	6.7	3.7	5.8	3.1	5.5	3.1	6.	6.5	6.	4.4	4.2	3.1
15	6.6	3.7	7.3	4.4	7.1	4.8	5.8	3.7	5.6	3.8	5.	3.9
16	8.7	5.	7.6	4.3	7.4	4.3	6.9	4.3	6.7	4.2	6.4	5.4
17	8.5	5.	8.8	6.6	8.8	5.2	8.7	5.3	7.2	5.	8.	5.3
18	9.5	6.4	8.8	5.6	8.6	5.8	8.1	5.9	8.4	6.2	7.6	5.7
19	9.9	6.4	11.	7.	9.5	6.4	8.9	6.3	9.	6.6	8.3	6.3
20	12.	7.8	13.	7.1	11.	8.2	11.	7.2	10.	7.	9.9	7.
21	12.	7.8	13.	8.2	11.	7.8	11.	8.6	10.	7.8	9.8	7.8
22	13.	9.4	12.	8.6	13.	8.8	11.	8.6	12.	8.8	11.	8.8
23	14.	9.4	14.	9.5	13.	9.5	13.	9.3	13.	10.	12.	10.
24	15.	11.	15.	10.	15.	10.	14.	11.	13.	10.	14.	11.
25	15.	11.	16.	11.	15.	12.	14.	11.	14.	11.	14.	11.
26	17.	13.	16.	12.	16.	12.	15.	12.	15.	12.	14.	12.
27	18.	13.	18.	13.	18.	13.	17.	13.	17.	14.	16.	13.
28	19.	14.	18.	14.	18.	14.	17.	14.	17.	14.	16.	14.
29	19.	14.	19.	14.	19.	14.	18.	14.	18.	14.	17.	15.
30	21.	16.	20.	16.	20.	16.	20.	15.	19.	15.	19.	16.
31	22.	16.	22.	16.	22.	16.	21.	17.	21.	17.	20.	17.
32	23.	18.	22.	18.	22.	17.	21.	17.	21.	17.	21.	17.
33	24.	18.	23.	18.	23.	18.	23.	18.	22.	18.	21.	18.
34	25.	20.	25.	20.	25.	19.	25.	19.	24.	19.	23.	19.
35	26.	20.	26.	20.	25.	21.	25.	20.	26.	20.	24.	20.
36	27.	22.	27.	21.	26.	21.	26.	21.	25.	22.	25.	21.

Tableau 1.18: Nombre de chiffres exacts de $(n)_f(t)$ et de $ps(n)$.

$f(t) = \exp(5)\text{besselj0}(2\sqrt{5t})$

$G(x,t) = 1/((1-x)\exp(tx/(1-x)))$, $c_i = 5^i/i!$

$u_i(x) = x^i$, $f_i(t) = L_i^0(t)$, $L_i(f) = f(1/(i+1))$

	$t=0$		$t=\frac{1}{10}$		$t=\frac{1}{2}$		$t=1$		$t=\frac{3}{2}$		$t=2$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
8	.062	−1.	1.1	−.35	.71	−.6	.44	−.58	1.3	−.64	.39	−.3
15	4.4	2.	5.	3.	4.9	2.8	5.3	2.7	4.9	2.3	4.8	2.8
22	9.6	6.2	10.	6.8	10.	6.7	10.	6.7	10.	7.5	10.	6.5
29	16.	11.	16.	12.	17.	12.	17.	12.	16.	12.	16.	12.
37	24	18.	24.	18.	24.	19.	24.	19.	24.	19.	24.	19.

	$t=\frac{5}{2}$		$t=3$		$t=\frac{7}{2}$		$t=6$		$t=7$		$t=8$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
8	.43	−.89	1.	−.87	.24	.27	−.14	−.87	.26	−1.7	−.29	−1.7
15	4.8	2.2	4.9	2.2	4.5	2.5	4.6	1.5	4.	2.2	4.3	1.1
22	9.8	6.8	11.	6.3	9.7	7.5	9.4	5.9	9.6	5.6	9.	5.8
29	16.	12.	16.	12.	16.	12.	15.	11.	16.	11.	15.	11.
37	24.	18.	24.	19.	26.	18.	23.	18.	24.	17.	24.	17.

	$t=9$		$t=10$		$t=11$		$t=12$		$t=13$		$t=14$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
8	−.56	−1.3	−.31	−2.2	−.52	−2.4	−1.	−2.4	−1.1	−2.2	−.94	−2.9
15	3.7	1.4	4.9	.76	3.4	.52	3.3	1.7	3.5	.11	2.9	−.1
22	10.	5.1	8.6	5.4	9.3	4.8	8.3	4.7	8.4	4.6	8.1	4.1
29	15.	10.	15.	10.	14.	10.	15.	9.8	14.	9.6	14.	9.5
37	23.	19.	23.	17.	23.	17.	22.	17.	24.	16.	22.	16.

	$t=15$		$t=20$		$t=25$		$t=30$		$t=40$		$t=50$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
8	−.83	−3.3	−1.7	−4.2	−3.1	−4.2	−3.7	−6.4	−4.1	−8.1	−4.2	−9.2
15	2.9	.53	2.2	−1.3	1.1	−2.4	.32	−3.5	−1.5	−5.3	−3.4	−7.2
22	7.8	4.5	6.9	2.9	6.3	2.8	5.5	.84	3.3	−1.4	1.6	−2.
29	14.	9.1	13.	8.	12.	7.1	11.	6.	9.3	3.9	7.2	1.6
37	23.	16.	21.	15.	20.	14.	19.	13.	17.	11.	15.	8.3

	$t=60$		$t=70$		$t=80$		$t=90$		$t=100$		$t=200$	
n	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$	$(n)_f$	$ps(n)$
8	−4.	−10.	−3.8	−11.	−3.5	−11.	−3.2	−12.	−2.8	−12.	−1.1	−15.
15	−4.6	−10.	−5.1	−12.	−5.3	−13.	−5.2	−14.	−6.1	−15.	−6.7	−20.
22	−.28	−5.7	−2.	−7.9	−4.6	−9.9	−5.9	−12.	−6.9	−14.	−11.	−22.
29	5.8	−.068	3.3	−2.7	1.5	−4.6	−.58	−6.6	−2.6	−9.1	−11.	−23.
37	13.	6.3	11.	4.	9.3	2.	7.3	−.25	5.8	−1.7	−9.1	−20.

Nous observons que les signes qui apparaîssent dans les graphes, où nous représentons le nombre de chiffres exacts des approximants $(n)_f(t)_{n=0,1,...}$, et des sommes partielles $ps(n)_f(t)_{n=0,1,...}$, en fonction de la valeur de t; correspondent aux points donnés dans les tableaux correspondants. Les graphes sont construits en unissant ces points par des segments de droite.

1.4 Conclusion

L'effet cumulatif des erreurs d'arrondi dans le calcul des approximants de type Padé généralisés est important.
Si les approximants ne convergent pas assez vite, les erreurs d'arrondi risquent d'entâcher les résultats avant que la convergence ne puisse être observée.
Les implémentations en *Mathematica*[13], permettent de faire du calcul approché de haute précision et de minimiser l'effet cumulatif des erreurs d'arrondi, permettant ainsi d'observer la convergence ou la divergence réelle des approximants et d'en tirer des conclusions.

Nous avons montré qu'au moins dans certains cas, il est possible de trouver des suites de fonctionnelles qui produisent des approximants de type Padé généralisés qui convergent vers la série et plus vite que la suite des sommes partielles.
Nous avons réussi à approcher des fonctions relativement compliquées, comme celles des exemples 13 et 15 de la section 1.3.5, par des approximants généralisés plus simples que les séries correspondantes.

Il est vrai que, les fonctionnelles utilisés dans la section 1.3.5, correspondent toutes à l'évaluation de la fonction génératrice dans un point, c'est-à-dire à l'interpolation de Lagrange. Cela dit, nous aurions pu essayer des fonctionnelles définies par des intégrales, mais dans ce cas les approximants deviendraient plus compliqués que les séries, or nous avons jugé raisonnable de ne pas tenir compte des exemples pour lesquels cela se produirait.

Le calcul des approximants de type Padé généralisés en utilisant les relations de récurrence de la biorthogonalité, exige des conditions d'applicabilité plus fortes que les conditions d'applicabilité de la méthode de bordage par blocs. Ces dernières coïncident avec les conditions d'existence et d'unicité des approximants à calculer.

De ce fait, il serait important de faire l'implémentation de la méthode de bordage par blocs en *Mathematica*[13].

2

Polynômes biorthogonaux adjacents

2.1 Définition et cas particuliers

Dans la section 1.3 supposons que E est une algèbre commutative, et que $x_i = x^i$ où $x \in E$.

Alors,

$$N_{n+1}^{(i,j)} = x^j \begin{vmatrix} L_i(x^j) & \cdots & L_i(x^{j+n}) \\ \cdots & & \cdots \\ L_{i+n-1}(x^j) & \cdots & L_{i+n-1}(x^{j+n}) \\ 1 & \cdots & x^n \end{vmatrix}$$

et

$$D_n^{(i,j)} = \begin{vmatrix} L_i(x^j) & \cdots & L_i(x^{j+n-1}) \\ \cdots & & \cdots \\ L_{i+n-1}(x^j) & \cdots & L_{i+n-1}(x^{j+n-1}) \end{vmatrix}.$$

Supposons que

$$D_n^{(i,j)} \neq 0 \ , \ \forall i, j, n \in N_0,$$

et rappelons que

$$x_n^{(i,j)} = N_{n+1}^{(i,j)} / D_n^{(i,j)}.$$

Définissons le polynôme $P_n^{(i,j)}$ à partir de l'égalité

$$x_n^{(i,j)} = x^j P_n^{(i,j)}(x).$$

Alors, $P_n^{(i,j)}$ est un polynôme unitaire, de degré n qui satisfait les n conditions de biorthogonalité

$$L_p(x^j P_n^{(i,j)}(x)) = 0 \ , \ p = i, \ldots, i+n-1. \tag{2.1}$$

Réciproquement, tout polynôme unitaire de degré n, qui satisfait ces conditions coïncide avec $P_n^{(i,j)}$.

Définissons le déterminant $N_{n+1}^{(i,j)}(x)$ à partir de l'égalité suivante

$$N_{n+1}^{(i,j)} = x^j N_{n+1}^{(i,j)}(x);$$

alors

$$P_n^{(i,j)}(x) = N_{n+1}^{(i,j)}(x) / D_n^{(i,j)}.$$

Cette définition des polynômes biorthogonaux adjacents a été donnée par Brezinski [3]. Considérons les cas particuliers suivants de cette définition.

1. Soit L une fonctionnelle linéaire définie sur l'espace des polynômes complexes. Définissons les fonctionnelles linéaires L_p à partir de L, de la façon suivante

$$L_p(x^k) = L(x^{k+p}) \quad , p, k \geq 0.$$

C'est-à-dire, le moment d'ordre k de la fonctionnelle L_p est égal au moment d'ordre $k+p$ de la fonctionnelle L, ou d'une autre manière la suite des moments de L_p est obtenue à partir de la suite des moments de L, en ignorant les p premiers moments. Si nous dénotons $L(x^{k+p})$ par l_{k+p} , ces suites s'écrivent de la manière suivante

$$\begin{aligned} L &\longmapsto l_0, l_1, \ldots, l_p, l_{p+1}, \ldots \\ L_p &\longmapsto l_p, l_{p+1}, \ldots . \end{aligned}$$

L_p est couramment dénoté par $L^{(p)}$.

Ecrivons les déterminants $N_{n+1}^{(i,j)}(x)$ et $D_n^{(i,j)}$ dans ce cas particulier

$$N_{n+1}^{(i,j)}(x) = \begin{vmatrix} L(x^{i+j}) & \cdots & L(x^{i+j+n}) \\ \cdots & & \cdots \\ L(x^{i+j+n-1}) & \cdots & L(x^{i+j+2n-1}) \\ 1 & \cdots & x^n \end{vmatrix}$$

$$D_n^{(i,j)} = \begin{vmatrix} L(x^{i+j}) & \cdots & L(x^{i+j+n-1}) \\ \cdots & & \cdots \\ L(x^{i+j+n-1}) & \cdots & L(x^{i+j+2n-2}) \end{vmatrix}.$$

Nous voyons que $D_n^{(i,j)}$ coïncide avec le déterminant de Hankel $H_n^{(i+j)}$, et que les conditions de biorthogonalité de $P_n^{(i,j)}$ s'écrivent de la façon suivante :

$$L(x^{i+j+p} P_n^{(i,j)}) = L^{(i+j)}(x^p P_n^{(i,j)}) = 0 \quad , p = 0, \ldots, n-1.$$

Alors, $P_n^{(i,j)} = P_n^{(i+j)}$, où $\{P_n^{(i+j)}\}_{n \geq 0}$ est la famille des polynômes orthogonaux unitaires $(i+j)$—adjacents par rapport à la fonctionnelle L, c'est-à-dire la famille des polynômes orthogonaux unitaires par rapport à $L^{(i+j)}$.

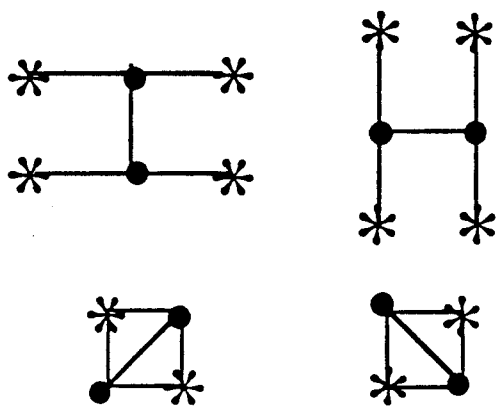
Si $i = j = 0$, $P_n^{(0,0)} = P_n^{(0)} = P_n$, où $\{P_n\}_{n \geq 0}$ est la famille des polynômes orthogonaux unitaires par rapport à L.

Disposons dans le plan les polynômes orthogonaux adjacents $P_n^{(k)}$, $n \geq 0, k \geq -n$ comme il est indiqué dans la figure 2.1 : dans chaque colonne, le degré n des polynômes est fixé, et dans chaque diagonale descendante, l'indice k est fixé, donc les polynômes situés sur la même diagonale descendante sont orthogonaux par rapport

Figure 2.1: Disposition des polynômes orthogonaux adjacents dans le plan.

$$\begin{array}{ccccc} & \vdots & \vdots & \vdots & \\ \cdots & P_{n-1}^{(k)} & P_n^{(k-1)} & P_{n+1}^{(k-2)} & \cdots \\ \cdots & P_{n-1}^{(k+1)} & P_n^{(k)} & P_{n+1}^{(k-1)} & \cdots \\ \cdots & P_{n-1}^{(k+2)} & P_n^{(k+1)} & P_{n+1}^{(k)} & \cdots \\ & \vdots & \vdots & \vdots & \end{array}$$

Figure 2.2: Relations parmi trois polynômes orthogonaux adjacents consécutifs.



à la fonctionnelle $L^{(k)}$. L'indice n croît de la gauche vers la droite, l'indice k croît du haut vers le bas.

En admettant cette disposition des polynômes dans le plan, il existe des relations de récurrence parmi trois polynômes orthogonaux adjacents consécutifs, correspondantes à toutes les directions. Nous schématisons ces relations dans la figure 2.2, où nous représentons les polynômes situés dans le membre droite des relations par des $\bullet$, et le polynôme situé dans le membre gauche par des $*$; c'est-à-dire si les polynômes représentés par des $\bullet$ sont connus, les relations permettent le calcul des polynômes représentés par des $*$.

Une partie de ces relations a été déduite dans [2] pour les polynômes $\tilde{P}_n^{(k)} = x^n P_n^{(k)}(x^{-1})$ dans le cas normal, à savoir le cas où les déterminants de Hankel $H_n^{(k)}$ sont supposés non nuls. Pour le cas non normal voir Draux [9].

2. Soient $L_0, \ldots, L_{d-1}$ d fonctionnelles linéaires définies sur l'espace des polynômes

complexes. Définissons les fonctionnelles d'ordre supérieur à d de la façon suivante

$$L_p(x^k) = L_{dq+r}(x^k) = L_r(x^{k+q}),$$

où $p \geq d$ et r et q sont respectivement le reste et le quotient de la division de p par d.

Comme r est un entier positif inférieur à d, le moment d'ordre k de L_p, où $p \geq d$, est défini comme le moment d'ordre $k+q$ de L_r, où L_r est une des d fonctionnelles connues $L_0, \ldots, L_{d-1}$.

Dans la table suivante nous représentons les moments des fonctionnelles L_i, et nous dénotons $L_r(x^{k+q})$ par $l_{r,k+q}$.

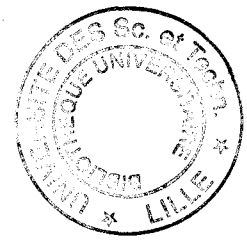
L_0	$\cdots$	L_{d-1}	L_d	$\cdots$	L_{2d-1}	L_{2d}	$\cdots$	L_{3d-1}	$\cdots$
l_{00}	$\cdots$	$l_{d-1,0}$	l_{01}	$\cdots$	$l_{d-1,1}$	l_{02}	$\cdots$	$l_{d-1,2}$	$\cdots$
l_{01}	$\cdots$	$l_{d-1,1}$	l_{02}	$\cdots$	$l_{d-1,2}$	l_{03}	$\cdots$	$l_{d-1,3}$	$\cdots$
l_{02}	$\cdots$	$l_{d-1,2}$	l_{03}	$\cdots$	$l_{d-1,3}$	l_{04}	$\cdots$	$l_{d-1,4}$	$\cdots$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$

Ecrivons les déterminants $N_{n+1}^{(i,j)}(x)$ et $D_n^{(i,j)}$ dans ce cas particulier.

Calculons le quotient q_i et le reste r_i de la division de i par d, et le quotient q et le reste r de la division de $n - d + r_i$ par d : $i = dq_i + r_i$, où $0 \leq r_i < d$ et $n = d - r_i + dq + r$, où $0 \leq r < d$.

$$N_{n+1}^{(i,j)}(x) = \begin{vmatrix} L_{r_i}(x^{j+q_i}) & \cdots & L_{r_i}(x^{j+q_i+n}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i}) & \cdots & L_{d-1}(x^{j+q_i+n}) \\ L_0(x^{j+q_i+1}) & \cdots & L_0(x^{j+q_i+n+1}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i+1}) & \cdots & L_{d-1}(x^{j+q_i+n+1}) \\ \vdots & & \vdots \\ L_0(x^{j+q_i+q}) & \cdots & L_0(x^{j+q_i+q+n}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i+q}) & \cdots & L_{d-1}(x^{j+q_i+q+n}) \\ L_0(x^{j+q_i+q+1}) & \cdots & L_0(x^{j+q_i+q+n+1}) \\ \cdots & & \cdots \\ L_{r-1}(x^{j+q_i+q+1}) & \cdots & L_{r-1}(x^{j+q_i+q+n+1}) \\ 1 & \cdots & x^n \end{vmatrix}$$

$$D_n^{(i,j)} = \begin{vmatrix} L_{r_i}(x^{j+q_i}) & \cdots & L_{r_i}(x^{j+q_i+n-1}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i}) & \cdots & L_{d-1}(x^{j+q_i+n-1}) \\ L_0(x^{j+q_i+1}) & \cdots & L_0(x^{j+q_i+n}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i+1}) & \cdots & L_{d-1}(x^{j+q_i+n}) \\ \vdots & & \vdots \\ L_0(x^{j+q_i+q}) & \cdots & L_0(x^{j+q_i+q+n-1}) \\ \cdots & & \cdots \\ L_{d-1}(x^{j+q_i+q}) & \cdots & L_{d-1}(x^{j+q_i+q+n-1}) \\ L_0(x^{j+q_i+q+1}) & \cdots & L_0(x^{j+q_i+q+n}) \\ \cdots & & \cdots \\ L_{r-1}(x^{j+q_i+q+1}) & \cdots & L_{r-1}(x^{j+q_i+q+n}) \end{vmatrix}.$$



Les conditions de biorthogonalité des polynômes $P_n^{(i,j)}$ s'écrivent de la façon suivante :

$$\begin{aligned} &L_p(x^{j+q_i}P_n^{(i,j)}(x)) = 0 \ , p = r_i, \ldots, d-1 \\ &L_p(x^{j+q_i+k}P_n^{(i,j)}(x)) = 0 \ , p = 0, \ldots, d-1 \ , \ k = 1, \ldots, q \\ &L_p(x^{j+q_i+q+1}P_n^{(i,j)}(x)) = 0 \ , p = 0, \ldots, r-1. \end{aligned}$$

Si $r_i = 0$, alors $D_n^{(dq_i,j)} = H_{(d,n)}^{(j+q_i)}$ et $P_n^{(dq_i,j)} = P_{(d,n)}^{(j+q_i)}$, où $H_{(d,n)}^{(j+q_i)}$ sont les déterminants de Hankel généralisés dans le sense de Van Iseghem [10] et $\{P_{(d,n)}^{(j+q_i)}\}_{n\geq 0}$ est la famille des polynômes orthogonaux vectoriels unitaires de dimension d, $(j+q_i)-$adjacents par rapport aux fonctionnelles $\{L_r\}_{r=0,\ldots,d-1}$. Si $d = 1$, nous retrouvons les polynômes orthogonaux adjacents par rapport à L_0.

Ces polynômes ont été définis et étudiés par Van Iseghem [10], qui a montré que : $\{P_{(d,n)}^{(k)}\}_{n\geq 0}$ satisfait une relation de récurrence d'ordre $d+1$, qui se réduit dans le cas où $d = 1$ et $k = 0$ à la relation d'orthogonalité classique et que les deux familles $\{P_{(d,n)}^{(k)}\}_{n\geq 0}$ et $\{P_{(d,n)}^{(k+1)}\}_{n\geq 0}$ sont liés par un algorithme QD. D'autres relations d'ordre $d+1$ peuvent aussi être déduites (voir Van Iseghem[10]).

Quand il n'y aura pas danger de confusion, nous dénoterons $P_{(d,n)}^{(k)}$ par $P_n^{(k)}$, et $H_{(d,n)}^{(k)}$ par $H_n^{(k)}$.

3. Soit L une fonctionnelle linéaire définie sur $span\{x^l, l \in Z\}$. Définissons les fonctionnelles linéaires L_p à partir de L, de la façon suivante

$$L_p(x^k) = L(x^{k-p}) \ , p, k \in N_0.$$

Ecrivons les déterminants $N_{n+1}^{(i,j)}(x)$ et $D_n^{(i,j)}$ dans ce cas particulier

$$N_{n+1}^{(i,j)}(x) = \begin{vmatrix} L(x^{j-i}) & \cdots & L(x^{j-i+n}) \\ \cdots & & \cdots \\ L(x^{j-i-n+1}) & \cdots & L(x^{j-i+1}) \\ 1 & \cdots & x^n \end{vmatrix}, \tag{2.2}$$

$$D_n^{(i,j)} = \begin{vmatrix} L(x^{j-i}) & \cdots & L(x^{j-i+n-1}) \\ \cdots & & \cdots \\ L(x^{j-i-n+1}) & \cdots & L(x^{j-i}) \end{vmatrix}. \tag{2.3}$$

Nous voyons que les conditions de biorthogonalité de $P_n^{(i,j)}$ s'écrivent de la façon suivante

$$L(x^{j-i-p}P_n^{(i,j)}) = 0 \ , p = 0, \ldots, n-1.$$

Nous remarquons que dans (2.2) et (2.3) ce ne sont pas les indices i et j qui apparaîssent, mais leur difference. Donc, il suffit de dénoter $D_n^{(i,j)}$ par $H_n^{[j-i]}$, et $P_n^{(i,j)}$ par $P_n^{[j-i]}$, où $\{P_n^{[j-i]}\}_{n\geq 0}$ est la famille des polynômes orthogonaux vectoriels unitaires de dimension -1, $(j-i)$−adjacents par rapport à L. Nous remarquons que $j-i \in Z$.

Ces polynômes ont été définis et étudiés par Brezinski [4], qui a montré qu'ils satisfont plusieurs relations de récurrence d'ordre 2.

2.2 Relations de récurrence

Plaçons nous dans le cas normal, c'est-à-dire supposons que

$$D_n^{(i,j)} \neq 0 \ , \forall i, j, n \in N_0. \tag{2.4}$$

Comme $\{x^j\}_{j=0,1,\ldots}$ est une famille linéairement indépendante, cette condition est équivalente à la condition : $\{L_i\}_{i=0,1,\ldots}$ linéairement indépendants (voir Davis [8]).

A partir de la définition de $P_n^{(i,j)}$ comme rapport de déterminants, il est facile de voir que

$$L_{i-1}(x^j P_n^{(i,j)}) = (-1)^n D_{n+1}^{(i-1,j)}/D_n^{(i,j)}, \tag{2.5}$$

$$L_{i+n}(x^j P_n^{(i,j)}) = D_{n+1}^{(i,j)}/D_n^{(i,j)}, \tag{2.6}$$

$$P_n^{(i,j)}(0) = (-1)^n D_n^{(i,j+1)}/D_n^{(i,j)}. \tag{2.7}$$

Donc, dans le cas normal

$$L_{i-1}(x^j P_n^{(i,j)}) \neq 0 \ , \tag{2.8}$$

$$L_{i+n}(x^j P_n^{(i,j)}) \neq 0 \ , \tag{2.9}$$

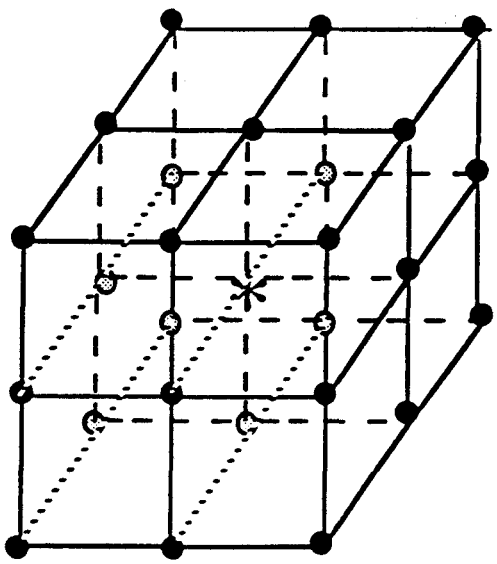
$$P_n^{(i,j)}(0) \neq 0. \tag{2.10}$$

C'est-à-dire, les conditions de biorthogonalité de $P_n^{(i,j)}$ ne sont pas étendues à $p = i-1$ ou à $p = i+n$, et le terme indépendant de ces polynômes est non nul.

Revenons au cas particulier des polynômes orthogonaux adjacents. Il est facile de voir que les relations de récurrence de la figure 2.2 permettent le calcul de la table des $P_n^{(k)}, n \geq 0, k \geq -n$, en gardant en mémoire seulement deux polynômes à la fois.

Les polynômes biorthogonaux sont représentés dans l'espace, puisqu'ils ont trois indices, et la table des $P_n^{(i,j)}$ est une table à trois dimensions. Représentons dans la figure 2.3 $P_n^{(i,j)}$ par une $*$, et ses 26 polynômes voisins par des $\bullet$. Cherchons toutes les relations

Figure 2.3 : $* = P_n^{(i,j)}$



qui permettent le calcul de $P_n^{(i,j)}$ à partir de 2 de ces polynômes. Nous avons $C_2^{26} = 325$ possiblités différentes.

Nous ne devons pas espérer pouvoir trouver des relations de récurrence correspondantes à toutes les possibilités, puisque dans le cas général des polynômes biorthogonaux, la seule hypothèse qui est faite sur les fonctionnelles L_i est qu'elles sont linéairement indépendantes.

La notion de polynômes biorthogonaux voisins ou adjacents consécutifs dépend évidemment de la façon dont ces polynômes sont disposés dans l'espace.

Plaçons les polynômes biorthogonaux adjacents $P_n^{(i,j)}$ de telle sorte que, quand ces polynômes coïncident avec les polynômes orthogonaux adjacents $P_n^{(k)}$, on obtienne la table de la figure 2.1. Soient i et j tels que $i + j = k$, alors dans le plan $i = const$, les polynômes biorthogonaux se disposent de la façon suivante.

Première disposition des $P_n^{(i,j)}$.

$$
\begin{array}{ccccc}
 & \vdots & \vdots & \vdots & \\
\cdots & P_{n-1}^{(i,j)} & P_n^{(i,j-1)} & P_{n+1}^{(i,j-2)} & \cdots \\
\cdots & P_{n-1}^{(i,j+1)} & P_n^{(i,j)} & P_{n+1}^{(i,j-1)} & \cdots \\
\cdots & P_{n-1}^{(i,j+2)} & P_n^{(i,j+1)} & P_{n+1}^{(i,j)} & \cdots \\
 & \vdots & \vdots & \vdots &
\end{array}
$$

Il est facile de voir que dans cette disposition, nous représentons $P_n^{(i,j)}$ dans l'espace par le point de coordonnées $(i, j+n, n)$.

Mais, nous pouvons aussi représenter $P_n^{(i,j)}$ par le point de coordonnées (i,j,n). Dans ce cas, et dans le plan $i = const$, les polynômes sont disposés de la façon suivante :

Deuxième disposition des $P_n^{(i,j)}$.

$$\begin{array}{ccccc} & \vdots & \vdots & \vdots & \\ \cdots & P_{n-1}^{(i,j-1)} & P_n^{(i,j-1)} & P_{n+1}^{(i,j-1)} & \cdots \\ \cdots & P_{n-1}^{(i,j)} & P_n^{(i,j)} & P_{n+1}^{(i,j)} & \cdots \\ \cdots & P_{n-1}^{(i,j+1)} & P_n^{(i,j+1)} & P_{n+1}^{(i,j+1)} & \cdots \\ & \vdots & \vdots & \vdots & \end{array}$$

Nous laissons au lecteur le soin d'écrire dans la figure 2.3 le nom des polynômes représentés par des •, dans le cas de la première et de la deuxième dispositions.

Remarquons que les polynômes $P_{n-1}^{(l,j+2)}$ et $P_{n+1}^{(l,j-2)}$, $l = i-1, i, i+1$, qui sont adjacents consécutifs à $P_n^{(i,j)}$ dans la première disposition, ne le sont pas dans la deuxième; et que les polynômes $P_{n-1}^{(l,j-1)}$ et $P_{n+1}^{(l,j+1)}$, $l = i-1, i, i+1$, qui sont adjacents consécutifs à $P_n^{(i,j)}$ dans la deuxième disposition, ne le sont pas dans la première. Les polynômes restants sont adjacents consécutifs à $P_n^{(i,j)}$ dans les deux dispositions.

Première disposition des $P_n^{(i,j)}$.

$$\begin{array}{ccccc} & \vdots & & & \\ \cdots & P_{n-1}^{(i,j-1)} & \vdots & \vdots & \\ \cdots & P_{n-1}^{(i,j)} & P_n^{(i,j-1)} & P_{n+1}^{(i,j-2)} & \cdots \\ \cdots & P_{n-1}^{(i,j+1)} & P_n^{(i,j)} & P_{n+1}^{(i,j-1)} & \cdots \\ \cdots & P_{n-1}^{(i,j+2)} & P_n^{(i,j+1)} & P_{n+1}^{(i,j)} & \cdots \\ & \vdots & \vdots & P_{n+1}^{(i,j+1)} & \cdots \\ & & & \vdots & \end{array}$$

Deuxième disposition des $P_n^{(i,j)}$.

$$\begin{array}{ccccc} & & & \vdots & \\ & \vdots & \vdots & P_{n+1}^{(i,j-2)} & \cdots \\ \cdots & P_{n-1}^{(i,j-1)} & P_n^{(i,j-1)} & P_{n+1}^{(i,j-1)} & \cdots \\ \cdots & P_{n-1}^{(i,j)} & P_n^{(i,j)} & P_{n+1}^{(i,j)} & \cdots \\ \cdots & P_{n-1}^{(i,j+1)} & P_n^{(i,j+1)} & P_{n+1}^{(i,j+1)} & \cdots \\ \cdots & P_{n-1}^{(i,j+2)} & \vdots & \vdots & \\ & \vdots & & & \end{array}$$

Ecrivons maintenant la définition de $P_n^{(k)}$ comme rapport de déterminants :

$$P_n^{(k)} = \begin{vmatrix} L(x^k) & \cdots & L(x^{k+n}) \\ \cdots & & \cdots \\ L(x^{k+n-1}) & \cdots & L(x^{k+2n-1}) \\ 1 & \cdots & x^n \end{vmatrix} / H_n^{(k)},$$

où

$$H_n^{(k)} = \begin{vmatrix} L(x^k) & \cdots & L(x^{k+n-1}) \\ \cdots & & \cdots \\ L(x^{k+n-1}) & \cdots & L(x^{k+2n-2}) \end{vmatrix}$$

représente les déterminants de Hankel.

La fonctionnelle L est définie sur l'espace des polynômes complexes, donc L en principe n'est pas définie pour les puissances négatives de la variable. Cependant, nous pouvons supposer que

$$L(x^k) = 0 \ , \forall k < 0,$$

ce qui nous permet de considérer des déterminants de Hankel $H_n^{(k)}$, avec des valeurs négatives de l'indice k. Il est évident que $H_n^{(k)} = 0 \ , \forall k < -n$, mais les déterminants $H_n^{(k)}$, pour les valeurs de $k \geq -n$, ne sont pas nécessairement nuls. Si nous supposons que $H_n^{(k)} \neq 0 \ , \forall n \geq 0, \forall k \geq -n$, nous pouvons définir les polynômes $P_n^{(k)}$, pour les mêmes valeurs de n et k.

Voyons que dans le cas des polynômes biorthogonaux cet artifice n'a pas d'intérêt.

En effet, si nous supposons que

$$L_i(x^j) = 0 \ , \forall i < 0 \text{ ou } j < 0,$$

alors $D_n^{(i,j)} = 0 \ , \forall i < 0$ ou $j < 0$, puisqu'il y aura au moins une ligne ou une colonne de valeurs nulles dans le déterminant $D_n^{(i,j)}$, et nous ne pourrons pas définir $P_n^{(i,j)}$ pour les valeurs de i ou j négatives.

Ceci implique que dans la première représentation dans chaque plan $i = const$, il n'y a pas de polynômes au dessus de la bissectrice $j = 0$.

Première disposition des $P_n^{(i,j)}$.

$$\begin{array}{ccccc} & \vdots & \vdots & \vdots & \\ \cdots & P_{n-1}^{(i,0)} & - & - & \cdots \\ \cdots & P_{n-1}^{(i,1)} & P_n^{(i,0)} & - & \cdots \\ \cdots & P_{n-1}^{(i,2)} & P_n^{(i,1)} & P_{n+1}^{(i,0)} & \cdots \\ & \vdots & \vdots & \vdots & \end{array}$$

Optons pour la deuxième disposition des polynômes biorthogonaux dans l'espace. Dans cette disposition deux polynômes biorthogonaux adjacents $P_{n_1}^{(i_1,j_1)}$ et $P_{n_2}^{(i_2,j_2)}$ sont appelés adjacent consécutifs ssi $|i_1 - i_2| \leq 1$, $|j_1 - j_2| \leq 1$ et $|n_1 - n_2| \leq 1$. Nous dirons aussi que $P_{n_1}^{(i_1,j_1)}$ et $P_{n_2}^{(i_2,j_2)}$ sont voisins.

2.2.1 Déduction

Le but de cette section est de déduire toutes les relations qui permettent le calcul de $P_n^{(i,j)}$ à partir de deux polynômes biorthogonaux adjacents consécutifs à $P_n^{(i,j)}$. C'est-à-dire, nous voulons trouver toutes les relations de la forme :

$$P_n^{(i,j)}(x) = f_1(x)P_{n_1}^{(i_1,j_1)}(x) + f_2(x)P_{n_2}^{(i_2,j_2)}(x), \tag{2.11}$$

où les indices i_1 et i_2 ne peuvent prendre que les valeurs $i-1$, i ou $i+1$; les indices j_1 et j_2 ne peuvent prendre que les valeurs $j-1$, j ou $j+1$; les degrés n_1 et n_2 ne peuvent prendre que les valeurs $n-1$, n et $n+1$; et les facteurs f_1 et f_2 sont du type :

$$f_1(x) = \sum_{m_1=-k_1}^{k_2} a_{m_1}x^{m_1} \quad , \quad f_2(x) = \sum_{m_2=-k_3}^{k_4} b_{m_2}x^{m_2},$$

où a_{m_1} et b_{m_2} sont des nombres complexes, et k_1, k_2, k_3 et k_4 sont des entiers non négatifs. Il est évident que les trois polynômes qui interviennent dans la relation doivent être différents.

Cette relation existe ssi le membre droite de (2.11) représente un polynôme unitaire, de degré n, qui satisfait les conditions de biorthogonalité (2.1).

Nous nous situons dans le cas général où la seule hypothèse qui est faite sur les fonctionnelles L_i est qu'elles sont linéairement indépendantes. Nous ne considérons pas les relations qui nous obligent à admettre des conditions suplémentaires sur les fonctionnelles.

Voyons quel parcours nous devons suivre pour déterminer les valeurs des coefficients a_{m_1} et b_{m_2}, et des indices i_1, i_2, j_1, j_2, n_1 et n_2.

Commençons par écrire les conditions de biorthogonalité :

$$\sum_{m_1=-k_1}^{k_2} a_{m_1} L_p(x^{j+m_1}P_{n_1}^{(i_1,j_1)}) + \sum_{m_2=-k_3}^{k_4} b_{m_2} L_p(x^{j+m_2}P_{n_2}^{(i_2,j_2)}) = 0, \tag{2.12}$$

$$p = i, \ldots, i+n-1.$$

Dans le cas général, si $j_1 \neq j+m_1$ ou $j_2 \neq j+m_2$, nous ne pouvons pas utiliser les conditions de biorthogonalité de $P_{n_1}^{(i_1,j_1)}$ ou $P_{n_2}^{(i_2,j_2)}$, respectivement. Et alors, les coefficients a_{m_1} et b_{m_2} devont satisfaire le système (2.12) constitué par n équations linéaires homogènes, où n est quelconque. Supposer que ce système est consistent, cela équivaut à admettre des conditions suplémentaires sur les fonctionnelles L_i, ce que nous excluons.

Supposons que $j_1 = j+m_1$ et $j_2 = j+m_2$. Ceci implique que $\exists! m_1 : a_{m_1} \neq 0$ et $\exists! m_2 : b_{m_2} \neq 0$, sinon j_1 devait être égal à deux valeurs différents, de même que j_2. En outre, comme j_1 et j_2 ne prennent que les valeurs $j-1$, j et $j+1$, les valeurs de m_1 et m_2 tels que a_{m_1} et b_{m_2} soient non nuls doivent être choisies parmi -1, 0 et 1. Donc,

$$f_1(x) = ax^{m_1} \text{ et } f_2(x) = bx^{m_2},$$

avec a et b non nuls, et m_1 et m_2 égaux à -1, 0 ou 1.

Alors, la relation (2.11) et les conditions de biorthogonalité (2.12) s'écrivent de la façon suivante :

$$P_n^{(i,j)}(x) = ax^{m_1} P_{n_1}^{(i_1,j+m_1)}(x) + bx^{m_2} P_{n_2}^{(i_2,j+m_2)}(x), \tag{2.13}$$

$$aL_p(x^{j+m_1} P_{n_1}^{(i_1,j+m_1)}) + bL_p(x^{j+m_2} P_{n_2}^{(i_2,j+m_2)}) = 0\,, \tag{2.14}$$

$$p = i, \ldots, i+n-1.$$

Il reste à déterminer a, b, i_1, i_2, m_1, m_2, n_1 et n_2.
Commençons par fixer les degrés n_1 et n_2. Il y a plusieurs cas à considérer :

1. Les deux degrés sont égaux à $n-1$: $n_1 = n_2 = n-1$.
2. Les deux degrés sont égaux à n : $n_1 = n_2 = n$.
3. Les deux degrés sont égaux à $n+1$: $n_1 = n_2 = n+1$.
4. L'un des degrés est égal à n et l'autre est égal à $n-1$: par exemple $n_1 = n$, $n_2 = n-1$.
5. L'un des degrés est égal à n et l'autre est égal à $n+1$: par exemple $n_1 = n$, $n_2 = n+1$.
6. L'un des degrés est égal à $n+1$ et l'autre est égal à $n-1$: par exemple $n_1 = n+1$, $n_2 = n-1$.

Dans chaque cas, les valeurs de n_1 et n_2 vont nous permettre de déterminer les valeurs possibles de m_1 et m_2, de telle sorte que le deuxième membre de (2.13) puisse représenter un polynôme de degré n. Pour chaque paire de valeurs possibles de m_1 et m_2, le fait que $P_n^{(i,j)}$, $P_{n_1}^{(i_1,j+m_1)}$ et $P_{n_2}^{(i_2,j+m_2)}$ soient des polynômes unitaires va nous permettre de fixer un des coefficients a ou b, ou éventuellement les deux.
Il reste à déterminer les indices i_1 et i_2 et un des coefficients a ou b, si les deux n'ont pas encore été déterminés.
Considérons maintenant toutes les valeurs possibles des indices i_1 et i_2.
Les conditions de biorthogonalité des polynômes $P_{n_1}^{(i_1,j+m_1)}$ et $P_{n_2}^{(i_2,j+m_2)}$:

$$L_p(x^{j+m_1} P_{n_1}^{(i_1,j+m_1)}) = 0 \quad , p = i_1, \ldots, i_1 + n_1 - 1,$$

$$L_p(x^{j+m_2} P_{n_2}^{(i_2,j+m_2)}) = 0 \quad , p = i_2, \ldots, i_2 + n_2 - 1,$$

impliquent que les conditions de biorthogonalité (2.14) sont vérifiées pour les valeurs de $p = p_1, \ldots, p_2$, où

$$p_1 = max\{i_1, i_2\} \quad \text{et} \quad p_2 = min\{i_1 + n_1 - 1, i_2 + n_2 - 1\}.$$

Nous remarquons que les conditions qui restent sont en nombre fixe.
A partir des valeurs possibles de i_1, i_2, n_1 et n_2, nous savons que $p_1 \geq i-1$ et $p_2 \leq i+n+1$.
Voyons que les cas $p_1 = i-1$, $p_2 = i+n$ et $p_2 = i+n+1$ sont impossibles.

Si $p_1 = i-1$ ou $p_2 = i+n$, alors nous obtenons

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 \text{ ou } L_{i+n}(x^j P_n^{(i,j)}) = 0,$$

ce qui est absurde, puisque nous sommes dans le cas normal.

Si $p_2 = i+n+1$, ce qui est équivalent à $i_1 = i_2 = i+1$ et $n_1 = n_2 = n+1$, alors m_1 et m_2 doivent être différents, sinon $P_{n_1}^{(i_1,j+m_1)} \equiv P_{n_2}^{(i_2,j+m_2)}$ ce qui est absurde. Nous avons trois possibilités pour les valeurs des indices m_1 et m_2, à savoir: un des indices est égal à -1 et l'autre est égal à 0, par exemple $m_1 = -1$ et $m_2 = 0$; un des indices est égal à -1 et l'autre est égal à 1, par exemple $m_1 = -1$ et $m_2 = 1$; et enfin un des indices est égal à 0 et l'autre est égal à 1, par exemple $m_1 = 0$ et $m_2 = 1$. Il est facile de voir que dans toutes les trois possiblités, le deuxième membre de (2.13) représente un polynôme de degré supérieur à n, ce qui est absurde.

Donc, $p_1 \geq i$ et $p_2 \leq i+n-1$.

Si $p_1 = i$ et $p_2 = i+n-1$, alors toutes les conditions de biorthogonalité (2.14) sont vérifiées, et la relation (2.13) existe.

Si $p_1 \neq i$ or $p_2 \neq i+n-1$, les conditions de biorthogonalité correspondantes aux valeurs de

$$p = i, \ldots, p_1 - 1 \quad \text{et} \quad p = p_2 + 1, \ldots, i+n-1$$

ne sont pas assurées.

Dans le cas où il reste un des coefficients a ou b à déterminer, il doit satisfaire le système constitué par les conditions de biorthogonalité correspondantes à ces valeurs de p. Si tel système est impossible, la relation (2.13) n'existe pas.

Et, nous finissons ici le parcours à suivre.

Avant de commencer à faire cette étude, établissons quelque notation.

Dénotons les coefficients des polynômes biorthogonaux $P_n^{(i,j)}$ dans la base canonique par $a_k^{(i,j,n)}$, $k = 0, \ldots, n$. Comme ces polynômes sont unitaires

$$a_n^{(i,j,n)} = 1.$$

L'égalité (2.7) donne l'expression du terme indépendant

$$a_0^{(i,j,n)} = (-1)^n D_n^{(i,j+1)} / D_n^{(i,j)}. \tag{2.15}$$

Soit $0 \leq i_1 < \cdots < i_n$ et $0 \leq j_1 < \cdots < j_n$, alors nous dénotons

$$\begin{pmatrix} i_1 & \cdots & i_n \\ j_1 & \cdots & j_n \end{pmatrix} = \begin{vmatrix} L_{i_1}(x^{j_1}) & \cdots & L_{i_1}(x^{j_n}) \\ \cdots & & \cdots \\ L_{i_n}(x^{j_1}) & \cdots & L_{i_n}(x^{j_n}) \end{vmatrix}.$$

Nous aurons besoin plus tard de l'expression de

$$a_{n-1}^{(i,j,n)} = -\begin{pmatrix} i & \cdots & i+n-2 & i+n-1 \\ j & \cdots & j+n-2 & j+n \end{pmatrix} / D_n^{(i,j)}. \tag{2.16}$$

Une première observation : voyons que les cas où un seul des indices m_1 ou m_2 est égal à -1 sont toujours impossibles.

Supposons, par exemple, que $m_1 = -1$ et $m_2 \neq -1$. Le cas $m_2 = -1$ et $m_1 \neq -1$ est absolument analogue.

Alors, le relation (2.13) s'écrit de la façon suivante :

$$P_n^{(i,j)}(x) = \frac{a}{x} P_{n_1}^{(i_1,j-1)}(x) + bx^{m_2} P_{n_2}^{(i_2,j+m_2)}(x).$$

Nous constatons que dans le deuxième membre de cette relation il apparaît

$$a a_0^{(i_1,j-1,n_1)} \frac{1}{x},$$

qui devra disparaître, puisque $P_n^{(i,j)}$ est un polynôme. Or ceci est impossible, parce que $a \neq 0$ et de (2.15) $a_0^{(i_1,j-1,n_1)} \neq 0$.

Commençons par fixer les degrés n_1 et n_2.

1 — $n_1 = n_2 = n - 1$.

<table>
<tr><td colspan="6">$n_1 = n_2 = n-1$</td></tr>
<tr><td>$m_1 = 1$
$m_2 = -1$</td><td colspan="2">$m_1 = 1$
$m_2 = 1$</td><td colspan="3">$m_1 = 1$, $m_2 = 0$</td></tr>
<tr><td>–</td><td colspan="2">$b = 1 - a$</td><td colspan="3">$a = 1$</td></tr>
<tr><td>–</td><td colspan="2">$i_1 = i - 1$</td><td colspan="3">$i_1 = i - 1$</td></tr>
<tr><td>–</td><td>$i_2 = i$</td><td>$i_2 = i + 1$</td><td>$i_2 = i - 1$</td><td>$i_2 = i$</td><td>$i_2 = i + 1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td colspan="2">$i_1 = i$</td><td colspan="3">$i_1 = i$</td></tr>
<tr><td>–</td><td colspan="2">$i_2 = i + 1$</td><td>$i_2 = i - 1$</td><td>$i_2 = i$</td><td>$i_2 = i + 1$</td></tr>
<tr><td>–</td><td colspan="2">–</td><td>–</td><td>F_1</td><td>–</td></tr>
<tr><td>–</td><td colspan="2">–</td><td colspan="3">$i_1 = i + 1$</td></tr>
<tr><td>–</td><td colspan="2">–</td><td>$i_2 = i - 1$</td><td>$i_2 = i$</td><td>$i_2 = i + 1$</td></tr>
<tr><td>–</td><td colspan="2">–</td><td>–</td><td>–</td><td>F_2</td></tr>
</table>

Expliquons en détail les différentes étapes représentées dans ce tableau.

Nous commençons par fixer les degrés $n_1 = n_2 = n-1$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = ax^{m_1} P_{n-1}^{(i_1,j+m_1)} + bx^{m_2} P_{n-1}^{(i_2,j+m_2)}$$

A partir des degrés des trois polynômes, nous avons trois possibilités pour les valeurs de m_1 et m_2, à savoir: une de ces valeurs est égal à 1 et l'autre est égal à -1, par exemple $m_1 = 1$ et $m_2 = -1$ (cas 1.1); les deux valeurs sont égaux à 1, $m_1 = m_2 = 1$ (cas 1.2); ou

une des valeurs est égal à 1 et l'autre est égal à 0, par exemple $m_1 = 1$ et $m_2 = 0$ (cas 1.3).

1.1 — $m_1 = 1$, $m_2 = -1$.

Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

1.2 — $m_1 = m_2 = 1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_{n-1}^{(i_1,j+1)} + bxP_{n-1}^{(i_2,j+1)}. \tag{2.17}$$

A partir de cette relation, nous voyons que le coefficient principal de $P_n^{(i,j)}$ est égal à $a + b$. Comme $P_n^{(i,j)}$ est unitaire $a + b = 1$, ou

$$b = 1 - a.$$

Considérons maintenant tous les valeurs possibles des indices i_1 et i_2. Remarquons que i_1 et i_2 doivent être différents, sinon les deux polynômes du membre droit de la relation (2.17) seraient égaux. Il reste trois cas possibles, à savoir: un des indices est égal à $i - 1$ et l'autre est égal à i, par exemple $i_1 = i - 1$ et $i_2 = i$ (cas 1.2.1); un des indices est égal à $i - 1$ et l'autre est égal à $i + 1$, par exemple $i_1 = i - 1$ et $i_2 = i + 1$ (cas 1.2.2); et enfin un des indices est égal à i et l'autre est égal à $i + 1$, par exemple $i_1 = i$ et $i_2 = i + 1$ (cas 1.2.3).

1.2.1 — $i_1 = i - 1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i + n - 2$ et $p = i + n - 1$.
Voyons si la relation

$$P_n^{(i,j)} = axP_{n-1}^{(i-1,j+1)} + (1 - a)xP_{n-1}^{(i,j+1)} \tag{2.18}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} aL_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + (1 - a)L_{i+n-2}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \\ aL_{i+n-1}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + (1 - a)L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \end{cases}.$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-2}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \quad , \quad L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) \neq 0 \text{ et}$$

$$L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) \neq 0,$$

ce qui implique

$$\begin{cases} a = 0 \\ eq.imp. \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.18) n'existe pas.

1.2.2 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = axP_{n-1}^{(i-1,j+1)} + (1-a)xP_{n-1}^{(i+1,j+1)} \tag{2.19}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} aL_i(x^{j+1}P_{n-1}^{(i-1,j+1)}) + (1-a)L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \\ aL_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + (1-a)L_{i+n-2}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \\ aL_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) + (1-a)L_{i+n-1}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i-1,j+1)}) = 0\,, \forall n \geq 3 \quad , \quad L_{i+n-2}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0\ \forall n \geq 3\,,$$

$$L_{i+n-1}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \quad , \quad L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) \neq 0\,,$$

$$L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) \neq 0 \text{ et } L_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} a = 1 \\ a = 0 \\ a = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.19) n'existe pas.

1.2.3 — $i_1 = i$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = axP_{n-1}^{(i,j+1)} + (1-a)xP_{n-1}^{(i+1,j+1)} \tag{2.20}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} aL_i(x^{j+1}P_{n-1}^{(i,j+1)}) + (1-a)L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \\ aL_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) + (1-a)L_{i+n-1}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \quad , \quad L_{i+n-1}(x^{j+1}P_{n-1}^{(i+1,j+1)}) = 0 \, ,$$

$$L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) \neq 0,$$

ce qui implique

$$\begin{cases} a = 1 \\ a = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.20) n'existe pas.

1.3 — $m_1 = 1$, $m_2 = 0$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_{n-1}^{(i_1,j+1)} + bP_{n-1}^{(i_2,j)}. \tag{2.21}$$

A partir de cette relation, nous voyons que le coefficient principal de $P_n^{(i,j)}$ est égal à a. Comme $P_n^{(i,j)}$ est unitaire

$$a = 1.$$

Représentons dans le tableau suivant tous les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 1.3.1) $i_2 = i$ (cas 1.3.2) $i_2 = i+1$ (cas 1.3.3)
$i_1 = i$	$i_2 = i-1$ (cas 1.3.4) $i_2 = i$ (cas 1.3.5) $i_2 = i+1$ (cas 1.3.6)
$i_1 = i+1$	$i_2 = i-1$ (cas 1.3.7) $i_2 = i$ (cas 1.3.8) $i_2 = i+1$ (cas 1.3.9)

1.3.1 — $i_1 = i-1$, $i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-3$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 \, ,$$

ce qui est absurde (voir (2.8)).

1.3.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i+n-2$ et $p = i+n-1$. Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i-1,j+1)} + bP_{n-1}^{(i,j)} \tag{2.22}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + bL_{i+n-2}(x^jP_{n-1}^{(i,j)}) = 0 \\ L_{i+n-1}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i,j)}) = 0 \end{cases}.$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-2}(x^jP_{n-1}^{(i,j)}) = 0 \quad , \quad L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) \neq 0 \text{ et}$$

$$L_{i+n-1}(x^jP_{n-1}^{(i,j)}) \neq 0 \, ,$$

ce qui implique

$$\begin{cases} eq.imp \\ - \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.22) n'existe pas.

1.3.3 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i-1,j+1)} + bP_{n-1}^{(i+1,j)} \tag{2.23}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} L_i(x^{j+1}P_{n-1}^{(i-1,j+1)}) + bL_i(x^jP_{n-1}^{(i+1,j)}) = 0 \\ L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + bL_{i+n-2}(x^jP_{n-1}^{(i+1,j)}) = 0 \\ L_{i+n-1}(x^{j+1}P_{n-1}^{(i-1,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i+1,j)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i-1,j+1)}) = 0 \, , \forall n \geq 3 \, , \; L_{i+n-2}(x^jP_{n-1}^{(i+1,j)}) = 0 \, , \forall n \geq 3 \, ,$$

$$L_{i+n-1}(x^jP_{n-1}^{(i+1,j)}) = 0 \quad , \quad L_i(x^jP_{n-1}^{(i+1,j)}) \neq 0 \, ,$$

$$L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) \neq 0 \, ,$$

ce qui implique

$$\begin{cases} b = 0 \\ eq.imp \\ - \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.23) n'existe pas.

1.3.4 — $i_1 = i$, $i_2 = i-1$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i+n-2$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i,j+1)} + bP_{n-1}^{(i-1,j)} \tag{2.24}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} L_{i+n-2}(x^{j+1}P_{n-1}^{(i,j+1)}) + bL_{i+n-2}(x^jP_{n-1}^{(i-1,j)}) = 0 \\ L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i-1,j)}) = 0 \end{cases}.$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-2}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \quad , \quad L_{i+n-2}(x^jP_{n-1}^{(i-1,j)}) \neq 0 \,,$$

$$L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) \neq 0 \,,$$

ce qui implique

$$\begin{cases} b = 0 \\ eq.imp \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.24) n'existe pas.

1.3.5 — $i_1 = i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i,j+1)} + bP_{n-1}^{(i,j)} \tag{2.25}$$

existe.
Ecrivons la condition de biorthogonalité correspondante à $p = i+n-1$:

$$L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i,j)}) = 0. \tag{2.26}$$

De (2.6) et (2.9), nous savons que

$$L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) = \frac{D_n^{(i,j+1)}}{D_{n-1}^{(i,j+1)}} \neq 0 \text{ et}$$

$$L_{i+n-1}(x^jP_{n-1}^{(i,j)}) = \frac{D_n^{(i,j)}}{D_{n-1}^{(i,j)}} \neq 0 \,.$$

Si nous prenons

$$b = -\frac{L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)})}{L_{i+n-1}(x^jP_{n-1}^{(i,j)})} = -\frac{D_n^{(i,j+1)}D_{n-1}^{(i,j)}}{D_{n-1}^{(i,j+1)}D_n^{(i,j)}},$$

la condition (2.26) est satisfaite, et la relation (2.25) vérifiée.
Dénotons la relation trouvée par F_1.

$$F_1 \ : \ P_n^{(i,j)}(x) = xP_{n-1}^{(i,j+1)}(x) - \frac{D_n^{(i,j+1)}D_{n-1}^{(i,j)}}{D_{n-1}^{(i,j+1)}D_n^{(i,j)}}P_{n-1}^{(i,j)}(x)$$

1.3.6 — $i_1 = i$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i,j+1)} + bP_{n-1}^{(i+1,j)} \tag{2.27}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} L_i(x^{j+1}P_{n-1}^{(i,j+1)}) + bL_i(x^jP_{n-1}^{(i+1,j)}) = 0 \\ L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i+1,j)}) = 0 \end{cases}$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \quad , \quad L_{i+n-1}(x^jP_{n-1}^{(i+1,j)}) = 0 \, ,$$

$$L_i(x^jP_{n-1}^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) \neq 0 \, ,$$

ce qui implique

$$\begin{cases} b = 0 \\ eq.imp \end{cases} ,$$

qui est un système impossible. Conclusion : la relation (2.27) n'existe pas.

1.3.7 — $i_1 = i+1$, $i_2 = i-1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = xP_{n-1}^{(i+1,j+1)} + bP_{n-1}^{(i-1,j)} \tag{2.28}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) + bL_i(x^jP_{n-1}^{(i-1,j)}) = 0 \\ L_{i+n-2}(x^{j+1}P_{n-1}^{(i+1,j+1)}) + bL_{i+n-2}(x^jP_{n-1}^{(i-1,j)}) = 0 \\ L_{i+n-1}(x^{j+1}P_{n-1}^{(i+1,j+1)}) + bL_{i+n-1}(x^jP_{n-1}^{(i-1,j)}) = 0 \end{cases} .$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_{n-1}^{(i-1,j)}) = 0 \,, \forall n \geq 3 \quad , \quad L_{i+n-2}(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \,,$$

$$L_{i+n-1}(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \quad , \quad L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) \neq 0 \,,$$

$$L_{i+n-2}(x^j P_{n-1}^{(i-1,j)}) \neq 0 \,,$$

ce qui implique

$$\begin{cases} eq.imp \\ b = 0 \\ - \end{cases} ,$$

qui est un système impossible. Conclusion : la relation (2.28) n'existe pas.

1.3.8 — $i_1 = i + 1$, $i_2 = i$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = x P_{n-1}^{(i+1,j+1)} + b P_{n-1}^{(i,j)} \tag{2.29}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i + n - 1$:

$$\begin{cases} L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) + b L_i(x^j P_{n-1}^{(i,j)}) = 0 \\ L_{i+n-1}(x^{j+1} P_{n-1}^{(i+1,j+1)}) + b L_{i+n-1}(x^j P_{n-1}^{(i,j)}) = 0 \end{cases} .$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_{n-1}^{(i,j)}) = 0 \quad , \quad L_{i+n-1}(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \,,$$

$$L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) \neq 0 \quad , \quad L_{i+n-1}(x^j P_{n-1}^{(i,j)}) \neq 0 \,,$$

ce qui implique

$$\begin{cases} eq.imp \\ b = 0 \end{cases} ,$$

qui est un système impossible. Conclusion : la relation (2.29) n'existe pas.

1.3.9 — $i_1 = i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = x P_{n-1}^{(i+1,j+1)} + b P_{n-1}^{(i+1,j)} \tag{2.30}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) + bL_i(x^jP_{n-1}^{(i+1,j)}) = 0. \tag{2.31}$$

De (2.5) et (2.8), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i+1,j+1)}) = (-1)^{n-1}\frac{D_n^{(i,j+1)}}{D_{n-1}^{(i+1,j+1)}} \neq 0 \text{ et}$$

$$L_i(x^jP_{n-1}^{(i+1,j)}) = (-1)^{n-1}\frac{D_n^{(i,j)}}{D_{n-1}^{(i+1,j)}} \neq 0 .$$

Si nous prenons

$$b = -\frac{L_i(x^{j+1}P_{n-1}^{(i+1,j+1)})}{L_i(x^jP_{n-1}^{(i+1,j)})} = -\frac{D_n^{(i,j+1)}D_{n-1}^{(i+1,j)}}{D_{n-1}^{(i+1,j+1)}D_n^{(i,j)}},$$

la condition (2.31) est satisfaite, et la relation (2.30) vérifiée.
Dénotons la relation trouvée par F_2.

$$F_2 \ : \ P_n^{(i,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) - \frac{D_n^{(i,j+1)}D_{n-1}^{(i+1,j)}}{D_{n-1}^{(i+1,j+1)}D_n^{(i,j)}}P_{n-1}^{(i+1,j)}(x)$$

2 — $n_1 = n_2 = n$.

$n_1 = n_2 = n$		
$m_1 = 0$ $m_2 = -1$	$m_1 = 0$ $m_2 = 0$	$m_1 = 1$ $m_2 = 1$
–	$b = 1 - a$	–
–	$i_1 = i - 1$ $i_2 = i + 1$	–
–	–	–

Expliquons en détail les différentes étapes représentées dans ce tableau.
Nous commençons par fixer les degrés $n_1 = n_2 = n$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = ax^{m_1}P_n^{(i_1,j+m_1)} + bx^{m_2}P_n^{(i_2,j+m_2)}$$

A partir des degrés des trois polynômes, nous avons trois possibilités pour les valeurs de m_1 et m_2, à savoir: une de ces valeurs est égal à 0 et l'autre est égal à -1, par exemple

$m_1 = 0$ et $m_2 = -1$ (cas 2.1); les deux valeurs sont égaux à 0, $m_1 = m_2 = 0$ (cas 2.2); et les deux valeurs sont égaux à 1, $m_1 = m_2 = 1$ (cas 2.3).

2.1 — $m_1 = 0$, $m_2 = -1$.

Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

2.2 — $m_1 = m_2 = 0$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = aP_n^{(i_1,j)} + bP_n^{(i_2,j)}. \tag{2.32}$$

A partir de cette relation, nous voyons que le coefficient principal de $P_n^{(i,j)}$ est égal à $a + b$. Comme $P_n^{(i,j)}$ est unitaire $a + b = 1$, ou

$$b = 1 - a.$$

Considérons maintenant toutes les valeurs possibles des indices i_1 et i_2. Puisque les trois polynômes qui interviennent dans la relation doivent être différents, $i_1 \neq i_2$, $i_1 \neq i$ et $i_2 \neq i$. Il reste un seul cas à étudier, à savoir: un des indices est égal à $i - 1$ et l'autre est égal à $i + 1$, par exemple $i_1 = i - 1$ et $i_2 = i + 1$ (cas 2.2.1).

2.2.1 — $i_1 = i - 1$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i + n - 1$.
Voyons si la relation

$$P_n^{(i,j)} = aP_n^{(i-1,j)} + (1 - a)P_n^{(i+1,j)} \tag{2.33}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i + n - 1$:

$$\begin{cases} aL_i(x^j P_n^{(i-1,j)}) + (1 - a)L_i(x^j P_n^{(i+1,j)}) = 0 \\ aL_{i+n-1}(x^j P_n^{(i-1,j)}) + (1 - a)L_{i+n-1}(x^j P_n^{(i+1,j)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_n^{(i-1,j)}) = 0\,, \forall n \geq 2 \quad , \quad L_{i+n-1}(x^j P_n^{(i+1,j)}) = 0\,, \forall n \geq 2\,,$$

$$L_i(x^j P_n^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^j P_n^{(i-1,j)}) \neq 0,$$

ce qui implique

$$\begin{cases} a = 1 \\ a = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.33) n'existe pas.

2.3 — $m_1 = m_2 = 1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_n^{(i_1,j+1)} + bxP_n^{(i_2,j+1)}. \tag{2.34}$$

A partir de cette relation, nous obtenons

$$P_n^{(i,j)}(0) = 0,$$

ce qui est absurde (voir (2.10)). Conclusion : les relations du type (2.34) n'existent pas.

3 — $n_1 = n_2 = n + 1$.

<table>
<tr><td colspan="7">$n_1 = n_2 = n+1$</td></tr>
<tr><td colspan="3">$m_1 = m_2 = -1$</td><td colspan="3">$m_1 = m_2 = 0$</td><td>$m_1 = m_2 = 1$</td></tr>
<tr><td colspan="3">$a = \frac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_2,j-1,n+1)} - a_0^{(i_1,j-1,n+1)}}$
$b = 1 - a$</td><td colspan="3">$a = \frac{1}{a_n^{(i_1,j,n+1)} - a_n^{(i_2,j,n+1)}}$
$b = -a$</td><td>–
–</td></tr>
<tr><td>$i_1 = i-1$
$i_2 = i$</td><td>$i_1 = i-1$
$i_2 = i+1$</td><td>$i_1 = i$
$i_2 = i+1$</td><td>$i_1 = i-1$
$i_2 = i$</td><td>$i_1 = i-1$
$i_2 = i+1$</td><td>$i_1 = i$
$i_2 = i+1$</td><td>–
–</td></tr>
<tr><td>F_3</td><td>–</td><td>–</td><td>F_4</td><td>–</td><td>–</td><td>–</td></tr>
</table>

Expliquons en détail les différentes étapes représentées dans ce tableau.
Nous commençons par fixer les degrés $n_1 = n_2 = n+1$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = ax^{m_1}P_{n+1}^{(i_1,j+m_1)} + bx^{m_2}P_{n+1}^{(i_2,j+m_2)}$$

A partir des degrés des trois polynômes, nous savons que $m_1 = m_2 = -1$ (cas 3.1), $m_1 = m_2 = 0$ (cas 3.2) ou $m_1 = m_2 = 1$ (cas 3.3).

3.1 — $m_1 = m_2 = -1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = \frac{a}{x}P_{n+1}^{(i_1,j-1)} + \frac{b}{x}P_{n+1}^{(i_2,j-1)}. \tag{2.35}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = (a+b)x^n + \ldots + (aa_0^{(i_1,j-1,n+1)} + ba_0^{(i_2,j-1,n+1)})\frac{1}{x}$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} a+b=1 \\ aa_0^{(i_1,j-1,n+1)} + ba_0^{(i_2,j-1,n+1)} = 0 \end{cases},$$

ce qui est équivalent à

$$\begin{cases} a = \dfrac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_2,j-1,n+1)} - a_0^{(i_1,j-1,n+1)}} \\ b = -\dfrac{a_0^{(i_1,j-1,n+1)}}{a_0^{(i_2,j-1,n+1)} - a_0^{(i_1,j-1,n+1)}} \end{cases}.$$

Considérons maintenant tous les valeurs possibles des indices i_1 et i_2. Remarquons que i_1 et i_2 doivent être différents, sinon les deux polynômes du membre droit de la relation (2.35) seraient égaux. Il reste trois cas possibles, à savoir: un des indices est égal à $i-1$ et l'autre est égal à i, par exemple $i_1 = i-1$ et $i_2 = i$ (cas 3.1.1); un des indices est égal à $i-1$ et l'autre est égal à $i+1$, par exemple $i_1 = i-1$ et $i_2 = i+1$ (cas 3.1.2); et enfin un des indices est égal à i et l'autre est égal à $i+1$, par exemple $i_1 = i$ et $i_2 = i+1$ (cas 3.1.3).

3.1.1 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = \frac{a}{x} P_{n+1}^{(i-1,j-1)} + \frac{b}{x} P_{n+1}^{(i,j-1)},$$

où

$$\begin{cases} a = \dfrac{a_0^{(i,j-1,n+1)}}{a_0^{(i,j-1,n+1)} - a_0^{(i-1,j-1,n+1)}} \\ b = -\dfrac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i,j-1,n+1)} - a_0^{(i-1,j-1,n+1)}} \end{cases}.$$

Ecrivons les coefficients a et b comme rapports de déterminants.
En utilisant (2.15), nous pouvons écrire

$$\begin{cases} a = \dfrac{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)}}{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)} - D_{n+1}^{(i,j-1)} D_{n+1}^{(i-1,j)}} \\ b = -\dfrac{D_{n+1}^{(i,j-1)} D_{n+1}^{(i-1,j)}}{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)} - D_{n+1}^{(i,j-1)} D_{n+1}^{(i-1,j)}} \end{cases}.$$

D'après l'identité de Sylvester (E.1)

$$D_{n+2}^{(i-1,j-1)} D_n^{(i,j)} = D_{n+1}^{(i-1,j-1)} D_{n+1}^{(i,j)} - D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j-1)},$$

et alors

$$\begin{cases} a = \dfrac{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)}}{D_{n+2}^{(i-1,j-1)} D_n^{(i,j)}} \\ b = -\dfrac{D_{n+1}^{(i,j-1)} D_{n+1}^{(i-1,j)}}{D_{n+2}^{(i-1,j-1)} D_n^{(i,j)}} \end{cases}.$$

Dénotons la relation trouvée par F_3.

$$F_3 \ : \ P_n^{(i,j)}(x) = \frac{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)}}{D_{n+2}^{(i-1,j-1)} D_n^{(i,j)}} \left(\frac{1}{x} P_{n+1}^{(i-1,j-1)}(x) - \frac{1}{x} P_{n+1}^{(i,j-1)}(x) \right)$$

3.1.2 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_{n+1}^{(i-1,j-1)} + \frac{b}{x} P_{n+1}^{(i+1,j-1)}, \tag{2.36}$$

où

$$\begin{cases} a = \dfrac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i+1,j-1,n+1)} - a_0^{(i-1,j-1,n+1)}} \\ b = -\dfrac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i+1,j-1,n+1)} - a_0^{(i-1,j-1,n+1)}} \end{cases},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) + bL_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0. \tag{2.37}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0\,, \forall n \geq 1 \quad \text{et} \quad L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) \neq 0,$$

donc la condition (2.37) est équivalente à $b = 0$, qui est impossible. Conclusion : la relation (2.36) n'existe pas.

3.1.3 — $i_1 = i$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0\,,$$

ce qui est absurde (voir (2.9)).

3.2 — $m_1 = m_2 = 0$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = aP_{n+1}^{(i_1,j)} + bP_{n+1}^{(i_2,j)}. \tag{2.38}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = (a+b)x^{n+1} + (aa_n^{(i_1,j,n+1)} + ba_n^{(i_2,j,n+1)})x^n + \ldots$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} a+b=0 \\ aa_n^{(i_1,j,n+1)} + ba_n^{(i_2,j,n+1)} = 1 \end{cases},$$

ce qui est équivalent à

$$\begin{cases} a = \dfrac{1}{a_n^{(i_1,j,n+1)} - a_n^{(i_2,j,n+1)}} \\ b = -a \end{cases}.$$

Considérons maintenant tous les valeurs possibles des indices i_1 et i_2. Remarquons que i_1 et i_2 doivent être différents, sinon les deux polynômes du membre droit de la relation (2.38) seraient égaux. Il reste trois cas possibles, à savoir: un des indices est égal à $i-1$ et l'autre est égal à i, par exemple $i_1 = i-1$ et $i_2 = i$ (cas 3.2.1); un des indices est égal à $i-1$ et l'autre est égal à $i+1$, par exemple $i_1 = i-1$ et $i_2 = i+1$ (cas 3.2.2); et enfin un des indices est égal à i et l'autre est égal à $i+1$, par exemple $i_1 = i$ et $i_2 = i+1$ (cas 3.2.3).

3.2.1 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = a(P_{n+1}^{(i-1,j)} - P_{n+1}^{(i,j)}),$$

où

$$a = \frac{1}{a_n^{(i-1,j,n+1)} - a_n^{(i,j,n+1)}}.$$

Ecrivons le coefficient a comme rapport de déterminants.
D'aprés l'identité de Sylvester (E.1) et l'égalité (2.16)

$$(-1)^n D_{n+2}^{(i-1,j)} D_n^{(i,j)} = a_n^{(i-1,j,n+1)} D_{n+1}^{(i-1,j)} (-1)^n D_{n+1}^{(i,j)} - (-1)^n D_{n+1}^{(i-1,j)} a_n^{(i,j,n+1)} D_{n+1}^{(i,j)},$$

où $(-1)^n D_{n+2}^{(i-1,j)}$ représente le déterminant $D_{n+2}^{(i-1,j)}$ avec la $(n+1)$—ème colonne placée en première position.
En divisant les deux membres de l'égalité précédente par $(-1)^n D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j)}$, nous obtenons :

$$\frac{D_{n+2}^{(i-1,j)} D_n^{(i,j)}}{D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j)}} = a_n^{(i-1,j,n+1)} - a_n^{(i,j,n+1)},$$

et alors

$$a = \frac{D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j)}}{D_{n+2}^{(i-1,j)} D_n^{(i,j)}}.$$

Dénotons la relation trouvée par F_4.

$$F_4 \ : \ P_n^{(i,j)}(x) = \frac{D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j)}}{D_n^{(i,j)} D_{n+2}^{(i-1,j)}} (P_{n+1}^{(i-1,j)}(x) - P_{n+1}^{(i,j)}(x))$$

3.2.2 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i-1,j)} - P_{n+1}^{(i+1,j)}), \tag{2.39}$$

où

$$a = \frac{1}{a_n^{(i-1,j,n+1)} - a_n^{(i+1,j,n+1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^j P_{n+1}^{(i-1,j)}) - L_i(x^j P_{n+1}^{(i+1,j)})) = 0. \tag{2.40}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^j P_{n+1}^{(i-1,j)}) = 0\ , \forall n \geq 1 \quad \text{et} \quad L_i(x^{j-1} P_{n+1}^{(i+1,j)}) \neq 0,$$

donc la condition (2.40) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.39) n'existe pas.

3.2.3 — $i_1 = i\ ,\ i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0\ ,$$

ce qui est absurde.

3.3 — $m_1 = m_2 = 1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_{n+1}^{(i_1,j+1)} + bxP_{n+1}^{(i_2,j+1)}. \tag{2.41}$$

A partir de cette relation, nous obtenons

$$P_n^{(i,j)}(0) = 0,$$

ce qui est absurde (voir (2.10)). Conclusion : les relations du type (2.41) n'existent pas.

4 — $n_1 = n\ ,\ n_2 = n - 1$.

<table>
<tr><td colspan="8">$n_1 = n\ ,\ n_2 = n-1$</td></tr>
<tr><td>$m_1 = 0$
$m_2 = -1$</td><td>$m_1 = -1$
$m_2 = 1$</td><td colspan="3">$m_1 = 0\ ,\ m_2 = 0$</td><td colspan="3">$m_1 = 0\ ,\ m_2 = 1$</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$a = 1$</td><td colspan="3">$b = 1 - a$</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$i_1 = i-1$</td><td colspan="3">$i_1 = i-1$</td></tr>
<tr><td>–</td><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>F_5</td><td>–</td><td>–</td><td>F_7</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$i_1 = i+1$</td><td colspan="3">$i_1 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>F_6</td><td>–</td><td>–</td><td>F_8</td></tr>
</table>

Expliquons en détail les différentes étapes représentées dans ce tableau.

Nous commençons par fixer les degrés $n_1 = n$ et $n_2 = n-1$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = ax^{m_1} P_n^{(i_1, j+m_1)} + bx^{m_2} P_{n-1}^{(i_2, j+m_2)}.$$

A partir des degrés des trois polynômes, nous avons quatre possibilités pour les valeurs de m_1 et m_2, à savoir: $m_1 = 0$ et $m_2 = -1$ (cas 4.1), $m_1 = -1$ et $m_2 = 1$ (cas 4.2), $m_1 = m_2 = 0$ (cas 4.3), et enfin $m_1 = 0$ et $m_2 = 1$ (cas 4.4).

4.1 — $m_1 = 0\ ,\ m_2 = -1$.

Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

4.2 — $m_1 = -1\ ,\ m_2 = 1$.

Comme $m_1 = -1$ et $m_2 \neq -1$, ce cas est impossible.

4.3 — $m_1 = m_2 = 0$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = aP_n^{(i_1,j)} + bP_{n-1}^{(i_2,j)}. \tag{2.42}$$

A partir de cette relation, nous voyons que le coefficient principal de $P_n^{(i,j)}$ est égal à a. Comme $P_n^{(i,j)}$ est unitaire

$$a = 1.$$

Considérons maintenant tous les valeurs possibles des indices i_1 et i_2. Remarquons que $i_1 \neq i$, sinon $P_n^{(i,j)}$ apparaîtrait dans le deuxième membre de (2.42).

Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 4.3.1) $i_2 = i$ (cas 4.3.2) $i_2 = i+1$ (cas 4.3.3)
$i_1 = i+1$	$i_2 = i-1$ (cas 4.3.4) $i_2 = i$ (cas 4.3.5) $i_2 = i+1$ (cas 4.3.6)

4.3.1 — $i_1 = i-1$, $i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-3$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 ,$$

ce qui est absurde (voir (2.8)).

4.3.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = P_n^{(i-1,j)} + bP_{n-1}^{(i,j)} \tag{2.43}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i+n-1$:

$$L_{i+n-1}(x^j P_n^{(i-1,j)}) + bL_{i+n-1}(x^j P_{n-1}^{(i,j)}) = 0. \tag{2.44}$$

De (2.6) et (2.9), nous savons que

$$L_{i+n-1}(x^j P_n^{(i-1,j)}) = \frac{D_{n+1}^{(i-1,j)}}{D_n^{(i-1,j)}} \neq 0 \text{ et}$$

$$L_{i+n-1}(x^j P_{n-1}^{(i,j)}) = \frac{D_n^{(i,j)}}{D_{n-1}^{(i,j)}} \neq 0 .$$

Si nous prenons

$$b = -\frac{L_{i+n-1}(x^j P_n^{(i-1,j)})}{L_{i+n-1}(x^j P_{n-1}^{(i,j)})} = -\frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j)}}{D_n^{(i-1,j)} D_n^{(i,j)}},$$

la condition (2.44) est satisfaite, et la relation (2.43) vérifié.

Dénotons la relation trouvée par F_5.

$$F_5 \ : \ P_n^{(i,j)}(x) = P_n^{(i-1,j)}(x) - \frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j)}}{D_n^{(i-1,j)} D_n^{(i,j)}} P_{n-1}^{(i,j)}(x)$$

4.3.3 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = P_n^{(i-1,j)} + bP_{n-1}^{(i+1,j)} \tag{2.45}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} L_i(x^j P_n^{(i-1,j)}) + bL_i(x^j P_{n-1}^{(i+1,j)}) = 0 \\ L_{i+n-1}(x^j P_n^{(i-1,j)}) + bL_{i+n-1}(x^j P_{n-1}^{(i+1,j)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_n^{(i-1,j)}) = 0\,, \forall n \geq 2 \quad , \quad L_{i+n-1}(x^j P_{n-1}^{(i+1,j)}) = 0\,,$$

$$L_i(x^j P_{n-1}^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^j P_n^{(i-1,j)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} b = 0 \\ eq.imp \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.45) n'existe pas.

4.3.4 — $i_1 = i+1$, $i_2 = i-1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = P_n^{(i+1,j)} + bP_{n-1}^{(i-1,j)} \tag{2.46}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} L_i(x^j P_n^{(i+1,j)}) + bL_i(x^j P_{n-1}^{(i-1,j)}) = 0 \\ L_{i+n-2}(x^j P_n^{(i+1,j)}) + bL_{i+n-2}(x^j P_{n-1}^{(i-1,j)}) = 0 \\ L_{i+n-1}(x^j P_n^{(i+1,j)}) + bL_{i+n-1}(x^j P_{n-1}^{(i-1,j)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_{n-1}^{(i-1,j)}) = 0\,, \forall n \geq 3 \quad , \quad L_{i+n-2}(x^j P_n^{(i+1,j)}) = 0\,, \forall n \geq 3\,,$$

$$L_{i+n-1}(x^j P_n^{(i+1,j)}) = 0\,, \forall n \geq 2 \quad , \quad L_i(x^j P_n^{(i+1,j)}) \neq 0 \text{ et}$$

$$L_{i+n-2}(x^j P_{n-1}^{(i-1,j)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} eq.imp \\ b = 0 \\ - \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.46) n'existe pas.

4.3.5 — $i_1 = i + 1$, $i_2 = i$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = P_n^{(i+1,j)} + bP_{n-1}^{(i,j)} \tag{2.47}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i + n - 1$:

$$\begin{cases} L_i(x^j P_n^{(i+1,j)}) + bL_i(x^j P_{n-1}^{(i,j)}) = 0 \\ L_{i+n-1}(x^j P_n^{(i+1,j)}) + bL_{i+n-1}(x^j P_{n-1}^{(i,j)}) = 0 \end{cases}$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_{n-1}^{(i,j)}) = 0 \quad , \quad L_{i+n-1}(x^j P_n^{(i+1,j)}) = 0 \, , \forall n \geq 2 \, ,$$

$$L_i(x^j P_n^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^j P_{n-1}^{(i,j)}) \neq 0 \, ,$$

ce qui implique

$$\begin{cases} eq.imp \\ b = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.47) n'existe pas.

4.3.6 — $i_1 = i + 1$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = P_n^{(i+1,j)} + bP_{n-1}^{(i+1,j)} \tag{2.48}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$L_i(x^j P_n^{(i+1,j)}) + bL_i(x^j P_{n-1}^{(i+1,j)}) = 0. \tag{2.49}$$

De (2.5) et (2.8), nous savons que

$$L_i(x^j P_n^{(i+1,j)}) = (-1)^n \frac{D_{n+1}^{(i,j)}}{D_n^{(i+1,j)}} \neq 0 \quad \text{et}$$

$$L_i(x^j P_{n-1}^{(i+1,j)}) = (-1)^{n-1} \frac{D_n^{(i,j)}}{D_{n-1}^{(i+1,j)}} \neq 0 .$$

Si nous prenons

$$b = -\frac{L_i(x^j P_n^{(i+1,j)})}{L_i(x^j P_{n-1}^{(i+1,j)})} = \frac{D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j)}}{D_n^{(i+1,j)} D_n^{(i,j)}},$$

la condition (2.49) est satisfaite, et la relation (2.48) vérifiée.
Dénotons la relation trouvée par F_6.

$$F_6 \ : \ P_n^{(i,j)}(x) = P_n^{(i+1,j)}(x) + \frac{D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j)}}{D_n^{(i+1,j)} D_n^{(i,j)}} P_{n-1}^{(i+1,j)}(x)$$

4.4 — $m_1 = 0$, $m_2 = 1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = a P_n^{(i_1,j)} + bx P_{n-1}^{(i_2,j+1)}. \tag{2.50}$$

A partir de cette relation, nous voyons que le coefficient principal de $P_n^{(i,j)}$ est égal à $a + b$. Comme $P_n^{(i,j)}$ est unitaire $a + b = 1$, ou

$$b = 1 - a.$$

Considérons maintenant tous les valeurs possibles des indices i_1 et i_2. Remarquons que $i_1 \neq i$, sinon $P_n^{(i,j)}$ apparaîtrait dans le deuxième membre de (2.42).
Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i - 1$	$i_2 = i - 1$ (cas 4.4.1) $i_2 = i$ (cas 4.4.2) $i_2 = i + 1$ (cas 4.4.3)
$i_1 = i + 1$	$i_2 = i - 1$ (cas 4.4.4) $i_2 = i$ (cas 4.4.5) $i_2 = i + 1$ (cas 4.4.6)

4.4.1 — $i_1 = i - 1$, $i_2 = i - 1$.

Dans ce cas $p_1 = i - 1$ et $p_2 = i + n - 3$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 ,$$

ce qui est absurde (voir (2.8)).

4.4.2 — $i_1 = i - 1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = aP_n^{(i-1,j)} + (1-a)xP_{n-1}^{(i,j+1)} \tag{2.51}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i + n - 1$:

$$aL_{i+n-1}(x^j P_n^{(i-1,j)}) + (1-a)L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0. \tag{2.52}$$

De (2.6) et (2.9), nous savons que

$$L_{i+n-1}(x^j P_n^{(i-1,j)}) = \frac{D_{n+1}^{(i-1,j)}}{D_n^{(i-1,j)}} \neq 0 \tag{2.53}$$

et

$$L_{i+n-1}(x^{j+1} P_{n-1}^{(i,j+1)}) = \frac{D_n^{(i,j+1)}}{D_{n-1}^{(i,j+1)}} \neq 0\,. \tag{2.54}$$

Si nous prenons

$$a = \frac{L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)})}{L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) - L_{i+n-1}(x^j P_n^{(i-1,j)})},$$

la condition (2.52) est satisfaite, et la relation (2.51) vérifiée.

Ecrivons a et $1-a$ comme rapports de déterminants.

De (2.53) et (2.54), écrivons

$$a = \frac{D_n^{(i,j+1)} D_n^{(i-1,j)} D_{n-1}^{(i,j+1)}}{D_{n-1}^{(i,j+1)}(D_n^{(i,j+1)} D_n^{(i-1,j)} - D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)})}$$

et

$$\begin{aligned} 1-a &= \frac{L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)})}{L_{i+n-1}(x^j P_n^{(i-1,j)}) - L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)})} \\ &= \frac{D_{n+1}^{(i-1,j)} D_n^{(i-1,j)} D_{n-1}^{(i,j+1)}}{D_n^{(i-1,j)}(D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)} - D_n^{(i-1,j)} D_n^{(i,j+1)})} \end{aligned}$$

D'aprés l'identité de Sylvester (E.1)

$$D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)} = D_n^{(i-1,j)} D_n^{(i,j+1)} - D_n^{(i,j)} D_n^{(i-1,j+1)},$$

et alors

$$a = \frac{D_n^{(i,j+1)} D_n^{(i-1,j)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}}$$

et

$$1-a = -\frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}}.$$

Dénotons la relation trouvée par F_7.

$$F_7 \ : \ P_n^{(i,j)}(x) = \frac{D_n^{(i,j+1)} D_n^{(i-1,j)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}} P_n^{(i-1,j)}(x) - \frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}} x P_{n-1}^{(i,j+1)}(x)$$

4.4.3 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = a P_n^{(i-1,j)} + (1-a) x P_{n-1}^{(i+1,j+1)} \tag{2.55}$$

existe.
Ecrivons les conditions de biorthogonalité correspondante à $p = i$ et $p = i+n-1$:

$$\begin{cases} a L_i(x^j P_n^{(i-1,j)}) + (1-a) L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \\ a L_{i+n-1}(x^j P_n^{(i-1,j)}) + (1-a) L_{i+n-1}(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^j P_n^{(i-1,j)}) = 0 \ , \forall n \geq 2 \quad , \quad L_{i+n-1}(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0 \ ,$$

$$L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^j P_n^{(i-1,j)}) \neq 0 \ ,$$

ce qui implique

$$\begin{cases} a = 1 \\ a = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.55) n'existe pas.

4.4.4 — $i_1 = i+1$, $i_2 = i-1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = a P_n^{(i+1,j)} + (1-a) x P_{n-1}^{(i-1,j+1)} \tag{2.56}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} a L_i(x^j P_n^{(i+1,j)}) + (1-a) L_i(x^{j+1} P_{n-1}^{(i-1,j+1)}) = 0 \\ a L_{i+n-2}(x^j P_n^{(i+1,j)}) + (1-a) L_{i+n-2}(x^{j+1} P_{n-1}^{(i-1,j+1)}) = 0 \\ a L_{i+n-1}(x^j P_n^{(i+1,j)}) + (1-a) L_{i+n-1}(x^{j+1} P_{n-1}^{(i-1,j+1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i-1,j+1)}) = 0\,,\forall n \geq 3 \quad , \quad L_{i+n-2}(x^jP_n^{(i+1,j)}) = 0\,,\forall n \geq 3\,,$$

$$L_{i+n-1}(x^jP_n^{(i+1,j)}) = 0\,,\forall n \geq 2 \quad , \quad L_i(x^jP_n^{(i+1,j)}) \neq 0\,,\text{et}$$

$$L_{i+n-2}(x^{j+1}P_{n-1}^{(i-1,j+1)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} a = 0 \\ a = 1 \\ - \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.56) n'existe pas.

4.4.5 — $i_1 = i+1$, $i_2 = i$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = aP_n^{(i+1,j)} + (1-a)xP_{n-1}^{(i,j+1)} \tag{2.57}$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} aL_i(x^jP_n^{(i+1,j)}) + (1-a)L_i(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \\ aL_{i+n-1}(x^jP_n^{(i+1,j)}) + (1-a)L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_{n-1}^{(i,j+1)}) = 0 \quad , \quad L_{i+n-1}(x^jP_n^{(i+1,j)}) = 0\,,\forall n \geq 2\,,$$

$$L_i(x^jP_n^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} a = 0 \\ a = 1 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.57) n'existe pas.

4.4.6 — $i_1 = i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.
Voyons si la relation

$$P_n^{(i,j)} = aP_n^{(i+1,j)} + (1-a)xP_{n-1}^{(i+1,j+1)} \tag{2.58}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^j P_n^{(i+1,j)}) + (1-a)L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) = 0. \tag{2.59}$$

De (2.5) et (2.8), nous savons que

$$L_i(x^j P_n^{(i+1,j)}) = (-1)^n \frac{D_{n+1}^{(i,j)}}{D_n^{(i+1,j)}} \neq 0 \tag{2.60}$$

et

$$L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) = (-1)^{n-1} \frac{D_n^{(i,j+1)}}{D_{n-1}^{(i+1,j+1)}} \neq 0\,. \tag{2.61}$$

Si nous prenons

$$a = \frac{L_i(x^{j+1} P_{n-1}^{(i+1,j+1)})}{L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) - L_i(x^j P_n^{(i+1,j)})},$$

la condition (2.59) est satisfaite, et la relation (2.58) vérifiée.

Ecrivons a et $1-a$ comme rapports de déterminants.

De (2.60) et (2.61), écrivons

$$a = \frac{D_n^{(i,j+1)} D_{n-1}^{(i+1,j+1)} D_n^{(i+1,j)}}{D_{n-1}^{(i+1,j+1)} (D_n^{(i,j+1)} D_n^{(i+1,j)} + D_{n-1}^{(i+1,j+1)} D_{n+1}^{(i,j)})}$$

et

$$\begin{aligned} 1-a &= \frac{L_i(x^j P_n^{(i+1,j)})}{L_i(x^j P_n^{(i+1,j)}) - L_i(x^{j+1} P_{n-1}^{(i+1,j+1)})} \\ &= \frac{D_{n+1}^{(i,j)} D_n^{(i+1,j)} D_{n-1}^{(i+1,j+1)}}{D_{n+1}^{(i,j)} (D_{n-1}^{(i+1,j+1)} D_n^{(i+1,j)} + D_n^{(i+1,j)} D_n^{(i,j+1)})} \end{aligned}$$

D'aprés l'identité de Sylvester (E.1)

$$D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j+1)} = D_n^{(i,j)} D_n^{(i+1,j+1)} - D_n^{(i+1,j)} D_n^{(i,j+1)},$$

et alors

$$a = \frac{D_n^{(i,j+1)} D_n^{(i+1,j)}}{D_n^{(i,j)} D_n^{(i+1,j+1)}}$$

et

$$1-a = \frac{D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j+1)}}{D_n^{(i+1,j)} D_n^{(i,j+1)}}.$$

Dénotons la relation trouvée par F_8.

$$F_8 \;:\; P_n^{(i,j)}(x) = \frac{D_n^{(i,j+1)} D_n^{(i+1,j)}}{D_n^{(i,j)} D_n^{(i+1,j+1)}} P_n^{(i+1,j)}(x) + \frac{D_n^{(i,j+1)} D_n^{(i+1,j)}}{D_n^{(i,j)} D_n^{(i+1,j+1)}} x P_{n-1}^{(i+1,j+1)}(x)$$

5 — $n_1 = n_2 = n+1$.

<table>
<tr><td colspan="7">$n_1 = n$, $n_2 = n+1$</td></tr>
<tr><td>$m_1 = 0$
$m_2 = -1$</td><td colspan="3">$m_1 = -1$, $m_2 = -1$</td><td colspan="3">$m_1 = 1$, $m_2 = 0$</td></tr>
<tr><td>–</td><td colspan="3">$a = -\frac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_1,j-1,n+1)}}$
$b = 1$</td><td colspan="3">$a = \frac{1}{a_{n-1}^{(i_1,j+1,n)} - a_n^{(i_2,j,n+1)}}$
$b = -a$</td></tr>
<tr><td>–</td><td colspan="3">$i_1 = i-1$</td><td colspan="3">$i_1 = i-1$</td></tr>
<tr><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td colspan="3">$i_1 = i$</td><td colspan="3">$i_1 = i$</td></tr>
<tr><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>F_9</td><td>F_{10}</td><td>–</td><td>F_{11}</td><td>F_{12}</td><td>–</td></tr>
<tr><td>–</td><td colspan="3">$i_1 = i+1$</td><td colspan="3">$i_1 = i+1$</td></tr>
<tr><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
</table>

Expliquons en détail les différentes étapes représentées dans ce tableau.

Nous commençons par fixer les degrés $n_1 = n$ et $n_2 = n+1$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = ax^{m_1} P_n^{(i_1,j+m_1)} + bx^{m_2} P_{n+1}^{(i_2,j+m_2)}$$

A partir des degrés des trois polynômes, nous avons trois possibilités pour les valeurs de m_1 et m_2, à savoir: $m_1 = 0$ et $m_2 = -1$ (cas 5.1), $m_1 = m_2 = -1$ (cas 5.2), et $m_1 = 1$ et $m_2 = 0$ (cas 5.3).

5.1 — $m_1 = 0$, $m_2 = -1$.

Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

5.2 — $m_1 = m_2 = -1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i_1,j-1)} + \frac{b}{x} P_{n+1}^{(i_2,j-1)}. \tag{2.62}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = bx^n + \ldots + (aa_0^{(i_1,j-1,n)} + ba_0^{(i_2,j-1,n+1)})\frac{1}{x}$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} b = 1 \\ aa_0^{(i_1,j-1,n)} + ba_0^{(i_2,j-1,n+1)} = 0 \end{cases},$$

ce qui est équivalent à

$$\begin{cases} a = -\dfrac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_1,j-1,n)}} \\ b = 1 \end{cases}.$$

Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 5.2.1) $i_2 = i$ (cas 5.2.2) $i_2 = i+1$ (cas 5.2.3)
$i_1 = i$	$i_2 = i-1$ (cas 5.2.4) $i_2 = i$ (cas 5.2.5) $i_2 = i+1$ (cas 5.2.6)
$i_1 = i+1$	$i_2 = i-1$ (cas 5.2.7) $i_2 = i$ (cas 5.2.8) $i_2 = i+1$ (cas 5.2.9)

5.2.1 — $i_1 = i-1$, $i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-2$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 ,$$

ce qui est absurde (voir (2.8)).

5.2.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.
Voyons si la relation la relation :

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i-1,j-1)} + \frac{1}{x} P_{n+1}^{(i,j-1)} \tag{2.63}$$

où

$$a = -\frac{a_0^{(i,j-1,n+1)}}{a_0^{(i-1,j-1,n)}},$$

existe.
Ecrivons la condition de biorthogonalité correspondante à $p = i+n-1$:

$$a L_{i+n-1}(x^{j-1} P_n^{(i-1,j-1)}) + L_{i+n-1}(x^{j-1} P_{n+1}^{(i,j-1)}) = 0. \tag{2.64}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^{j-1} P_{n+1}^{(i,j-1)}) = 0 \quad \text{et} \quad L_{i+n-1}(x^{j-1} P_n^{(i-1,j-1)}) \neq 0,$$

donc la condition (2.64) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.63) n'existe pas.

5.2.3 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i-1,j-1)} + \frac{1}{x} P_{n+1}^{(i+1,j-1)} \tag{2.65}$$

où

$$a = -\frac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i-1,j-1,n)}},$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} aL_i(x^{j-1}P_n^{(i-1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \\ aL_{i+n-1}(x^{j-1}P_n^{(i-1,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \end{cases}$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j-1}P_n^{(i-1,j-1)}) = 0\,, \forall n \geq 2 \quad , \quad L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0\,, \forall n \geq 2\,,$$

$$L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j-1}P_n^{(i-1,j-1)}) \neq 0,$$

ce qui implique

$$\begin{cases} eq.imp. \\ a = 0 \end{cases},$$

qui est un système impossible. Conclusion : la relation (2.65) n'existe pas.

5.2.4 — $i_1 = i$, $i_2 = i-1$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i,j-1)} + \frac{1}{x} P_{n+1}^{(i-1,j-1)},$$

où

$$a = -\frac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i,j-1,n)}}.$$

En utilisant (2.15), écrivons le coefficient a comme rapport de déterminants.

$$a = \frac{D_{n+1}^{(i-1,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i-1,j-1)} D_n^{(i,j)}}.$$

Dénotons la relation trouvée par F_9.

$$F_9 \ : \ P_n^{(i,j)}(x) = \tfrac{1}{x} P_{n+1}^{(i-1,j-1)}(x) + \frac{D_{n+1}^{(i-1,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i-1,j-1)} D_n^{(i,j)}} \tfrac{1}{x} P_n^{(i,j-1)}(x)$$

5.2.5 — $i_1 = i$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i,j-1)} + \frac{1}{x} P_{n+1}^{(i,j-1)},$$

où

$$a = -\frac{a_0^{(i,j-1,n+1)}}{a_0^{(i,j-1,n)}}.$$

En utilisant (2.15), écrivons le coefficient a comme rapport de déterminants.

$$a = \frac{D_{n+1}^{(i,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i,j-1)} D_n^{(i,j)}}.$$

Dénotons la relation trouvée par F_{10}.

$$F_{10} \ : \ P_n^{(i,j)}(x) = \tfrac{1}{x} P_{n+1}^{(i,j-1)}(x) + \frac{D_{n+1}^{(i,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i,j-1)} D_n^{(i,j)}} \tfrac{1}{x} P_n^{(i,j-1)}(x)$$

5.2.6 — $i_1 = i$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i,j-1)} + \frac{1}{x} P_{n+1}^{(i+1,j-1)} \tag{2.66}$$

où

$$a = -\frac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i,j-1,n)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a L_i(x^{j-1} P_n^{(i,j-1)}) + L_i(x^{j-1} P_{n+1}^{(i+1,j-1)}) = 0. \tag{2.67}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j-1} P_n^{(i,j-1)}) = 0 \quad \text{et} \quad L_i(x^{j-1} P_{n+1}^{(i+1,j-1)}) \neq 0 \,,$$

donc la condition (2.67) est impossible. Conclusion : la relation (2.66) n'existe pas.

5.2.7 — $i_1 = i+1$, $i_2 = i-1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_n^{(i+1,j-1)} + \frac{1}{x} P_{n+1}^{(i-1,j-1)} \tag{2.68}$$

où

$$a = -\frac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i+1,j-1,n)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^{j-1}P_n^{(i+1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0. \tag{2.69}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0 \ , \forall n \geq 1 \quad \text{et} \quad L_i(x^{j-1}P_n^{(i+1,j-1)}) \neq 0 \ ,$$

donc la condition (2.69) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.68) n'existe pas.

5.2.8 — $i_1 = i+1$, $i_2 = i$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0 \ ,$$

ce qui est absurde (voir (2.9)).

5.2.9 — $i_1 = i+1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0 \ ,$$

ce qui est absurde (voir (2.9)).

5.3 — $m_1 = 1$, $m_2 = 0$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_n^{(i_1,j+1)} + bP_{n+1}^{(i_2,j)}. \tag{2.70}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = (a+b)x^{n+1} + (aa_{n-1}^{(i_1,j+1,n)} + ba_n^{(i_2,j,n+1)})x^n + \dots$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} a+b=0 \\ aa_{n-1}^{(i_1,j+1,n)} + ba_n^{(i_2,j,n+1)} = 1 \end{cases},$$

ce qui est équivalent à

$$\begin{cases} a = \frac{1}{a_{n-1}^{(i_1,j+1,n)} - a_n^{(i_2,j,n+1)}} \\ b = -a \end{cases}.$$

Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 5.3.1) $i_2 = i$ (cas 5.3.2) $i_2 = i+1$ (cas 5.3.3)
$i_1 = i$	$i_2 = i-1$ (cas 5.3.4) $i_2 = i$ (cas 5.3.5) $i_2 = i+1$ (cas 5.3.6)
$i_1 = i+1$	$i_2 = i-1$ (cas 5.3.7) $i_2 = i$ (cas 5.3.8) $i_2 = i+1$ (cas 5.3.9)

5.3.1 — $i_1 = i-1$, $i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-2$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0 ,$$

ce qui est absurde (voir (2.8)).

5.3.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = a(xP_n^{(i-1,j+1)} - P_{n+1}^{(i,j)}) \tag{2.71}$$

où

$$a = \frac{1}{a_{n-1}^{(i-1,j+1,n)} - a_n^{(i,j,n+1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i+n-1$:

$$a(L_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) - L_{i+n-1}(x^j P_{n+1}^{(i,j)})) = 0. \tag{2.72}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^j P_{n+1}^{(i,j)}) = 0 , \forall n \geq 1 \quad \text{et} \quad L_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) \neq 0 ,$$

donc la condition (2.72) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.71) n'existe pas.

5.3.3 — $i_1 = i - 1$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i + n - 1$.
Voyons si la relation

$$P_n^{(i,j)} = a(xP_n^{(i-1,j+1)} - P_{n+1}^{(i+1,j)}) \tag{2.73}$$

où

$$a = \frac{1}{a_{n-1}^{(i-1,j+1,n)} - a_n^{(i+1,j,n+1)}},$$

existe.
Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i + n - 1$:

$$\begin{cases} a(L_i(x^{j+1}P_n^{(i-1,j+1)}) - L_i(x^jP_{n+1}^{(i+1,j)})) = 0 \\ a(L_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) - L_{i+n-1}(x^jP_{n+1}^{(i+1,j)})) = 0 \end{cases}$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+1}P_n^{(i-1,j+1)}) = 0\ ,\forall n \geq 2 \quad , \quad L_{i+n-1}(x^jP_{n+1}^{(i+1,j)}) = 0\ ,\forall n \geq 2\ ,$$

$$L_i(x^jP_{n+1}^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j+1}P_n^{(i-1,j+1)}) \neq 0,$$

ce qui implique que le système précédent est équivalent à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.73) n'existe pas.

5.3.4 — $i_1 = i$, $i_2 = i - 1$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = a(xP_n^{(i,j+1)} - P_{n+1}^{(i-1,j)}),$$

où

$$a = \frac{1}{a_{n-1}^{(i,j+1,n)} - a_n^{(i-1,j,n+1)}}.$$

Ecrivons le coefficient a comme rapport de déterminants.
De (2.16), nous pouvons écrire :

$$a = (D_n^{(i,j+1)}D_{n+1}^{(i-1,j)})/$$

$$/(\begin{pmatrix} i-1 & \dots & i+n-2 & i+n-1 \\ j & \dots & j+n-1 & j+n+1 \end{pmatrix} D_n^{(i,j+1)}-$$

$$-\begin{pmatrix} i & \dots & i+n-2 & i+n-1 \\ j & \dots & j+n-1 & j+n+1 \end{pmatrix} D_{n+1}^{(i-1,j)}).$$

D'après l'identité de Schweins (E.17) le dénominateur du membre droit de l'égalité précédente est égal à

$$D_{n+1}^{(i-1,j+1)} D_n^{(i,j)},$$

et alors

$$a = \frac{D_n^{(i,j+1)} D_{n+1}^{(i-1,j)}}{D_{n+1}^{(i-1,j+1)} D_n^{(i,j)}}$$

Dénotons la relation trouvée par F_{11}.

$$F_{11} \ : \ P_n^{(i,j)}(x) = \frac{D_n^{(i,j+1)} D_{n+1}^{(i-1,j)}}{D_{n+1}^{(i-1,j+1)} D_n^{(i,j)}} (x P_n^{(i,j+1)}(x) - P_{n+1}^{(i-1,j)}(x))$$

5.3.5 — $i_1 = i$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées. Donc, nous avons trouvé la relation suivante :

$$P_n^{(i,j)} = a(x P_n^{(i,j+1)} - P_{n+1}^{(i,j)}),$$

où

$$a = \frac{1}{a_{n-1}^{(i,j+1,n)} - a_n^{(i,j,n+1)}}.$$

Ecrivons le coefficient a comme rapport de déterminants.
De (2.16), nous pouvons écrire :

$$a = (D_n^{(i,j+1)} D_{n+1}^{(i,j)})/$$

$$/(\begin{pmatrix} i & \dots & i+n-1 & i+n \\ j & \dots & j+n-1 & j+n+1 \end{pmatrix} D_n^{(i,j+1)} -$$

$$-\begin{pmatrix} i & \dots & i+n-2 & i+n-1 \\ j+1 & \dots & j+n-1 & j+n+1 \end{pmatrix} D_{n+1}^{(i,j)}).$$

D'après l'identité de Schweins (E.15) le dénominateur du membre droit de l'égalité précédente est égal à

$$D_{n+1}^{(i,j+1)} D_n^{(i,j)},$$

et alors

$$a = \frac{D_n^{(i,j+1)} D_{n+1}^{(i,j)}}{D_{n+1}^{(i,j+1)} D_n^{(i,j)}}.$$

Dénotons la relation trouvée par F_{12}.

$$F_{12} : \quad P_n^{(i,j)}(x) = \frac{D_n^{(i,j+1)} D_{n+1}^{(i,j)}}{D_{n+1}^{(i,j+1)} D_n^{(i,j)}} (x P_n^{(i,j+1)}(x) - P_{n+1}^{(i,j)}(x))$$

5.3.6 — $i_1 = i$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(x P_n^{(i,j+1)} - P_{n+1}^{(i+1,j)}) \tag{2.74}$$

où

$$a = \frac{1}{a_{n-1}^{(i,j+1,n)} - a_n^{(i+1,j,n+1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^{j+1} P_n^{(i,j+1)}) - L_i(x^j P_{n+1}^{(i+1,j)})) = 0. \tag{2.75}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j+1} P_n^{(i,j+1)}) = 0 \quad \text{et} \quad L_i(x^j P_{n+1}^{(i+1,j)}) \neq 0 ,$$

donc la condition (2.75) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.74) n'existe pas.

5.3.7 — $i_1 = i + 1$, $i_2 = i - 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(x P_n^{(i+1,j+1)} - P_{n+1}^{(i-1,j)}), \tag{2.76}$$

où

$$a = \frac{1}{a_{n-1}^{(i+1,j+1,n)} - a_n^{(i-1,j,n+1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^{j+1} P_n^{(i+1,j+1)}) - L_i(x^j P_{n+1}^{(i-1,j)})) = 0. \tag{2.77}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^j P_{n+1}^{(i-1,j)}) = 0 \ , \forall n \geq 1 \quad \text{et} \quad L_i(x^{j+1} P_n^{(i+1,j+1)}) \neq 0 ,$$

donc la condition (2.77) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.76) n'existe pas.

5.3.8 — $i_1 = i+1$, $i_2 = i$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0\,,$$

ce qui est absurde (voir (2.9)).

5.3.9 — $i_1 = i+1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n$, ce qui implique

$$L_{i+n}(x^j P_n^{(i,j)}) = 0\,,$$

ce qui est absurde (voir (2.9)).

6 — $n_1 = n-1$, $n_2 = n+1$.

<table>
<tr><td colspan="5">$n_1 = n-1$, $n_2 = n+1$</td></tr>
<tr><td>$m_1 = 0$
$m_2 = -1$</td><td>$m_1 = 1$
$m_2 = -1$</td><td colspan="3">$m_1 = -1$, $m_2 = -1$</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$a = -\frac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_1,j-1,n-1)}}$
$b = 1$</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$i_1 = i-1$</td></tr>
<tr><td>–</td><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$i_1 = i$</td></tr>
<tr><td>–</td><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td colspan="3">$i_1 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>$i_2 = i-1$</td><td>$i_2 = i$</td><td>$i_2 = i+1$</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
</table>

Expliquons en détail les différentes étapes représentées dans ce tableau.

Nous commençons par fixer les degrés $n_1 = n-1$ et $n_2 = n+1$, c'est-à-dire nous cherchons toutes les relations du type :

$$P_n^{(i,j)} = a x^{m_1} P_{n-1}^{(i_1,j+m_1)} + b x^{m_2} P_{n+1}^{(i_2,j+m_2)}.$$

A partir des degrés des trois polynômes, nous avons trois possibilités pour les valeurs de m_1 et m_2, à savoir: $m_1 = 0$ et $m_2 = -1$ (cas 6.1), $m_1 = 1$ et $m_2 = -1$ (cas 6.2), et $m_1 = m_2 = -1$ (cas 6.3).

6.1 — $m_1 = 0$, $m_2 = -1$.

Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

6.2 — $m_1 = 1$, $m_2 = -1$.
Comme $m_2 = -1$ et $m_1 \neq -1$, ce cas est impossible.

6.3 — $m_1 = -1$, $m_2 = -1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = \frac{a}{x} P_{n-1}^{(i_1,j-1)} + \frac{b}{x} P_{n+1}^{(i_2,j-1)}. \tag{2.78}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = bx^n + \ldots + (aa_0^{(i_1,j-1,n-1)} + ba_0^{(i_2,j-1,n+1)}) \frac{1}{x}.$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} b = 1 \\ aa_0^{(i_1,j-1,n-1)} + ba_0^{(i_2,j-1,n+1)} = 0 \end{cases},$$

qui est équivalent au système

$$\begin{cases} a = -\dfrac{a_0^{(i_2,j-1,n+1)}}{a_0^{(i_1,j-1,n-1)}} \\ b = 1 \end{cases}.$$

Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 6.3.1) $i_2 = i$ (cas 6.3.2) $i_2 = i+1$ (cas 6.3.3)
$i_1 = i$	$i_2 = i-1$ (cas 6.3.4) $i_2 = i$ (cas 6.3.5) $i_2 = i+1$ (cas 6.3.6)
$i_1 = i+1$	$i_2 = i-1$ (cas 6.3.7) $i_2 = i$ (cas 6.3.8) $i_2 = i+1$ (cas 6.3.9)

6.3.1 — $i_1 = i-1$, $i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-3$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0,$$

ce qui est absurde (voir (2.8)).

6.3.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-3$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i+n-2$ et $p = i+n-1$. Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_{n-1}^{(i-1,j-1)} + \frac{1}{x} P_{n+1}^{(i,j-1)}, \tag{2.79}$$

où

$$a = -\frac{a_0^{(i,j-1,n+1)}}{a_0^{(i-1,j-1,n-1)}},$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} aL_{i+n-2}(x^{j-1}P_{n-1}^{(i-1,j-1)}) + L_{i+n-2}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0 \\ aL_{i+n-1}(x^{j-1}P_{n-1}^{(i-1,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0 \end{cases}.$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-2}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0\,, \forall n \geq 2 \quad , \quad L_{i+n-1}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0\,, \forall n \geq 1 \text{ et}$$

$$L_{i+n-2}(x^{j-1}P_{n-1}^{(i-1,j-1)}) \neq 0,$$

donc le système précédent se réduit à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.79) n'existe pas.

6.3.3 — $i_1 = i-1$, $i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-3$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x} P_{n-1}^{(i-1,j-1)} + \frac{1}{x} P_{n+1}^{(i+1,j-1)}, \tag{2.80}$$

où

$$a = -\frac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i-1,j-1,n-1)}},$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} aL_i(x^{j-1}P_{n-1}^{(i-1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \\ aL_{i+n-2}(x^{j-1}P_{n-1}^{(i-1,j-1)}) + L_{i+n-2}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \\ aL_{i+n-1}(x^{j-1}P_{n-1}^{(i-1,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j-1}P_{n-1}^{(i-1,j-1)}) = 0\ ,\forall n \geq 3\quad ,\quad L_{i+n-2}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0\ ,\forall n \geq 3,$$

$$L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0\ ,\forall n \geq 2\quad ,\quad L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) \neq 0 \quad \text{et}$$

$$L_{i+n-2}(x^{j-1}P_{n-1}^{(i-1,j-1)}) \neq 0,$$

donc le système précédent est équivalent au système

$$\begin{cases} eq.imp. \\ a = 0 \\ - \end{cases},$$

qui est impossible. Conclusion : la relation (2.80) n'existe pas.

6.3.4 — $i_1 = i\ ,\ i_2 = i-1$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i,j-1)} + \frac{1}{x}P_{n+1}^{(i-1,j-1)}, \tag{2.81}$$

où

$$a = -\frac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i,j-1,n-1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i+n-1$:

$$aL_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0. \tag{2.82}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0 \quad \text{et} \quad L_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) \neq 0,$$

donc la condition (2.82) est équivalente à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.81) n'existe pas.

6.3.5 — $i_1 = i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i,j-1)} + \frac{1}{x}P_{n+1}^{(i,j-1)}, \tag{2.83}$$

où

$$a = -\frac{a_0^{(i,j-1,n+1)}}{a_0^{(i,j-1,n-1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i + n - 1$:

$$aL_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0. \tag{2.84}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^{j-1}P_{n+1}^{(i,j-1)}) = 0\,, \forall n \geq 1 \quad \text{et} \quad L_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) \neq 0,$$

donc la condition (2.84) est équivalente à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.83) n'existe pas.

6.3.6 — $i_1 = i\ ,\ i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 2$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i,j-1)} + \frac{1}{x}P_{n+1}^{(i+1,j-1)}, \tag{2.85}$$

où

$$a = -\frac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i,j-1,n-1)}},$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, et $p = i + n - 1$:

$$\begin{cases} aL_i(x^{j-1}P_{n-1}^{(i,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \\ aL_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) + L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0 \end{cases}$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j-1}P_{n-1}^{(i,j-1)}) = 0 \quad , \quad L_{i+n-1}(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0\,, \forall n \geq 2,$$

$$L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j-1}P_{n-1}^{(i,j-1)}) \neq 0\,,$$

donc le système précédent est équivalent au système

$$\begin{cases} eq.imp. \\ a = 0 \end{cases},$$

qui est impossible. Conclusion : la relation (2.85) n'existe pas.

6.3.7 — $i_1 = i + 1\ ,\ i_2 = i - 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i+1,j-1)} + \frac{1}{x}P_{n+1}^{(i-1,j-1)}, \tag{2.86}$$

où

$$a = -\frac{a_0^{(i-1,j-1,n+1)}}{a_0^{(i+1,j-1,n-1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0. \tag{2.87}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j-1}P_{n+1}^{(i-1,j-1)}) = 0\ , \forall n \geq 1 \quad \text{et} \quad L_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) \neq 0,$$

donc la condition (2.87) est équivalente à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.86) n'existe pas.

6.3.8 — $i_1 = i + 1$, $i_2 = i$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce que veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i+1,j-1)} + \frac{1}{x}P_{n+1}^{(i,j-1)}, \tag{2.88}$$

où

$$a = -\frac{a_0^{(i,j-1,n+1)}}{a_0^{(i+1,j-1,n-1)}},$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i,j-1)}) = 0. \tag{2.89}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^{j-1}P_{n+1}^{(i,j-1)}) = 0 \quad \text{et} \quad L_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) \neq 0,$$

donc la condition (2.89) est équivalente à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.88) n'existe pas.

6.3.9 — $i_1 = i + 1$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = \frac{a}{x}P_{n-1}^{(i+1,j-1)} + \frac{1}{x}P_{n+1}^{(i+1,j-1)}, \tag{2.90}$$

où

$$a = -\frac{a_0^{(i+1,j-1,n+1)}}{a_0^{(i+1,j-1,n-1)}}, \tag{2.91}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$aL_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) + L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = 0. \tag{2.92}$$

De (2.5) et (2.8), nous savons que

$$L_i(x^{j-1}P_{n-1}^{(i+1,j-1)}) = (-1)^{n-1}\frac{D_n^{(i,j-1)}}{D_{n-1}^{(i+1,j-1)}} \neq 0$$

et

$$L_i(x^{j-1}P_{n+1}^{(i+1,j-1)}) = (-1)^{n+1}\frac{D_{n+2}^{(i,j-1)}}{D_{n+1}^{(i+1,j-1)}} \neq 0.$$

De (2.91) et (2.15), nous écrivons

$$a = -\frac{D_{n+1}^{(i+1,j)}D_{n-1}^{(i+1,j-1)}}{D_{n+1}^{(i+1,j-1)}D_{n-1}^{(i+1,j)}}.$$

Alors, la condition de biorthogonalité (2.92) s'écrit de la façon suivante

$$\frac{D_{n+2}^{(i,j-1)}D_{n-1}^{(i+1,j)} - D_{n+1}^{(i+1,j)}D_n^{(i,j-1)}}{D_{n+1}^{(i+1,j-1)}D_{n-1}^{(i+1,j)}} = 0,$$

qui est équivalent à

$$D_{n+2}^{(i,j-1)}D_{n-1}^{(i+1,j)} - D_{n+1}^{(i+1,j)}D_n^{(i,j-1)} = 0, \tag{2.93}$$

puisque

$$D_{n+1}^{(i+1,j-1)}D_{n-1}^{(i+1,j)} \neq 0.$$

Voyons que $\exists\{L_i\}_{i=0,1,\ldots}$, et $\exists i_0, j_0, n_0 \geq 0$, tels que (2.93) ne se vérifie pas, et par conséquent la condition de biorthogonalité (2.92) non plus.

Définissons $L_i(x^j) = 1/(i+j+1)$, $i,j = 0,1,\ldots$ et choisissons $i_0 = 0$, $j_0 = 1$ et $n_0 = 1$. Nous pouvons vérifier en utilisant par exemple *Mathematica*[13], que

$$D_3^{(0,0)}D_0^{(1,1)} - D_2^{(1,1)}D_1^{(0,0)} = -1/270 \neq 0.$$

Pour que ce contre exemple soit valable, nous vérifions aussi l'hypothèse (2.4) pour les indices i, j et n correspondants, c'est-à-dire nous vérifions que

$$D_n^{(i,j)} \neq 0 \ , n = 1,2,3 \ , i,j = 0,\ldots,3-n.$$

Conclusion: la condition de biorthogonalité (2.92) peut ne pas être vérifiée, donc dans le cas général nous ne pouvons pas affirmer que la relation (2.90) existe.

Nous finissons ici cette longue étude.

Remarquons que les polynômes qui apparaîssent dans les deuxièmes membres des 12 relations que nous avons trouvées, sont adjacents consécutifs à $P_n^{(i,j)}$ selon les première et deuxième dispositions des polynômes biorthogonaux dans l'espace. Donc, si au début de cette étude nous avions opté pour la première disposition, nous aurions obtenu toutes ces relations. La question que nous nous posons maintenant est de savoir si nous n'aurions pas trouvé d'autres relations. Pour répondre à cette question rappelons que $P_{n-1}^{(l,j+2)}$ et $P_{n+1}^{(l,j-2)}$, $l = i-1, i, i+1$ sont les seuls polynômes adjacents consécutifs à $P_n^{(i,j)}$ selon la première disposition, qui ne sont pas adjacents consécutifs à $P_n^{(i,j)}$ par rapport à la deuxième disposition. Voyons s'il existe des relations de récurrence où $P_n^{(i,j)}$ est calculé à partir d'un ou deux de ces polynômes.

Plus précisement, voyons si les relations suivantes existent

$$1 - P_n^{(i,j)}(x) = ax^{m_1} P_{n_1}^{(i_1,j+m_1)}(x) + bx^2 P_{n-1}^{(i_2,j+2)}(x)$$

$$2 - P_n^{(i,j)}(x) = ax^2 P_{n-1}^{(i_1,j+2)}(x) + bx^2 P_{n-1}^{(i_2,j+2)}(x)$$

$$3 - P_n^{(i,j)}(x) = ax^{m_1} P_{n_1}^{(i_1,j+m_1)}(x) + \tfrac{b}{x^2} P_{n+1}^{(i_2,j-2)}(x)$$

$$4 - P_n^{(i,j)}(x) = \tfrac{a}{x^2} P_{n+1}^{(i_1,j-2)}(x) + \tfrac{b}{x^2} P_{n+1}^{(i_2,j-2)}(x)$$

$$5 - P_n^{(i,j)}(x) = ax^2 P_{n-1}^{(i_1,j+2)}(x) + \tfrac{b}{x^2} P_{n+1}^{(i_2,j-2)}(x),$$

où m_1 ne peut prendre que les valeurs -1, 0 ou 1; n_1 ne peut prendre que les valeurs $n-1$, n ou $n+1$; et enfin i_1 et i_2 ne peuvent prendre que les valeurs $i-1$, i ou $i+1$.

Les facteurs des polynômes $P_{n-1}^{(l,j+2)}$ et $P_{n+1}^{(l,j-2)}$, $l = i_1, i_2$ ont été choisis de telle sorte que leurs conditions de biorthogonalité puissent être appliquées.

Commençons à faire cette étude.

1 — $P_n^{(i,j)}(x) = ax^{m_1} P_{n_1}^{(i_1,j+m_1)}(x) + bx^2 P_{n-1}^{(i_2,j+2)}(x)$

Comme la deuxième partie du second membre de cette relation est un polynôme de degré $n+1$, alors $m_1 + n_1 = n+1$. Il y a deux possiblités pour les valeurs de m_1 et n_1, à savoir: $m_1 = 1$ et $n_1 = n$ (cas 1.1); et $m_1 = 0$ et $n_1 = n+1$ (cas 1.2).

1.1 — $m_1 = 1$, $n_1 = n$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = axP_n^{(i_1,j+1)} + bx^2 P_{n-1}^{(i_2,j+2)}. \tag{2.94}$$

A partir de cette relation, nous obtenons

$$P_n^{(i,j)}(0) = 0,$$

ce qui est absurde (voir 2.10). Conclusion : les relations du type (2.94) n'existent pas.

1.2 — $m_1 = 0$, $n_1 = n+1$.

Dans ce groupe, nous cherchons toutes les relations du type

$$P_n^{(i,j)} = aP_{n+1}^{(i_1,j)} + bx^2 P_{n-1}^{(i_2,j+2)}. \tag{2.95}$$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = (a+b)x^{n+1} + (aa_n^{(i_1,j,n+1)} + ba_{n-2}^{(i_2,j+2,n-1)})x^n + \ldots.$$

Comme $P_n^{(i,j)}$ est un polynôme unitaire de degré n

$$\begin{cases} a+b=0 \\ aa_n^{(i_1,j,n+1)} + ba_{n-2}^{(i_2,j+2,n-1)} = 0 \end{cases},$$

ce qui est équivalent à

$$\begin{cases} a = \dfrac{1}{a_n^{(i_1,j,n+1)} - a_{n-2}^{(i_2,j+2,n-1)}} \\ b = -a \end{cases}.$$

Représentons dans le tableau suivant toutes les valeurs possibles des indices i_1 et i_2.

$i_1 = i-1$	$i_2 = i-1$ (cas 1.2.1) $i_2 = i$ (cas 1.2.2) $i_2 = i+1$ (cas 1.2.3)
$i_1 = i$	$i_2 = i-1$ (cas 1.2.4) $i_2 = i$ (cas 1.2.5) $i_2 = i+1$ (cas 1.2.6)
$i_1 = i+1$	$i_2 = i-1$ (cas 1.2.7) $i_2 = i$ (cas 1.2.8) $i_2 = i+1$ (cas 1.2.9)

1.2.1 — $i_1 = i_2 = i-1$.

Dans ce cas $p_1 = i-1$ et $p_2 = i+n-3$, ce qui implique

$$L_{i-1}(x^j P_n^{(i,j)}) = 0,$$

ce qui est absurde (voir (2.8)).

1.2.2 — $i_1 = i-1$, $i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i+n-1$.
Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i-1,j)} - x^2 P_{n-1}^{(i,j+2)}), \tag{2.96}$$

où

$$a = \frac{1}{a_n^{(i-1,j,n+1)} - a_{n-2}^{(i,j+2,n-1)}}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i + n - 1$:

$$a(L_{i+n-1}(x^j P_{n+1}^{(i-1,j)}) - L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)})) = 0. \tag{2.97}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^j P_{n+1}^{(i-1,j)}) = 0 \quad \text{et} \quad L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)}) \neq 0\,,$$

donc la condition (2.97) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.96) n'existe pas.

1.2.3 — $i_1 = i - 1$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i-1,j)} - x^2 P_{n-1}^{(i+1,j+2)}), \tag{2.98}$$

où

$$a = \frac{1}{a_n^{(i-1,j,n+1)} - a_{n-2}^{(i+1,j+2,n-1)}}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^j P_{n+1}^{(i-1,j)}) - L_i(x^{j+2} P_{n-1}^{(i+1,j+2)})) = 0. \tag{2.99}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^j P_{n+1}^{(i-1,j)}) = 0\,, \forall n \geq 1 \quad \text{et} \quad L_i(x^{j+2} P_{n-1}^{(i+1,j+2)}) \neq 0\,,$$

donc la condition (2.99) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.98) n'existe pas.

1.2.4 — $i_1 = i$, $i_2 = i - 1$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i + n - 2$ et $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i,j)} - x^2 P_{n-1}^{(i-1,j+2)}), \tag{2.100}$$

où

$$a = \frac{1}{a_n^{(i,j,n+1)} - a_{n-2}^{(i-1,j+2,n-1)}}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i+n-2$ et $p = i+n-1$:

$$\begin{cases} a(L_{i+n-2}(x^j P_{n+1}^{(i,j)}) - L_{i+n-2}(x^{j+2} P_{n-1}^{(i-1,j+2)})) = 0 \\ a(L_{i+n-1}(x^j P_{n+1}^{(i,j)}) - L_{i+n-1}(x^{j+2} P_{n-1}^{(i-1,j+2)})) = 0 \end{cases}.$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-2}(x^j P_{n+1}^{(i,j)}) = 0\,, \forall n \geq 2 \quad , \quad L_{i+n-1}(x^j P_{n+1}^{(i,j)}) = 0\,, \forall n \geq 1 \text{ et}$$

$$L_{i+n-2}(x^{j+2} P_{n-1}^{(i-1,j+2)}) \neq 0\,,$$

ce qui implique

$$\begin{cases} a = 0 \\ - \end{cases},$$

qui est un système impossible, parce que a doit être non nul. Conclusion : la relation (2.100) n'existe pas.

1.2.5 — $i_1 = i_2 = i$.

Dans ce cas $p_1 = i$ et $p_2 = i + n - 2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i,j)} - x^2 P_{n-1}^{(i,j+2)}), \tag{2.101}$$

où

$$a = \frac{1}{a_n^{(i,j,n+1)} - a_{n-2}^{(i,j+2,n-1)}}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i + n - 1$:

$$a(L_{i+n-1}(x^j P_{n+1}^{(i,j)}) - L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)})) = 0. \tag{2.102}$$

De (2.1) et (2.9), nous savons que

$$L_{i+n-1}(x^j P_{n+1}^{(i,j)}) = 0\,, \forall n \geq 1 \text{ et } L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)}) \neq 0\,,$$

donc la condition (2.102) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.101) n'existe pas.

1.2.6 — $i_1 = i$, $i_2 = i + 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i,j)} - x^2 P_{n-1}^{(i+1,j+2)}), \tag{2.103}$$

où

$$a = \frac{1}{a_n^{(i,j,n+1)} - a_{n-2}^{(i+1,j+2,n-1)}}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^j P_{n+1}^{(i,j)}) - L_i(x^{j+2} P_{n-1}^{(i+1,j+2)})) = 0. \tag{2.104}$$

De (2.1) et (2.8), nous savons que

$$L_i(x^j P_{n+1}^{(i,j)}) = 0 \quad \text{et} \quad L_i(x^{j+2} P_{n-1}^{(i+1,j+2)}) \neq 0\,,$$

donc la condition (2.104) est équivalente à $a = 0$, qui est impossible. Conclusion : la relation (2.103) n'existe pas.

1.2.7 — $i_1 = i + 1$, $i_2 = i - 1$.

Dans ce cas $p_1 = i + 1$ et $p_2 = i + n - 3$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$, $p = i + n - 2$ et $p = i + n - 1$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i+1,j)} - x^2 P_{n-1}^{(i-1,j+2)}), \tag{2.105}$$

où

$$a = \frac{1}{a_n^{(i+1,j,n+1)} - a_{n-2}^{(i-1,j+2,n-1)}}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$, $p = i + n - 2$ et $p = i + n - 1$:

$$\begin{cases} a(L_i(x^j P_{n+1}^{(i+1,j)}) - L_i(x^{j+2} P_{n-1}^{(i-1,j+2)})) = 0 \\ a(L_{i+n-2}(x^j P_{n+1}^{(i+1,j)}) - L_{i+n-2}(x^{j+2} P_{n-1}^{(i-1,j+2)})) = 0 \\ a(L_{i+n-1}(x^j P_{n+1}^{(i+1,j)}) - L_{i+n-1}(x^{j+2} P_{n-1}^{(i-1,j+2)})) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+2} P_{n-1}^{(i-1,j+2)}) = 0\,, \forall n \geq 3 \quad , L_{i+n-2}(x^j P_{n+1}^{(i+1,j)}) = 0\,, \forall n \geq 3\,,$$

$$L_{i+n-1}(x^j P_{n+1}^{(i+1,j)}) = 0\,, \forall n \geq 2 \quad , \quad L_i(x^j P_{n+1}^{(i+1,j)}) \neq 0 \text{ et}$$

$$L_{i+n-2}(x^{j+2} P_{n-1}^{(i-1,j+2)}) \neq 0\,,$$

ce qui implique que le système précédent est équivalent à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.105) n'existe pas.

1.2.8 — $i_1 = i + 1$, $i_2 = i$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-2$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celles correspondantes à $p = i$ et $p = i+n-1$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i+1,j)} - x^2 P_{n-1}^{(i,j+2)}), \tag{2.106}$$

où

$$a = \frac{1}{a_n^{(i+1,j,n+1)} - a_{n-2}^{(i,j+2,n-1)}}$$

existe.

Ecrivons les conditions de biorthogonalité correspondantes à $p = i$ et $p = i+n-1$:

$$\begin{cases} a(L_i(x^j P_{n+1}^{(i+1,j)}) - L_i(x^{j+2} P_{n-1}^{(i,j+2)})) = 0 \\ a(L_{i+n-1}(x^j P_{n+1}^{(i+1,j)}) - L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)})) = 0 \end{cases}.$$

De (2.1), (2.8) et (2.9), nous savons que

$$L_i(x^{j+2} P_{n-1}^{(i,j+2)}) = 0 \quad , L_{i+n-1}(x^j P_{n+1}^{(i+1,j)}) = 0 \, , \forall n \geq 2 \, ,$$

$$L_i(x^j P_{n+1}^{(i+1,j)}) \neq 0 \quad \text{et} \quad L_{i+n-1}(x^{j+2} P_{n-1}^{(i,j+2)}) \neq 0 \, ,$$

ce qui implique que le système précédent est équivalent à l'équation $a = 0$, qui est impossible. Conclusion : la relation (2.106) n'existe pas.

1.2.9 — $i_1 = i_2 = i+1$.

Dans ce cas $p_1 = i+1$ et $p_2 = i+n-1$, ce qui veut dire que toutes les conditions de biorthogonalité sont vérifiées sauf celle correspondante à $p = i$.

Voyons si la relation

$$P_n^{(i,j)} = a(P_{n+1}^{(i+1,j)} - x^2 P_{n-1}^{(i+1,j+2)}), \tag{2.107}$$

où

$$a = \frac{1}{a_n^{(i+1,j,n+1)} - a_{n-2}^{(i+1,j+2,n-1)}} \tag{2.108}$$

existe.

Ecrivons la condition de biorthogonalité correspondante à $p = i$:

$$a(L_i(x^j P_{n+1}^{(i+1,j)}) - L_i(x^{j+2} P_{n-1}^{(i+1,j+2)})) = 0. \tag{2.109}$$

De (2.5) et (2.8), nous savons que

$$L_i(x^j P_{n+1}^{(i+1,j)}) = (-1)^{n+1} D_{n+2}^{(i,j)} / D_{n+1}^{(i+1,j)} \neq 0$$

et

$$L_i(x^{j+2} P_{n-1}^{(i+1,j+2)}) = (-1)^{n-1} D_n^{(i,j+2)} / D_{n-1}^{(i+1,j+2)} \neq 0.$$

De (2.108) et (2.16), nous écrivons

$$a = \frac{D_{n+1}^{(i+1,j)} D_{n-1}^{(i+1,j+2)}}{\begin{pmatrix} i+1,\ldots, & i+n\,, & i+n+1 \\ j & \ldots, j+n-1, & j+n+1 \end{pmatrix} D_{n-1}^{(i+1,j+2)} - D_{n+1}^{(i+1,j)} \begin{pmatrix} i+1,\ldots, i+n-2, & i+n-1 \\ j+2,\ldots, j+n-1, & j+n+1 \end{pmatrix}}.$$

Alors, la condition de biorthogonalité (2.109) s'écrit de la façon suivante

$$\frac{D_{n+2}^{(i,j)} D_{n-1}^{(i+1,j+2)} - D_{n+1}^{(i+1,j)} D_{n}^{(i,j+2)}}{\begin{pmatrix} i+1,\ldots, & i+n\,, & i+n+1 \\ j & \ldots, j+n-1, & j+n+1 \end{pmatrix} D_{n-1}^{(i+1,j+2)} - D_{n+1}^{(i+1,j)} \begin{pmatrix} i+1,\ldots, i+n-2, & i+n-1 \\ j+2,\ldots, j+n-1, & j+n+1 \end{pmatrix}} = 0.$$

Voyons que $\exists \{L_i\}_{i=0,1,\ldots}$, et $\exists i_0, j_0, n_0 \geq 0$ tels que cette condition ne se vérifie pas. Définissons $L_i(x^j) = 1/(i+j+1)$, $i,j = 0,1,\ldots$ et choisissons $i_0 = 0$, $j_0 = 0$ et $n_0 = 4$. Nous pouvons vérifier en utilisant par exemple *Mathematica*[13], que

$$\frac{D_6^{(0,0)} D_3^{(1,2)} - D_5^{(1,0)} D_4^{(0,2)}}{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 5 \end{pmatrix} D_3^{(1,2)} - D_5^{(1,0)} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \end{pmatrix}} = 1/4480 \neq 0.$$

Pour que ce contre exemple soit valable, nous vérifions aussi l'hypothèse (2.4) pour les indices i, j et n correspondants, c'est-à-dire, nous vérifions que

$$D_n^{(i,j)} \neq 0\ , n = 1,\ldots,6\,,\ i,j = 0,\ldots,n-6.$$

Conclusion: la condition de biorthogonalité (2.109) peut ne pas être vérifiée, donc dans le cas général nous ne pouvons pas affirmer que la relation (2.107) existe.

Nous concluons que le groupe de relations numéro 1 est vide.

2 — $P_n^{(i,j)}(x) = ax^2 P_{n-1}^{(i_1,j+2)}(x) + bx^2 P_{n-1}^{(i_2,j+2)}(x)$

A partir de cette relation, nous obtenons

$$P_n^{(i,j)}(0) = 0,$$

ce qui est absurde (voir (2.10)). Conclusion : ce groupe de relations est vide.

3 — $P_n^{(i,j)}(x) = ax^{m_1} P_{n_1}^{(i_1,j+m_1)}(x) + \frac{b}{x^2} P_{n+1}^{(i_2,j-2)}(x)$

Multiplions les deux membres de cette relation par x^2 :

$$x^2 P_n^{(i,j)}(x) = ax^{m_1+2} P_{n_1}^{(i_1,j+m_1)}(x) + bP_{n+1}^{(i_2,j-2)}(x).$$

Comme m_1 peut prendre que les valeurs -1, 0 et 1, alors $m_1 + 2 \geq 1$, et de la dernière relation nous obtenons

$$P_{n+1}^{(i_2,j-2)}(0) = 0,$$

ce qui est absurde (voir (2.10)). Conclusion :ce groupe de relations est vide.

4 — $P_n^{(i,j)}(x) = \frac{a}{x^2}P_{n+1}^{(i_1,j-2)}(x) + \frac{b}{x^2}P_{n+1}^{(i_2,j-2)}(x)$

A partir de cette relation, écrivons $P_n^{(i,j)}$ dans la base canonique :

$$P_n^{(i,j)} = (a+b)x^{n-1} + \ldots + (aa_1^{(i_1,j-2,n+1)} + ba_1^{(i_2,j-2,n+1)})\frac{1}{x} + \qquad (2.110)$$

$$(aa_0^{(i_1,j-2,n+1)} + ba_0^{(i_2,j-2,n+1)})\frac{1}{x^2}.$$

Même s'il existe des coefficients a et b non nuls tels que

$$\begin{cases} aa_1^{(i_1,j-2,n+1)} + ba_1^{(i_2,j-2,n+1)} = 0 \\ aa_0^{(i_1,j-2,n+1)} + ba_0^{(i_2,j-2,n+1)} = 0 \end{cases},$$

le deuxième membre de (2.110) représente un polynôme de degré inférieur à n. Conclusion : ce groupe de relations est vide.

5 — $P_n^{(i,j)}(x) = ax^2P_{n-1}^{(i_1,j+2)}(x) + \frac{b}{x^2}P_{n+1}^{(i_2,j-2)}(x)$

Multiplions les deux membres de cette relation par x^2

$$x^2P_n^{(i,j)}(x) = ax^4P_{n-1}^{(i_1,j+2)}(x) + bP_{n+1}^{(i_2,j-2)}(x).$$

A partir de cette relation, nous obtenons

$$P_{n+1}^{(i_2,j-2)}(0) = 0,$$

ce qui est absurde (voir (2.10)). Conclusion :ce groupe de relations est vide.

En résumé, nous n'avons pas trouvé de relations nouvelles.
Reécrivons les relations $F_1, \ldots, F_{12}$, en dénotant le l−ème coefficient de F_k par $C_{(k,l)}^{(i,j,n)}$, où i, j et n sont les indices du polynôme qui figure dans le membre gauche de F_k. Si F_k a un seul coefficient ou si ses coefficient coïncident, nous les dénoterons simplement par $C_k^{(i,j,n)}$.

$$F_1 : \quad P_n^{(i,j)}(x) = xP_{n-1}^{(i,j+1)}(x) + C_1^{(i,j,n)}P_{n-1}^{(i,j)}(x)$$

$$C_1^{(i,j,n)} = -\frac{L_{i+n-1}(x^{j+1}P_{n-1}^{(i,j+1)})}{L_{i+n-1}(x^jP_{n-1}^{(i,j)})} = -\frac{D_n^{(i,j+1)}D_{n-1}^{(i,j)}}{D_{n-1}^{(i,j+1)}D_n^{(i,j)}}$$

$$F_2 : \quad P_n^{(i,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) + C_2^{(i,j,n)}P_{n-1}^{(i+1,j)}(x)$$

$$C_2^{(i,j,n)} = -\frac{L_i(x^{j+1}P_{n-1}^{(i+1,j+1)})}{L_i(x^jP_{n-1}^{(i+1,j)})} = -\frac{D_n^{(i,j+1)}D_{n-1}^{(i+1,j)}}{D_{n-1}^{(i+1,j+1)}D_n^{(i,j)}}$$

$$F_3 : \quad P_n^{(i,j)}(x) = C_{(3,1)}^{(i,j,n)} \tfrac{1}{x} P_{n+1}^{(i-1,j-1)}(x) + C_{(3,2)}^{(i,j,n)} \tfrac{1}{x} P_{n+1}^{(i,j-1)}(x)$$

$$C_{(3,1)}^{(i,j,n)} = \frac{D_{n+1}^{(i,j)} D_{n+1}^{(i-1,j-1)}}{D_{n+2}^{(i-1,j-1)} D_n^{(i,j)}}, \quad C_{(3,2)}^{(i,j,n)} = -\frac{D_{n+1}^{(i,j-1)} D_{n+1}^{(i-1,j)}}{D_{n+2}^{(i-1,j-1)} D_n^{(i,j)}}$$

$$F_4 : \quad P_n^{(i,j)}(x) = C_4^{(i,j,n)} (P_{n+1}^{(i-1,j)}(x) - P_{n+1}^{(i,j)}(x))$$

$$C_4^{(i,j,n)} = \frac{D_{n+1}^{(i-1,j)} D_{n+1}^{(i,j)}}{D_n^{(i,j)} D_{n+2}^{(i-1,j)}}$$

$$F_5 : \quad P_n^{(i,j)}(x) = P_n^{(i-1,j)}(x) + C_5^{(i,j,n)} P_{n-1}^{(i,j)}(x)$$

$$C_5^{(i,j,n)} = -\frac{L_{i+n-1}(x^j P_n^{(i-1,j)})}{L_{i+n-1}(x^j P_{n-1}^{(i,j)})} = -\frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j)}}{D_n^{(i-1,j)} D_n^{(i,j)}}$$

$$F_6 : \quad P_n^{(i,j)}(x) = P_n^{(i+1,j)}(x) + C_6^{(i,j,n)} P_{n-1}^{(i+1,j)}(x)$$

$$C_6^{(i,j,n)} = -\frac{L_i(x^j P_n^{(i+1,j)})}{L_i(x^j P_{n-1}^{(i+1,j)})} = \frac{D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j)}}{D_n^{(i+1,j)} D_n^{(i,j)}}$$

$$F_7 : \quad P_n^{(i,j)}(x) = C_{(7,1)}^{(i,j,n)} P_n^{(i-1,j)}(x) + C_{(7,2)}^{(i,j,n)} x P_{n-1}^{(i,j+1)}(x)$$

$$C_{(7,1)}^{(i,j,n)} = \frac{L_{i+n-1}(x^{j+1} P_{n-1}^{(i,j+1)})}{L_{i+n-1}(x^{j+1} P_{n-1}^{(i,j+1)}) - L_{i+n-1}(x^j P_n^{(i-1,j)})} = \frac{D_n^{(i,j+1)} D_n^{(i-1,j)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}}$$

$$C_{(7,2)}^{(i,j,n)} = \frac{L_{i+n-1}(x^j P_n^{(i-1,j)})}{L_{i+n-1}(x^j P_n^{(i-1,j)}) - L_{i+n-1}(x^{j+1} P_{n-1}^{(i,j+1)})} = -\frac{D_{n+1}^{(i-1,j)} D_{n-1}^{(i,j+1)}}{D_n^{(i,j)} D_n^{(i-1,j+1)}}$$

$$F_8 : \quad P_n^{(i,j)}(x) = C_{(8,1)}^{(i,j,n)} P_n^{(i+1,j)}(x) + C_{(8,2)}^{(i,j,n)} x P_{n-1}^{(i+1,j+1)}(x)$$

$$C_{(8,1)}^{(i,j,n)} = \frac{L_i(x^{j+1} P_{n-1}^{(i+1,j+1)})}{L_i(x^{j+1} P_{n-1}^{(i+1,j+1)}) - L_i(x^j P_n^{(i+1,j)})} = \frac{D_n^{(i,j+1)} D_n^{(i+1,j)}}{D_n^{(i,j)} D_n^{(i+1,j+1)}}$$

$$C_{(8,2)}^{(i,j,n)} = \frac{L_i(x^j P_n^{(i+1,j)})}{L_i(x^j P_n^{(i+1,j)}) - L_i(x^{j+1} P_{n-1}^{(i+1,j+1)})} = \frac{D_{n+1}^{(i,j)} D_{n-1}^{(i+1,j+1)}}{D_n^{(i,j)} D_n^{(i+1,j+1)}}$$

$$F_9 : \quad P_n^{(i,j)}(x) = \frac{1}{x} P_{n+1}^{(i-1,j-1)}(x) + C_9^{(i,j,n)} \frac{1}{x} P_n^{(i,j-1)}(x)$$

$$C_9^{(i,j,n)} = \frac{D_{n+1}^{(i-1,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i-1,j-1)} D_n^{(i,j)}}$$

$$F_{10} : \quad P_n^{(i,j)}(x) = \frac{1}{x} P_{n+1}^{(i,j-1)}(x) + C_{10}^{(i,j,n)} \frac{1}{x} P_n^{(i,j-1)}(x)$$

$$C_{10}^{(i,j,n)} = \frac{D_{n+1}^{(i,j)} D_n^{(i,j-1)}}{D_{n+1}^{(i,j-1)} D_n^{(i,j)}}$$

$$F_{11} : \quad P_n^{(i,j)}(x) = C_{11}^{(i,j,n)} (x P_n^{(i,j+1)}(x) - P_{n+1}^{(i-1,j)}(x))$$

$$C_{11}^{(i,j,n)} = \frac{D_n^{(i,j+1)} D_{n+1}^{(i-1,j)}}{D_{n+1}^{(i-1,j+1)} D_n^{(i,j)}}$$

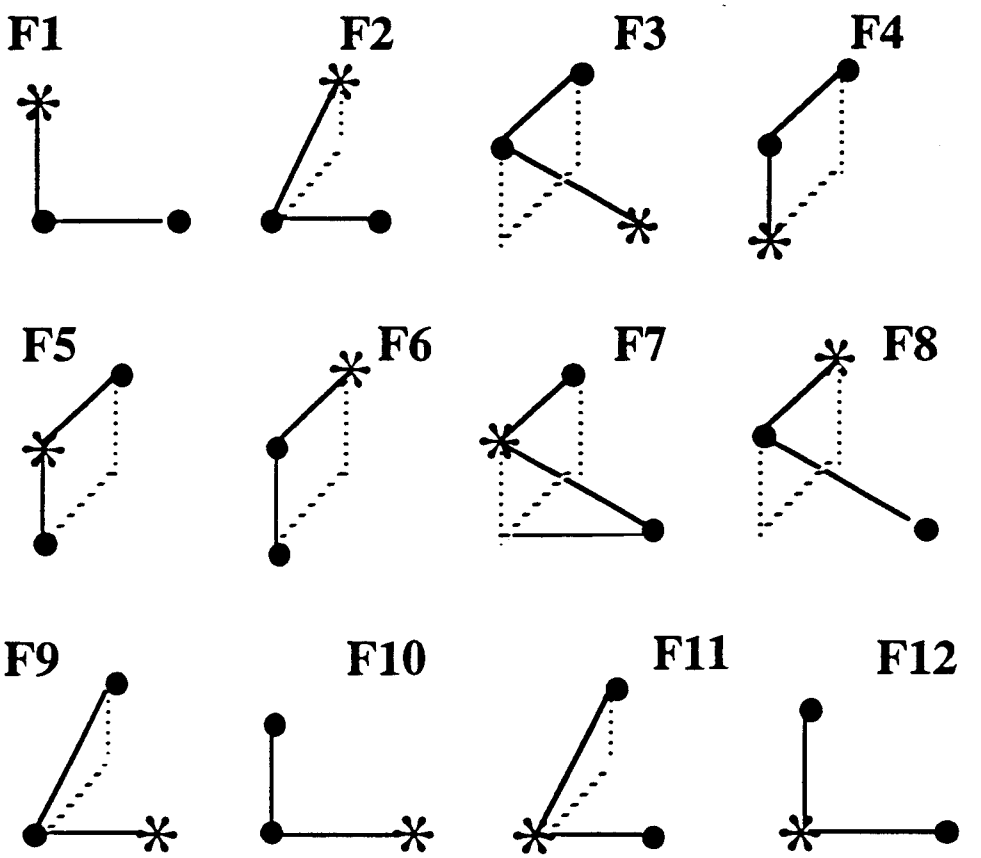
$$F_{12} : \quad P_n^{(i,j)}(x) = C_{12}^{(i,j,n)} (x P_n^{(i,j+1)}(x) - P_{n+1}^{(i,j)}(x))$$

$$C_{12}^{(i,j,n)} = \frac{D_n^{(i,j+1)} D_{n+1}^{(i,j)}}{D_{n+1}^{(i,j+1)} D_n^{(i,j)}}$$

Les relations F_1, F_2, F_5 et F_6 avaient déjà été déduites par Brezinski [3] en utilisant les identités de Sylvester et de Schweins.

En admettant la deuxième disposition des polynômes biorthogonaux adjacents dans le plan, nous schematisons les relations que nous venons d'écrire dans la figure 2.4.

Figure 2.4 : Relations parmi trois polynômes biorthogonaux adjacents consécutifs.



2.2.2 Cas particuliers

Nous nous proposons d'écrire les relations $F_1, \ldots, F_{12}$ dans les cas particuliers cités dans la section 2.1.

1. **Polynômes Orthogonaux Adjacents**

   En sachant que $L_i(x^j) = L(x^{i+j})$, $D_n^{(i,j)} = H_n^{(i+j)}$ et $P_n^{(i,j)} = P_n^{(i+j)}$; $C_{(k,l)}^{(i,j,n)} = C_{(k,l)}^{(i+j,n)}$ et nous obtenons facilement l'expression particulière des relations $F_1, \ldots, F_{12}$.

$$F_1^{poa} : \quad P_n^{(l)}(x) = xP_{n-1}^{(l+1)}(x) + C_1^{(l,n)} P_{n-1}^{(l)}(x)$$

$$C_1^{(l,n)} = -\frac{L(x^{l+n}P_{n-1}^{(l+1)})}{L(x^{l+n-1}P_{n-1}^{(l)})} = -\frac{H_n^{(l+1)}H_{n-1}^{(l)}}{H_{n-1}^{(l+1)}H_n^{(l)}}$$

$$F_2^{poa} : \quad P_n^{(l)}(x) = xP_{n-1}^{(l+2)}(x) + C_2^{(l,n)} P_{n-1}^{(l+1)}(x)$$

$$C_2^{(l,n)} = -\frac{L(x^{l+1}P_{n-1}^{(l+2)})}{L(x^l P_{n-1}^{(l+1)})} = -\frac{H_n^{(l+1)}H_{n-1}^{(l+1)}}{H_{n-1}^{(l+2)}H_n^{(l)}}$$

$$F_3^{poa} : \quad P_n^{(l)}(x) = C_{(3,1)}^{(l,n)} \tfrac{1}{x} P_{n+1}^{(l-2)}(x) + C_{(3,2)}^{(l,n)} \tfrac{1}{x} P_{n+1}^{(l-1)}(x)$$

$$C_{(3,1)}^{(l,n)} = \frac{H_{n+1}^{(l)}H_{n+1}^{(l-2)}}{H_{n+2}^{(l-2)}H_n^{(l)}}, \quad C_{(3,2)}^{(l,n)} = -\frac{H_{n+1}^{(l-1)}H_{n+1}^{(l-1)}}{H_{n+2}^{(l-2)}H_n^{(l)}}$$

$$F_4^{poa} : \quad P_n^{(l)}(x) = C_4^{(l,n)}(P_{n+1}^{(l-1)}(x) - P_{n+1}^{(l)}(x))$$

$$C_4^{(l,n)} = \frac{H_{n+1}^{(l-1)}H_{n+1}^{(l)}}{H_n^{(l)}H_{n+2}^{(l-1)}}$$

$$F_5^{poa} : \quad P_n^{(l)}(x) = P_n^{(l-1)}(x) + C_5^{(l,n)} P_{n-1}^{(l)}(x)$$

$$C_5^{(l,n)} = -\frac{L(x^{l+n-1}P_n^{(l-1)})}{L(x^{l+n-1}P_{n-1}^{(l)})} = -\frac{H_{n+1}^{(l-1)}H_{n-1}^{(l)}}{H_n^{(l-1)}H_n^{(l)}}$$

$$F_6^{poa} \; : \; P_n^{(l)}(x) = P_n^{(l+1)}(x) + C_6^{(l,n)} P_{n-1}^{(l+1)}(x)$$

$$C_6^{(l,n)} = -\frac{L(x^l P_n^{(l+1)})}{L(x^l P_{n-1}^{(l+1)})} = \frac{H_{n+1}^{(l)} H_{n-1}^{(l+1)}}{H_n^{(l+1)} H_n^{(l)}}$$

$$F_7^{poa} \; : \; P_n^{(l)}(x) = C_{(7,1)}^{(l,n)} P_n^{(l-1)}(x) + C_{(7,2)}^{(l,n)} x P_{n-1}^{(l+1)}(x)$$

$$C_{(7,1)}^{(l,n)} = \frac{L(x^{l+n} P_{n-1}^{(l+1)})}{L(x^{l+n} P_{n-1}^{(l+1)}) - L(x^{l+n-1} P_n^{(l-1)})} = \frac{H_n^{(l+1)} H_n^{(l-1)}}{H_n^{(l)} H_n^{(l)}}$$

$$C_{(7,2)}^{(l,n)} = \frac{L(x^{l+n-1} P_n^{(l-1)})}{L(x^{l+n-1} P_n^{(l-1)}) - L(x^{l+n} P_{n-1}^{(l+1)})} = -\frac{H_{n+1}^{(l-1)} H_{n-1}^{(l+1)}}{H_n^{(l)} H_n^{(l)}}$$

$$F_8^{poa} \; : \; P_n^{(l)}(x) = C_{(8,1)}^{(l,n)} P_n^{(l+1)}(x) + C_{(8,2)}^{(l,n)} P_{n-1}^{(l+2)}(x)$$

$$C_{(8,1)}^{(l,n)} = \frac{L(x^{l+1} P_{n-1}^{(l+2)})}{L(x^{l+1} P_{n-1}^{(l+2)}) - L(x^l P_n^{(l+1)})} = \frac{H_n^{(l+1)} H_n^{(l+1)}}{H_n^{(l)} H_n^{(l+2)}}$$

$$C_{(8,2)}^{(l,n)} = \frac{L(x^l P_n^{(l+1)})}{L(x^l P_n^{(l+1)}) - L(x^{l+1} P_{n-1}^{(l+2)})} = \frac{H_{n+1}^{(l)} H_{n-1}^{(l+2)}}{H_n^{(l)} H_n^{(l+2)}}$$

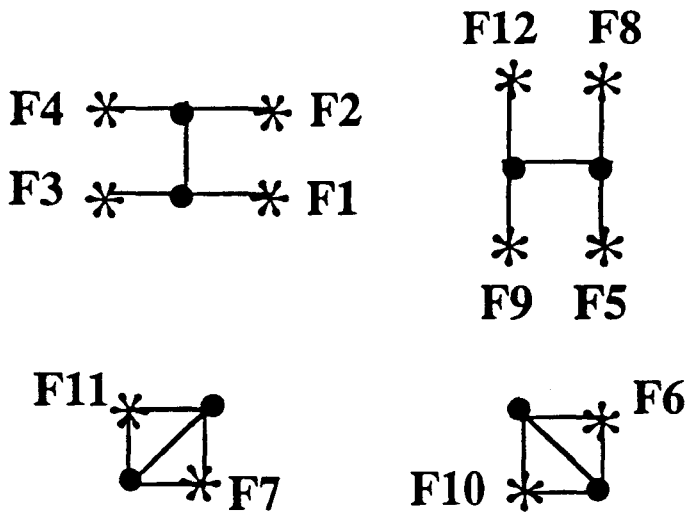
$$F_9^{poa} \; : \; P_n^{(l)}(x) = \frac{1}{x} P_{n+1}^{(l-2)}(x) + C_9^{(l,n)} \frac{1}{x} P_n^{(l-1)}(x)$$

$$C_9^{(l,n)} = \frac{H_{n+1}^{(l-1)} H_n^{(l-1)}}{H_{n+1}^{(l-2)} H_n^{(l)}}$$

$$F_{10}^{poa} \; : \; P_n^{(l)}(x) = \frac{1}{x} P_{n+1}^{(l-1)}(x) + C_{10}^{(l,n)} \frac{1}{x} P_n^{(l-1)}(x)$$

$$C_{10}^{(l,n)} = \frac{H_{n+1}^{(l)} H_n^{(l-1)}}{H_{n+1}^{(l-1)} H_n^{(l)}}$$

Figure 2.5 : Relations parmi trois polynômes orthogonaux adjacents consécutifs.



$$F_{11}^{poa} \; : \; P_n^{(l)}(x) = C_{11}^{(l,n)}(xP_n^{(l+1)}(x) - P_{n+1}^{(l-1)}(x))$$

$$C_{11}^{(l,n)} = \frac{H_n^{(l+1)}H_{n+1}^{(l-1)}}{H_{n+1}^{(l)}H_n^{(l)}}$$

$$F_{12}^{poa} \; : \; P_n^{(l)}(x) = C_{12}^{(l,n)}(xP_n^{(l+1)}(x) - P_{n+1}^{(l)}(x))$$

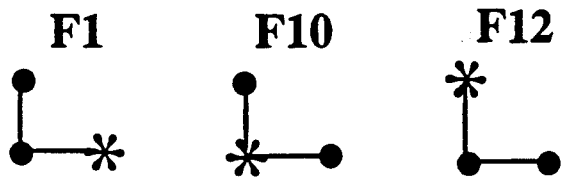
$$C_{12}^{(l,n)} = \frac{H_n^{(l+1)}H_{n+1}^{(l)}}{H_{n+1}^{(l+1)}H_n^{(l)}}$$

En admettant la disposition habituelle des polynômes orthogonaux adjacents dans le plan, que nous avons rappelée dans la figure 2.1, nous schematisons les relations que nous venons d'obtenir dans la figure 2.5.

2. Polynômes Orthogonaux Vectoriels de dimension d

En sachant que $L_{qd+r}(x^k) = L_r(x^{k+q})$, $0 \leq r < d$, $D_n^{(dq,j)} = H_{(d,n)}^{(j+q,)}$ et $P_n^{(dq,j)} = P_{(d,n)}^{(j+q,)}$, nous voyons que F_1, F_{10} et F_{12} sont les seules relations qui peuvent relier uniquement des polynômes orthogonaux vectoriels de dimension d, parce que les indices supérieurs droites sont constants dans ces formules. En remarquant que

Figure 2.6 : Relations parmi trois polyômes vectoriels de dim. d adjacents consécutifs.



$C_k^{(dq_i,j,n)} = C_k^{(j+q_i,n)}$, $k = 1, 10, 12$; nous écrivons facilement l'expression particulière des relations F_1, F_{10} et F_{12}.

$$F_1^{povd} : P_{(d,n)}^{(l)}(x) = xP_{(d,n-1)}^{(l+1)}(x) + C_1^{(l,n)} P_{(d,n-1)}^{(l)}(x)$$

$$C_1^{(l,n)} = -\frac{L_{n-1}(x^{l+1}P_{(d,n-1)}^{(l+1)})}{L_{n-1}(x^l P_{(d,n-1)}^{(l)})} = -\frac{H_{(d,n)}^{(l+1)} H_{(d,n-1)}^{(l)}}{H_{(d,n)}^{(l)} H_{(d,n-1)}^{(l+1)}}$$

$$F_{10}^{povd} : P_{(d,n)}^{(l)}(x) = \frac{1}{x} P_{(d,n+1)}^{(l-1)}(x) + C_{10}^{(l,n)} \frac{1}{x} P_{(d,n)}^{(l-1)}(x)$$

$$C_{10}^{(l,n)} = \frac{H_{(d,n+1)}^{(l)} H_{(d,n)}^{(l-1)}}{H_{(d,n+1)}^{(l-1)} H_{(d,n)}^{(l)}}$$

$$F_{12}^{povd} : P_{(d,n)}^{(l)}(x) = C_{12}^{(l,n)}(xP_{(d,n)}^{(l+1)}(x) - P_{(d,n+1)}^{(l)}(x))$$

$$C_{12}^{(l,n)} = \frac{H_{(d,n)}^{(l+1)} H_{(d,n+1)}^{(l)}}{H_{(d,n+1)}^{(l+1)} H_{(d,n)}^{(l)}}$$

La relation F_1^{povd} est une des relations du QD algorithme déduit par Van Iseghem [10].

Supposons que les polynômes $P_{(d,n)}^{(l)}$ sont disposés dans le plan de la même façon que les polynômes $P_n^{(l)}$. Dans la figure 2.6 nous schématisons les relations que nous venons d'obtenir.

3. Polynômes Orthogonaux Vectoriels de dimension -1

En sachant que $L_i(x^j) = L(x^{j-i})$, $D_n^{(i,j)} = H_n^{[j-i]}$ et $P_n^{(i,j)} = P_n^{[j-i]}$; $C_{(k,l)}^{(i,j,n)} = C_{(k,l)}^{[j-i,n]}$ et nous obtenons facilement l'expression particulière des relations $F_1, \ldots, F_{12}$.

$$F_1^{pov-1} : \quad P_n^{[l]}(x) = xP_{n-1}^{[l+1]}(x) + C_1^{[l,n]} P_{n-1}^{[l]}(x)$$

$$C_1^{[l,n]} = -\frac{L(x^{l-n+2}P_{n-1}^{[l+1]})}{L(x^{l+n-1}P_{n-1}^{[l]})} = -\frac{H_n^{[l+1]}H_{n-1}^{[l]}}{H_{n-1}^{[l+1]}H_n^{[l]}}$$

$$F_2^{pov-1} : \quad P_n^{[l]}(x) = xP_{n-1}^{[l]}(x) + C_2^{[l,n]} P_{n-1}^{[l-1]}(x)$$

$$C_2^{[l,n]} = -\frac{L(x^{l+1}P_{n-1}^{[l]})}{L(x^l P_{n-1}^{[l-1]})} = -\frac{H_n^{[l+1]}H_{n-1}^{[l-1]}}{H_{n-1}^{[l]}H_n^{[l]}}$$

$$F_3^{pov-1} : \quad P_n^{[l]}(x) = C_{(3,1)}^{[l,n]} \frac{1}{x} P_{n+1}^{[l]}(x) + C_{(3,2)}^{[l,n]} \frac{1}{x} P_{n+1}^{[l-1]}(x)$$

$$C_{(3,1)}^{[l,n]} = \frac{H_{n+1}^{[l]}H_{n+1}^{[l]}}{H_{n+2}^{[l]}H_n^{[l]}}, \quad C_{(3,2)}^{[l,n]} = -\frac{H_{n+1}^{[l-1]}H_{n+1}^{[l+1]}}{H_{n+2}^{[l]}H_n^{[l]}}$$

$$F_4^{pov-1} : \quad P_n^{[l]}(x) = C_4^{[l,n]}(P_{n+1}^{[l+1]}(x) - P_{n+1}^{[l]}(x))$$

$$C_4^{[l,n]} = \frac{H_{n+1}^{[l+1]}H_{n+1}^{[l]}}{H_n^{[l]}H_{n+2}^{[l+1]}}$$

$$F_5^{pov-1} : \quad P_n^{[l]}(x) = P_n^{[l+1]}(x) + C_5^{[l,n]} P_{n-1}^{[l]}(x)$$

$$C_5^{[l,n]} = -\frac{L(x^{l-n+1}P_n^{[l+1]})}{L(x^{l-n+1}P_{n-1}^{[l]})} = -\frac{H_{n+1}^{[l+1]}H_{n-1}^{[l]}}{H_n^{[l+1]}H_n^{[l]}}$$

$$F_6^{pov-1} : \quad P_n^{[l]}(x) = P_n^{[l-1]}(x) + C_6^{[l,n]} P_{n-1}^{[l-1]}(x)$$

$$C_6^{[l,n]} = -\frac{L(x^l P_n^{[l-1]})}{L(x^l P_{n-1}^{[l-1]})} = \frac{H_{n+1}^{[l]}H_{n-1}^{[l-1]}}{H_n^{[l-1]}H_n^{[l]}}$$

$$F_7^{pov-1} : P_n^{[l]}(x) = C_{(7,1)}^{[l,n]} P_n^{[l+1]}(x) + C_{(7,2)}^{[l,n]} x P_{n-1}^{[l+1]}(x)$$

$$C_{(7,1)}^{[l,n]} = \frac{L(x^{l-n+2} P_{n-1}^{[l+1]})}{L(x^{l-n+2} P_{n-1}^{[l+1]}) - L(x^{l-n+1} P_n^{[l+1]})} = \frac{H_n^{[l+1]} H_n^{[l+1]}}{H_n^{[l]} H_n^{[l+2]}}$$

$$C_{(7,2)}^{[l,n]} = \frac{L(x^{l-n+1} P_n^{[l+1]})}{L(x^{l-n+1} P_n^{[l+1]}) - L(x^{l-n+2} P_{n-1}^{[l+1]})} = -\frac{H_{n+1}^{[l+1]} H_{n-1}^{[l+1]}}{H_n^{[l]} H_n^{[l+2]}}$$

$$F_8^{pov-1} : P_n^{[l]}(x) = C_{(8,1)}^{[l,n]} P_n^{[l-1]}(x) + C_{(8,2)}^{[l,n]} P_{n-1}^{[l]}(x)$$

$$C_{(8,1)}^{[l,n]} = \frac{L(x^{l+1} P_{n-1}^{[l]})}{L(x^{l+1} P_{n-1}^{[l]}) - L(x^l P_n^{[l-1]})} = \frac{H_n^{[l+1]} H_n^{[l-1]}}{H_n^{[l]} H_n^{[l]}}$$

$$C_{(8,2)}^{[l,n]} = \frac{L(x^l P_n^{[l-1]})}{L(x^l P_n^{[l-1]}) - L(x^{l+1} P_{n-1}^{[l]})} = \frac{H_{n+1}^{[l]} H_{n-1}^{[l]}}{H_n^{[l]} H_n^{[l]}}$$

$$F_9^{pov-1} : P_n^{[l]}(x) = \frac{1}{x} P_{n+1}^{[l]}(x) + C_9^{[l,n]} \frac{1}{x} P_n^{[l-1]}(x)$$

$$C_9^{[l,n]} = \frac{H_{n+1}^{[l+1]} H_n^{[l-1]}}{H_{n+1}^{[l]} H_n^{[l]}}$$

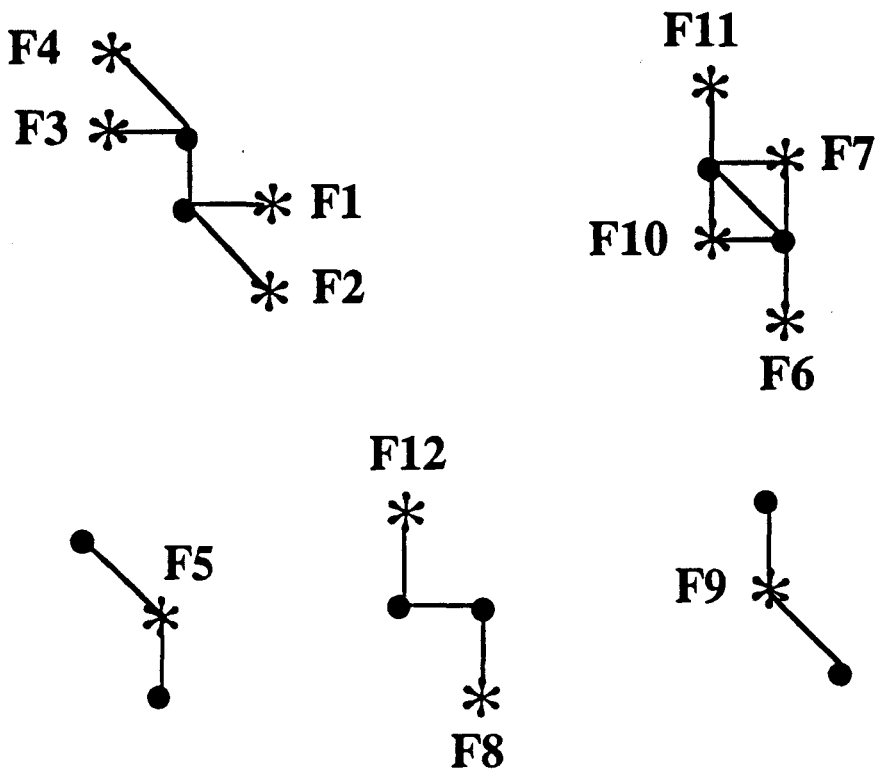
$$F_{10}^{pov-1} : P_n^{[l]}(x) = \frac{1}{x} P_{n+1}^{[l-1]}(x) + C_{10}^{[l,n]} \frac{1}{x} P_n^{[l-1]}(x)$$

$$C_{10}^{[l,n]} = \frac{H_{n+1}^{[l]} H_n^{[l-1]}}{H_{n+1}^{[l-1]} H_n^{[l]}}$$

$$F_{11}^{pov-1} : P_n^{[l]}(x) = C_{11}^{[l,n]} (x P_n^{[l+1]}(x) - P_{n+1}^{[l+1]}(x))$$

$$C_{11}^{[l,n]} = \frac{H_n^{[l+1]} H_{n+1}^{[l+1]}}{H_{n+1}^{[l+2]} H_n^{[l]}}$$

Figure 2.7 : Relations parmi trois polynômes vectoriels de dim. -1, adjacents consécutifs.



$$F_{12}^{pou-1} \; : \; P_n^{[l]}(x) = C_{12}^{[l,n]}(xP_n^{[l+1]}(x) - P_{n+1}^{[l]}(x))$$

$$C_{12}^{[l,n]} = \frac{H_n^{[l+1]}H_{n+1}^{[l]}}{H_{n+1}^{[l+1]}H_n^{[l]}}$$

Supposons que les polynômes $P_n^{[l]}$ sont disposés dans le plan de la même façon que les polynômes $P_n^{(l)}$. Dans la figure 2.7 nous schématisons les relations que nous venons d'obtenir.

Certaines de ces relations avaient déjà été déduites par Brezinski [4].

2.2.3 Conclusion

Le calcul de la suite

$$\{P_k^{(i,j)}\}_{k=0,1,\ldots}, \text{ avec les indices i et j fixés,} \qquad (2.111)$$

peut être fait à partir des conditions initialles suivantes :

$$P_0^{(i,j)} = 1, \forall i, j \in N_0,$$

en utilisant uniquement la relation F_1.

L'implémentation récursive et itérative de cette méthode de calcul, dans le langage *Mathematica*[13] est faite de façon analogue à ce qu'on a fait dans la section 1.3.1, pour les suites $\{L_k^{(i,j)}(G)\}_{k=0,1,...}$ et $\{c(x_k^{(i,j)})\}_{k=0,1,...}$, avec les indices i et j fixés.

Pour calculer $\{P_k^{(i,j)}\}_{k=0,...,n}$, avec les indices i et j fixés, nous sommes obligés de passer par le calcul des polynômes

$$\{P_k^{(i,j+l)}\}_{k=0,...,n\,;\,l=0,...,n-k}, \tag{2.112}$$

et des scalaires

$$\{L_{i+m}(x^{j+l}P_k^{(i,j+l)})\}_{k=0,...,n\,;\,l=0,...,n-k\,;\,m=k,...,n-1}. \tag{2.113}$$

Ceci entraîne un volume considérable d'espace de mémoire à utiliser et de calculs à faire. En outre, les éléments (2.112) et (2.113) doivent exister.

Le but de ce chapitre était de déduire toutes les relations de récurrence parmi trois polynômes biorthogonaux adjacents consécutifs, et d'essayer d'utiliser simultanément ces relations pour calculer la table des polynômes biorthogonaux, ou des suites particulières de polynômes biorthogonaux, en dépensant moins de mémoire et en faisant moins de calculs.

L'utilisation simultanée des relations $F_1, \ldots, F_{12}$ s'avére sans intérêt.

Dans le cas des polynômes orthogonaux adjacents, nous savons qu' à partir des conditions initialles $P_0^{(n)} = 1, \forall n \geq 0$; en utilisant simultanément les relations $F_1^{poa}, \ldots, F_{12}^{poa}$; il est possible de calculer la table de ces polynômes en ne gardant en mémoire que deux polynômes à la fois. Cette méthode de calcul a été implémentée par Brezinski en *Fortran* (voir [2]). Nous pouvons aussi calculer des suites particulières de ces polynômes, en utilisant simultanément deux relations et en ne gardant que deux polynômes en mémoire.

Dans le cas des polynômes orthogonaux vectoriels de dimension -1, il est possible de calculer des suites particulières intéréssantes de ces polynômes en utilisant simultanément deux relations choisies parmi $F_1^{pov-1}, \ldots, F_{12}^{pov-1}$, et en ne gardant que deux polynômes en mémoire (voir Brezinski [4]).

A partir de $F_1^{pov-1}, \ldots, F_{12}^{pov-1}$, il est possible de déduire d'autres relations à trois termes. L'utilisation de toutes ces relations, en partant des conditions initialles $P_0^{[k]} = 1, \forall k \in Z$; permet le calcul de la table des $P_n^{[k]}, \forall n \in N_0, \forall k \in Z$ en ne gardant en mémoire que deux polynômes.

2.3 Algorithmes de type QD

Dans cette section, nous chercherons à résoudre le problème du calcul des coefficients des relations de la section précédente.

Nous ne devons pas envisager de calculer ces coefficients à partir de leurs expressions comme rapports de déterminants. Rappelons que le calcul d'un déterminant de dimension n coûte $n.n!$ multiplications. Cherchons plutôt des relations parmi ces coefficients afin de constituer des algorithmes qui permettent leur calcul de façon efficace et si possible stable.

2.3.1 Déduction

Voyons que les relations $F_1, \ldots, F_{12}$ ne sont pas indépendantes.

Commençons par montrer que F_1, F_{10} et F_{12} sont équivalentes, aussi bien que F_2, F_9 et F_{11}; F_4, F_5 et F_6; et enfin F_3, F_7 et F_8. Nous divisions ainsi les 12 relations en 4 groupes.

Choisissons un représentant de chacun de ces 4 groupes, par exemple F_1 pour le premier groupe, F_2 pour le deuxième, F_6 pour le troisième et F_8 pour le quatrième. Ce choix est complètement arbitraire.

En résolvant F_1, F_2, F_6 et F_8 par rapport à chacun des deux polynômes qui figurent dans les deuxièmes membres de ces relations, nous obtenons respectivement :

$$P_{n-1}^{(i,j+1)}(x) = \tfrac{1}{x}P_n^{(i,j)}(x) - C_1^{(i,j,n)}\tfrac{1}{x}P_{n-1}^{(i,j)}(x)$$

$$P_{n-1}^{(i,j)}(x) = \tfrac{1}{C_1^{(i,j,n)}}(P_n^{(i,j)}(x) - xP_{n-1}^{(i,j+1)}(x))$$

$$P_{n-1}^{(i+1,j+1)}(x) = \tfrac{1}{x}P_n^{(i,j)}(x) - C_2^{(i,j,n)}\tfrac{1}{x}P_{n-1}^{(i+1,j)}(x)$$

$$P_{n-1}^{(i+1,j)}(x) = \tfrac{1}{C_2^{(i,j,n)}}(P_n^{(i,j)}(x) - xP_{n-1}^{(i+1,j+1)}(x))$$

$$P_n^{(i+1,j)}(x) = P_n^{(i,j)}(x) - C_6^{(i,j,n)}P_{n-1}^{(i+1,j)}(x)$$

$$P_{n-1}^{(i+1,j)}(x) = \tfrac{1}{C_6^{(i,j,n)}}(P_n^{(i,j)}(x) - P_n^{(i+1,j)}(x))$$

$$P_{n-1}^{(i+1,j+1)}(x) = \tfrac{1}{C_{(8,2)}^{(i,j,n)}}\tfrac{1}{x}P_n^{(i,j)}(x) - \tfrac{C_{(8,1)}^{(i,j,n)}}{C_{(8,2)}^{(i,j,n)}}\tfrac{1}{x}P_n^{(i+1,j)}(x)$$

$$P_n^{(i+1,j)}(x) = \tfrac{1}{C_{(8,1)}}P_n^{(i,j)}(x) - \tfrac{C_{(8,2)}^{(i,j,n)}}{C_{(8,1)}^{(i,j,n)}}xP_{n-1}^{(i+1,j+1)}(x).$$

Changeons convenablement les indices i, j et n de telle sorte que le polynôme figurant dans le premier membre de ces relations soit $P_n^{(i,j)}$.

$$P_n^{(i,j)}(x) = \tfrac{1}{x}P_{n+1}^{(i,j-1)}(x) - C_1^{(i,j-1,n+1)}\tfrac{1}{x}P_n^{(i,j-1)}(x)$$

$$P_n^{(i,j)}(x) = \tfrac{1}{C_1^{(i,j,n+1)}}(P_{n+1}^{(i,j)}(x) - xP_n^{(i,j+1)}(x))$$

$$P_n^{(i,j)}(x) = \tfrac{1}{x}P_{n+1}^{(i-1,j-1)}(x) - C_2^{(i-1,j-1,n+1)}\tfrac{1}{x}P_n^{(i,j-1)}(x)$$

$$P_n^{(i,j)}(x) = \tfrac{1}{C_2^{(i-1,j,n+1)}}(P_{n+1}^{(i-1,j)}(x) - xP_n^{(i,j+1)}(x))$$

$$P_n^{(i,j)}(x) = P_n^{(i-1,j)}(x) - C_6^{(i-1,j,n)}P_{n-1}^{(i,j)}(x)$$

$$P_n^{(i,j)}(x) = \tfrac{1}{C_6^{(i-1,j,n+1)}}(P_{n+1}^{(i-1,j)}(x) - P_{n+1}^{(i,j)}(x))$$

$$P_n^{(i,j)}(x) = \tfrac{1}{C_{(8,2)}^{(i-1,j-1,n+1)}}\tfrac{1}{x}P_{n+1}^{(i-1,j-1)}(x) - \tfrac{C_{(8,1)}^{(i-1,j-1,n+1)}}{C_{(8,2)}^{(i-1,j-1,n+1)}}\tfrac{1}{x}P_{n+1}^{(i,j-1)}(x)$$

$$P_n^{(i,j)}(x) = \tfrac{1}{C_{(8,1)}^{(i-1,j,n)}}P_n^{(i-1,j)}(x) - \tfrac{C_{(8,2)}^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}}xP_{n-1}^{(i,j+1)}(x)$$

Ces relations sont équivalentes à F_{10}, F_{12}, F_9, F_{11}, F_5, F_4, F_3 et F_7 respectivement. Donc, en égalant les coefficients correspondants, nous obtenons :

$$C_{(3,1)}^{(i,j,n)} = \frac{1}{C_{(8,2)}^{(i-1,j-1,n+1)}} \tag{2.114}$$

$$C_{(3,2)}^{(i,j,n)} = -\frac{C_{(8,1)}^{(i-1,j-1,n+1)}}{C_{(8,2)}^{(i-1,j-1,n+1)}} \tag{2.115}$$

$$C_4^{(i,j,n)} = \frac{1}{C_6^{(i-1,j,n+1)}} \tag{2.116}$$

$$C_5^{(i,j,n)} = -C_6^{(i-1,j,n)} \tag{2.117}$$

$$C_{(7,1)}^{(i,j,n)} = \frac{1}{C_{(8,1)}^{(i-1,j,n)}} \tag{2.118}$$

$$C_{(7,2)}^{(i,j,n)} = -\frac{C_{(8,2)}^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}} \tag{2.119}$$

$$C_9^{(i,j,n)} = -C_2^{(i-1,j-1,n+1)} \tag{2.120}$$

$$C_{10}^{(i,j,n)} = -C_1^{(i,j-1,n+1)} \tag{2.121}$$

$$C_{11}^{(i,j,n)} = -\frac{1}{C_2^{(i-1,j,n+1)}} \tag{2.122}$$

$$C_{12}^{(i,j,n)} = -\frac{1}{C_1^{(i,j,n+1)}}. \tag{2.123}$$

Par correspondance avec les relations $F_1, \ldots, F_{12}$, les coefficients $C_1, \ldots, C_{12}$ se divisent en quatre groupes, à savoir : $\{C_1, C_{10}, C_{12}\}$, $\{C_2, C_9, C_{11}\}$, $\{C_4, C_5, C_6\}$ et $\{C_3, C_7, C_8\}$.

Voyons que dans chaque groupe, deux quelconques des trois coefficients s'écrivent uniquement en fonction du troisième coefficient.

Les égalités (2.121) et (2.123) donnent l'expression de C_{10} et C_{12} en fonction de C_1. A partir de ces mêmes égalités, nous écrivons immédiatement C_1 et C_{10} en fonction de C_{12}, et C_1 et C_{12} en fonction de C_{10} :

$$\begin{cases} C_1^{(i,j,n)} = -\frac{1}{C_{12}^{(i,j,n-1)}} \\ C_{10}^{(i,j,n)} = \frac{1}{C_{12}^{(i,j-1,n)}} \end{cases}, \begin{cases} C_1^{(i,j,n)} = -C_{10}^{(i,j+1,n-1)} \\ C_{12}^{(i,j,n)} = \frac{1}{C_{10}^{(i,j+1,n)}} \end{cases}.$$

Les égalités (2.120) et (2.122) donnent l'expression de C_9 et C_{11} en fonction de C_2. A partir de ces mêmes égalités, nous écrivons immédiatement C_2 et C_9 en fonction de C_{11}, et C_2 et C_{11} en fonction de C_9 :

$$\begin{cases} C_2^{(i,j,n)} = -\frac{1}{C_{11}^{(i+1,j,n-1)}} \\ C_9^{(i,j,n)} = \frac{1}{C_{11}^{(i,j-1,n)}} \end{cases}, \begin{cases} C_2^{(i,j,n)} = -C_9^{(i+1,j+1,n-1)} \\ C_{11}^{(i,j,n)} = \frac{1}{C_9^{(i,j+1,n)}} \end{cases}.$$

Les égalités (2.116) et (2.117) donnent l'expression de C_4 et C_5 en fonction de C_6. A partir de ces mêmes égalités, nous écrivons immédiatement C_4 et C_6 en fonction de C_5, et C_5 et C_6 en fonction de C_4 :

$$\begin{cases} C_4^{(i,j,n)} = -\frac{1}{C_5^{(i,j,n+1)}} \\ C_6^{(i,j,n)} = -C_5^{(i+1,j,n)} \end{cases}, \begin{cases} C_5^{(i,j,n)} = -\frac{1}{C_4^{(i,j,n-1)}} \\ C_6^{(i,j,n)} = \frac{1}{C_4^{(i+1,j,n-1)}} \end{cases}.$$

Les égalités (2.114), (2.115), (2.118) et (2.119) donnent l'expression de C_3 et C_7 en fonction de C_8. A partir de ces mêmes égalités, nous écrivons immédiatement C_3 et C_8 en fonction de C_7, et C_7 et C_8 en fonction de C_3 :

$$\begin{cases} C_{(3,1)}^{(i,j,n)} = -\frac{C_{(7,1)}^{(i,j-1,n+1)}}{C_{(7,2)}^{(i,j-1,n+1)}} \\ C_{(3,2)}^{(i,j,n)} = \frac{1}{C_{(7,2)}^{(i,j-1,n+1)}} \end{cases}, \begin{cases} C_{(8,1)}^{(i,j,n)} = \frac{1}{C_{(7,1)}^{(i+1,j,n)}} \\ C_{(8,2)}^{(i,j,n)} = -\frac{C_{(7,2)}^{(i+1,j,n)}}{C_{(7,1)}^{(i+1,j,n)}} \end{cases},$$

$$\begin{cases} C_{(7,1)}^{(i,j,n)} = -\frac{C_{(3,1)}^{(i,j+1,n-1)}}{C_{(3,2)}^{(i,j+1,n-1)}} \\ C_{(7,2)}^{(i,j,n)} = \frac{1}{C_{(3,2)}^{(i,j+1,n-1)}} \end{cases}, \begin{cases} C_{(8,1)}^{(i,j,n)} = -\frac{C_{(3,2)}^{(i+1,j+1,n-1)}}{C_{(3,1)}^{(i+1,j+1,n-1)}} \\ C_{(8,2)}^{(i,j,n)} = \frac{1}{C_{(3,1)}^{(i+1,j+1,n-1)}} \end{cases}.$$

Ainsi, par correspondance avec les formules qui représentent les quatres groupes de relations, nous pouvons représenter le premier groupe de coefficients par C_1, le deuxième par C_2, le troisième par C_6 et le quatrième par C_8. Ici, nous dénotons $C_{(k,l)}^{(i,j,n)}$ simplement par C_k.

Voyons maintenant que les quatre groupes de relations ne sont pas indépendants. En effet,

1. F_1 et $F_2 \Longrightarrow F_6, F_8$
2. F_1 et $F_6 \Longrightarrow F_2, F_8$
3. F_1 et $F_8 \Longrightarrow F_2, F_6$
4. F_2 et $F_6 \Longrightarrow F_1, F_8$
5. F_2 et $F_8 \Longrightarrow F_1, F_6$
6. F_6 et $F_8 \Longrightarrow F_1, F_2$.

Démontrons ces implications.
Nous rappelons que $C_{(k,1)}^{(i,j,n)} + C_{(k,2)}^{(i,j,n)} = 1$, $k = 3, 7, 8$.

1. F_1 et $F_2 \Longrightarrow F_6, F_8$

   Remplaçons dans F_1, i par $i+1$

$$P_n^{(i+1,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) + C_1^{(i+1,j,n)} P_{n-1}^{(i+1,j)}(x). \tag{2.124}$$

   Ceci est équivalent à

$$xP_{n-1}^{(i+1,j+1)}(x) = P_n^{(i+1,j)}(x) - C_1^{(i+1,j,n)} P_{n-1}^{(i+1,j)}(x). \tag{2.125}$$

   Remplaçons dans F_2, $xP_{n-1}^{(i+1,j+1)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = P_n^{(i+1,j)}(x) + (C_2^{(i,j,n)} - C_1^{(i+1,j,n)}) P_{n-1}^{(i+1,j)}(x).$$

   Cette relation est équivalente à F_6, donc en égalant les coefficients correspondants, nous obtenons

$$C_6^{(i,j,n)} = C_2^{(i,j,n)} - C_1^{(i+1,j,n)}. \tag{2.126}$$

   La relation (2.124) peut s'écrire de la façon suivante

$$P_{n-1}^{(i+1,j)}(x) = \frac{1}{C_1^{(i+1,j,n)}} (P_n^{(i+1,j)}(x) - xP_{n-1}^{(i+1,j+1)}(x)).$$

   Remplaçons dans F_2, $P_{n-1}^{(i+1,j)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = \frac{C_2^{(i,j,n)}}{C_1^{(i+1,j,n)}} P_n^{(i+1,j)}(x) + (1 - \frac{C_2^{(i,j,n)}}{C_1^{(i+1,j,n)}}) xP_{n-1}^{(i+1,j+1)}(x).$$

   Cette relation est équivalente à F_8, donc en égalant les coefficients correspondants, nous obtenons

$$C_{(8,1)}^{(i,j,n)} = \frac{C_2^{(i,j,n)}}{C_1^{(i+1,j,n)}} \tag{2.127}$$

$$C_{(8,2)}^{(i,j,n)} = 1 - \frac{C_2^{(i,j,n)}}{C_1^{(i+1,j,n)}}. \tag{2.128}$$

A partir des égalités de (2.114) à (2.119) , et des égalités (2.126), (2.127) et (2.128) que nous venons d'obtenir, les coefficients C_3, C_4, C_5 et C_7 peuvent aussi s'écrire en fonction de C_1 et C_2.

$$C_{(3,1)}^{(i,j,n)} = \frac{C_1^{(i,j-1,n+1)}}{C_1^{(i,j-1,n+1)} - C_2^{(i-1,j-1,n+1)}}$$

$$C_{(3,2)}^{(i,j,n)} = -\frac{C_2^{(i-1,j-1,n+1)}}{C_1^{(i,j-1,n+1)} - C_2^{(i-1,j-1,n+1)}}$$

$$C_4^{(i,j,n)} = \frac{1}{C_2^{(i-1,j,n+1)} - C_1^{(i,j,n+1)}}$$

$$C_5^{(i,j,n)} = C_1^{(i,j,n)} - C_2^{(i-1,j,n)}$$

$$C_{(7,1)}^{(i,j,n)} = \frac{C_1^{(i,j,n)}}{C_2^{(i-1,j,n)}}$$

$$C_{(7,2)}^{(i,j,n)} = 1 - \frac{C_1^{(i,j,n)}}{C_2^{(i-1,j,n)}}$$

2. F_1 et $F_6 \Longrightarrow F_2$, F_8

Remplaçons dans F_6, $P_n^{(i+1,j)}(x)$ par le deuxième membre de (2.124)

$$P_n^{(i,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) + (C_1^{(i+1,j,n)} + C_6^{(i,j,n)})P_{n-1}^{(i+1,j)}(x).$$

Cette relation est équivalente à F_2, donc en égalant les coefficients correspondants, nous obtenons

$$C_2^{(i,j,n)} = C_1^{(i+1,j,n)} + C_6^{(i,j,n)}, \tag{2.129}$$

qui est équivalente à (2.126).

La relation (2.124) peut s'écrire de la façon suivante

$$P_{n-1}^{(i+1,j)}(x) = \frac{1}{C_1^{(i+1,j,n)}}(P_n^{(i+1,j)}(x) - xP_{n-1}^{(i+1,j+1)}(x)).$$

Remplaçons dans F_6, $P_{n-1}^{(i+1,j)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = (1 + \frac{C_6^{(i,j,n)}}{C_1^{(i+1,j,n)}})P_n^{(i+1,j)}(x) - \frac{C_6^{(i,j,n)}}{C_1^{(i+1,j,n)}}xP_{n-1}^{(i+1,j+1)}(x).$$

Cette relation est équivalente à F_8, donc en égalant les coefficients correspondants, nous obtenons

$$C_{(8,1)}^{(i,j,n)} = 1 + \frac{C_6^{(i,j,n)}}{C_1^{(i+1,j,n)}} \tag{2.130}$$

$$C_{(8,2)}^{(i,j,n)} = -\frac{C_6^{(i,j,n)}}{C_1^{(i+1,j,n)}}. \tag{2.131}$$

A partir des égalités (2.114), (2.115), (2.118), (2.119), (2.120) et (2.122), et des égalités (2.129), (2.130) et (2.131) que nous venons d'obtenir, les coefficients C_3, C_7, C_9 et C_{11} peuvent aussi s'écrire en fonction de C_1 et C_6.

$$C_{(3,1)}^{(i,j,n)} = -\frac{C_1^{(i,j-1,n+1)}}{C_6^{(i-1,j-1,n+1)}}$$

$$C_{(3,2)}^{(i,j,n)} = 1 + \frac{C_1^{(i,j-1,n+1)}}{C_6^{(i-1,j-1,n+1)}}$$

$$C_{(7,1)}^{(i,j,n)} = \frac{C_1^{(i,j,n)}}{C_1^{(i,j,n)} + C_6^{(i-1,j,n)}}$$

$$C_{(7,2)}^{(i,j,n)} = \frac{C_6^{(i-1,j,n)}}{C_1^{(i,j,n)} + C_6^{(i-1,j,n)}}$$

$$C_9^{(i,j,n)} = -C_1^{(i,j-1,n+1)} - C_6^{(i-1,j-1,n+1)}$$

$$C_{11}^{(i,j,n)} = -\frac{1}{C_1^{(i,j,n+1)} + C_6^{(i-1,j,n+1)}}$$

3. F_1 et $F_8 \Longrightarrow F_2$, F_6

Remplaçons dans F_8, $P_n^{(i+1,j)}(x)$ par le deuxième membre de (2.124)

$$P_n^{(i,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) + C_1^{(i+1,j,n)} C_{(8,1)}^{(i,j,n)} P_{n-1}^{(i+1,j)}(x).$$

Cette relation est équivalente à F_2, en égalant les coefficients correspondants, nous obtenons

$$C_2^{(i,j,n)} = C_1^{(i+1,j,n)} C_{(8,1)}^{(i,j,n)}, \tag{2.132}$$

qui est équivalente à (2.127).

Remplaçons dans F_8, $xP_{n-1}^{(i+1,j+1)}(x)$ par le deuxième membre de (2.125)

$$P_n^{(i,j)}(x) = P_n^{(i+1,j)}(x) - C_1^{(i+1,j,n)} C_{(8,2)}^{(i,j,n)} P_{n-1}^{(i+1,j)}(x).$$

Cette relation est équivalente à F_6, donc en égalant les coefficients correspondants, nous obtenons

$$C_6^{(i,j,n)} = -C_1^{(i+1,j,n)} C_{(8,2)}^{(i,j,n)}, \tag{2.133}$$

qui est équivalente à (2.131).

A partir des égalités (2.116), (2.117), (2.120) et (2.122), et des égalités (2.132) et (2.133) que nous venons d'obtenir, les coefficients C_4, C_5, C_9 et C_{11} peuvent aussi s'écrire en fonction de C_1 et C_8.

$$C_4^{(i,j,n)} = -\frac{1}{C_1^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)}}$$

$$C_5^{(i,j,n)} = C_1^{(i,j,n)}C_{(8,2)}^{(i-1,j,n)}$$

$$C_9^{(i,j,n)} = -C_1^{(i,j-1,n+1)}C_{(8,1)}^{(i-1,j-1,n+1)}$$

$$C_{11}^{(i,j,n)} = -\frac{1}{C_1^{(i,j,n+1)}C_{(8,1)}^{(i-1,j,n+1)}}$$

4. F_2 et $F_6 \Longrightarrow F_1$, F_8

Remplaçons dans F_2, i par $i-1$

$$P_n^{(i-1,j)}(x) = xP_{n-1}^{(i,j+1)}(x) + C_2^{(i-1,j,n)}P_{n-1}^{(i,j)}(x). \tag{2.134}$$

Rappelons que F_6 peut être écrite de la façon suivante

$$P_n^{(i,j)}(x) = P_n^{(i-1,j)}(x) - C_6^{(i-1,j,n)}P_{n-1}^{(i,j)}(x).$$

Remplaçons dans cette relation $P_n^{(i-1,j)}(x)$ par le deuxième membre de (2.134)

$$P_n^{(i,j)}(x) = xP_{n-1}^{(i,j+1)}(x) + (C_2^{(i-1,j,n)} - C_6^{(i-1,j,n)})P_{n-1}^{(i,j)}(x).$$

Cette relation est équivalente à F_1, donc en égalant les coefficients correspondants, nous obtenons

$$C_1^{(i,j,n)} = C_2^{(i-1,j,n)} - C_6^{(i-1,j,n)}, \tag{2.135}$$

qui est équivalent à (2.126).

Rappelons que F_2 est équivalente à

$$P_{n-1}^{(i+1,j)}(x) = \frac{1}{C_2^{(i,j,n)}}(P_n^{(i,j)}(x) - xP_{n-1}^{(i+1,j+1)}(x)).$$

Remplaçons dans F_6, $P_{n-1}^{(i+1,j)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = \frac{C_2^{(i,j,n)}}{C_2^{(i,j,n)} - C_6^{(i,j,n)}}P_n^{(i+1,j)}(x) - \frac{C_6^{(i,j,n)}}{C_2^{(i,j,n)} - C_6^{(i,j,n)}}xP_{n-1}^{(i+1,j+1)}(x).$$

Cette relation est équivalente à F_8, donc en égalant les coefficients correspondants, nous obtenons

$$C_{(8,1)}^{(i,j,n)} = \frac{C_2^{(i,j,n)}}{C_2^{(i,j,n)} - C_6^{(i,j,n)}} \tag{2.136}$$

$$C_{(8,2)}^{(i,j,n)} = -\frac{C_6^{(i,j,n)}}{C_2^{(i,j,n)} - C_6^{(i,j,n)}}. \tag{2.137}$$

A partir des égalités (2.114), (2.115), (2.118), (2.119), (2.121) et (2.123) , et des égalités (2.135), (2.136) et (2.137) que nous venons d'obtenir, les coefficients C_3, C_7, C_{10} et C_{12} peuvent aussi s'écrire en fonction de C_2 et C_6.

$$C_{(3,1)}^{(i,j,n)} = 1 - \frac{C_2^{(i-1,j-1,n+1)}}{C_6^{(i-1,j-1,n+1)}}$$

$$C_{(3,2)}^{(i,j,n)} = \frac{C_2^{(i-1,j-1,n+1)}}{C_6^{(i-1,j-1,n+1)}}$$

$$C_{(7,1)}^{(i,j,n)} = 1 - \frac{C_6^{(i-1,j,n)}}{C_2^{(i-1,j,n)}}$$

$$C_{(7,2)}^{(i,j,n)} = \frac{C_6^{(i-1,j,n)}}{C_2^{(i-1,j,n)}}$$

$$C_{10}^{(i,j,n)} = C_6^{(i-1,j-1,n+1)} - C_2^{(i-1,j-1,n+1)}$$

$$C_{12}^{(i,j,n)} = \frac{1}{C_6^{(i-1,j,n+1)} - C_2^{(i-1,j,n+1)}}$$

5. F_2 et $F_8 \Longrightarrow F_1$, F_6

Rappelons que F_8 peut être écrite de la façon suivante

$$P_n^{(i,j)}(x) = \frac{1}{C_{(8,1)}^{(i-1,j,n)}} P_n^{(i-1,j)}(x) - \frac{C_{(8,2)}^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}} x P_{n-1}^{(i,j+1)}(x).$$

Remplaçons dans cette relation $P_n^{(i-1,j)}(x)$ par le deuxième membre de (2.134)

$$P_n^{(i,j)}(x) = x P_{n-1}^{(i,j+1)}(x) + \frac{C_2^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}} P_{n-1}^{(i,j)}(x).$$

Cette relation est équivalente à F_1, donc en égalant les coefficients correspondants, nous obtenons

$$C_1^{(i,j,n)} = \frac{C_2^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}}, \tag{2.138}$$

qui est équivalente à (2.127).

F_2 est équivalente à

$$xP_{n-1}^{(i+1,j+1)}(x) = P_n^{(i,j)}(x) - C_2^{(i,j,n)} P_{n-1}^{(i+1,j)}(x).$$

Remplaçons dans F_8 $xP_{n-1}^{(i+1,j+1)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = P_n^{(i+1,j)}(x) - \frac{C_2^{(i,j,n)} C_{(8,2)}^{(i,j,n)}}{C_{(8,1)}^{(i,j,n)}} P_{n-1}^{(i+1,j)}(x).$$

Cette relation est équivalente à F_6, donc en égalant les coefficients correspondants, nous obtenons

$$C_6^{(i,j,n)} = -\frac{C_2^{(i,j,n)} C_{(8,2)}^{(i,j,n)}}{C_{(8,1)}^{(i,j,n)}}, \tag{2.139}$$

qui est équivalente à (2.136) ou (2.137).

A partir des égalités (2.116), (2.117), (2.121) et (2.123), et des égalités (2.138) et (2.139) que nous venons d'obtenir, les coefficients C_4, C_5, C_{10} et C_{12} peuvent aussi s'écrire en fonction de C_2 et C_8.

$$C_4^{(i,j,n)} = -\frac{C_{(8,1)}^{(i-1,j,n+1)}}{C_2^{(i-1,j,n+1)} C_{(8,2)}^{(i-1,j,n+1)}}$$

$$C_5^{(i,j,n)} = \frac{C_2^{(i-1,j,n)} C_{(8,2)}^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}}$$

$$C_{10}^{(i,j,n)} = -\frac{C_2^{(i-1,j-1,n+1)}}{C_{(8,1)}^{(i-1,j-1,n+1)}}$$

$$C_{12}^{(i,j,n)} = -\frac{C_{(8,1)}^{(i-1,j,n+1)}}{C_2^{(i-1,j,n+1)}}$$

6. F_6 et $F_8 \Longrightarrow F_1$, F_2

Remplaçons dans F_6, i par $i-1$

$$P_n^{(i-1,j)}(x) = P_n^{(i,j)}(x) + C_6^{(i-1,j,n)} P_{n-1}^{(i,j)}(x). \tag{2.140}$$

Rappelons que F_8 peut s'écrire de la façon suivante

$$P_n^{(i,j)}(x) = \frac{1}{C_{(8,1)}^{(i-1,j,n)}} P_n^{(i-1,j)}(x) - \frac{C_{(8,2)}^{(i-1,j,n)}}{C_{(8,1)}^{(i-1,j,n)}} xP_{n-1}^{(i,j+1)}(x).$$

Remplaçons dans cette relation $P_n^{(i-1,j)}(x)$ par le deuxième membre de (2.140)

$$P_n^{(i,j)}(x) = xP_{n-1}^{(i,j+1)}(x) - \frac{C_6^{(i-1,j,n)}}{C_{(8,2)}^{(i-1,j,n)}} P_{n-1}^{(i,j)}(x).$$

Cette relation est équivalente à F_1, donc en égalant les coefficients correspondants, nous obtenons

$$C_1^{(i,j,n)} = -\frac{C_6^{(i-1,j,n)}}{C_{(8,2)}^{(i-1,j,n)}}, \tag{2.141}$$

qui est équivalente à (2.131).

Rappelons que F_6 est équivalente à

$$P_n^{(i+1,j)}(x) = P_n^{(i,j)}(x) - C_6^{(i,j,n)} P_{n-1}^{(i+1,j)}(x).$$

Remplaçons dans F_8, $P_n^{(i+1,j)}(x)$ par le deuxième membre de la relation précédente

$$P_n^{(i,j)}(x) = xP_{n-1}^{(i+1,j+1)}(x) - \frac{C_6^{(i,j,n)} C_{(8,1)}^{(i,j,n)}}{C_{(8,2)}^{(i,j,n)}} P_{n-1}^{(i+1,j)}(x).$$

Cette relation est équivalente à F_2, donc en égalant les coefficients correspondants, nous obtenons

$$C_2^{(i,j,n)} = -\frac{C_6^{(i,j,n)} C_{(8,1)}^{(i,j,n)}}{C_{(8,2)}^{(i,j,n)}}, \tag{2.142}$$

qui est équivalente à (2.136) ou (2.137).

A partir des égalités (2.120), (2.121), (2.122) et (2.123) , et des égalités (2.141) et (2.142) que nous venons d'obtenir, les coefficients C_9, C_{10}, C_{11} et C_{12} peuvent aussi s'écrire en fonction de C_6 et C_8.

$$C_9^{(i,j,n)} = \frac{C_{(8,1)}^{(i-1,j-1,n+1)} C_6^{(i-1,j-1,n+1)}}{C_{(8,2)}^{(i-1,j-1,n+1)}}$$

$$C_{10}^{(i,j,n)} = \frac{C_6^{(i-1,j-1,n+1)}}{C_{(8,2)}^{(i-1,j-1,n+1)}}$$

$$C_{11}^{(i,j,n)} = \frac{C_{(8,2)}^{(i-1,j,n+1)}}{C_{(8,1)}^{(i-1,j,n+1)} C_6^{(i-1,j,n+1)}}$$

$$C_{12}^{(i,j,n)} = -\frac{C_{(8,2)}^{(i-1,j,n+1)}}{C_6^{(i-1,j,n+1)}}$$

Conclusion :
- Parmi $F_1, \ldots, F_{12}$ il n'y a que deux relations indépendantes.
- Chaque coefficient peut toujours s'écrire en fonction de deux autres coefficients quelconques à condition qu'ils appartiennent tous les trois à des groupes différents.

Nous venons d'écrire tous les coefficients en fonction soit de C_1 et C_2, soit de C_1 et C_6, soit de C_2 et C_6, soit de C_1 et C_8, soit de C_2 et C_8, soit de C_6 et C_8. Donc, à partir des tables de chacun de ces paires de coefficients, nous pouvons facilement calculer les tables de tous les autres coefficients.

Dans une première phase, cherchons des algorithmes capables de calculer C_1, C_2, C_6 et C_8.

Comme les C_i s'écrivent comme rapports de déterminants, nous pouvons chercher des relations entre eux en utilisant les identités de Sylvester, Schweins et autres. Mais nous pouvons aussi utiliser une autre méthode, qui consiste à obtenir deux relations différentes parmi les mêmes polynômes biorthogonaux, et après à égaler les coefficients correspondants. Cette procédure va nous permetre de déduire des relations entre les coefficients C_1 et C_2, C_1 et C_6, et C_2 et C_6.

Commençons par chercher des relations entre les coefficients C_1 et C_2.

Nous pouvons écrire $P_{n+1}^{(i,j)}$ comme combinaison polynomiale de $P_{n-1}^{(i+1,j+2)}$, $P_{n-1}^{(i+1,j+1)}$ et $P_{n-1}^{(i+1,j)}$, en utilisant par exemple les deux méthodes qui suivent :

1. De F_1

$$P_{n+1}^{(i,j)} = xP_n^{(i,j+1)} + C_1^{(i,j,n+1)} P_n^{(i,j)}.$$

Remplaçons dans cette relation $P_n^{(i,j+1)}$ et $P_n^{(i,j)}$ par leurs expressions données par F_2, à savoir

$$P_n^{(i,j+1)} = xP_{n-1}^{(i+1,j+2)} + C_2^{(i,j+1,n)} P_{n-1}^{(i+1,j+1)}$$

et

$$P_n^{(i,j)} = xP_{n-1}^{(i+1,j+1)} + C_2^{(i,j,n)} P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$\begin{aligned} P_{n+1}^{(i,j)} = x^2 P_{n-1}^{(i+1,j+2)} + (C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)}) x P_{n-1}^{(i+1,j+1)} + \\ + C_1^{(i,j,n+1)} C_2^{(i,j,n)} P_{n-1}^{(i+1,j)}. \end{aligned} \tag{2.143}$$

2. De F_2

$$P_{n+1}^{(i,j)} = xP_n^{(i+1,j+1)} + C_2^{(i,j,n+1)} P_n^{(i+1,j)}.$$

Remplaçons dans cette relation $P_n^{(i+1,j+1)}$ et $P_n^{(i+1,j)}$ par leurs expressions données par F_1, à savoir

$$P_n^{(i+1,j+1)} = xP_{n-1}^{(i+1,j+2)} + C_1^{(i+1,j+1,n)} P_{n-1}^{(i+1,j+1)}$$

et

$$P_n^{(i+1,j)} = xP_{n-1}^{(i+1,j+1)} + C_1^{(i+1,j,n)} P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$P_{n+1}^{(i,j)} = x^2 P_{n-1}^{(i+1,j+2)} + (C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)}) x P_{n-1}^{(i+1,j+1)} + \\ + C_1^{(i+1,j,n)} C_2^{(i,j,n+1)} P_{n-1}^{(i+1,j)}. \tag{2.144}$$

A partir des relations (2.143) et (2.144) écrivons les trois derniers termes de $P_{n+1}^{(i,j)}$ dans la base canonique; nous obtenons respectivement :

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} + \\ + (a_{n-2}^{(i+1,j+2,n-1)} + C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)}) x^n + \\ + (a_{n-3}^{(i+1,j+2,n-1)} + a_{n-2}^{(i+1,j+1,n-1)} (C_1^{(i,j,n+1)} + \\ + C_2^{(i,j+1,n)}) + C_1^{(i,j,n+1)} C_2^{(i,j,n)}) x^{n-1} + \dots \end{aligned} \tag{2.145}$$

et

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} + \\ + (a_{n-2}^{(i+1,j+2,n-1)} + C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)}) x^n + \\ + (a_{n-3}^{(i+1,j+2,n-1)} + a_{n-2}^{(i+1,j+1,n-1)} (C_1^{(i+1,j+1,n)} + \\ + C_2^{(i,j,n+1)}) + C_1^{(i+1,j,n)} C_2^{(i,j,n+1)}) x^{n-1} + \dots . \end{aligned} \tag{2.146}$$

En égalant les coefficients correspondants dans (2.145) et (2.146), nous obtenons :

$$\boxed{\begin{aligned} C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)} &= C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)} \\ C_1^{(i,j,n+1)} C_2^{(i,j,n)} &= C_1^{(i+1,j,n)} C_2^{(i,j,n+1)}. \end{aligned}} \tag{2.147}$$

Cherchons maintenant des relations entre les coefficients C_1 et C_6.
Nous pouvons écrire $P_{n+1}^{(i,j)}$ comme combinaison polynomiale de $P_n^{(i,j+1)}$, $P_n^{(i+1,j)}$ et $P_{n-1}^{(i+1,j)}$ en utilisant par exemple les deux méthodes qui suivent :

1. De F_1

$$P_{n+1}^{(i,j)} = x P_n^{(i,j+1)} + C_1^{(i,j,n+1)} P_n^{(i,j)}.$$

Remplaçons dans cette relation $P_n^{(i,j)}$ par son expression donnée par F_6, à savoir

$$P_n^{(i,j)} = P_n^{(i+1,j)} + C_6^{(i,j,n)} P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$P_{n+1}^{(i,j)} = x P_n^{(i,j+1)} + C_1^{(i,j,n+1)} P_n^{(i+1,j)} + \\ + C_1^{(i,j,n+1)} C_6^{(i,j,n)} P_{n-1}^{(i+1,j)}. \tag{2.148}$$

2. De F_2

$$P_{n+1}^{(i,j)} = xP_n^{(i+1,j+1)} + C_2^{(i,j,n+1)}P_n^{(i+1,j)}.$$

Remplaçons dans cette relation $P_n^{(i+1,j+1)}$ par son expression donnée par F_5, à savoir

$$P_n^{(i+1,j+1)} = P_n^{(i,j+1)} + C_5^{(i+1,j+1,n)}P_{n-1}^{(i+1,j+1)}.$$

Nous obtenons ainsi

$$P_{n+1}^{(i,j)} = xP_n^{(i,j+1)} + C_5^{(i+1,j+1,n)}xP_{n-1}^{(i+1,j+1)} + C_2^{(i,j,n+1)}P_n^{(i+1,j)}.$$

Remplaçons dans cette relation $P_{n-1}^{(i+1,j+1)}$ par son expression donnée par F_{10}, à savoir

$$P_{n-1}^{(i+1,j+1)} = \frac{1}{x}P_n^{(i+1,j)} + C_{10}^{(i+1,j+1,n-1)}\frac{1}{x}P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$\begin{aligned} P_{n+1}^{(i,j)} = xP_n^{(i,j+1)} &+ (C_2^{(i,j,n+1)} + C_5^{(i+1,j+1,n)})P_n^{(i+1,j)} + \\ &+ C_5^{(i+1,j+1,n)}C_{10}^{(i+1,j+1,n-1)}P_{n-1}^{(i+1,j)}. \end{aligned} \tag{2.149}$$

A partir des relations (2.148) et (2.149) écrivons les trois derniers termes de $P_{n+1}^{(i,j)}$ dans la base canonique; nous obtenons respectivement :

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} &+ \\ &+ (a_{n-1}^{(i,j+1,n)} + C_1^{(i,j,n+1)})x^n + \\ &+ (a_{n-2}^{(i,j+1,n)} + a_{n-1}^{(i+1,j,n)}C_1^{(i,j,n+1)} + C_1^{(i,j,n+1)}C_6^{(i,j,n)})x^{n-1} + \ldots \end{aligned} \tag{2.150}$$

et

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} &+ \\ &+ (a_{n-1}^{(i,j+1,n)} + C_2^{(i,j,n+1)} + C_5^{(i+1,j+1,n)})x^n + \\ &+ (a_{n-2}^{(i,j+1,n)} + a_{n-1}^{(i+1,j,n)}(C_2^{(i,j,n+1)} + C_5^{(i+1,j+1,n)}) + \\ &+ C_5^{(i+1,j+1,n)}C_{10}^{(i+1,j+1,n-1)})x^{n-1} + \ldots. \end{aligned} \tag{2.151}$$

En égalant les coefficients correspondants dans (2.150) et (2.151), nous obtenons :

$$\begin{aligned} C_1^{(i,j,n+1)} &= C_2^{(i,j,n+1)} + C_5^{(i+1,j+1,n)} \\ C_1^{(i,j,n+1)}C_6^{(i,j,n)} &= C_5^{(i+1,j+1,n)}C_{10}^{(i+1,j+1,n-1)} \end{aligned} \tag{2.152}$$

Rappelons qu'en utilisant (2.121), (2.117) et (2.129), nous pouvons écrire les coefficients C_{10} en fonction de C_1, C_5 en fonction de C_6, et C_2 en fonction de C_1 et C_6 respectivement. Et alors les égalités (2.152) s'écrivent de la façon suivante :

$$\boxed{\begin{array}{c} C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)} = C_1^{(i+1,j,n+1)} + C_6^{(i,j,n+1)} \\ C_1^{(i,j,n+1)} C_6^{(i,j,n)} = C_1^{(i+1,j,n)} C_6^{(i,j+1,n)}. \end{array}} \tag{2.153}$$

Cherchons finalement des relations entre les coefficients C_2 et C_6.
Nous pouvons écrire $P_{n+1}^{(i,j)}$ comme combinaison polynomiale de $P_n^{(i+1,j+1)}$, $P_n^{(i,j)}$ et $P_{n-1}^{(i+1,j)}$ en utilisant par exemple les deux méthodes qui suivent :

1. De F_2

$$P_{n+1}^{(i,j)} = xP_n^{(i+1,j+1)} + C_2^{(i,j,n+1)} P_n^{(i+1,j)}.$$

Remplaçons dans cette relation $P_n^{(i+1,j)}$ par son expression donnée par F_5, à savoir

$$P_n^{(i+1,j)} = P_n^{(i,j)} + C_5^{(i+1,j,n)} P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$P_{n+1}^{(i,j)} = xP_n^{(i+1,j+1)} + C_2^{(i,j,n+1)} P_n^{(i,j)} + C_2^{(i,j,n+1)} C_5^{(i+1,j,n)} P_{n-1}^{(i+1,j)}. \tag{2.154}$$

2. De F_1

$$P_{n+1}^{(i,j)} = xP_n^{(i,j+1)} + C_1^{(i,j,n+1)} P_n^{(i,j)}.$$

Remplaçons dans cette relation $P_n^{(i,j+1)}$ par son expression donnée par F_6, à savoir

$$P_n^{(i,j+1)} = P_n^{(i+1,j+1)} + C_6^{(i,j+1,n)} P_{n-1}^{(i+1,j+1)}.$$

Nous obtenons ainsi

$$P_{n+1}^{(i,j)} = xP_n^{(i+1,j+1)} + C_6^{(i,j+1,n)} xP_{n-1}^{(i+1,j+1)} + C_1^{(i,j,n+1)} P_n^{(i,j)}.$$

Remplaçons dans cette relation $P_{n-1}^{(i+1,j+1)}$ par son expression donnée par F_9, à savoir

$$P_{n-1}^{(i+1,j+1)} = \frac{1}{x} P_n^{(i,j)} + C_9^{(i+1,j+1,n-1)} \frac{1}{x} P_{n-1}^{(i+1,j)}.$$

Nous obtenons ainsi

$$\begin{aligned} P_{n+1}^{(i,j)} = {} & xP_n^{(i+1,j+1)} + (C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)}) P_n^{(i,j)} + \\ & + C_6^{(i,j+1,n)} C_9^{(i+1,j+1,n-1)} P_{n-1}^{(i+1,j)}. \end{aligned} \tag{2.155}$$

A partir des relations (2.154) et (2.155) écrivons les trois derniers termes de $P_{n+1}^{(i,j)}$ dans la base canonique; nous obtenons respectivement :

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} + \\ +(a_{n-1}^{(i+1,j+1,n)} + C_2^{(i,j,n+1)})x^n + \\ +(a_{n-2}^{(i+1,j+1,n)} + a_{n-1}^{(i,j,n)} C_2^{(i,j,n+1)} + C_2^{(i,j,n+1)} C_5^{(i+1,j,n)})x^{n-1} + \dots \end{aligned} \tag{2.156}$$

et

$$\begin{aligned} P_{n+1}^{(i,j)}(x) = x^{n+1} + \\ +(a_{n-1}^{(i+1,j+1,n)} + C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)})x^n + \\ +(a_{n-2}^{(i+1,j+1,n)} + a_{n-1}^{(i,j,n)}(C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)}) + \\ +C_6^{(i,j+1,n)} C_9^{(i+1,j+1,n-1)})x^{n-1} + \dots . \end{aligned} \tag{2.157}$$

En égalant les coefficients correspondants dans (2.156) et (2.157), nous obtenons :

$$\begin{aligned} C_2^{(i,j,n+1)} &= C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)} \\ C_2^{(i,j,n+1)} C_5^{(i+1,j,n)} &= C_6^{(i,j+1,n)} C_9^{(i+1,j+1,n-1)} \end{aligned} \tag{2.158}$$

Rappelons qu'en utilisant (2.117), (2.120) et (2.135), nous pouvons écrire les coefficients C_5 en fonction de C_6, C_9 en fonction de C_2, et C_1 en fonction de C_2 et C_6 respectivement. Et alors les égalités (2.158) s'écrivent de la façon suivante :

$$\boxed{\begin{aligned} C_2^{(i,j,n+1)} + C_6^{(i+1,j+1,n)} &= C_2^{(i+1,j,n+1)} + C_6^{(i,j,n+1)} \\ C_2^{(i,j,n+1)} C_6^{(i,j,n)} &= C_2^{(i,j,n)} C_6^{(i,j+1,n)}. \end{aligned}} \tag{2.159}$$

Nous aurions pu déduire plus facilement les relations (2.153) et (2.159). En effet, ces relations s'obtiennent à partir des relations (2.147) en utilisant (2.129) et (2.135) respectivement. De la même façon, à partir des relations (2.153) nous pouvons déduire les relations (2.147) et (2.159) en utilisant (2.126) et (2.135) respectivement. Et, à partir des relations (2.159) nous pouvons déduire les relations (2.147) et (2.153) en utilisant (2.126) et (2.129) respectivement.

Nous allons déduire les relations entre les coefficients C_1 et C_8, C_2 et C_8, et C_6 et C_8 à partir des relations (2.147), (2.153) et (2.159).

Commençons par les relations entre les coefficients C_1 et C_8.

Ecrivons les coefficients C_2, qui apparaîssent dans les relations (2.147), en fonction de C_1 et C_8 en utilisant (2.132); nous obtenons :

$$C_1^{(i,j,n+1)} + C_1^{(i+1,j+1,n)} C_{(8,1)}^{(i,j+1,n)} = C_1^{(i+1,j+1,n)} + C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)} \tag{2.160}$$

et

$$C_1^{(i,j,n+1)} C_1^{(i+1,j,n)} C_{(8,1)}^{(i,j,n)} = C_1^{(i+1,j,n)} C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}. \tag{2.161}$$

Dans la relation (2.160) mettons en évidence $C_1^{(i+1,j+1,n)}$ et dans la relation (2.161) éliminons le facteur commun $C_1^{(i+1,j,n)}$; nous obtenons :

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)}(1 - C_{(8,1)}^{(i,j+1,n)}) + C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)} \tag{2.162}$$

et

$$C_1^{(i,j,n+1)} C_{(8,1)}^{(i,j,n)} = C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}.$$

Finalement, dans la relation (2.162), remplaçons $(1 - C_{(8,1)}^{(i,j+1,n)})$ par $C_{(8,2)}^{(i,j+1,n)}$; nous obtenons ainsi :

$$\boxed{\begin{aligned} C_1^{(i,j,n+1)} &= C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)} + C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)} \\ C_1^{(i,j,n+1)} C_{(8,1)}^{(i,j,n)} &= C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}. \end{aligned}} \tag{2.163}$$

Nous pouvons obtenir ces mêmes relations ou des relations équivalentes à celles-ci à partir de (2.153) et (2.159) en utilisant (2.133), et (2.132) et (2.133) respectivement.

Cherchons maintenant des relations entre les coefficients C_2 et C_8.

Ecrivons les coefficients C_1, qui apparaîssent dans les relations (2.147), en fonction de C_2 et C_8 en utilisant (2.138); nous obtenons :

$$\frac{C_2^{(i-1,j,n+1)}}{C_{(8,1)}^{(i-1,j,n+1)}} + C_2^{(i,j+1,n)} = \frac{C_2^{(i,j+1,n)}}{C_{(8,1)}^{(i,j+1,n)}} + C_2^{(i,j,n+1)}$$

et

$$\frac{C_2^{(i-1,j,n+1)}}{C_{(8,1)}^{(i-1,j,n+1)}} C_2^{(i,j,n)} = \frac{C_2^{(i,j,n)}}{C_{(8,1)}^{(i,j,n)}} C_2^{(i,j,n+1)}.$$

Ecrivons ces relations de façon équivalente sans dénominateurs :

$$C_2^{(i-1,j,n+1)} C_{(8,1)}^{(i,j+1,n)} + C_2^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)} C_{(8,1)}^{(i,j+1,n)} = \tag{2.164}$$

$$= C_2^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)} + C_2^{(i,j,n+1)} C_{(8,1)}^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)},$$

$$C_2^{(i-1,j,n+1)} C_2^{(i,j,n)} C_{(8,1)}^{(i,j,n)} = C_2^{(i,j,n)} C_2^{(i,j,n+1)} C_{(8,1)}^{(i-1,j,n+1)}. \tag{2.165}$$

Dans la relation (2.164) mettons en évidence le facteur $C_2^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)}$ et dans la relation (2.165) éliminons le facteur commun $C_2^{(i,j,n)}$; nous obtenons :

$$C_2^{(i-1,j,n+1)} C_{(8,1)}^{(i,j+1,n)} = C_2^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)} (1 - C_{(8,1)}^{(i,j+1,n)}) + \tag{2.166}$$

$$+ C_2^{(i,j,n+1)} C_{(8,1)}^{(i,j+1,n)} C_{(8,1)}^{(i-1,j,n+1)}$$

et

$$C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j,n)} = C_{(8,1)}^{(i-1,j,n+1)}C_2^{(i,j,n+1)}.$$

Finalement, dans la relation (2.166) remplaçons $(1-C_{(8,1)}^{(i,j+1,n)})$ par $C_{(8,2)}^{(i,j+1,n)}$; nous obtenons ainsi :

$$\boxed{\begin{aligned} C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)} &= C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)} + \\ &+ C_2^{(i,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)} \\ C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j,n)} &= C_{(8,1)}^{(i-1,j,n+1)}C_2^{(i,j,n+1)}. \end{aligned}} \tag{2.167}$$

Nous pouvons obtenir ces mêmes relations ou des relations équivalentes à celles-ci à partir de (2.153) et (2.159) en utilisant (2.138) et (2.139), et (2.139) respectivement.

Cherchons enfin des relations entre les coefficients C_6 et C_8.

Ecrivons les coefficients C_1, qui apparaîssent dans les relations (2.153), en fonction de C_6 et C_8 en utilisant (2.141); nous obtenons :

$$-\frac{C_6^{(i-1,j,n+1)}}{C_{(8,2)}^{(i-1,j,n+1)}} + C_6^{(i,j+1,n)} = -\frac{C_6^{(i,j,n+1)}}{C_{(8,2)}^{(i,j,n+1)}} + C_6^{(i,j,n+1)}$$

et

$$-\frac{C_6^{(i-1,j,n+1)}}{C_{(8,2)}^{(i-1,j,n+1)}}C_6^{(i,j,n)} = -\frac{C_6^{(i,j,n)}}{C_{(8,2)}^{(i,j,n)}}C_6^{(i,j+1,n)}.$$

Ecrivons ces relations de façon équivalente sans dénominateurs :

$$C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)} + C_6^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)} = \tag{2.168}$$

$$= C_6^{(i,j,n+1)}C_{(8,2)}^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)} + C_6^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)},$$

$$C_6^{(i-1,j,n+1)}C_6^{(i,j,n)}C_{(8,2)}^{(i,j,n)} = C_6^{(i,j,n)}C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}. \tag{2.169}$$

Dans la relation (2.168) mettons en évidence le facteur $C_6^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)}$ et dans la relation (2.169) éliminons le facteur commun $C_6^{(i,j,n)}$; nous obtenons :

$$\begin{aligned} C_6^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)} &= C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)} + \\ &C_6^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)}(1 - C_{(8,2)}^{(i,j,n+1)}) \end{aligned} \tag{2.170}$$

et

$$C_6^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n)} = C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}.$$

Finalement, dans la relation (2.170) remplaçons $(1-C_{(8,2)}^{(i,j,n+1)})$ par $C_{(8,1)}^{(i,j,n+1)}$; nous obtenons ainsi :

$$\boxed{\begin{aligned} C_6^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)} &= C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n+1)} + \\ &+ C_6^{(i,j,n+1)}C_{(8,2)}^{(i-1,j,n+1)}C_{(8,1)}^{(i,j,n+1)} \\ C_6^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n)} &= C_6^{(i,j+1,n)}C_{(8,2)}^{(i-1,j,n+1)}. \end{aligned}} \tag{2.171}$$

Nous pouvons obtenir ces mêmes relations ou des relations équivalentes à celles-ci à partir de (2.147) et (2.159), en utilisant (2.141) et (2.142), et (2.142) respectivement.

Voyons comment à partir des relations (2.147), (2.153), (2.159), (2.163), (2.167) et (2.171) et des conditions initialles convenablement choisies, nous déduisons des algorithmes capables de calculer les coefficients C_1 et C_2, C_1 et C_6, C_2 et C_6, C_1 et C_8, C_2 et C_8, et C_6 et C_8.

A partir des définitions des coefficients, nous voyons que quand $n = 0$ ou $n = 1$ ils s'écrivent de façon simple en fonction des moments des fonctionnelles L_i, et sont donc facilement calculables. Alors, $C_{(k,l)}^{(i,j,0)}$ ou $C_{(k,l)}^{(i,j,1)}$ sont les conditions initialles à considérer à priori.

Commençons par l'algorithme C_1C_2.

Si nous partons des conditions initialles $C_1^{(i,j,1)}$ et $C_2^{(i,j,1)}$, les relations (2.147) ne fonctionnent pas. Essayons de déduire d'autres relations à partir de celles-là.

Dénotons les relations (2.147) par (2.147)−1 et (2.147)−2 respectivement.

De (2.147)−1

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)} - C_2^{(i,j+1,n)} \tag{2.172}$$

et

$$C_2^{(i,j,n+1)} = C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}. \tag{2.173}$$

Remplaçons dans (2.147)−2, $C_1^{(i,j,n+1)}$ par le deuxième membre de (2.172); nous obtenons :

$$(C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)} - C_2^{(i,j+1,n)})C_2^{(i,j,n)} = C_1^{(i+1,j,n)}C_2^{(i,j,n+1)}.$$

Cette relation est équivalente à

$$C_2^{(i,j,n+1)}(C_2^{(i,j,n)} - C_1^{(i+1,j,n)}) = C_2^{(i,j,n)}(C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}),$$

que nous écrivons de la façon suivante

$$C_2^{(i,j,n+1)} = C_2^{(i,j,n)}\frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}. \tag{2.174}$$

Remplaçons dans (2.147)−2, $C_2^{(i,j,n+1)}$ par le deuxième membre de (2.173); nous obtenons :

$$C_1^{(i,j,n+1)}C_2^{(i,j,n)} = C_1^{(i+1,j,n)}(C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}).$$

Cette relation est équivalente à

$$C_1^{(i,j,n+1)}(C_2^{(i,j,n)} - C_1^{(i+1,j,n)}) = C_1^{(i+1,j,n)}(C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}),$$

que nous écrivons de la façon suivante

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j,n)} \frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}. \tag{2.175}$$

A partir des relations (2.172) et (2.174), nous obtenons l'algorithme suivant :

- ALGO C_1C_2 version 1.

$$C_2^{(i,j,1)} = C_1^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_2^{(i,j,n+1)} = C_2^{(i,j,n)} \frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)} + C_2^{(i,j,n+1)} - C_2^{(i,j+1,n)}$$

A partir des relations (2.147)−2 et (2.174), nous obtenons l'algorithme suivant :

- ALGO C_1C_2 version 2.

$$C_2^{(i,j,1)} = C_1^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_2^{(i,j,n+1)} = C_2^{(i,j,n)} \frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j,n)} C_2^{(i,j,n+1)} / C_2^{(i,j,n)}$$

A partir des relations (2.173) et (2.175), nous obtenons l'algorithme suivant :

- ALGO C_1C_2 version 3.

$$C_1^{(i,j,1)} = C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j,n)} \frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}$$

$$C_2^{(i,j,n+1)} = C_1^{(i,j,n+1)} + C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}$$

A partir des relations (2.147)−2 et (2.175), nous obtenons l'algorithme suivant :

- ALGO C_1C_2 version 4.

$$C_1^{(i,j,1)} = C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j,n)} \frac{C_2^{(i,j+1,n)} - C_1^{(i+1,j+1,n)}}{C_2^{(i,j,n)} - C_1^{(i+1,j,n)}}$$

$$C_2^{(i,j,n+1)} = C_1^{(i,j,n+1)} C_2^{(i,j,n)} / C_1^{(i+1,j,n)}$$

L'algorithme C_1C_6 qui suit est obtenu directement des relations (2.153):

- ALGO C_1C_6.

$$C_1^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}, C_6^{(i,j,1)} = \frac{L_{i+1}(x^{j+1})}{L_{i+1}(x^j)} - \frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j,n)} C_6^{(i,j+1,n)} / C_6^{(i,j,n)}$$

$$C_6^{(i,j,n+1)} = C_1^{(i,j,n+1)} + C_6^{(i,j+1,n)} - C_1^{(i+1,j,n+1)}$$

L'algorithme C_2C_6 qui suit est obtenu directement des relations (2.159):

- ALGO C_2C_6.

$$C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}, C_6^{(i,j,1)} = \frac{L_{i+1}(x^{j+1})}{L_{i+1}(x^j)} - \frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_2^{(i,j,n+1)} = C_2^{(i,j,n)} C_6^{(i,j+1,n)} / C_6^{(i,j,n)}$$

$$C_6^{(i,j,n+1)} = C_2^{(i,j,n+1)} + C_6^{(i+1,j+1,n)} - C_2^{(i+1,j,n+1)}$$

Déduisons maintenant l'algorithme C_1C_8.

Si nous partons des conditions initialles $C_1^{(i,j,1)}$, $C_{(8,1)}^{(i,j,1)}$ et $C_{(8,2)}^{(i,j,1)}$ les relations (2.163) ne fonctionnent pas. Essayons de déduire d'autres relations à partir de celles-là.

Dénotons les relations (2.163) par (2.163)−1 et (2.163)−2 respectivement.

Remplaçons dans (2.163)−1, $C_1^{(i+1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}$ par le premier membre de (2.163)−2; nous obtenons :

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)} + C_1^{(i,j,n+1)} C_{(8,1)}^{(i,j,n)}.$$

Mettons en évidence $C_1^{(i,j,n+1)}$

$$C_1^{(i,j,n+1)}(1 - C_{(8,1)}^{(i,j,n)}) = C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)}.$$

Remplaçons $(1 - C_{(8,1)}^{(i,j,n)})$ par $C_{(8,2)}^{(i,j,n)}$

$$C_1^{(i,j,n+1)} C_{(8,2)}^{(i,j,n)} = C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)}. \tag{2.176}$$

A partir de cette relation et de la relation (2.163)−1, nous obtenons l'algorithme suivant :

- ALGO C_1C_8 version 1.

$$C_1^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_{(8,1)}^{(i,j,1)} = \frac{L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})}$$

$$C_{(8,2)}^{(i,j,1)} = \frac{L_i(x^j)L_{i+1}(x^{j+1}) - L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})} = 1 - C_{(8,1)}^{(i,j,1)}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)} / C_{(8,2)}^{(i,j,n)}$$

$$C_{(8,1)}^{(i,j,n+1)} = (C_1^{(i,j,n+1)} - C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)}) / C_1^{(i+1,j,n+1)}$$

$$C_{(8,2)}^{(i,j,n+1)} = 1 - C_{(8,1)}^{(i,j,n+1)}$$

A partir des relations (2.163)−2 et (2.176), nous obtenons l'algorithme suivant :

- ALGO C_1C_8 version 2.

$$C_1^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_{(8,1)}^{(i,j,1)} = \frac{L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})}$$

$$C_{(8,2)}^{(i,j,1)} = \frac{L_i(x^j)L_{i+1}(x^{j+1}) - L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})} = 1 - C_{(8,1)}^{(i,j,1)}$$

$$C_1^{(i,j,n+1)} = C_1^{(i+1,j+1,n)} C_{(8,2)}^{(i,j+1,n)} / C_{(8,2)}^{(i,j,n)}$$

$$C_{(8,1)}^{(i,j,n+1)} = C_1^{(i,j,n+1)} C_{(8,1)}^{(i,j,n)} / C_1^{(i+1,j,n+1)}$$

$$C_{(8,2)}^{(i,j,n+1)} = 1 - C_{(8,1)}^{(i,j,n+1)}$$

Déduisons maintenant l'algorithme C_2C_8.

Si nous partons des conditions initialles $C_2^{(i,j,1)}$, $C_{(8,1)}^{(i,j,1)}$ et $C_{(8,2)}^{(i,j,1)}$ les relations (2.167) ne fonctionnent pas. Essayons d'obtenir d'autres relations à partir de celles-là.

Dénotons les relations (2.167) par (2.167)−1 et (2.167)−2 respectivement.

Remplaçons dans (2.167)−1, $C_2^{(i,j,n+1)}C_{(8,1)}^{(i-1,j,n+1)}$ par le premier membre de (2.167)−2; nous obtenons :

$$C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)} = C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)} + C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,1)}^{(i,j,n)}.$$

Mettons en évidence $C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)}$

$$C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)}(1 - C_{(8,1)}^{(i,j,n)}) = C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}.$$

Remplaçons $(1 - C_{(8,1)}^{(i,j,n)})$ par $C_{(8,2)}^{(i,j,n)}$

$$C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,2)}^{(i,j,n)} = C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}. \tag{2.177}$$

Nous observons que cette relation s'obtient de la relation (2.153)−2 en écrivant les coefficients C_1 et C_6 en fonction de C_2 et C_8.

De cette relation, nous écrivons

$$C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)} = \frac{C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}}{C_{(8,2)}^{(i,j,n)}}.$$

Remplaçons dans la relation (2.167)−1, $C_2^{(i-1,j,n+1)}C_{(8,1)}^{(i,j+1,n)}$ par le deuxième membre de la relation précédente; nous obtenons :

$$C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)} = C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}C_{(8,2)}^{(i,j,n)} +$$

$$+C_2^{(i,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n)}.$$

Mettons en évidence $C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}$, nous obtenons

$$C_2^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j+1,n)}(1 - C_{(8,2)}^{(i,j,n)}) = C_2^{(i,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,1)}^{(i-1,j,n+1)}C_{(8,2)}^{(i,j,n)}.$$

Remplaçons $(1 - C_{(8,2)}^{(i,j,n)})$ par $C_{(8,1)}^{(i,j,n)}$ et éliminons le facteur commun $C_{(8,1)}^{(i-1,j,n+1)}$, nous obtenons :

$$C_2^{(i,j+1,n)}C_{(8,2)}^{(i,j+1,n)}C_{(8,1)}^{(i,j,n)} = C_2^{(i,j,n+1)}C_{(8,1)}^{(i,j+1,n)}C_{(8,2)}^{(i,j,n)}. \tag{2.178}$$

Nous observons que cette relation peut-être obtenue à partir de (2.159)−2 en écrivant les coefficients C_6 en fonction de C_2 et C_8.

A partir des relations (2.167)−1 et (2.178), nous obtenons l'algorithme suivant :

- ALGO C_2C_8 version 1.

$$C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_{(8,1)}^{(i,j,1)} = \frac{L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})}$$

$$C_{(8,2)}^{(i,j,1)} = \frac{L_i(x^j)L_{i+1}(x^{j+1}) - L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})} = 1 - C_{(8,1)}^{(i,j,1)}$$

$$C_2^{(i,j,n+1)} = (C_2^{(i,j+1,n)} C_{(8,2)}^{(i,j+1,n)} C_{(8,1)}^{(i,j,n)}) / (C_{(8,1)}^{(i,j+1,n)} C_{(8,2)}^{(i,j,n)})$$

$$C_{(8,1)}^{(i,j,n+1)} = \frac{C_2^{(i,j,n+1)} C_{(8,1)}^{(i+1,j+1,n)}}{C_2^{(i+1,j+1,n)} C_{(8,2)}^{(i+1,j+1,n)} + C_2^{(i+1,j,n+1)} C_{(8,1)}^{(i+1,j+1,n)}}$$

$$C_{(8,2)}^{(i,j,n+1)} = 1 - C_{(8,1)}^{(i,j,n+1)}$$

A partir des relations (2.167)−2 et (2.178), nous obtenons l'algorithme suivant :

- ALGO C_2C_8 version 2.

$$C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_{(8,1)}^{(i,j,1)} = \frac{L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})}$$

$$C_{(8,2)}^{(i,j,1)} = \frac{L_i(x^j)L_{i+1}(x^{j+1}) - L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})} = 1 - C_{(8,1)}^{(i,j,1)}$$

$$C_2^{(i,j,n+1)} = (C_2^{(i,j+1,n)} C_{(8,2)}^{(i,j+1,n)} C_{(8,1)}^{(i,j,n)}) / (C_{(8,1)}^{(i,j+1,n)} C_{(8,2)}^{(i,j,n)})$$

$$C_{(8,1)}^{(i,j,n+1)} = C_2^{(i,j,n+1)} C_{(8,1)}^{(i+1,j,n)} / C_2^{(i+1,j,n+1)}$$

$$C_{(8,2)}^{(i,j,n+1)} = 1 - C_{(8,1)}^{(i,j,n+1)}$$

A partir des relations (2.177) et (2.178), nous obtenons l'algorithme suivant :

- ALGO C_2C_8 version 3.

$$C_2^{(i,j,1)} = -\frac{L_i(x^{j+1})}{L_i(x^j)}$$

$$C_{(8,1)}^{(i,j,1)} = \frac{L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})}$$

$$C_{(8,2)}^{(i,j,1)} = \frac{L_i(x^j)L_{i+1}(x^{j+1}) - L_i(x^{j+1})L_{i+1}(x^j)}{L_i(x^j)L_{i+1}(x^{j+1})} = 1 - C_{(8,1)}^{(i,j,1)}$$

$$C_2^{(i,j,n+1)} = (C_2^{(i,j+1,n)} C_{(8,2)}^{(i,j+1,n)} C_{(8,1)}^{(i,j,n)}) / (C_{(8,1)}^{(i,j+1,n)} C_{(8,2)}^{(i,j,n)})$$

$$C_{(8,1)}^{(i,j,n+1)} = (C_2^{(i,j,n+1)} C_{(8,1)}^{(i+1,j+1,n)} C_{(8,2)}^{(i+1,j,n)}) / (C_2^{(i+1,j+1,n)} C_{(8,2)}^{(i+1,j+1,n)})$$

$$C_{(8,2)}^{(i,j,n+1)} = 1 - C_{(8,1)}^{(i,j,n+1)}$$

Pour le calcul des coefficients C_6 et C_8 nous n'avons pas trouvé de relations de récurrence capables de constituer un algorithme qui utilise les conditions initialles $C_6^{(i,j,1)}$, $C_{(8,1)}^{(i,j,1)}$ et $C_{(8,2)}^{(i,j,1)}$. Cependant, les coefficients C_6 et C_8 peuvent toujours être calculés en utilisant simultanément deux algorithmes C_lC_6 et C_mC_8 qui fonctionnent avec les conditions initialles correspondantes à l'indice $n = 1$.

D'un autre côté, si nous supposons que $C_6^{(i,j,n+1)}$, $C_{(8,1)}^{(i,j,n+1)}$ et $C_{(8,2)}^{(i,j,n+1)}$ sont connus $\forall i, j \in N_0$ et n fixé, à partir des relations (2.171) nous pouvons établir un algorithme capable de calculer $C_6^{(i,j,k)}$, $C_{(8,1)}^{(i,j,k)}$, et $C_{(8,2)}^{(i,j,k)}$, $\forall i, j \in N_0$ et $k \leq n$:

- ALGO C_6C_8, version 1, forme progressive.

$C_6^{(i,j,n+1)}$, $C_{(8,1)}^{(i,j,n+1)}$ et $C_{(8,2)}^{(i,j,n+1)}$ sont connus $\forall i, j \in N_0$ et n est fixé

$$C_6^{(i,j,n)} = \frac{C_6^{(i-1,j-1,n+1)} C_{(8,2)}^{(i,j-1,n+1)} - C_6^{(i,j-1,n+1)} C_{(8,2)}^{(i-1,j-1,n+1)} C_{(8,1)}^{(i,j-1,n+1)}}{C_{(8,2)}^{(i-1,j-1,n+1)} C_{(8,2)}^{(i,j-1,n+1)}}$$

$$C_{(8,2)}^{(i,j,n)} = C_6^{(i,j+1,n)} C_{(8,2)}^{(i-1,j,n+1)} / C_6^{(i-1,j,n+1)}$$

$$C_{(8,1)}^{(i,j,n)} = 1 - C_{(8,2)}^{(i,j,n)}$$

Essayons de trouver un algorithme plus simple que celui-ci.

Dénotons les relations (2.171) par (2.171)−1 et (2.171)−2 respectivement.

Remplaçons dans (2.171)−1, $C_6^{(i,j+1,n)} C_{(8,2)}^{(i-1,j,n+1)}$ par le premier membre de (2.171)−2; nous obtenons :

$$C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n+1)} = C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n)} C_{(8,2)}^{(i,j,n+1)} +$$

$$C_6^{(i,j,n+1)} C_{(8,2)}^{(i-1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}.$$

Mettons en évidence $C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n+1)}$

$$C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n+1)} (1 - C_{(8,2)}^{(i,j,n)}) = C_6^{(i,j,n+1)} C_{(8,2)}^{(i-1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}.$$

Remplaçons $(1 - C_{(8,2)}^{(i,j,n)})$ par $C_{(8,1)}^{(i,j,n)}$

$$C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n+1)} C_{(8,1)}^{(i,j,n)} = C_6^{(i,j,n+1)} C_{(8,2)}^{(i-1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}. \tag{2.179}$$

Nous observons que cette relation s'obtient de la relation (2.147)–2 en écrivant les coefficients C_1 et C_2 en fonction de C_6 et C_8.

A partir des relations (2.171)–2 et (2.179), nous déduisons l'algorithme suivant :

- ALGO C_6C_8 version 2, forme progressive.

$C_6^{(i,j,n+1)}$, $C_{(8,1)}^{(i,j,n+1)}$ et $C_{(8,2)}^{(i,j,n+1)}$ sont connus $\forall i,j \in N_0$ et n est fixé

$$C_{(8,1)}^{(i,j,n)} = (C_6^{(i,j,n+1)} C_{(8,2)}^{(i-1,j,n+1)} C_{(8,1)}^{(i,j,n+1)}) / (C_6^{(i-1,j,n+1)} C_{(8,2)}^{(i,j,n+1)})$$

$$C_{(8,2)}^{(i,j,n)} = 1 - C_{(8,1)}^{(i,j,n)}$$

$$C_6^{(i,j,n)} = C_6^{(i-1,j-1,n+1)} C_{(8,2)}^{(i,j-1,n+1)} / C_{(8,2)}^{(i-1,j-1,n+1)}$$

Nous pourrions encore déduire d'autres versions des algorithmes C_1C_2, C_1C_6, C_2C_6, C_1C_8, C_2C_8 et C_6C_8.

Supposons maintenant que nous voulions calculer des coefficients autres que C_1, C_2, C_6 et C_8, par exemple les coefficients C_{12}.

Voyons que ce calcul peut être fait en utilisant les six algorithmes déjà déduits.

Rappelons l'identité (2.123)

$$C_{12}^{(i,j,n)} = -\frac{1}{C_1^{(i,j,n+1)}}.$$

A partir de cette identité et de (2.135), (2.138), et (2.141), nous écrivons les coefficients C_{12} en fonction de C_2 et C_6, de C_2 et C_8, et de C_6 et C_8 respectivement:

$$C_{12}^{(i,j,n)} = \frac{1}{C_6^{(i-1,j,n+1)} - C_2^{(i-1,j,n+1)}}, \tag{2.180}$$

$$C_{12}^{(i,j,n)} = -\frac{C_{(8,1)}^{(i-1,j,n+1)}}{C_2^{(i-1,j,n+1)}}, \tag{2.181}$$

$$C_{12}^{(i,j,n)} = \frac{C_{(8,2)}^{(i-1,j,n+1)}}{C_6^{(i-1,j,n+1)}}. \tag{2.182}$$

Ennonçons maintenant les différentes méthodes que nous pouvons utiliser :
- Commencer par calculer les coefficients C_1 à partir d'un des algorithmes C_1C_2, C_1C_6 ou C_1C_8, et ensuite calculer les coefficients C_{12} à partir de l'égalité (2.123).
- Commencer par calculer les coefficients C_2 et C_6 à partir de l'algorithme C_2C_6, et ensuite calculer les coefficients C_{12} à partir de l'égalité (2.180).
- Commencer par calculer les coefficients C_2 et C_8 à partir de l'algorithme C_2C_8, et ensuite calculer les coefficients C_{12} à partir de l'égalité (2.181).
- Commencer par calculer les coefficients C_6 et C_8 à partir de l'algorithme C_6C_8, et ensuite calculer les coefficients C_{12} à partir de l'égalité (2.182).

Mais, nous pouvons aussi calculer les coefficients C_{12} directement à partir d'un algorithme C_lC_{12} à déduire.

Il est important de remarquer que ces différents procédés ne sont pas numériquement équivalents.

Montrons que nous pouvons toujours écrire les algorithmes C_1C_2, C_1C_6, C_2C_6, C_1C_8, C_2C_8 et C_6C_8 en fonction de C_l et C_m, à condition que C_l et C_m n'appartiennent pas au même groupe de coefficients.

Soit C_iC_j un des six algorithmes déjà déduits. Nous savons que ces six algorithmes correspondent aux combinaisons deux à deux de C_1, C_2, C_6 et C_8, qui sont les représentants des quatre groupes de coefficients. Alors, il n'y a que trois cas à considérer :

— C_i et C_l coïncident ou appartiennent au même ensemble de coefficients, de même que C_j et C_m. Alors C_i s'écrit uniquement en fonction de C_l, et C_j s'écrit uniquement en fonction de C_m.

— C_i et C_l coïncident ou appartiennent au même groupe de coefficients, mais C_j et C_m non. Comme C_i et C_j n'appartiennent pas au même groupe non plus, C_j, C_l et C_m appartiennent tous les trois à des ensembles différents, et alors C_j peut s'écrire en fonction de C_l et C_m. C_i s'écrit uniquement en fonction de C_l.

— C_i et C_j n'appartiennent pas aux mêmes groupes que C_l et C_m. Alors C_i, C_l et C_m, et C_j, C_l et C_m appartiennent à des ensembles différents, donc C_i et C_j peuvent s'écrire en fonction de C_l et C_m.

Nous pouvons essayer de déduire les algorithmes C_lC_m, où C_l et C_m n'appartiennent pas au même ensemble de coefficients, à partir des algorithmes C_iC_j écrits en fonction des coefficients C_l et C_m. Si C_l et C_m appartiennent au même groupe de coefficients, cette méthode ne peut pas être appliquée.

Donnons deux exemples des algorithmes C_lC_m.

L'algorithme $C_{11}C_{12}$ qui suit a été obtenu en écrivant la version 2 de l'algorithme C_1C_2 en fonction de C_{11} et C_{12}, en utilisant les égalités

$$C_1^{(i,j,n)} = -\frac{1}{C_{12}^{(i,j,n-1)}} \text{ et } C_2^{(i,j,n)} = -\frac{1}{C_{11}^{(i+1,j,n-1)}}.$$

- ALGO $C_{11}C_{12}$ version 2.

$$C_{11}^{(i,j,0)} = \frac{L_{i-1}(x^j)}{L_{i-1}(x^{j+1})} \ , \ C_{12}^{(i,j,0)} = \frac{L_i(x^j)}{L_i(x^{j+1})}$$

$$C_{11}^{(i,j,n)} = \frac{C_{12}^{(i,j+1,n-1)} C_{11}^{(i,j+1,n-1)}}{C_{12}^{(i,j,n-1)}} \frac{C_{11}^{(i,j,n-1)} - C_{12}^{(i,j,n-1)}}{C_{11}^{(i,j+1,n-1)} - C_{12}^{(i,j+1,n-1)}}$$

$$C_{12}^{(i,j,n)} = \frac{C_{12}^{(i+1,j,n-1)} C_{11}^{(i+1,j,n)}}{C_{11}^{(i+1,j,n-1)}}$$

L'algorithme C_9C_{12} qui suit a été obtenu en écrivant la version 2 de l'algorithme C_1C_2 en fonction de C_9 et C_{12}, en utilisant les égalités

$$C_1^{(i,j,n)} = -\frac{1}{C_{12}^{(i,j,n-1)}} \text{ et } C_2^{(i,j,n)} = -C_9^{(i+1,j+1,n-1)}.$$

- ALGO C_9C_{12} version 2.

$$C_9^{(i,j,0)} = \frac{L_{i-1}(x^j)}{L_{i-1}(x^{j-1})} \ , \ C_{12}^{(i,j,0)} = \frac{L_i(x^j)}{L_i(x^{j+1})}$$

$$C_9^{(i,j,n)} = \frac{C_9^{(i,j,n-1)} C_{12}^{(i,j-1,n-1)}}{C_{12}^{(i,j,n-1)}} \frac{1 - C_9^{(i,j+1,n-1)} C_{12}^{(i,j,n-1)}}{1 - C_9^{(i,j,n-1)} C_{12}^{(i,j-1,n-1)}}$$

$$C_{12}^{(i,j,n)} = \frac{C_{12}^{(i+1,j,n-1)} C_9^{(i+1,j+1,n-1)}}{C_9^{(i+1,j+1,n)}}$$

2.3.2 Cas particuliers

Nous nous proposons d'écrire les algorithmes déduits dans la section précédente dans les cas particuliers des polynômes orthogonaux adjacents et des polynômes orthogonaux vectoriels de dimension -1.

Nous rappelons que dans le cas des polynômes orthogonaux vectoriels de dimension d, F_1, F_{10} et F_{12} sont les seules relations qui existent, et comme C_1, C_{10} et C_{12} appartiennent au même ensemble de coefficients, nous n'avons pas déduit des algorithmes correspondants à ces polynômes.

1. **Polynômes Orthogonaux Adjacents**

   En sachant que $L_i(x^j) = L(x^{i+j})$ et $C_{(k,l)}^{(i,j,n)} = C_{(k,l)}^{(i+j,n)}$, nous obtenons facilement l'expression particulière des algorithmes déduits dans la section précédente.

   - ALGO C_1C_2 version 1.

$$C_2^{(l,1)} = C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{(l,n+1)} = C_2^{(l,n)} \frac{C_2^{(l+1,n)} - C_1^{(l+2,n)}}{C_2^{(l,n)} - C_1^{(l+1,n)}}$$

$$C_1^{(l,n+1)} = C_1^{(l+2,n)} + C_2^{(l,n+1)} - C_2^{(l+1,n)}$$

- ALGO C_1C_2 version 2.

$$C_2^{(l,1)} = C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{(l,n+1)} = C_2^{(l,n)} \frac{C_2^{(l+1,n)} - C_1^{(l+2,n)}}{C_2^{(l,n)} - C_1^{(l+1,n)}}$$

$$C_1^{(l,n+1)} = C_1^{(l+1,n)} C_2^{(l,n+1)} / C_2^{(l,n)}$$

- ALGO C_1C_2 version 3.

$$C_1^{(l,1)} = C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{(l,n+1)} = C_1^{(l+1,n)} \frac{C_2^{(l+1,n)} - C_1^{(l+2,n)}}{C_2^{(l,n)} - C_1^{(l+1,n)}}$$

$$C_2^{(l,n+1)} = C_1^{(l,n+1)} + C_2^{(l+1,n)} - C_1^{(l+2,n)}$$

- ALGO C_1C_2 version 4.

$$C_1^{(l,1)} = C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{(l,n+1)} = C_1^{(l+1,n)} \frac{C_2^{(l+1,n)} - C_1^{(l+2,n)}}{C_2^{(l,n)} - C_1^{(l+1,n)}}$$

$$C_2^{(l,n+1)} = C_1^{(l,n+1)} C_2^{(l,n)} / C_1^{(l+1,n)}$$

- ALGO C_1C_6.

$$C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}\ ,\ C_6^{(l,1)} = \frac{L(x^{l+2})}{L(x^{l+1})} - \frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{(l,n+1)} = C_1^{(l+1,n)} C_6^{(l+1,n)} / C_6^{(l,n)}$$

$$C_6^{(l,n+1)} = C_1^{(l,n+1)} + C_6^{(l+1,n)} - C_1^{(l+1,n+1)}$$

- ALGO C_2C_6.

$$C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}\ ,\ C_6^{(l,1)} = \frac{L(x^{l+2})}{L(x^{l+1})} - \frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{(l,n+1)} = C_2^{(l,n)} C_6^{(l+1,n)} / C_6^{(l,n)}$$

$$C_6^{(l,n+1)} = C_2^{(l,n+1)} + C_6^{(l+2,n)} - C_2^{(l+1,n+1)}$$

- ALGO C_1C_8 version 1.

$$C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{(l,1)} = \frac{L(x^{l+1})L(x^{l+1})}{L(x^l)L(x^{l+2})}\ ,\ C_{(8,2)}^{(l,1)} = 1 - C_{(8,1)}^{(l,1)}$$

$$C_1^{(l,n+1)} = C_1^{(l+2,n)} C_{(8,2)}^{(l+1,n)} / C_{(8,2)}^{(l,n)}$$

$$C_{(8,1)}^{(l,n+1)} = (C_1^{(l,n+1)} - C_1^{(l+2,n)} C_{(8,2)}^{(l+1,n)}) / C_1^{(l+1,n+1)}$$

$$C_{(8,2)}^{(l,n+1)} = 1 - C_{(8,1)}^{(l,n+1)}$$

- ALGO C_1C_8 version 2.

$$C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{(l,1)} = \frac{L(x^{l+1})L(x^{l+1})}{L(x^l)L(x^{l+2})}\ ,\ C_{(8,2)}^{(l,1)} = 1 - C_{(8,1)}^{(l,1)}$$

$$C_1^{(l,n+1)} = C_1^{(l+2,n)} C_{(8,2)}^{(l+1,n)} / C_{(8,2)}^{(l,n)}$$

$$C_{(8,1)}^{(l,n+1)} = C_1^{(l,n+1)} C_{(8,1)}^{(l,n)} / C_1^{(l+1,n+1)}$$

$$C_{(8,2)}^{(l,n+1)} = 1 - C_{(8,1)}^{(l,n+1)}$$

- ALGO C_2C_8 version 1.

$$C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{(l,1)} = \frac{L(x^{l+1})L(x^{l+1})}{L(x^l)L(x^{l+2})}\ ,\ C_{(8,2)}^{(l,1)} = 1 - C_{(8,1)}^{(l,1)}$$

$$C_2^{(l,n+1)} = (C_2^{(l+1,n)} C_{(8,2)}^{(l+1,n)} C_{(8,1)}^{(l,n)}) / (C_{(8,1)}^{(l+1,n)} C_{(8,2)}^{(l,n)})$$

$$C_{(8,1)}^{(l,n+1)} = \frac{C_2^{(l,n+1)} C_{(8,1)}^{(l+2,n)}}{C_2^{(l+2,n)} C_{(8,2)}^{(l+2,n)} + C_2^{(l+1,n+1)} C_{(8,1)}^{(l+2,n)}}$$

$$C_{(8,2)}^{(l,n+1)} = 1 - C_{(8,1)}^{(l,n+1)}$$

- ALGO C_2C_8 version 2.

$$C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{(l,1)} = \frac{L(x^{l+1})L(x^{l+1})}{L(x^l)L(x^{l+2})}\ ,\ C_{(8,2)}^{(l,1)} = 1 - C_{(8,1)}^{(l,1)}$$

$$C_2^{(l,n+1)} = (C_2^{(l+1,n)} C_{(8,2)}^{(l+1,n)} C_{(8,1)}^{(l,n)}) / (C_{(8,1)}^{(l+1,n)} C_{(8,2)}^{(l,n)})$$

$$C_{(8,1)}^{(l,n+1)} = C_2^{(l,n+1)} C_{(8,1)}^{(l+1,n)} / C_2^{(l+1,n+1)}$$

$$C_{(8,2)}^{(l,n+1)} = 1 - C_{(8,1)}^{(l,n+1)}$$

- ALGO C_2C_8 version 3.

$$C_2^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{(l,1)} = \frac{L(x^{l+1})L(x^{l+1})}{L(x^l)L(x^{l+2})}\ ,\ C_{(8,2)}^{(l,1)} = 1 - C_{(8,1)}^{(l,1)}$$

$$C_2^{(l,n+1)} = (C_2^{(l+1,n)} C_{(8,2)}^{(l+1,n)} C_{(8,1)}^{(l,n)})/(C_{(8,1)}^{(l+1,n)} C_{(8,2)}^{(l,n)})$$

$$C_{(8,1)}^{(l,n+1)} = (C_2^{(l,n+1)} C_{(8,1)}^{(l+2,n)} C_{(8,2)}^{(l+1,n)})/(C_2^{(l+2,n)} C_{(8,2)}^{(l+2,n)})$$

$$C_{(8,2)}^{(l,n+1)} = 1 - C_{(8,1)}^{(l,n+1)}$$

- ALGO C_6C_8, version 1, forme progressive.

$C_6^{(l,n+1)}$, $C_{(8,1)}^{(l,n+1)}$ et $C_{(8,2)}^{(l,n+1)}$ sont connus $\forall l \in N_0$ et n est fixé

$$C_6^{(l,n)} = \frac{C_6^{(l-2,n+1)} C_{(8,2)}^{(l-1,n+1)} - C_6^{(l-1,n+1)} C_{(8,2)}^{(l-2,n+1)} C_{(8,1)}^{(l-1,n+1)}}{C_{(8,2)}^{(l-2,n+1)} C_{(8,2)}^{(l-1,n+1)}}$$

$$C_{(8,2)}^{(l,n)} = C_6^{(l+1,n)} C_{(8,2)}^{(l-1,n+1)} / C_6^{(l-1,n+1)}$$

$$C_{(8,1)}^{(l,n)} = 1 - C_{(8,2)}^{(l,n)}$$

- ALGO C_6C_8 version 2, forme progressive.

$C_6^{(l,n+1)}$, $C_{(8,1)}^{(l,n+1)}$ et $C_{(8,2)}^{(l,n+1)}$ sont connus $\forall l \in N_0$ et n est fixé

$$C_{(8,1)}^{(l,n)} = (C_6^{(l,n+1)} C_{(8,2)}^{(l-1,n+1)} C_{(8,1)}^{(l,n+1)})/(C_6^{(l-1,n+1)} C_{(8,2)}^{(l,n+1)})$$

$$C_{(8,2)}^{(l,n)} = 1 - C_{(8,1)}^{(l,n)}$$

$$C_6^{(l,n)} = C_6^{(l-2,n+1)} C_{(8,2)}^{(l-1,n+1)} / C_{(8,2)}^{(l-2,n+1)}$$

- ALGO $C_{11}C_{12}$ version 2.

$$C_{11}^{(l,0)} = \frac{L(x^{l-1})}{L(x^l)} \ , \ C_{12}^{(l,0)} = \frac{L(x^l)}{L(x^{l+1})}$$

$$C_{11}^{(l,n)} = \frac{C_{12}^{(l+1,n-1)} C_{11}^{(l+1,n-1)}}{C_{12}^{(l,n-1)}} \frac{C_{11}^{(l,n-1)} - C_{12}^{(l,n-1)}}{C_{11}^{(l+1,n-1)} - C_{12}^{(l+1,n-1)}}$$

$$C_{12}^{(l,n)} = \frac{C_{12}^{(l+1,n-1)} C_{11}^{(l+1,n)}}{C_{11}^{(l+1,n-1)}}$$

- ALGO C_9C_{12} version 2.

$$C_9^{(l,0)} = \frac{L(x^{l-1})}{L(x^{l-2})} \ , \ C_{12}^{(l,0)} = \frac{L(x^l)}{L(x^{l+1})}$$

$$C_9^{(l,n)} = \frac{C_9^{(l,n-1)} C_{12}^{(l-1,n-1)}}{C_{12}^{(l,n-1)}} \frac{1 - C_9^{(l+1,n-1)} C_{12}^{(l,n-1)}}{1 - C_9^{(l,n-1)} C_{12}^{(l-1,n-1)}}$$

$$C_{12}^{(l,n)} = \frac{C_{12}^{(l+1,n-1)} C_9^{(l+2,n-1)}}{C_9^{(l+2,n)}}$$

Le fameux QD algorithme des polynômes orthogonaux adjacents est l'algorithme C_1C_5.

Nous pouvons déduire C_1C_5 à partir par exemple de l'algorithme C_1C_6, en sachant que

$$C_6^{(l,n)} = -C_5^{(l+1,n)},$$

puisque $C_6^{(i,j,n)} = -C_5^{(i+1,j,n)}$.

Nous obtenons :

- ALGO C_1C_5.

$$C_1^{(l,1)} = -\frac{L(x^{l+1})}{L(x^l)} \ , \ C_5^{(l,1)} = \frac{L(x^l)}{L(x^{l-1})} - \frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{(l,n+1)} = C_1^{(l+1,n)} C_5^{(l+2,n)} / C_5^{(l+1,n)}$$

$$C_5^{(l,n+1)} = C_1^{(l,n+1)} + C_5^{(l+1,n)} - C_1^{(l-1,n+1)}$$

En faisant

$$C_1^{(l,n)} = -q_l^{(n)} \text{ et } C_5^{(l,n)} = -e_{l-1}^{(n)},$$

les relations de l'algorithme C_1C_5 s'écrivent de la façon suivante :

$$q_{n+1}^{(l)} e_n^{(l)} = q_n^{(l+1)} e_n^{(l-1)}$$

$$e_{n+1}^{(l)} + q_{n+1}^{(l)} = q_{n+1}^{(l)} + e_n^{(l+1)},$$

qui est la notation habituelle du QD algorithme [2].

2. Polynômes Orthogonaux Vectoriels de dimension -1

En sachant que $L_i(x^j) = L(x^{j-i})$ et $C_{(k,l)}^{(i,j,n)} = C_{(k,l)}^{[j-i,n]}$, nous obtenons facilement l'expression particulière des algorithmes déduits dans la section précédente.

- ALGO C_1C_2 version 1.

$$C_2^{[l,1]} = C_1^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{[l,n+1]} = C_2^{[l,n]} \frac{C_2^{[l+1,n]} - C_1^{[l,n]}}{C_2^{[l,n]} - C_1^{[l-1,n]}}$$

$$C_1^{[l,n+1]} = C_1^{[l,n]} + C_2^{[l,n+1]} - C_2^{[l+1,n]}$$

- ALGO C_1C_2 version 2.

$$C_2^{[l,1]} = C_1^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{[l,n+1]} = C_2^{[l,n]} \frac{C_2^{[l+1,n]} - C_1^{[l,n]}}{C_2^{[l,n]} - C_1^{[l-1,n]}}$$

$$C_1^{[l,n+1]} = C_1^{[l-1,n]} C_2^{[l,n+1]} / C_2^{[l,n]}$$

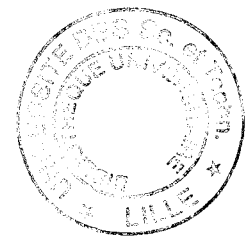
- ALGO C_1C_2 version 3.

$$C_1^{[l,1]} = C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{[l,n+1]} = C_1^{[l-1,n]} \frac{C_2^{[l+1,n]} - C_1^{[l,n]}}{C_2^{[l,n]} - C_1^{[l-1,n]}}$$

$$C_2^{[l,n+1]} = C_1^{[l,n+1]} + C_2^{[l+1,n]} - C_1^{[l,n]}$$

- ALGO C_1C_2 version 4.



$$C_1^{[l,1]} = C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{[l,n+1]} = C_1^{[l-1,n]} \frac{C_2^{[l+1,n]} - C_1^{[l,n]}}{C_2^{[l,n]} - C_1^{[l-1,n]}}$$

$$C_2^{[l,n+1]} = C_1^{[l,n+1]} C_2^{[l,n]} / C_1^{[l-1,n]}$$

- ALGO C_1C_6.

$$C_1^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}\ ,\ C_6^{[l,1]} = \frac{L(x^l)}{L(x^{l-1})} - \frac{L(x^{l+1})}{L(x^l)}$$

$$C_1^{[l,n+1]} = C_1^{[l-1,n]} C_6^{[l+1,n]} / C_6^{[l,n]}$$

$$C_6^{[l,n+1]} = C_1^{[l,n+1]} + C_6^{[l+1,n]} - C_1^{[l-1,n+1]}$$

- ALGO C_2C_6.

$$C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}\ ,\ C_6^{[l,1]} = \frac{L(x^l)}{L(x^{l-1})} - \frac{L(x^{l+1})}{L(x^l)}$$

$$C_2^{[l,n+1]} = C_2^{[l,n]} C_6^{[l+1,n]} / C_6^{[l,n]}$$

$$C_6^{[l,n+1]} = C_{2.}^{[l,n+1]} + C_6^{[l,n]} - C_2^{[l-1,n+1]}$$

- ALGO C_1C_8 version 1.

$$C_1^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{[l,1]} = \frac{L(x^{l+1})L(x^{l-1})}{L(x^l)L(x^l)}\ ,\ C_{(8,2)}^{[l,1]} = 1 - C_{(8,1)}^{[l,1]}$$

$$C_1^{[l,n+1]} = C_1^{[l,n]} C_{(8,2)}^{[l+1,n]} / C_{(8,2)}^{[l,n]}$$

$$C_{(8,1)}^{[l,n+1]} = (C_1^{[l,n+1]} - C_1^{[l,n]} C_{(8,2)}^{[l+1,n]}) / C_1^{[l-1,n+1]}$$

$$C_{(8,2)}^{[l,n+1]} = 1 - C_{(8,1)}^{[l,n+1]}$$

- ALGO C_1C_8 version 2.

$$C_1^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{[l,1]} = \frac{L(x^{l+1})L(x^{l-1})}{L(x^l)L(x^l)}\ ,\ C_{(8,2)}^{[l,1]} = 1 - C_{(8,1)}^{[l,1]}$$

$$C_1^{[l,n+1]} = C_1^{[l,n]} C_{(8,2)}^{[l+1,n]} / C_{(8,2)}^{[l,n]}$$

$$C_{(8,1)}^{[l,n+1]} = C_1^{[l,n+1]} C_{(8,1)}^{[l,n]} / C_1^{[l-1,n+1]}$$

$$C_{(8,2)}^{[l,n+1]} = 1 - C_{(8,1)}^{[l,n+1]}$$

- ALGO C_2C_8 version 1.

$$C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{[l,1]} = \frac{L(x^{l+1})L(x^{l-1})}{L(x^l)L(x^l)}\ ,\ C_{(8,2)}^{[l,1]} = 1 - C_{(8,1)}^{[l,1]}$$

$$C_2^{[l,n+1]} = (C_2^{[l+1,n]} C_{(8,2)}^{[l+1,n]} C_{(8,1)}^{[l,n]}) / (C_{(8,1)}^{[l+1,n]} C_{(8,2)}^{[l,n]})$$

$$C_{(8,1)}^{[l,n+1]} = \frac{C_2^{[l,n+1]} C_{(8,1)}^{[l,n]}}{C_2^{[l,n]} C_{(8,2)}^{[l,n]} + C_2^{[l-1,n+1]} C_{(8,1)}^{[l,n]}}$$

$$C_{(8,2)}^{[l,n+1]} = 1 - C_{(8,1)}^{[l,n+1]}$$

- ALGO C_2C_8 version 2.

$$C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{[l,1]} = \frac{L(x^{l+1})L(x^{l-1})}{L(x^l)L(x^l)}\ ,\ C_{(8,2)}^{[l,1]} = 1 - C_{(8,1)}^{[l,1]}$$

$$C_2^{[l,n+1]} = (C_2^{[l+1,n]} C_{(8,2)}^{[l+1,n]} C_{(8,1)}^{[l,n]}) / (C_{(8,1)}^{[l+1,n]} C_{(8,2)}^{[l,n]})$$

$$C_{(8,1)}^{[l,n+1]} = C_2^{[l,n+1]} C_{(8,1)}^{[l-1,n]} / C_2^{[l-1,n+1]}$$

$$C_{(8,2)}^{[l,n+1]} = 1 - C_{(8,1)}^{[l,n+1]}$$

- ALGO C_2C_8 version 3.

$$C_2^{[l,1]} = -\frac{L(x^{l+1})}{L(x^l)}$$

$$C_{(8,1)}^{[l,1]} = \frac{L(x^{l+1})L(x^{l-1})}{L(x^l)L(x^l)}\ ,\ C_{(8,2)}^{[l,1]} = 1 - C_{(8,1)}^{[l,1]}$$

$$C_2^{[l,n+1]} = (C_2^{[l+1,n]} C_{(8,2)}^{[l+1,n]} C_{(8,1)}^{[l,n]})/(C_{(8,1)}^{[l+1,n]} C_{(8,2)}^{[l,n]})$$

$$C_{(8,1)}^{[l,n+1]} = (C_2^{[l,n+1]} C_{(8,1)}^{[l,n]} C_{(8,2)}^{[l-1,n]})/(C_2^{[l,n]} C_{(8,2)}^{[l,n]})$$

$$C_{(8,2)}^{[l,n+1]} = 1 - C_{(8,1)}^{[l,n+1]}$$

- ALGO C_6C_8, version 1, forme progressive.

$C_6^{[l,n+1]}$, $C_{(8,1)}^{[l,n+1]}$ et $C_{(8,2)}^{[l,n+1]}$ sont connus $\forall l \in Z$ et n est fixé

$$C_6^{[l,n]} = \frac{C_6^{[l,n+1]} C_{(8,2)}^{[l-1,n+1]} - C_6^{[l-1,n+1]} C_{(8,2)}^{[l,n+1]} C_{(8,1)}^{[l-1,n+1]}}{C_{(8,2)}^{[l,n+1]} C_{(8,2)}^{[l-1,n+1]}}$$

$$C_{(8,2)}^{[l,n]} = C_6^{[l+1,n]} C_{(8,2)}^{[l+1,n+1]} / C_6^{[l+1,n+1]}$$

$$C_{(8,1)}^{[l,n]} = 1 - C_{(8,2)}^{[l,n]}$$

- ALGO C_6C_8 version 2, forme progressive.

$C_6^{[l,n+1]}$, $C_{(8,1)}^{[l,n+1]}$ et $C_{(8,2)}^{[l,n+1]}$ sont connus $\forall l \in Z$ et n est fixé

$$C_6^{[l,n]} = (C_6^{[l,n+1]} C_{(8,2)}^{[l+1,n+1]} C_{(8,1)}^{[l,n+1]})/(C_6^{[l+1,n+1]} C_{(8,2)}^{[l,n+1]})$$

$$C_{(8,2)}^{[l,n]} = C_6^{[l+1,n]} C_{(8,2)}^{[l+1,n+1]} / C_6^{[l+1,n+1]}$$

$$C_{(8,1)}^{[l,n]} = 1 - C_{(8,2)}^{[l,n]}$$

- ALGO $C_{11}C_{12}$ version 2.

$$C_{11}^{[l,0]} = \frac{L(x^{l+1})}{L(x^{l+2})} \ , \ C_{12}^{[l,0]} = \frac{L(x^l)}{L(x^{l+1})}$$

$$C_{11}^{[l,n]} = \frac{C_{12}^{[l+1,n-1]} C_{11}^{[l+1,n-1]}}{C_{12}^{[l,n-1]}} \frac{C_{11}^{[l,n-1]} - C_{12}^{[l,n-1]}}{C_{11}^{[l+1,n-1]} - C_{12}^{[l+1,n-1]}}$$

$$C_{12}^{[l,n]} = \frac{C_{12}^{[l-1,n-1]} C_{11}^{[l-1,n]}}{C_{11}^{[l-1,n-1]}}$$

- ALGO C_9C_{12} version 2.

$$C_9^{[l,0]} = \frac{L(x^{l+1})}{L(x^l)} \ , \ C_{12}^{[l,0]} = \frac{L(x^l)}{L(x^{l+1})}$$

$$C_9^{[l,n]} = \frac{C_9^{[l,n-1]} C_{12}^{[l-1,n-1]}}{C_{12}^{[l,n-1]}} \frac{1 - C_9^{[l+1,n-1]} C_{12}^{[l,n-1]}}{1 - C_9^{[l,n-1]} C_{12}^{[l-1,n-1]}}$$

$$C_{12}^{[l,n]} = \frac{C_{12}^{[l-1,n-1]} C_9^{[l,n-1]}}{C_9^{[l,n]}}$$

2.3.3 Conclusion

L'implémentation des algorithmes de type QD pour le calcul des coefficients des relations de récurrence des polynômes biorthogonaux pose de sérieux problèmes d'encombrement de mémoire et de volume de calculs. Une étude de minimisation de ces problèmes, semblable à celle qui a été faite dans la section 1.3.1, s'impose.

En outre, il se pose la question de la stabilité numérique de ces algorithmes. En effet, il est connu que l'algorithme QD pour les polynômes orthogonaux adjacents est numériquement instable à cause du calcul des différences et des quotients entre des valeurs très proches. Nous remarquons que la même situation est susceptible de se produire pour ce qui concerne les autres algorithmes de type QD.

L'implémentation en *Mathematica*[13] permet de faire du calcul approché de haute précision et permet ainsi de minimiser l'effet cumulatif des erreurs d'arrondi.

Cependant, il serait toujours intéressant d'étudier les différentes versions des algorithmes du point de vue de la stabilité numérique.

GENPADETYPE et modules

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C    PROGRAM NAME  - GENPADETYPE ( GENERALIZED PADE-TYPE APPROXIMANTS )        +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C  DESCRITPION :                                                               +
C                                                                              +
C       MAIN PROGRAM THAT COMPUTES A SEQUENCE OF GENERALIZED PADE-TYPE         +
C       APPROXIMANTS USING THE BLOCK BORDERING METHOD.                         +
C       IT IS A CONVERSATIONAL PROGRAM. DURING RUNTIME THE USER IS             +
C       INVITED TO CHOSE THE SERIES, THE GENERATING FUNCTION, THE BASIS        +
C       OF POLYNOMIALS AND THE FUNCTIONALS AMONG OTHERS, OF THE EXAMPLE        +
C       HE WANTS TO TEST.                                                      +
C                                                                              +
C                                                                              +
C  MODULES REQUIRED :                                                          +
C                                                                              +
C                                                                              +
C                                                                              +
C   - SUBROUTINES      BASISPOLY, BETABOR, BLBORD, DERIVEPOLY,                 +
C        FUNGENERATING, GAUSS, INVERS, MODIFIEDMOMENTS,                        +
C        PRIMITIVEPOLY, READPOLY, RCPROD, SYSTEM,                              +
C        WRITEPOLY, WRITING.                                                   +
C                                                                              +
C                                                                              +
C    - INTEGER FUNCTIONS  PFACT.                                               +
C                                                                              +
C                                                                              +
C    - DOUBLE PRECISION FUNCTION   ELFUNGEN, F, FUNPOLY, GPADETYPE,            +
C                                  HORNER, INNERG.                             +
C                                                                              +
C                                                                              +
C      THE CODES OF THE SUBROUTINES  BASISPOLY, FUNGENERATING AND              +
C      MODIFIEDMOMENTS, AND THE FUNCTIONS  ELFUNGEN, F AND FUNPOLY             +
C      ARE WRITTEN ACCORDING TO THE OPTIONS OF THE SERIES, THE                 +
C      GENERATING FUNCTIONS, THE BASIS OF POLYNOMIALS AND THE                  +
C      FUNCTIONALS GIVEN IN THE CALLING PROGRAM.                               +
C                                                                              +
C                                                                              +
C     WHEN THE USER WANTS TO TEST NEW EXAMPLES, HE MUST ADD                    +
C     THE NECESSARY NEW OPTIONS TO THE                                         +
C                                                                              +
C     - MENU OF SERIES                                                         +
C     - MENU OF GENERATING FUNCTION                                            +
C     - MENU OF BASIS OF POLYNOMIALS                                           +
C     - MENU OF FUNCTIONALS                                                    +
C                                                                              +
C     IN THE CALLING PROGRAM, AND THE CORRESPONDING NEW CODE TO THE            +
C                                                                              +
C     SUBROUTINES      - BASISPOLY                                             +
C                      - FUNGENERATING                                         +
C                      - MODIFIEDMOMENTS                                       +
C                                                                              +
C     AND THE                                                                  +
C                                                                              +
C     FUNCTIONS        - ELFUNGEN                                              +
C                      - F                                                     +
C                      - FUNPOLY .                                             +
C                                                                              +
C                                                                              +
C     THE BLOCK BORDERING METHOD IS IMPLEMENTED IN THE MODULES BLBORD,         +
C     BETABOR, GAUSS, INVERS, RCPROD, AND INNERG, THAT WERE WRITTEN            +
C     AND PUBLISHED BY                                                         +
C                                                                              +
C           C. BREZINSKI, M. REDIVO ZAGLIA, Extrapolation Methods.             +
C           Theory and Practice, North-Holland, Amsterdam, 1991.               +
C                                                                              +
```

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C                              Zelia DA ROCHA                                  +
C            Laboratoire d'Analyse Numerique et d'Optimisation                 +
C                              UFR IEEA - M3                                   +
C                                   USTL                                       +
C                        59655 - Villeneuve d'Ascq Cedex                       +
C                                   FRANCE                                     +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C                                                                              +
      PROGRAM GENPADETYPE
C
C        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      DOUBLE PRECISION  F, GPADETYPE
C
C          ... SPECIFICATIONS FOR VARIABLES AND CONSTANTS
C
      INTEGER IER, INIT, IR, IL, I, I1, I2, IL1, N, K, J, OPS,
     *        OPD, OPG, OPB, OPF, OPAI, OPC1, OPC2
      DOUBLE PRECISION EPS, GPT, FT, T
C
C          ... SPECIFICATIONS FOR PARAMETER CONSTANTS
C
      PARAMETER (IR=40 , IL=20 , EPS=1.0D-31)
C
C       IR IS THE MAXIMUM NUMBER OF APPROXIMANTS TO BE CALCULATED
C
C
C          ... SPECIFICATIONS FOR VECTORS AND MATRICES
C
      DOUBLE PRECISION M(IR,IR),A(IR),D(IR),PB(0:IR-1,-1:IR-1),
     *                 L(0:IR-1,0:IL),FKT(0:IR-1),C(0:IR-1),
     *                 WBETA(IR,2*IR),WK(IR),WV1(-1:IR-1),WV2(-1:IR-1)
C
C
C
      OPEN (1, FILE="RESGENPADETYPE", STATUS="NEW")
C
C           ... EXECUTABLE STATEMENTS
C
100   WRITE(9,*)'*** MENU OF SERIES***'
      WRITE(9,*)
      WRITE(9,*)'(1) F(T)=EXP(-T)'
      WRITE(9,*)'(2) F(T)=LOG(1+T)/T'
      WRITE(9,*)'(4) SUM OF SERIES NOT KNOWN'
      READ(9,*)OPS
      SELECT CASE(OPS)
       CASE(1)
         WRITE(1,*)'F(T)=EXP(-T)'
       CASE(2)
         WRITE(1,*)'F(T)=LOG(1+T)/T'
       CASE(4)
         WRITE(1,*)'SUM OF SERIES NOT KNOWN'
       CASE DEFAULT
          WRITE(9,*)'WRONG VALUE'
          WRITE(9,*)'TRY AGAIN'
          GO TO 100
      END SELECT
      WRITE(1,*)
C
C
C
200   WRITE(9,*)'(1) SUM OF SERIES AVAILABLE'
      WRITE(9,*)'(2) SUM OF SERIES NOT AVAILABLE'
      READ(9,*)OPD
```

```
      IF(OPD.NE.1.AND.OPD.NE.2)THEN
        WRITE(9,*)'WRONG VALUE'
        WRITE(9,*)'TRY AGAIN'
        GO TO 200
      END IF
C
C
C
300   WRITE(9,*)'*** MENU OF GENERATING FUNCTIONS ***'
      WRITE(9,*)'                                    '
      WRITE(9,*)'(1) G(X,T)=1/(1-XT)'
      WRITE(9,*)'(2) G(X,T)=EXP(-XT)'
      WRITE(9,*)'(3) G(X,T)=LOG(1+XT)/T'
      READ(9,*)OPG
      SELECT CASE(OPG)
       CASE(1)
         WRITE(1,*)'G(X,T)=1/(1-XT)'
       CASE(2)
         WRITE(1,*)'G(X,T)=EXP(-XT)'
       CASE(3)
         WRITE(1,*)'G(X,T)=LOG(1+XT)/T'
       CASE DEFAULT
         WRITE(9,*)'WRONG VALUE'
         WRITE(9,*)'TRY AGAIN'
         GO TO 300
      END SELECT
      WRITE(1,*)
C
C
C
400   WRITE(9,*)'*** MENU OF BASIS OF POLYNOMIALS ***'
      WRITE(9,*)
      WRITE(9,*)'(1) CANONICAL BASIS'
      WRITE(9,*)'(2) FIRST KIND TCHEBYSHEV BASIS'
      WRITE(9,*)'(3) SECOND KIND TCHEBYSHEV BASIS'
      WRITE(9,*)'(4) BASIS TO GIVE BY THE USER'
      READ(9,*)OPB
      SELECT CASE(OPB)
       CASE(1)
         WRITE(1,*)'CANONICAL BASIS'
       CASE(2)
         WRITE(1,*)'FIRST KIND TCHEBYSHEV BASIS'
       CASE(3)
         WRITE(1,*)'SECOND KIND TCHEBYSHEV BASIS'
       CASE(4)
         WRITE(1,*)'BASIS TO GIVE BY THE USER'
       CASE DEFAULT
         WRITE(9,*)'WRONG VALUE'
         WRITE(9,*)'TRY AGAIN'
         GO TO 400
      END SELECT
      WRITE(1,*)
C
C
C
500   WRITE(9,*)'POINT WHERE THE APPROXIMANTS ARE CALCULATED ?'
      READ(9,*)T
      WRITE(1,15)T
15    FORMAT(/,'T =',D10.2,/)
      SELECT CASE(OPD)
       CASE(1)
         FT=F(OPS,T)
  WRITE(9,25)T,FT
         WRITE(1,25)T,FT
25       FORMAT('F(',D10.2,')=',D24.16)
       CASE(2)
         FT=0.0D00
      END SELECT
```

```
      WRITE(1,35)
C
C
C
600   K=0;INIT=0;IER=0
C
C
C        ... COMPUTATION OF THE K-TH GENERALIZED PADE-TYPE APPROXIMANT
C            (K) (T).
C               F
C
C        ... CHOICE OF THE FUNCTIONAL Lk
C
700   WRITE(9,*)'DEFINITION OF THE FUNCTIONAL L ',K
      WRITE(9,*)
      WRITE(1,*)'DEFINITION OF THE FUNCTIONAL L ',K
      WRITE(1,*)
      WRITE(9,*)'*** MENU OF FUNCTIONALS ***'
      WRITE(9,*)
      WRITE(9,*)'(1) L (F)=F(X )'
      WRITE(9,*)'      k       k  '
      WRITE(9,*)'(2) L (F)=D(J )F(X )'
      WRITE(9,*)'      k       k    k '
      WRITE(9,*)'(3) L (F)=INTEGRAL OF F BETWEEN X  AND Y  '
      WRITE(9,*)'      k                                k      k '
      WRITE(9,*)'(4) L (F)=A   *F(X  )+...+A  *F(X  )'
      WRITE(9,*)'      k     k1     k1       kn    kn '
      WRITE(9,*)'(5) L (F)=A  *D(J  )F(X  )+...+A  *D(J  )F(X  )'
      WRITE(9,*)'      k     k1    k1    k1       kn    kn    kn '
      WRITE(9,*)'(6) L (F)=A  *INTF(X   ,Y  )+...+A  *INTF(X  , Y  )'
      WRITE(9,*)'      k     k1       k1   k1       kn       kn   kn '
      READ(9,*)OPF
      SELECT CASE(OPF)
       CASE(1)
        WRITE(1,*)'L (F)=F(X )'
 WRITE(1,*)' k       k  '
       CASE(2)
        WRITE(1,*)'L (F)=D(J )F(X )'
 WRITE(1,*)' k       k    k  '
       CASE(3)
        WRITE(1,*)'L (F)=INTEGRAL OF F BETWEEN X  AND Y '
 WRITE(1,*)' k                              k      k '
       CASE(4)
        WRITE(1,*)'L (F)=A  *F(X  )+...+A  *F(X  )'
 WRITE(1,*)' k     k1    k1       kn    kn '
       CASE(5)
        WRITE(1,*)'L (F)=A  *D(J  )F(X  )+...+A  *D(J  )F(X  )'
 WRITE(1,*)' k     k1    k1    k1       kn    kn    kn'
       CASE(6)
        WRITE(1,*)'L (F)=A  *INTF(X  ,Y  )+...+A  *INTF(X  ,Y  )'
 WRITE(1,*)' k     k1       k1  k1       kn       kn  kn  '
       CASE DEFAULT
        WRITE(9,*)'WRONG VALUE'
 WRITE(9,*)'TRY AGAIN'
        GO TO 700
      END SELECT
C
C           ... DEFINITION OF THE FUNCTIONAL Lk
C
      L(K,0)=DBLE(OPF)
      IF(OPF.EQ.1.OR.OPF.EQ.2.OR.OPF.EQ.3)THEN
        WRITE(9,*)'X',K,'?'
 WRITE(1,*)'X',K,'?'
        READ(9,*)L(K,1)
 WRITE(1,45)L(K,1)
      ELSE
       SELECT CASE(OPF)
        CASE(4)
```

```
  IL1=INT((IL-1)/2)
 CASE(5,6)
  IL1=INT((IL-1)/3)
       END SELECT
800    WRITE(9,*)'NUMBER OF TERMS <= ',IL1,'?'
       WRITE(1,*)'NUMBER OF TERMS <= ',IL1,'?'
       READ(9,*)N
       IF(N.GT.IL1)THEN
  WRITE(9,*)'THE NUMBER OF TERMS MUST BE SMALLER THAN ',IL1
  WRITE(9,*)'TRY AGAIN'
  WRITE(1,*)'THE NUMBER OF TERMS MUST BE SMALLER THAN ',IL1
  WRITE(1,*)'TRY AGAIN'
  GO TO 800
       END IF
       WRITE(1,55)N
       L(K,1)=DBLE(N)
900    WRITE(9,*)'Ai=1/(NUMBER OF TERMS)'
       WRITE(9,*)'(1) YES  /  (2) NO'
       WRITE(1,*)'Ai=1/(NUMBER OF TERMS)'
       WRITE(1,*)'(1) YES  /  (2) NO'
       READ(9,*)OPAI
       WRITE(1,*)OPAI
       SELECT CASE(OPAI)
  CASE(1)
          DO 10 I=1,N
             L(K,I+1)=1.00D00/N
10         CONTINUE
          CASE(2)
            DO 20 I=1,N
              WRITE(9,*)'A',I,'='
       WRITE(1,*)'A',I,'='
       READ(9,*)L(K,I+1)
        WRITE(1,45)L(K,I+1)
20          CONTINUE
          CASE DEFAULT
     WRITE(9,*)'WRONG VALUE'
     WRITE(9,*)'TRY AGAIN'
     GO TO 900
       END SELECT
       DO 30 I=1,N
         I1=N+1+I
         WRITE(9,*)'X',I,'='
         WRITE(1,*)'X',I,'='
         READ(9,*)L(K,I1)
         WRITE(1,45)L(K,I1)
30     CONTINUE
      END IF
      SELECT CASE(OPF)
      CASE(1)
       CONTINUE
      CASE(2)
       WRITE(9,*)'ORDER OF THE DERIVATIVE ?'
       WRITE(1,*)'ORDER OF THE DERIVATIVE ?'
       READ(9,*)J
       WRITE(1,55)J
       L(K,2)=DBLE(J)
      CASE(3)
       WRITE(9,*)'Y',K,'='
       WRITE(1,*)'Y',K,'='
       READ(9,*)L(K,2)
       WRITE(1,45)L(K,2)
      CASE(4)
       CONTINUE
      CASE (5)
       DO 40 I=1,N
         I2=2*N+1+I
         WRITE(9,*)'J',I,'='
         WRITE(1,*)'J',I,'='
```

```
         READ(9,*)J
  L(K,I2)=DBLE(J)
         WRITE(1,55)J
40      CONTINUE
       CASE(6)
        DO 50 I=1,N
           I2=2*N+1+I
           WRITE(9,*)'Y',I,'='
           WRITE(1,*)'Y',I,'='
           READ(9,*)L(K,I2)
           WRITE(1,45)L(K,I2)
50      CONTINUE
       END SELECT
C
C            ... END OF THE DEFINITION OF THE FUNCTIONAL Lk
C
C
C          ... CALL FUNGENERATING
C
       CALL FUNGENERATING (OPG, L, IR, T, K, FKT)
C
C          ... CALL BASISPOLY
C
       CALL BASISPOLY (OPB, IR, WV1, K, PB)
C
C          ... CALL MODIFIEDMOMENTS
C
       CALL MODIFIEDMOMENTS (OPS, OPG, K, C)
C
C          ... CALL SYSTEM
C
       CALL SYSTEM (L, IR, WV1, WV2, PB, C, K, M, D)
C
C          ... CALL BLBORD
C
       CALL BLBORD (M, D, IR, A, WBETA, WK, EPS, INIT, IER)
       IF(IER.NE.0.AND.IER.NE.2100)THEN
          WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE BLBORD'
  WRITE(9,*)'     '
          WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE BLBORD'
  WRITE(1,*)'     '
          SELECT CASE(IER)
            CASE(200)
         WRITE(9,*)'CALL OF THE SUBROUTINE BLBORD WITH IER<>0'
         WRITE(1,*)'CALL OF THE SUBROUTINE BLBORD WITH IER<>0'
            CASE(300)
         WRITE(9,*)'ROWS DIMENSION TOO SMALL COMPARED TO '
         WRITE(9,*)'NUMBER OF CALLS OF SUBROUTINE BLBORDER'
         WRITE(9,*)'IR < K+1'
         WRITE(1,*)'ROWS DIMENSION TOO SMALL COMPARED TO '
         WRITE(1,*)'NUMBER OF CALLS OF SUBROUTINE BLBORDER'
         WRITE(1,*)'IR < K+1'
            END SELECT
    PAUSE
          STOP
        END IF
        IF(IER.EQ.2100)THEN
          WRITE(9,*)'WARNING MESSAGE FROM THE SUBROUTINE BLBORD'
          WRITE(9,*)'SINGULAR MATRIX'
          WRITE(1,*)'WARNING MESSAGE FROM THE SUBROUTINE BLBORD'
          WRITE(1,*)'SINGULAR MATRIX'
          IER=0
          GO TO 1000
         END IF
C
C          ... CALL GPADETYPE
C
       GPT=GPADETYPE (K, FKT, A)
```

```
C
C        ... CALL WRITING
C
      CALL WRITING (OPD, EPS, T, K, GPT, FT)
1000  WRITE(9,*)'(1) COMPUTE NEXT APPROXIMANT'
      WRITE(9,*)'(2) STOP'
      READ(9,*)OPC1
      IF(OPC1.NE.1.AND.OPC1.NE.2)THEN
         WRITE(9,*)'WRONG VALUE'
  WRITE(9,*)'TRY AGAIN'
         GO TO 1000
      END IF
      IF(OPC1.EQ.1)THEN
         K=K+1
         IF(K.GT.(IR-1))THEN
             WRITE(9,*)'K IS GREATHER THAN ITS MAXIMUM VALUE ',IR-1
      WRITE(1,*)'K IS GREATHER THAN ITS MAXIMUM VALUE ',IR-1
      PAUSE
      GO TO 1100
  END IF
        GO TO 700
      END IF
1100  WRITE(9,*)'(1) NEW EXAMPLE'
      WRITE(9,*)'(2) NEW POINT '
      WRITE(9,*)'(3) SAME EXAMPLE, SAME POINT,'
      WRITE(9,*)'    NEW SEQUENCE OF FUNCTIONALS L  '
      WRITE(9,*)'                                 K '
      WRITE(9,*)'(4) STOP'
      READ(9,*)OPC2
      WRITE(1,65)
      SELECT CASE(OPC2)
       CASE(1)
        GO TO 100
       CASE(2)
        GO TO 500
       CASE(3)
        GO TO 600
       CASE(4)
        STOP
       CASE DEFAULT
        WRITE(9,*)'WRONG VALUE'
 WRITE(9,*)'TRY AGAIN'
        GO TO 1100
      END SELECT
35    FORMAT(70('-'))
45    FORMAT(D24.16)
55    FORMAT(I2)
65    FORMAT(/,70('*'),/)
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                  +
C   SUBROUTINE NAME     -     READPOLY                                             +
C                                                                                  +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                  +
C   DESCRIPTION:                                                                   +
C                                                                                  +
C      READS THE DEGREE AND THE COEFFICIENTS OF A POLYNOMIAL                       +
C                                                                                  +
C   USAGE:                                                                         +
C            CALL READPOLY ( P )                                                   +
C                                                                                  +
C   ARGUMENTS:                                                                     +
C                                                                                  +
C            P - OUTPUT REAL VECTOR                                                +
C                                                                                  +
```

```
C                P(X) = p  + p  X + p  X**2 + ... + p  X**N                +
C                       0    1      2               N                      +
C                                                                          +
C                P   =  ( N, p  , p  , p  , ... , p  )                     +
C                           0    1    2          N                        +
C                                                                          +
C                                                                          +
C   REMARKS:                                                               +
C                                                                          +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                  +
C          PROCEDURE MUST BE IN DOUBLE PRECISION                           +
C                                                                          +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE READPOLY (P)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      DOUBLE PRECISION P(-1:-1)
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I,N
C
C        ... EXECUTABLE STATEMENTS
C
      WRITE(9,*)'DEGREE OF THE POLYNOMIAL ?'
      READ(9,*)N
      P(-1)=DBLE(N)
      WRITE(9,*)'COEFFICIENTS OF THE POLYNOMIAL (ONE BY LINE) ?'
      DO 10 I=0,N
         READ(9,*)P(I)
10    CONTINUE
      RETURN
      END
C
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                          +
C   SUBROUTINE NAME     -     WRITEPOLY                                    +
C                                                                          +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                          +
C   DESCRIPTION:                                                           +
C                                                                          +
C      WRITES THE DEGREE AND THE COEFFICIENTS OF A POLYNOMIAL              +
C                                                                          +
C   USAGE:                                                                 +
C            CALL  WRITEPOLY ( P )                                         +
C                                                                          +
C   ARGUMENTS:                                                             +
C                                                                          +
C            P - INPUT REAL VECTOR                                         +
C                                                                          +
C                P(X) = p  + p  X + p  X**2 + ... + p  X**N                +
C                       0    1      2               N                      +
C                                                                          +
C                P   =  ( N, p  , p  , p  , ... , p  )                     +
C                           0    1    2          N                        +
C                                                                          +
C                                                                          +
C   REMARKS:                                                               +
C                                                                          +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                  +
C          PROCEDURE MUST BE IN DOUBLE PRECISION                           +
C                                                                          +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
```

```
      SUBROUTINE WRITEPOLY (P)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      DOUBLE PRECISION P(-1:-1)
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER ID,I
C
C        ... EXECUTABLE STATEMENTS
C
      ID=IDINT(P(-1))
      WRITE(9,*)'(',ID,(P(I),I=0,ID),')'
      WRITE(9,*)
      PAUSE
      RETURN
      END
C
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME      -      HORNER                                         +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C    DOUBLE PRECISION FUNCTION :                                               +
C                                                                              +
C      EVALUATES A POLYNOMIAL IN A POINT USING HORNER ALGORITHM                +
C                                                                              +
C   USAGE:                                                                     +
C            CALL  HORNER ( P , X )                                            +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C            P  - INPUT REAL VECTOR CONTAINING THE DEGREE AND                  +
C                 THE COEFFICIENTS OF A POLYNOMIAL.                            +
C                                                                              +
C            X  - INPUT REAL VALUE. IT REPRESENTS THE POINT WHERE              +
C                 THE POLYNOMIAL P IS EVALUATED.                               +
C                                                                              +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION                               +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DOUBLE PRECISION FUNCTION HORNER ( P, X )
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      DOUBLE PRECISION P(-1:-1), X
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER ID,I
      DOUBLE PRECISION PX
C
C        ... EXECUTABLE STATEMENTS
C
      ID=IDINT(P(-1))
      PX=P(ID)
      DO 10 I=1,ID
```

```
         PX=PX*X+P(ID-I)
10    CONTINUE
      HORNER=PX
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     PRIMITIVEPOLY                                  +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C      COMPUTES THE PRIMITIVE OF A POLYNOMIAL                                  +
C                                                                              +
C   USAGE:                                                                     +
C            CALL  PRIMITIVEPOLY ( P , PRIVP )                                 +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C            P  - INPUT REAL VECTOR CONTAINING THE DEGREE AND                  +
C                 THE COEFFICIENTS OF A POLYNOMIAL.                            +
C                                                                              +
C         PRIV  - OUTPUT REAL VECTOR CONTAINING THE DEGREE AND                 +
C                 THE COEFFICIENTS OF THE PRIMITIVE OF P                       +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION                               +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE PRIMITIVEPOLY (P, PRIVP)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      DOUBLE PRECISION P(-1:-1), PRIVP(-1:-1)
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER PID,I
C
C        ... EXECUTABLE STATEMENTS
C
      PRIVP(-1)=P(-1)+1
      PID=IDINT(PRIVP(-1))
      PRIVP(0)=0.0D00
      DO 10 I=1,PID
         PRIVP(I)=P(I-1)/I
10    CONTINUE
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME     -     PFACT                                            +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C    INTEGER FUNCTION:                                                         +
C                                                                              +
```

```
C     COMPUTES THE PARTIAL FACTORIAL BETWEEN TWO NON NEGATIVE INTEGERS      +
C     I AND J ( I <= J ) :                                                  +
C                          I * (I+1) * ... * J.                             +
C                                                                           +
C                                                                           +
C   USAGE:                                                                  +
C            PFACT ( I , J )                                                +
C                                                                           +
C   ARGUMENTS:                                                              +
C                                                                           +
C        I , J  - INPUT INTEGER VALUES.                                     +
C                                                                           +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      INTEGER FUNCTION PFACT(I, J)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER I,J
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER K,P

C
C        ... EXECUTABLE STATEMENTS
C
      IF(I.LT.0.OR.J.LT.0)THEN
        WRITE(9,*)'STOP DUE TO ERROR IN THE FUNCTION PFACT'
 WRITE(9,*)'I AND/OR J ARE NEGATIVE'
 WRITE(9,*)'I=',I,'J=',J
 PAUSE
 WRITE(1,*)'STOP DUE TO ERROR IN THE FUNCTION PFACT'
 WRITE(1,*)'I AND/OR J ARE NEGATIVE'
 WRITE(1,*)'I=',I,'J=',J
        STOP
      END IF
      IF(J.LT.I)THEN
        WRITE(9,*)'STOP DUE TO ERROR IN THE FUNCTION PFACT'
 WRITE(9,*)'J < I'
        WRITE(9,*)'I=',I,'J=',J
 PAUSE
        WRITE(1,*)'STOP DUE TO ERROR IN THE FUNCTION PFACT'
 WRITE(1,*)'J < I'
        WRITE(1,*)'I=',I,'J=',J
        STOP
      END IF
      IF(J.EQ.0)THEN
        PFACT=1
 RETURN
      END IF
      P=1
      DO 10 K=I,J
        P=P*K
10    CONTINUE
      PFACT=P
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                           +
C     SUBROUTINE NAME     -     DERIVEPOLY                                  +
C                                                                           +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                           +
C   DESCRIPTION:                                                            +
C                                                                           +
```

```
C      COMPUTES THE DERIVATIVE OF ORDER J OF A POLYNOMIAL.                    +
C                                                                             +
C   USAGE:                                                                    +
C            CALL  DERIVEPOLY ( P , J , DJP )                                 +
C                                                                             +
C   ARGUMENTS:                                                                +
C                                                                             +
C            P  - INPUT REAL VECTOR CONTAINING THE DEGREE AND                 +
C                 THE COEFFICIENTS OF A POLYNOMIAL.                           +
C                                                                             +
C            J  - ORDER OF THE DERIVATIVE.                                    +
C                                                                             +
C          (J)                                                                +
C         D  P  - OUTPUT REAL VECTOR CONTAINING THE DEGREE AND                +
C                 THE COEFFICIENTS OF THE POLYNOMIAL DERIVATIVE               +
C                 OF ORDER J OF P.                                            +
C                                                                             +
C                                                                             +
C   EXTERNAL MODULES:                                                         +
C                                                                             +
C       - FUNCTION PFACT                                                      +
C                                                                             +
C                                                                             +
C                                                                             +
C   REMARKS:                                                                  +
C                                                                             +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                     +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                             +
C                                                                             +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE DERIVEPOLY (P, J, DJP)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER J
      DOUBLE PRECISION P(-1:-1),DJP(-1:-1)
C
C        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      INTEGER PFACT
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      DOUBLE PRECISION Z
      INTEGER ID,DID,I,I1,J1
C
C        ... EXECUTABLE STATEMENTS
C
      ID=IDINT(P(-1))
      IF(J.LE.0)THEN
C
C        ... WE DON'T DERIVE
C
         DO 10 I=-1,ID
             DJP(I)=P(I)
10       CONTINUE
      ELSE IF(J.GT.ID.OR.ID.EQ.0)THEN
C
C             ... P IS A CONSTANT OR THE ORDER OF THE DERIVATIVE
C                 IS GREATER THAN THE DEGREE OF P
C
               DJP(-1)=0.0D00
               DJP(0)=0.0D00
            ELSE IF(J.EQ.ID)THEN
                  DJP(-1)=0.0D00
     DJP(0)=PFACT(1,ID)*P(ID)
          ELSE
```

```
            DID=ID-J; DJP(-1)=DBLE(DID)
            DO 20 I=0,DID
                    DJP(I)=P(I+J)*PFACT(I+1,I+J)
20                CONTINUE
      END IF
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     BASISPOLY                                      +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C      COMPUTES THE POLYNOMIAL OF DEGREE K OF THE BASIS OF POLYNOMIALS         +
C      CHOSEN BY THE USER AND STORES IT IN THE (K+1)-TH ROW OF THE             +
C      MATRIX PB.                                                              +
C                                                                              +
C   USAGE:                                                                     +
C            CALL  BASISPOLY ( OPB, IR, WV, K, PB )                            +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C      OPB - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION OF THE MENU         +
C            OF POLYNOMIALS.                                                   +
C                                                                              +
C       IR - INPUT INTEGER DIMENSION EXACTLY AS SPECIFIED IN THE               +
C            DIMENSION STATEMENT IN THE CALLING PROGRAM FOR THE                +
C            MATRIX PB.                                                        +
C                                                                              +
C       WV - WORKING REAL VECTOR OF DIMENSION (-1:IR-1).                       +
C                                                                              +
C        K - INPUT INTEGER VALUE. IT REPRESENTS THE DEGREE OF THE              +
C            POLYNOMIAL TO BE COMPUTED.                                        +
C                                                                              +
C       PB - INPUT / OUTPUT REAL MATRIX OF DIMENSION (0:IR-1,-1:IR-1).         +
C            AS INPUT PB CONTAINS IN THE K FIRST ROWS, THE K FIRST             +
C            POLYNOMIALS P  ,P  ,..., P    OF THE BASIS OF POLYNOMIALS         +
C                        0   1        K-1                                      +
C            CHOSEN BY THE USER.                                               +
C            DURING THE CALL OF THE SUBROUTINE THE POLYNOMIAL P                +
C                                                             k                +
C            IS COMPUTED AND STORED IN THE (K+1)-TH ROW OF PB.                 +
C            AS OUTPUT PB CONTAINS, IN THE K+1 FIRST ROWS, THE                 +
C            K+1 POLYNOMIALS P  ,...,P  .                                      +
C                             0       K                                        +
C                                                                              +
C                                                                              +
C   EXTERNAL MODULES:                                                          +
C                                                                              +
C       - SUBROUTINE READPOLY                                                  +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE BASISPOLY (OPB ,IR, WV ,K ,PB)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPB,IR,K
```

```
      DOUBLE PRECISION PB(0:IR-1,-1:-1),WV(-1:-1)
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER ID,J
C
C        ... EXECUTABLE STATEMENTS
C
      SELECT CASE(OPB)
         CASE(1)
C
C          ... CANONICAL BASIS
C
            PB(K,-1)=DBLE(K)
            DO 10 J=0,K-1
              PB(K,J)=0.0D00
10          CONTINUE
            PB(K,K)=1.0D00
         CASE(2)
C
C          ... FIRST KIND TCHEBYSHEF BASIS
C
            SELECT CASE(K)
               CASE(0)
                 PB(0,-1)=0.0D00;PB(0,0)=1.0D00
               CASE(1)
                 PB(1,-1)=1.0D00;PB(1,0)=0.0D00;PB(1,1)=1.0D00
               CASE DEFAULT
                 PB(K,-1)=DBLE(K);PB(K,0)=-PB(K-2,0)
          DO 20 J=1,K-2
                    PB(K,J)=2.*PB(K-1,J-1)-PB(K-2,J)
20                CONTINUE
                  PB(K,K-1)=2.*PB(K-1,K-2)
                  PB(K,K)=2.*PB(K-1,K-1)
            END SELECT
          CASE(3)
C
C          ... SECOND KIND TCHEBYSHEF BASIS
C
            SELECT CASE(K)
              CASE(0)
                PB(0,-1)=0.0D00;PB(0,0)=1.0D00
       CASE(1)
                PB(1,-1)=1.0D00;PB(1,0)=0.0D00;PB(1,1)=2.0D00
              CASE DEFAULT
                PB(K,-1)=DBLE(K);PB(K,0)=PB(K-2,0)
                DO 30 J=1,K-2
                   PB(K,J)=2.*PB(K-1,J-1)+PB(K-2,J)
30              CONTINUE
                PB(K,K-1)=2.*PB(K-1,K-2)
                PB(K,K)=2.*PB(K-1,K-1)
              END SELECT
          CASE(4)
C
C             ... BASIS GIVEN BY THE USER
C
            WRITE(9,*)'GIVE A POLYNOMIAL OF DEGREE = ',K
100      CALL READPOLY (WV)
      ID=IDINT(WV(-1))
      IF(ID.NE.K)THEN
               WRITE(9,*)'THE DEGREE OF THE POLYNOMIAL IS <> ',K
         WRITE(9,*)'TRY AGAIN'
         GO TO 100
      ENDIF
      DO 40 J=-1,ID
               PB(K,J)=WV(J)
40          CONTINUE
          CASE DEFAULT
```

```
      WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE BASISPOLY.'
    WRITE(9,*)'CODE OF BASISPOLY INCOMPLETE.'
      WRITE(9,*)'BASIS OF POLYNOMIALS UNDEFINED.'
    PAUSE
    WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE BASISPOLY.'
    WRITE(1,*)'CODE OF BASISPOLY INCOMPLETE.'
      WRITE(1,*)'BASIS OF POLYNOMIALS UNDEFINED.'
    STOP
      END SELECT
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     MODIFIEDMOMENTS                                +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C      COMPUTES THE MODIFIED MOMENT OF ORDER K, CORRESPONDING TO               +
C      THE OPTIONS OF THE SERIES AND THE GENERATING FUNCTION, AND              +
C      STORES IT IN THE (K+1)-TH COMPONENT OF THE VECTOR C.                    +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C            CALL  MODIFIEDMOMENTS ( OPS, OPG, K, C )                          +
C                                                                              +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C        OPS - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF THE MENU OF THE SERIES.                                      +
C                                                                              +
C        OPG - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF THE MENU OF THE GENERATING FUNCTIONS.                        +
C                                                                              +
C         C  - INPUT / OUTPUT REAL VECTOR OF DIMENSION (0:IR-1).               +
C              AS INPUT C CONTAINS IN THE K FIRST COMPONENTS, THE K            +
C              FIRST MODIFIED MOMENTS C  ,C  ,...,C   .                        +
C                                     0   1       K-1                          +
C              DURING THE CALL OF THE SUBROUTINE THE MOMENT C  IS              +
C                                                            K                 +
C              CALCULATED AND STORED IN THE (K+1)-TH COMPONENT OF C.           +
C              AS OUTPUT C CONTAINS, IN THE K+1 FIRST COMPONENTS THE           +
C              MOMENTS C  ,...,C  .                                            +
C                       0       K                                              +
C                                                                              +
C          K - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF THE             +
C              MODIFIED MOMENT TO BE COMPUTED.                                 +
C                                                                              +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE MODIFIEDMOMENTS (OPS, OPG, K, C)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPS,OPG,K
      DOUBLE PRECISION C(0:0)
C
```

```
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER K1
C
C        ... EXECUTABLE STATEMENTS
C
      SELECT CASE(OPS)
        CASE(1)
C
C         ... F(T)=EXP(-T)
C
          SELECT CASE(OPG)
             CASE(1)
C
C                      ... G(X,T)=1/(1-XT)
C
                IF(K.EQ.0)THEN
             C(0)=1.0D00
          ELSE
             C(K)=-C(K-1)/K
          END IF
             CASE(2)
C
C                    ...  G(X,T)=EXP(-XT)
C
                C(K)=1.0D00
       CASE DEFAULT
         GO TO 100
     END SELECT
        CASE(2)
C
C      ... F(T)=LOG(1+T)/T
C
         SELECT CASE(OPG)
                CASE(1)
C
C                     ... G(X,T)=1/(1-XT)
C
                   C(K)=((-1.0)**K)/(K+1.0D00)
                CASE(3)
C
C                     ... G(X,T)=LOG(1+XT)
C
                   C(K)=1.0D00
                CASE DEFAULT
                  GO TO 100
               END SELECT
        CASE DEFAULT
  WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE'
  WRITE(9,*)'MODIFIEDMOMENTS.'
  WRITE(9,*)'SERIES UNDEFINED.'
  WRITE(9,*)'CODE OF MODIFIEDMOMENTS INCOMPLETE OR WRONG CHOICES'
  WRITE(9,*)'OF THE GENERATING FUNCTION AND/OR SERIES.'
  PAUSE
  WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE'
  WRITE(1,*)'MODIFIEDMOMENTS.'
  WRITE(1,*)'SERIES UNDEFINED.'
  WRITE(1,*)'CODE OF MODIFIEDMOMENTS INCOMPLETE OR WRONG CHOICES'
  WRITE(1,*)'OF THE GENERATING FUNCTION AND/OR SERIES.'
  STOP
 END SELECT
 RETURN
100     WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE MODIFIEDMOMENTS.'
        WRITE(9,*)'GENERATING FUNCTION UNDEFINED.'
 WRITE(9,*)'CODE OF MODIFIEDMOMENTS INCOMPLETE OR WRONG CHOICES'
 WRITE(9,*)'OF THE GENERATING FUNCTION AND/OR SERIES.'
 PAUSE
 WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE MODIFIEDMOMENTS.'
```

```
 WRITE(1,*)'GENERATING FUNCTION UNDEFINED.'
 WRITE(1,*)'CODE OF MODIFIEDMOMENTS INCOMPLETE OR WRONG CHOICES'
 WRITE(1,*)'OF THE GENERATING FUNCTION AND/OR SERIES.'
 STOP
 END
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME   -   F                                                    +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C     DOUBLE PRECISION FUNCTION:                                               +
C                                                                              +
C              COMPUTES THE VALUE OF THE SERIES IN A POINT.                    +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C            F ( OPS , T )                                                     +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C        OPS  - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                  +
C               OF THE MENU OF THE SERIES.                                     +
C                                                                              +
C         T   - INPUT REAL VALUE. IT REPRESENTS THE POINT WHERE                +
C               THE SERIES OF FUNCTIONS IS EVALUATED.                          +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DOUBLE PRECISION FUNCTION  F(OPS, T)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPS
      DOUBLE PRECISION T
C
C        ... EXECUTABLE STATEMENTS
C
      SELECT CASE(OPS)
        CASE(1)
C
C           ... F(T)=EXP(-T)
C
            F=DEXP(-T)
        CASE(2)
C
C           ... F(T)=LOG(1+T)/T
C
            F=DLOG(1+T)/T
        CASE DEFAULT
     WRITE(9,*)'STOP DUE TO ERROR IN THE FUNCTION F.'
     WRITE(9,*)'CODE OF F INCOMPLETE.'
     WRITE(9,*)'SERIES UNDEFINED.'
     PAUSE
     WRITE(1,*)'STOP DUE TO ERROR IN THE FUNCTION F.'
     WRITE(1,*)'CODE OF F INCOMPLETE.'
     WRITE(1,*)'SERIES UNDEFINED.'
     STOP
       END SELECT
```

```
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME      -      ELFUNGEN                                       +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C     DOUBLE PRECISION FUNCTION :                                              +
C                                                                              +
C       ACCORDING TO THE VALUE OF OP, COMPUTES THE APPLICATION OF ONE          +
C       OF THE FOLLOWING FUNCTIONALS TO THE GENERATING FUNCTION                +
C       G(X)=G(X,T).                                                           +
C                                                                              +
C                                                                              +
C      - (OP=1) EVALUATION IN A POINT X.                                       +
C                                                                              +
C      - (OP=2) EVALUATION OF THE DERIVATIVE OF ORDER J, WITH                  +
C               RESPECT TO THE FIRST VARIABLE, IN THE POINT X.                 +
C                                                                              +
C      - (OP=3) EVALUATION OF THE INTEGRAL, WITH RESPECT TO THE                +
C               FIRST VARIABLE, BETWEEN THE POINTS X AND Y.                    +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C                    ELFUNGEN  ( OPG, OP, X, T, J, Y )                         +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C                                                                              +
C        OPG - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF THE MENU OF THE GENERATING FUNCTIONS.                        +
C                                                                              +
C         OP - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF THE FUNCTIONALS.                                             +
C                                                                              +
C          X - INPUT REAL VALUE. IT REPRESENTS THE POINT WHERE THE             +
C              GENERATING FUNCTION OR ITS DERIVATIVE OF ORDER J                +
C              ARE EVALUATED, IF OP=1 OR OP=2 RESPECTIVELY.                    +
C                                                                              +
C          T - INPUT REAL VALUE. IT REPRESENTS THE VALUE OF THE                +
C              SECOND VARIABLE OF G.                                           +
C                                                                              +
C          J - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF THE             +
C              DERIVATIVE, IF OP=2.                                            +
C                                                                              +
C       X, Y - INPUT REAL VALUES. THEY REPRESENT THE LIMITS OF THE             +
C              INTEGRAL, IF OP=3.                                              +
C                                                                              +
C                                                                              +
C                                                                              +
C   EXTERNAL MODULES:                                                          +
C                                                                              +
C       - FUNCTION PFACT                                                       +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DOUBLE PRECISION FUNCTION ELFUNGEN ( OPG, OP, X, T, J, Y )
```

```
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPG,OP,J
      DOUBLE PRECISION X,T,Y
C
C        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      INTEGER PFACT
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I,J1
      DOUBLE PRECISION Z1,Z2,Z3,Z4,Z5,GXT1,GXT2
C
C        ... EXECUTABLE STATEMENTS
C
      IF(OP.NE.1.AND.OP.NE.2.AND.OP.NE.3)THEN
         WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE ELFUNGEN.'
         WRITE(9,*)'UNDEFINED VALUE OF THE ARGUMENT OP.'
         PAUSE
         WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE ELFUNGEN.'
         WRITE(1,*)'UNDEFINED VALUE OF THE ARGUMENT OP.'
         STOP
      END IF
      SELECT CASE(OPG)
        CASE(1)
C
C  ... G(X,T)=1/(1-XT)
C
          SELECT CASE(OP)
            CASE(1)
              ELFUNGEN=1/(1-X*T)
            CASE(2)
              SELECT CASE(J)
         CASE(0)
           ELFUNGEN=1/(1-X*T)
         CASE DEFAULT
                 ELFUNGEN=((T**J)/((1-X*T)**(J+1)))*PFACT(1,J)
       END SELECT
            CASE(3)
              ELFUNGEN=(DLOG(1-X*T)-DLOG(1-Y*T))/T
            END SELECT
        CASE(2)
C
C    ... G(X,T)=EXP(-XT)
C
            SELECT CASE(OP)
       CASE(1)
         ELFUNGEN=DEXP(-X*T)
       CASE(2)
         ELFUNGEN=((-T)**J)*DEXP(-X*T)
       CASE(3)
         ELFUNGEN=(DEXP(-X*T)-DEXP(-Y*T))/T
     END SELECT
       CASE(3)
C
C       ... G(X,T)=LOG(1+XT)/T
C
            SELECT CASE(OP)
       CASE(1)
         ELFUNGEN=DLOG(1+X*T)/T
       CASE(2)
         SELECT CASE(J)
           CASE(0)
             ELFUNGEN=DLOG(1+X*T)/T
           CASE(1)
             ELFUNGEN=1/(1+X*T)
```

```
          CASE DEFAULT
        J1=J-1
        Z1=(T**J1)/((1+X*T)**J)
        ELFUNGEN=((-1)**J1)*Z1*PFACT(1,J1)
                 END SELECT
         CASE(3)
           Z1=1+X*T; Z2=1+Y*T
    ELFUNGEN=(Z2*(-1+DLOG(Z2))-Z1*(-1+DLOG(Z1)))/(T**2)
      END SELECT
       CASE DEFAULT
        WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE ELFUNGEN.'
        WRITE(9,*)'CODE OF ELFUNGEN INCOMPLETE.'
        WRITE(9,*)'GENERATING FUNCTION UNDEFINED.'
        PAUSE
        WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE ELFUNGEN.'
        WRITE(1,*)'CODE OF ELFUNGEN INCOMPLETE.'
        WRITE(1,*)'GENERATING FUNCTION UNDEFINED.'
        STOP
       END SELECT
       RETURN
       END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     FUNGENERATING                                  +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C             COMPUTES THE APPLICATION OF THE Lk FUNCTIONAL                    +
C             TO THE GENERATING FUNCTION G IN THE POINT T                      +
C             Fk(T)=Lk(G(X,T)).                                                +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C            CALL  FUNGENERATING (OPG, L, IR, T, K, FKT)                       +
C                                                                              +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C        OPG - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF THE MENU OF THE GENERATING FUNCTIONS.                        +
C                                                                              +
C          L - INPUT REAL MATRIX OF DIMENSION (0:IR-1,0:IL)                    +
C              CONTAINING, IN THE K+1 FIRST ROWS THE DEFINITIONS               +
C              OF THE K+1 FIRST FUNCTIONALS L  , L  ,...,L  .                  +
C                                           0    1       K                     +
C                                                                              +
C         IR - INPUT INTEGER ROWS DIMENSION OF THE MATRIX L.                   +
C                                                                              +
C          T - INPUT REAL VALUE. IT REPRESENTS THE POINT WHERE THE             +
C              THE FUNCTION F (.)=L (G(X,.)) IS EVALUATED.                     +
C                              K     K                                         +
C                                                                              +
C          K - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF                 +
C              THE FUNCTIONAL TO BE APPLIED.                                   +
C                                                                              +
C        FKT - INPUT / OUTPUT REAL VECTOR OF DIMENSION (0:IR-1).               +
C              AS INPUT FKT CONTAINS IN THE FIRST K COMPONENTS                 +
C              THE K VALUES F (T)=L (G(X,T)) ,..., F    (T)=L    (G(X,T)).    +
C                             0     0                K-1     K-1               +
C              DURING THE CALL OF THE SUBROUTINE F (T)=L (G(X,T)) IS           +
C                                                 K     K                      +
C              COMPUTED AND STORED IN THE K+1 COMPONENT OF FKT.                +
C              AS OUTPUT FKT CONTAINS IN THE FIRST K+1 COMPONENTS              +
C              THE VALUES F (T)=L (G(X,T)) ,..., F (T)=L (G(X,T)).             +
```

```
C                          0     0                K     K                  +
C                                                                          +
C                                                                          +
C   EXTERNAL MODULES:                                                      +
C                                                                          +
C       - FUNCTION ELFUNGEN                                                +
C                                                                          +
C                                                                          +
C   REMARKS:                                                               +
C                                                                          +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                  +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                          +
C                                                                          +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE  FUNGENERATING (OPG, L, IR, T, K, FKT)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPG,IR,K
      DOUBLE PRECISION T
      DOUBLE PRECISION L(0:IR-1,0:0),FKT(0:0)
C
C        ... SPECIFICATIONS OF EXTERNAL FUNCTIONS
C
      DOUBLE PRECISION ELFUNGEN
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER OPEF,N,I,I1,I2,J
      DOUBLE PRECISION S
C
C        ... EXECUTABLE STATEMENTS
C
      OPEF=IDINT(L(K,0))
C
C       ... OPEF IS THE OPTION OF THE FUNCTIONAL Lk
C
      SELECT CASE(OPEF)
       CASE(1)
           FKT(K)=ELFUNGEN (OPG,1,L(K,1),T,0,0)
       CASE(2)
           J=IDINT(L(K,2))
           FKT(K)=ELFUNGEN (OPG,2,L(K,1),T,J,0)
       CASE(3)
           FKT(K)=ELFUNGEN (OPG,3,L(K,1),T,0,L(K,2))
       CASE(4)
    N=IDINT(L(K,1))
    S=0.0D00
    DO 10 I=1,N
      I1=N+1+I
            S=S+L(K,I+1)*ELFUNGEN (OPG,1,L(K,I1),T,0,0)
10         CONTINUE
           FKT(K)=S
       CASE(5)
           N=IDINT(L(K,1))
    S=0.0D00
    DO 20 I=1,N
      I2=2*N+1+I
      J=IDINT(L(K,I2))
      I1=N+1+I
            S=S+L(K,I+1)*ELFUNGEN (OPG,2,L(K,I1),T,J,0)
20         CONTINUE
           FKT(K)=S
       CASE(6)
           N=IDINT(L(K,1))
    S=0.0D00
    DO 30 I=1,N
```

```
      I1=N+1+I
      I2=2*N+1+I
            S=S+L(K,I+1)*ELFUNGEN (OPG,3,L(K,I1),T,0,L(K,I2))
30          CONTINUE
            FKT(K)=S
        CASE DEFAULT
     WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE FUNGENERATING'
     WRITE(9,*)'CODE OF FUNGENERATING INCOMPLETE.'
     WRITE(9,*)'FUNCTIONAL Lk UNDEFINED.'
     PAUSE
     WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE FUNGENERATING'
     WRITE(1,*)'CODE OF FUNGENERATING INCOMPLETE.'
     WRITE(1,*)'FUNCTIONAL Lk UNDEFINED.'
     STOP
       END SELECT
       RETURN
       END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME     -     FUNPOLY                                          +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C    DOUBLE PRECISION FUNCTION:                                                +
C                                                                              +
C     COMPUTES THE APPLICATION OF THE FUNCTIONAL L  TO THE POLYNOMIAL U.        +
C                                                J                             +
C                                                                              +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C              FUNPOLY (L, IR, J, WV, V)                                       +
C                                                                              +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C                                                                              +
C          L - INPUT REAL MATRIX OF DIMENSION (0:IR-1,0:IL),                   +
C              CONTAINING THE DEFINITIONS OF THE FUNCTIONALS L  ,L  ,....      +
C                                                             0   1            +
C                                                                              +
C         IR - INPUT INTEGER ROWS DIMENSION OF THE MATRIX L.                   +
C                                                                              +
C          J - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF THE             +
C              FUNCTIONAL THAT SHOULD BE APPLIED TO THE VECTOR V.              +
C                                                                              +
C         WV - WORKING REAL VECTOR OF DIMENSION (-1:IR-1).                     +
C                                                                              +
C          V - INPUT REAL VECTOR OF DIMENSION (-1:IR-1).                         +
C                                                                              +
C                                                                              +
C                                                                              +
C   EXTERNAL MODULES:                                                          +
C                                                                              +
C       - SUBROUTINE  DERIVEPOLY, INTEGRALPOLY.                                +
C                                                                              +
C       - DOUBLE PRECISION FUNCTION HORNER.                                    +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
C
      DOUBLE PRECISION FUNCTION FUNPOLY (L, IR, J, WV, V)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER IR,J
      DOUBLE PRECISION L(0:IR-1,0:0),V(-1:-1),WV(-1:-1)
C
C        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      DOUBLE PRECISION HORNER
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER J1,OPF,I,N,K,K1,K2
      DOUBLE PRECISION S
C
C        ... EXECUTABLE STATEMENTS
C
      OPF=IDINT(L(J,0))
C
C     ... OPF = OPTION OF THE MENU OF FUNCTIONALS L
C                                                  J
C
      SELECT CASE (OPF)
        CASE(1)
          FUNPOLY=HORNER( V, L(J,1) )
        CASE(2)
          J1=IDINT(L(J,2))
   CALL DERIVEPOLY( V, J1, WV )
   FUNPOLY=HORNER( WV, L(J,1) )
        CASE(3)
          CALL PRIMITIVEPOLY(V,WV)
   FUNPOLY=HORNER(WV, L(J,2))-HORNER(WV, L(J,1))
       CASE(4)
          N=IDINT(L(J,1))
   S=0.0D00
   DO 10 K=1,N
     K1=N+1+K
            S=S+L(J,K+1)*HORNER(V, L(J,K1))
10         CONTINUE
           FUNPOLY=S
        CASE(5)
          N=IDINT(L(J,1))
   S=0.0D00
   DO 20 K=1,N
     k2=2*N+1+K
     J1=IDINT(L(J,K2))
     K1=N+1+K
     CALL DERIVEPOLY( V, J1, WV )
            S=S+L(J,K+1)*HORNER( WV, L(J,K1) )
20         CONTINUE
           FUNPOLY=S
        CASE(6)
          N=IDINT(L(J,1))
   S=0.0D00
   CALL PRIMITIVEPOLY( V, WV )
   DO 30 K=1,N
     K1=N+1+K
     K2=2*N+1+K
            S=S+L(J,K+1)*( HORNER(WV, L(J,K2))-HORNER(WV, L(J,K1)) )
30         CONTINUE
           FUNPOLY=S
       CASE DEFAULT
          WRITE(9,*)'STOP DUE TO ERROR IN THE SUBROUTINE FUNPOLY.'
   WRITE(9,*)'CODE OF FUNPOLY INCOMPLETE.'
   WRITE(9,*)'FUNCTIONAL Lk UNDEFINED.'
   PAUSE
```

```
      WRITE(1,*)'STOP DUE TO ERROR IN THE SUBROUTINE FUNPOLY.'
      WRITE(1,*)'CODE OF FUNPOLY INCOMPLETE.'
      WRITE(1,*)'FUNCTIONAL Lk UNDEFINED.'
      STOP
         END SELECT
         RETURN
         END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     SYSTEM                                       +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C     COMPUTES THE ROW AND THE COLUMN OF ORDER K+1 OF THE MATRIX M    AND      +
C                                                                  k+1         +
C     THE COMPONENT OF ORDER K+1 OF THE RIGHT HAND SIDE D    OF THE SYSTEM     +
C                                                        K+1                   +
C     M    A   = D    WHOSE SOLUTIONS A   = (a  ,...,a  ) ARE THE             +
C      K+1  K+1   K+1                  K+1    0       k                       +
C     COEFFICIENTS OF THE (K)-TH GENERALIZED PADE-TYPE APPROXIMANT:            +
C                                                                              +
C                   (K)  (T) = a * F (T) + ... + a * F (T)                     +
C                      F        0   0             k   k                       +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C         CALL  SYSTEM (L, IR, WV1, WV2, PB, C, K, M, D)                       +
C                                                                              +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C                                                                              +
C         L - INPUT REAL MATRIX OF DIMENSIONS (0:IR-1,0:IL),                   +
C             CONTAINING IN THE K+1 FIRST COMPONENTS THE DEFINITIONS           +
C             OF THE FUNCTIONALS L ,..., L .                                   +
C                                 0      K                                     +
C                                                                              +
C        IR - INPUT INTEGER ROWS OR COLUMNS DIMENSIONS OF THE                  +
C             MATRICES L, PB AND M AND THE VECTOR D.                           +
C                                                                              +
C    WV1,WV2 - WORKING REAL VECTORS OF DIMENSION (-1:IR-1).                    +
C                                                                              +
C         PB - INPUT REAL MATRIX OF DIMENSION (0:IR-1,-1:IR-1),                +
C              CONTAINING IN THE K+1 FIRST ROWS THE K+1                        +
C              POLYNOMIALS P ,..., P   OF THE BASIS OF POLYNOMIALS             +
C                           0      K                                           +
C              CHOSEN BY THE USER .                                            +
C                                                                             +
C         C  - INPUT REAL VECTOR OF DIMENSION (0:IR-1), CONTAINING IN          +
C              THE K+1 FIRST COMPONENTS THE MODIFIED MOMENTS                   +
C               C  ,..., C   .                                                 +
C                0       K                                                     +
C                                                                              +
C          K - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF THE             +
C              GENERALIZED PADE-TYPE APPROXIMANT TO BE COMPUTED.               +
C                                                                              +
C          M - INPUT / OUTPUT REAL SQUARE MATRIX OF DIMENSION (IR,IR).         +
C              AS INPUT M CONTAINS IN THE FIRST K ROWS AND COLUMS THE          +
C              MATRIX M  .                                                     +
C                      k                                                       +
C              DURING THE CALL OF THE SUBROUTINE A NEW ROW AND COLUMN          +
C              ARE COMPUTED AND ADDED TO  M   TO FORM THE BORDERED             +
C                                         k                                    +
C              MATRIX M   .                                                    +
```

```
C                    K+1                                                    +
C              AS OUTPUT M CONTAINS IN THE K+1 FIRST ROWS ANS COLUMNS       +
C              THE MATRIX M    .                                            +
C                         k+1                                               +
C                                                                           +
C          D - INPUT / OUTPUT REAL VECTOR OF DIMENSION (IR).                +
C              AS INPUT D CONTAINS IN THE FIRST K COMPONENTS THE VECTOR      +
C              D  .                                                         +
C               k                                                           +
C              DURING THE CALL OF THE SUBROUTINE A NEW COMPONENT IS         +
C              COMPUTED AND ADDED TO D   TO FORM THE BORDERED VECTOR D    .  +
C                                     k                                 k+1  +
C              AS OUTPUT D CONTAINS IN THE K+1 FIRST COMPONENTS THE         +
C              VECTOR D  .                                                  +
C                      k+1                                                  +
C                                                                           +
C                                                                           +
C    EXTERNAL MODULES:                                                      +
C                                                                           +
C        - DOUBLE PRECISION FUNCTION FUNPOLY.                               +
C                                                                           +
C                                                                           +
C    REMARKS:                                                               +
C                                                                           +
C        -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                  +
C           PROCEDURE MUST BE IN DOUBLE PRECISION.                          +
C                                                                           +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE SYSTEM (L, IR, WV1, WV2, PB, C, K, M, D)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER IR,K
      DOUBLE PRECISION M(IR,1), D(1), PB(0:IR-1,-1:-1),
     *                 L(0:IR-1,0:0),C(0:0),WV1(-1:-1),WV2(-1:-1)
C
C        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      DOUBLE PRECISION FUNPOLY
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I,I1,J
C
C        ... EXECUTABLE STATEMENTS
C
C
C       ... DEFINITION OF THE ROW OF ORDER K+1 OF THE MATRIX M
C
      DO 10 I=-1,K
        WV1(I)=PB(K,I)
10    CONTINUE
      DO 20 J=0,K
        M(K+1,J+1)=FUNPOLY(L,IR,J,WV2,WV1)
20    CONTINUE
C
C       ... DEFINITION OF THE COMPONENT OF ORDER K+1 OF THE VECTOR D
C
      D(K+1)=C(K)
C
C       ... DEFINITION OF THE COLUMN OF ORDER K+1 OF THE MATRIX M
C
      DO 30 I=0,K-1
         DO 40 J=-1,I
           WV1(J)=PB(I,J)
40       CONTINUE
         M(I+1,K+1)=FUNPOLY(L,IR,K,WV2,WV1)
```

```
30    CONTINUE
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     FUNCTION NAME   -   GPADETYPE                                            +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C      DOUBLE PRECISION FUNCTION:                                              +
C                                                                              +
C           COMPUTES THE VALUE OF THE K-TH GENERALIZED PADE-TYPE               +
C           APPROXIMANT IN THE POINT T:                                        +
C                                                                              +
C                   (K)  (T) = a * F (T) + ... + a * F (T)                     +
C                     F        0   0             k   k                         +
C                                                                              +
C   USAGE:                                                                     +
C            GPADETYPE (K, FKT, A)                                             +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C         K  - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER                    +
C              OF THE GENERALIZED PADE-TYPE APPROXIMANT.                       +
C                                                                              +
C        FKT - INPUT REAL VECTOR OF DIMENSION (0:IR-1), CONTAINING IN          +
C              THE K+1 FIRST COMPONENTS THE VALUES F (T),...,F  (T).           +
C                                                  0        k                  +
C                                                                              +
C         A  - INPUT REAL VECTOR OF DIMENSION (IR), CONTAINING IN THE          +
C              K+1 FIRST COMPONENTS THE COEFFICIENTS  a  ,...,a  .             +
C                                                     0      k                 +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DOUBLE PRECISION FUNCTION GPADETYPE (K, FKT, A)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER K
      DOUBLE PRECISION A(1),FKT(0:0)
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I
      DOUBLE PRECISION S
C
C        ... EXECUTABLE STATEMENTS
C
      S=0.0D00
      DO 10 I=0,K
        S=S+A(I+1)*FKT(I)
10    CONTINUE
      GPADETYPE=S
      RETURN
      END
C
C
```

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C     SUBROUTINE NAME     -     WRITING                                        +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C        PRINTS AND WRITES THE K-TH GENERALIZED PADE-TYPE APPROXIMANT          +
C        IN THE POINT T ( (K) (T) ), AND THE CORRESPONDING ABSOLUTE AND        +
C                       F                                                      +
C        RELATIVE ERRORS IF THE SERIES IS AVAILABLE.                           +
C                                                                              +
C                                                                              +
C   USAGE:                                                                     +
C             CALL  WRITING (OPD, EPS, T, K, GPT, FT)                          +
C                                                                              +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C                                                                              +
C        OPD - INPUT INTEGER VALUE. IT REPRESENTS THE OPTION                   +
C              OF AVAILABILITY OF THE DEFINITION OF THE SERIES.                +
C                                                                              +
C        EPS - INPUT REAL VALUE USED IN TESTS FOR ZERO.                        +
C              IF DABS(X) .LT. EPS, THEN X IS CONSIDERED TO BE ZERO.           +
C                                                                              +
C          K - INPUT INTEGER VALUE. IT REPRESENTS THE ORDER OF THE             +
C              GENERALIZED PADE-TYPE APPROXIMANT.                              +
C                                                                              +
C        GPT - INPUT REAL VALUE. IT REPRESENTS THE VALUE OF THE                +
C              GENERALIZED PADE-TYPE APPROXIMANT IN THE POINT T.               +
C                                                                              +
C         FT - INPUT REAL VALUE. IT REPRESENTS THE VALUE OF THE                +
C              SERIES IN THE POINT T, IF THE SERIES IS AVAILABLE.              +
C                                                                              +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       -  ALL THE REAL ARGUMENTS PASSED FROM THE CALLING                      +
C          PROCEDURE MUST BE IN DOUBLE PRECISION.                              +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE WRITING (OPD, EPS, T, K, GPT, FT)
C
C        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER OPD,K
      DOUBLE PRECISION EPS,T,GPT,FT
C
C        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
       DOUBLE PRECISION RELERROR,ABSERROR,Z
C
C        ... EXECUTABLE STATEMENTS
C
C        ... OPD = 1 THE SERIES IS AVAILABLE
C        ... OPD = 2 THE SERIES IS NOT AVAILABLE
C
      SELECT CASE(OPD)
       CASE(1)
  Z=DABS(FT)
         ABSERROR=DABS(GPT-FT)
  IF(Z.LT.EPS)THEN
     WRITE(9,100)K,GPT,ABSERROR
     WRITE(1,100)K,GPT,ABSERROR
100         FORMAT('(',I2,')=',D24.16,2X,'ABS ERROR=',D10.2,/)
```

```
  ELSE
     RELERROR=ABSERROR/Z
     WRITE(9,200)K,GPT,ABSERROR,RELERROR
           WRITE(1,200)K,GPT,ABSERROR,RELERROR
200        FORMAT('(',I2,')=',D24.16,2X,'ABS ERROR=',
     *             D10.2,2X,'REL ERROR=',D 10.2/)
        END IF
       CASE(2)
         WRITE(9,300)K,GPT
  WRITE(1,300)K,GPT
300      FORMAT('(',I2,')=',D24.16,/)
      END SELECT
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                     +
C   SUBROUTINE NAME     - BLBORD                                                      +
C                                                                                     +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                     +
C   DESCRIPTION:                                                                      +
C                                                                                     +
C           COMPUTES THE SOLUTION z    OF THE BORDERED SYSTEM                         +
C                                  k+1                                                +
C           A    z    = d  IN TERMS OF THE SOLUTION z  OF THE PREVIOUS               +
C            k+1  k+1    k+1                          k                              +
C           SYSTEM A  z  = d  .                                                       +
C                  k  k    k                                                          +
C                                                                                     +
C   USAGE:                                                                            +
C           CALL BLBORD (A, D, IR, Z, BETA, WK, EPS, INIT, IER)                       +
C                                                                                     +
C   ARGUMENTS:                                                                        +
C                                                                                     +
C           A       - INPUT REAL SQUARE MATRIX OF DIMENSION (IR,IR).                  +
C                                                                                     +
C           D       - INPUT REAL VECTOR OF DIMENSION IR.                              +
C                                                                                     +
C           IR      - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE                +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR                 +
C                     THE MATRIX A.                                                   +
C                                                                                     +
C           Z       - OUTPUT REAL VECTOR OF DIMENSION IR CONTAINING                   +
C                     AFTER THE k-TH CALL, IN THE FIRST k COMPONENTS,                 +
C                     THE SOLUTION z  OF A z = d .                                    +
C                                   k     k k   k                                     +
C                                                                                     +
C           BETA    - WORKING REAL MATRIX OF DIMENSION (IR,2*IR).                     +
C                     IF THE MATRIX beta IS NON SINGULAR (INDS.EQ.0) ITS FIRST        +
C                                      k                              -1              +
C                     COLUMNS CONTAIN, IN OUTPUT, THE MATRIX beta .                   +
C                                                                    k                +
C                                                                                     +
C           WK      - WORKING REAL VECTOR OF DIMENSION IR.                            +
C                                                                                     +
C           EPS     - INPUT REAL VALUE USED IN TESTS FOR ZERO.                        +
C                     IF DABS(X) .LT. EPS, THEN X IS CONSIDERED TO BE ZERO.           +
C                                                                                     +
C           INIT    - INPUT/OUTPUT INTEGER TO BE SET TO ZERO BEFORE THE               +
C                     FIRST CALL OF THE SUBROUTINE. ITS VALUE IS CHANGED              +
C                     TO 1 BY THE SUBROUTINE DURING THE FIRST CALL.                   +
C                     FOR A NEW APPLICATION OF THE METHOD INIT MUST SET               +
C                     AGAIN TO ZERO.                                                  +
C                                                                                     +
C           IER     - INPUT/OUTPUT INDEX WARNING/ERROR.                               +
C                     IER =  200    CALL OF THE SUBROUTINE WITH A NON ZERO            +
```

```
C                                  IER VALUE.                                  +
C                     IER =  300   ROWS DIMENSION TOO SMALL IN THE CALLING     +
C                                  PROCEDURE COMPARED TO NUMBER_OF_CALLS.      +
C                     IER = 2100   SINGULAR MATRIX IN THE ALGORITHM.           +
C                                                                              +
C   EXTERNAL MODULES:                                                          +
C                                                                              +
C        - SUBROUTINES     BETABOR, RCPROD                                     +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE      +
C          IN DOUBLE PRECISION. THE VECTORS Z AND WK, AND THE MATRIX BETA      +
C          MUST NOT BE MODIFIED BY THE USER BETWEEN TWO CONSECUTIVE CALLS      +
C          OF THE SUBROUTINE.                                                  +
C        - IN THE CALLING PROCEDURE, THE ROWS DIMENSION FOR THE MATRICES A     +
C          AND BETA MUST BE THE SAME.                                          +
C        - WHEN THE SUBROUTINE RETURNS AN IER VALUE OF 2100, THE USER MUST     +
C          GIVE AN ADDITIONAL ROW AND COLUMN OF THE A-ARRAY AND SET IER TO     +
C          ZERO BEFORE THE NEXT CALL.                                          +
C        - IN THE CALLING PROCEDURE THE MINIMAL DIMENSIONS FOR THE             +
C          MATRIX AND VECTOR ARGUMENTS (IR=NUMBER_OF_CALLS) ARE:               +
C           A(IR,IR)                                                           +
C           D(IR)                                                              +
C           Z(IR)                                                              +
C           BETA(IR,2*IR)                                                      +
C           WK(IR)                                                             +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE BLBORD (A, D, IR, Z, BETA, WK, EPS, INIT, IER)
C
C                        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER IER, INIT, IR
      DOUBLE PRECISION EPS
      DOUBLE PRECISION A(IR,1), BETA(IR,1), D(1), Z(1), WK(1)
C
C                        ... SPECIFICATIONS FOR LOCAL VARIABLES AND
C                            CONSTANTS
C
      DOUBLE PRECISION COEFF, MONE, ONE, ZERO
      INTEGER I, IFLAGM, IN, INDS, INK, IP, J
      SAVE IFLAGM, INK, IP
C
C                        ... SPECIFICATIONS FOR PARAMETER CONSTANTS
C
      PARAMETER (ZERO=0.0D0, ONE=1.0D0, MONE=-1.0D0)
C
C                        ... EXECUTABLE STATEMENTS
C
      IF (INIT .EQ. 0) THEN
C
C                        ... FIRST CALL OF THE SUBROUTINE
C
         IER    = 0
         INIT   = 1
         IFLAGM = 0
         INK    = 1
         IF (DABS(A(1,1)) .LT. EPS) THEN
            INK    = INK + 1
            IFLAGM = 1
            IER    = 2100
            RETURN
         END IF
         Z(1)   = D(1) / A(1,1)
         A(1,1) = 1.0D0 / A(1,1)
         RETURN
```

```
      END IF
C
C                        ... NEXT CALLS OF THE SUBROUTINE
C
      IF (IER .NE. 0) THEN
         IER = 200
         RETURN
      END IF
C
C                        ... IFLAGM = 1 IF PREVIOUS A  ARRAY IS SINGULAR
C                                                    1
C
      IF (IFLAGM .EQ. 1) THEN
C
C                        ... COPIES THE NEW A ARRAY IN BETA ARRAY.
C                                             1
C
         DO 20 I = 1, INK
            DO 10 J = 1, INK
               BETA (I,J) = A(I,J)
   10       CONTINUE
   20    CONTINUE
C
C                        ... CONTROLS THE SINGULARITY OF A
C                                                      1
C
         CALL INVERS (BETA, INK, IR, EPS, INDS)
C
C                        ... IF A IS SINGULAR, RETURNS TO CALLING PROGRAM
C                              1
C                            FOR ADDITIONAL ROW AND COLUMN
C
         IF (INDS .EQ. 1) THEN
            INK = INK + 1
            IF (INK .GT. IR) THEN
               IER = 300
            ELSE
               IER = 2100
            END IF
         ELSE
C
C                        ... IF A IS NON-SINGULAR, COMPUTES z  AND RETURNS
C                              1                              1
C                            TO CALLING PROGRAM
C
C                                          -1
C                        ... COPIES BETA = A  ARRAY IN A  ARRAY.
C                                          1           1
C
            DO 40 I = 1, INK
               DO 30 J = 1, INK
                  A(I,J) = BETA (I,J)
   30          CONTINUE
   40       CONTINUE
            IFLAGM = 0
C
C                        ... COMPUTES z
C                                    1
C
            CALL RCPROD (A, 1, 1, IR, D, 1, 1, IR, Z, 1, 1, IR,
     *                   INK, INK, 1, ZERO, ONE, 0, WK)
         END IF
         RETURN
      END IF
C
C                        ... THE A  ARRAY IS NON-SINGULAR
C                              1
C
```

```
         INK = INK + 1
         IF (INK .GT. IR) THEN
            IER  = 300
            RETURN
         END IF
C                        ... IFLAGM = 0 IF beta  IS NON-SINGULAR.
C                                             k
C
      IF (IFLAGM .EQ. 0) IP  = 1
C
C                        ... IFLAGM = 2 IF beta  IS SINGULAR.
C                                             k
C
      IF (IFLAGM .EQ. 2) IP  = IP + 1
C
C                        ... COMPUTES THE NEW beta
C                                               k
C
      IN  = INK - IP
      CALL BETABOR (A, BETA, IN, IP, IR, WK, EPS, INDS)
C
C                        ... IF beta IS SINGULAR, RETURNS TO CALLING PROGRAM
C                                 k
C                            FOR ADDITIONAL ROW AND COLUMN
C
      IF (INDS .EQ. 1) THEN
         IFLAGM = 2
         IER = 2100
         RETURN
      END IF
C
C                                                              -1
C                        ... IF beta IS NON-SINGULAR, COMPUTES A    AND z
C                                 k                            k+1      z+1
C
      IFLAGM = 0
C                                      -1
C                        ... COPIES beta  IN A-ARRAY IN a
C                                       k               k
C
      DO 60 I = 1, IP
         DO 50 J = 1, IP
            A(IN+I,IN+J) = BETA(I,J)
   50    CONTINUE
   60 CONTINUE
C
C                                                 -1
C                        ... COMPUTES THE VALUE OF - A   u   AND
C                                                   k   k
C                            STORES IT IN A-ARRAY IN u
C                                                   k
C
      CALL RCPROD (A, 1, 1, IR, A, 1, IN+1, IR, A, 1, IN+1, IR,
     *             IN, IN, IP, ZERO, MONE, 1, WK)
C
C                        ... COMPUTES THE VALUE OF v  z  AND
C                                                  k  k
C                            STORES IT IN Z-VECTOR (ELEMENTS n +1 TO n + p )
C                                                          k       k   k
C
      CALL RCPROD (A, IN+1, 1, IR, Z, 1, 1, IR, Z, IN+1, 1, IR,
     *             IP, IN, 1, ZERO, ONE, 0, WK)
C
C                        ... COMPUTES THE VALUE OF  f - v  z  AND
C                                                    k   k  k
C                            STORES IT IN Z-VECTOR (ELEMENTS n +1 TO n + p )
C                                                          k       k   k
C
```

```
      DO 70 I = 1, IP
         Z(IN+I) = D(IN+I) - Z(IN+I)
   70 CONTINUE
C
C                                                      -1
C                        ... COMPUTES THE VALUE OF  beta  v  AND
C                                                     k  k
C                            STORES IT IN A-ARRAY IN v
C                                                    k
C
      CALL RCPROD (A, IN+1, IN+1, IR, A, IN+1, 1, IR, A, IN+1, 1, IR,
     *             IP, IP, IN, ZERO, ONE, 1, WK)
C
C                                                     -1    -1
C                        ... COMPUTES THE VALUE OF - beta  v  A  AND
C                                                     k  k  k
C                            STORES IT IN A-ARRAY IN v
C                                                    k
C
      CALL RCPROD (A, IN+1, 1, IR, A, 1, 1, IR, A, IN+1, 1, IR,
     *             IP, IN, IN, ZERO, MONE, 0, WK)
C
C                                                   -1    -1        -1   -1
C                        ... COMPUTES THE VALUE OF A  +  A   u  beta   v A
C                                                   k     k   k     k  k k
C                            AND STORES IT IN A-ARRAY
C
      CALL RCPROD (A, 1, IN+1, IR, A, IN+1, 1, IR, A, 1, 1, IR,
     *             IN, IP, IN, ONE, ONE, 0, WK)
C                                                     -1        -1
C                        ... COMPUTES THE VALUE OF - A   u  beta   AND
C                                                     k   k     k
C                            STORES IT IN A-ARRAY IN u
C                                                    k
C
      IF (IP .EQ. 1) THEN
         DO 80 I = 1, IN
            A(I,IN+1) = A(I,IN+1) * A(IN+1,IN+1)
   80    CONTINUE
C
C                                                      -1        -1
C                        ... COMPUTES THE VALUE OF z - A   u  beta   (f - v z )
C                                                     k   k   k     k    k   k k
C                            AND STORES IT IN Z-VECTOR (ELEMENTS 1 TO n )
C                                                                    k
C
         DO 90 I =1, IN
            Z(I) = Z(I) + A(I,IN+1) * Z(IN+1)
   90    CONTINUE
C                                                     -1
C                        ... COMPUTES THE VALUE OF beta   (f - v z ) AND
C                                                     k    k   k k
C                            STORES IT IN Z-VECTOR (ELEMENTS n + 1 TO n + p )
C                                                            k        k   k
C
         Z(IN+1) = Z(IN+1) * A(IN+1,IN+1)
      ELSE
C                                                     -1        -1
C                        ... COMPUTES THE VALUE OF - A   u  beta   AND
C                                                     k   k     k
C                            STORES IT IN A-ARRAY IN u
C                                                    k
C
         CALL RCPROD (A, 1, IN+1, IR, A, IN+1, IN+1, IR, A, 1, IN+1, IR,
     *                IN, IP, IP, ZERO, ONE, 0, WK)
C
C                                                       -1        -1
C                        ... COMPUTES THE VALUE OF z - A   u  beta   (f - v z )
```

```
C                                                  k   k   k     k    k   k k
C                          AND STORES IT IN Z-VECTOR (ELEMENTS 1 TO n )
C                                                                    k
C
      CALL RCPROD (A, 1, IN+1, IR, Z, IN+1, 1, IR, Z, 1, 1, IR,
     *             IN, IP, 1, ONE, ONE, 0, WK)
C                                                 -1
C                      ... COMPUTES THE VALUE OF beta   (f - v z ) AND
C                                                   k    k   k k
C                          STORES IT IN Z-VECTOR (ELEMENTS n + 1 TO n + p )
C                                                        k        k   k
C
      CALL RCPROD (A, IN+1, IN+1, IR, Z, IN+1, 1, IR, Z, IN+1, 1, IR,
     *             IP, IP, 1, ZERO, ONE, 1, WK)
      END IF
C
C                                                -1
C                      ... SOLUTION z    AND ARRAY A    COMPUTED
C                                  k+1            k+1
C
      RETURN
      END
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                   +
C   SUBROUTINE NAME     - BETABOR                                                   +
C                                                                                   +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                                   +
C   DESCRIPTION:                                                                    +
C                                                                                   +
C          COMPUTES THE ARRAY beta  OF DIMENSIONS p  x p  AND DETERMINES            +
C                                 k                k    k                           +
C          IF IT IS OR NOT A SINGULAR MATRIX.                                       +
C                                                                                   +
C   USAGE:                                                                          +
C          CALL BETABOR (A, BETA, IN, IP, IR, WK, EPS, INDS)                        +
C                                                                                   +
C   ARGUMENTS:                                                                      +
C                                                                                   +
C          A      - INPUT REAL SQUARE MATRIX OF DIMENSION (IR,IR).                  +
C                                                                                   +
C          BETA   - OUTPUT REAL MATRIX OF DIMENSION (IR,2*IR).                      +
C                   IF THE MATRIX beta IS NON SINGULAR (INDS.EQ.0) ITS FIRST        +
C                                     k                          -1                 +
C                   COLUMNS CONTAIN, IN OUTPUT, THE MATRIX beta .                   +
C                                                              k                    +
C                                                                                   +
C          IN     - INPUT INTEGER DIMENSION OF THE SQUARE MATRIX A  .               +
C                                                                 k                 +
C                                                                                   +
C          IP     - INPUT INTEGER DIMENSION p OF THE SQUARE MATRIX beta .           +
C                                             k                          k          +
C                                                                                   +
C          IR     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE                +
C                   DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR                 +
C                   THE MATRICES A AND BETA.                                        +
C                                                                                   +
C          WK     - WORKING REAL VECTOR OF DIMENSION IR.                            +
C                                                                                   +
C          EPS    - INPUT REAL VALUE USED IN TESTS FOR ZERO.                        +
C                   IF DABS(X) .LT. EPS, THEN X IS CONSIDERED TO BE ZERO.           +
C                                                                                   +
C          INDS   - OUTPUT INTEGER VALUE.                                           +
C                   IF INDS = 0 THEN beta  IS NON-SINGULAR.                         +
C                                      k                                            +
C                   IF INDS = 1 THEN beta  IS SINGULAR.                             +
```

```
C                                        k                                     +
C                                                                              +
C   EXTERNAL MODULES:                                                          +
C                                                                              +
C       - SUBROUTINES    INVERS, RCPROD                                        +
C                                                                              +
C   REMARKS:                                                                   +
C                                                                              +
C       - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE       +
C         IN DOUBLE PRECISION.                                                 +
C       - IN THE CALLING PROCEDURE, THE ROWS DIMENSION FOR THE MATRIX A       +
C         AND BETA MUST BE THE SAME.                                           +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE BETABOR (A, BETA, IN, IP, IR, WK, EPS, INDS)
C
C                          ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER IN, INDS, IP, IR
      DOUBLE PRECISION EPS
      DOUBLE PRECISION A(IR,1), BETA(IR,1), WK(1)
C
C                          ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I, J
C
C                          ... EXECUTABLE STATEMENTS
C
      INDS = 0
C                                                    -1
C                          ... COMPUTES THE VALUE OF v  A   AND
C                                                   k  k
C                              STORES IT IN BETA-ARRAY
C
      CALL RCPROD (A, IN+1, 1, IR, A, 1, 1, IR, BETA, 1, 1, IR,
     *             IP, IN, IN, 0.0D0, 1.0D0, 0, WK)
C
C                                                    -1
C                          ... COMPUTES THE VALUE OF (v A  ) u   AND
C                                                     k k    k
C                              STORES IT IN BETA-ARRAY
C
      CALL RCPROD (BETA, 1, 1, IR, A, 1, IN+1, IR, BETA, 1, 1, IR,
     *             IP, IN, IP, 0.0D0, 1.0D0, 0, WK)
C
C                                                                 -1
C                          ... COMPUTES THE VALUE OF beta  = a  - v A  u
C                                                       k    k    k k  k
C
      DO 20 J = 1, IP
         DO 10 I = 1, IP
            BETA(I,J) = A(IN+I,IN+J) - BETA(I,J)
   10    CONTINUE
   20 CONTINUE
C                                                   -1
C                          ... COMPUTES THE INVERSE beta   OR THE SINGULARITY OF
C                              beta .                k
C                                  k
C
      IF (IP .EQ. 1) THEN
         IF (DABS(BETA(1,1)) .LT. EPS) THEN
            INDS = 1
         ELSE
            BETA(1,1) = 1.0D0 / BETA(1,1)
         END IF
      ELSE
         CALL INVERS (BETA, IP, IR, EPS, INDS)
```

```
      END IF
      RETURN
      END
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   SUBROUTINE NAME     - GAUSS                                                +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C           COMPUTES THE SOLUTION OF A SYSTEM OF LINEAR EQUATIONS WITH A AS    +
C           COEFFICIENT MATRIX AND WITH THE VECTORS OF THE CANONICAL BASIS     +
C           AS RIGHT HAND SIDES.                                               +
C           THE METHOD IS GAUSSIAN ELIMINATION.                                +
C           IF THE MATRIX A IS NON-SINGULAR THE SOLUTIONS ARE THE ROWS OF      +
C           THE INVERSE OF THE MATRIX A.                                       +
C           THE COLUMNS N+1,....,2N OF A CONTAIN THE N RIGHT HAND SIDES.       +
C                                                                              +
C   USAGE:                                                                     +
C           CALL GAUSS (A, N, IR, EPS, INDS)                                   +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C           A       - INPUT REAL MATRIX OF DIMENSION (IR,2*IR).                +
C                                                                              +
C           N       - INPUT INTEGER EQUAL TO THE DIMENSION OF THE SYSTEM.      +
C                                                                              +
C           IR      - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE         +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR          +
C                     THE MATRIX A.                                            +
C                                                                              +
C           EPS     - INPUT REAL VALUE USED IN TESTS FOR ZERO.                 +
C                     IF DABS(X) .LT. EPS, THEN X IS CONSIDERED TO BE ZERO.    +
C                                                                              +
C           INDS    - OUTPUT INTEGER VALUE.                                    +
C                     IF INDS = 0 THEN THE MATRIX A IS NON-SINGULAR.           +
C                     IF INDS = 1 THEN THE MATRIX A IS SINGULAR.               +
C                      .                                                       +
C   REMARKS:                                                                   +
C                                                                              +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE      +
C          IN DOUBLE PRECISION.                                                +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE GAUSS (A, N, IR, EPS, INDS)
C
C                     ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER INDS, IR, N
      DOUBLE PRECISION EPS
      DOUBLE PRECISION A(IR,1)
C
C                     ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I, IPIVOT, J, K
      DOUBLE PRECISION TMP, PIVOT
C
C                     ... EXECUTABLE STATEMENTS
C
      INDS = 0
C
C                     ... TRIANGULARIZATION
C
      DO 50 K = 1, N-1
C
C                     ... SEARCH FOR THE PIVOT (PARTIAL PIVOTING)
```

```
C
      PIVOT = 0.0D0
      DO 10 I = K, N
         TMP = DABS(A(I,K))
         IF (TMP .GE. PIVOT) THEN
            IPIVOT = I
            PIVOT  = TMP
         END IF
   10 CONTINUE
C
C                        ... TEST FOR SINGULAR MATRIX
C
      IF (DABS(PIVOT) .LT. EPS) THEN
         INDS = 1
         RETURN
      END IF
C
C                        ... EXCHANGES THE ROW K WITH THE ROW
C                            CONTAINING THE MAXIMUM PIVOT
C
      DO 20 J = K, N+N
         TMP          = A(K,J)
         A(K,J)       = A(IPIVOT,J)
         A(IPIVOT,J) = TMP
   20 CONTINUE
C
C                        ... APPLIES THE ELEMENTS TRANSFORMATION
C
      DO 40 I = K+1, N
         TMP = A(I,K) / A(K,K)
         DO 30 J = K+1, N+N
           A(I,J) = A(I,J) - TMP * A(K,J)
   30    CONTINUE
   40 CONTINUE
   50 CONTINUE
C
C                        ... TEST FOR SINGULAR MATRIX
C                            (PIVOT OF THE LAST ROW)
C
      IF (DABS(A(N,N)) .LT. EPS) THEN
         INDS = 1
         RETURN
      END IF
C
C                        ... SOLVING THE TRIANGULAR SYSTEMS
C                            FOR THE N RIGHT HAND SIDES
C
      DO 80 K = 1, N
         A(N,N+K) = A(N,N+K) / A(N,N)
         DO 70 I = N-1, 1, -1
            A(I,N+K) = A(I,N+K)
            DO 60 J = I+1, N
               A(I,N+K) = A(I,N+K) - A(I,J) * A(J,N+K)
   60       CONTINUE
            A(I,N+K) = A(I,N+K) / A(I,I)
   70    CONTINUE
   80 CONTINUE
      RETURN
      END
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   SUBROUTINE NAME     - INVERS                                               +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C          DETERMINES IF A MATRIX IS SINGULAR OR NOT.                          +
```

```
C           IF THE MATRIX IS NON-SINGULAR THE SUBROUTINE RETURNS THE          +
C           INVERSE MATRIX.                                                   +
C                                                                             +
C   USAGE:                                                                    +
C           CALL INVERS (A, N, IR, EPS, INDS)                                 +
C                                                                             +
C   ARGUMENTS:                                                                +
C                                                                             +
C           A       - INPUT REAL MATRIX OF DIMENSION (IR,2*IR).               +
C                                                                             +
C           N       - INPUT INTEGER INDICATING THE DIMENSION OF THE A-ARRAY.  +
C                                                                             +
C           IR      - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE        +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR         +
C                     THE MATRIX A.                                           +
C                                                                             +
C           EPS     - INPUT REAL VALUE USED IN TESTS FOR ZERO.                +
C                     IF DABS(X) .LT. EPS, THEN X IS CONSIDERED TO BE ZERO.   +
C                                                                             +
C           INDS    - OUTPUT INTEGER VALUE.                                   +
C                     IF INDS = 0 THEN THE MATRIX A IS NON-SINGULAR.          +
C                     IF INDS = 1 THEN THE MATRIX A IS SINGULAR.              +
C                                                                             +
C   EXTERNAL MODULES:                                                         +
C                                                                             +
C       - SUBROUTINE     GAUSS                                                +
C                                                                             +
C   REMARKS:                                                                  +
C                                                                             +
C        - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE     +
C          IN DOUBLE PRECISION.                                               +
C                                                                             +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE INVERS (A, N, IR, EPS, INDS)
C
C                           ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER INDS, IR, N
      DOUBLE PRECISION EPS
      DOUBLE PRECISION A(IR,1)
C
C                           ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I, J
C
C                           ... EXECUTABLE STATEMENTS
C
C                           ... ADDS, TO THE GIVEN MATRIX, N COLUMN VECTORS
C                               WHICH FORM THE CANONICAL BASIS
C
      DO 20 I = 1, N
         DO 10 J = N+1, N+N
            A(I,J) = 0.0D0
   10    CONTINUE
         A(I,N+I) = 1.0D0
   20 CONTINUE
C
C                           ... SOLVES THE SYSTEM WITH GAUSS METHOD. IF THE MATRIX
C                               A IS NON-SINGULAR THEN, IN THE SAME ARRAY A,
C                               (ROWS 1 TO N AND COLUMNS N+1 to 2*N) THERE IS,
C                               IN OUTPUT, THE INVERSE MATRIX. OTHERWISE INDS IS
C                               SET TO 1 IN OUTPUT.
C
      CALL GAUSS (A, N, IR, EPS, INDS)
C
      IF (INDS .EQ. 0) THEN
C
```

```
C                       ... SUPERSEDES THE ORIGINAL MATRIX A WITH THE INVERSE
C                           MATRIX.
C
         DO 40 I = 1, N
            DO 30 J = 1, N
               A(I,J) = A(I,N+J)
   30       CONTINUE
   40    CONTINUE
      END IF
      RETURN
      END
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   SUBROUTINE NAME     - RCPROD                                               +
C                                                                              +
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                              +
C   DESCRIPTION:                                                               +
C                                                                              +
C           COMPUTES THE PRODUCT OF TWO MATRICES.                              +
C                                                                              +
C   USAGE:                                                                     +
C           CALL RCPROD (A, IARST, IACST, IRA, B, IBRST, IBCST, IRB,           +
C                        C, ICRST, ICCST, IRC, NAROW, NACOL, NBCOL,            +
C                        C1COEF, C2COEF, IFLAG, WK)                            +
C                                                                              +
C   ARGUMENTS:                                                                 +
C                                                                              +
C           A       - INPUT REAL MATRIX OF DIMENSION (IRA,IRB).                +
C                                                                              +
C           IARST   - INPUT INDEX OF THE STARTING ROW OF THE MATRIX A.         +
C                                                                              +
C           IACST   - INPUT INDEX OF THE STARTING COLUMN OF THE MATRIX A.      +
C                                                                              +
C           IRA     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE         +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR          +
C                     THE MATRIX A.                                            +
C                                                                              +
C           B       - INPUT REAL MATRIX OF DIMENSION (IRB,*).                  +
C                                                                              +
C           IBRST   - INPUT INDEX OF THE STARTING ROW OF THE MATRIX B.         +
C                                                                              +
C           IBCST   - INPUT INDEX OF THE STARTING COLUMN OF THE MATRIX B.      +
C                                                                              +
C           IRB     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE         +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR          +
C                     THE MATRIX B.                                            +
C                                                                              +
C           C       - OUTPUT REAL PRODUCT MATRIX OF DIMENSION (IRA,*).         +
C                                                                              +
C           ICRST   - INPUT INDEX OF THE STARTING ROW OF THE MATRIX C.         +
C                                                                              +
C           ICCST   - INPUT INDEX OF THE STARTING COLUMN OF THE MATRIX C.      +
C                                                                              +
C           IRC     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE         +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR          +
C                     THE MATRIX C.                                            +
C                                                                              +
C           NAROW   - INPUT NUMBER OF ROWS OF THE MATRIX A TO BE CONSIDERED.   +
C                                                                              +
C           NACOL   - INPUT NUMBER OF COLUMNS OF THE MATRIX A AND NUMBER OF    +
C                     ROWS OF THE MATRIX B TO BE CONSIDERED.                   +
C                                                                              +
C           NBCOL   - INPUT NUMBER OF COLUMNS OF THE MATRIX B TO BE CONSIDERED.+
C                                                                              +
C           C1COEF  - INPUT REAL NUMBER.                                       +
C                     IF C1COEF = 0.0D0 THEN THE PRODUCT MATRIX IS STORED IN   +
C                     THE C-ARRAY.                                             +
```

```
C                    .
C                    IF C1COEF NOT EQUAL TO ZERO, THEN THE PREVIOUS C-ARRAY    +
C                    IS MULTIPLIED BY C1COEF AND ADDED TO THE PRODUCT MATRIX.  +
C                                                                              +
C           C2COEF - INPUT REAL NUMBER.                                        +
C                    THE PRODUCT MATRIX IS ALWAYS MULTYPLIED BY C2COEF. THUS   +
C                    THE VALUE OF C2COEF MUST BE SET TO 1.0D0 IN THE NORMAL    +
C                    USE OF THE SUBROUTINE.                                    +
C                                                                              +
C           IFLAG  - IF IFLAG = 0 THEN THE RESULTANT C-ARRAY IS NOT COINCIDENT +
C                                 WITH THE B-ARRAY, BUT IT MAY COINCIDE WITH   +
C                                 THE A-ARRAY.                                 +
C                    IF IFLAG NOT EQUAL TO ZERO THEN THE RESULTANT C-ARRAY     +
C                                 COINCIDES WITH THE B-ARRAY.                  +
C                                                                              +
C           WK     - REAL WORKING VECTOR. WK HAS TO BE DIMENSIONED AT LEAST    +
C                    TO THE GREATEST DIMENSION BETWEEN A AND B.                +
C                                                                              +
C   EXTERNAL MODULE:                                                           +
C                                                                              +
C       - DOUBLE PRECISION FUNCTION  INNERG                                    +
C                                                                              +
C   REMARKS:                                                                   +
C       - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE       +
C         IN DOUBLE PRECISION.                                                 +
C                                                                              +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE RCPROD (A, IARST, IACST, IRA, B, IBRST, IBCST, IRB,
     *                   C, ICRST, ICCST, IRC, NAROW, NACOL, NBCOL,
     *                   C1COEF, C2COEF, IFLAG, WK)
C
C                        ... SPECIFICATIONS FOR ARGUMENTS
C
      INTEGER IARST, IACST, IRA, IBRST, IBCST, IRB, ICRST, ICCST, IRC,
     *        NAROW, NACOL, NBCOL, IFLAG
      DOUBLE PRECISION C1COEF, C2COEF
      DOUBLE PRECISION A(IRA,1), B(IRB,1), C(IRC,1), WK(1)
C
C                        ... SPECIFICATIONS FOR EXTERNAL FUNCTIONS
C
      DOUBLE PRECISION INNERG
C
C                        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      INTEGER I, J, IASTART, IBSTART, INDCROW, INDCCOL
C
C                        ... EXECUTABLE STATEMENTS
C
      IF (IFLAG .EQ. 0 ) THEN
         DO 30 I = 1, NAROW
            IASTART = IARST + I - 1
            INDCROW = ICRST + I - 1
            DO 10 J = 1, NBCOL
               IBSTART = IBCST + J - 1
               WK(J) = INNERG (A, IASTART, IACST, IRA,
     *                         B, IBRST, IBSTART, IRB, NACOL)
   10       CONTINUE
            DO 20 J = 1, NBCOL
               INDCCOL = ICCST + J - 1
               C(INDCROW, INDCCOL) = C1COEF * C(INDCROW, INDCCOL)
     *                               + C2COEF * WK(J)
   20       CONTINUE
   30    CONTINUE
      ELSE
         DO 60 J = 1, NBCOL
            IBSTART = IBCST + J - 1
            INDCCOL = ICCST + J - 1
            DO 40 I = 1, NAROW
```

```
               IASTART = IARST + I - 1
               WK(I) = INNERG (A, IASTART, IACST, IRA,
     *                         B, IBRST, IBSTART, IRB, NACOL)
   40       CONTINUE
            DO 50 I = 1, NAROW
               INDCROW = ICRST + I - 1
               C(INDCROW, INDCCOL) = C1COEF * C(INDCROW, INDCCOL)
     *                             + C2COEF * WK(I)
   50       CONTINUE
   60    CONTINUE
      END IF
      RETURN
      END

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                               +
C   FUNCTION NAME     - INNERG                                                  +
C                                                                               +
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C                                                                               +
C   DESCRIPTION:                                                                +
C                                                                               +
C           DOUBLE PRECISION FUNCTION:                                          +
C           COMPUTES THE INNER PRODUCT BETWEEN TWO VECTORS WHICH ARE            +
C           A ROW AND A COLUMN OR A PART OF A ROW OR OF A COLUMN OF TWO         +
C           GENERAL ARRAYS.                                                     +
C                                                                               +
C   USAGE:                                                                      +
C           INNERG (A, IAROW, IACOL, IRA, B, IBROW, IBCOL, IRB, IP)             +
C                                                                               +
C   ARGUMENTS:                                                                  +
C                                                                               +
C           A       - INPUT REAL MATRIX WHICH CONTAINS THE ROW VECTOR.          +
C                                                                               +
C           IAROW   - INPUT ROW INDEX OF THE FIRST ELEMENT OF THE ROW VECTOR    +
C                     IN THE MATRIX A.                                          +
C                                                                               +
C           IACOL   - INPUT COLUMN INDEX OF THE FIRST ELEMENT OF THE ROW        +
C                     VECTOR IN THE MATRIX A.                                   +
C                                                                               +
C           IRA     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE          +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR THE       +
C                     MATRIX A.                                                 +
C                                                                               +
C           B       - INPUT REAL MATRIX WHICH CONTAINS THE COLUMN VECTOR.       +
C                                                                               +
C           IBROW   - INPUT ROW INDEX OF THE FIRST ELEMENT OF THE COLUMN        +
C                     VECTOR IN THE MATRIX B.                                   +
C                                                                               +
C           IBCOL   - INPUT COLUMN INDEX OF THE FIRST ELEMENT OF THE COLUMN     +
C                     VECTOR IN THE MATRIX B.                                   +
C                                                                               +
C           IRB     - INPUT ROWS DIMENSION EXACTLY AS SPECIFIED IN THE          +
C                     DIMENSION STATEMENTS IN THE CALLING PROGRAM FOR THE       +
C                     MATRIX B.                                                 +
C                                                                               +
C           IP      - NUMBER OF ELEMENTS TO BE CONSIDERED IN EACH VECTOR        +
C                     STARTING FROM THE FIRST ONE.                              +
C                                                                               +
C   REMARKS:                                                                    +
C                                                                               +
C         - ALL THE REAL ARGUMENTS PASSED FROM THE CALLING PROGRAM MUST BE      +
C           IN DOUBLE PRECISION.                                                +
C         - IN THE CALLING PROCEDURE, IF THE DIMENSIONS OF THE TWO MATRICES     +
C           A AND B ARE A(IRA,ICA) AND B(IRB,ICB) THEN WE MUST HAVE             +
C           (IAROW.LE. IRA).AND.(IP .LE. ICA).AND.(IP .LE. IRB)                 +
C           .AND.(IBCOL .LE.ICB).                                               +
C                                                                               +
```

```
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DOUBLE PRECISION FUNCTION INNERG(A, IAROW, IACOL, IRA,
     *                                 B, IBROW, IBCOL, IRB, IP)
C
C                        ... SPECIFICATIONS FOR ARGUMENTS
C
      DOUBLE PRECISION A(1), B(1)
      INTEGER IAROW, IACOL, IBROW, IBCOL, IRA, IRB, IP
C
C                        ... SPECIFICATIONS FOR LOCAL VARIABLES
C
      DOUBLE PRECISION ACC
      INTEGER I, IASTART, IBSTART
C
C                        ... EXECUTABLE STATEMENTS
C
      IASTART = (IACOL - 1) * IRA + IAROW
      IBSTART = (IBCOL - 1) * IRB + IBROW
      ACC  = 0.0D0
      DO 10 I = 1, IP
         ACC = ACC + A(IASTART+IRA*(I-1)) * B(IBSTART+I-1)
   10 CONTINUE
C
C                        ... SETS THE RESULT VALUE
C
      INNERG = ACC
      RETURN
      END
```

B

Explications sur Mathematica

Cet annexe a été rédigé dans le but de permettre au lecteur qui ne connaît pas *Mathematica* de comprendre entièrement les sections 1.3.2 et 1.3.4.

– Commentaires de la section 1.3.2.Ê

Nous reprenons içi quelques explications de *Mathematica* [13, pages 93,596,644] :

- Une *liste* est un objet très général, qui sert à représenter des collections d'expressions éventuellement de type différent.

  {a,b,c} représente le vecteur (a,b,c).

  {{a,b},{c,d}} représente la matrice $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

  Note Importante : Dans ce travail nous avons supposé que la liste {{a,b},{c,d}} représente la matrice $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$.

- l[[i]] donne la $i^{\text{ème}}$ sous-list de l.

  l[[i,j,...]] donne la part de l correspondant à l[[i]][[j]]....

  Exemple :

  In[1]:=l={{a,b},{c,d}}

  Out[1]={{a,b},{c,d}}

  In[2]:=l[[1]]

  Out[2]={a,b}

  In[3]:=l[[1,2]]

  Out[3]=b

- *Append*[*liste*, *élément*] donne la liste avec l'élément ajouté à la fin.

  AppendTo[*liste*, *élément*] est équivalent à *liste* = *Append*[*liste*, *élément*].

  Exemple :

  In[4]:=v={a,b}

Out[4]={a,b}

In[5]:=Append[v,c]

Out[5]={a,b,c}

In[6]:=v

Out[6]={a,b}

In[7]:=AppendTo[v,c]

Out[7]={a,b,c}

In[8]:=v

Out[8]={a,b,c}

- Les lignes étiquetées par *In*[*n*] sont ce que l'utilisateur saisi et celles étiquetées par *Out*[*n*] sont ce que *Mathematica* écrit comme réponse.

– Commentaires de la section 1.3.4.Ê

Nous reprenons içi quelques explications de *Mathematica* [13, pages 325,330,333,549]. Il y a quatre types de nombres en *Mathematica* :

- **Integer** — entier exact de longueur arbitraire.
- **Rational** — entier/entier dans la forme réduite.

  In[1] := 12344/2222

  Out[1] = $\frac{6172}{1111}$
- **Real** — un nombre réel est identifié par la présence explicite d'un point décimal, et peut avoir n'importe quel nombre de chiffres.

  Mathematica distingue deux types de nombres réels: des nombres réels de basse précision et des nombres réels de haute précision et agit différemment en présence de l'un ou de l'autre type.

  On dit qu'un nombre réel est de basse précision s'il a un nombre de chiffres significatifs non supérieur à la "précision de la machine" (donnée, par exemple, par Precision[1.0]). Dans le cas où ce nombre est inférieur à la précision de la machine, *Mathematica* ajoute des zéros à droite du dernier chiffre significatif jusqu'à atteindre cette précision.

  On dit qu'un nombre réel est de haute précision s'il a un nombre de chiffres significatifs supérieur à la précision de la machine.

  Mathematica suppose toujours que les chiffres qu'on saisit sont corrects.

  In[2] := Precision[1.]

  Out[2] = 19

La précision du Macintosh SE/30 est 19.

In[3] := Precision[1.2345]

Out[3] = 19

1.2345 a la précision de la machine, parce qu'il est interpreté comme 1.234500000000000000.

In[4] := Precision[1.23456789123456789123456789123456789]

Out[4] = 36

C'est un nombre réel de haute précision.

Note:Precision[x] = nombre de chiffres significatives de x.

- **Complex** — complexe de la forme *nombre* + *nombreI*.

  In[5] := 4 + 7/8 *I*

  Out[5] = $4 + \frac{7\,I}{8}$

  In[6] := 4 + 5.6 *I*

  Out[6] = 4 + 5.6 *I*

Il y a un autre type d'objet indivisible qui s'appelle *Symbol*. Un symbole est le plus simple objet *Mathematica* avec un nom. Le nom d'un symbole est constitué par une suite de lettres ou chiffres, commençant obligatoirement par une lettre. *Mathematica* distingue les lettres majuscules des lettres minuscules. La première lettre d'un symbole qui appartient au système est toujours majuscule, par convention.

x, y1, t — symboles définis par l'utilisateur.

Pi, I — symboles appartenant au système.

Les nombres entiers et rationnels sont traités par *Mathematica* exactement. Cependant, les nombres réels sont traités approximativement.

Dans un système symbolique comme *Mathematica*, on peut souvent représenter des nombres réels d'une façon exacte. Par exemple, le symbole *Pi* est une représentation exacte de la constante mathématique π. Si l'on veut connaître une approximation de *Pi*, on est obligé de se servir de la fonction évaluation numérique *N*.

N[x,n] — donne une approximation numérique de x, avec au plus n chiffres significatifs exacts.

In[7] :=N[Pi,6]

Out[7] = 3.14159

In[8] :=N[Pi,30]

Out[8] = 3.14159265358979323846264338328

Une des plus importantes capacités de *Mathematica* est de faire du calcul symbolique aussi bien que du calcul numérique. Dans les deux cas on peut avoir des résultats exacts ou approchés.

- **Calcul numérique exact:**

  In[9] := 1/2 * Sqrt[3] + 123 + 7/2 * Sqrt[3] + 4 + 2/6 * Log[22/2]

  Out[9] = $127 + 4\ \mathrm{Sqrt}[3] + \frac{\mathrm{Log}[11]}{3}$

- **Calcul numérique approché de basse précision:**

  In[10] := 1./2 * Sqrt[3.] + 123. + 7/2 * Sqrt[3.] + 4 + 2/6 * Log[22./2]

  Out[10] = 134.728

  La présence des points décimaux fait en sorte que *Mathematica* donne un résultat approché.

  Le calcul est fait dans la précision de la machine, c'est-à-dire avec dix-neuf chiffres significatifs, et le résultat a lui aussi dix-neuf chiffres significatifs, mais il est écrit, par convention, avec seulement six. Si l'on veut voir plus de chiffres, on est obligé d'utiliser la fonction *N*.

  In[11] := *N*[1./2 * Sqrt[3.] + 123. + 7/2 * Sqrt[3.] + 4 + 2/6 * Log[22./2] , 19]

  Out[11] = 134.7275016545416327

- **Calcul numérique approché de haute précision:**

  Si l'on veut obtenir un résultat avec une précision supérieure à la précision de la machine, on ne doit pas appliquer la fonction *N* à l'expression 1./2 * Sqrt[3.] + 123. + 7/2 * Sqrt[3.] + 4 + 2/6 * Log[22./2], mais à une expression qui a elle même une précision supérieure ou égale à la précision désirée.

  In[12] := *N*[1./2 * Sqrt[3.] + 123. + 7/2 * Sqrt[3.] + 4 + 2/6 * Log[22./2] , 45]

  Out[12] = 134.7275016545416327

  In[13] := *N*[*N*[1/2 * Sqrt[3] + 123 + 7/2 * Sqrt[3] + 4 + 2/6 * Log[22/2], 50], 45]

  Out[13] = 134.727501654541632688797099892011865901045123

  Nous pouvons obtenir un résultat avec une précision arbitraire si nous appliquons la fonction *N* à l'expression exacte:

  In[14] := *N*[1/2 * Sqrt[3] + 123 + 7/2 * Sqrt[3] + 4 + 2/6 * Log[22/2] ,100]

  Out[14] = 134.727501654541632688797099892011865901045123
  29988732823729608382085125660893961238601626964969653 92

Les mêmes commentaires sont valables pour les différents types de calcul symbolique:

- **Calcul symbolique exact:**

  In[15] := 2/3 + 1/3 * *x* − 1/2 * *x* + 123/4 * *x* ∧ 2 + *x* ∧ 2

  Out[15] = $\frac{2}{3} - \frac{x}{6} + \frac{127\,x^2}{4}$

- **Calcul symbolique approché de basse précision:**

  In[16] := 2./3 + 1./3 * *x* − 1./2 * *x* + 123./4 * *x* ∧ 2 + *x* ∧ 2

  Out[16] = 0.666667 − 0.166667 x + 31.75 x^2

  In[17] := *N*[2./3 + 1./3 * *x* − 1./2 * *x* + 123./4 * *x* ∧ 2 + *x* ∧ 2, 19]

  Out[17] = 0.6666666666666666667 − 0.1666666666666666667 x + 31.75 x^2

- **Calcul symbolique approché de haute précision:**

$In[18] := N[\ N[\ 2/3 + 1/3 * x - 1/2 * x + 123/4 * x \wedge 2 + x \wedge 2\ ,\ 50\ ]\ ,\ 45\ ]$

$Out[18] = 0.66666666666666666666666666666666666666666666667 -$
$0.16666666666666666666666666666666666666666666667\ x + 31.75\ x^2$

$In[19] := N[\ 2/3 + 1/3 * x - 1/2 * x + 123/4 * x \wedge 2 + x \wedge 2\ ,\ 100\ ]$

$Out[19] = 0.66666666666666666666666666666666666666666666666$
$66666666666666666666666666666666666666666666666667 -$
$0.16666666666666666666666666666666666666666666666666$
$66666666666666666666666666666666666666666666667\ x + 31.75\ x^2$

Quand on fait du calcul de basse précision, *Mathematica* effectue tous les calculs avec une précision fixe (la précision de la machine) et donne les résultats sans s'occuper de leur précision réelle. Quand on fait du calcul de haute précision *Mathematica* prend en considération la précision des éléments qui constituent l'expression à évaluer et donne le résultat avec la précision qui peut être justifiée.

$In[20] := N[\ 1.11111111111111111 - 1.11111111111111000\ ,\ 19\ ]$
$Out[20] = 1.11022302462515654\ 10^{-16}$
Mathematica fait du calcul de basse précision. Les derniers seize chiffres du résultat n'ont aucune signification.

$In[21] := N[\ 1.111111111111111111111 - 1.111111111111111111000\ ,\ 24\ ]$
$Out[21] = 1.1\ 10^{-21}$
Mathematica fait du calcul de haute précision, et montre seulement les chiffres du résultat que ce language considère significatifs.

C

GPADETYPE.M, BE.M, BIF.M et MATHEMATICA SESSION.M

```
(*_____________________________________________________________*)
(*_____________________ Package GPADETYPE.M _____________________*)
(*_____________________________________________________________*)

BeginPackage["GPADETYPE`","BE`","BIF`"]


GPADETYPE::usage="is a package that contains the
definition of the function tablegpt.
It calls the packages BE and BIF."

tablegpt::usage="
tablegpt[ k , t , toption  , coption  ,  psoption ,
                resultprec , preoption , preprec  ,
                cprec , aroption , compoption , norm ]
calculates the (k+1) first generalized Pade-type
approximants in the point t .
The argument t  can be a symbolic expression
(toption  == \"VAR\") , or a numerical expression
(toption  == \"SCL\") .  In the first case we obtain symbolic
results,  in the second case we obtain numerical results.
If coption  == \"EC\" ,  we do exact calculus.
If coption  == \"AC\" ,  we do approximated calculus with
precision cprec. This means that we use the function
N[- , cprec]  to do all the intermediate computations.
If cprec < = \"machine precision\" , we do  low precision
computations.  If cprec > \"machine precision\" , we do high
precision computations.
If coption == \"AC\", the numerical results or the numerical
part of the symbolic results are printed with resultprec
decimal digits of precision.
The default values of cprec and resultprec  are the machine
precision ( given for example by  Precision[1.0] ).
If resultprec or cprec are greather than  Precision[1.0],
we must have  resultprec  < = cprec .
If psoption  == 1, the partial sums of the series are also
calculated according to the values of toption  and coption.
If psoption != 1, partial sums are not calculated. The default
value of psoption  is 0.
We remark that it is interesting to calculate the partial sums
of the series especially when toption  == \"SCL\", that is when
we get numerical results. We compare the sequence of
generalized Pade-type approximants with the sequence of
partial sums of the series.
If toption == \"SCL\" and preoption == 1, the relative errors
of the approximants with respect to the series are calculated
with precision cprec, and are written with preprec decimal
digits of precision. Futhermore, if psoption  == 1, the relative
errors of the partial sums with respect to the series are
calculated with precision cprec, and are written with preprec
decimal digits of precision.
If toption  == \"SCL\" and preoption  == 2, the number of the
exact digits of the approximants in comparison with the series
are calculated with precision cprec, and are written with
preprec decimal digits of precision. Futhermore, if
psoption == 1, the number of the exact digits of the partial
sums in comparison with the series are calculated with precision
cprec,  and are written with preprec decimal digits of precision.
The default value of preprec is 2.
If compoption == 0 , we do real calculus. If compoption == 1,
we do complex calculus. The default value of compoption is
equal to 0.
If compoption == 1, toption == \"SCL\" and preoption  == 1, the
relative errors of the real and imaginary parts of the
approximants, with respect to the real and imaginary parts of
the series, are calculated with precision cprec and are written
with preprec decimal digits of precision. If psoption == 1,
we do the same for the partial sums.
```

```
If compoption == 1, toption == \"SCL\" and preoption  == 2, we
calculate the number of exact digits instead of the relative
errors.
If compoption == 1 and toption == \"SCL\",  we calculate also
the p-norm ( p >= 1) of the vector of relative errors
( if preoption  == 1 ), or the p-norm ( p >= 1) of the vector of
the number of exact digits  ( if preoption  == 2 ).
The value of p is given by the argument norm.
If norm == -1, we calculate the infinity norm of the
vector of relative errors ( if preoption  == 1 ), or the
component of minimum absolute value of the vector of the
number of exact digits  ( if preoption  == 2 ).
The default value of norm is equal to -1.
Norms are calculated with precision cprec and are written
with preprec decimal digits of precision.
If toption  == \"VAR\" or preoption  is different from 1 and 2 ,
neither the relative errors nor the number of exact digits are
calculated.
The default value of preoption  is 0.
If toption  == \"SCL\" and coption  == \"EC\", we obtain
the exact numerical expressions of the approximants in the
point t . Futhermore, if psoption  == 1, we obtain also the
exact numerical expressions of the partial sums in the point
t. If preoption  == 1, relative errors of these exact expressions
are calculated with precision cprec .
If preoption  == 2, the number of exact digits are calculated
with precision cprec . We should take a sufficiently big value to
cprec  in order that the errors or the number of exact digits
could be correctly calculated. Futhermore, if aroption  == 1,
the exact expressions just cited are written with resultprec
decimal digits of precision. If aroption  != 1 , these values are
not written. The default value of aroption  is 0.
The value of the series in t  is written with resultprec  decimal
digits of precision, if toption  == \"SCL\",  coption == \"EC\"
and aroption == 1; or if toption  == \"SCL\" and
coption == \"AC\".
We must give the first four arguments of tablegpt ,
however the last eight ones can be omitted taking their default
values.

Reference:  Z. DA ROCHA.
                    Implementention of the recurrence
                    relations of biorthogonality.
                    In C. Brezinski, P. Gonzalez-Vera and
                    N. Hayek-Calil, eds., Extrapolation and
                    Rational Approximation. Tenerife 1992,
                    volume 3 of Numerical Algorithms, pp.
                    173-183, Basel 1992. J. C. Baltzer."

Begin["`Private`"]

(*_____
_____ERROR MESSAGES OF TABLEGPT_____
_____*)

tablegpt::badarg1=
        "wrong value to the first argument
                k  must be a non negative integer
                try again"

tablegpt::badarg3=
        "wrong value to the third argument
                toption  must be equal to \"SCL\" or \"VAR\"
                try again"

tablegpt::badarg4=
        "wrong value to the fourth argument
                coption  must be equal to \"AC\" or \"EC\"
```

```
                try again"

tablegpt::badarg6=
        "wrong value to the sixth argument
                resultprec  must be a non negative integer
                try again"

tablegpt::badarg8=
        "wrong value to the eighth argument
                preprec  must be a non negative integer
                try again"

tablegpt::badarg9=
        "wrong value to the nineth argument
                cprec  must be a non negative integer
                try again"

tablegpt::badarg69=
      "wrong values to the sixth and/or the
             nineth arguments.
             We must have resultprec < = cprec  ,
             if at least one of these arguments are greather
             than the machine precision.
             Try again."

tablegpt::badarg12=
        "wrong value to the twelfth argument
                norm  must be equal to -1 or
                greater than or equal to 1
                try again"

(*_____________________________________________________________________
_____________________DEFINITIONS OF THE PRIVATE FUNCTIONS_____________
_____________________________OF TABLEGPT_____________________________
_____________________________________________________________________*)

functionalc[poly_,var_]:=
Block[{n},
         Sum[ Coefficient[poly,var,n]*cm[[n+1]] ,
            {n,0,Exponent[ poly ,var ]}
                ]/; PolynomialQ[poly,var]
         ] ;


(*_____________________________________________________________________
_____________________________DEFINITION OF TABLEGPT__________________
_____________________________________________________________________*)

tablegpt[ k_ , t_ , toption_ , coption_ ,
                psoption_:0 ,
                resultprec_:(Precision[1.0]) ,
                preoption_:0 , preprec_:2 ,
                cprec_:(Precision[1.0]) ,
                aroption_:0 ,
                compoption_:0 , norm_:-1 ]:=

Block[ { cm, wn1, wn2, wf1, wf2, w3, w1, wc, wx, wf,
             sert, xarray, larray, fps1, fps2, nps1, nps2,
             regpt1, regpt2, regpt, reps1, reps2, reps ,
             edgpt1, edgpt2, edgpt, edps1, edps2, edps ,
             n, m, i } ,
 (*_____________________________________________________________________
 ____________________________CONTROL OF ARGUMENTS_____________________
 _____________________________________________________________________*)

 If [ Or[ Not[ IntegerQ[k] ] ,  k < 0 ] ,
      Return[ Message[tablegpt::badarg1] ]
     ];
```

```
 If [ And [ toption != "SCL" , toption != "VAR" ] ,
      Return[ Message[tablegpt::badarg3] ]
    ];

 If[ And [ coption != "EC",  coption != "AC" ] ,
      Return[ Message[tablegpt::badarg4] ]
    ] ;

If [ Or[ Not[ IntegerQ[ resultprec ] ] ,  resultprec < 0 ] ,
      Return[ Message[tablegpt::badarg6] ]
    ] ;

If [ Or[ Not[ IntegerQ[ preprec ] ] ,  preprec < 0 ] ,
      Return[ Message[tablegpt::badarg8] ]
    ] ;

If [ Or[ Not[ IntegerQ[ cprec ] ] ,  cprec < 0 ] ,
      Return[ Message[tablegpt::badarg9] ]
    ];

If[ And[ resultprec  >  cprec ,
             Or[ resultprec > (Precision[1.0]) ,
                   cprec > (Precision[1.0])
                 ]
           ] ,
         Return [ Message [tablegpt::badarg69 ] ]
    ];

 If [ And[ norm != -1 ,  norm < 1 ]  ,
       Return[ Message[tablegpt::badarg12] ]
    ] ;

(*_____________________________________________________________________
 ____________________ computation of the first approximant ___________
_____________________________________________________________________
_____________________________________________________________________
_____________________ computation of c(X[0,0,0,x]) ___________________
_____________________ computation of L[0,0,0,g[x,t],x] _______________
_____________________________________________________________________*)
If[ coption == "AC"  ,

     cm={ N[ c[0] , cprec ] };
     wx={ N[ u[0,x] , cprec ] };
     wf={ N[ f[0,t] , cprec ] };
     wc={ N[ functionalc[ wx[[1]] , x ], cprec ] };
     wl={ N[ initialf[ 0, wx[[1]], x ] , cprec ] };
     larray={ { N[ (initialf[ 0, g[x,t], x] / wl[[1]] ), cprec ] } }

 ,  (*coption == "EC" *)

     cm={ c[0] };
     wx={ u[0,x] };
     wf={ f[0,t] };
     wc={ functionalc[ wx[[1]] , x ] };
     wl={ initialf[ 0, wx[[1]], x ] };
     larray={ {  initialf[ 0, g[x,t], x] / wl[[1]]  } }

    ] ; (* end of If *)

    xarray={ Append[ wc, wl[[1]]  ] };
(*_____________________________________________________________________
_____________________ xarray[[1,1]]=c(X[0,0,0,x])_____________________
_____________________larray[[1,1]]=L[0,0,0,g[x,t],x]___________________
_____________________________________________________________________
_____________________printing of the first approximant_________________
_____________________________________________________________________*)
```

```
If[  toption == "SCL" ,

    If[  coption == "EC" ,

         wf1= xarray[[1,1]]*larray[[1,1]] ;
         Print[ "fgpt[", 0, "]=", Expand[ wf1] ] ;
         wn1=wf1 ;
         If[  psoption == 1 ,
               fps1= cm[[1]]*wf[[1]] ;
               Print[ "fps[",0,"]=",  fps1 ] ;
               nps1=fps1
             ] ; (* end of If[ psoption ] *)
         If[  aroption == 1 ,
                sert= N[ series[t] , resultprec ] ;
                Print[ "f[", t,"]=", sert ] ;
                Print[ "ngpt[",0,"]=", N[ wn1 , resultprec ] ] ;
                If[ psoption == 1 ,
                     Print[ "nps[",0,"]=",N[ nps1 , resultprec ] ]
                   ] (* end of If[ psoption ] *)
             ] (* end of If[ aroption ] *)

       , (* coption == "AC" *)

         sert= N[ series[t] , resultprec ] ;
         Print[ "f[", t,"]=", sert ] ;
         wn1=N[ xarray[[1,1]]*larray[[1,1]] , cprec ] ;
         Print[ "ngpt[",0,"]=", N[ wn1 , resultprec ] ] ;
         If[ psoption == 1 ,
              nps1=N[ cm[[1]]*wf[[1]] , cprec ] ;
              Print[ "nps[",0,"]=", N[ nps1 , resultprec ] ]
             ] (* end of If[ psoption ] *)

      ] ; (* end of If [ coption ]*)

      If[ preoption == 1 ,

           If[ compoption == 0 ,

                Print[ "regpt[", 0, "]=",
                          N[ N[
                               Abs[ ( wn1-series[t] ) / series[t] ]
                              , cprec ]
                            , preprec ]
                     ]  (* end of Print *)

              ] ; (* end of If[ compoption == 0 ] *)

            If[ compoption == 1 ,

                regpt1=N[ Abs[ ( Re[ wn1]-Re[ series[t] ] ) /
                                          Re[ series[t] ]
                             ] ,
                        cprec ] ;
                regpt2=N[ Abs[ ( Im[wn1] - Im[ series[t] ] ) /
                                           Im[ series[t] ]
                             ] ,
                        cprec ] ;

                If[ norm == -1,
                     regpt=N[ Max[ regpt1, regpt2 ] , cprec ]
                     ,
                     regpt=N[ ( regpt1^norm +
                                regpt2^norm )^(1/norm) , cprec ]
                   ] ; (* end of If[ norm ] *)

                Print["regpt[", 0, "]=[", N[ regpt1, preprec ] ,
                      "," , N[ regpt2, preprec ] , "] , norm regpt[" ,
                      0 , "]=", N[ regpt, preprec ]
```

```
                    ] (* end of Print *)

               ] ; (* end of If[ compoption == 1 ] *)

          If[ psoption == 1 ,

               If[ compoption == 0 ,

                    Print[ "reps[", 0, "]=",
                              N[ N[
                                   Abs[ ( nps1-series[t] ) / series[t] ]
                                 , cprec ]
                               , preprec ]
                            ] (* end of Print *)

                    ] ; (* end of If[ compoption == 0 ] *)

                 If[ compoption == 1 ,

                      reps1=N[ Abs[ ( Re[ nps1 ]-Re[ series[t] ]  ) /
                                                  Re[ series[t] ]
                                  ] ,
                               cprec ] ;
                      reps2=N[ Abs[ ( Im[ nps1 ] - Im[ series[t] ] ) /
                                                   Im[ series[t] ]
                                  ] ,
                               cprec ] ;

                       If[ norm == -1,
                             reps=N[ Max[ reps1, reps2 ] , cprec ]
                            ,
                             reps=N[ ( reps1^norm +
                                       reps2^norm )^(1/norm) , cprec ]
                         ] ; (* end of If[ norm ] *)

                       Print["reps[", 0, "]=[", N[ reps1, preprec ] ,
                             "," , N[ reps2, preprec ] , "] , norm reps[" ,
                             0 , "]=", N[ reps, preprec ]
                           ] (* end of Print *)

                    ] (* end of If[ compoption == 1 ] *)

               ] (* end of If[ psoption ] *)

     ] ; (* end If[ preoption==1 ] *)

     If[ preoption == 2 ,

          If[ compoption == 0 ,

               Print[ "edgpt[", 0, "]=",
                         N[ N[
                              -Log[ 10, Abs[ wn1-series[t] ] ]
                            , cprec ]
                          , preprec ]
                       ]  (* end of Print *)

               ] ; (* end of If[ compoption == 0 ]*)

             If[ compoption == 1 ,

                  edgpt1= N[ -Log[10,
                                  Abs[ Re[ wn1 ] - Re[ series[t] ] ]
                                    ],
                             cprec ] ;
                  edgpt2= N[ -Log[10,
                                  Abs[ Im[ wn1 ] - Im[ series[t] ] ]
                                ],
```

```
                        cprec ] ;

                If[ norm == -1,
                     edgpt=N[ Min[ Abs[edgpt1], Abs[edgpt2]  ] ,
                             cprec ]
                     ,
                     edgpt=N[ ( Abs[edgpt1]^norm +
                               Abs[edgpt2]^norm )^(1/norm) ,
                             cprec ]
                    ] ; (* end of If[ norm ] *)

                  Print["edgpt[", 0, "]=[", N[ edgpt1, preprec ] ,
                        "," , N[ edgpt2, preprec ] , "] , norm edgpt[" ,
                        0 , "]=" ,N[ edgpt, preprec ]
                        ] (* end of Print *)

             ] ; (* end of If[ compoption == 1 ] *)

        If[ psoption == 1 ,

             If[ compoption == 0 ,

                  Print[ "edps[", 0, "]=",
                             N[ N[
                                  -Log[ 10, Abs[ nps1-series[t] ] ]
                                 , cprec ]
                               , preprec ]
                        ] (* end of Print *)

                ] ; (* end of If[ compoption == 0 ] *)

                If[ compoption == 1 ,

                     edps1=N[ -Log[ 10,
                                    Abs[ Re[ nps1 ]-Re[ series[t] ] ]
                                       ] ,
                             cprec ] ;
                     edps2=N[ -Log[ 10,
                                    Abs[ Im[ nps1 ] - Im[ series[t] ] ]
                                       ] ,
                             cprec ] ;

                     If[ norm == -1,
                         edps=N[ Min[ Abs[edps1], Abs[edps2] ] , cprec ]
                          ,
                         edps=N[ ( Abs[edps1]^norm +
                                   Abs[edps2]^norm )^(1/norm) , cprec ]
                        ] ; (* end of If[ norm ] *)

                     Print["edps[", 0, "]=[", N[ edps1, preprec ] ,
                           "," , N[ edps2, preprec ] , "] , norm edps[" ,
                           0, "]=" , N[ edps, preprec ]
                           ] (* end of Print *)

                   ]  (* end of If[ compoption == 1] *)

              ]  (* end of If[ psoption ] *)

      ] (* end of If[ preoption==2 ] *)

     , (* toption == "VAR" *)

       If[ coption == "EC",

            wf1= xarray[[1,1]]*larray[[1,1]]  ;
            Print[ "fgpt[", 0, "]=", Together[ wf1]  ] ;
            If[ psoption == 1 ,
                 fps1 = cm[[1]]*wf[[1]] ;
```

```
                Print[ "fps[",0,"]=",  fps1 ]
               ] (* end of If[ psoption ] *)

          , (* coption == "AC" *)

            wn1=N[ xarray[[1,1]]*larray[[1,1]]
                        , cprec ] ;
            Print[ "ngpt[",0,"]=",N[ Together[wn1], resultprec ]  ] ;
            If[ psoption == 1 ,
                nps1 = N[ cm[[1]]*wf[[1]] , cprec ] ;
                Print[ "nps[",0,"]=", N[ nps1 , resultprec ] ]
                ] (* end of If[ psoption ] *)

           ]   (* end of If [ coption ] *)

    ]; (* end of If [ toption ] *)

    Print[" "];

(*_____________________________________________________________________*)
Do[(*{i,2,k+1}*)
(*_____________________________________________________________________
____________________computation of the i-th approximant_________________
_____________________________________________________________________*)
 If[ coption == "AC" ,

      AppendTo[ cm, N[ c[i-1] , cprec ] ];
      AppendTo[ wx, N[ u[i-1,x]  , cprec ] ];
      AppendTo[ wf, N[ f[i-1,t]  , cprec ] ];
      AppendTo[ wc, N[ functionalc[ wx[[i]], x], cprec] ];
      AppendTo[ wl, N[ initialf[ i-1, wx[[1]], x], cprec] ]

     , (* coption == "EC" *)

     AppendTo[ cm, c[i-1] ];
     AppendTo[ wx, u[i-1,x] ];
     AppendTo[ wf, f[i-1,t] ];
     AppendTo[ wc, functionalc[ wx[[i]], x]  ];
     AppendTo[ wl, initialf[ i-1, wx[[1]], x]  ]

    ]; (* end of If *)
(*_____________________________________________________________________
____________________ computation of c(X[0,0,i-1,x]) ___________________
____________________ computation of L[0,0,i-1,g[x,t],x] ________________
_____________________________________________________________________
____________________ computation of the new elements of the ___________
_____________________________ arrays X1 and L1 ______________________
_____________________________(  first part )_____________________________
_____________________simultaneous computation_____________________
____________________of the new column of the array X and_____________
_____________________ the new row of the array L ____________________
_____________________________________________________________________*)
  AppendTo[ xarray , { wc[[i]] } ];
  Do[ (* {n,2,i} *)
     If[ coption == "AC"  ,

            w3=N[ initialf[ n-2, wx[[i]], x ] , cprec ];
            If[ n == 2 ,
               xarray[[ i-1, 2]]= N[ w3/ xarray[[i-1,2]] , cprec ]
               ] ; (* end of If *)
             AppendTo[ xarray[[i]] , w3 ] ;
             AppendTo[ larray[[n-1]], N[ w3/ wl[[n-1]] , cprec ] ]

        , (* coption == "EC" *)

            w3=initialf[ n-2, wx[[i]], x ] ;
            If[ n == 2 ,
               xarray[[i-1, 2]]= w3/ xarray[[i-1,2]]
```

```
                ] ; (* end of If *)
               AppendTo[ xarray[[i]] , w3 ] ;
               AppendTo[ larray[[n-1]], w3/ wl[[n-1]]  ]

          ] (* end of If *)

     ,{n,2,i}
    ];(*end of Do*)

(* ____________________________________________________________________
_____________________________( second part )_____________________________
_____________________simultaneous computation_____________________
_____________________ of the new column of the array L and _____________________
_____________________ the new row of the array X _____________________
____________________________________________________________________*)
 If[ coption == "AC"  ,
      AppendTo[ larray ,
              { N[initialf[ i-1, g[x,t], x ]/ wl[[i]] , cprec ] }
                ]
    , (* coption == "EC" *)
      AppendTo[ larray , { initialf[ i-1, g[x,t], x ]/ wl[[i]] } ]
    ]; (* end of If *)

   AppendTo[ xarray[[1]], wl[[i]] ];

   Do[ (* {n,2,i} *)
       If[ coption == "AC"  ,

            w4= N[initialf[ i-1, wx[[n]], x ] , cprec ];
            AppendTo[ larray[[i]] , N[ w4/ wl[[i]] , cprec ] ];
            If[ n==2 ,
                 larray[[i-1,2]]=N[ larray[[i,2]]-larray[[i-1,2]] , cprec ]
               ]; (* end of If *)
             AppendTo[ xarray[[n]], w4 ]

            , (* coption == "EC" *)

            w4=initialf[ i-1, wx[[n]], x ] ;
            AppendTo[ larray[[i]] , w4/ wl[[i]] ] ;
            If[ n==2 ,
                 larray[[i-1,2]]=larray[[i,2]]-larray[[i-1,2]]
               ]; (* end of If *)
             AppendTo[ xarray[[n]], w4 ]

            ]; (* end of If *)

          ,{n,2,i}
         ]; (* end of Do *)

(*____________________________________________________________________*)
Do[(*{m,2,i}*)
(*____________________________________________________________________
_____________________ computation of the new elements of the _____________________
_____________________ arrays Xm and Lm  (m >= 2) _____________________
____________________________________________________________________
_____________________________ (first part ) _____________________________
_____________________computation of the new elements_____________________
_____________________ placed at a same row _____________________
_____________________ of the arrays X and L _____________________
____________________________________________________________________*)
  Block[{n1},
   Do[(* {n,1,i-m} *)
       n1=n+1;
       If[ coption == "AC"  ,

            xarray[[n,i]]= N[ xarray[[n1,i]]-
                                    xarray[[n,m]]*
                                    xarray[[n,i]]
```

```
                                        , cprec ] ;
          larray[[n,i]]=N[ ( larray[[n1,i]]-
                                          larray[[n,i]] ) /
                                          larray[[n,m]]
                                       , cprec ]

        , (* coption == "EC" *)

         xarray[[n,i]]= xarray[[n1,i]]-
                                    xarray[[n,m]]*
                                    xarray[[n,i]] ;
         larray[[n,i]]= ( larray[[n1,i]]-
                                     larray[[n,i]] ) /
                                     larray[[n,m]]

       ] (* end of If *)

     ,{n,1,i-m}
    ] (* end of Do *)

  ];(* end of Block *)

(*_____________________________________________________________________
_____________________ (second part ) _____________________
_____________________computation of the new elements_____________________
_____________________placed at a same column_____________________
_____________________ of the arrays X and L _____________________
_____________________________________________________________________*)

Block[{n1,n2,n3,m1},

      n1=i-m+1; n2=n1+1; n3=n1-1; m1=m+1;
      If[ coption  == "AC" ,

           xarray[[n1,1]]=N[ xarray[[n2,1]]-
                                        xarray[[n1,m]]*
                                        xarray[[n1,1]]
                                      , cprec ];
           larray[[n1,1]]=N[( larray[[n2,1]]-
                                       larray[[n1,1]] )/
                                       larray[[n1,m]]
                                     , cprec ]

        , (* coption == "EC" *)

          xarray[[n1,1]]=xarray[[n2,1]]-
                                    xarray[[n1,m]]*
                                    xarray[[n1,1]] ;
          larray[[n1,1]]=( larray[[n2,1]]-
                                     larray[[n1,1]] )/
                                     larray[[n1,m]]

          ] ;(* end of If *)
       Do[ (*{n,m1,i}*)
            If[ coption == "AC"  ,

                 xarray[[n1,n]]=N[  xarray[[n2,n]]-
                                              xarray[[n1,m]]*
                                              xarray[[n1,n]]
                                            , cprec ];
                larray[[n1,n]]=N[( larray[[n2,n]]-
                                             larray[[n1,n]] )/
                                             larray[[n1,m]]
                                          , cprec ]

              ,  (* coption == "EC" *)

               xarray[[n1,n]]=xarray[[n2,n]]-
```

```
                                        xarray[[n1,m]]*
                                        xarray[[n1,n]];
                larray[[n1,n]]=( larray[[n2,n]]-
                                        larray[[n1,n]] )/
                                        larray[[n1,m]]

              ]; (* end of If *)

          If[ n == m1 ,

               If[ coption == "AC"  ,

                    xarray[[n3,m1]]=N[ xarray[[n1,m1]] /
                                                xarray[[n3,m1]]
                                              , cprec ];
                    larray[[n3,m1]]= N[ larray[[n1,m1]]-
                                                larray[[n3,m1]]
                                              , cprec ]

                 ,   (* coption == "EC" *)

                   xarray[[n3,m1]]=xarray[[n1,m1]] /
                                              xarray[[n3,m1]];
                   larray[[n3,m1]]= larray[[n1,m1]]-
                                              larray[[n3,m1]]

                  ] (* end of If [ coption ] *)

             ]  (*end of If*)

        ,{n,m1,i}
       ](* end of Do *)

    ] (*end of Block*)

,{m,2,i}
]; (*end of Do *)
(*_____________________________________________________________________
_____________end of the computation of c(X[0,0,i-1,x]) and _____________
_____________________________L[0,0,i-1,g[x,t],x]_____________________________
_____________________xarray[[1,1]]=c(X[0,0,i-1,x])_____________________
_____________________larray[[1,1]]=L[0,0,i-1,g[x,t],x]_____________________
_____________________________________________________________________
_____________________printing of the i-th approximant_____________________
_____________________________________________________________________*)

If[ toption == "SCL"  ,

     If[  coption == "EC" ,

          wf2= wf1 + xarray[[1,1]]*larray[[1,1]] ;
          Print["fgpt[",i-1,"]=", Expand[ wf2 ] ] ;
          wn2= wf2 ;
          If[ psoption == 1 ,
               fps2 = fps1 + cm[[i]]*wf[[i]] ;
               Print[ "fps[", i-1, "]=",  fps2 ] ;
               nps2=fps2
             ] ;
          If[ aroption == 1 ,
               Print[ "f[", t,"]=", sert ] ;
               Print[ "ngpt[", i-1, "]=", N[ wn2 , resultprec ] ] ;
               If[ psoption == 1,
                    Print[ "nps[",i-1,"]=", N[ nps2 , resultprec ] ]
                 ] (* end of If[ psoption ] *)
             ] (* end of If[ aroption ] *)

        , (* coption == "AC" *)
```

```
        Print[ "f[", t,"]=", sert ] ;
        wn2=N[ wn1 + xarray[[1,1]]*larray[[1,1]] , cprec ] ;
        Print[ "ngpt[", i-1, "]=", N[ wn2 , resultprec ] ] ;
        If[ psoption == 1 ,
            nps2=N[ nps1 + cm[[i]]*wf[[i]] , cprec ] ;
            Print[ "nps[", i-1, "]=", N[ nps2 , resultprec ] ]
           ] (* end of If[ psoption ] *)

      ]; (* end of If[ coption ] *)

    If[ preoption == 1 ,

        If[ compoption == 0 ,

            Print[ "regpt[", i-1, "]=",
                   N[ N[
                        Abs[ (wn2-series[t] ) / series[t] ]
                        , cprec  ]
                      , preprec ]
                ]  ;(* end of Print *)

          ] ; (* end of If [ compoption == 0 ] *)

        If[ compoption == 1 ,

            regpt1=N[ Abs[ ( Re[ wn2 ]-Re[ series[t] ] ) /
                                      Re[ series[t] ]
                         ] ,
                    cprec ] ;
            regpt2=N[ Abs[ ( Im[ wn2 ] - Im[ series[t] ] ) /
                                       Im[ series[t] ]
                         ] ,
                    cprec ] ;

            If[ norm == -1,
                regpt=N[ Max[ regpt1, regpt2 ] , cprec ]
                ,
                regpt=N[ ( regpt1^norm +
                           regpt2^norm )^(1/norm) , cprec ]
              ] ; (* end of If[ norm ] *)

            Print["regpt[", i-1, "]=[", N[ regpt1, preprec ] ,
                  "," , N[ regpt2, preprec ] , "] , norm regpt[" ,
                  i-1 , "]=" , N[ regpt, preprec ]
                ] (* end of Print *)

          ] ; (* end of If[ compoption == 1 ] *)

        If[ psoption == 1 ,

            If [ compoption == 0 ,

                 Print[ "reps[", i-1, "]=",
                        N[ N[
                             Abs[ (nps2-series[t] ) / series[t] ]
                           , cprec  ]
                         , preprec ]
                       ] (* end of Print *)

               ] ; (* end of If[ compoption == 0 ] *)

             If [ compoption == 1,

                  reps1=N[ Abs[ ( Re[ nps2 ]-Re[ series[t] ] ) /
                                           Re[ series[t] ]
                               ] ,
                          cprec ] ;
                  reps2=N[ Abs[ ( Im[ nps2 ] - Im[ series[t] ] ) /
```

```
                                                    Im[ series[t] ]
                                        ] ,
                                    cprec ] ;

                            If[ norm == -1,
                                 reps=N[ Max[ reps1, reps2 ] , cprec ]
                                ,
                                 reps=N[ ( reps1^norm +
                                           reps2^norm )^(1/norm) , cprec ]
                                ] ; (* end of If[ norm ] *)

                            Print["reps[", i-1, "]=[", N[ reps1, preprec ] ,
                                 "," , N[ reps2, preprec ] , "] , norm reps[" ,
                                 i-1 , "]=" , N[ reps, preprec ]
                                ] (* end of Print *)

                           ] ; (* end of If [ compoption == 1 ] *)

                      ] (* end of If[ psoption ] *)

          ] ; (* end of If[ preoption==1 ] *)

        If[ preoption == 2 ,

            If[ compoption == 0 ,

                 Print[ "edgpt[", i-1, "]=",
                        N[ N[
                             -Log[ 10, Abs[ wn2-series[t] ] ]
                            , cprec ]
                          , preprec ]
                      ] ; (* end of Print *)

            ] ; (* end of If[ compoption ] *)

             If[ compoption == 1 ,

                 edgpt1= N[ -Log[10,
                                 Abs[ Re[ wn2 ]- Re[ series[t] ]  ]
                               ],
                            cprec ] ;
                   edgpt2= N[ -Log[10,
                                   Abs[ Im[ wn2 ] - Im[ series[t] ]  ]
                                 ],
                              cprec ] ;

                    If[ norm == -1,
                      edgpt=N[ Min[ Abs[edgpt1], Abs[edgpt2] ] , cprec ]
                      ,
                      edgpt=N[ ( Abs[edgpt1]^norm +
                                 Abs[edgpt2]^norm )^(1/norm) ,
                               cprec ]
                      ] ; (* end of If[ norm ] *)

                      Print["edgpt[", i-1, "]=[", N[ edgpt1, preprec ] ,
                            "," , N[ edgpt2, preprec ] , "] , norm edgpt[" ,
                            i-1 , "]=" , N[ edgpt, preprec ]
                           ] (* end of Print *)

                ] ; (* end of If[ compoption == 1 ] *)

               If[ psoption == 1 ,

                   If[ compoption == 0 ,

                        Print[ "edps[", i-1, "]=",
                               N[ N[
                                    -Log[ 10, Abs[ nps2-series[t] ] ]
```

```
                                        , cprec  ]
                                      , preprec ]
                                    ] (* end of Print *)

                              ] ; (* end of If[ compoption ] *)

                          If[ compoption == 1 ,

                               edps1=N[ -Log[ 10,
                                              Abs[ Re[ nps2 ]-Re[ series[t] ] ]
                                                 ] ,
                                        cprec ] ;
                                edps2=N[ -Log[ 10,
                                              Abs[ Im[ nps2 ] - Im[ series[t] ] ]
                                                    ] ,
                                        cprec ] ;

                                If[ norm == -1,
                                   edps=N[ Min[ Abs[edps1], Abs[edps2] ] , cprec ]
                                     ,
                                   edps=N[ ( Abs[edps1]^norm +
                                             Abs[edps2]^norm )^(1/norm) , cprec ]
                                     ] ; (* end of If[ norm ] *)

                                Print["edps[", i-1, "]=[", N[ edps1, preprec ] ,
                                      "," , N[ edps2, preprec ] , "] , norm edps[" ,
                                      i-1 , "]=" , N[ edps, preprec ]
                                     ] (* end of Print *)

                              ] ; (* end of  If[ compoption == 1 ] *)

                        ] (* end of If[ psoption ] *)

                ] (* end of If[ preoption==2 ] *)

        , (* toption == "VAR" *)

         If[ coption == "EC" ,

               wf2 = wf1 + xarray[[1,1]]*larray[[1,1]]  ;
               Print[ "fgpt[", i-1, "]=", Together[ wf2]  ] ;
               If[ psoption == 1 ,
                    fps2 = fps1 + cm[[i]]*wf[[i]] ;
                    Print[ "fps[", i-1 , "]=", fps2 ]
                  ] (* end of If[ psoption ] *)

             , (* coption == "AC" *)

               wn2=N[ wn1 + xarray[[1,1]]*larray[[1,1]]
                          , cprec ] ;
               Print["ngpt[",i-1,"]=", N[ Together[wn2] , resultprec ] ] ;
               If[ psoption == 1 ,
                    nps2=N[ nps1 + cm[[i]]*wf[[i]] , cprec ] ;
                    Print[ "nps[",i-1,"]=", N[ nps2 , resultprec ] ]
                  ] (* end of If[ psoption ] *)

            ](* end of If [ coption ] *)

      ];  (* end of If[ toption ] *)

     Print[" "];

    If[ coption == "EC",
         wf1=wf2;
         fps1=fps2
              ,
         wn1=wn2 ;
         nps1=nps2
```

```
    ]
,{i,2,k+1}
](* end of Do *)

(*_____________________________________________________________________*)
](* end of Block *)


End[ ]

Protect[ tablegpt ]

EndPackage[ ]
```

```
(*************************************************************)
(********************** Package BE.M **********************)
(*************************************************************)

BeginPackage["BE`"]

Clear[ series, g, c, u, f ]

BE::usage="is a package that contains the definitions of:

- the series of functions
- the generating function
- the  modified moments of the functional c , ci=c(Ui(x))
- the polynomials Ui
- the functions fi

, that the package GPADETYPE.M needs"

series::usage="series[t] is the closed form of the
               series of functions"

g::usage="g[x,t] is the closed form of the generating
          function"

c::usage="c[i] is the i-th  modified moment of
          the functional c"

u::usage="u[i,x] is the polynomial corresponding
          to the moment ci"

f::usage="f[i,t] are the elementary functions"


Begin["`Private`"]



(*********************************************************************
***************   EXPANSIONS IN  POWER SERIES   *******************
*********************************************************************)

(*****************************   EXAMPLE 1    ************************)

(********   initialf[i,f[x],x]=f[-1/(i+1)]  , i=0,1,2,...   *******)
series[t_]:= Log[1+t]/t ;
g[x_,t_]:= 1/(1-x*t) ;
c[i_]:= (-1)^i/(i+1) ;
u[i_,x_]:= x^i ;
f[0,t_]:= 1 ;
f[i_,t_]:= t^i ;


(*************************   EXAMPLE 5    ************************)

(********   initialf[i,f[x],x]=f[-1/(i+1)]  , i=0,1,2,...   *******)
(*series[t_]:= Exp[-t] ;
g[x_,t_]:= 1/(1-x*t) ;
c[i_]:= (-1)^i/i! ;
u[i_,x_]:= x^i ;
f[0,t_]:= 1 ;
f[i_,t_]:= t^i ; *)


(*************************   EXAMPLE 9    ************************)

(********   initialf[i,f[x],x]=f[-1/(i+1)]  , i=0,1,2,...   *******)
(*series[t_]:= 1/Sqrt[1+t] ;
g[x_,t_]:= 1/(1-x*t)  ;
```

```
c[0]:=1 ;
c[i_]:= (-1)^i*Factorial2[ 2*i-1 ]/Factorial2[ 2*i ] ;
u[i_,x_]:= x^i  ;
f[0,t_]:=1 ;
f[i_,t_]:= t^i ; *)


(************************** EXAMPLE 10 ************************)

(******** initialf[i,f[x],x]=f[1/(i+1)] , i=0,1,2,... *********)
(*series[t_]:=Exp[7*t] ;
g[x_,t_]:= 1/(1-x*t)  ;
c[i_]:= 7^i/i! ;
u[i_,x_]:= x^i  ;
f[0,t_]:=1 ;
f[i_,t_]:= t^i ; *)



(*********************************************************************
*     EXPANSIONS IN THE FIRST KIND TCHEBICHEV  POLYNOMIALS      *
*********************************************************************)


(************************  EXAMPLE 11  ***********************)

(******* initialf[i,f[x],x]=f[1/(i+1)] , i=0,1,2,...  *******)
(*series[t_]:=Exp[7*t] ;
g[x_,t_]:=(1-x*t)/(1-2*x*t+x*x)  ;
c[0]:= BesselI[ 0, 7 ]  ;
c[i_]:= 2*BesselI[ i, 7 ] ;
u[i_,x_]:=x^i  ;
f[i_,t_]:=ChebyshevT[ i, t ] ; *)


(************************ EXAMPLE 12 ************************)

(******* initialf[i,f[x],x]=f[1/(i+1)] , i=0,1,2,...  ******)
(*series[t_]:=Exp[t]*Cos[Sqrt[1-t*t]]  ;
g[x_,t_]:=(1-x*t)/(1-2*x*t+x*x)  ;
c[i_]:= 1/i! ;
u[i_,x_]:=x^i  ;
f[i_,t_]:=ChebyshevT[ i, t ] ; *)


(************************** EXAMPLE 13 ************************)

(******* initialf[i,f[x],x]=f[1/(i+1)] , i=0,1,2,...  *******)
(*series[t_]:=Exp[4*t]*Cos[4*Sqrt[1-t*t]]  ;
g[x_,t_]:=(1-x*t)/(1-2*x*t+x*x)  ;
c[i_]:= 4^i/i! ;
u[i_,x_]:=x^i  ;
f[i_,t_]:=ChebyshevT[ i, t ] ; *)


(*********************************************************************
************* EXPANSIONS IN LAGUERRE POLYNOMIALS *************
*********************************************************************)


(************************** EXAMPLE 14 ************************)

(****** initialf[i,f[x],x]=f[1/(i+2)] , i=0,1,2,...  ******)
(*series[t_]:=Exp[1]*BesselJ[ 0, 2*Sqrt[t] ] ;
g[x_,t_]:=Exp[-x/(1-x)*t]/(1-x)  ;
c[i_]:= 1/i! ;
u[i_,x_]:=x^i  ;
f[i_,t_]:=LaguerreL[ i, 0, t ] ; *)
```

```
(*************************** EXAMPLE 15 ***************************)

(******      initialf[i,f[x],x]=f[1/(i+2)] , i=0,1,2,...     ******)
(*series[t_]:=Exp[5]*BesselJ[ 0, 2*Sqrt[5*t] ] ;
g[x_,t_]:=Exp[-x/(1-x)*t]/(1-x)  ;
c[i_]:= 5^i/i! ;
u[i_,x_]:=x^i  ;
f[i_,t_]:=LaguerreL[ i, 0, t ] ; *)


End[ ]

EndPackage[ ]
```

```
(*************************************************************)
(********************** Package BIF.M ************************)
(*************************************************************)

BeginPackage["BIF`"]

Clear[ initialf ]

BIF::usage="is a package,that contains the definitions
            of the initial functionals, that the package GPADETYPE.M
            needs"

initialf::usage="initialf[i,body,var] is the definition of
                 the i-th initial functional applied to Function[var,body]"


Begin["`Private`"]


initialf[i_,body_,var_] :=
      Function[var,body][-1/(i + 1)] ;

(*initialf[i_,body_,var_] :=
        Function[var,body][-1/(i + 2)] ;  *)

(*initialf[i_,body_,var_] :=
        Function[var,body][+1/(i + 1)] ;  *)

(*initialf[i_,body_,var_] :=
        Function[var,body][+1/(i + 2)] ;  *)

(*initialf[i_,body_,var_] :=
         Integrate[body,{var,-1/(i+1),-1/(i+2)}] ; *)


End[ ]

EndPackage[ ]
```

```
(*_____________________________________________________________________
_____________________Mathematica SESSION_____________________
_____________________________________________________________________

          WE GIVE BELOW SOME EXAMPLES TO SHOW THE CAPABILITIES
              OF THE FUNCTION tablegpt, AND ALSO THE TYPE OF
              RESULTS THAT CAN BE OBTAINED, IF THE ARGUMENTS
                    TOPTION AND COPTION ARE WRONG.

_____________________________________________________________________*)

<<GPADETYPE.M

series[t]

Log[1 + t]
----------
    t

g[x,t]

   1
-------
1 - t x

c[i]

    i
(-1)
-----
1 + i

u[i,x]

 i
x

f[0,t]

1

f[i,t]

 i
t

initialf[i,f[x],x]

      1
f[-(-----)]
    1 + i

(*_____________________________________________________________________
            Let's remark that the series, the generating
            function, the moments, the basis of polynomials,
            the elementary functions and the initials
            functionals are defined in an exact way.
  _____________________________________________________________________*)

tablegpt[3,t,"VAR","EC"]

fgpt[0]=1/(1 + t)

fgpt[1]=2/(2 + t)

fgpt[2]=(6 + 5 t)/(2 (1 + t) (3 + t))

                              2       3
```

```
fgpt[3]=(144 + 228 t + 108 t  + 19 t )/
                           2      3

   (6 (1 + t) (2 + t) (3 + t) (4 + t))

(*_____________________________________________________________________
             t is a symbolic expression, then we should
             take toption == "VAR". Taking coption == "EC",
             we get the formal expressions of the
             approximants in the variable t.
  _____________________________________________________________________*)

tablegpt[3,t,"VAR","EC",1]

fgpt[0]=1/(1 + t)
fps[0]=1

fgpt[1]=2/(2 + t)
fps[1]=1 - t/2

fgpt[2]=(6 + 5 t)/(2 (1 + t) (3 + t))
                    2
fps[2]=1 - t/2 + t /3

                                 2       3
fgpt[3]=(144 + 228 t + 108 t  + 19 t )/

   (6 (1 + t) (2 + t) (3 + t) (4 + t))
                    2      3
fps[3]=1 - t/2 + t /3 - t /4

(*_____________________________________________________________________
               Taking psoption == 1, we get also the
               formal expressions of the partial
               sums of the series.
   _____________________________________________________________________*)

tablegpt[3,t,"VAR","AC",0,30,0,0,40]

          1.
ngpt[0]=------
        1. + t

ngpt[1]=(2. + 2. t)/((1. + t) (2. + 1. t))

                              2
ngpt[2]=(6. + 8. t + 2.5 t )/

   ((1. + t) (3. + 1. t) (2. + 1. t))

                                 2
ngpt[3]=(24. + 38. t + 18. t  +

                                        3
     3.16666666666666666666666666667 t )/

   ((1. + t) (3. + 1. t) (2. + 1. t) (4. + 1. t))

(*_____________________________________________________________________
               coption == "AC" and cprec == 40, then
               the intermediate calculations are made with
               precision 40, using the function N[-,40].

               resultprec == 30, then the results are printed
               with 30 decimal digits of precision.

               As toption == "VAR", the relative errors or
               the number of exact digits are not calculated,
               and the seventh and the eighth arguments are
```

```
                not employed.
  ____________________________________________________________________*)

tablegpt[3,t,"VAR","AC",1,30,0,0,40]

ngpt[0]=1./(1. + t)
nps[0]=1.

ngpt[1]=(2. + 2. t)/((1. + t) (2. + 1. t))
nps[1]=1. - 0.5 t

                              2
ngpt[2]=(6. + 8. t + 2.5 t )/

   ((1. + t) (3. + 1. t) (2. + 1. t))

                                                          2
nps[2]=1. - 0.5 t + 0.33333333333333333333333333333 t

                                 2
ngpt[3]=(24. + 38. t + 18. t  +

                                          3
     3.16666666666666666666666666667 t )/

   ((1. + t) (3. + 1. t) (2. + 1. t) (4. + 1. t))

                                                          2
nps[3]=1. - 0.5 t + 0.33333333333333333333333333333 t  -

         3
   0.25 t

(*_____________________________________________________________________
                Taking psoption == 1 , we get also the
                approximated symbolic expressions of the
                partial sums of the series.
  ____________________________________________________________________*)

(*_____________________________MISTAKE_____________________________*)

tablegpt[3,t,"SCL","EC"]

fgpt[0]=1/(1 + t)

fgpt[1]=1/(1 + t/2)

fgpt[2]=3/(4 (1 + t/3)) + 1/(4 (1 + t))

fgpt[3]=16/(9 (1 + t/4)) - 9/(4 (1 + t/3)) +

   4/(3 (1 + t/2)) + 5/(36 (1 + t))

(*_____________________________________________________________________
              TOPTION SHOULD BE EQUAL TO "VAR", NOT TO "SCL"

              In spite of the error, we obtain the exact
              expression of the approximants in the point t.

              The missing arguments take their default values.
              In particular, preoption == 0 and aroption == 0,
              then the relative errors or the number of exact
              digits are not calculated, and the numerical
              approximations of the exact expressions of the
              approximants are not written.
  ____________________________________________________________________*)

(*_____________________________MISTAKE_____________________________*)

tablegpt[2,t,"SCL","EC",1,19,0,0,0,1]
```

```
fgpt[0]=1/(1 + t)
fps[0]=1
f[t]=Log[1. + t]/t
ngpt[0]=1/(1. + t)
nps[0]=1.

fgpt[1]=1/(1 + t/2)
fps[1]=1 - t/2
f[t]=Log[1. + t]/t
ngpt[1]=1/(1. + 0.5 t)
nps[1]=1. - 0.5 t

fgpt[2]=3/(4 (1 + t/3)) + 1/(4 (1 + t))
                    2
fps[2]=1 - t/2 + t /3
f[t]=Log[1. + t]/t
ngpt[2]=1/(1. + 0.5 t) +

   0.125 (6. (1/(1. + 0.3333333333333333333 t) -

         1./(1. + 0.5 t)) -

      2. (1/(1. + 0.5 t) - 1./(1. + t)))
                                          2
nps[2]=1. - 0.5 t + 0.3333333333333333333 t

(*_____________________________________________________________________
              TOPTION SHOULD BE EQUAL TO "VAR", NOT TO "SCL"

              psoption == 1 and coption == "EC", then the
              exact expression of the partial sums are also
              calculated.

              aroption == 1, then the exact expression of
              the approximants and the partial sums are
              written with resultprec == 19 decimal digits
              of precision.

              As toption == "SCL", coption == "EC" and
              aroption == 1, the value of the series in the
              point t is written with resultprec == 19
              digits of precision.
  _____________________________________________________________________*)

(*_____________________________MISTAKE_____________________________*)

tablegpt[2,t,"SCL","AC",0,25,0,0,30]

f[t]=Log[1. + t]/t
ngpt[0]=1./(1. + t)

f[t]=Log[1. + t]/t
ngpt[1]=1./(1. + t) + 1. (1./(1. + 0.5 t) - 1./(1. + t))

f[t]=Log[1. + t]/t
ngpt[2]=1./(1. + t) + 1.

    (1./(1. + 0.5 t) - 1./(1. + t)) +

   0.125 (6. (1./(1. + 0.3333333333333333333333333 t) -

         1./(1. + 0.5 t)) -

      2. (1./(1. + 0.5 t) - 1./(1. + t)))


(*_____________________________________________________________________
```

```
                TOPTION SHOULD BE EQUAL TO "VAR", NOT TO "SCL"

                coption == "AC" and cprec == 30, then the
                intermediate calculations are made with
                precision 30, using the function N[-,30].

                resultprec == 25, then the results are printed
                with 25 digits of precision.

                As toption == "SCL" and coption == "AC" the
                value of the series in the point t is written
                with resultprec == 25 digits of precision.

                We take preoption == 0, then the relative
                errors or the number of exact digits are not
                calculated.
  ____________________________________________________________________*)

(*_____________________________MISTAKE_____________________________*)

tablegpt[2,t,"SCL","AC",1,25,0,0,30]

f[t]=Log[1. + t]/t
ngpt[0]=1./(1. + t)
nps[0]=1.

f[t]=Log[1. + t]/t
ngpt[1]=1./(1. + t) + 1. (1./(1. + 0.5 t) - 1./(1. + t))
nps[1]=1. - 0.5 t

f[t]=Log[1. + t]/t
ngpt[2]=1./(1. + t) + 1.

    (1./(1. + 0.5 t) - 1./(1. + t)) +

   0.125 (6. (1./(1. + 0.33333333333333333333333333 t) -

         1./(1. + 0.5 t)) -

      2. (1./(1. + 0.5 t) - 1./(1. + t)))
                                                    2
nps[2]=1. - 0.5 t + 0.33333333333333333333333333 t

(* ____________________________________________________________

                Taking psoption == 1 , we get also the
                approximated values of the partial sums
                of the series that are calculated with
                precision cprec == 30, and are written
                with resultprec == 25 digits of precision.
  ____________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",0,19,0,0,0,1]

fgpt[0]=10/11
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909

fgpt[1]=20/21
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809

fgpt[2]=325/341
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985

fgpt[3]=839495/880803
f[1/10]=0.9531017980432485999
```

```
ngpt[3]=0.9531018854386281609

(*_____________________________________________________________________
              1/10 is a numerical expression, then we should
              take toption == "SCL".

              As 1/10 is exact, and the definitions of g,
              ci, ui, fi and initialf are exacts, we can
              obtain the exact numerical expressions of the
              approximants taking coption == "EC".

              aroption == 1, then the exact expressions of
              the approximants are written with
              resultprec ==19 decimal digits of precision.

              We take preoption == 0, then the relative errors
              or the number of exact digits are not calculated.

              As toption == "SCL", coption == "EC" and
              aroption == 1 the value of the series in the
              point 1/10 is written with resultprec == 19
              digits of precision.
 ____________________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",1,19,0,0,0,1]

fgpt[0]=10/11
fps[0]=1
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
nps[0]=1.

fgpt[1]=20/21
fps[1]=19/20
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809
nps[1]=0.95

fgpt[2]=325/341
fps[2]=143/150
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985
nps[2]=0.9533333333333333333

fgpt[3]=839495/880803
fps[3]=11437/12000
f[1/10]=0.9531017980432485999
ngpt[3]=0.9531018854386281609
nps[3]=0.9530833333333333334

(*_____________________________________________________________________
                Taking psoption == 1, we obtain also the exact
                numerical expressions of the partial sums of the
                series.

                As aroption == 1, these exact expressions are
                written with resultprec == 19 decimal digits of
                precision.
  ___________________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",0,19,1,2,19,1]

fgpt[0]=10/11
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
regpt[0]=0.046
```

```
fgpt[1]=20/21
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809
regpt[1]=0.00076

fgpt[2]=325/341
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985
regpt[2]=0.000024

fgpt[3]=839495/880803
f[1/10]=0.9531017980432485999
ngpt[3]=0.9531018854386281609
                -8
regpt[3]=9.2 10

(*_____________________________________________________________________
                As preoption == 1, the relative errors of
                the exact numerical expressions of the
                approximants with respect to the series are
                calculated with precision cprec == 19, and are
                written with preprec == 2 decimal digits of
                precision.
  _____________________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",0,19,2,2,19,1]

fgpt[0]=10/11
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
edgpt[0]=1.4

fgpt[1]=20/21
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809
edgpt[1]=3.1

fgpt[2]=325/341
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985
edgpt[2]=4.6

fgpt[3]=839495/880803
f[1/10]=0.9531017980432485999
ngpt[3]=0.9531018854386281609
edgpt[3]=7.1


(*_____________________________________________________________________
              As preoption == 2, the number of exact
              digits of the numerical exact expressions of
              the approximants in comparison with the series
              are calculated with precision cprec == 19, and
              are written with preprec == 2 decimal digits of
              precision.
  _____________________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",1,19,1,2,19,1]

fgpt[0]=10/11
fps[0]=1
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
nps[0]=1.
regpt[0]=0.046
reps[0]=0.049
```

```
fgpt[1]=20/21
fps[1]=19/20
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809
nps[1]=0.95
regpt[1]=0.00076
reps[1]=0.0033

fgpt[2]=325/341
fps[2]=143/150
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985
nps[2]=0.9533333333333333333
regpt[2]=0.000024
reps[2]=0.00024

fgpt[3]=839495/880803
fps[3]=11437/12000
f[1/10]=0.9531017980432485999
ngpt[3]=0.9531018854386281609
nps[3]=0.9530833333333333334
                -8
regpt[3]=9.2 10
reps[3]=0.000019

(*_____________________________________________________________________

                As preoption == 1, the relative errors
                of the exact numerical expression of the
                approximants and the partial sums, with respect
                to the series, are calculated with precision
                cprec == 19, and are written with preprec == 2
                decimal digits of precision.
  _____________________________________________________________________*)

tablegpt[3,1/10,"SCL","EC",1,19,2,2,19,1]

fgpt[0]=10/11
fps[0]=1
f[1/10]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
nps[0]=1.
edgpt[0]=1.4
edps[0]=1.3

fgpt[1]=20/21
fps[1]=19/20
f[1/10]=0.9531017980432485999
ngpt[1]=0.9523809523809523809
nps[1]=0.95
edgpt[1]=3.1
edps[1]=2.5

fgpt[2]=325/341
fps[2]=143/150
f[1/10]=0.9531017980432485999
ngpt[2]=0.9530791788856304985
nps[2]=0.9533333333333333333
edgpt[2]=4.6
edps[2]=3.6

fgpt[3]=839495/880803
fps[3]=11437/12000
f[1/10]=0.9531017980432485999
ngpt[3]=0.9531018854386281609
nps[3]=0.9530833333333333334
edgpt[3]=7.1
edps[3]=4.7
```

```
(*_____________________________________________________________________
                As preoption == 2, we calculate the number of
                exact digits instead of relative errors.
  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","EC",0,19,1,2,19,1,1]

fgpt[0]=55/61 - (5 I)/61
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[0]=0.901639344262295082 - 0.08196721311475409836 I
regpt[0]=[0.051,0.87] , norm regpt[0]=0.87

fgpt[1]=210/221 - (10 I)/221
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[1]=0.9502262443438914027 - 0.04524886877828054299 I
regpt[1]=[0.00021,0.032] , norm regpt[1]=0.032

fgpt[2]=55775/58682 - (2575 I)/58682
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[2]=0.9504618111175488225 - 0.04388057666746191336 I
regpt[2]=[0.000037,0.0012] , norm regpt[2]=0.0012

(*_____________________________________________________________________

                1/10+I/10 is a complex number, then we should
                take compoption == 1.

                preoption == 1, then the relative errors of the
                real and imaginary parts of the exact numerical
                expressions of the approximants, with respect to
                the real and imaginary parts of the series, are
                calculated with precision cprec == 19 and are
                written with preprec == 2 decimal digits of
                precision.

                norm takes his default value (norm == -1), then
                the infinity norms of the vectors of relative
                errors are calculated with precision cprec == 19
                and are written with preprec == 2 decimal digits
                of precision.
  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","EC",1,19,1,2,19,1,1]

fgpt[0]=55/61 - (5 I)/61
fps[0]=1
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[0]=0.901639344262295082 - 0.08196721311475409836 I
nps[0]=1.
regpt[0]=[0.051,0.87] , norm regpt[0]=0.87
reps[0]=[0.052,1.] , norm reps[0]=1.

fgpt[1]=210/221 - (10 I)/221
fps[1]=19/20 - I/20
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[1]=0.9502262443438914027 - 0.04524886877828054299 I
nps[1]=0.95 - 0.05 I
```

```
regpt[1]=[0.00021,0.032] , norm regpt[1]=0.032
reps[1]=[0.00045,0.14] , norm reps[1]=0.14

fgpt[2]=55775/58682 - (2575 I)/58682
fps[2]=19/20 - (13 I)/300
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[2]=0.9504618111175488225 - 0.04388057666746191336 I
nps[2]=0.95 - 0.04333333333333333333 I
regpt[2]=[0.000037,0.0012] , norm regpt[2]=0.0012
reps[2]=[0.00045,0.011] , norm reps[2]=0.011

(*_____________________________________________________________________

             psoption == 1, then we calculate the exact
             numerical expressions of the partial sums of the
             series.

             As aroption == 1, these exact expressions are
             written with resultprec == 19 decimal digits of
             precision.

             norm takes his default value (norm == -1), then
             the infinity norms of the vectors of the relative
             errors of the partial sums are also calculated
             with precision cprec == 19 and are written with
             preprec == 2 decimal digits of precision.

  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","EC",1,19,2,2,19,1,1]

fgpt[0]=55/61 - (5 I)/61
fps[0]=1
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[0]=0.901639344262295082 - 0.08196721311475409836 I
nps[0]=1.
edgpt[0]=[1.3,1.4] , norm edgpt[0]=1.3
edps[0]=[1.3,1.4] , norm edps[0]=1.3

fgpt[1]=210/221 - (10 I)/221
fps[1]=19/20 - I/20
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[1]=0.9502262443438914027 - 0.04524886877828054299 I
nps[1]=0.95 - 0.05 I
edgpt[1]=[3.7,2.8] , norm edgpt[1]=2.8
edps[1]=[3.4,2.2] , norm edps[1]=2.2

fgpt[2]=55775/58682 - (2575 I)/58682
fps[2]=19/20 - (13 I)/300
f[1/10 + I/10]=0.9504265828666385413 -

   0.04382771085918740628 I
ngpt[2]=0.9504618111175488225 - 0.04388057666746191336 I
nps[2]=0.95 - 0.04333333333333333333 I
edgpt[2]=[4.5,4.3] , norm edgpt[2]=4.3
edps[2]=[3.4,3.3] , norm edps[2]=3.3

(*_____________________________________________________________________

             preoption == 2, then the numbers of exact digits
             of the real and imaginary parts of the exact
             numerical expressions of the approximants and the
```

```
            partial sums, with respect to the real and imaginary
            parts of the series, are calculated with precision
            cprec == 19 and are written with preprec == 2 decimal
            digits of precision.

            norm takes his default value (norm == -1), then
            the components of minimum absolute value of the
            vectors of the numbers of exact digits of the
            approximants and the partial sums are calculated
            with precision cprec == 19 and are written with
            preprec == 2 decimal digits of precision.
  ____________________________________________________________________*)

tablegpt[3,1/10,"SCL","AC",0,30,1,2,40]

f[1/10]=0.953101798043248600439521232 8
ngpt[0]=0.909090909090909090909090909091
regpt[0]=0.046

f[1/10]=0.953101798043248600439521232 8
ngpt[1]=0.952380952380952380952380952381
regpt[1]=0.00076

f[1/10]=0.953101798043248600439521232 8
ngpt[2]=0.953079178885630498533724340176
regpt[2]=0.000024

f[1/10]=0.953101798043248600439521232 8
ngpt[3]=0.953101885438628160894093230836
                -8
regpt[3]=9.2 10

(*__________________________________________________________________

            coption == "AC" and cprec == 40, then the
            intermediate calculations are done with
            precision 40, using the function N[-,40].

            resultprec == 30, then the results are printed
            with 30 decimal digits of precision.

            This is possible, because the precision of
            1/10 is greather than 40.
            (Precision[1/10] == Infinity)

            As toption == "SCL" and coption == "AC" the
            value of the series in t == 1/10 is written
            with resultprec == 30 decimal digits of precision.
  ____________________________________________________________________*)

tablegpt[3,0.1,"SCL","AC",0,30,1,2,40]

f[0.1]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
regpt[0]=0.046

f[0.1]=0.9531017980432485999
ngpt[1]=0.952380952380952381
regpt[1]=0.00076

f[0.1]=0.9531017980432485999
ngpt[2]=0.9530791788856304986
regpt[2]=0.000024

f[0.1]=0.9531017980432485999
ngpt[3]=0.953101885438628161
                -8
regpt[3]=9.2 10
```

```
(*_____________________________________________________________________

              If we replace 1/10 by 0.1, which is a low
              precision number, calculations are done in
              the machine precision, and the results can
              not have more than 19 decimal digits of
              precision.

              Precision[1/10]=Infinity , Precision[0.1]=19

              Thus, we should ask for:
  _____________________________________________________________________*)

tablegpt[3,0.1,"SCL","AC",0,19,1,2]

f[0.1]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
regpt[0]=0.046

f[0.1]=0.9531017980432485999
ngpt[1]=0.952380952380952381
regpt[1]=0.00076

f[0.1]=0.9531017980432485999
ngpt[2]=0.9530791788856304986
regpt[2]=0.000024

f[0.1]=0.9531017980432485999
ngpt[3]=0.953101885438628161
                -8
regpt[3]=9.2 10

(*_____________________________________________________________________*)

tablegpt[3,0.1,"SCL","AC",1,19,2,2]

f[0.1]=0.9531017980432485999
ngpt[0]=0.9090909090909090909
nps[0]=1.              .
edgpt[0]=1.4
edps[0]=1.3

f[0.1]=0.9531017980432485999
ngpt[1]=0.952380952380952381
nps[1]=0.95
edgpt[1]=3.1
edps[1]=2.5

f[0.1]=0.9531017980432485999
ngpt[2]=0.9530791788856304986
nps[2]=0.9533333333333333333
edgpt[2]=4.6
edps[2]=3.6

f[0.1]=0.9531017980432485999
ngpt[3]=0.953101885438628161
nps[3]=0.9530833333333333334
edgpt[3]=7.1
edps[3]=4.7

(*_____________________________________________________________________

              Taking psoption == 1, we obtain also the
              approximated values of the partial sums of the
              series.
  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","AC",0,30,1,2,40,0,1,2]
```

```
f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[0]=0.90163934426229508196721311 4754 -

   0.08196721311475409836065573770 49 I
regpt[0]=[0.051,0.87] , norm regpt[0]=0.87

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[1]=0.95022624434389140271493212 6697 -

   0.04524886877828054298642534 3665 I
regpt[1]=[0.00021,0.032] , norm regpt[1]=0.032

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[2]=0.95046181111754882246685525 374 -

   0.04388057666746191336355270 7815 I
regpt[2]=[0.000037,0.0012] , norm regpt[2]=0.0012

(*_____________________________________________________________________

              1/10+I/10 is a complex number, then we should
              take compoption == 1.

              preoption == 1, then the relative errors of the
              real and imaginary parts of the approximants are
              calculated with precision cprec == 40 and are
              written with preprec == 2 decimal digits of
              precision.

              norm == 2, then the euclidian norms of the vectors
              of relative errors are calculated with precision
              cprec == 40 and are written with preprec == 2
              decimal digits of precision.
  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","AC",1,30,1,2,40,0,1,2]

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[0]=0.90163934426229508196721311 4754 -

   0.08196721311475409836065573770 49 I
nps[0]=1.
regpt[0]=[0.051,0.87] , norm regpt[0]=0.87
reps[0]=[0.052,1.] , norm reps[0]=1.

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[1]=0.95022624434389140271493212 6697 -

   0.04524886877828054298642534 3665 I
nps[1]=0.95 - 0.05 I
regpt[1]=[0.00021,0.032] , norm regpt[1]=0.032
reps[1]=[0.00045,0.14] , norm reps[1]=0.14

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.04382771085918740783966241 22 I
ngpt[2]=0.95046181111754882246685525 374 -
```

```
   0.043880576667461913363552707815 I
nps[2]=0.95 - 0.043333333333333333333333333333333 I
regpt[2]=[0.000037,0.0012] , norm regpt[2]=0.0012
reps[2]=[0.00045,0.011] , norm reps[2]=0.011

(*_____________________________________________________________________

            psoption == 1, then the partial sums of the
            series are also calculated with precision
            cprec == 40 and are written with
            resultprec == 30 decimal digits of precision.

            norm == 2, then the euclidian norms of the
            vectors of relative errors of the partial
            sums are also calculated with precision
            cprec == 40 and are written with
            preprec == 2 decimal digits of precision.
  _____________________________________________________________________*)

tablegpt[2,1/10+I/10,"SCL","AC",1,30,2,2,40,0,1,2]

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.043827710859187407839662412 2 I
ngpt[0]=0.90163934426229508196721311 4754 -

   0.081967213114754098360655737704 9 I
nps[0]=1.
edgpt[0]=[1.3,1.4] , norm edgpt[0]=1.9
edps[0]=[1.3,1.4] , norm edps[0]=1.9

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.043827710859187407839662412 2 I
ngpt[1]=0.95022624434389140271493212 6697 -

   0.045248868778280542986425339 3665 I
nps[1]=0.95 - 0.05 I
edgpt[1]=[3.7,2.8] , norm edgpt[1]=4.7
edps[1]=[3.4,2.2] , norm edps[1]=4.

f[1/10 + I/10]=0.95042658286663854282352916 53 -

   0.043827710859187407839662412 2 I
ngpt[2]=0.95046181111754882246685525374 -

   0.043880576667461913363552707815 I
nps[2]=0.95 - 0.043333333333333333333333333333333 I
edgpt[2]=[4.5,4.3] , norm edgpt[2]=6.2
edps[2]=[3.4,3.3] , norm edps[2]=4.7

(*_____________________________________________________________________

            preoption == 2, the number of exact digits are
            calculated instead of relative errors.

            Be careful about the interpretation we give
            to the value of the euclidian norm of the vectors
            of the numbers of exact digits!

            Remark that norm edgpt[2]=6.2  but the real and
            imaginary parts of ngpt[2]  have non more than
            4 correct digits.

  _____________________________________________________________________*)

(*_____________________________MISTAKE_____________________________*)
```

```
tablegpt[3,1/10,"VAR","EC",0,19,1,2,19,1]

fgpt[0]=10/11

fgpt[1]=20/21

fgpt[2]=325/341

fgpt[3]=839495/880803

(*______________________________________________________________________
              TOPTION SHOULD BE EQUAL TO "SCL", NOT TO "VAR"

              We get only the exact expressions of the
              approximants.

              preoption == 1, but relative errors are not
              calculated, because toption == "VAR".
              aroption == 1, but the numerical approximations
              of the approximants are not written, because
              toption == "VAR".

              The values of resultprec and cprec are not
              employed.
______________________________________________________________________*)

tablegpt[3,1/10,"VAR","EC",1,19,1,2,19,1]

fgpt[0]=10/11
fps[0]=1

fgpt[1]=20/21
fps[1]=19/20

fgpt[2]=325/341
fps[2]=143/150

fgpt[3]=839495/880803
fps[3]=11437/12000

(*______________________________________________________________________

               Taking psoption == 1, we get also the exact
               numerical expressions of the partial sums.
______________________________________________________________________*)

(*________________________________MISTAKE____________________________*)

tablegpt[3,1/10,"VAR","AC",0,30,2,2,40]

ngpt[0]=0.909090909090909090909090909091

ngpt[1]=0.952380952380952380952380952381

ngpt[2]=0.953079178885630498533724340176

ngpt[3]=0.953101885438628160894093230836

:[font = input; preserveAspect; ]
(*______________________________________________________________________
                TOPTION SHOULD BE EQUAL TO "SCL", NOT TO "VAR"

                coption == "AC" and cprec == 40, then the
                intermediate calculations are done with
                precision 40, using the function N[-,40].

                resultprec == 30, then the results are printed
```

```
                with 30 decimal digits of precision.

                preoption == 2, but the number of exact digits
                are not calculated, because toption == "VAR".

                Remark: Precision[1/10] >= 40.
_____________________________________________________________________*)

tablegpt[3,1/10,"VAR","AC",1,30,2,2,40]

ngpt[0]=0.909090909090909090909090909091
nps[0]=1.

ngpt[1]=0.952380952380952380952380952381
nps[1]=0.95

ngpt[2]=0.953079178885630498533724340176
nps[2]=0.953333333333333333333333333333

ngpt[3]=0.953101885438628160894093230836
nps[3]=0.953083333333333333333333333333

(*_____________________________________________________________________

                Taking psoption == 1, we get also the
                approximated numerical expressions of
                the partial sums, calculated with cprec == 40
                digits and written with resultprec == 30 digits.

  _____________________________________________________________________*)
```

D

BRR.M, BRE.M et BRIF.M

```
(*************************************************************************)
(**************************** Package BRR.M ******************************)
(*************************************************************************)

BeginPackage["BRR`","BRE`","BRIF`"]


BRR::usage="Is a package that contains the
definition of the function tabgptrr.
It calls the packages BRE.M and BRIF.M"

tabgptrr::usage="
tabgptrr[ k, t, toption, roption, psoption,
                resultprec, preoption, preprec,
                prerresprec, aroption, compoption, norm ]
calculates the (k+1) first generalized Pade-type approximants
in the point t.
The argument t can be a symbolic expression
(toption == \"VAR\"), or a numerical expression
(toption == \"SCL\"). In the first case we obtain
symbolic results, in the second case we obtain numerical
results.
When we expect exact results, we make roption == \"ER\".
When we expect approximated results, we make
roption == \"AR\".
When roption == \"AR\", the numerical results or the numerical
part of the symbolic results are printed with  resultprec decimal
digits of precision.
The default value of resultprec is the machine precision
( Precision[1.0] ).
If psoption == 1, the partial sums of the series are also
calculated. If psoption is different from 1, partial sums
are not calculated. The default value of psoption is zero.
If toption  == \"SCL\"  and preoption == 1,
the relative errors of the approximants with respect to the
series are calculated with precision preprec , and are written
with prerresprec  decimal digits of precision. Futhermore, if
psoption == 1, the relative errors of the partial sums with
respect to the series are calculated with precision preprec ,
and are written with prerresprec decimal digits of precision.
If toption  == \"SCL\"  and preoption == 2,
the number of exact digits of the approximants with respect
to the series are calculated with precision preprec , and are
written with prerresprec decimal digits of precision. Futhermore,
if psoption == 1, the number of exact digits of the partial sums
with respect to the series are calculated with precision
preprec, and are written with prerresprec decimal digits of
precision.
If compoption == 0 , we do real calculus. If compoption == 1,
we do complex calculus. The default value of compoption is
equal to 0.
If compoption == 1, toption == \"SCL\" and preoption  == 1, the
relative errors of the real and imaginary parts of the
approximants, with respect to the real and imaginary parts of
the series, are calculated with precision preprec and are written
with prerresprec decimal digits of precision. If psoption == 1,
we do the same for the partial sums.
If compoption == 1, toption == \"SCL\" and preoption  == 2, we
calculate the number of exact digits instead of the relative
errors.
If compoption == 1 and toption == \"SCL\",  we calculate also
the p-norm ( p >= 1) of the vector of relative errors
( if preoption  == 1 ), or the p-norm ( p >= 1) of the vector of
the number of exact digits  ( if preoption  == 2 ).
The value of p is given by the argument norm.
If norm == -1, we calculate the infinity norm of the
vector of relative errors ( if preoption  == 1 ), or the
component of minimum absolute value of the vector of the
```

```
number of exact digits  ( if preoption  == 2 ).
The default value of norm is equal to -1.
Norms are calculated with precision preprec and are written
with preresprec decimal digits of precision.
If toption  == \"VAR\" or preoption  is different from 1 and 2,
the relative errors or the number of exact digits are not
calculated.
The default values of preoption, preprec and preresprec are
0, Precision[1.0]  and 2 respectively.
If toption  == \"SCL\" and roption  == \"EC\", we obtain
the exact numerical expressions of the approximants
in the point t . Futhermore, if psoption == 1, we obtain also
the exact numerical expressions of the partial sums in the
point t. If preoption == 1, relative errors of these exact
expressions are calculated with precision preprec .
If preoption == 2 , the number of exact digits are calculated
with precision preprec. We should take a sufficiently
big value to preprec  in order that these values could be
correctly calculated. Futhermore, if aroption == 1,
the exact expressions just cited are written with resultprec
decimal digits of precision. If aroption is different from 1,
these values are not written.
The default value of aroption is zero.
We must give the first four arguments of tabgptrr ,
however the last eight can be omitted taking their default
values.

Reference:  Z. DA ROCHA.
                    Implementention of the recurrence
                    relations of biorthogonality.
                    In C. Brezinski, P. Gonzalez-Vera and
                    N. Hayek-Calil, eds., Extrapolation and
                    Rational Approximation. Tenerife 1992,
                    volume 3 of Numerical Algorithms, pp.
                    173-183, Basel 1992. J. C. Baltzer."


x::usage="x[i,j,k,var] is a linear expression of
Xj,...,X(j+k)."      .

l::usage="l[i,j,k,exp,var] is a linear functional
applied to Function[var,exp]."

key::usage="key[i,j,k,functionalc,exp,varx,vart]
is a function in the variable vart, calculated
in a recursive way using x[i,j,k,varx] and
l[i,j,k,exp,varx]."


Begin["`Private`"]

(*_____
_____ERROR MESSAGES OF tabgptrr_____
_____*)

tabgptrr::badarg1=
         "wrong value to the first argument
                  k  must be a non negative integer
                  try again"

tabgptrr::badarg3=
          "wrong value to the third argument
                  toption  must be equal to \"SCL\" or \"VAR\"
                  try again"

tabgptrr::badarg4=
         "wrong value to the fourth argument
                  roption  must be equal to \"AR\" or \"ER\"
```

```
                try again"

tabgptrr::badarg6=
        "wrong value to the sixth argument
                resultprec  must be a non negative integer
                try again"

tabgptrr::badarg8=
        "wrong value to the eighth argument
                preprec  must be a non negative integer
                try again"

tabgptrr::badarg9=
        "wrong value to the nineth argument
                preresprec  must be a non negative integer
                try again"

tabgptrr::badarg89=
       "wrong values to the eighth and/or the
            nineth arguments.
            We must have preresprec < = preprec
            Try again."

tablegpt::badarg12=
        "wrong value to the twelfth argument
                norm  must be equal to -1 or
                greater than or equal to 1
                try again"

(*_____________________________________________________________________
_____________________ DEFINITIONS OF THE AUXILIARY FUNCTIONS _____________
_____________________________ OF TABGPTRR _____________________________
_____________________________________________________________________*)

(*_____________________________________________________________________
_____________________________ RELATION X _____________________________
_____________________________________________________________________*)

x[ i_, j_, 0, var_ ]:= x[i,j,0,var]= urr[j,var] ;

x[ i_, j_, k_, var_]:= x[i,j,k,var]=
    x[ i, j+1, k-1, var ]-
    initialfrr[ i+k-1, x[i,j+1,k-1,var], var ]*
    x[ i, j, k-1, var ]/
    initialfrr[ i+k-1, x[i,j,k-1,var], var ] ;

(*_____________________________________________________________________
_____________________________ RELATION L _____________________________
_____________________________________________________________________*)

l[ i_, j_, 0, body_, var_ ]:= l[ i, j, 0, body, var ] =
  initialfrr[ i, body, var ]/
  initialfrr[ i, x[i,j,0,var], var ] ;

l[ i_, j_, k_, body_, var_ ]:= l[ i, j, k, body, var ] =
   (l[ i+1, j, k-1, body, var ]-l[ i, j, k-1, body, var ])/
   (l[ i+1, j, k-1, x[i,j+k,0,var], var ]-
    l[ i, j, k-1, x[i,j+k,0,var], var ]) ;

 (*_____________________________________________________________________
_____________________________ RELATION K _____________________________
_____________________________________________________________________*)

key[ i_, j_, -1, functional_, body_, varx_, vart_ ]:=
key[ i, j, -1, functional, body, varx, vart ]=0 ;

key[ i_, j_, k_, functional_, body_, varx_, vart_ ]:=
key[ i, j, k, functional, body, varx, vart ]=
```

```
        key[ i, j, k-1, functional, body, varx, vart ] +
        l[ i, j, k, body, varx ]*functional[ x[ i, j,  k, varx ], varx ] ;

(*_____________________________________________________________________
_____________________ DEFINITIONS OF THE PRIVATE FUNCTIONS _____________
_____________________________ OF TABGPTRR _____________________________
_____________________________________________________________________*)

functionalc[ poly_, var_ ] :=
  Block[{n},
      Sum[ Coefficient[poly,var,n]*crr[n]
             , { n,0,Exponent[poly,var] }
            ]
          ] ;

(*_____________________________________________________________________
_____________________________ DEFINITION OF tabgptrr ___________________
_____________________________________________________________________*)


tabgptrr[ k_, t_, toption_, roption_, psoption_:0,
                  resultprec_:(Precision[1.]), preoption_:0,
                  preprec_:(Precision[1.]), prerespprec_:(Precision[1.]),
                  aroption_:0, compoption_:0, norm_:-1 ]:=

 Block[ { w , i , ps , ergptrr1 , ergptrr2 , ergptrr ,
              erpsrr1 , erpsrr2 , erpsrr , edgptrr1 , edgptrr2 ,
              edgptrr , edpsrr1 , edpsrr2 , edpsrr },

 (*_____________________________________________________________________
 _____________________________ CONTROL OF ARGUMENTS ____________________
 _____________________________________________________________________*)

 If [ Or[ Not[ IntegerQ[k] ] ,  k < 0 ] ,
      Return[ Message[tabgptrr::badarg1] ]
     ];

 If [ And [ toption != "SCL" , toption != "VAR" ] ,
      Return[ Message[tabgptrr::badarg3] ]
     ];

 If[ And [ roption != "ER",  roption != "AR" ] ,
      Return[ Message[tabgptrr::badarg4] ]
    ] ;

If [ Or[ Not[ IntegerQ[ resultprec ] ] ,  resultprec < 0 ] ,
      Return[ Message[tabgptrr::badarg6] ]
    ];

If [ Or[ Not[ IntegerQ[ preprec ] ] ,  preprec < 0 ] ,
      Return[ Message[tabgptrr::badarg8] ]
    ];

If [ Or[ Not[ IntegerQ[ prerespprec ] ] ,  prerespprec < 0 ] ,
      Return[ Message[tabgptrr::badarg9] ]
    ];

If [  prerespprec  >  preprec ,
         Return [ Message [tabgptrr::badarg89 ] ]
     ];

 If [ And[ norm != -1 ,  norm < 1 ]  ,
       Return[ Message[tablegpt::badarg12] ]
    ] ;

 (*_____________________________________________________________________*)
 If[ psoption == 1 ,
       ps = 0
```

```
    ] ;

Do[ (* { i, 0, k } *)

    If[ psoption == 1 ,

    (* ____________________________________________________________
     __________ CALCULATION OF THE i-th PARTIAL SUM OF THE SERIES _______
     ____________________________________________________________*)

        ps = ps + crr[ i ] * frr[ i, t ]
      ] ;

     (* ____________________________________________________________
     _____________CALCULATION OF THE i-th APPROXIMANT_____________
     ____________________________________________________________*)

      w=key[ 0, 0, i, functionalc, grr[x,t], x, t ] ;

      (*____________________________________________________________
      _____________PRINTING OF THE i-th APPROXIMANT_____________
      ____________________________________________________________*)

      If[ toption == "SCL",

          If[ roption == "ER",

                Print[ "fgptrr[", i, "]=", Expand[w] ] ;
                If[ psoption == 1 ,
                     Print[ "fpsrr[", i, "]=", ps ]
                  ] ;
                If[ aroption == 1,
                      Print[ "ngptrr[", i, "]=", N[w,resultprec] ] ;
                      If[ psoption == 1 ,
                           Print[ "npsrr[", i, "]=", N[ps,resultprec] ]
                        ] (* end of If[ psoption ] *)
                  ]  (* end of If[ aroption ] *)

              , (* roption == "AR" *)

                Print[ "ngptrr[", i, "]=", N[w,resultprec] ] ;
                If[ psoption == 1 ,
                      Print[ "npsrr[", i, "]=", N[ps,resultprec] ]
                   ] (* end of If[ psoption ] *)
           ] ; (* end of If[ roption ] *)

           If[ preoption == 1 ,

               If[ compoption == 0 ,

                    Print[ "ergptrr[" , i , "]=" ,
                     N[ N[ Abs[ (w-seriesrr[t])/seriesrr[t] ]
                          , preprec ]
                       , preresprec ]
                          ] (* end of Print *)

                    ] ; (* end of If[ compoption == 0 ] *)

                  If[ compoption == 1,

ergptrr1=N[ N[ Abs[ (Re[w]-Re[seriesrr[t]])/Re[seriesrr[t] ] ]
                  , preprec ]
              , preresprec ] ;
ergptrr2=N[ N[ Abs[ (Im[w]-Im[seriesrr[t]])/Im[seriesrr[t] ] ]
                  , preprec ]
              , preresprec ] ;

                    If[ norm == -1,
```

```
                    ergptrr=N[ Max[ ergptrr1, ergptrr2 ] , preprec ]
                  ,
                    ergptrr=N[ ( ergptrr1^norm +
                                 ergptrr2^norm )^(1/norm) , preprec ]
                  ] ; (* end of If[ norm == -1] *)

    Print[ "ergptrr[" , i , "]=[" , ergptrr1, "," , ergptrr2, "]",
           " , norm ergptrr[" , i , "]=" , N[ ergptrr , preresprec ]
         ] (* end of Print *)

              ] ; (* end of If[ compoption == 1 ] *)

          If[ psoption == 1 ,

              If[ compoption == 0,

                    Print[ "erpsrr[" , i ,"]=" ,
                     N[ N[ Abs[ (ps-seriesrr[t])/seriesrr[t] ]
                         , preprec ]
                       , preresprec ]
                         ] (* end of Print *)

                  ] ; (* end of If[ compoption == 0 ] *)

               If[ compoption == 1 ,

    erpsrr1=N[ N[ Abs[ (Re[ps]-Re[seriesrr[t]])/Re[seriesrr[t]] ]
                , preprec ]
              , preresprec ] ;
    erpsrr2=N[ N[ Abs[ (Im[ps]-Im[seriesrr[t]])/Im[seriesrr[t]] ]
                , preprec ]
             , preresprec ] ;

                 If[ norm == -1,
                      erpsrr=N[ Max[ erpsrr1, erpsrr2 ] , preprec ]
                    ,
                     erpsrr=N[ ( erpsrr1^norm +
                                 erpsrr2^norm )^(1/norm) , preprec ]
                  ] ; (* end of If[ norm== -1 ] *)

      Print[ "erpsrr[" , i , "]=[" , erpsrr1, "," , erpsrr2, "]",
             " , norm erpsrr[", i , "]=", N[ erpsrr, preresprec ]
           ] (* end of Print *)

                    ] (* end of If[ compoption == 1 ] *)

                  ] (* end of If [ psoption == 1 ] *)

              ] ; (* end of If [ preoption == 1 ] *)

          If[ preoption == 2 ,

              If[ compoption == 0 ,

                 Print["edgptrr[",i,"]=",N[ N[ -Log[10,
                       Abs[ w-seriesrr[t] ]
                                                ]
                                            , preprec ]
                                          , preresprec ]
                    ] ;(* end of Print *)

                ] ; (* end of If[ compoption == 0 ] *)

             If[ compoption == 1,

    edgptrr1=N[ N[ -Log[10, Abs[ Re[w]-Re[seriesrr[t]] ]  ]
                  , preprec ]
                , preresprec ] ;
```

```
edgptrr2=N[ N[ -Log[10, Abs[ Im[w]-Im[seriesrr[t]] ]  ]
                    , preprec ]
                , prerespprec ] ;

                If[ norm == -1,
                     edgptrr=N[ Min[ Abs[edgptrr1], Abs[edgptrr2]  ] ,
                                preprec ]
                     ,
                     edgptrr=N[ ( Abs[edgptrr1]^norm +
                                  Abs[edgptrr2]^norm )^(1/norm) ,
                                preprec ]
                  ] ; (* end of If[ norm == -1 ] *)

                Print["edgptrr[" , i , "]=[", edgptrr1 , "," , edgptrr2 ,
                     "]" , " , norm edgptrr[", i , "]=" ,
                     N[ edgptrr , prerespprec ]
                     ] (* end of Print *)

                  ] ; (* end of If[ compoption == 1 ] *)

             If[ psoption == 1 ,

                  If[ compoption == 0 ,

                    Print["edpsrr[",i,"]=",N[ N[ -Log[10,
                                                 Abs[ ps-seriesrr[t] ]
                                                      ]
                                                   , preprec ]
                                              , prerespprec ]
                         ] (* end of Print *)

                     ] ; (* end of compoption == 0 *)

                   If[ compoption == 1 ,

        edpsrr1=N[ N[ -Log[10, Abs[ Re[ps]-Re[seriesrr[t] ] ] ]
                    , preprec ]
                , prerespprec ] ;
        edpsrr2=N[ N[ -Log[10, Abs[ Im[ps]-Im[seriesrr[t] ] ] ]
                     , preprec ]
                , prerespprec ] ;

                         If[ norm == -1,
               edpsrr=N[ Min[ Abs[edpsrr1], Abs[edpsrr2] ] , preprec ]
                         ,
               edpsrr=N[ ( Abs[edpsrr1]^norm +
                           Abs[edpsrr2]^norm )^(1/norm) , preprec ]
                           ] ; (* end of If[ norm ] *)
        Print["edpsrr[" , i , "]=[" , edpsrr1 , "," , edpsrr2, "]",
              " , norm edpsrr[", i , "]=" ,  N[ edpsrr, prerespprec ]
              ] (* end of Print *)

                     ] (* end of If[ compoption == 1 ] *)

                  ] (* end of psoption == 1 *)

               ] (* end of If [ preoption == 2 ] *)

           , (* toption == "VAR" *)

           If[ roption == "ER",

               Print[ "fgptrr[", i, "]=", Together[w] ] ;
               If [ psoption == 1 ,
                    Print[ "fpsrr[", i, "]=", ps ]
                  ] (* end of If[psoption] *)

               , (* roption == "AR" *)
```

```
                Print[ "ngptrr[", i, "]=", N[ Together[w], resultprec ] ] ;
                 If [ psoption == 1 ,
                       Print[ "npsrr[", i, "]=", N[ ps, resultprec ] ]
                     ] (* end of If[psoption] *)

               ] (* end of If[roption] *)

          ] ; (* end of If[toption] *)

         Print[" "]
         (*_____________________________________________________________*)

       ,{ i, 0, k }

       (*_____________________________________________________________*)

     ] (* end of Do *)

   ] (* end of Block *)

End[ ]

Protect[ tabgptrr ]

EndPackage[ ]
```

```
(*************************************************************************)
(*************************** Package BRE.M ***************************)
(*************************************************************************)

BeginPackage["BRE`"]

Clear[ seriesrr, grr, crr, urr ,frr ]

BRE::usage="is a package that contains the definitions of:

- the series of functions
- the generating function
- the  modified moments of the functional c , ci=c(Ui(x))
- the polynomials Ui
- the elementary functions fi

, that the package BRR.M needs."

seriesrr::usage="seriesrr[t] is the closed form of the
                 series of functions"

grr::usage="grr[x,t] is the closed form of the generating
            function"

crr::usage="crr[i] is the i-th  modified moment of
            the functional c"

urr::usage="urr[i,x] is the polynomial corresponding
            to the moment ci"

frr::usage="frr[i,t] is the i-th elementary function"


Begin["`Private`"]


(****************************** EXAMPLE 1 ********************************)
(*****   initialfrr[ i, f[x], x ] = f[-1/(i + 1)]  ; i=0,1,...   ******)

seriesrr[t_]:=Log[1+t]/t ;
grr[x_,t_]:=1/(1-x*t) ;
crr[i_]:= crr[i]= (-1)^i/(i+1) ;
urr[i_,x_]:= x^i ;
frr[0,t_]:= 1;
frr[i_,t_]:= t^i ;


End[ ]

EndPackage[ ]
```

```
(*************************************************************************)
(************************** Package BRIF.M ***************************)
(*************************************************************************)

BeginPackage["BRIF`"]

Clear[ initialfrr ]

BRIF::usage="is a package, that contains the definitions
             of the initial functionals, that the package BRR.M
             needs"

initialfrr::usage="initialfrr[i,body,var] is the definition of
                   the i-th initial functional applied to Function[var,body]"


Begin["`Private`"]

initialfrr[i_,body_,var_] := initialfrr[i,body,var]=
                        Function[var,body][-1/(i + 1)] ;


End[ ]

EndPackage[ ]
```

E

Identités de Sylvester et de Schweins dans un espace vectoriel

Soient $b_1, \ldots, b_n$ des éléments d'un espace vectoriel sur un corps K, et a_{ij} des éléments de K.

L'identité de Sylvester est la suivante :

$$\begin{vmatrix} b_1 & \cdots & b_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-1,1} & \cdots & a_{n-1,n} \end{vmatrix} \begin{vmatrix} a_{12} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,2} & \cdots & a_{n-2,n-1} \end{vmatrix} = \tag{E.1}$$

$$= \begin{vmatrix} b_1 & \cdots & b_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} \begin{vmatrix} a_{12} & \cdots & a_{1,n} \\ \cdots & & \cdots \\ a_{n-1,2} & \cdots & a_{n-1,n} \end{vmatrix} - \begin{vmatrix} b_2 & \cdots & b_n \\ a_{12} & \cdots & a_{1,n} \\ \cdots & & \cdots \\ a_{n-2,2} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} \end{vmatrix}.$$

Soient $c_1, \ldots, c_n$ et $e_1, \ldots, e_n$ des éléments de K. L'identité de Schweins est la suivante :

$$\begin{vmatrix} b_1 & \cdots & b_n \\ c_1 & \cdots & c_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ e_1 & \cdots & e_{n-1} \end{vmatrix} = \tag{E.2}$$

$$= \begin{vmatrix} b_1 & \cdots & b_n \\ a_{11} & \cdots & a_{1,n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ e_1 & \cdots & e_n \end{vmatrix} \begin{vmatrix} c_1 & \cdots & c_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} - \begin{vmatrix} b_1 & \cdots & b_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} \begin{vmatrix} c_1 & \cdots & c_n \\ a_{11} & \cdots & a_{1,n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ e_1 & \cdots & e_n \end{vmatrix}.$$

Etant donné que tout corps est un espace vectoriel sur lui même, les identités de Sylvester et de Schweins sont encore valables si $b_1, \ldots, b_n$ appartiennent à K.

L'identité de Schweins est aussi vérifiée quand $b_1, \ldots, b_n$ sont des éléments de K et $c_1, \ldots, c_n$ des éléments d'un espace vectoriel sur K.

Nous pouvons facilement obtenir des formes équivalentes de l'identité de Schweins où les vecteurs $(b_1, \ldots, b_n)$, $(b_1, \ldots, b_{n-1})$, $(c_1, \ldots, c_n)$, $(c_1, \ldots, c_{n-1})$, $(e_1, \ldots, e_n)$ et $(e_1, \ldots, e_{n-1})$

occupent des places différentes. Pour cela, nous rappelons que le déterminant d'une matrice change de signe quand on permute des lignes ou des colonnes, et que les déterminants d'une matrice et de sa transposée coïncident.

Dans le premier membre de l'égalité (E.2), plaçons le vecteur $(e_1, \ldots, e_{n-1})$ comme première ligne; ceci correspond à faire $n-2$ permutations élémentaires, et nous devons multiplier la matrice correspondante par $(-1)^{n-2}$. Dans le deuxième membre de l'égalité (E.2) plaçons les deux vecteurs $(e_1, \ldots, e_n)$ comme deuxièmes lignes; ceci correspond à faire dans les deux cas $n-2$ permutations élémentaires, et nous devons multiplier les matrices correspondantes par $(-1)^{n-2}$. Nous obtenons ainsi la version suivante de l'identité de Schweins :

$$\begin{vmatrix} b_1 & \cdots & b_n \\ c_1 & \cdots & c_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} e_1 & \cdots & e_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} = \tag{E.3}$$

$$== \begin{vmatrix} b_1 & \cdots & b_n \\ e_1 & \cdots & e_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} c_1 & \cdots & c_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} - \begin{vmatrix} b_1 & \cdots & b_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} \begin{vmatrix} c_1 & \cdots & c_n \\ e_1 & \cdots & e_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \end{vmatrix}.$$

Dans chaque matrice des l'égalités (E.2) et (E.3), inversons l'ordre des colonnes. Il est facile de voir que pour inverser l'ordre des lignes ou des colonnes d'une matice de dimension n, nous devons faire $\sum_{i=1}^{n-1} i = n(n-1)/2$ permutations élémentaires et multiplier la matrice correspondante par $(-1)^{n(n-1)/2}$. Nous obtenons ainsi les versions suivantes de l'identité de Schweins :

$$\begin{vmatrix} b_n & \cdots & b_1 \\ c_n & \cdots & c_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ e_{n-1} & \cdots & e_1 \end{vmatrix} = \tag{E.4}$$

$$= \begin{vmatrix} b_n & \cdots & b_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ e_n & \cdots & e_1 \end{vmatrix} \begin{vmatrix} c_{n-1} & \cdots & c_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} - \begin{vmatrix} b_{n-1} & \cdots & b_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_n & \cdots & c_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ e_n & \cdots & e_1 \end{vmatrix}.$$

$$\begin{vmatrix} b_n & \cdots & b_1 \\ c_n & \cdots & c_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} e_{n-1} & \cdots & e_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} = \tag{E.5}$$

$$= \begin{vmatrix} b_n & \cdots & b_1 \\ e_n & \cdots & e_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_{n-1} & \cdots & c_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} - \begin{vmatrix} b_{n-1} & \cdots & b_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_n & \cdots & c_1 \\ e_n & \cdots & e_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \end{vmatrix}.$$

Dans le premier membre de l'égalité (E.2), plaçons les vecteurs $(b_1, \ldots, b_n)$ et $(c_1, \ldots, c_n)$ comme dernière et avant-dernière lignes; ceci correspond à faire respectivement $n-1$ et $n-2$ permutations élémentaires, et nous devons multiplier la matrice correspondante par $(-1)^{2n-3}$. Dans le deuxième membre de (E.2), plaçons les vecteurs $(b_1, \ldots, b_n)$, $(c_1, \ldots, c_{n-1})$, $(b_1, \ldots, b_{n-1})$ et $(c_1, \ldots, c_n)$ comme dernières lignes; ceci correspond à faire respectivement $n-1$, $n-2$, $n-2$ et $n-1$ permutations élémentaires, et nous devons multiplier les matrices correspondantes par $(-1)^{n-1}$, $(-1)^{n-2}$, $(-1)^{n-2}$ et $(-1)^{n-1}$. Nous obtenons ainsi la version suivante de l'identité de Schweins :

$$\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ c_1 & \cdots & c_n \\ b_1 & \cdots & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ e_1 & \cdots & e_{n-1} \end{vmatrix} = \tag{E.6}$$

$$= \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ e_1 & \cdots & e_n \\ b_1 & \cdots & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ c_1 & \cdots & c_{n-1} \end{vmatrix} - \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ b_1 & \cdots & b_{n-1} \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ e_1 & \cdots & e_n \\ c_1 & \cdots & c_n \end{vmatrix}.$$

Dans l'égalité (E.6), plaçons les vecteurs $(e_1, \ldots, e_n)$ et $(e_1, \ldots, e_{n-1})$ comme premières lignes. Nous obtenons ainsi la version suivante de l'identité de Schweins :

$$\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ c_1 & \cdots & c_n \\ b_1 & \cdots & b_n \end{vmatrix} \begin{vmatrix} e_1 & \cdots & e_{n-1} \\ a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \end{vmatrix} = \tag{E.7}$$

$$= \begin{vmatrix} e_1 & \cdots & e_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ b_1 & \cdots & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ c_1 & \cdots & c_{n-1} \end{vmatrix} - \begin{vmatrix} a_{11} & \cdots & a_{1,n-1} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n-1} \\ b_1 & \cdots & b_{n-1} \end{vmatrix} \begin{vmatrix} e_1 & \cdots & e_n \\ a_{11} & \cdots & a_{1n} \\ \cdots & & \cdots \\ a_{n-2,1} & \cdots & a_{n-2,n} \\ c_1 & \cdots & c_n \end{vmatrix}.$$

Dans chaque matrice des égalités (E.6) et (E.7), inversons l'ordre des colonnes. Nous obtenons les deux versions suivantes de l'identité de Schweins :

$$\begin{vmatrix} a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ c_n & \cdots & c_1 \\ b_n & \cdots & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ e_{n-1} & \cdots & e_1 \end{vmatrix} = \tag{E.8}$$

$$= \begin{vmatrix} a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ e_n & \cdots & e_1 \\ b_n & \cdots & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ c_{n-1} & \cdots & c_1 \end{vmatrix} - \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ b_{n-1} & \cdots & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ e_n & \cdots & e_1 \\ c_n & \cdots & c_1 \end{vmatrix}.$$

$$\begin{vmatrix} a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ c_n & \cdots & c_1 \\ b_n & \cdots & b_1 \end{vmatrix} \begin{vmatrix} e_{n-1} & \cdots & e_1 \\ a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \end{vmatrix} = \tag{E.9}$$

$$= \begin{vmatrix} e_n & \cdots & e_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ b_n & \cdots & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ c_{n-1} & \cdots & c_1 \end{vmatrix} - \begin{vmatrix} a_{1,n-1} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n-1} & \cdots & a_{n-2,1} \\ b_{n-1} & \cdots & b_1 \end{vmatrix} \begin{vmatrix} e_n & \cdots & e_1 \\ a_{1n} & \cdots & a_{11} \\ \cdots & & \cdots \\ a_{n-2,n} & \cdots & a_{n-2,1} \\ c_n & \cdots & c_1 \end{vmatrix}.$$

Dans l'égalité (E.2), remplaçons chaque matrice par sa transposée. Nous obtenons la version suivante de l'identité de Schweins :

$$\begin{vmatrix} b_1 & c_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & \vdots & & \vdots \\ b_n & c_n & a_{1,n} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & e_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & e_{n-1} \end{vmatrix} = \tag{E.10}$$

$$= \begin{vmatrix} b_1 & a_{11} & \cdots & a_{n-2,1} & e_1 \\ \vdots & \vdots & & \vdots & \vdots \\ b_n & a_{1,n} & \cdots & a_{n-2,n} & e_n \end{vmatrix} \begin{vmatrix} c_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots \\ c_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} -$$

$$-\begin{vmatrix} b_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots \\ b_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} \begin{vmatrix} c_1 & a_{11} & \cdots & a_{n-2,1} & e_1 \\ \vdots & \vdots & & \vdots & \vdots \\ c_n & a_{1n} & \cdots & a_{n-2,n} & e_n \end{vmatrix}.$$

Dans le premier membre de l'égalité (E.10), plaçons le vecteur $(e_1, \ldots, e_{n-1})^T$ comme première colonne. Dans le deuxième membre de (E.10), plaçons les deux vecteurs $(e_1, \ldots, e_n)^T$ comme deuxièmes colonnes. Nous obtenons ainsi la version suivante de l'identité de Schweins :

$$\begin{vmatrix} b_1 & c_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & \vdots & & \vdots \\ b_n & c_n & a_{1,n} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} e_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & & \vdots & \vdots \\ e_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} = \tag{E.11}$$

$$= \begin{vmatrix} b_1 & e_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots & \vdots \\ b_n & e_n & a_{1,n} & \cdots & a_{n-2,n} \end{vmatrix} \begin{vmatrix} c_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots \\ c_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} -$$

$$- \begin{vmatrix} b_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots \\ b_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} \begin{vmatrix} c_1 & e_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & \vdots & & \vdots \\ c_n & e_n & a_{1n} & \cdots & a_{n-2,n} \end{vmatrix}.$$

Dans chaque matrice des égalités (E.10) et (E.11), inversons l'ordre des lignes. Nous obtenons ainsi les versions suivantes de l'identité de Schweins :

$$\begin{vmatrix} b_n & c_n & a_{1,n} & \cdots & a_{n-2,n} \\ \vdots & \vdots & \vdots & & \vdots \\ b_1 & c_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & e_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & e_1 \end{vmatrix} = \tag{E.12}$$

$$= \begin{vmatrix} b_n & a_{1n} & \cdots & a_{n-2,n} & e_n \\ \vdots & \vdots & & \vdots & \vdots \\ b_1 & a_{11} & \cdots & a_{n-2,1} & e_1 \end{vmatrix} \begin{vmatrix} c_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & \vdots & & \vdots \\ c_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} -$$

$$- \begin{vmatrix} b_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & \vdots & & \vdots \\ b_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_n & a_{1n} & \cdots & a_{n-2,n} & e_n \\ \vdots & \vdots & & \vdots & \vdots \\ c_1 & a_{11} & \cdots & a_{n-2,1} & e_1 \end{vmatrix}.$$

$$\begin{vmatrix} b_n & c_n & a_{1n} & \cdots & a_{n-2,n} \\ \vdots & \vdots & \vdots & & \vdots \\ b_1 & c_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} e_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & & \vdots & \vdots \\ e_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} = \tag{E.13}$$

$$= \begin{vmatrix} b_n & e_n & a_{1,n} & \cdots & a_{n-2,n} \\ \vdots & \vdots & & \vdots & \vdots \\ b_1 & e_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & \vdots & & \vdots \\ c_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} -$$

$$-\begin{vmatrix} b_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & \vdots & & \vdots \\ b_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} \begin{vmatrix} c_n & e_n & a_{1n} & \cdots & a_{n-2,n} \\ \vdots & \vdots & & & \vdots \\ c_1 & e_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix}.$$

Dans le premier membre de l'égalité (E.10), plaçons les vecteurs $(b_1,\ldots,b_n)^T$ et $(c_1,\ldots,c_n)^T$ comme dernière et avant-dernière colonnes. Dans le deuxième membre de (E.10), plaçons les vecteurs $(b_1,\ldots,b_n)^T$, $(c_1,\ldots,c_{n-1})^T$, $(b_1,\ldots,b_{n-1})^T$ et $(c_1,\ldots,c_n)^T$ comme dernières colonnes. Nous obtenons ainsi la version suivante de l'identité de Schweins :

$$\begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & c_1 & b_1 \\ \vdots & & \vdots & \vdots & \vdots \\ a_{1n} & \cdots & a_{n-2,n} & c_n & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & e_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & e_{n-1} \end{vmatrix} = \tag{E.14}$$

$$= \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & e_1 & b_1 \\ \vdots & & \vdots & \vdots & \vdots \\ a_{1n} & \cdots & a_{n-2,n} & e_n & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & c_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & c_{n-1} \end{vmatrix} -$$

$$- \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & b_{n-1} \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & e_1 & c_1 \\ \vdots & & \vdots & \vdots & \vdots \\ a_{1n} & \cdots & a_{n-2,n} & e_n & c_n \end{vmatrix}.$$

Dans l'égalité E.14, plaçons les vecteurs $(e_1,\ldots,e_n)^T$ et $(e_1,\ldots,e_{n-1})^T$ comme premières colonnes. Nous obtenons la version suivante de l'identité de Schweins :

$$\begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & c_1 & b_1 \\ \vdots & & \vdots & \vdots & \vdots \\ a_{1,n} & \cdots & a_{n-2,n} & c_n & b_n \end{vmatrix} \begin{vmatrix} e_1 & a_{11} & \cdots & a_{n-2,1} \\ \vdots & \vdots & & \vdots \\ e_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \end{vmatrix} = \tag{E.15}$$

$$= \begin{vmatrix} e_1 & a_{11} & \cdots & a_{n-2,1} & b_1 \\ \vdots & \vdots & & \vdots & \vdots \\ e_n & a_{1,n} & \cdots & a_{n-2,n} & b_n \end{vmatrix} \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & c_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & c_{n-1} \end{vmatrix} -$$

$$- \begin{vmatrix} a_{11} & \cdots & a_{n-2,1} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{1,n-1} & \cdots & a_{n-2,n-1} & b_{n-1} \end{vmatrix} \begin{vmatrix} e_1 & a_{11} & \cdots & a_{n-2,1} & c_1 \\ \vdots & \vdots & & \vdots & \vdots \\ e_n & a_{1n} & \cdots & a_{n-2,n} & c_n \end{vmatrix}.$$

Dans chaque matrice des égalités (E.14) et (E.15), inversons l'ordre des lignes. Nous obtenons les versions suivantes des l'identités de Schweins :

$$\begin{vmatrix} a_{1n} & \cdots & a_{n-2,n} & c_n & b_n \\ \vdots & & \vdots & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & c_1 & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & e_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & e_1 \end{vmatrix} = \tag{E.16}$$

$$= \begin{vmatrix} a_{1n} & \cdots & a_{n-2,n} & b_n & e_n \\ \vdots & & \vdots & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & b_1 & e_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & c_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & c_1 \end{vmatrix} -$$

$$-\begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & b_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & b_1 \end{vmatrix} \begin{vmatrix} a_{1n} & \cdots & a_{n-2,n} & c_n & e_n \\ \vdots & \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & c_1 & e_1 \end{vmatrix}.$$

$$\begin{vmatrix} a_{1,n} & \cdots & a_{n-2,n} & c_n & b_n \\ \vdots & & \vdots & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & c_1 & b_1 \end{vmatrix} \begin{vmatrix} e_{n-1} & a_{1,n-1} & \cdots & a_{n-2,n-1} \\ \vdots & \vdots & & \vdots \\ e_1 & a_{11} & \cdots & a_{n-2,1} \end{vmatrix} = \tag{E.17}$$

$$= \begin{vmatrix} e_n & a_{1,n} & \cdots & a_{n-2,n} & b_n \\ \vdots & \vdots & & \vdots & \vdots \\ e_1 & a_{11} & \cdots & a_{n-2,1} & b_1 \end{vmatrix} \begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & c_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & c_1 \end{vmatrix} -$$

$$-\begin{vmatrix} a_{1,n-1} & \cdots & a_{n-2,n-1} & b_{n-1} \\ \vdots & & \vdots & \vdots \\ a_{11} & \cdots & a_{n-2,1} & b_1 \end{vmatrix} \begin{vmatrix} e_n & a_{1n} & \cdots & a_{n-2,n} & c_n \\ \vdots & \vdots & & \vdots & \vdots \\ e_1 & a_{11} & \cdots & a_{n-2,1} & c_1 \end{vmatrix}.$$

Bibliography

[1] C. BREZINSKI [1979]: Rational approximation to formal power series, J. Approx. Theory, 25, 295-317.

[2] C. BREZINSKI [1980]: *Padé-Type Approximation and General Orthogonal Polynomials*, Birkhäuser-Verlag, Basel.

[3] C. BREZINSKI [1991]: *Biorthogonality and its Applications to Numerical Analysis*, Marcel Dekker, New York.

[4] C. BREZINSKI []: A unified approach to various orthogonalities, Ann. Fac. Sci. Toulouse, to appear.

[5] C. BREZINSKI, M. REDIVO ZAGLIA [1991]: *Extrapolation Methods. Theory and Practice.*, North-Holland, Amsterdam.

[6] C. BREZINSKI, M. REDIVO ZAGLIA, and H. SADOK [1992]: A breakdown-free Lanczos type algorithm for solving linear systems, Numer. Math., to appear.

[7] Z. DA ROCHA [1992]: Implementation of the recurrence relations of biorthogonality. In C. Brezinski, P. González-Vera and N. Hayek-Calil, eds., Extrapolation and Rational Approximation. Tenerife, Numerical Algorithms ,3 ,173-183 , Basel. J. C. Baltzer.

[8] P.J. DAVIS [1975]: *Interpolation and Approximation*, Dover, New York.

[9] A. DRAUX [1983]: *Polynômes Orthogonaux Formels. Applications.*, LNM 974 Springer-Verlag , Berlin.

[10] J. VAN ISEGHEM [1987]: Vector orthogonal relations. Vector QD-algorithm., J. Comput. Appl. Math.,19 ,141-150 ,Elsevier Science Publishers, North-Holland.

[11] J. VAN ISEGHEM []: Recurrence relations on the table of vector orthogonal polynomials, to appear.

[12] LEBEDEV [1965]: *Special functions and their applications.*, Prentice-Hall, inc.

[13] S. WOLFRAM [1988]:*Mathematica TM . A system for doing Mathematics by Computer*, Addison-Wesley, Reading.

[14] S. PASZKOWSKI [1984]: Polynômes et séries de Tchebichev, Publication ANO - 140, Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3, Université des Sciences et Technologies de Lille, F-59655 V. N. d'Ascq Cedex, France.