USTL

LABORATOIRE D'ANALYSE NUMERIQUE ET D'OPTIMISATION

n° d'ordre : 1853

THESE

Nouveau régime

présentée à
l'Université des Sciences et Technologies de Lille

pour obtenir le titre de

DOCTEUR en MATHEMATIQUES
par

Sonia KRZYWORZCKA

EXTENSION DES METHODES DE TYPE LANCZOS A LA RESOLUTION DE SYSTEMES NON LINEAIRES

soutenue le 19 décembre 1996 devant la commission d'examen

Membres du jury

Président :	C. BREZINSKI, Professeur, Un. Lille 1
Rapporteurs :	A. BULTHEEL, Professeur, Leuven, Belgique
	J. DELLA DORA, Professeur, IMAG, Grenoble
Membres :	B. GERMAIN BONNE, Professeur, Un. Lille I
	H. SADOK, Professeur, Un. du Littoral

Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3, USTL ,
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
M. 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

REMERCIEMENTS

Je tiens tout d'abord à remercier tout particulièrement Monsieur C. Brezinski, Professeur à l'université des sciences et technologies de Lille, pour m'avoir toujours soutenue dans ce travail. Ses conseils, ses réponses à mes interrogations et toute son attention m'ont permis d'aboutir.

Je tiens également à remercier Monsieur A. Bultheel, Professeur à l'université de Leuven (Belgique), et Monsieur J. Della Dora, Professeur au LMC/IMAG à Grenoble, pour avoir bien voulu juger ce travail et apporter des suggestions.

Mes remerciements vont aussi à Monsieur B. Germain-Bonne, Professeur à l'université des sciences et technologies de Lille, et à Monsieur H. Sadok, Professeur à l'université du Littoral, pour avoir accepté de participer à mon jury.

Je tiens enfin à exprimer toute ma gratitude envers ma famille et mes amis pour l'écoute et le soutien qu'ils m'ont prodigué.

TABLE DES MATIERES

INTRODUCTION

En analyse numérique comme dans de nombreux domaines scientifiques nous sommes souvent amenés à résoudre des systèmes d'équations non linéaires et en particulier des problèmes de recherche de points fixes. Dans cette thèse nous nous intéressons à cette recherche par le biais de nouvelles méthodes de résolution.

En 1950 Lanczos a proposé un algorithme permettant, entre autres, la résolution de systèmes linéaires du type $Ax = b$. La méthode obtenue construit une suite de vecteurs $(x_k)_{k\in I\!N}$ qui donne théoriquement la solution exacte du système en un nombre fini d'itérations inférieur à la dimension de celui-ci. Cette méthode a été liée à la théorie des polynômes orthogonaux grâce à la relation $r_k = P_k(A)r_0$ où P_k est le polynôme de degré au plus k appartenant à la famille de polynômes orthogonaux par rapport à la fonctionnelle linéaire c de moments $c(\xi^i) = (y, A^i r_0)$ avec y un vecteur arbitraire non nul. Les différentes relations de récurrence utilisées pour calculer ces polynômes ont alors permis d'aboutir à différents algorithmes d'implémentation de la méthode de Lanczos connus sous le nom de méthodes de type Lanczos dont Lanczos-Orthodir, Lanczos-Orthores, Lanczos-Orthomin.
Le CGS (Conjugate Gradient Squared) résoud aussi des systèmes d'équations linéaires ; il est caractérisé à l'itération k par le vecteur résidu $r_k = P_k(A)^2 r_0$. Les méthodes de type Lanczos et le CGS étant largement étudiés et étant efficaces en ce qui concerne les systèmes d'équations linéaires il est intéressant de voir dans le premier chapitre comment ils peuvent être étendus au cas non linéaire. Pour cela une interprétation de ces algorithmes basée sur des transformations de suites de vecteurs est utilisée. Les algorithmes de recherche de point fixe ainsi obtenus sont à convergence quadratique locale.

Dans le second chapitre nous travaillons sur les méthodes CGM (conjugate gradient methods), méthodes caractérisées par un vecteur résidu à l'itération k vérifiant $r_k = Q_k(A)P_k(A)r_0$ où Q_k est un polynôme donné. Le lien entre l'epsilon algorithme topologique et des méthodes CGM nous permet de construire les méthodes que nous appelons NCGM (nonlinear CGM). Nous donnons un théorème de convergence quadratique locale et des exemples

numériques pour chaque nouvelle méthode écrite. Par soucis d'économie de calculs nous abordons ensuite les méthodes incomplètes issues des méthodes nouvellement construites dans les chapitres I et II.

Le but du troisième chapitre est d'utiliser la théorie des polynômes bi-orthogonaux pour obtenir des algorithmes de résolution de systèmes non linéaires. Ces algorithmes sont les extensions d'algorithmes connus dans le domaine linéaire (RRE, MPE, MMPE) et dont nous rappelons le lien avec les méthodes de type Lanczos. Une étude numérique et de convergence est faite. Enfin nous comparons les méthodes non linéaires de ces trois chapitres avec une méthode non linéaire d'Henrici et une méthode de Newton inexacte.

Le dernier chapitre est consacré à l'étude numérique des méthodes hybrides non linéaires mettant en oeuvre les nouvelles méthodes étudiées dans ce travail. En effet en utilisant des suites de vecteurs construites par deux algorithmes non linéaires choisis parmis les trois chapitres précédents nous construisons une troisième suite de vecteurs : la suite hybride, dont nous testons les résultats.

Chapitre I
Extension des méthodes de type Lanczos et du CGS à la résolution de systèmes non linéaires

1 Introduction et notations: deux transformations de suites de vecteurs

Donnons tout d'abord des notations. On note $(.,.)$ le produit scalaire dans $\mathbb{C}$ tel que $(\alpha x, \beta y) = \alpha\overline{\beta}(x,y) = \alpha\overline{\beta}\sum_{i=1}^{p} x_i\overline{y_i}$ pour α et β des nombres complexes et, x et y deux vecteurs de $\mathbb{C}^p$. Nous allons utiliser la norme euclidienne sur les vecteurs $\|x\|$, et la norme sur les matrices $\|A\| = \max_{i=1\ldots p}\sum_{j=1}^{k}|a_{ij}|$, où A est une matrice $p \times k$ dont les coefficients sont notés a_{ij}.
Soit $(u_n)_{n\in\mathbb{N}}$ une suite de vecteurs et $(c_n)_{n\in\mathbb{N}}$ une suite de scalaires.

Nous définissons comme dans [16] les quantités suivantes:

- $$\varepsilon_{i,2k}^{(n)} = \frac{\begin{vmatrix} u_{n+i} & \cdots & u_{n+i+k} \\ c_n & \cdots & c_{n+k} \\ \vdots & \ddots & \vdots \\ c_{n+k-1} & \cdots & c_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_n & \cdots & c_{n+k} \\ \vdots & \ddots & \vdots \\ c_{n+k-1} & \cdots & c_{n+2k-1} \end{vmatrix}}$$

- $$G_j^{(i)}(k,n) = \frac{\begin{vmatrix} \varepsilon_{i,2k}^{(n)} & \cdots & \varepsilon_{i+j,2k}^{(n)} \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \ddots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \ddots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}.$$

Dans chacune de ces deux formules le déterminant du numérateur représente en fait le vecteur obtenu en développant ce déterminant selon sa première ligne suivant les règles habituelles.

En utilisant l'identité de Sylvester, nous obtenons le calcul récursif suivant de ces vecteurs:

$$\begin{cases} \varepsilon_{i,0}^{(n)} = u_{n+i} \\ \varepsilon_{i,2k}^{(n)} = \varepsilon_{i,2k-2}^{(n)} - r_k^{(n)} \frac{\varepsilon_{i,2k-2}^{(n+1)} - \varepsilon_{i,2k-2}^{(n)}}{r_k^{(n+1)} - r_k^{(n)}} \end{cases}$$

et

$$\begin{cases} G_0^{(i)}(k,n) = \varepsilon_{i,2k}^{(n)} \\ G_j^{(i)}(k,n) = G_{j-1}^{(i)}(k,n) - r_j^{(n+i)} \frac{G_{j-1}^{(i+1)}(k,n) - G_{j-1}^{(i)}(k,n)}{r_j^{(n+i+1)} - r_j^{(n+i)}} \end{cases} .$$

Les quantités auxiliaires $r_k^{(n)}$ sont calculées par le rs-algorithme de Pye et Atchison [40], dont les règles sont:

$$\begin{cases} s_0^{(n)} = 1 \quad \text{et} \quad r_1^{(n)} = c_n \quad \text{pour } n = 0, 1, \ldots \\ s_{k+1}^{(n)} = s_k^{(n+1)} \left(\frac{r_{k+1}^{(n+1)}}{r_{k+1}^{(n)}} - 1 \right) \quad \text{pour } k, n = 0, 1, \ldots \\ r_{k+2}^{(n)} = r_{k+1}^{(n+1)} \left(\frac{s_{k+1}^{(n+1)}}{s_{k+1}^{(n)}} - 1 \right) \quad \text{pour } k, n = 0, 1, \ldots . \end{cases}$$

2 Lien entre la première transformation de suites de vecteurs et l'epsilon algorithme topologique

Rappelons tout d'abord en quoi consiste l'epsilon algorithme topologique [6]. Nous partons d'une suite de vecteurs $(u_n)_{n \in \mathbb{N}}$ vérifiant:

$$(\mathbf{h}) \quad a_0(u_n - u) + \cdots + a_k(u_{n+k} - u) = 0 \quad \forall n \in \mathbb{N}$$

où les a_i sont des scalaires réels tels que a_k est non nul et $a_0 + \cdots + a_k = 1$. Soit y un vecteur arbitraire non nul, alors nous obtenons:

$$a_0(y, \Delta u_n) + \cdots + a_k(y, \Delta u_{n+k}) = 0 \quad \forall n \in \mathbb{N}.$$

En résolvant alors le système:

$$\begin{cases} a_0 + \cdots + a_k = 1 \\ a_0(y, \Delta u_n) + \cdots + a_k(y, \Delta u_{n+k}) = 0 \\ \vdots \\ a_0(y, \Delta u_{n+k-1}) + \cdots + a_k(y, \Delta u_{n+2k-1}) = 0 \end{cases}$$

nous obtenons

$$a_0u_n + \cdots + a_ku_{n+k} = \frac{\begin{vmatrix} u_n & \cdots & u_{n+k} \\ (y,\Delta u_n) & \cdots & (y,\Delta u_{n+k}) \\ \vdots & \ddots & \vdots \\ (y,\Delta u_{n+k-1}) & \cdots & (y,\Delta u_{n+2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (y,\Delta u_n) & \cdots & (y,\Delta u_{n+k}) \\ \vdots & \ddots & \vdots \\ (y,\Delta u_{n+k-1}) & \cdots & (y,\Delta u_{n+2k-1}) \end{vmatrix}}.$$

Nous notons alors

$$e_{i,k}(u_n) = \frac{\begin{vmatrix} u_{n+i} & \cdots & u_{n+k+i} \\ (y,\Delta u_n) & \cdots & (y,\Delta u_{n+k}) \\ \vdots & \ddots & \vdots \\ (y,\Delta u_{n+k-1}) & \cdots & (y,\Delta u_{n+2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (y,\Delta u_n) & \cdots & (y,\Delta u_{n+k}) \\ \vdots & \ddots & \vdots \\ (y,\Delta u_{n+k-1}) & \cdots & (y,\Delta u_{n+2k-1}) \end{vmatrix}},$$

ceci définit l'epsilon algorithme topologique.

Nous pouvons noter que dans le cas d'une suite $(u_n)_{n\in\mathbb{N}}$ vérifiant l'hypothèse **(h)** nous avons $\forall n \in \mathbb{N}, e_{i,k}(u_n) = u$, où u est un vecteur constant par rapport à n ; ce cas particulier de suite $(u_n)_{n\in\mathbb{N}}$ est utilisé pour construire ce que nous notons $e_{i,k}(u_n)$; ensuite nous utilisons la suite $e_{i,k}(u_n)$ associée à une suite quelconque $(u_n)_{n\in\mathbb{N}}$; les vecteurs $e_{i,k}(u_n)$ varient alors selon la valeur de n.

Ainsi dans le cas particulier où $c_n = (y,\Delta u_n)\ \forall n$ avec y un vecteur arbitraire non nul, nous avons

$$\epsilon_{i,k}(u_n) = \varepsilon_{i,2k}^{(n)}.$$

Les vecteurs $\varepsilon_{0,2k}^{(n)}$ peuvent alors être calculés comme les vecteurs $e_{0,k}(u_n)$ par le TEA1(premier epsilon algorithme topologique), dont les règles sont:

$$\varepsilon_{0,-1}^{(n)} = 0 \quad \text{et} \quad \varepsilon_{0,0}^{(n)} = u_n \quad \text{pour } n = 0,1,\ldots$$

$$\varepsilon_{0,2k+1}^{(n)} = \varepsilon_{0,2k-1}^{(n+1)} + \frac{y}{(y, \varepsilon_{0,2k}^{(n+1)} - \varepsilon_{0,2k}^{(n)})} \quad \text{pour } n, k = 0, 1, \ldots$$

$$\varepsilon_{0,2k+2}^{(n)} = \varepsilon_{0,2k}^{(n+1)} + \frac{\varepsilon_{0,2k}^{(n+1)} - \varepsilon_{0,2k}^{(n)}}{(\varepsilon_{0,2k+1}^{(n+1)} - \varepsilon_{0,2k+1}^{(n)}, \varepsilon_{0,2k}^{(n+1)} - \varepsilon_{0,2k}^{(n)})} \quad \text{pour } n, k = 0, 1, \ldots.$$

Les vecteurs $\varepsilon_{k,2k}^{(n)}$ peuvent être calculés comme les vecteurs $e_{k,k}(u_n)$ par le TEA2 (second epsilon algorithme topologique), dont les règles sont:

$$\varepsilon_{-1,-1}^{(n)} = 0 \qquad \text{et} \qquad \varepsilon_{0,0}^{(n)} = u_n \quad \text{pour } n = 0, 1, \ldots$$

$$\varepsilon_{k,2k+1}^{(n)} = \varepsilon_{k-1,2k-1}^{(n+1)} + \frac{y}{(y, \varepsilon_{k,2k}^{(n+1)} - \varepsilon_{k,2k}^{(n)})} \quad \text{pour } n, k = 0, 1, \ldots$$

$$\varepsilon_{k+1,2k+2}^{(n)} = \varepsilon_{k,2k}^{(n+1)} + \frac{\varepsilon_{k,2k}^{(n+2)} - \varepsilon_{k,2k}^{(n+1)}}{(\varepsilon_{k,2k+1}^{(n+1)} - \varepsilon_{k,2k+1}^{(n)}, \varepsilon_{k,2k}^{(n+2)} - \varepsilon_{k,2k}^{(n+1)})} \text{ pour } n, k = 0, 1, \ldots$$

3 Les méthodes de type Lanczos et l'epsilon algorithme topologique

3.1 La méthode de Lanczos

La méthode de Lanczos [4, 34] est utilisée pour résoudre un système d'équations linéaires. Elle donne lieu à une méthode itérative qui donne la solution exacte en un nombre fini d'itérations inférieur à la dimension du système. Considerons le système suivant avec A une matrice carrée de $\mathbb{C}^p \times \mathbb{C}^p$, b et x des vecteurs de $\mathbb{C}^p$:

$$Ax = b.$$

La méthode de Lanczos pour résoudre ce système consiste en la construction d'une suite de vecteurs complexes $(x_k)_{k \in \mathbb{N}}$ comme suit:

- choisir x_0 et y deux vecteurs non nuls
- poser $r_0 = b - Ax_0$
- déterminer x_k tel que:

$$x_k - x_0 \in Span(r_0, Ar_0, \cdots, A^{k-1}r_0) = K_k(A, r_0)$$

$$r_k = b - Ax_k \perp Span(y, A^*y, \cdots, A^{*^{k-1}}y) = K_k(A^*, y).$$

La première condition ci-dessus s'écrit ainsi avec $\alpha_1, \alpha_2, \ldots, \alpha_k$ des coefficients réels :

$$x_k - x_0 = -\alpha_1 r_0 - \alpha_2 Ar_0 - \cdots - \alpha_k A^{k-1} r_0$$

et donc nous avons

$$r_k = r_0 + \alpha_1 Ar_0 + \alpha_2 A^2 r_0 + \cdots + \alpha_k A^k r_0.$$

La seconde condition donne les relations d'orthogonalité suivantes

$$(A^{*i}y, r_k) = 0 \quad \forall i = 0, \ldots, k-1$$

ce qui donne le système

$$\begin{cases} \alpha_1(y, Ar_0) + \cdots + \alpha_k(y, A^k r_0) = -(y, r_0) \\ \vdots \\ \alpha_1(A^{*^{k-1}}y, Ar_0) + \cdots + \alpha_k(A^{*^{k-1}}y, A^k r_0) = -(A^{*^{k-1}}y, r_0), \end{cases}$$

où A^* est la matrice transposée conjuguée de A.

Si le déterminant de ce système est non nul alors x_k existe et s'écrit

$$x_k = \frac{\begin{vmatrix} x_0 & -r_0 & \cdots & -A^{k-1}r_0 \\ (y, r_0) & (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \vdots & \cdots & \vdots \\ (A^{*^{k-1}}y, r_0) & (A^{*^{k-1}}y, Ar_0) & \cdots & (A^{*^{k-1}}y, A^k r_0) \end{vmatrix}}{\begin{vmatrix} (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \cdots & \vdots \\ (A^{*^{k-1}}y, Ar_0) & \cdots & (A^{*^{k-1}}y, A^k r_0) \end{vmatrix}}.$$

Si nous posons $P_k(\xi) = 1 + \alpha_1\xi + \cdots + \alpha_k\xi^k$ nous avons

$$P_k(\xi) = \frac{\begin{vmatrix} 1 & \xi & \cdots & \xi^k \\ (y, r_0) & (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \vdots & \cdots & \vdots \\ (A^{*k-1}y, r_0) & (A^{*k-1}y, Ar_0) & \cdots & (A^{*k-1}y, A^k r_0) \end{vmatrix}}{\begin{vmatrix} (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \cdots & \vdots \\ (A^{*k-1}y, Ar_0) & \cdots & (A^{*k-1}y, A^k r_0) \end{vmatrix}}$$

et de plus

$$r_k = P_k(A)r_0.$$

De plus, si nous posons

$$c_i = (y, A^i r_0) \quad i = 0, 1, \ldots$$

et si nous définissons la fonctionnelle linéaire c comme suit

$$c(\xi^i) = c_i \quad i = 0, 1, \ldots$$

alors nous avons

$$c(\xi^i P_k(\xi)) = 0 \quad i = 0, \ldots, k-1.$$

Ces conditions montrent que P_k est le polynôme de degré au plus k appartenant à la famille de polynômes orthogonaux par rapport à la fonctionnelle linéaire c [5].

Les différentes possibilités de calcul récursif des polynômes P_k conduisent à différentes procédures de calcul récursif de r_k et de x_k . De telles méthodes sont appelées les méthodes de type Lanczos. Nous pouvons citer par exemple le BCG (biconjugate gradient method) [25].

3.2 TEA1-Lanczos

Nous nous plaçons dans le cas particulier où $c_n = (y, \Delta u_n)$ avec des vecteurs u_n genérés, u_0 étant choisi, par

$$u_{n+1} = (I + A)u_n - b = Bu_n - b$$

avec B une matrice carrée $p \times p$ telle que $A = B - I$ est inversible et b est un vecteur de $\mathbb{C}^p$. Par récurrence sur k nous obtenons

$$\forall k \in \mathbb{N}, \forall i \in \mathbb{N}, \Delta^k u_i = \sum_{j=0}^{k} (-1)^j C_k^j u_{i+k-j}.$$

Posons $x_k = \varepsilon_{0,2k}^{(0)}$, alors nous avons $x_0 = u_0$ et $\Delta u_0 = Au_0 - b = -r_0$. Nous obtenons de plus

$$x_k = \frac{\begin{vmatrix} x_0 & \Delta u_0 & \cdots & \Delta^k u_0 \\ (y, \Delta u_0) & (y, \Delta^2 u_0) & \cdots & (y, \Delta^{k+1} u_0) \\ \vdots & \vdots & \cdots & \vdots \\ (y, \Delta^k u_0) & (y, \Delta^{k+1} u_0) & \cdots & (y, \Delta^{2k} u_0) \end{vmatrix}}{\begin{vmatrix} (y, \Delta^2 u_0) & \cdots & (y, \Delta^{k+1} u_0) \\ \vdots & \cdots & \vdots \\ (y, \Delta^{k+1} u_0) & \cdots & (y, \Delta^{2k} u_0) \end{vmatrix}};$$

en effet

$$\varepsilon_{0,2k}^{(0)} = \frac{\begin{vmatrix} u_0 & \cdots & u_k \\ c_0 & \cdots & c_k \\ \vdots & \ddots & \vdots \\ c_{k-1} & \cdots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_0 & \cdots & c_k \\ \vdots & \ddots & \vdots \\ c_{k-1} & \cdots & c_{2k-1} \end{vmatrix}};$$

nous gardons la première colonne de chacun des deux déterminants de ce quotient et nous notons col_j chaque $j^{\text{ième}}$ colonne $j = 0, \ldots, k$; nous remplaçons chaque colonne $j, j = k, \ldots, 1$, pour les deux déterminants, par la ' combinaison linéaire de colonnes ' suivantes

$$\sum_{i=0}^{j} (-1)^i C_j^i col_{j-i},$$

avec une notation erronnée mais utilisée pour plus de facilité. Nous arrivons

alors au quotient

$$\varepsilon_{0,2k}^{(0)} = \frac{\begin{vmatrix} u_0 & \Delta u_0 & \cdots & \Delta^k u_0 \\ (y,\Delta u_0) & (y,\Delta^2 u_0) & \cdots & (y,\Delta^{k+1} u_0) \\ (y,\Delta u_1) & (y,\Delta^2 u_1) & \cdots & (y,\Delta^{k+1} u_1) \\ \vdots & \vdots & \cdots & \vdots \\ (y,\Delta u_{k-1}) & (y,\Delta^2 u_{k-1}) & \cdots & (y,\Delta^{k+1} u_{k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & 0 & \cdots & 0 \\ (y,\Delta u_0) & (y,\Delta^2 u_0) & \cdots & (y,\Delta^{k+1} u_0) \\ (y,\Delta u_1) & (y,\Delta^2 u_1) & \cdots & (y,\Delta^{k+1} u_1) \\ \vdots & \vdots & \cdots & \vdots \\ (y,\Delta u_{k-1}) & (y,\Delta^2 u_{k-1}) & \cdots & (y,\Delta^{k+1} u_{k-1}) \end{vmatrix}};$$

nous faisons ensuite de même pour les lignes de ces déterminants ; nous notons $lig_i, i = 0,\ldots,k$ les $i^{\text{ièmes}}$ lignes d'un déterminant puis nous remplaçons chaque ligne lig_i pour $i = k,\ldots,2$ par la ' combinaison linéaire de lignes 'suivante

$$\sum_{j=0}^{i-1} (-1)^j C_{i-1}^j lig_{i-j};$$

nous obtenons alors exactement l'écriture de $x_k = \varepsilon_{0,2k}^{(0)}$ voulue.

De plus étant donnée la construction de la suite $(u_n)_{n\in\mathbb{N}}$ nous avons

$$\Delta^k u_n = A\Delta^{k-1} u_{n-1} = A^{k-1}\Delta u_0 = -A^{k-1} r_0.$$

En remplaçant, dans l'écriture de x_k, $\Delta^k u_0$ par $-A^{k-1} r_0$ nous obtenons ainsi le même rapport de déterminants que celui définissant précédemment l'itéré x_k de la méthode de Lanczos.

3.3 TEA2-Lanczos

$(u_n)_{n\in\mathbb{N}}$ est toujours la suite genérée par $u_{n+1} = (I+A)u_n - b$ avec $B = I+A$ une matrice carrée $p \times p$ inversible et b un vecteur de $\mathbb{C}^p$.

Nous avons alors cf.[17]

$$\Delta u_{n+1} = (I+A)\Delta u_n = (I+A)^{n+1}\Delta u_0$$

et

$$\Delta^k u_n = A\Delta^{k-1} u_n = A^{k-1}\Delta u_n.$$

Grâce à ces deux expressions nous obtenons

$$\Delta^j u_k = -(I+A)^k A^{j-1} r_0.$$

Nous posons en outre $x_k = \varepsilon^{(0)}_{k,2k}$ et $r_k = b - Ax_k$. En utilisant la définition de $\varepsilon^{(0)}_{k,2k}$ et les propriétés de l'opérateur Δ ci-dessus nous obtenons

$$r_k = \frac{\begin{vmatrix} (I+A)^k r_0 & (I+A)^k A r_0 & \cdots & (I+A)^k A^k r_0 \\ (y, r_0) & (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \vdots & \cdots & \vdots \\ (y, A^{k-1} r_0) & (y, A^k r_0) & \cdots & (y, A^{2k-1} r_0) \end{vmatrix}}{\begin{vmatrix} (y, Ar_0) & \cdots & (y, A^k r_0) \\ \vdots & \cdots & \vdots \\ (y, A^k r_0) & \cdots & (y, A^{2k-1} r_0) \end{vmatrix}}.$$

Nous voyons donc que $r_k = (I+A)^k P_k(A) r_0$.
Cette identité montre que, comme dans les algorithmes de type Lanczos, les polynômes P_k peuvent être calculés récursivement et nous obtenons alors des relations de récurrence pour les r_k. Nous pouvons de plus remarquer que si $\|I + A\|$ est strictement inférieure à 1 alors nous obtenons une méthode meilleure que la méthode de Lanczos où le vecteur résidu est $r_k = P_k(A) r_0$. Pour illustrer ceci nous allons construire de nouveaux algorithmes que nous appellerons TEA2-Orthodir, TEA2-Orthores, et TEA2-Orthomin.

3.3.1 TEA2-Orthodir

Lanczos-Orthodir [4, 34] est la méthode de type Lanczos qui utilise les relations de récurrence suivantes:

$$(i) \quad P_{k+1}(\xi) = P_k(\xi) - \lambda_k \xi P^{(1)}_k(\xi)$$

$$(ii) \quad P^{(1)}_k(\xi) = (\xi - a_k) P^{(1)}_{k-1}(\xi) - b_k P^{(1)}_{k-2}(\xi)$$

où les $P^{(1)}_k$ sont les polynômes unitaires tels que

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} d_1 & d_2 & \cdots & d_{k+1} \\ \vdots & \vdots & \cdots & \vdots \\ d_k & d_{k+1} & \cdots & d_{2k} \\ 1 & \xi & \cdots & \xi^k \end{vmatrix}}{\begin{vmatrix} d_1 & \cdots & d_k \\ \vdots & \cdots & \vdots \\ d_k & \cdots & d_{2k-1} \end{vmatrix}}$$

avec $d_i = d(\xi^i) = (A^{*^i} y, r_0)$.
$P_k^{(1)}$ est donc le polynôme unitaire de degré k appartenant à la famille de polynômes orthogonaux par rapport à la fonctionnelle linéaire $d^{(1)}$ définie par

$$d^{(1)}(\xi^i) = d_{i+1} \quad i = 0, 1, \ldots .$$

Nous posons $\varphi_k(\xi) = (1+\xi)^k P_k(\xi)$ et $\rho_k(\xi) = (1+\xi)^k P_k^{(1)}(\xi)$.
Les relations précédentes nous donnent alors

$$\varphi_{k+1}(\xi) = (1+\xi)[\varphi_k(\xi) - \lambda_k \xi \rho_k(\xi)]$$

et

$$\rho_k(\xi) = (1+\xi)[(\xi - a_k)\rho_{k-1}(\xi) - b_k(1+\xi)\rho_{k-2}(\xi)].$$

De plus nous avons $r_k = \varphi_k(A) r_0$, et nous posons $q_k = \rho_k(A) r_0$, d'où

$$r_{k+1} = (I + A)[r_k - \lambda_k A q_k]$$

$$q_k = (I + A)(A - a_k I) q_{k-1} - b_k (I + A) q_{k-2}.$$

De plus $r_k = b - A x_k$ d'où

$$x_{k+1} = x_k - r_k + \lambda_k (I + A) q_k.$$

Regardons maintenant comment calculer les coefficients a_k, b_k et λ_k. Nous utilisons pour cela les propriétés des polynômes orthogonaux P_k et $P_k^{(1)}$ suivantes:

$$d(\xi^i P_k(\xi)) = (y, A^i P_k(A) r_0) = 0 \text{ pour } i = 0, \cdots, k-1$$

$$d(\xi^{i+1} P_k^{(1)}(\xi)) = 0 \text{ pour } i = 0, \cdots, k-1.$$

En multipliant (i) par ξ^k et en lui appliquant l'application linéaire d nous avons

$$d(\xi^k P_{k+1}(\xi)) = d(\xi^k P_k(\xi)) - \lambda_k d(\xi^{k+1} P_k^{(1)}(\xi)).$$

Or $d(\xi^k P_{k+1}(\xi)) = 0$ ainsi

$$\lambda_k = \frac{d(\xi^k P_k(\xi))}{d(\xi^{k+1} P_k^{(1)}(\xi))} = \frac{d((1+\xi)^k P_k(\xi))}{d(\xi(1+\xi)^k P_k^{(1)}(\xi))} = \frac{(y, (I+A)^k P_k(A) r_0)}{(y, A(I+A)^k P_k^{(1)}(A) r_0)}.$$

donc

$$\lambda_k = \frac{(y, r_k)}{(y, Aq_k)}.$$

De même en multipliant (ii) par ξ^{k-1} puis en lui appliquant la fonctionnelle d nous avons

$$d(\xi^{k-1} P_k^{(1)}(\xi)) = d(\xi^k P_{k-1}^{(1)}(\xi)) - a_k d(\xi^{k-1} P_{k-1}^{(1)}(\xi)) - b_k d(\xi^{k-1} P_{k-2}^{(1)}(\xi)).$$

Grâce à la nullité de $d(\xi^{k-1} P_{k-1}^{(1)}(\xi))$ nous obtenons

$$b_k = \frac{d(\xi^k P_{k-1}^{(1)}(\xi))}{d(\xi^{k-1} P_{k-2}^{(1)}(\xi))} = \frac{d((1+\xi)^{k-1} \xi P_{k-1}^{(1)}(\xi))}{d(\xi(1+\xi)^{k-2} P_{k-2}^{(1)}(\xi))}.$$

$$b_k = \frac{(y, Aq_{k-1})}{(y, Aq_{k-2})}.$$

En utilisant (ii) multipliée par ξ^k nous avons

$$a_k = \frac{[d(\xi^{k+1} P_{k-1}^{(1)}(\xi)) - b_k d(\xi^k P_{k-2}^{(1)}(\xi))]}{d(\xi^k P_{k-1}^{(1)}(\xi))}.$$

D'où

$$a_k = \frac{[(y, A^2 q_{k-1}) - b_k (y, A^2 q_{k-2})]}{(y, Aq_{k-1})}.$$

En rassemblant toutes ces formules, nous obtenons finalement le TEA2-Orthodir algorithme:

- choisir x_0 et y
  poser $r_0 = b - Ax_0$, $q_0 = y$, $q_{-1} = y$

- pour $k = 0, 1, \cdots$ calculer

$$\lambda_k = \frac{(y, r_k)}{(y, Aq_k)}$$

$$x_{k+1} = x_k - r_k + \lambda_k (I + A) q_k$$

$$r_{k+1} = (I + A)[r_k - \lambda_k A q_k]$$

- si r_k est non nul alors calculer

$$b_{k+1} = \frac{(y, Aq_k)}{(y, Aq_{k-1})}$$

$$a_{k+1} = \frac{[(y, A^2 q_k) - b_{k+1}(y, A^2 q_{k-1})]}{(y, Aq_k)}$$

$$q_{k+1} = (I + A)(A - a_{k+1} I) q_k - b_{k+1}(I + A) q_{k-1}$$

3.3.2 TEA2-Orthores

Lanczos-Orthores utilise la relation de récurrence à trois termes suivante

$$(iii) \quad P_{k+1}(\xi) = -\eta_k [(\xi - \gamma_k) P_k(\xi) - \delta_k P_{k-1}(\xi)]$$

avec $P_0(\xi) = 1$ et $P_{-1}(\xi) = 0$.
Nous posons à nouveau $\varphi_k(\xi) = (1 + \xi)^k P_k(\xi)$. La relation de récurrence ci-dessus nous donne

$$\varphi_{k+1}(\xi) = -\eta_k (1 + \xi)[(\xi - \gamma_k) - \delta_k (1 + \xi) \varphi_{k-1}(\xi)].$$

Soit $r_k = \varphi_k(A) r_0$, nous avons alors

$$r_{k+1} = -\eta_k (I + A)[(A - \gamma_k I) r_k - \delta_k (I + A) r_{k-1}].$$

De plus, sachant que $r_k = b - A x_k$, nous obtenons

$$x_{k+1} = x_k + \eta_k [(A - \gamma_k I + I) r_k - \delta_k (I + 2A) r_{k-1}].$$

Calculons maintenant les coefficients η_k, δ_k et γ_k:
Comme $P_{k+1}(0) = P_k(0) = P_{k-1}(0) = 1$ nous avons par (iii)

$$\eta_k = \frac{1}{\gamma_k + \delta_k}.$$

En multipliant (iii) par ξ^{k-1} et sachant que $d(\xi^{k-1}P_{k+1}) = d(\xi^{k-1}P_k) = 0$ nous obtenons

$$\delta_k = \frac{d(\xi^k P_k(\xi))}{d(\xi^{k-1}P_{k-1}(\xi))} = \frac{(y,(I+A)^k P_k(A)r_0)}{(y,(I+A)^{k-1}P_{k-1}(A)r_0)}.$$

Donc

$$\delta_k = \frac{(y,r_k)}{(y,r_{k-1})}.$$

De même grâce à (iii) multipliée par ξ^k et sachant que $d(\xi^k P_{k+1}(\xi)) = 0$ nous écrivons

$$\gamma_k = \frac{[d(\xi^{k+1}P_k(\xi)) - \delta_k d(\xi^k P_{k-1}(\xi))]}{d(\xi^k P_k(\xi))},$$

$$\gamma_k = \frac{[(y,A(I+A)^k P_k(A)r_0) - \delta_k(y,A(I+A)^{k-1}P_{k-1}(A)r_0)]}{(y,(I+A)^k P_k(A)r_0)}.$$

D'où

$$\gamma_k = \frac{[(y,Ar_k) - \delta_k(y,Ar_{k-1})]}{(y,r_k)}.$$

Ces formules nous permettent d'écrire l'algorithme TEA2-Orthores comme suit:

- choisir x_0 et y
  poser $r_0 = b - Ax_0$, $r_{-1} = y$
- pour $k = 0, 1, \cdots$ calculer

$$\delta_k = \frac{(y,r_k)}{(y,r_{k-1})}$$

$$\gamma_k = \frac{[(y,Ar_k) - \delta_k(y,Ar_{k-1})]}{(y,r_k)}$$

$$\eta_k = \frac{1}{\gamma_k + \delta_k}$$

$$x_{k+1} = x_k + \eta_k[(A - \gamma_k I + I)r_k - \delta_k(I+2A)r_{k-1}]$$

$$r_{k+1} = -\eta_k(I+A)[(A - \gamma_k I)r_k - \delta_k(I+A)r_{k-1}]$$

- Si r_{k+1} est non nul alors continuer.

3.3.3 TEA2-Orthomin

Les formules de récurrence utilisée par la méthode Lanczos- Orthomin sont

$$(iv) \quad P_{k+1}(\xi) = P_k(\xi) - \beta_k \xi Q_k(\xi)$$

$$(v) \quad Q_k(\xi) = P_k(\xi) + \alpha_k Q_{k-1}(\xi)$$

où $Q_k(\xi) = (-1)^k \dfrac{\begin{vmatrix} d_0 & \cdots & d_{k-1} \\ \vdots & \cdots & \vdots \\ d_{k-1} & \cdots & d_{2k-2} \end{vmatrix}}{\begin{vmatrix} d_1 & \cdots & d_k \\ \vdots & \cdots & \vdots \\ d_k & \cdots & d_{2k-1} \end{vmatrix}} P_k^{(1)}(\xi).$

Nous posons

$$\varphi_k(\xi) = (1+\xi)^k P_k(\xi)$$

$$\psi_k(\xi) = (1+\xi)^k Q_k(\xi).$$

Les relations de récurrence précédentes nous donnent:

$$\varphi_{k+1}(\xi) = (1+\xi)[\varphi_k(\xi) - \beta_k \xi \psi_k(\xi)]$$

$$\psi_k(\xi) = \varphi_k(\xi) + \alpha_k(1+\xi)\psi_{k-1}(\xi).$$

Nous avons de plus

$$r_k = \varphi_k(A) r_0$$

et nous posons

$$t_k = \psi_k(A) r_0$$

ce qui nous donne

$$r_{k+1} = (I + A)(r_k - \beta_k A t_k)$$

et

$$t_k = r_k + \alpha_k (I + A) t_{k-1}.$$

Sachant que $r_k = b - Ax_k$, nous avons

$$x_{k+1} = x_k - r_k + \beta_k (I + A) t_k.$$

Calculons maintenant les coefficients α_k et β_k.
Nous utilisons tout d'abord (v) multipliée par ξ^k et les propriétés d'orthogonalité de $P_k^{(1)}$ et par conséquent de Q_k ; nous obtenons

$$\alpha_k = -\frac{d(\xi^k P_k(\xi))}{d(\xi^k Q_{k-1}(\xi))},$$

$$\alpha_k = -\frac{d((1+\xi)^k P_k(\xi))}{d(\xi(1+\xi)^{k-1} Q_{k-1}(\xi))},$$

$$\alpha_k = -\frac{(y,(I+A)^k P_k(A) r_0)}{(y, A(I+A)^{k-1} Q_{k-1}(A) r_0)}.$$

Donc

$$\alpha_k = -\frac{(y, r_k)}{(y, At_{k-1})}.$$

En utilisant (iv) multipliée par ξ^k nous avons aussi

$$\beta_k = \frac{d(\xi^k P_k(\xi))}{d(\xi^{k+1} Q_k(\xi))}, = \frac{d((1+\xi)^k P_k(\xi))}{d(\xi(1+\xi)^k Q_k(\xi))}$$

d'où

$$\beta_k = \frac{(y, r_k)}{(y, At_k)}.$$

En rassemblant toutes ces formules, nous obtenons finalement l'algorithme TEA2-Orthomin:

- choisir x_0 et y
  poser $r_0 = b - Ax_0$, $t_0 = y$
- pour $k = 0, 1, \cdots$ calculer

$$\beta_k = \frac{(y, r_k)}{(y, At_k)}$$

$$x_{k+1} = x_k - r_k + \beta_k (I+A) t_k$$

$$r_{k+1} = (I+A)(r_k - \beta_k A t_k)$$

- Si r_{k+1} est non nul calculer

$$\alpha_{k+1} = -\frac{(y, r_{k+1})}{(y, At_k)}$$

$$t_{k+1} = r_{k+1} + \alpha_{k+1}(I+A) t_k.$$

4 La seconde transformation de suites de vecteurs et le CGS

Le CGS [44] est défini par le résidu à l'itération k ,r_k, donné par

$$r_k = P_k(A)^2 r_0$$

Le calcul récursif des r_k est rendu possible par la mise au carré des relations utilisées pour le calcul des P_k dans la méthode du gradient biconjugué [25] ou dans d'autres algorithmes de type Lanczos.

Revenons aux transformations de suites de vecteurs évoquées dans le premier paragraphe. Nous considérons le cas particulier où $c_n = (y, \Delta u_n)$ avec des vecteurs u_n générés par

$$u_0 \text{ donné}, \quad u_{n+1} = Bu_n - b \text{ pour } n = 0, 1, \cdots$$

avec B une matrice carrée telle que $B = I + A$ soit régulière et b est un vecteur. Nous posons

$$Q_j^{(i,n)}(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^j \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \cdots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \cdots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}.$$

En utilisant la définition de $G_j^{(i)}(k, n)$, nous obtenons alors

$$G_j^{(i)}(k, n) - x = Q_j^{(i,n)}(B)(\varepsilon_{i,2k}^{(n)} - x).$$

De plus, grâce à la définition de $\varepsilon_{i,2k}^{(n)}$, nous avons

$$\varepsilon_{i,2k}^{(n)} - x = Q_k^{(0,n)}(B)(u_{n+i} - x)$$

ce qui nous donne

$$G_j^{(i)}(k, n) - x = Q_j^{(i,n)}(B) Q_k^{(0,n)}(B)(u_{n+i} - x) \qquad (1)$$

d'où en particulier

$$G_k^{(0)}(k,n) - x = [Q_k^{(0,n)}(B)]^2(u_n - x).$$

Si nous définissons le vecteur résidu $r(z)$ par $r(z) = b - Az = A(x - z)$, alors nous avons

$$\Delta u_i = B^i \Delta u_0 = B^i r(u_0).$$

Nous posons alors $c^{(j)}$ la fonctionnelle telle que

$$c^{(j)}(\xi^i) = c_{j+i} = (y, \Delta u_{i+j}) = (y, B^{i+j} r(u_0)).$$

Les $Q_j^{(i,n)}$ satisfont aux conditions

$$c^{(n+i)}(\xi^m Q_j^{(i,n)}) = 0 \text{ pour } m = 0, \cdots, j-1$$

et $Q_j^{(i,n)}(1) = 1$.
Posons maintenant

$$d^{(j)}(\xi^i) = d_{j+i} = (y, A^{i+j} r(u_0))$$

et

$$P_j^{(i,n)}(\xi) = Q_j^{(i,n)}(1 + \xi).$$

Nous avons alors

$$P_j^{(i,n)}(0) = Q_j^{(i,n)}(1) = 1$$

et par la formule binômiale avec $B^i = (I + A)^i$ nous obtenons

$$d^{(0)}((1+\xi)^m P_j^{(0,0)}(\xi))$$

$$= d^{(0)}((1+\xi)^m Q_j^{(0,0)}(1+\xi))$$

$$= c^{(0)}(\xi^m Q_j^{(0,0)}(\xi)) = 0 \text{ pour } m = 0, \cdots, j-1.$$

Comme $1, (1+\xi), \cdots, (1+\xi)^{j-1}$ est une base de l'espace vectoriel des polynômes de degré au plus $j-1$, nous en déduisons que $P_j^{(0,0)}$ est le polynôme de degré au plus j appartenant à la famille de polynômes orthogonaux par rapport à $d^{(0)}$, et nous avons $P_j^{(0,0)}(0) = 1$. Nous en déduisons que

$$Q_j^{(0,0)}(B) = Q_j^{(0,0)}(I + A) = P_j^{(0,0)}(A).$$

L'équation (1) nous donne alors en particulier

$$r(G_j^{(0)}(k,0)) = A(x - G_j^{(0)}(k,0)) = AQ_j^{(0,0)}(B)Q_k^{(0,0)}(B)(x - u_0).$$

Or la matrice $Q_j^{(0,0)}(B)$ étant une combinaison linéaire de B et donc de A (car $B = I + A$) nous avons

$$AQ_j^{(0,0)}(B) = Q_j^{(0,0)}(B)A.$$

Ainsi

$$r(G_j^{(0)}(k,0)) = Q_j^{(0,0)}(B)AQ_k^{(0,0)}(B)(x - u_0) = Q_j^{(0,0)}(B)Q_k^{(0,0)}(B)A(x - u_0),$$

d'où

$$r(G_j^{(0)}(k,0)) = P_j^{(0,0)}(A)P_k^{(0,0)}(A)r(u_0).$$

Pour $j = k$ nous avons alors

$$r(G_k^{(0)}(k,0)) = [P_k^{(0,0)}(A)]^2 r(u_0).$$

Cette dernière relation montre que $G_k^{(0)}(k,0)$ correspond au $k - ieme$ itéré x_k obtenu par le CGS.

Nous pouvons remarquer par ailleurs que, de la même manière, nous aurions pu obtenir

$$r(\varepsilon_{0,2k}^{(0)}) = P_k^{(0,0)}(A)r(u_0)$$

et

$$r(\varepsilon_{k,2k}^{(0)}) = (I + A)^k P_k^{(0,0)}(A)r(u_0)$$

ce qui montre, par une autre méthode que celle utilisée dans le troisième paragraphe, les liens respectifs entre $\varepsilon_{0,2k}^{(0)}$ ou $\varepsilon_{k,2k}^{(0)}$ et le k^{ieme} itéré d'une méthode de Lanczos.

5 Extension aux systèmes non linéaires

Dans ce paragraphe nous allons montrer comment les méthodes précédemment évoquées fournissent, en étant appliquées à des systèmes d'équations non linéaires, des méthodes de résolution à convergence quadratique locale.

Ce type de méthodes a été trouvé indépendemment et presque simultanément par C.Brezinski [3] et par E.Gekeler [26].

5.1 Algorithmes

Nous considérons le système de point fixe $G(x) = x$, où $G : D \subset \mathbb{C}^p \to \mathbb{C}^p$ avec D un ouvert convexe de $\mathbb{C}^p$. Nous notons x^* une solution de ce système et J la matrice jacobienne $G'(x^*)$. u_0 étant choisi nous supposons que la suite $(u_n)_{n\in\mathbb{N}}$ est générée par

$$u_{n+1} = G(u_n)$$

et nous avons toujours $c_n = (y, \Delta u_n)$ avec y un vecteur arbitraire non nul. Pour résoudre le système $G(x) = x$ de p équations à p inconnues nous considérons les algorithmes iNTEA, (N pour non linéaire, TEA pour rappeler l'origine de ce nouvel algorithme, et i en tant que coefficient choix de la méthode utilisée), et les algorithmes iNCGS, (N pour non linéaire, CGS ou Conjugate Gradient Squared pour marquer l'idée première utilisée dans ces méthodes, et i comme indice de choix de méthode). Ces algorithmes sont les suivants:

- choisir un point initial x_0 dans D
- à l'itération k
  - poser $u_0 = x_k$
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$ où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$
  - calculer $x_{k+1} = \begin{cases} \varepsilon_{i,2d_k}^{(0)} & \text{pour le iNTEA } i = 0, \ldots, d_k \\ G_i^{(0)}(d_k, 0) & \text{pour le iNCGS } i = 1, \ldots, d_k \end{cases}$.

Dans ces algorithmes nous pouvons remarquer que pour $i = 0$ le iNTEA calcule à l'itération k le vecteur $\varepsilon_{0,2d_k}^{(0)}$ qui dans le cas linéaire est calculé par le TEA1 ; "0NTEA" pourrait donc encore s'appeler NTEA1 (algorithme non linéaire issu du premier epsilon algorithme topologique). De même pour $i = d_k$, le iNTEA représente en fait le NTEA2 (algorithme non linéaire issu du second epsilon algorithme topologique). Enfin le NCGS (nonlinear conjugate gradient squared) est un cas particulier du iNCGS pour $i = d_k$. NTEA1 et NTEA2 auraient encore pu s'appeler méthodes de Lanczos non linéaires vus les liens précédemment évoqués entre les méthodes de Lanczos et les deux epsilon algorithmes topologiques .

Nous pouvons encore noter que les $G_i^{(0)}(d_k, 0)$ sont calculés pour $i = 1, \ldots, d_k$ et non pour $i = 0, \ldots, d_k$ car pour $i = 0$, $G_i^{(0)}(d_k, 0) = \varepsilon_{0,2d_k}^{(0)}$.

L'intérêt de ces nouvelles méthodes réside notamment dans le fait qu'elles ne nécessitent aucun calcul de dérivées et aucune inversion de matrices, contrairement à d'autres méthodes non linéaires; en effet nous pouvons noter que Nonlinear Orthomin(1), Newton-Orthomin [22], ainsi que Nonlinear Krylov subspace projection methods [19] sont également des méthodes généralisant des méthodes linéaires, mais que ces dernières nécessitent en théorie le calcul de la matrice jacobienne du système qui en pratique est approximée.

5.2 Théorèmes de convergence quadratique

Nous supposons dans cette partie que G satisfait l'hypothèse **H** suivante

$$\mathbf{H}\begin{cases} \text{la matrice } I - J \text{ est régulière.} \\ \text{la dérivée de Fréchet } G' \text{ de } G \text{ satisfait la condition de Lipschitz} \\ \quad \exists L > 0, \forall x, y \in D, \|G'(x) - G'(y)\| \leq L\|x - y\|. \end{cases}$$

Nous notons H_k la matrice suivante

$$H_k = \begin{pmatrix} 1 & \cdots & 1 \\ (y, \frac{\Delta u_0}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_{d_k}}{\|\Delta u_0\|}) \\ \vdots & \ddots & \vdots \\ (y, \frac{\Delta u_{d_k-1}}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_{2d_k-1}}{\|\Delta u_0\|}) \end{pmatrix}.$$

La convergence quadratique du NTEA1 a été prouvée dans [35] en s'appuyant sur les résultats émis dans [31]. Il faut également se référer à [46] où la démonstration d'un théorème de convergence de méthodes non linéaires similaires est donnée.

Théorème 1 *Si G satisfait l'hypothèse* **H** *et s'il existe α strictement positif et K tels que $\forall k \geq K, \mid detH_k \mid \geq \alpha$ Alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|\varepsilon_{0,2d_k}^{(0)} - x^*\| = O(\|x_k - x^*\|^2).$$

Démontrons maintenant un résultat similaire pour le iNTEA.

Théorème 2 *Si G satisfait l'hypothèse* **H** *et s'il existe α strictement positif et K tels que $\forall k \geq K, \mid detH_k \mid \geq \alpha$ Alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|\varepsilon_{i,2d_k}^{(0)} - x^*\| = O(\|x_k - x^*\|^2) \quad \textit{pour } i = 1, \ldots, d_k.$$

Ce théorème prouve en particulier la convergence quadratique du NTEA2 pour $i = d_k$.

Démonstration

Posons

$$\varepsilon_{i,2d_k}^{(0)} = \frac{\begin{vmatrix} u_i & \cdots & u_{i+d_k} \\ c_0 & \cdots & c_{d_k} \\ \vdots & \ddots & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_0 & \cdots & c_{d_k} \\ \vdots & \ddots & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}$$

où $c_i = (y, \Delta u_i)$.

En développant le déterminant du numérateur par rapport à sa première ligne, nous obtenons:

$$\varepsilon_{i,2d_k}^{(0)} = \sum_{j=0}^{d_k} b_j^{(k)} u_{i+j}$$

et nous savons que les $b_j^{(k)}$ sont des coefficients barycentriques, c'est à dire que

$$\sum_{j=0}^{d_k} b_j^{(k)} = 1.$$

Prouvons tout d'abord que les $b_j^{(k)}$ sont bornés indépendemment de k. Les $b_j^{(k)}$ sont solutions du système suivant :

$$\begin{cases} b_0^{(k)} + \cdots + b_{d_k}^{(k)} = 1 \\ b_0^{(k)}(y, \frac{\Delta u_0}{\|\Delta u_0\|}) + \cdots + b_{d_k}^{(k)}(y, \frac{\Delta u_{d_k}}{\|\Delta u_0\|}) = 0 \\ \vdots \\ b_0^{(k)}(y, \frac{\Delta u_{d_k-1}}{\|\Delta u_0\|}) + \cdots + b_{d_k}^{(k)}(y, \frac{\Delta u_{2d_k-1}}{\|\Delta u_0\|}) = 0. \end{cases}$$

Comme H_k est la matrice du système et $b^{(k)} = \begin{pmatrix} b_0^{(k)} \\ \vdots \\ b_{d_k}^{(k)} \end{pmatrix}$.

Le système précédent s'écrit alors

$$H_k b^{(k)} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

De plus, par hypothèse,

$$\exists \alpha > 0, \exists K \in I\!N, \forall k \geq K, \mid det H_k \mid \geq \alpha.$$

Grâce à un lemme de Kato [33], nous obtenons alors:

$$\| H_k^{-1} \| \leq \frac{\| H_k \|^{d_k}}{\mid det H_k \mid} \leq \frac{\| H_k \|^{d_k}}{\alpha}.$$

De plus, si nous notons (h_{ij}) les éléments de la matrice H_k nous avons : $\exists \beta, \| H_k \| \leq \beta$ parce que $\| H_k \| = \max\limits_{i=1}^{d_k+1} \sum\limits_{j=1}^{d_k+1} \mid h_{ij} \mid$ est bornée ; en effet nous avons

- $\sum_{j=1}^{d_k+1} \mid h_{1j} \mid = d_k$ pour la première ligne de la matrice
- pour $i \geq 2, \mid h_{ij} \mid \leq \| y \| \frac{\| \Delta u_{i+j-2} \|}{\| \Delta u_0 \|}$ par l'inégalité de Cauchy-Schwartz et comme expliqué dans [38] pp. 70-73 nous avons

$$\frac{\Delta u_{i+j-2}}{\| \Delta u_0 \|} = J^{i+j-2} \frac{\Delta u_0}{\| \Delta u_0 \|} + O(\| x_k - x^* \|^2).$$

Aussi

$$\frac{\| \Delta u_{i+j-2} \|}{\| \Delta u_0 \|} \leq \| J \|^{i+j-2} + AM^2$$

où A est une constante et $\| x_k - x^* \| \leq M$.

Ainsi pour tout i , $\sum\limits_{j=1}^{d_k+1} \mid h_{ij} \mid$ est bornée, donc $\exists \beta, \| H_k \| \leq \beta$.

Par conséquent,

$$\| b^k \| \leq \beta^{d_k} / \alpha.$$

Prouvons maintenant que nous avons la relation

$$\varepsilon^{(0)}_{i,2d_k} - x^* = J^i(\varepsilon^{(0)}_{0,2d_k} - x^*) + O(\|x_k - x^*\|^2) \quad \text{pour } i = 1, \ldots, d_k.$$

- Prouvons le tout d'abord pour $i = 1$:

$$\varepsilon^{(0)}_{0,2d_k} = \sum_{j=0}^{d_k} b_j^{(k)} u_j.$$

Comme $\sum_{j=0}^{d_k} b_j^{(k)} = 1$,nous avons

$$\varepsilon^{(0)}_{0,2d_k} - x^* = \sum_{j=0}^{d_k} b_j^{(k)}(u_j - x^*).$$

De la même manière ,nous avons

$$\varepsilon^{(0)}_{1,2d_k} - x^* = \sum_{j=0}^{d_k} b_j^{(k)}(u_{j+1} - x^*).$$

En outre

$$u_{j+1} = G(u_j) = x^* + J(u_j - x^*) + O(\|u_j - x^*\|^2).$$

Donc,

$$\varepsilon^0_{1,2d_k} - x^* = \sum_{j=0}^{d_k} b_j^k J(u_j - x^*) + \sum_{j=0}^{d_k} b_j^k O(\|u_j - x^*\|^2).$$

Comme les b_j^k sont bornés et que

$$u_j - x^* = J^j(x_k - x^*) + O(\|x_k - x^*\|^2).$$

Nous obtenons

$$\sum_{j=0}^{d_k} b_j^k O(\|u_j - x^*\|^2) = O(\|x_k - x^*\|^2).$$

Par conséquent,

$$\varepsilon^0_{1,2d_k} - x^* = J(\varepsilon_{0,2d_k} - x^*) + O(\|x_k - x^*\|^2).$$

- De même si nous supposons que

$$\varepsilon^0_{p-1,2d_k} - x^* = J^{p-1}(\varepsilon_{0,2d_k} - x^*) + O(\|x_k - x^*\|^2)$$

comme nous avons de plus

$$\varepsilon^0_{p,2d_k} - x^* = J(\varepsilon_{p-1,2d_k} - x^*) + O(\|x_k - x^*\|^2),$$

alors

$$\varepsilon^0_{p,2d_k} - x^* = J^p(\varepsilon_{0,2d_k} - x^*) + O(\|x_k - x^*\|^2).$$

Nous avons donc prouvé par récurrence la relation souhaitée.

Comme nous sommes de plus dans les conditions du théorème 1, nous avons

$$\|\varepsilon^{(0)}_{0,2d_k} - x^*\| = O(\|x_k - x^*\|^2),$$

nous pouvons en conclure que

$$\|\varepsilon^{(0)}_{i,2d_k} - x^*\| = O(\|x_k - x^*\|^2) \text{ pour } i = 1, \ldots, d_k,$$

ce qui prouve la convergence quadratique locale de iNTEA pour $i = 1, \ldots, d_k$.

■

Nous notons maintenant

$$H_{k,i} = \begin{pmatrix} 1 & \cdots & 1 \\ (y, \frac{\Delta u_0}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_i}{\|\Delta u_0\|}) \\ \vdots & \ddots & \vdots \\ (y, \frac{\Delta u_{i-1}}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_{2i-1}}{\|\Delta u_0\|}) \end{pmatrix}.$$

Nous avons en particulier $H_{k,d_k} = H_k$.

Pour le iNCGS, nous avons le

Théorème 3 *i étant choisi entre 1 et d_k,*
si G satisfait l'hypothèse **H** *et s'il existe α et β strictement positifs et K tels que $\forall k \geq K \mid detH_{k,i} \mid \geq \alpha$ et $\mid detH_k \mid \geq \beta$, alors il existe un voisinage U de x^* tel que $\forall x_0 \in U$*

$$\|G_i^{(0)}(d_k, 0) - x^*\| = O(\|x_k - x^*\|^2).$$

Nous pouvons remarquer que pour $i = d_k$, ce théorème prouve la convergence quadratique locale du NCGS.

Démonstration

i étant un entier entre 1 et d_k, nous avons

$$G_i^{(0)}(d_k, 0) = \sum_{j=0}^{i} b_j^{k,i} \varepsilon_{j,2d_k}^{(0)}$$

où $\sum_{j=0}^{i} b_j^{k,i} = 1$, en conséquence

$$G_i^{(0)}(d_k, 0) - x^* = \sum_{j=0}^{i} b_j^{k,i} (\varepsilon_{j,2d_k}^{(0)} - x^*).$$

Les b_j^{k,d_k} sont les mêmes coefficients b_j^k que dans le second théorème puisque les déterminants définissant $G_{d_k}^{(0)}(d_k, 0)$ et $\varepsilon_{i,2d_k}^{(0)}$ sont similaires. Aussi ils sont bornés indépendemment de k.

Les autres coefficients $b_j^{k,i}$ sont solutions du système

$$H_{k,i} \begin{pmatrix} b_0^{k,i} \\ \vdots \\ \vdots \\ b_i^{k,i} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

De plus par un lemme de Kato [33] nous avons

$$\|H_{k,i}^{-1}\| \leq \frac{\|H_{k,i}\|^i}{\mid detH_{k,i} \mid} \leq \frac{\|H_{k,i}\|^i}{\alpha},$$

où il existe α strictement positif et K tels que $\forall k \geq K \mid detH_{k,i} \mid \geq \alpha$.
En outre $\|H_{k,i}\| = max_{jj=1}^{i+1} \sum_{j=1}^{i+1} \mid h_{ij} \mid$
et nous avons prouvé dans le théorème 2 de ce chapitre que

$$\forall i = 1, \ldots, d_k + 1, \ \forall j = 1, \ldots, d_k + 1, \ \mid h_{ij} \mid \text{ est fini.}$$

Donc cela reste vrai pour les éléments de la matrice tronquée $H_{k,i}$. Nous obtenons donc $\|H_{k,i}^{-1}\| \leq \frac{\beta^i}{\alpha}$, ce qui nous permet de conclure que les coefficients $b_i^{k,i}$ sont bornés indépendemment de k.
De plus, comme nous le disent les théorèmes précédents, nous avons

$$\|\varepsilon_{j,2d_k}^{(0)} - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour } j = 0, \ldots, d_k$$

si bien que

$$\|G_i^0(d_k, 0) - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour } i = 1, \ldots, d_k$$

ce qui prouve la convergence quadratique locale des méthodes iNCGS, dont le NCGS. ■

5.3 Méthode de tri

La méthode NCGS utilisée précédemment effectue en fait, en dimension d_k, une moyenne $G^{(0)}_{d_k}(d_k, 0)$ des vecteurs $\varepsilon^{(0)}_{0,2d_k} \dots \varepsilon^{(0)}_{d_k,2d_k}$.
Nous pouvons ainsi calculer différentes moyennes à chaque itération et ne garder que la meilleure d'entre elles, c'est à dire celle qui correspond à un vecteur solution donnant la plus petite norme infinie de l'erreur, espérant ainsi améliorer le résultat. Nous pourrions donc appliquer n'importe quelle permutation sur les vecteurs définissant la première ligne du numérateur de $G^{(0)}_{d_k}(d_k, 0)$; en effet nous avons

$$G^{(0)}_{d_k}(d_k, 0) = \frac{\begin{vmatrix} \varepsilon^{(0)}_{0,2d_k} & \cdots & \varepsilon^{(0)}_{d_k,2d_k} \\ c_0 & \cdots & c_{d_k} \\ \vdots & \ddots & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_0 & \cdots & c_{d_k} \\ \vdots & \ddots & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}.$$

Alors nous pouvons permuter les vecteurs $\varepsilon^{(0)}_{0,2d_k} \dots \varepsilon^{(0)}_{d_k,2d_k}$ ce qui change les poids attribués à chacun d'eux. Nous pouvons remarquer que ceci reste applicable à toute méthode iNCGS. Pour des raisons de coût de calcul évidentes nous restreignons l'ensemble des permutations aux seules permutations que nous appelerons 'endroit' et 'envers' décrites ci-après.
Les méthodes 'endroit' vont calculer les vecteurs moyenne avec premières lignes suivantes:

- $\varepsilon^{(0)}_{0,2d_k}, \dots, \varepsilon^{(0)}_{i,2d_k}$, qui correspond au vecteur obtenu par la méthode iNCGS.
- $\varepsilon^{(0)}_{1,2d_k}, \varepsilon^{(0)}_{2,2d_k}, \dots, \varepsilon^{(0)}_{i,2d_k}, \varepsilon^{(0)}_{0,2d_k}$, avec un premier décalage dans les indices.
- $\varepsilon^{(0)}_{2,2d_k}, \varepsilon^{(0)}_{3,2d_k}, \dots, \varepsilon^{(0)}_{i,2d_k}, \varepsilon^{(0)}_{0,2d_k}, \varepsilon^{(0)}_{1,2d_k}$.
- $\dots$
- $\varepsilon^{(0)}_{i,2d_k}, \varepsilon^{(0)}_{0,2d_k}, \dots, \varepsilon^{(0)}_{i-1,2d_k}$.

Nous obtenons ainsi $i+1$ moyennes différentes.
Les 'méthodes 'envers' vont calculer les vecteurs moyenne avec premières lignes suivantes:

- $\varepsilon^{(0)}_{i,2d_k}, \varepsilon^{(0)}_{i-1,2d_k}, \ldots, \varepsilon^{(0)}_{0,2d_k}$.
- $\varepsilon^{(0)}_{i-1,2d_k}, \varepsilon^{(0)}_{i-2,2d_k}, \ldots, \varepsilon^{(0)}_{0,2d_k}, \varepsilon^{(0)}_{i,2d_k}$.
- $\ldots$
- $\varepsilon^{(0)}_{0,2d_k}, \varepsilon^{(0)}_{i,2d_k}, \ldots, \varepsilon^{(0)}_{1,2d_k}$.

Nous obtenons là aussi $i+1$ moyennes différentes.
A chaque itération nous modifions donc les algorithmes iNCGS en choisissant parmis les $2i+2$ vecteurs à notre disposition.

5.4 Exemples numériques

La programmation des différents algorithmes de ce chapitre et des suivants est faite en Fortran. Les calculs sont effectués en double précision, soit avec 16 chiffres significatifs pour représenter la mantisse d'un réel.

En pratique, le degré d_k du polynôme minimal de J pour le vecteur $x_k - x^*$ n'est pas connu, aussi dans les exemples ci-dessous nous prendrons $d_k = p$. la dimension du système. En réalité il faut vérifier pour cela que les vecteurs colonnes $\Delta u_0, \Delta u_1, \ldots, \Delta u_{p-1}$ sont linéairement indépendents sinon nous prenons $d_k = l_k$ où l_k est défini par $l_k = max_{i \in \{1,\ldots,p\}}\{\text{Rank}(\Delta U_{0,i}) = i\}$, avec $\Delta U_{0,i}$ la matrice $p \times i$ dont les colonnes sont $\Delta u_0, \ldots, \Delta u_{i-1}$. Nous pouvons remarquer que dans certains cas, une solution du sytème est atteinte lors du calcul des vecteurs $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$; alors nous avons $u_j = u_{j-1}$ à partir d'un certain rang j et il y a division par zéro; nous stoppons donc l'algorithme dès que $u_j = u_{j-1}$.

Il a été noté dans les exemples que le choix du vecteur arbitraire y influe sur les résultats. Pour certains exemples un bon choix semble être $y = \Delta u_0$ mais ce choix n'est pas toujours le meilleur. Pour l'exemple 7 notamment un bon choix du vecteur y semble être un vecteur propre de la matrice utilisée pour la construction de l'exemple ; ce choix n'est cependant généralement pas possible en pratique, les vecteurs propres de la matrice n'étant pas connus. Dans chaque exemple nous affichons la méthode choisie et, à chaque

itération, si la solution du système est connue, nous affichons la norme infinie de l'erreur, sinon nous affichons la norme infinie du vecteur différence entre le vecteur obtenu à l'itération et son image par la fonction G.

Pour chacune des méthodes iNTEA ou iNCGS employées ci-après le nombre d'évaluations de fonction non linéaire est de $2p$ soit deux fois la dimension du système par itération.

Nous appliquons la méthode de tri uniquement à l'algorithme NCGS pour chaque exemple; cette programmation nous montre que de la méthode de tri résulte un algorithme assez instable; il est cependant intéressant de voir que le vecteur résultat est mélioré dans certains exemples par rapport au vecteur obtenu par le NCGS.

Nous rappelons que les méthodes employées ne nécessitent aucune inversion de matrice ni aucun calcul de dérivée.

5.4.1 Exemple 1

Cet exemple est donné dans [3].

Soit à trouver la solution unique $\begin{pmatrix} -1 \\ +1 \end{pmatrix}$ du système de dimension 2 , $G(x) = x$, où G est définie par

$$\begin{aligned} G_1(x) &= -\frac{x_{(2)}^4}{4} - \frac{3}{4} \\ G_2(x) &= -0.405e^{1-x_{(1)}^2} + 1.405 \end{aligned}$$

avec $x_{(1)}$ et $x_{(2)}$ les coordonnées du vecteur x, et $G(x) = \begin{pmatrix} G_1(x) \\ G_2(x) \end{pmatrix}$.

Nous prenons de plus $u_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ et $y = \begin{pmatrix} -0.75 \\ 0.405e + 1.405 \end{pmatrix}$.

Dans cet exemple le choix ci-dessus du vecteur arbitraire y intervenant dans le produit scalaire $c_n = (y, \Delta u_n)$ est meilleur que le choix Δu_0. C'est ce choix que nous utilisons ci-après.

it.	NTEA1	iNTEA avec $i = 1$
1	$1.444439489766888E-001$	$1.544532535435110E-001$
2	$3.378795021218095E-002$	$3.353356481069969E-002$
3	$2.943343112489782E-003$	$2.640263548193578E-003$
4	$2.193007931394764E-005$	$1.781668887446131E-005$
5	$1.137686500563007E-009$	$8.169211973552137E-010$
6	$6.661338147750939E-016$	$1.110223024625157E-015$
7	$2.220446049250313E-016$	$8.881784197001252E-016$

it.	NTEA2	iNCGS avec i=1
1	$1.404096054971112E-001$	$5.963157472852210E-001$
2	$3.268667683588089E-002$	$3.182020194776491E-002$
3	$2.699459109712810E-003$	$3.428836062641660E-003$
4	$1.939107749937552E-005$	$2.903446757838394E-005$
5	$9.389130584125382E-010$	$1.971195318084540E-009$
6	$6.661338147750939E-016$	$1.110223024625157E-016$
7	$4.440892098500626E-016$	
8	$2.220446049250313E-016$	

it.	NCGS	tri appl. au NCGS
1	$1.495386699122028E-001$	$1.359617211443339E-001$
2	$3.138346988966034E-002$	$2.930407151655878E-002$
3	$2.160961459930721E-003$	$2.432659679717619E-003$
4	$1.586914675910656E-005$	$1.997222312577485E-005$
5	$8.695155706561764E-010$	$1.377219449594236E-009$
6	$8.548717289613705E-015$	$7.771561172376096E-016$

Dans cet exemple toutes les méthodes se valent.
Nous pouvons remarquer l'obtention d'un chiffre exact supplémentaire du vecteur résultat après avoir appliqué la méthode de tri à l'algorithme NCGS.

5.4.2 Exemple 2

Cet exemple se trouve dans [3].
Soit à trouver le point fixe $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ du système de dimension 2, $G(x) = x$ suivant

$$G_1(x) = \frac{x_{(2)}^2}{2} + x_{(1)} - \frac{1}{2}$$
$$G_2(x) = sin(x_{(1)}) + sin(x_{(2)} - 1) + 1.$$

Nous prenons comme vecteur initial $x_0 = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$
et comme vecteur arbitraire $y = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, ce choix étant ici meilleur que le choix Δu_0.
Nous obtenons les tableaux de résultats suivants

it.	NTEA1	iNTEA avec $i = 1$
1	$8.159870861905361E-001$	$8.095185740496529E-002$
2	$3.766080148207191E-001$	$2.968937774020075E-003$
3	$5.395845994824269E-002$	$1.945853822105404E-005$
4	$6.056098275672994E-003$	$7.615098551738697E-010$
5	$6.826380123524580E-005$	$6.658432100691460E-017$
6	$1.603951194084630E-009$	
7	$2.092510192897024-017$	

it.	NTEA2
1	$2.542105176295828E-002$
2	$2.279230092803619E-003$
3	$1.380471170275843E-005$
4	$3.730676878532790E-010$
5	$5.466960916424836E-017$

it.	iNCGS avec $i = 1$	NCGS
1	$2.247686377260849E-002$	$1.132746613490681E-001$
2	$9.8030486550987713E-002$	$1.599401722825006E-002$
3	$1.068159253889300E-002$	$6.546554418965642E-004$
4	$6.493336424190126E-004$	$7.074705396663221E-008$
5	$9.666702376434106E-006$	$3.7583837648948251E-012$
6	$4.017803087167437E-008$	
7	$9.730607251119167E-015$	

Dans ce deuxième exemple les méthodes 1NTEA et NTEA2 sont nettement meilleures que NTEA1, puisque nous obtenons une norme de l'erreur en 10^{-17} dès la cinquième itération pour les deux premières méthodes tandis qu'il faut attendre la septième itération dans le cas du NTEA1. De même le NCGS, qui effectue en fait une moyenne des méthodes iNTEA pour i allant de 0 à p, est meilleur que la méthode 1NCGS qui elle n'effectue qu'une moyenne entre le NTEA1 et le iNTEA pour $i = 1$.
Dans cet exemple la méthode de tri ne rend pas meilleur l'algorithme NCGS.

5.4.3 Exemple 3

Cet exemple est donné dans [41].
Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 2.25 & 0.01 & 0.05 & 0.50 \\ 0.01 & 1.75 & 0.00 & 0.05 \\ 0.05 & 0.00 & 1.75 & 0.01 \\ 0.50 & 0.05 & 0.01 & 2.25 \end{pmatrix}$$

$$b = (-0.81, -0.31, -0.31, -0.81)^T$$

et

$$Q(x) = -0.5(x_{(1)}^2 + x_{(1)}x_{(4)}, x_{(2)}^2, x_{(3)}^2, x_{(1)}x_{(4)} + x_{(4)}^2)^T.$$

La solution de ce système est $(1, 1, 1, 1)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (1.2, 1.2, 1.2, 1.2)^T$.
Pour le vecteur arbitraire, le vecteur $y = (-1, 1, 2, 1)^T$ donne de meilleurs

résultats que le vecteur $\Delta u_0 = (-0.409, -0.285, -0.287, -0.683)^T$.
Ces résultats sont les suivants:

it.	NTEA1	iNTEA avec $i = 2$
1	$1.990020638415947E-003$	$9.827890712010845E-004$
2	$1.137487815050520E-008$	$8.209710466999809E-009$
3	$1.043609643147647E-014$	$1.546210004228968E-014$

it.	NTEA2	iNCGS avec i=2
1	$6.646839494364176E-004$	$1.206666054631367E-002$
2	$1.623915668602649E-009$	$6.766923225098864E-006$
3	$1.465494392505207E-014$	$2.193936143868314E-010$
4		$2.264854970235319E-014$

it.	NCGS	tri
1	$9.827890712010845E-004$	$7.764146731639165E-004$
2	$3.5844771595350279E-010$	$3.584477159535027E-010$
3	$6.306066779870889E-014$	$2.442490654175344E-015$
4	0.00000000000000000	

5.4.4 Exemple 4

Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 3.9 & -3.7 & 2.4 & -0.6 \\ 2.4 & -2.0 & 2.2 & -0.6 \\ 2.4 & -3.6 & 4.1 & -0.9 \\ 2.8 & -5.2 & 4.8 & -0.4 \end{pmatrix}$$

$$b = (-0.75, -0.75, -0.75, -0.75)^T$$

et

$$Q(x) = -0.25(x_{(1)}^2, x_{(2)}^2, x_{(3)}^2, x_{(4)}^2)^T.$$

La solution de ce système est $(3, 3, 3, 3)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (2.5, 2.5, 2.5, 2.5)^T$.

Pour le vecteur arbitraire, le vecteur $y = (-1, 1, 2, 1)^T$ donne de meilleurs résultats que le vecteur Δu_0.
Ces résultats sont les suivants:

it.	NTEA1	NTEA2
1	$1.259919054419356E-006$	$1.259919049978464E-006$
2	$1.731947918415244E-014$	$1.287858708565182E-014$
3	$1.332267629550188E-014$	

it.	iNCGS avec $i = 1$	NCGS
1	$1.259919082841066E-006$	$1.259919066409765E-006$
2	$1.7371947918415244-014$	$1.9539925233460276E-014$

5.4.5 Exemple 5

Nous voulons résoudre le système de dimension 6, $G(x) = x$ avec G définie par

$$\begin{aligned}
G_1(x) &= -0.75 - \frac{x_{(2)}^2 x_{(4)} x_{(6)}}{4} \\
G_2(x) &= -0.405e^{1+x_{(1)}x_{(2)}} + 1.405 \\
G_3(x) &= \frac{x_{(4)}x_{(6)}}{2} - 1.5 \\
G_4(x) &= 0.605e^{1-x_{(3)}^2} + 0.395 \\
G_5(x) &= \frac{x_{(2)}x_{(6)}}{2} - 1.5 \\
G_6(x) &= x_{(1)}x_{(5)}.
\end{aligned}$$

La solution de ce système est $(-1, 1, -1, 1, -1, 1)^T$. Nous prenons comme vecteur initial le vecteur $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T$. Pour le vecteur arbitraire y, le vecteur $(-1, 1, 2, 3, 1, 2)^T$ donne de bons résultats.

it.	NTEA1	iNTEA avec $i = 4$
1	$1.561846639127710E-001$	$2.558164062279267E-002$
2	$1.594392796396393E-002$	$6.566362154303285E-005$
3	$1.300385281381988E-004$	$1.408989591666909E-009$
4	$1.284059480965993E-008$	$1.643130076445232E-014$
5	$7.882583474838611E-015$	$3.108624468950438E-014$

it.	iNCGS avec $i = 2$	NCGS
1	$6.566529980303237E-002$	$2.700400154056015E-002$
2	$1.660916378010000E-003$	$1.734739479852654E-002$
3	$2.823890996594969E-007$	$1.183753610951577E-004$
4	$5.451195050909519E-014$	$1.569808505408332E-009$
5	$5.906386491005833E-014$	$1.969535645685028E-013$
6	$9.658940314238862E-015$	$6.239453398393380E-014$
7	$9.992007221626409E-016$	

it.	NTEA2
1	$2.010798727599328E-002$
2	$2.135145237347480E-005$
3	$7.360978493409220E-011$
4	$1.021405182655144E-014$
5	$1.443289932012704E-015$

5.4.6 Exemple 6

Cet exemple est donné dans [20].
Nous considérons maintenant le système non linéaire de dimension 10, $G(x) = x$ avec G définie comme suit

$$G(x) = x - hF(x)$$

où
$$\begin{aligned} F_1(x) &= (3 - 5x_{(1)})x_{(1)} + 1 - 2x_{(2)} \\ F_i(x) &= (3 - 5x_{(i)})x_{(i)} + 1 - x_{(i-1)} - 2x_{(i+1)} \text{ pour } i = 2, \ldots, 9 \\ F_{10}(x) &= (3 - 5x_{(10)})x_{(10)} + 1 - x_{(9)}. \end{aligned}$$

Nous prenons comme vecteur initial le vecteur

$$x_0 = (-1, -1, -1. -1, -1, -1, -1, -1, -1, -1)^T.$$

Pour le vecteur arbitraire y, le vecteur Δu_0 est un bon choix

$$\Delta u_0 = (1.25, 1, 1, 1, 1, 1, 1, 1, 1, 1.5)^T.$$

Dans cet exemple la solution n'est pas connue, aussi à chaque itération, nous afficherons la différence $(G(x_k) - x_k)$.

Dans le cas du choix $h = 0.2$, toutes les méthodes convergent comme nous pouvons le voir dans les tableaux de résultats:

it.	NTEA1	iNTEA avec $i = 3$
1	$2.630237080388331E-002$	$5.286970398248214E-003$
2	$5.972108676496890E-004$	$3.179747875942107E-005$
3	$9.067648150007379E-006$	$8.987911415125893E-009$
4	$1.479749878185999E-009$	$1.609823385706477E-014$
5	$5.190292640122607E-014$	$3.108624468950438E-014$
6	$6.639133687258436E-014$	
7	$7.771561172376096E-016$	

it.	NTEA2
1	$4.449150952551217E-003$
2	$2.630818999321827E-005$
3	$4.508985629314566E-009$
4	$1.043609643147647E-014$

it.	iNCGS avec $i = 4$	NCGS
1	$2.216961396448280E-003$	$2.054485535969652E-003$
2	$7.939903763909406E-007$	$1.466167983465994E-006$
3	$1.931788062847772E-014$	$1.887379141862766E-015$
4	$8.326672684688674E-016$	

Nous notons que dans cet exemple le NTEA2 et le iNTEA pour $i = 3$ convergent nettement plus rapidement que le NTEA1.

Nous pouvons remarquer que le choix du scalaire h est prépondérant; ainsi pour $h = 0.25$, les méthodes NTEA1 et iNCGS pour $i = 1$ convergent tandis que iNTEA pour $i = 3$, NTEA2, et NCGS divergent.

Une étude du choix de ce paramètre est donnée dans [11].

Remarquons encore que la dimension de ce système peut s'élever à 30 en donnant des résultats similaires sur les méthodes. Au dessus de cette dimension la place mémoire demandée est trop importante ce qui nécessite une optimisation supplémentaire de la programmation à ce niveau (en cours).

5.4.7 Exemple 7

Nous construisons cet exemple comme le fait M. Altman dans [1]. Soit A une matrice de dimension p vérifiant $Ax = \lambda x$, où x est un vecteur non nul de $\mathbb{R}^p$ et λ un scalaire. Nous pouvons écrire

$$\frac{(x, Ax)}{(x, x)} = \lambda.$$

Nous allons alors résoudre le système $G(x) = x$ où

$$G(x) = Ax - \frac{(x, Ax)}{(x, x)}x + x.$$

Soient en particulier

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 2 & 0 & 1 \\ 2 & -1 & 3 & -1 \\ 3 & 2 & 2 & 2 \\ 4 & -1 & 1 & -2 \end{pmatrix}.$$

Nous calculons

$$A = PDP^{-1} = \begin{pmatrix} 3/8 & -5/8 & 9/8 & -3/8 \\ 97/32 & 153/32 & -69/32 & -33/32 \\ -41/16 & -1/16 & 61/16 & -23/16 \\ 159/32 & 103/32 & -123/32 & 33/32 \end{pmatrix},$$

et nous résolvons alors le système $Ax - \dfrac{(x, Ax)}{(x, x)}x + x = x$, qui doit, étant donnée la construction de G, nous donner en solution un vecteur propre de la matrice A. Dans cet exemple nous notons λ_i les valeurs propres de A ($\lambda_1 = 1$, $\lambda_2 = 2, \ldots$), E_i les espaces propres associés à ces valeurs propres ($E_1 = \text{Vect}((1, 2, 3, 4)^T)$, $E_2 = \text{Vect}((2, -1, 2, -1)^T), \ldots$), et u_i les vecteurs colonnes de la matrice P.

Dans le tableau suivant nous prenons $y = u_1 = (1, 2, 3, 4)^T$ et $x_0 = (-0.5, 2.5, 1.5, 0.5)^T$ proche de u_3.

it.	NTEA1	NCGS
1	$7.280777913031500E-003$	$2.429394687752495E-003$
2	$2.211389680262243E-007$	$2.304594889808609E-002$
3	$1.554312234475219E-014$	$1.988341588332609E-005$
4	$8.881784197001252E-016$	$1.773976521235454E-010$
5		$3.552713678800501E-015$

Dans ce tableau nous approchons par les deux méthodes un vecteur propre de l'espace E_4. Nous pouvons de plus noter que ce même vecteur ou un vecteur lui étant proportionnel est retrouvé en solution approchée quand nous utilisons la méthode NTEA1 avec des conditions de départ différentes en particulier avec : $y = u_1$ mais $x_0 = (0.5, -1.5, 1.5, -2.5)^T$ proche de u_4, $y = u_3$ et x_0 proche de u_3, $y = u_1$ et $x_0 = (0.5, 1.5, 2.5, 3.5)$ proche de u_1. Nous verrons de plus que cette remarque se vérifie dans les chapitres suivants à savoir, quelque soit le vecteur initial choisi et le vecteur propre utilisé pour définir y, la solution approchée est un vecteur de l'espace propre E_4.

Chapitre II
Généralisation au cas non linéaire de méthodes CGM

1 Les méthodes NCGM : Algorithmes et résultats de convergence

Nous considérons le système de p équations linéaires à p inconnues

$$Ax = b.$$

Pour résoudre ce système nous avons abordé, au chapitre 1, la méthode de Lanczos qui consiste en la construction de la suite de vecteurs résidus

$$r_k = P_k(A)r_0$$

avec P_k la famille de polynômes orthogonaux par rapport à la fonctionnelle linéaire c définie, y étant un vecteur arbitraie non nul, par

$$c(\xi^i) = c_i = (y, A^i r_0),$$

où nous notons $(\ldots)$ le produit scalaire tel que

$$(\alpha x, \beta y) = \alpha\overline{\beta}(x, y) = \alpha\overline{\beta} \sum_{i=1}^{p} x_i \overline{y_i}$$

pour α et β des nombres complexes et x et y deux vecteurs de $\mathbb{C}^p$.

Dans ce chapitre, nous nous intéressons aux méthodes CGM [14, 28] , pour lesquelles le vecteur résidu à l'itération k, r_k, est construit comme suit

$$r_k = Q_k(A)P_k(A)r_0$$

où Q_k est un polynôme arbitraire de degré inférieur ou égal à k tel que $Q_k(0) = 1$ et P_k est le polynôme précédemment utilisé dans la méthode de Lanczos.

Cette classe de méthodes CGM inclut en particulier la méthodes CGS (Conjugate Gradient Squared) [44] vue antérieurement et qui correspond au choix $P_k = Q_k$. D'autres méthodes CGM sont obtenues grâce aux différents choix possibles pour les polynômes Q_k (cf. [14]). Ce choix est établi afin de rendre facile le calcul récursif du produit P_kQ_k ou, du moins, de ne pas nécessiter trop de mise en mémoire de vecteurs auxiliaires.

Dans ce qui suit, nous allons utiliser la norme euclidienne sur les vecteurs $\|x\|$, et la norme infinie sur les matrices $\|A\| = \max_{i=1\ldots p} \sum_{j=1}^{k} |a_{ij}|$, où A est

une matrice $p \times k$ dont les coefficients sont notés a_{ij}.
Nous voulons alors résoudre le système de point fixe $G(x) = x$, où
$G : D \subset \mathbb{C}^p \rightarrow \mathbb{C}^p$ avec D un ouvert convexe de $\mathbb{C}^p$. Nous notons x^* une solution de ce système et J la matrice jacobienne $G'(x^*)$. Nous supposons que G satisfait l'hypothèse **H** suivante

$$\mathbf{H}\begin{cases} \text{la matrice } I - J \text{ est régulière.} \\ \text{la dérivée de Fréchet } G' \text{ de } G \text{ satisfait la condition de Lipschitz} \\ \quad \exists L > 0, \forall x, y \in D, \|G'(x) - G'(y)\| \leq L\|x - y\|. \end{cases}$$

Nous notons H_k la matrice suivante

$$H_k = \begin{pmatrix} 1 & \cdots & 1 \\ (y, \frac{\Delta u_0}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_{d_k}}{\|\Delta u_0\|}) \\ \vdots & \ddots & \vdots \\ (y, \frac{\Delta u_{d_k-1}}{\|\Delta u_0\|}) & \cdots & (y, \frac{\Delta u_{2d_k-1}}{\|\Delta u_0\|}) \end{pmatrix}.$$

Maintenant nous allons nous intéresser à des méthodes CGM particulières. Nous rappelons brièvement en quoi consiste la méthode CGM utilisée puis nous établissons sa généralisation au cas non linéaire. De telles généralisations seront appelées NCGM, Nonlinear Conjugate Gradient Multiplied. Des théorèmes de convergence quadratique locale sont ensuite établis.

1.1 NCGS

Nous rappelons que le CGS est la méthode CGM utilisant $Q_k = P_k$, c'est à dire la méthode construisant la suite de vecteurs résidus

$$r_k = P_k(A)^2 r_0.$$

La généralisation de cette méthode à la résolution de systèmes non linéaires a été traitée dans le premier chapitre et a été notée NCGS (Nonlinear Conjugate Gradient Squared). Sa convergence quadratique locale a alors été démontrée.

1.2 iNCGS

Nous noterons iCGS les méthodes pour lesquelles le vecteur résidu à l'itération k, r_k^i, se calcule par

$$r_k^i = P_i(A)P_k(A)r_0^i \quad \text{pour} \quad i = 1, \ldots, k-1,$$

avec P_i le polynôme de degré au plus i appartenant à la famille de polynômes orthogonaux P_k définis dans la méthode de Lanczos. Ces méthodes ont été généralisées au cas non linéaire dans le premier chapitre sous le nom de méthodes iNCGS. Leur convergence quadratique locale a alors été traitée.

1.3 NCG-Min

1.3.1 CG-Min

La méthode CG-Min utilise les polynômes de moindres carrés [13] soit les polynômes $Q_k^{(m)}$, polynômes de degré au plus k tels que

$$\sum_{k=0}^{m} [c(\xi^i Q_k^{(m)})]^2 = min \quad \text{pour} \quad m = k-1, \ldots .$$

En particulier nous considérons le polynôme $Q_1^{(k)}$ normalisé par la condition

$$Q_1^{(k)}(0) = 1.$$

En l'écrivant $Q_1^{(k)}(\xi) = 1 - a_k \xi$, nous obtenons

$$a_k = \frac{c_0 c_1 + \ldots + c_k c_{k+1}}{c_1^2 + \ldots + c_{k+1}^2}.$$

Nous rappelons que le calcul des polynômes P_k nécessite la connaissance des moments $c_0, \ldots, c_{2k-1}$. Ces mêmes moments nous permettent de calculer le coefficient a_{2k-2}. Nous allons donc construire la suite de vecteurs résidus suivante

$$r_k = Q_1^{(2k-2)}(A) P_k(A) r_0 \quad \text{pour} \quad k = 1, 2, \ldots .$$

La méthode permettant de résoudre le système

$$Ax = b$$

en construisant une suite de vecteurs x_k approchant la solution et tels que le $k^{\text{ième}}$ vecteur résidu $r_k = b - Ax_k$ vérifie l'égalité ci-dessus est appelée CG-Min à cause de la propriété de minimisation des polynômes orthogonaux de moindres carrés $Q_1^{(k)}$.

1.3.2 Lien avec l'epsilon algorithme topologique

Nous nous plaçons dans le cas où $c_n = (y, \Delta u_n)$ avec u_n les vecteurs générés, u_0 étant choisi, par

$$u_{n+1} = (I - A)u_n + b \quad \text{pour} \quad n = 0, 1, \ldots.$$

Nous pouvons remarquer que dans le premier chapitre la suite u_n était itérée par $u_{n+1} = (I + A)u_n - b$; cette itération de u_n aurait donné dans ce chapitre un raisonnement et des des relations similaires en remplaçant $I - A$ par $I + A$ lorsque ça apparait.
A est une matrice carrée inversible, b est un vecteur, et y est un vecteur arbitraire non nul. Nous notons x la solution unique du système $Ax = b$. De plus nous avons toujours, avec $B = I - A$,

$$Q_j^{(i,n)}(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^j \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \cdots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_{n+i} & \cdots & c_{n+i+j} \\ \vdots & \cdots & \vdots \\ c_{n+i+j-1} & \cdots & c_{n+i+2j-1} \end{vmatrix}}.$$

Nous pouvons alors écrire

$$(2.1) \quad \varepsilon_{i+1,2k}^{(n)} = B\varepsilon_{i,2k}^{(n)} + b.$$

$$(2.2) \quad \varepsilon_{i,2k}^{(n)} - x = Q_k^{(0,n)}(B)(u_{n+i} - x).$$

Nous posons

$$T_1^{2k-2}(\xi) = \frac{Q_1^{(2k-2)}(\xi)}{Q_1^{(2k-2)}(1)} = \frac{\begin{vmatrix} 1 & \xi \\ a_{2k-2} & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ a_{2k-2} & 1 \end{vmatrix}}$$

où $a_{2k-2} = (c_0c_1 + \ldots + c_{2k-2}c_{2k-1})/(c_1^2 + \ldots + c_{2k-1}^2)$.
Nous posons aussi

$$CGMin_k = \frac{\begin{vmatrix} \varepsilon_{0,2k}^{(0)} & \varepsilon_{1,2k}^{(0)} \\ a_{2k-2} & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ a_{2k-2} & 1 \end{vmatrix}}.$$

Alors nous avons

$$CGMin_k - x = \frac{\begin{vmatrix} \varepsilon_{0,2k}^{(0)} - x & \varepsilon_{1,2k}^{(0)} - x \\ a_{2k-2} & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ a_{2k-2} & 1 \end{vmatrix}}.$$

Comme de plus grâce à (2.1) et à la définition de x nous avons

$$\varepsilon_{1,2k}^{(0)} - x = (B\varepsilon_{0,2k}^{(0)} + b) - (b + Bx) = B(\varepsilon_{0,2k}^{(0)} - x)$$

alors

$$CGMin_k - x = T_1^{(2k-2)}(B)(\varepsilon_{0,2k}^{(0)} - x).$$

La relation (2.2) nous permet alors d'écrire

$$CGMin_k - x = T_1^{(2k-2)}(B)Q_k^{(0,0)}(B)(u_0 - x).$$

Comme vu précédemment en posant

$$P_j^{(i,n)}(\xi) = Q_j^{(i,n)}(1 - \xi),$$

nous avons $Q_j^{(0,0)}(B) = P_j^{(0,0)}(A)$.
De même si nous posons

$$S_1^{(2k-2)}(\xi) = T_1^{(2k-2)}(1 - \xi),$$

nous avons $S_1^{(2k-2)}(A) = T_1^{(2k-2)}(B)$, et donc

$$r(CGMin_k) = S_1^{(2k-2)}(A)P_k^{(0,0)}(A)r(u_0).$$

1.3.3 Cas non linéaire

Nous considérons le système de point fixe $G(x) = x$, où $G : D \subset \mathbb{C}^p \to \mathbb{C}^p$ avec D un ouvert convexe de $\mathbb{C}^p$. Nous notons x^* une solution de ce système et J la matrice jacobienne $G'(x^*)$. Nous supposons que la suite (u_n) est générée, u_0 étant choisi, par

$$u_{n+1} = G(u_n)$$

et nous avons $c_n = (y, \Delta u_n)$ avec y un vecteur arbitraire non nul.
Pour résoudre le système $G(x) = x$ de p équations à p inconnues nous considérons l'algorithme suivant

- choisir un point initial x_0 dans D
- à l'itération k
  - poser $u_0 = x_k$
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$ où d_k est le degré du polynôme minimal de J en le vecteur $x_k - x^*$
  - calculer x_{k+1}=$CGMin_{d_k}$.

Vue la construction de $CGMin_{d_k}$ nous pouvons dire que la méthode non linéaire obtenue ci-dessus est issue de la méthode CG-Min ; c'est pourquoi nous l'appelons NCG-Min (Nonlinear CG-Min).

1.3.4 Résultat de convergence

Théorème 1 *Nous supposons que G satisfait l'hypothèse* **H**, *que* $\mid a_{2d_k-2} \mid$ *est borné indépendemment de k avec aussi $a_{2d_k-2} \neq 1$; si de plus il existe α strictement positif et un entier K tels que $\forall k \geq K, \mid detH_k \mid \geq \alpha$, alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|CGMin_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Démonstration
Nous avons

$$CGMin_{d_k} = \frac{\begin{vmatrix} \varepsilon^{(0)}_{0,2d_k} & \varepsilon^{(0)}_{1,2d_k} \\ a_{2d_k-2} & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ a_{2d_k-2} & 1 \end{vmatrix}}.$$

En développant le déterminant du numérateur suivant sa première ligne nous obtenons

$$CGMin_{d_k} = b_1 \varepsilon^{(0)}_{0,2d_k} + b_2 \varepsilon^{(0)}_{1,2d_k},$$

où $b_1 = \frac{1}{1-a_{2d_k-2}}$ et $b_2 = \frac{a_{2d_k-2}}{1-a_{2d_k-2}}$.
De plus $| a_{2d_k-2} |$ est supposée bornée indépendemment de k et $1 - a_{2d_k-2}$ est supposé non nul donc les coefficients b_1 et b_2 sont bornés indépendemment de k. Nous savons de plus, grâce aux théorèmes 1 et 2 du chapitre 1, dont les hypothèses sont ici vérifiées, que

$$\|\varepsilon^{(0)}_{j,2d_k} - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour} \quad j = 0, \ldots, d_k.$$

Nous pouvons alors conclure de

$$CGMin_{d_k} - x^* = b_1(\varepsilon^{(0)}_{0,2d_k} - x^*) + b_2(\varepsilon^{(0)}_{1,2d_k} - x^*)$$

avec b_1 et b_2 bornés indépendemment de k que

$$\|CGMin_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Nous obtenons donc la convergence quadratique locale de l'algorithme NCG-Min. ■

1.4 NCG-Adj

1.4.1 CG-Adj

La méthode CG-Adj se sert de deux familles de polynômes adjacents $\{P_k\}$ et $\{P^{(1)}_k\}$, d'où le nom de cette méthode.
La famille $\{P_k\}$ étant la famille de polynômes orthogonaux par rapport à la fonctionnelle c telle que

$$c(\xi^i) = c_i = (y, A^i r_0) \quad \text{pour} \quad i = 0, 1, \ldots,$$

$\{P^{(1)}_k\}$ est la famille de polynômes orthogonaux unitaires par rapport à la fonctionnelle linéaire $c^{(1)}$ définie par

$$c^{(1)}(\xi^i) = c(\xi^{i+1}) = c_{i+1} \quad \text{pour} \quad i = 0, 1, \ldots.$$

Les familles $\{P_k\}$ et $\{P_k^{(1)}\}$ sont alors dites adjacentes. Nous posons en outre

$$Q_k(\xi) = \xi^k P_k^{(1)}(\xi^{-1}).$$

La méthode CG-Adj résoud alors le système linéaire $Ax = b$ grâce à la construction d'une suite de vecteurs x_k tels que le vecteur résidu $r_k = b - Ax_k$ vérifie

$$r_k = Q_k(A)P_k(A)r_0.$$

1.4.2 Lien avec l'epsilon algorithme topologique

Nous nous plaçons à nouveau dans le cas où $c_n = (y, \Delta u_n)$ avec u_n les vecteurs générés, u_0 étant choisi, par

$$u_{n+1} = (I - A)u_n + b \quad \text{pour} \quad n = 0, 1, \ldots.$$

Les polynômes $Q_j^{(i,n)}$ sont définis comme précédemment et nous avons toujours $P_j^{(i,n)}(\xi) = Q_j^{(i,n)}(1 - \xi)$.
Nous posons

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^k \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix}}.$$

Par construction de $P_k^{(1)}$ nous avons en notre possession deux familles de polynômes orthogonaux adjacents $\{Q_k^{(0,0)}\}$ et $\{P_k^{(1)}\}$.

Nous prenons alors $Q_k(\xi) = \xi^k P_k^{(1)}(\xi^{-1})$. Nous avons donc

$$Q_k(\xi) = \begin{vmatrix} \xi^k & \cdots & 1 \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix} \Big/ \begin{vmatrix} 1 & \cdots & 1 \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix}.$$

Posons

$$CGAdj_k = \frac{\begin{vmatrix} \varepsilon_{k,2k}^{(0)} & \cdots & \varepsilon_{0,2k}^{(0)} \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_1 & \cdots & c_{k+1} \\ \vdots & \cdots & \vdots \\ c_k & \cdots & c_{2k} \end{vmatrix}}.$$

Nous avons alors

$$CGAdj_k - x = Q_k(B)P_k^{(0,0)}(A)(u_0 - x)$$

où $P_k^{(0,0)}$ est défini comme précédemment.
Soit maintenant R_k le polynôme tel que $Q_k(B) = R_k(A)$, alors nous avons

$$r(CGAdj_k) = R_k(A)P_k^{(0,0)}(A)r(u_0).$$

1.4.3 Cas non linéaire

Nous nous plaçons dans le cas de la résolution du système $G(x) = x$ avec les notations précédentes. La suite (u_n) est générée, u_0 étant choisi, par

$$u_{n+1} = G(u_n)$$

et nous avons $c_n = (y, \Delta u_n)$ avec y un vecteur arbitraire non nul.
La résolution de ce système peut se faire grâce à l'algorithme suivant

- choisir un point initial x_0 dans D
- à l'itération k
  - poser $u_0 = x_k$
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k, 2d_k + 1$ où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$
  - calculer x_{k+1}=$CGAdj_{d_k}$.

Etant donné la définition de $CGAdj_k$, nous pouvons dire que cette méthode généralise au cas non linéaire la méthode CGM appelée CG-Adj, c'est pourquoi nous la baptisons NCG-Adj.

Nous pouvons remarquer que dans cette méthode nous avons besoin des moments $c_0, \ldots, c_{2d_k}$, et non plus comme pour les méthodes précédentes des moments $c_0, \ldots, c_{2d_k-1}$; il faut donc à chaque itération k calculer un vecteur u_{2d_k+1} supplémentaire.

1.4.4 Résultat de convergence

Nous notons TH_k la matrice suivante

$$TH_k = \begin{pmatrix} 1 & \cdots & 1 \\ \frac{c_1}{\|\Delta u_0\|} & \cdots & \frac{c_{d_k+1}}{\|\Delta u_0\|} \\ \vdots & \cdots & \vdots \\ \frac{c_{d_k}}{\|\Delta u_0\|} & \cdots & \frac{c_{2d_k}}{\|\Delta u_0\|} \end{pmatrix}.$$

Théorème 2 *Si G satisfait l'hypothèse* **H** *et s'il existe α et β strictement positifs et un entier K tels que $\forall k \geq K, \mid detH_k \mid \geq \beta$ et $\mid detTH_k \mid \geq \alpha$, alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|CGAdj_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Démonstration

Nous avons

$$CGAdj_{d_k} = \frac{\begin{vmatrix} \varepsilon^{(0)}_{d_k,2d_k} & \cdots & \varepsilon^{(0)}_{0,2d_k} \\ c_1 & \cdots & c_{d_k+1} \\ \vdots & \cdots & \vdots \\ c_{d_k} & \cdots & c_{2d_k} \end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ c_1 & \cdots & c_{d_k+1} \\ \vdots & \ldots & \vdots \\ c_{d_k} & \cdots & c_{2d_k} \end{vmatrix}}.$$

En développant le déterminant du numérateur suivant sa première ligne nous avons l'écriture suivante

$$CGAdj_{d_k} = \sum_{j=0}^{d_k} a_j^{(k)} \varepsilon_{d_k-j,2d_k}^{(0)}.$$

Les coefficients $a_j^{(k)}$ sont solution du système

$$TH_k \begin{pmatrix} a_0^{(k)} \\ \vdots \\ a_{d_k}^{(k)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

En utilisant un lemme de Kato [33] et sachant qu'il existe α strictement positif et un entier K tels que $\forall k \geq K, \mid detTH_k \mid \geq \alpha$, nous obtenons

$$\|TH_k^{-1}\| \leq \frac{\|TH_k\|^{d_k}}{\alpha}.$$

De plus les lignes de la matrice TH_k se retrouvent toutes dans la matrice H_k sauf la ligne dont les éléments sont $c_{d_k}, \cdots, c_{2d_k}$. Nous ne devons donc ici nous intéresser qu'à cette dernière ligne. En effet

$$\|TH_k\| = sup(max_{1 \leq i \leq d_k+1, i \neq 2} \sum_{j=1}^{d_k+1} \mid h_{ij} \mid, \sum_{j=1}^{d_k+1} th_{d_k+1,j})$$

avec des notations évidentes.

Nous avons déjà prouvé que $max_{i=1}^{d_k+1} \sum_{j=1}^{d_k+1} \mid h_{ij} \mid$ est borné indépendemment de k donc $max_{i=1, i \neq 2}^{d_k+1} \sum_{j=1}^{d_k+1} \mid h_{ij} \mid$ l'est aussi.

De plus

$$\mid th_{d_k+1,j} \mid \leq \|y\| \frac{\|\Delta u_{d_k-1+j}\|}{\|\Delta u_0\|}$$

par l'inégalité de Cauchy-Schwarz, et

$$\frac{\Delta u_{d_k-1+j}}{\|\Delta u_0\|} = J^{d_k-1+j} \frac{\Delta u_0}{\|\Delta u_0\|} + O(\|x_k - x^*\|^2).$$

Ainsi pour tout j entre 1 et d_k, $\mid th_{d_k+1,j} \mid$ est borné, donc $\|TH_k\|$ et donc $\|TH_k^{-1}\|$ le sont. Nous en déduisons que les coefficients $a_j^{(k)}$ sont bornés indépendemment de k. De plus les résultats des deux premiers théorèmes du chapitre 1 nous disent

$$\|\varepsilon_{j,2d_k}^{(0)} - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour} \quad j = 0, \ldots, d_k.$$

L'écriture

$$CGAdj_{d_k} - x^* = \sum_{j=0}^{d_k} a_j^{(k)} (\varepsilon^{(0)}_{d_k - j, 2d_k} - x^*)$$

avec des coefficients $a_j^{(k)}$ bornés indépendemment de k nous permet alors de conclure que

$$\|CGAdj_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

■

1.5 NCG-Ort

1.5.1 CG-Ort

Nous connaissons déjà la fonctionnelle linéaire c sur l'espace des polynômes, fonctionnelle par rapport à laquelle la famille $\{P_k\}$ est une famille de polynômes orthogonaux ; nous définissons maintenant une seconde fonctionnelle ϵ sur l'espace des polynômes. Nous prenons alors pour $\{Q_k\}$ la famille de polynômes orthogonaux par rapport à ϵ normalisée par $Q_k(0) = 1$.
La méthode CG-Ort est alors la méthode construisant les vecteurs x_k qui approchent la solution du système $Ax = B$ tels que le $k^{\text{ième}}$ vecteur résidu $r_k = b - Ax_k$ vérifie

$$r_k = Q_k(A)P_k(A)r_0.$$

Nous retrouvons en cas particulier de cette méthode la méthode CGS pour $\epsilon = c$.

1.5.2 Lien avec l'epsilon algorithme topologique

Nous nous plaçons dans le cas où $c(\xi^i) = c_i = (y, \Delta u_i)$ avec u_i les vecteurs générés par

$$u_{i+1} = (I - A)u_i + b \quad \text{pour} \quad i = 0, 1, \ldots,$$

où u_0 est un vecteur arbitraire et y un vecteur arbitraire non nul.
Nous posons alors

$$\epsilon(\xi^i) = \epsilon_i = (z, \Delta u_i)$$

avec z un vecteur arbitraire non nul.

Soit

$$R_j^{(i,n)}(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^j \\ e_{n+i} & \cdots & e_{n+i+j} \\ \vdots & \cdots & \vdots \\ e_{n+i+j-1} & \cdots & e_{n+i+2j-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ e_{n+i} & \cdots & e_{n+i+j} \\ \vdots & \cdots & \vdots \\ e_{n+i+j-1} & \cdots & e_{n+i+2j-1} \end{vmatrix}},$$

et

$$CGOrt_k = \frac{\begin{vmatrix} \varepsilon_{0,2k}^{(0)} & \cdots & \varepsilon_{k,2k}^{(0)} \\ e_0 & \cdots & e_k \\ \vdots & \cdots & \vdots \\ e_{k-1} & \cdots & e_{2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ e_0 & \cdots & e_k \\ \vdots & \cdots & \vdots \\ e_{k-1} & \cdots & e_{2k-1} \end{vmatrix}}.$$

Les vecteurs $\varepsilon_{i,2k}^{(0)}$ sont toujours calculés grâce aux moments de la fonctionnelle linéaire c. Nous obtenons alors de façon similaire à celle utilisée dans la sous-section 1.4 du premier chapitre pour la méthode CGS

$$r(CGOrt_k) = T_k^{(0,0)}(A)P_k^{(0,0)}(A)r_0$$

où $T_j^{(i,n)}(\xi) = R_j^{(i,n)}(1-\xi)$: le $k^{\text{ième}}$ vecteur résidu obtenu ici est donc identique à celui obtenu par la méthode CG-Ort.

1.5.3 Cas non linéaire

Pour résoudre le système de point fixe $G(x) = x$ avec les notations précédentes nous considérons l'algorithme suivant

- choisir un point initial x_0 dans D
- à l'itération k

- poser $u_0 = x_k$
- calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$ où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$
- calculer x_{k+1}=$CGOrt_{d_k}$.

Cet algorithme sera noté NCG-Ort en rappel de la méthode linéaire CG-Ort qu'il généralise au cas non linéaire.
Si nous prenons z et y égaux nous retrouvons la méthode NCGS. Cependant, connaissant l'influence du choix de y sur les résultats il est intéressant de ne pas s'imposer, comme c'était le cas dans les méthodes iNCGS, un choix identique de y dans les deux étapes de calcul à savoir

- le calcul des vecteurs $\varepsilon_{i,2d_k}^{(0)}$
- le calcul du vecteur résultat.

1.5.4 Résultat de convergence

Nous notons $H_k(z)$ la matrice suivante

$$H_k(z) = \begin{pmatrix} 1 & \cdots & 1 \\ \frac{e_0}{\|\Delta u_0\|} & \cdots & \frac{e_{d_k}}{\|\Delta u_0\|} \\ \vdots & \cdots & \vdots \\ \frac{e_{d_k-1}}{\|\Delta u_0\|} & \cdots & \frac{e_{2d_k-1}}{\|\Delta u_0\|} \end{pmatrix},$$

avec $e_i = (z, \Delta u_i)$ et z un vecteur arbitraire non nul, à fortiori non identique au vecteur y déjà utilisé dans les moments $c_i = (y, \Delta u_i)$.
Avec cette notation nous avons en fait $H_k(y) = H_k$.

Théorème 3 *Si G satisfait l'hypothèse* **H** *et s'il existe α et β strictement positifs et un entier K tels que $\forall k \geq K, \mid detH_k \mid \geq \beta$ et $\mid detH_k(z) \mid \geq \alpha$, alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|CGOrt_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Démonstration

Nous avons

$$CGOrt_{d_k} = \frac{\begin{vmatrix} \varepsilon^{(0)}_{0,2d_k} & \cdots & \varepsilon^{(0)}_{d_k,2d_k} \\ e_0 & \cdots & e_{d_k+1} \\ \vdots & \cdots & \vdots \\ e_{d_k} & \cdots & e_{2d_k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ e_0 & \cdots & e_{d_k} \\ \vdots & \cdots & \vdots \\ e_{d_k-1} & \cdots & e_{2d_k-1} \end{vmatrix}}.$$

En développant le déterminant du numérateur suivant sa première ligne nous avons l'écriture suivante

$$CGOrt_{d_k} = \sum_{j=0}^{d_k} \epsilon_j^{(k)} \varepsilon_{j,2d_k^{(0)}}.$$

Les coefficients $\epsilon_j^{(k)}$ sont solution du système

$$H_k(z) \begin{pmatrix} \epsilon_0^{(k)} \\ \vdots \\ \epsilon_{d_k}^{(k)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

De plus seul le vecteur z est nouveau dans l'écriture de la matrice $H_k(z)$ par rapport à celle de H_k. En supposant qu'il existe α strictement positif et un entier K tels que $\forall k \geq K, \mid detH_k(z) \mid \geq \alpha$, nous pouvons donc aisément montrer que $\|H_k(z)^{-1}\|$ est bornée indépendemment de k et donc que les coefficients $\epsilon_j^{(k)}$ le sont.

En outre comme G satisfait l'hypothèse **H** et qu'il existe β strictement positif et un entier K tels que $\forall k \geq K, \mid detH_k \mid \geq \beta$, nous avons

$$\|\varepsilon^{(0)}_{j,2d_k} - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour} \quad j = 0, \ldots, d_k.$$

Ceci nous permet de conclure que

$$\|CGOrt_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Nous avons donc obtenue la convergence quadratique locale de la méthode NCG-Ort. ■

1.6 iNCG-Ort

i étant un indice entre 1 et d_k, nous appellerons iNCG-Ort toute méthode dont l'algorithme s'écrit comme suit

- choisir un point initial x_0 dans D
- à l'itération k
  - poser $u_0 = x_k$
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$ où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$
  - calculer x_{k+1}=$CGOrt^i_{d_k}$.

où

$$CGOrt^i_k = \frac{\begin{vmatrix} \varepsilon^{(0)}_{0,2k} & \cdots & \varepsilon^{(0)}_{i,2k} \\ e_0 & \cdots & e_i \\ \vdots & \cdots & \vdots \\ e_{i-1} & \cdots & e_{2i-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ e_0 & \cdots & e_i \\ \vdots & \cdots & \vdots \\ e_{i-1} & \cdots & e_{2i-1} \end{vmatrix}}.$$

Ces méthodes comprennent en cas particulier le NCG-Ort pour $i = d_k$. Avec les notations précédentes nous avons dans le cas linéaire

$$r^i_k = r(CGOrt^i_k) = T^{(0,0)}_i(A)P^{(0,0)}_k(A)r^i_0.$$

Nous pouvons donc dire que les algorithmes iNCG-Ort généralisent au cas non linéaire les algorithmes iCG-Ort, où iCG-Ort est la méthode CGM définie par les vecteurs r^i_k énoncés ci-dessus.

Pour e et c deux fonctionnelles linéaires égales nous retrouvons les méthodes iNCGS.

1.6.1 Résultat de convergence

Nous posons

$$H_{k,i}(z) = \begin{pmatrix} 1 & \cdots & 1 \\ \frac{e_0}{\|\Delta u_0\|} & \cdots & \frac{e_i}{\|\Delta u_0\|} \\ \vdots & \cdots & \vdots \\ \frac{e_{i-1}}{\|\Delta u_0\|} & \cdots & \frac{e_{2i-1}}{\|\Delta u_0\|} \end{pmatrix}$$

avec $e_j = (z, \Delta u_j)$ et z un vecteur arbitraire non nul.
Nous retrouvons donc avec cette notation $H_{k,d_k}(z) = H_k(z)$.

Théorème 4 *Pour i choisi entre 1 et d_k,*
si G satisfait l'hypothèse **H** *et s'il existe α et β strictement positifs et un entier K tels que $\forall k \geq K, \mid detH_k \mid \geq \beta$ et $\mid detH_{k,i}(z) \mid \geq \alpha$, alors il existe un voisinage U de x^* tel que, $\forall x_0 \in U$*

$$\|CGOrt^i_{d_k} - x^*\| = O(\|x_k - x^*\|^2).$$

Démonstration
Nous avons

$$CGOrt^i_{d_k} = \frac{\begin{vmatrix} \varepsilon^{(0)}_{0,2d_k} & \cdots & \varepsilon^{(0)}_{i,2d_k} \\ e_0 & \cdots & e_i \\ \vdots & \cdots & \vdots \\ e_i & \cdots & e_{2i-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ e_0 & \cdots & e_i \\ \vdots & \cdots & \vdots \\ e_{i-1} & \cdots & e_{2i-1} \end{vmatrix}}.$$

En développant le déterminant du numérateur suivant sa première ligne nous avons l'écriture suivante

$$CGOrt^i_{d_k} = \sum_{j=0}^{i} e^{k,i}_j \varepsilon_{j,2d^{(0)}_k}.$$

De plus ces coefficients vérifient $\sum_{j=0}^{i} e^{k,i}_j = 1$, nous avons donc

$$CGOrt^i_{d_k} - x^* = \sum_{j=0}^{i} e^{k,i}_j (\varepsilon_{j,2d^{(0)}_k} - x^*).$$

En outre les coefficients $e_j^{k,i}$ sont bornés indépendemment de k, la démonstration étant la même que pour les coefficients $e_j^{(k)}$ du paragraphe précédent avec cette fois $H_{k,i}(z)$ comme matrice du système et non $H_k(z)$.
Or nous avons émis les hypothèses suffisantes pour avoir

$$\|\varepsilon_{j,2d_k}^{(0)} - x^*\| = O(\|x_k - x^*\|^2) \quad \text{pour} \quad j = 0, \ldots, d_k.$$

Nous pouvons alors conclure

$$\|CGOrt_{d_k}^i - x^*\| = O(\|x_k - x^*\|^2).$$

Nous avons donc établi la convergence quadratique locale des méthodes iNCG-Ort pour $i = 1, \ldots, d_k$. ■

2 Exemples numériques

Dans cette partie nous reprenons les exemples numériques du premier chapitre; ceci nous permettra de comparer les méthodes utilisées ici avec les précédentes iNTEA et iNCGS.

Nous travaillons toujours en double précision.

Pour chaque exemple et chaque méthode utilisée nous affichons la norme infinie de l'erreur si nous connaissons le vecteur solution, et la norme infinie du vecteur différence entre le vecteur obtenu à l'itération et son image par la fonction G sinon.

Nous pouvons remarquer que les méthodes NCGM évoquées ici ne nécessitent aucune inversion de matrice ni aucun calcul de dérivée.

Nous prenons en pratique $d_k = p$, la dimension du système. Pour chacune des méthodes de ce chapitre le nombre d'évaluations de fonctions non linéaires est de $2p$ soit deux fois la dimension du système sauf pour la méthode NCG-Adj qui nécessite $2p + 1$ évaluations par itération.

2.1 Exemple 1

Cet exemple est donné dans [3].
Soit à trouver la solution unique $\begin{pmatrix} -1 \\ +1 \end{pmatrix}$ du système de dimension 2 ,

$G(x) = x$, où G est définie par

$$G_1(x) = -\frac{x_{(2)}^4}{4} - \frac{3}{4}$$
$$G_2(x) = -0.405e^{1-x_{(1)}^2} + 1.405$$

où nous notons $x_{(1)}$ et $x_{(2)}$ les coordonnées du vecteur de dimension 2, x, et $G(x) = \begin{pmatrix} G_1(x) \\ G_2(x) \end{pmatrix}$.

Nous prenons comme vecteur initial le vecteur $u_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Nous prenons $y = \begin{pmatrix} -0.75 \\ 0.405e + 1.405 \end{pmatrix}$.

Pour les méthodes iNCG-Ort nous avons besoin d'un second vecteur arbitraire z intervenant dans le produit scalaire $d_n = (z, \Delta u_n)$; nous le prendrons égal à

$$z = \begin{pmatrix} +2 \\ -1 \end{pmatrix}.$$

it.	NCG-Min	NCG-Adj
1	$5.004410185430234E-001$	$1.495247245347567E-001$
2	$3.936733854091679E-002$	$4.608339910333736E-002$
3	$4.513606756642918E-003$	$2.886827276635251E-003$
4	$5.264977381402858E-005$	$2.715537069764018E-005$
5	$6.640451063006190E-009$	$1.329227283797252E-009$
6	$1.998401444325282E-015$	$3.330669073875470E-016$

it.	NCG-Ort	iNCG-Ort avec $i = 1$
1	$1.429954514226767E-001$	$2.719389461334600E-001$
2	$3.117808695999214E-002$	$4.431120375192366E-002$
3	$2.739120512601501E-003$	$4.239740988388641E-003$
4	$2.542044618369932E-005$	$4.113825002205473E-005$
5	$2.230953310977668E-009$	$3.792533287416688E-009$
6	$2.886579564025407E-015$	$2.220446049250313E-016$
7	$4.440892098500626E-016$	

2.2 Exemple 2

Cet exemple se trouve dans [3].

Soit à trouver le point fixe $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ du système de dimension 2, $G(x) = x$ suivant

$$G_1(x) = \frac{x_{(2)}^2}{2} + x_{(1)} - \frac{1}{2}$$
$$G_2(x) = sin(x_{(1)}) + sin(x_{(2)} - 1) + 1.$$

Pour les méthodes NCG-Adj et iNCG-Ort, nous prenons comme vecteur initial $x_0 = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$.

Nous pouvons remarquer que ce choix du vecteur initial fait converger la méthode NCG-Min vers une autre solution que la solution souhaitée; en effet nous nous approchons alors du point fixe $\begin{pmatrix} \pi \\ 1 \end{pmatrix}$, et non de la solution $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ voulue. Pour converger vers ce point fixe nous allons choisir un vecteur de départ plus proche de la solution, à savoir nous prenons $x_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ pour la méthode NCG-Min.

Nous gardons en premier vecteur arbitraire $y = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, ce choix rendant la convergence plus rapide que le choix Δu_0.

Nous prenons en second vecteur arbitraire $z = \begin{pmatrix} -1 \\ +2 \end{pmatrix}$.

Nous obtenons les tableaux de résultats suivants

it.	NCG-Min	NCG-Adj
1	$2.357481636595310E-001$	$6.219704463260318E-001$
2	$3.774057609537318E-002$	$6.134991532915857E-002$
3	$8.759830960913817E-005$	$1.760568746077205E-002$
4	$4.456708314488893E-005$	$7.069023747989789E-004$
5	$2.010347723785392E-009$	$1.375469052347899E-006$
6	$5.133497286718863E-017$	$3.090232135295043E-012$

it.	NCG-Ort	iNCG-Ort avec $i = 1$
1	$1.582710589820964E-001$	$4.767453490171354E-001$
2	$3.184839862668171E-003$	$1.895902828252881E-002$
3	$5.388725340836153E-006$	$1.463053163870232E-002$
4	$4.530487096587876E-012$	$6.453608942961871E-007$
5	$1.604619215278547E-017$	$2.967637078639807E-010$
6		$1.680513367352532E-017$

Nous pouvons donc retenir de cet exemple que si le choix des vecteurs y et z est important, celui du vecteur initial x_0 ne l'est pas moins.

2.3 Exemple 3

Cet exemple est donné dans [41].
Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 2.25 & 0.01 & 0.05 & 0.50 \\ 0.01 & 1.75 & 0.00 & 0.05 \\ 0.05 & 0.00 & 1.75 & 0.01 \\ 0.50 & 0.05 & 0.01 & 2.25 \end{pmatrix}$$

$$b = (-0.81, -0.31, -0.31, -0.81)^T$$

et

$$Q(x) = -0.5(x_{(1)}^2 + x_{(1)}x_{(4)}, x_{(2)}^2, x_{(3)}^2, x_{(1)}x_{(4)} + x_{(4)}^2)^T.$$

La solution de ce système est $(1, 1, 1, 1)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (1.2, 1.2, 1.2, 1.2)^T$.
Pour le vecteur arbitraire y, le vecteur $(-1, 1, 2, 1)^T$ donne de meilleurs résultats que le vecteur $\Delta u_0 = (-0.409, -0.285, -0.287, -0.683)^T$.
En second vecteur arbitraire nous prenons $z = (3, 0, 1, 2)^T$.
Ces résultats sont les suivants:

it.	NCG-Min	NCG-Adj
1	$6.565247339349822E-003$	$5.294827636040189E-002$
2	$3.064190313795123E-005$	$2.514203424454076E-003$
3	$3.692435246449577E-011$	$5.446308206868977E-006$
4	$1.953992523340276E-014$	$2.989208880421756E-011$
5		$1.088018564132653E-014$

it.	NCG-Ort	iNCG-Ort avec $i = 3$
1	$1.693973254812686E-004$	$4.136288778431307E-004$
2	$1.877820121620744E-011$	$2.494240369799172E-010$
3	$1.998401444325282E-015$	$2.553512956637860E-014$

Dans cet exemple, nous pouvons noter l'obtention d'un chiffre exact supplémentaire à l'itération 3 de l'algorithme NCG-Ort par rapport à la plupart des autres algorithmes, ceux du chapitre 1 y compris.

2.4 Exemple 4

Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 3.9 & -3.7 & 2.4 & -0.6 \\ 2.4 & -2.0 & 2.2 & -0.6 \\ 2.4 & -3.6 & 4.1 & -0.9 \\ 2.8 & -5.2 & 4.8 & -0.4 \end{pmatrix}$$

$$b = (-0.75, -0.75, -0.75, -0.75)^T$$

et

$$Q(x) = -0.25(x_{(1)}^2, x_{(2)}^2, x_{(3)}^2, x_{(4)}^2)^T.$$

La solution de ce système est $(3, 3, 3, 3)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (2.5, 2.5, 2.5, 2.5)^T$.
Pour le vecteur arbitraire y, le vecteur $y = (-1, 1, 2, 1)^T$ donne de meilleurs résultats que le vecteur Δu_0.
Nous prenons en vecteur arbitraire z le vecteur $z = (3, 0, 1, 2)^T$.
Ces résultats sont les suivants:

it.	NCG-Min	NCG-Adj
1	$1.259919077956084E-006$	$1.259919085949690E-006$
2	$2.975397705995420E-014$	$1.509903313490213E-014$
3	$1.643130076445232E-014$	

it.	NCG-Ort	iNCG-Ort avec $i=2$
1	$1.259919066409765E-006$	$1.259919085505601E-006$
2	$1.776356839400250E-014$	$1.465494392505207E-014$

2.5 Exemple 5

Nous voulons résoudre le système de dimension 6, $G(x) = x$ avec G définie par

$$\begin{aligned}
G_1(x) &= -0.75 - \frac{x_{(2)}^2 x_{(4)} x_{(6)}}{4} \\
G_2(x) &= -0.405e^{1+x_{(1)}x_{(2)}} + 1.405 \\
G_3(x) &= \frac{x_{(4)}x_{(6)}}{2} - 1.5 \\
G_4(x) &= 0.605e^{1-x_{(3)}^2} + 0.395 \\
G_5(x) &= \frac{x_{(2)}x_{(6)}}{2} - 1.5 \\
G_6(x) &= x_{(1)}x_{(5)}.
\end{aligned}$$

La solution de ce système est $(-1, 1, -1, 1, -1, 1)^T$.
Nous prenons comme vecteur initial pour les méthodes NCG-Min et iNCG-Ort le vecteur

$$x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T.$$

Pour avoir la convergence de NCG-Adj il faut prendre un vecteur initial plus proche de la solution, nous prendrons donc pour cette méthode

$$x_0 = (-0.7, 0.7, -0.7, 0.7, -0.7, 0.7)^T.$$

En effet, si nous gardons le vecteur initial x_0 utilisé pour les autres méthodes nous obtenons le résultat suivant

- la première norme de l'erreur est égale à $4.866939014916561E-001$
- la deuxième norme égale 2.410867779672894

- puis il y a "overflow" dans le calcul de l'exponentiel.

Pour le vecteur arbitraire y, le vecteur $(=1,1,2,3,1,2)^T$ donne de bons résultats. Le vecteur arbitraire z utile aux méthodes iNCG-Ort sera pris égal à

$$(1,2,3,0,4,1)^T.$$

it.	NCG-Min	NCG-Adj
1	$9.714099076196114E-002$	$1.804732797225553E-001$
2	$4.462171746042021E-003$	1.143708661016041
3	$3.840461142434748E-006$	1.263588365715089
4	$4.368017059164231E-011$	$7.562185158661827E-002$
5	$6.883382752675971E-015$	$2.444578940576769E-002$
6		$5.426838828016312E-003$
7		$8.567754963784147E-005$
8		$4.396119379634911E-008$
9		$8.526512829121202E-014$

it.	NCG-Ort	iNCG-Ort avec $i = 3$
1	$2.394726065317898E-002$	$2.921903794778014E-002$
2	$1.607226217554825E-003$	$6.717715264595858E-004$
3	$1.790614745811325E-006$	$1.208450085554702E-007$
4	$1.036726260394971E-012$	$1.443289932012704E-015$
5	$2.987610159266296E-013$	
6	$4.840572387365683E-014$	

2.6 Exemple 6

Cet exemple est donné dans [20].
Nous considérons maintenant le système non linéaire de dimension 10, $G(x) = x$ avec G définie comme suit

$$G(x) = x - hF(x)$$

où

$$F_1(x) = (3 - 5x_{(1)})x_{(1)} + 1 - 2x_{(2)}$$
$$F_i(x) = (3 - 5x_{(i)})x_{(i)} + 1 - x_{(i-1)} - 2x_{(i+1)} \text{ pour } i = 2, \ldots, 9$$
$$F_{10}(x) = (3 - 5x_{(10)})x_{(10)} + 1 - x_{(9)}.$$

Nous prenons comme vecteur initial le vecteur

$$x_0 = (-1, -1, -1, -1, -1, -1, -1, -1, -1, -1)^T.$$

Pour le vecteur arbitraire y, le vecteur Δu_0 est un bon choix

$$\Delta u_0 = (1.25, 1, 1, 1, 1, 1, 1, 1, 1, 1.5)^T.$$

Pour le vecteur arbitraire z nous choisissons le vecteur suivant

$$z = (-1, 2, 1, -1, 3, 1, -1, 2, 1, 0)^T.$$

Dans cet exemple la solution n'est pas connue, aussi à chaque itération, nous afficherons la différence $(G(x_k) - x_k)$.

Dans le cas du choix $h = 0.2$, toutes les méthodes convergent comme nous pouvons le voir dans les tableaux de résultats:

it.	NCG-Min	NCG-Adj
1	$4.124926472152513E-003$	$5.407230884496084E-002$
2	$2.314136433795300E-005$	$5.284328970602270E-004$
3	$1.298262575222253E-008$	$9.433787100887603E-006$
4	$3.361755318564974E-013$	$1.891109491225507E-011$
5	$1.665334536937735E-015$	$3.508304757815495E-014$

it.	NCG-Ort	iNCG-Ort avec $i = 6$
1	$8.437836595364678E-004$	$6.202883534497383E-004$
2	$1.470107459655701E-007$	$7.713812638643347E-008$
3	$8.976153154094391E-011$	$2.775557561562891E-016$

Dans cet exemple la méthode iNCG-Ort pour $i = 6$ s'avère être la meilleure en donnant une norme infinie de la différence entre le vecteur obtenu

à l'itération et son image par G en 'E-016' à la troisième itération alors que toutes les autres méthodes utilisées jusque là, y compris celles du premier chapitre, donnent une norme infinie au minimum en 'E-014' à la troisième ou quatrième itération ou en 'E-015' à la cinquième itération.

2.7 Exemple 7

Nous construisons cet exemple comme le fait M. Altman dans [1]. Soient

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 2 & 0 & 1 \\ 2 & -1 & 3 & -1 \\ 3 & 2 & 2 & 2 \\ 4 & -1 & 1 & -2 \end{pmatrix}.$$

Nous calculons

$$A = PDP^{-1} = \begin{pmatrix} 3/8 & -5/8 & 9/8 & -3/8 \\ 97/32 & 153/32 & -69/32 & -33/32 \\ -41/16 & -1/16 & 61/16 & -23/16 \\ 159/32 & 103/32 & -123/32 & 33/32 \end{pmatrix}.$$

Nous résolvons alors le système $Ax - \dfrac{(x, Ax)}{(x, x)}x + x = x$, qui doit, étant donnée sa construction, nous donner en solution un vecteur propre de la matrice A. Les notations dans cet exemple restent celles du chapitre I.

Dans le tableau suivant nous prenons $y = u_1 = (1, 2, 3, 4)^T$, $z = u_2 = (2, -1, 2, -1)^T$, et $x_0 = (-0.5, 2.5, 1.5, 0.5)^T$ proche de u_3.

it.	NCG-Min	NCG-Adj
1	$2.499821342747333E-002$	$3.223091363037112E-003$
2	$1.291525787627812E-005$	$1.389435428016172E-003$
3	$1.776356839400250E-015$	$4.628987713140020E-007$
4	0.000000000000000	$2.078337502098293E-013$
5		$1.776356839400250E-015$

it.	NCG-Ort (i=4)
1	$1.693932341370497E-003$
2	$3.307626552859766E-004$
3	$1.774092872608435E-008$
4	$1.776356839400250E-015$

Ici comme au chapitre I nous pouvons remarquer que les différentes méthodes mises en oeuvre aboutissent toutes, et ce malgrè un vecteur initial choisi proche de u_3, à une convergence rapide vers un vecteur proportionnel à u_4.

3 Les méthodes incomplètes

Pour l'ensemble des algorithmes non linéaires émis dans ce chapitre et dans le précédent nous avons eu à établir une suite de vecteurs u_j comme suit :

$$u_j = G(u_{j-1}) \text{ pour } j = 1, \ldots, 2d_k \text{ ou } 2d_k + 1$$

avec d_k le degré du polynôme minimal de $G'(x^*)$ pour le vecteur $x_k - x^*$.

Pour les méthodes incomplètes issues des méthodes précédentes nous calculons cette fois

$$u_j = G(u_{j-1}) \text{ pour } j = 1, \ldots, 2m_k \text{ ou } 2m_k + 1 \text{ avec } m_k < d_k.$$

Nous obtenons ainsi les algorithmes suivants de résolution du système de point fixe habituel $G(x) = x$:

- choisir un point initial x_0 dans D
- à l'itération k
  - poser $u_0 = x_k$
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, l_k$ où $l_k = 2m_k$, sauf dans le cas du INC-Adj où $l_k = 2m_k + 1$, avec $m_k < d_k$ le degré du polynôme minimal de $G'(x^*)$ pour le vecteur $x_k - x^*$

$$- \text{ calculer } x_{k+1} = \begin{cases} \varepsilon^{(0)}_{i,2m_k} & \text{pour INC-iNTEA } i = 0, \ldots, m_k \\ G^{(0)}_i(m_k, 0) & \text{pour INC-iNCGS } i = 1, \ldots, m_k \\ CGMin_{m_k} & \text{pour INC-Min} \\ CGAdj_{m_k} & \text{pour INC-Adj} \\ CGOrt^i_{m_k} & \text{pour INC-iOrt } i = 1, \ldots, m_k \end{cases}$$

L'intérêt majeur de ces méthodes incomplètes est l'encombrement mémoire et le coût de calcul plus réduits.

Nous noterons INC-iNTEA, INC-iNCGS, INC-Min, INC-Adj, et INC-iOrt les méthodes incomplètes issues des méthodes iNTEA, iNCGS, NCG-Min, NCG-Adj, et iNCG-Ort. Nous avons vu qu'en pratique d_k étant inconnu nous prenons à la place la dimension p du système ; ici nous testerons donc les exemples en choisissant $m_k < p$.

Intéressons nous maintenant aux résultats de ces méthodes incomplètes dans le cas des exemples numériques étudiés précédemment. Dans chaque sous-chapitre nous redonnons le numéro des exemples auxquelles la méthode incomplète est appliquée ; Nous ne reparlons pas ici des données de ces exemples qui sont les mêmes que celles utilisées pour les méthodes complètes correspondantes.

3.1 INC-iNTEA

Ici nous calculons le vecteur $\varepsilon^{(0)}_{i,2m_k}$ à partir de $2m_k$ vecteurs u_j.

exemple 3 Dans cet exemple nous avons pris $d_k = 4$.

Nous constatons dans les tableaux de résultats ci-dessous que pour INC-iNTEA avec $i = 2$ les résultats sont similaires à ceux obtenus par iNTEA pour $i = 2$ mais en calculant cette fois 6 vecteurs par itération et non plus 8. Par INC-0NTEA nous obtenons une convergence plus lente vers la solution mais en construisant une suite de 4 vecteurs u_j par itération et non de 8 soit la moitié des vecteurs par itération.

it.	$i = 2, m_k = 3$	$i = 0, m_k = 2$
1	$4.203446308125747E-003$	$1.060537310649285E-001$
2	$2.660548270583263E-008$	$7.846276339619518E-002$
3	$1.976196983832779E-014$	$1.255104597154377E-002$
4		$3.148202385756527E-004$
5		$2.296943871771262E-007$
6		$1.709743457922741E-014$

exemple 4 Ici $d_k = 4$.

it.	$i = 0, m_k = 2$	$i = 2, m_k = 3$
1	$8.715239934460328E-003$	$1.970582879109450E-004$
2	$1.406703020734312E-008$	$6.217248937900877E-015$
3	$9.325873406851315E-015$	

Dans le cas du INC-iNTEA avec $i = 0$ nous obtenons d'aussi bons résultats que par la méthode complète avec 4 vecteurs u_j à calculer en moins par itération. Quant à INC-iNTEA avec $i = 2$ la convergence est plus rapide qu'avec le iNTEA pour $i = 2$ avec en plus moins de vecteurs à calculer par itération.

exemple 6 La dimension p du système est ici 10.

it.	$i = 0, m_k = 8$
1	$4.453926374287598E-003$
2	$4.059551874480150E-005$
3	$3.369706963463059E-008$
4	$2.333633286610848E-012$
5	$1.804112415015879E-013$

Nous obtenons ici de bons résultats avec 4 vecteurs u_j en moins par itération par rapport à la méthode complète.

3.2 INC-iNCGS

Dans cet algorithme nous calculons le vecteur $G_i^{(0)}(m_k, 0)$ à partir de $2m_k$ vecteurs u_j pour $i = 1, \ldots, m_k$.

exemple 3 Cet exemple est de dimension 4.

it.	$i = 2, m_k = 3$
1	$4.961802356084544E-002$
2	$5.988811064318034E-004$
3	$2.000295563631127E-007$
4	$7.038813976123492E-014$
5	$1.199040866595169E-014$

Les résultats sont à peu près identiques à ceux obtenus par le iNCGS avec $i = 2$ mais avec 2 vecteurs u_j en moins par itération.

exemple 6 La dimension de cet exemple est 10.

it.	$i = 4, m_k = 8$
1	$3.607077273740523E-004$
2	$3.365645739883405E-008$
3	$6.383782391594650E-015$

La méthode iNCGS avec $i = 4$ appliquée à cet exemple donnait un résultat similaire mais nécessitait 4 vecteurs u_j en plus par itération.

3.3 INC-Min

A chaque itération de cet algorithme nous posons $x_{k+1} = CGMin_{m_k}$ vecteur calculé à partir de $2m_k$ vecteurs u_j.

exemple 3 Dans cet exemple $p = 4$.

Les résultats sont ici légèrement meilleurs à ceux obtenus par la méthode complète INC-Min avec 2 vecteurs u_j en moins par itération.

it.	$m_k = 3$
1	$2.126394629028039E-002$
2	$3.869632687786284E-004$
3	$1.091064980585088E-007$
4	$7.438494264988549E-015$

exemple 6 La dimension de cet exemple est 10.

it.	$m_k = 8$
1	$9.667471446873632E-004$
2	$1.090442463186347E-005$
3	$5.607635467086425E-010$
4	$2.435274204515281E-012$
5	$5.162537064506978E-015$

La méthode NCG-Min appliquée à cet exemple donnait un résultat équivalent mais nécessitait 4 vecteurs u_j supplémentaires par itération.

3.4 INC-Adj

A chaque itération de cet algorithme nous calculons $x_{k+1} = CGAdj_{m_k}$. Pour cela nous avons à construire une suite de $2m_k + 1$ vecteurs u_j.

exemple 6 La dimension de cet exemple est 10.

it.	$m_k = 8$
1	$3.702303707123211E-004$
2	$3.470065346178330E-010$
3	$2.030042800527099E-013$
4	$1.054711873393899E-015$

La méthode incomplète INC-Adj appliquée à cet exemple numéro 6 donne une convergence plus rapide vers la solution que la méthode complète correspondante. En effet celle-ci donnait une norme de l'erreur en $E-011$ à la quatrième itération avec pour les quatre premières itérations 16 vecteurs u_j en plus à calculer par rapport à la méthode incomplète.

3.5 INC-iOrt

Nous calculons le vecteur $CGOrt^i_{m_k}$ à partir de $2m_k$ vecteurs u_j.

exemple 3 Dans cet exemple $p = 4$.

it.	$i = 3, m_k = 3$
1	$1.893573800126447E-003$
2	$1.879230030255858E-008$
3	$3.719247132494274E-014$

Les résultats sont à peu près identiques à ceux obtenus par le iNCG-Ort avec $i = 3$ avec 2 vecteurs u_j en moins par itération.

exemple 4 La dimension de cet exemple est 4.

it.	$i = 2, m_k = 2$
1	$8.715239934499408E-003$
2	$1.406785177238135E-008$
3	$1.421085471520200E-014$

La convergence de l'algorithme est ici plus lente que celle de l'algorithme complet iNCG-Ort pour $i = 2$ mais nous calculons 4 vecteurs u_j en moins par itération. Nous avons donc ici un choix à faire entre le coût et la rapidité de convergence des algorithmes.

exemple 6 La dimension de cet exemple est 10.

it.	$i = 6. m_k = 7$
1	$4.953009854443957E-002$
2	$5.505184331243695E-004$
3	$1.709104469060918E-009$
4	$3.885780586188048E-016$

Les résultats issus de la méthode incomplète restent corrects par rapport à ceux obtenus grâce à la méthode complète correspondante alors que nous calculons ici 6 vecteurs u_j en moins par itération.

Chapitre III
Les polynômes biorthogonaux et les systèmes non linéaires

1 Les polynômes biorthogonaux

Cette première partie de chapitres comme les deux suivantes s'appuie sur les résultats émis dans [8, 9].
Nous donnons tout d'abord des résultats préliminaires sur les polynômes biorthogonaux ; pour la plupart des définitions et règles données nous rappelons entre parenthèses les résultats parallèles utilisés pour définir les polynômes orthogonaux.
Nous notons $L_0, L_1, \ldots$ des formes linéaires définies sur l'espace des polynômes à coefficients complexes (pour les polynômes orthogonaux nous n'avions eu besoin que d'une forme linéaire c définie sur le même espace).
Nous notons P_k le polynôme de degré au plus k tel que

$$L_i(P_k) = 0 \quad \text{pour} \quad i = 0, \cdots, k-1. \tag{1}$$

(nous notions P_k le polynôme de degré au plus k tel que $c(\xi^i P_k) = 0$ pour $i = 0, \ldots, k-1$, ce qui définissait à un facteur multiplicatif près la famille de polynômes orthogonaux P_k par rapport à la fonctionnelle linéaire c).
Nous posons enfin $c_{ij} = L_i(\xi^j)$ et nous définissons la forme linéaire c par

$$c(\xi^i \bar{\xi}^k) = c_{ki} = L_k(\xi^i)$$

(pour les polynômes orthogonaux nous avions utilisé des moments à un seul indice $c_i = c(\xi^i)$).
La famille de polynômes P_k est alors une famille de polynômes biorthogonaux par rapport à la forme linéaire c ; Les polynômes biorthogonaux P_k sont complètement définis à un facteur multiplicatif près par les conditions de biorthogonalité (1).
Nous pourrons également utiliser la forme linéaire $c^{(1)}$ définie par

$$c^{(1)}(\xi^i \bar{\xi}^k) = c_{k,i+1}$$

et la famille $P_k^{(1)}$ de polynômes biorthogonaux par rapport à $c^{(1)}$ (définie pour les polynômes orthogonaux par la famille $P_k^{(1)}$ de polynômes orthogonaux par rapport à la forme linéaire $c^{(1)}$ telle que $c^{(1)}(\xi^i) = c(\xi^{i+1}) = c_{i+1}$).

Lemme 1 *Soient l_i des formes linéaires définies sur l'espace des polynômes et soit Q_k le polynôme tel que*

$$l_i(Q_k) = 0 \quad \textit{pour} \quad i = 0, \ldots, k-1$$

et $Q_k(1) = 1$.

Soient L_i *les formes linéaires définies par*

$$L_i(\xi^j) = \sum_{p=0}^{i} a_{ip} l_p((1-\xi)^j)$$

avec les a_{ii} *tous non nuls.*

Alors le polynôme P_k *tel que*

$$P_k(\xi) = Q_k(1-\xi)$$

satisfait aux conditions

$$L_i(P_k) = 0 \quad \textit{pour} \quad i = 0, \ldots, k-1$$

et $P_k(0) = 1$.

Nous pouvons remarquer qu'avec les normalisations $Q_k(1) = 1$ ou $P_k(0) = 1$ les polynômes biorthogonaux du lemme sont définis de manière unique (et non plus à un facteur multiplicatif près).

2 Transformation de suites de vecteurs

Nous cherchons à résoudre le système

$$Ax = b.$$

Soit (u_k) la suite de vecteurs générés par

$$u_0 \text{ donné et } \quad u_{k+1} = Bu_k + b$$

où u_0 est un vecteur arbitraire et B est la matrice telle que $A = I - B$.

La résolution du système peut se faire par la transformation de la suite (u_k) en la suite de vecteurs (x_k) définis par

$$x_k = \frac{\begin{vmatrix} u_0 & \cdots & u_k \\ d_{00} & \cdots & d_{0k} \\ \vdots & & \vdots \\ d_{k-1,0} & \cdots & d_{k-1,k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ d_{00} & \cdots & d_{0k} \\ \vdots & & \vdots \\ d_{k-1,0} & \cdots & d_{k-1,k} \end{vmatrix}} \quad \text{pour} \quad k = 0, 1, \ldots,$$

les d_{ij} étant des scalaires.
Dans cette définition le déterminant du numérateur représente en fait le vecteur obtenu en développant ce déterminant par rapport à sa première ligne suivant les règles habituelles.

Le choix des d_{ij} est multiple mais nous pouvons cependant citer certains choix connus :

- $$d_{ij} = (z, \Delta u_{i+j})$$

  avec z un vecteur arbitraire.
  Nous retrouvons alors l'epsilon algorithme topologique ou TEA [6] qui avait été défini précédemment grâce aux moments à un indice

  $$c_i = (z, \Delta u_i).$$

  Nous avons en effet

  $$\forall i, \forall j, \exists k \text{ tel que } d_{ij} = (z, \Delta u_{i+j}) = (z, \Delta u_k) = c_k$$

  d'où la similitude entre les deux définitions.

- $$d_{ij} = (\Delta u_i, \Delta u_j).$$

  La suite de vecteurs (x_k) définie avec ces moments représente la méthode MPE ou Minimal Polynomial Extrapolation method [21, 27, 47].

- $$d_{ij} = (\Delta^2 u_i, \Delta u_j).$$

  Ce choix donne lieu à la méthode RRE ou Reduced Rank Extrapolation method [24, 37, 32].

- $$d_{ij} = (z_i, \Delta u_j)$$

  avec z_i des vecteurs arbitraires.
  La méthode issue de ce choix est nommée MMPE ou Modified Minimal Polynomial Extrapolation method [6, 43, 39].

Nous pouvons remarquer que les moments définis en deuxième, troisième et quatrième choix ne peuvent pas directement s'apparenter à des moments à un seul indice et qu'il n'est donc pas possible de les rapprocher par ce biais d'une méthode utilisée dans les précédents chapitres comme nous l'avions fait pour le premier choix.

3 Lien avec les méthodes de type Lanczos

Nous posons

$$Q_k(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^k \\ d_{00} & \cdots & d_{0k} \\ \vdots & & \vdots \\ d_{k-1,0} & \cdots & d_{k-1,k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ d_{00} & \cdots & d_{0k} \\ \vdots & & \vdots \\ d_{k-1,0} & \cdots & d_{k-1,k} \end{vmatrix}},$$

les moments d_{ij} étant ceux utilisés ci-dessus.
Nous définissons par ailleurs les formes linéaires l_i par

$$l_i(\xi^j) = d_{ij}.$$

Alors nous avons

$$l_i(Q_k) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1 \quad \text{et} \quad Q_k(1) = 1.$$

Nous pouvons utiliser le lemme énoncé dans la première section de ce chapitre comme suit :
Soient L_i les formes linéaires définies par

$$L_i(\xi^j) = c_{ij} = \sum_{p=0}^{i} a_{ip} l_p((1-\xi)^j)$$

avec $\forall i, a_{ii} \neq 0$, alors le polynôme P_k défini par

$$P_k(\xi) = Q_k(1-\xi)$$

satisfait les conditions de biorthogonalité

$$L_i(P_k) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1,$$

et $P_k(0) = 1$.

P_k est alors entièrement défini par ces conditions et nous pouvons écrire

$$P_k(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^k \\ c_{00} & \cdots & c_{0k} \\ \vdots & & \vdots \\ c_{k-1,0} & \cdots & c_{k-1,k} \end{vmatrix}}{\begin{vmatrix} c_{00} & \cdots & c_{0k} \\ \vdots & & \vdots \\ c_{k-1,0} & \cdots & c_{k-1,k} \end{vmatrix}}.$$

Notons maintenant $P_k^{(1)}$ le polynôme unitaire défini par

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} c_{01} & \cdots & c_{0,k+1} \\ \vdots & & \vdots \\ c_{k-1,1} & \cdots & c_{k-1,k+1} \\ 1 & \cdots & \xi^k \end{vmatrix}}{\begin{vmatrix} c_{01} & \cdots & c_{0,k+1} \\ \vdots & & \vdots \\ c_{k-1,1} & \cdots & c_{k-1,k+1} \end{vmatrix}}.$$

P_k et $P_k^{(1)}$ existent et sont uniques si et seulement si le déterminant du dénominateur dans leur écriture est non nul. Alors

$$L_i(\xi P_k^{(1)}(\xi)) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1.$$

Les familles P_k et $P_k^{(1)}$ sont dites des familles adjacentes de polynômes biorthogonaux et nous avons les relations de récurrence suivantes entre ces polynômes :

$$\begin{cases} P_{k+1}(\xi) = P_k(\xi) - \lambda_k \xi P_k^{(1)}(\xi) \quad \text{pour} \quad k = 0, 1, \ldots \\ P_0(\xi) = P_0^{(1)}(\xi) = 1 \end{cases} \tag{2}$$

$$P_{k+1}^{(1)}(\xi) = \alpha_k P_{k+1}(\xi) + \sum_{i=0}^{k} \beta_{ki} P_i^{(1)}(\xi) \tag{3}$$

où les coefficients $\lambda_k, \alpha_k, \beta_{ki}$ se calculent grâce aux conditions de biorthogonalité énoncées ci-dessus sur les polynômes P_k et $P_k^{(1)}$ et à leur définition : pour avoir $L_k(P_{k+1}) = 0$ nous prenons

$$\lambda_k = L_k(P_k)/L_k(\xi P_k^{(1)}).$$

$P_k^{(1)}$ étant unitaire nous devons de plus poser

$$\alpha_k = -1/\lambda_k.$$

Les coefficients $\beta_{k0}, \ldots, \beta_{kk}$ se calculent eux grâce au système triangulaire obtenu par

$$L_i(\xi P_{k+1}^{(1)}) = \alpha_k L_i(\xi P_{k+1}) + \sum_{j=0}^{k} \beta_{kj} L_i(\xi P_j^{(1)}) = 0 \quad \text{pour} \quad i = 0, \ldots, k.$$

Si nous posons de plus $V_{k+1}(\xi) = -\lambda_k P_{k+1}^{(1)}(\xi)$, nous avons alors

$$L_i(\xi V_k) = 0 \quad \text{pour} \quad i = 0, \ldots, k-1$$

$$P_{k+1}(\xi) = P_k(\xi) - \lambda_k' \xi V_k(\xi) \tag{4}$$

$$V_{k+1}(\xi) = P_{k+1}(\xi) + \sum_{i=0}^{k} \beta_{ki}' V_i(\xi), \tag{5}$$

avec

$$\lambda_k' = L_k(P_k)/L_k(\xi V_k)$$

et comme nous avons

$$L_i(\xi V_{k+1}) = L_i(P_{k+1}) + \sum_{j=0}^{k} \beta_{kj}' L_i(\xi V_j) = 0 \quad \text{pour} \quad i = 0, \ldots, k,$$

nous obtenons le système triangulaire suivant

$$\begin{cases} \beta_{k0}' L_0(\xi V_0) = -L_0(\xi P_{k+1}) \\ \beta_{k0}' L_1(\xi V_0) + \beta_{k1}' L_1(\xi V_1) = -L_1(\xi P_{k+1}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \beta_{k0}' L_k(\xi V_0) + \ldots + \beta_{kk}' L_k(\xi V_k) = -L_k(\xi P_{k+1}), \end{cases}$$

système fournissant les scalaires $\beta'_{k0}, \ldots, \beta'_{kk}$.
Les relations de récurrence (4) et (5), vues leurs similitudes avec les relations de récurrence utilisées pour écrire Lanczos-Orthomin avec des polynômes orthogonaux, nous permettent d'écrire une généralisation de Lanczos-Orthomin avec des polynômes biorthogonaux. Nous appelons cet algorithme Bi-Orthomin. Il s'obtient comme suit :
Nous posons $r_k = P_k(A)r_0$ et $p_k = V_k(A)r_0$. Les relations de récurrence (4) et (5) donnent alors

$$r_{k+1} = r_k - \lambda'_k A p_k$$

et

$$p_{k+1} = r_{k+1} + \sum_{i=0}^{k} \beta'_{ki} p_i.$$

De plus comme $r_k = b - Ax_k$, nous obtenons

$$x_{k+1} = x_k + \lambda'_k p_k.$$

Nous avons alors l'algorithme Bi-Orthomin suivant

- choisir x_0
  poser $r_0 = b - Ax_0$, $p_0 = b - Ax_0$.
- pour $k = 0, 1, \cdots$ calculer

$$\lambda'_k = \frac{L_k(P_k)}{L_k(\xi V_k)}$$

$$x_{k+1} = x_k + \lambda'_k p_k$$

$$r_{k+1} = r_k - \lambda'_k A p_k.$$

- si r_{k+1} est non nul alors calculer

$$\beta'_{k0}, \ldots, \beta'_{kk} \text{ par } \begin{cases} \beta'_{k0} L_0(\xi V_0) = -L_0(\xi P_{k+1}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \beta'_{k0} L_k(\xi V_0) + \ldots + \beta'_{kk} L_k(\xi V_k) = -L_k(\xi P_{k+1}) \end{cases}$$

$$p_{k+1} = r_{k+1} + \sum_{i=0}^{k} \beta'_{ki} p_i.$$

Par ailleurs nous pouvons écrire une relation de la forme

$$P_{k+1}^{(1)}(\xi) = \xi P_k^{(1)} + \sum_{i=0}^{k} \beta_{ki} P_i^{(1)}. \tag{6}$$

Les équations (2) et (6) nous permettent alors de généraliser Lanczos-Orthodir en utilisant des polynômes biorthogonaux. En effet si nous posons $z_k = P_k^{(1)}(A)r_0$ et $r_k = P_k(A)r_0$ alors l'équation (2) donne

$$r_{k+1} = r_k - \lambda_k A z_k.$$

Comme de plus $r_k = b - Ax_k$ nous avons

$$x_{k+1} = x_k + \lambda_k z_k.$$

En outre l'équation (6) donne la relation

$$z_{k+1} = Az_k + \sum_{i=0}^{k} \beta_{ki} z_i.$$

Nous obtenons ainsi l'algorithme que nous appelons Bi-Orthodir

- choisir x_0
  poser $r_0 = b - Ax_0$, $z_0 = b - Ax_0$.
- pour $k = 0, 1, \cdots$ calculer

$$\lambda_k = \frac{L_k(P_k)}{L_k(\xi P_k^{(1)})}$$

$$x_{k+1} = x_k + \lambda_k z_k$$

$$r_{k+1} = r_k - \lambda_k' A z_k.$$

- si r_{k+1} est non nul alors calculer

$$\alpha_k = \frac{L_k(\xi P_k^{(1)})}{L_k(P_k)}$$

les coefficients $\beta_{k0} \ldots, \beta_{kk}$ par

$$\begin{cases} \alpha_k L_0(\xi P_{k+1}) + \beta_{k0} L_0(\xi P_0^{(1)}) = 0 \\ \alpha_k L_1(\xi P_{k+1}) + \beta_{k0} L_1(\xi P_0^{(1)}) + \beta_{k1} L_1(\xi P_1^{(1)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \alpha_k L_k(\xi P_{k+1}) + \beta_{k0} L_k(\xi P_0^{(1)}) + \ldots + \beta_{kk} L_k(\xi P_k^{(1)}) = 0 \end{cases}$$

$$z_{k+1} = A z_k + \sum_{i=0}^{k} \beta_{ki} z_i.$$

Nous pouvons noter qu'ici nous n'avons pas de récurrence à trois termes reliant les polynômes biorthogonaux P_{k-1}, P_k, P_{k+1}, c'est pourquoi nous ne pouvons pas réécrire Lanczos-Orthores. Par contre il est certainement possible d'obtenir d'autres relations de récurrence sur les familles de polynômes biorthogonaux P_k et $P_k^{(1)}$ et, en combinant ces relations, d'obtenir des généralisations d'autres méthodes de type Lanczos comme il a été fait dans [2] avec les polynômes orthogonaux.

4 Extension aux systèmes non linéaires

4.1 Algorithmes

Nous considérons le problème de point fixe $G(x) = x$, où $G : D \subset \mathbb{C}^p \to \mathbb{C}^p$ avec D un ouvert convexe de $\mathbb{C}^p$. Nous notons x^* une solution de ce système et J la matrice jacobienne $G'(x^*)$. Nous supposons que la suite $(u_n)_{n \in \mathbb{N}}$ est générée, u_0 étant choisi, par

$$u_{n+1} = G(u_n).$$

4.1.1 NTEA1

Soit z un vecteur arbitraire non nul. Pour résoudre le système $G(x) = x$ de p équations à p inconnues nous considérons l'algorithme suivant

- choisir un point initial x_0 dans D,
- à l'itération k
  - poser $u_0 = x_k$,

- calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, 2d_k$, où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$,
- calculer x_{k+1} où

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ (z, \Delta u_0) & \cdots & (z, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (z, \Delta u_{d_k-1}) & \cdots & (z, \Delta u_{2d_k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (z, \Delta u_0) & \cdots & (z, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (z, \Delta u_{d_k-1}) & \cdots & (z, \Delta u_{2d_k-1}) \end{vmatrix}}.$$

Comme nous l'avons vu précédemment nous retrouvons alors le NTEA1 du premier chapitre. En effet l'usage des polynômes biorthogonaux aboutit au calcul à chaque itération k du vecteur x_{k+1} tel que

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ d_{00} & \cdots & d_{0,d_k} \\ \vdots & & \vdots \\ d_{d_k-1,0} & \cdots & d_{d_k-1,d_k} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ d_{00} & \cdots & d_{0,d_k} \\ \vdots & & \vdots \\ d_{d_k-1,0} & \cdots & d_{d_k-1,d_k} \end{vmatrix}}.$$

et l'usage des polynômes orthogonaux donnait lieu à la méthode NTEA1 avec le calcul à chaque itération k du vecteur x_{k+1} tel que

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ c_0 & \cdots & c_{d_k} \\ \vdots & & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ c_0 & \cdots & c_{d_k} \\ \vdots & & \vdots \\ c_{d_k-1} & \cdots & c_{2d_k-1} \end{vmatrix}}.$$

Or ayant ici l'égalité $d_{ij} = (z, \Delta u_{i+j}) = c_{i+j}$, les deux vecteurs x_{k+1} écrits ci-dessus sont identiques.

4.1.2 NMPE

Nous considérons maintenant l'algorithme suivant

- choisir un point initial x_0 dans D,
- à l'itération k
  - poser $u_0 = x_k$,
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, d_k + 1$, où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$,
  - calculer x_{k+1} où

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ (\Delta u_0, \Delta u_0) & \cdots & (\Delta u_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (\Delta u_{d_k-1}, \Delta u_0) & \cdots & (\Delta u_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (\Delta u_0, \Delta u_0) & \cdots & (\Delta u_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (\Delta u_{d_k-1}, \Delta u_0) & \cdots & (\Delta u_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}.$$

Nous donnerons dans la sous-section 4.1.5 un algorithme de calcul récursif de ces vecteurs. La méthode non linéaire obtenue alors est une extension du MPE d'où le nom NMPE pour Nonlinear Minimal Polynomial Extrapolation method. Nous avons en effet pris $d_{ij} = (\Delta u_i, \Delta u_j)$ comme pour la méthode linéaire MPE. Dans cet algorithme le nombre d'évaluations de fonctions non linéaires est $d_k + 1$ par itération.

4.1.3 NRRE

Nous notons NRRE ou Nonlinear Reduced Rank Extrapolation method l'algorithme suivant

- choisir un point initial x_0 dans D,

- à l'itération k
  - poser $u_0 = x_k$,
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, d_k + 1$, où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$,
  - calculer x_{k+1} où

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ (\Delta^2 u_0, \Delta u_0) & \cdots & (\Delta^2 u_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (\Delta^2 u_{d_k-1}, \Delta u_0) & \cdots & (\Delta^2 u_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (\Delta^2 u_0, \Delta u_0) & \cdots & (\Delta^2 u_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (\Delta^2 u_{d_k-1}, \Delta u_0) & \cdots & (\Delta^2 u_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}.$$

la méthode NRRE utilise les moments d_{ij} vus pour la méthode RRE à savoir $d_{ij} = (\Delta^2 u_i, \Delta u_j)$.

4.1.4 NMMPE

Le NMMPE est le nom donné à la méthode non linéaire issue du MMPE. C'est-à-dire nous posons $d_{ij} = (z_i, \Delta u_j)$ où les z_i sont des vecteurs arbitraires et nous construisons l'algorithme NMMPE comme suit

- choisir un point initial x_0 dans D.
- à l'itération k
  - poser $u_0 = x_k$,
  - calculer $u_j = G(u_{j-1})$ pour $j = 1, \ldots, d_k + 1$, où d_k est le degré du polynôme minimal de J pour le vecteur $x_k - x^*$,

– calculer x_{k+1} où

$$x_{k+1} = \frac{\begin{vmatrix} u_0 & \cdots & u_{d_k} \\ (z_0, \Delta u_0) & \cdots & (z_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (z_{d_k-1}, \Delta u_0) & \cdots & (z_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (z_0, \Delta u_0) & \cdots & (z_0, \Delta u_{d_k}) \\ \vdots & & \vdots \\ (z_{d_k-1}, \Delta u_0) & \cdots & (z_{d_k-1}, \Delta u_{d_k}) \end{vmatrix}}.$$

Nous pouvons remarquer que dans cette méthode nous avons à choisir d_k vecteurs z_i ce qui est beaucoup pour un système de grande dimension. Cependant n'ayant pas de critère de choix optimal sur les vecteurs arbitraires intervenant dans les différentes méthodes il est intéressant d'avoir un échantillon de choix grand sur ces vecteurs, et de ne pas s'imposer d_k mêmes vecteurs $z_i = z$. Par ailleurs il peut être intéressant de prendre ces vecteurs égaux aux vecteurs de la base canonique ce qui diminue fortement le nombre de calculs à effectuer. Notons encore que ces d_k vecteurs z_i sont choisis en début d'algorithme et restent les mêmes pour toutes les itérations ; par contre dans le cas du NMPE et du NRRE nous utilisons les moments respectifs $(\Delta u_i, \Delta u_j)$ et $(\Delta^2 u_i, \Delta u_j)$ où la suite de vecteurs (u_j) est reconstruite à chaque itération k et donc où Δu_i et $\Delta^2 u_i$ sont recalculés à chaque nouvelle itération. Nous ne pouvons donc ni dire que la méthode NMPE est définie comme la méthode NMMPE avec $z_i = \Delta u_i$ ni dire que la méthode NRRE a la même définition que la méthode NMMPE avec $z_i = \Delta^2 u_i$. Il faudrait pour cela choisir d_k nouveaux vecteurs z_i par itération dans l'algorithme NMMPE.

4.1.5 Le $S\beta$-algorithme

Pour la première méthode non linéaire ci-dessus un calcul récursif des vecteurs x_{k+1} se fait grâce à l'epsilon algorithme topologique. Pour le NMPE, le NRRE, et le NMMPE les vecteurs x_{k+1} vont être calculés de manière récursive grâce au $S\beta$-algorithme [30] comme suit :

si nous posons

$$\bar{e}_k(u_n) = \frac{\begin{vmatrix} u_n & \cdots & u_{n+k} \\ (z_1, \Delta u_n) & \cdots & (z_1, \Delta u_{n+k}) \\ \vdots & & \vdots \\ (z_k, \Delta u_n) & \cdots & (z_k, \Delta u_{n+k}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (z_1, \Delta u_n) & \cdots & (z_1, \Delta u_{n+k}) \\ \vdots & & \vdots \\ (z_k, \Delta u_n) & \cdots & (z_k, \Delta u_{n+k}) \end{vmatrix}}$$

et

$$\beta_k^{(n)} = \frac{\begin{vmatrix} \Delta u_n & \cdots & \Delta u_{n+k} \\ (z_1, \Delta u_n) & \cdots & (z_1, \Delta u_{n+k}) \\ \vdots & & \vdots \\ (z_k, \Delta u_n) & \cdots & (z_k, \Delta u_{n+k}) \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ (z_1, \Delta u_n) & \cdots & (z_1, \Delta u_{n+k}) \\ \vdots & & \vdots \\ (z_k, \Delta u_n) & \cdots & (z_k, \Delta u_{n+k}) \end{vmatrix}}$$

alors nous avons l'algorithme suivant

$$\beta_0^{(n)} = \Delta u_n \qquad \text{et} \qquad \bar{e}_0(u_n) = u_n \quad \text{pour } k, n = 0, 1, \ldots$$

$$\bar{e}_{k+1}(u_n) = \frac{(z_{k+1}, \beta_k^{(n+1)})\bar{e}_k(u_n) - (z_{k+1}, \beta_k^{(n)})\bar{e}_k(u_{n+1})}{(z_{k+1}, \beta_k^{(n+1)}) - (z_{k+1}, \beta_k^{(n)})} \quad \text{pour } k, n = 0, 1, \ldots$$

$$\beta_{k+1}^{(n)} = \frac{(z_{k+1}, \beta_k^{(n+1)})\beta_k^{(n)} - (z_{k+1}, \beta_k^{(n)})\beta_k^{(n+1)}}{(z_{k+1}, \beta_k^{(n+1)}) - (z_{k+1}, \beta_k^{(n)})} \quad \text{pour } k, n = 0, 1, \ldots.$$

Cet algorithme a été obtenu grâce à une extension de l'identité de Sylvester.

Un autre algorithme permettant la programmation du MPE et du RRE est donné dans [42].

4.2 Résultats de convergence

Nous supposons que G satisfait l'hypothèse **H** suivante

$$\mathbf{H}\begin{cases} \text{la matrice } I-J \text{ est régulière,} \\ \text{la dérivée de Fréchet } G' \text{ de } G \text{ satisfait la condition de Lipschitz} \\ \quad \exists L>0, \forall x,y\in D, \|G'(x)-G'(y)\|\leq L\|x-y\|. \end{cases}$$

Nous notons $MMPE_k$ la matrice

$$MMPE_k=\begin{pmatrix} 1 & \cdots & 1 \\ (z_0,\frac{\Delta u_0}{\|\Delta u_0\|}) & \cdots & (z_0,\frac{\Delta u_{d_k}}{\|\Delta u_0\|}) \\ \vdots & & \vdots \\ (z_{d_k-1},\frac{\Delta u_0}{\|\Delta u_0\|}) & \cdots & (z_{d_k-1},\frac{\Delta u_{d_k}}{\|\Delta u_0\|}) \end{pmatrix},$$

MPE_k la matrice

$$MPE_k=\begin{pmatrix} 1 & \cdots & 1 \\ (\Delta u_0,\frac{\Delta u_0}{\|\Delta u_0\|^2}) & \cdots & (\Delta u_0,\frac{\Delta u_{d_k}}{\|\Delta u_0\|^2}) \\ \vdots & & \vdots \\ (\Delta u_{d_k-1},\frac{\Delta u_0}{\|\Delta u_0\|^2}) & \cdots & (\Delta u_{d_k-1},\frac{\Delta u_{d_k}}{\|\Delta u_0\|^2}) \end{pmatrix},$$

et RRE_k la matrice

$$RRE_k=\begin{pmatrix} 1 & \cdots & 1 \\ (\Delta^2 u_0,\frac{\Delta u_0}{\|\Delta u_0\|^2}) & \cdots & (\Delta^2 u_0,\frac{\Delta u_{d_k}}{\|\Delta u_0\|^2}) \\ \vdots & & \vdots \\ (\Delta^2 u_{d_k-1},\frac{\Delta u_0}{\|\Delta u_0\|^2}) & \cdots & (\Delta^2 u_{d_k-1},\frac{\Delta u_{d_k}}{\|\Delta u_0\|^2}) \end{pmatrix}.$$

Alors nous avons le résultat suivant

Théorème 1 *Si G satisfait l'hypothèse* **H** *et s'il existe α strictement positif et K tels que $\forall k\geq K, |\det H_k|\geq\alpha$ Alors il existe un voisinage U de x^* tel que, $\forall x_0\in U$*

$$\|x_{k+1}-x^*\|=O(\|x_k-x^*\|^2),$$

où x_{k+1} est le résultat à l'itération k obtenu par

$$\begin{cases} \textit{le NMPE quand } H_k = MPE_k \\ \textit{le NRRE quand } H_k = RRE_k \\ \textit{le NMMPE quand } H_k = MMPE_k \end{cases}$$

Ce théorème traduit donc les convergences quadratiques locales respectives des méthodes NMPE, NRRE, et NMMPE.

Démonstration

Les démonstrations se font de la même manière que dans [35] pour le NTEA1. Nous pouvons en effet écrire ici aussi

$$x_{k+1} = x_k - (a_0^k \Delta u_0 + \ldots + a_{d_k-1}^k \Delta u_{d_k-1})$$

où les a_i^k sont des coefficients connus. Pour montrer que ces coefficients sont bornés indépendemment de k nous avons à majorer $\|(MU_{0,d_k})^{-1}\|$ où

$$MU_{0,d_k} = \begin{pmatrix} (z_0, \Delta^2 u_0) & \cdots & (z_0, \Delta^2 u_{d_k-1}) \\ \vdots & & \vdots \\ (z_{d_k-1}, \Delta^2 u_0) & \cdots & (z_{d_k-1}, \Delta^2 u_{d_k-1}) \end{pmatrix}, \text{ pour le NMMPE.}$$

$$MU_{0,dk} = \begin{pmatrix} (\Delta u_0, \Delta^2 u_0) & \cdots & (\Delta u_0, \Delta^2 u_{d_k-1}) \\ \vdots & & \vdots \\ (\Delta u_{d_k-1}, \Delta^2 u_0) & \cdots & (\Delta u_{d_k-1}, \Delta^2 u_{d_k-1}) \end{pmatrix}, \text{ pour le NMPE.}$$

$$MU_{0,d_k} = \begin{pmatrix} (\Delta^2 u_0, \Delta^2 u_0) & \cdots & (\Delta^2 u_0, \Delta^2 u_{d_k-1}) \\ \vdots & & \vdots \\ (\Delta^2 u_{d_k-1}, \Delta^2 u_0) & \cdots & (\Delta^2 u_{d_k-1}, \Delta^2 u_{d_k-1}) \end{pmatrix}, \text{ pour le NRRE.}$$

C'est pour cela que pour les méthodes NMPE et NRRE nous avons pris des matrices H_k avec des coefficients divisés par $\|\Delta u_0\|^2$ et non par $\|\Delta u_0\|$ comme pour les autres méthodes. En effet pour le NMPE et le NRRE nous majorons cette fois $\|MU_{0,d_k}\|$ par $C\|x_k - x^*\|^2$ où C est indépendant de k et non plus par $C\|x_k - x^*\|$, d'où la nécessité d'avoir

$$det(MU_{0,d_k}) = \|\Delta u_0\|^{2d_k} \times det(H_k)$$

au lieu de $det(MU_{0,d_k}) = \|\Delta u_0\|^{d_k} \times det(H_k)$ comme pour les autres méthodes.

■

Il est à noter qu'une autre démonstration de la convergence de ces méthodes est donnée dans [31].

4.3 Exemples numériques

Nous rappelons qu'en pratique nous prenons à la place de d_k la dimension du système. Les calculs sont toujours effectués en double précision. Dans chaque exemple nous affichons la norme infinie de l'erreur si la solution est connue, sinon nous affichons la norme infinie du vecteur différence entre le vecteur obtenu à l'itération et son image par la fonction G.

Pour chacune des méthodes employées ci-après le nombre d'évaluations de fonction non linéaire est de $d_k + 1$ par itération, soit en pratique une fois de plus que la dimension du système.

4.3.1 Exemple 1

Nous reprenons l'exemple 1 employé dans les précédents chapitres et tiré de [3]. Soit à trouver la solution unique $\begin{pmatrix} -1 \\ +1 \end{pmatrix}$ du système de dimension 2 , $G(x) = x$, où G est définie par

$$G_1(x) = -\frac{x_{(2)}^4}{4} - \frac{3}{4}$$
$$G_2(x) = -0.405e^{1-x_{(1)}^2} + 1.405$$

où nous notons $x_{(1)}$ et $x_{(2)}$ les coordonnées du vecteur de dimension 2, x, et $G(x) = \begin{pmatrix} G_1(x) \\ G_2(x) \end{pmatrix}$.

Nous prenons comme vecteur initial le vecteur $u_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Pour la méthode NMMPE nous choisissons les deux vecteurs z_0 et z_1 suivant $z_0 = \begin{pmatrix} -1 \\ +1 \end{pmatrix}$, et $z_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

it.	NMMPE	NMPE
1	$1.608759730828518E-001$	$1.608759730828518E-001$
2	$4.444171829838128E-002$	$4.444171829838128E-002$
3	$3.446942074448955E-003$	$3.446942074448844E-003$
4	$1.344094608413116E-005$	$1.344094608701774E-005$
5	$5.456946006177077E-011$	$5.456968210637569E-011$
6	$1.110223024625157E-015$	$8.881784197001252E-016$
7	$4.440892098500626E-016$	

it.	NRRE
1	$1.608759730828518E-001$
2	$4.444171829838128E-002$
3	$3.446942074449066E-003$
4	$1.344094608368707E-005$
5	$5.457234664163479E-011$
6	$2.775557561562891E-015$
7	$2.220446049250313E-016$

4.3.2 Exemple 2

Cet exemple se trouve également dans [3].

Soit à trouver le point fixe $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ du système de dimension 2, $G(x) = x$ suivant

$$G_1(x) = \frac{x_{(2)}^2}{2} + x_{(1)} - \frac{1}{2}$$
$$G_2(x) = sin(x_{(1)}) + sin(x_{(2)} - 1) + 1.$$

Nous prenons comme vecteur initial $x_0 = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$.

Nous prenons en premier vecteur arbitraire $z_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, et nous prenons en second vecteur arbitraire $z_1 = \begin{pmatrix} -1 \\ +1 \end{pmatrix}$.

Nous obtenons les tableaux de résultats suivants

it.	NMMPE	NMPE
1	$2.9808720124030 20E-001$	$2.980872012403022E-001$
2	$1.089737539816198E-001$	$1.089737539816200E-001$
3	$5.666530999026698E-005$	$5.666530999048902E-005$
4	$3.865636699629249E-009$	$3.865636699629249E-009$
5	$3.198396404541095E-016$	$1.038665672980163E-016$

it.	NRRE
1	$2.980872012403020E-001$
2	$1.089737539816197E-001$
3	$5.666530999026698E-005$
4	$3.865636921673854E-009$
5	$7.876728893958288E-017$

4.3.3 Exemple 3

Cet exemple est donné dans [41].
Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 2.25 & 0.01 & 0.05 & 0.50 \\ 0.01 & 1.75 & 0.00 & 0.05 \\ 0.05 & 0.00 & 1.75 & 0.01 \\ 0.50 & 0.05 & 0.01 & 2.25 \end{pmatrix}$$

$$b = (-0.81, -0.31, -0.31, -0.81)^T$$

et

$$Q(x) = -0.5(x_{(1)}^2 + x_{(1)}x_{(4)}, x_{(2)}^2, x_{(3)}^2, x_{(1)}x_{(4)} + x_{(4)}^2)^T.$$

La solution de ce système est $(1, 1, 1, 1)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (1.2, 1.2, 1.2, 1.2)^T$.
Pour les vecteurs arbitraires z_i de la méthode NMMPE nous prenons $z_0 = (-1, 1, 2, 1)^T$, $z_1 = (1, 2, 3, 4)^T$, $z_2 = (5, 1, 7, 2)^T$, $z_3 = (1, 1, 1, 1)^T$.
Nous obtenons alors les tableaux de résultats suivants:

it.	NMMPE	NMPE
1	$1.554743249986390E-002$	$8.700519344504665E-003$
2	$6.952292006268124E-005$	$5.085227009438142E-005$
3	$8.504433823830482E-010$	$7.815510461028907E-010$
4	$5.551115123125783E-015$	$3.974598428158060E-014$

it.	NRRE
1	$9.761552021656739E-003$
2	$7.514061280335937E-004$
3	$1.814986272385966E-008$
4	$2.664535259100376E-015$

4.3.4 Exemple 4

Nous voulons résoudre le système de dimension 4, $G(x) = x$ avec

$$G(x) = b + Ax + Q(x)$$

$$A = \begin{pmatrix} 3.9 & -3.7 & 2.4 & -0.6 \\ 2.4 & -2.0 & 2.2 & -0.6 \\ 2.4 & -3.6 & 4.1 & -0.9 \\ 2.8 & -5.2 & 4.8 & -0.4 \end{pmatrix}$$

$$b = (-0.75, -0.75, -0.75, -0.75)^T$$

et

$$Q(x) = -0.25(x_{(1)}^2, x_{(2)}^2, x_{(3)}^2, x_{(4)}^2)^T.$$

La solution de ce système est $(3, 3, 3, 3)^T$.
Nous prenons comme vecteur initial le vecteur $x_0 = (2.5, 2.5, 2.5, 2.5)^T$.
Nous prenons de plus $z_0 = (-1, 1, 2, 1)^T$, $z_1 = (-3, 2, 0, 1)^T$, $z_2 = (5, 1, 7, 2)$, et $z_3 = (1, 1, 3, 1)^T$. Nous obtenons alors

it.	NMMPE	NMPE
1	$2.482829769951245E-002$	$9.698156897347490E-001$
2	$2.786908599450300E-006$	$5.446502334965242E-003$
3	$2.864375403532904E-013$	$2.421131479346528E-005$
4		$1.093880541702674E-011$
5		$1.421085471520200E-014$

it.	NRRE
1	$4.110288094002001E-002$
2	$3.079195948441082E-005$
3	$2.442224200649434E-011$
4	$6.217248937900877E-015$

Nous pouvons noter dans cet exemple l'obtention d'un chiffre exact supplémentaire pour le NRRE par rapport à toutes les autres méthodes utilisées dans ce travail. Par contre les trois méthodes ci-dessus convergent moins vite, sur cet exemple, que les méthodes iNTEA, iNCGS, et NCGM.

4.3.5 Exemple 5

Nous voulons résoudre le système de dimension 6, $G(x) = x$ avec G définie par

$$\begin{aligned}
G_1(x) &= -0.75 - \frac{x_{(2)}^2 x_{(4)} x_{(6)}}{4} \\
G_2(x) &= -0.405 e^{1+x_{(1)}x_{(2)}} + 1.405 \\
G_3(x) &= \frac{x_{(4)}x_{(6)}}{2} - 1.5 \\
G_4(x) &= 0.605 e^{1-x_{(3)}^2} + 0.395 \\
G_5(x) &= \frac{x_{(2)}x_{(6)}}{2} - 1.5 \\
G_6(x) &= x_{(1)}x_{(5)}.
\end{aligned}$$

La solution de ce système est $(-1, 1, -1, 1, -1, 1)^T$.
Nous prenons comme vecteur initial $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T$.
Comme vecteurs arbitraires z_i à utiliser pour la méthode NMMPE nous prenons tout d'abord les suivants

$$z_0 = (-1, 1, 2, 3, 1, 2)^T, z_1 = (1, 2, 3, 0, 4, 1)^T,$$

$$z_2 = (-5, 1, 5, 2, 5, 3)^T, z_3 = (-1, 1, -1, 1, -1, 1)^T,$$

$$z_4 = (-1, -2, -3, 1, 2, 3)^T, z_5 = (-10, 5, 4, 7, 8, 1)^T,$$

puis nous prenons $z_i = \epsilon_i$ où ϵ_i est le $i^{\text{ème}}$ vecteur de la base canonique associée à $\mathbb{R}^6$.

it.	NMMPE avec z_i aléatoires	NMMPE avec $z_i = \epsilon_i$
1	$9.2618583580338 14E-002$	$9.261858358035890E-002$
2	$3.205739059967239E-003$	$3.205739059965129E-003$
3	$1.427194070813265E-006$	$1.427194066261350E-006$
4	$1.725841691779806E-012$	$1.731725873810319E-012$
5	$3.774758283725532E-015$	$5.884182030513330E-015$
6	$2.220446049250313E-016$	$1.110223024625157E-015$

it.	NRRE	NMPE
1	$9.261858358033848E-002$	$9.261858358033770E-002$
2	$3.205739059967128E-003$	$3.205739059957469E-003$
3	$1.427194117886721E-006$	$1.427194065151127E-006$
4	$1.727507026316744E-012$	$1.734834498279270E-012$
5	$1.465494392505207E-014$	$1.776356839400250E-015$
9	$8.881784197001252E-016$	$8.881784197001252E-016$

Nous pouvons noter dans cet exemple la bonne marche du NMMPE avec les vecteurs z_i choisis dans la base canonique de $\mathbb{R}^6$ ce qui est d'autant plus appréciable que ce choix occasionne une réduction du nombre de produits scalaires à effectuer dans l'algorithme.

4.3.6 Exemple 6

Cet exemple est donné dans [20].
Nous considérons le système non linéaire de dimension 10,
$G(x) = x$ avec G définie comme suit

$$G(x) = x - hF(x)$$

où

$$\begin{aligned}
&F_1(x) = (3 - 5x_{(1)})x_{(1)} + 1 - 2x_{(2)} \\
&F_i(x) = (3 - 5x_{(i)})x_{(i)} + 1 - x_{(i-1)} - 2x_{(i+1)} \text{ pour } i = 2, \ldots, 9 \\
&F_{10}(x) = (3 - 5x_{(10)})x_{(10)} + 1 - x_{(9)}.
\end{aligned}$$

Nous prenons comme vecteur initial le vecteur

$$x_0 = (-1.,-1.,-1.,-1.,-1.,-1.,-1.,-1.,-1.,-1)^T.$$

Pour les vecteurs arbitraires z_i de la méthode NMMPE nous prenons

- au hasard les dix vecteurs suivants

$$z_0 = (-1.1, 2.3, 1.2, 0.1, 4.2)^T,$$

$$z_1 = (1.2, 3.0, 4.1, 2.3, 4.5)^T,$$

$$z_2 = (-5.1, 5.2, 3.5, 5.-1, 6.5)^T,$$

$$z_3 = (-1, 1, -1, 1, -1, 1, -1, 1, -1, 1)^T,$$
$$z_4 = (-1, -2, -3, 1, 2, 3, -1, -2, -3, 1)^T,$$
$$z_5 = (-10, 2, 4, 7, 8, 1, 9, 3, 7, 2)^T,$$
$$z_6 = (0, 1, -2, 4, 7, 9, 3, 2, 8, 1)^T,$$
$$z_7 = (4, 3, 0, 1, 2, 7, -1, -4, -2, 9)^T,$$
$$z_8 = (-100, 0, 1, 2, 200, 4, 7, -1, 99, 1)^T,$$
$$z_9 = (0.1, 0.2, -9, 100, -0.05, 1, 4, -200, 1, 2)^T.$$

- nous remplaçons uniquement les 2 derniers vecteurs comme suit

  $z_8 = (-1, -2, 1, 2, 3, 7, 1, 8, 4, 2)^T$ et $z_9 = (1, 2, 4, 7, 9, -1, 2, 8, 7, -4)^T$.

- en troisième lieu il parait intéressant de voir les résultats obtenus par la méthode NMMPE en faisant un tirage aléatoire des coordonnées des vecteurs z_i. Pour ce faire nous utilisons la commande Random[Real,{-10000,10000}] en Mathematica qui choisit un nombre réel compris entre -10000 et 10000.

- enfin nous prenons pour vecteurs z_i les vecteurs e_i de la base canonique associée à $\mathbb{R}^{10}$.

Dans cet exemple la solution n'est pas connue, aussi à chaque itération, nous afficherons la différence $(G(x_k) - x_k)$.

Nous prenons $h = 0.2$ pour les tableaux de résultats suivants :

it.	NMMPE avec le choix 1	NMMPE avec le choix 2
1	$1.860040916901173E-006$	$1.713158479902943E-004$
2	$4.996003610813204E-016$	$6.304956556846264E-013$
3		$1.110223024625157E-016$

it.	NMMPE avec le choix 3	NMMPE avec $z_i = e_i$
1	$1.713158479916821E-004$	$1.713158479915156E-004$
2	$6.305511668358577E-013$	$6.304956556846264E-013$
3	$1.665334536937735E-016$	$4.996003610813204E-016$

it.	NRRE	NMPE
1	$1.713158479915711E-004$	$1.713158479914600E-004$
2	$6.305511668358577E-013$	$6.304956556846264E-013$
3	$5.551115123125783E-017$	$5.551115123125783E-017$

Nous pouvons noter dans cet exemple le très bon résultat obtenu grâce à la méthode NMMPE avec les vecteurs z_i choisis 'à la main' avec une différence $(G(x_k) - x_k)$ en $E - 016$ dès la deuxième itération ; ceci est le meilleur résultat obtenu sur cet exemple parmis toutes les méthodes utilisées et ce malgré le choix aléatoire de dix vecteurs à dix composantes. En choisissant 2 derniers vecteurs autres que ceux choisis précédemment il faut attendre la troisième itération pour avoir une bonne estimation de la solution. Le NMMPE utilisé en choisissant les vecteurs z_i par Mathematica nous fournit également des résultats corrects malgré le choix aléatoire de 10 vecteurs soit de 100 coordonnées réelles. Enfin les résultats obtenus en prenant les z_i égaux aux vecteurs de la base canonique de $\mathbb{R}^{10}$ sont équivalents à ceux obtenus en moyenne par les autres méthodes avec un coût de calcul moindre.

4.3.7 Exemple 7

Nous construisons cet exemple comme le fait M. Altman dans [1]. Soient

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \quad \text{et} \quad P = \begin{pmatrix} 1 & 2 & 0 & 1 \\ 2 & -1 & 3 & -1 \\ 3 & 2 & 2 & 2 \\ 4 & -1 & 1 & -2 \end{pmatrix}.$$

Nous calculons

$$A = PDP^{-1} = \begin{pmatrix} 3/8 & -5/8 & 9/8 & -3/8 \\ 97/32 & 153/32 & -69/32 & -33/32 \\ -41/16 & -1/16 & 61/16 & -23/16 \\ 159/32 & 103/32 & -123/32 & 33/32 \end{pmatrix}.$$

Nous résolvons alors le système $Ax - \dfrac{(x, Ax)}{(x, x)}x + x = x$, qui doit, étant donnée sa construction, nous donner en solution un vecteur propre de la matrice A. Les notations dans cet exemple restent celles du chapitre I.

Pour les 3 méthodes utilisées ci-après nous prenons x_0 proche de u_3 soit $x_0 = (-0.5, 2.5, 1.5, 0.5)^T$. Pour la méthode NMMPE nous avons à choisir 4 vecteurs arbitraires y_i ; connaissant déjà 4 vecteurs : les vecteurs engendrant les espaces propres de A, nous prendrons naturellement $y_1 = u_1, y_2 = u_2, \ldots$.

it.	NMPE	NRRE
1	$1.928433730205881E-002$	$2.550554319487297E-002$
2	$2.178680596131910E-003$	$3.783848935512424E-005$
3	$3.686159377735976E-009$	$2.474547233788341E-010$
4	$2.220446049250313E-015$	0.000000000000000

it.	NMMPE
1	$1.112277212289032E-001$
2	$1.615796823741800E-004$
3	$8.639199533533315E-008$
4	$1.776356839400250E-015$

Dans chaque cas nous aboutissont à une approximation d'un vecteur propre de l'espace propre E_4. Nous soulignons encore dans cet exemple le choix des vecteurs arbitraires de la méthode NMMPE.

5 Bilan

5.1 Comparaison avec deux autres méthodes

5.1.1 Newton Lanczos

[22, 23]
Soit à résoudre le problème

$$F(x) = 0$$

où F est un opérateur non linéaire d'un espace réel Euclidien ou de Hilbert vers lui même.

Les méthodes de Newton inexactes résolvent ce système en couplant la méthode de Newton avec une méthode de résolution de systèmes linéaires. Nous rappelons que la méthode de Newton résoud le système non linéaire $F(x) = 0$ par la construction d'une suite x_n approximant x comme suit

- choisir x_0 réel,
- à chaque itération k faire $x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}$.

Les méthodes de Newton inexactes sont généralement utilisées lorsque la dimension du jacobien du système est petite, ou quand la matrice jacobienne du système est creuse. La convergence locale de ces méthodes est au moins linéaire. Pour obtenir une convergence quadratique locale il faut que F' soit de Lipschitz.
L'algorithme correspondant à ces méthodes est le suivant

- choisir un point initial x_0,
- à l'itération k
  - résoudre le système linéaire en Δ_k suivant

$$F'(x_k)\Delta_k = -F(x_k),$$

  - poser $x_{k+1} = x_k + \Delta_k$.

La solution Δ_k du système linéaire interne à chaque itération de cet algorithme peut être donnée grâce à une méthode de type Lanczos. Nous appellerons Newton Lanczos les méthodes non linéaires résultantes. En particulier dans les exemples ci-dessous nous utiliserons l'algorithme Lanczos/Orthodir pour résoudre le système linéaire $F'(x_k)\Delta_k = -F(x_k)$.

Nous pouvons remarquer que dans le cas des algorithmes présentés jusqu'ici le problème à résoudre était un problème de point fixe $G(x) = x$ tandis que pour les méthodes de Newton inexactes nous cherchons les zéros d'un opérateur non linéaire F ; pour retrouver les exemples précédents nous prendrons donc cette fois $F(x) = G(x) - x$. Donnons maintenant les résultats numériques obtenus en appliquant Newton Lanczos aux six exemples précédemment résolus par d'autres méthodes.

exemple 1

Cet exemple est de dimension 2. Pour ce premier exemple le vecteur initial x_0 est toujours $(0,0)$.

it.	Newton Inexact
1	$6.959041405259134E-001$
2	$2.341231193765203E-001$
3	$7.871846117813178E-002$
4	$1.206468843579371E-002$
5	$2.234794871972291E-004$
6	$1.732210454154920E-008$
7	$3.945411705674751E-016$
8	$1.084202172485504E-019$

Nous pouvons remarquer que la solution est approchée avec plus de précision par la méthode de Newton inexacte que par les algorithmes précédents (avec une norme finale de l'erreur en E-019 à comparer à une norme minimale de l'erreur en E-016 pour les autres algorithmes). Cependant la convergence est moins rapide par Newton Lanczos puisque nous obtenions un résultat correct dès la sixième itération par les autres algorithmes, avec par exemple $1.110223024625157E-016$ en norme infinie de l'erreur en sixième itération pour ce même exemple par le iNCGS avec $i = 1$.

exemple 2

Nous devons ici prendre un vecteur initial x_0 plus proche de la solution que le vecteur $(0.5,-1)$ utilisé dans la plus part des autres algorithmes ; la convergence de la méthode de Newton inexacte se fait sinon vers une autre solution que la solution $(0,1)$ souhaitée comme ce fut déjà le cas quand nous avions utilisé le NCG-Min sur cet exemple. Comme pour le NCG-Min nous prenons x_0 de coordonnées $(0.5,0.5)$. Alors nous obtenons le tableau de résultats suivant

it.	Newton Inexact
1	$2.500000000000003E-001$
2	$2.500000000000011E-002$
3	$3.048780487804915E-004$
4	$4.646114733013965E-008$
5	$1.079323262709320E-015$

En comparaison avec le NCG-Min pour lequel les conditions de départ sont identiques l'algorithme de Newton Lanczos est sur cet exemple plus rapide puisque, dès la cinquième itération nous obtenons ici une norme de l'erreur en E-015, tandis qu'elle était en E-009 à la même itération du NCG-Min. Par contre à la sixième itération nous avons un breakdown pour la méthode Newton inexacte tandis que le NCG-Min approche ensuite la solution avec 2 chiffres exacts supplémentaires par rapport à la meilleure approximation de Newton Inexact. De plus certains autres algorithmes tels que NCG-Ort, NTEA2, iNTEA avec $i = 1$, et NRRE donnaient une convergence plus rapide vers une solution approchée à E-017 près.

Dans les exemples qui suivent les conditions initiales sont identiques à celles utilisées pour les autres algorithmes.

exemple 3

Nous étudions ici un exemple de dimension 4.

it.	Newton Inexact
1	$6.615898231683273E-002$
2	$1.266100109641734E-002$
3	$6.364848308720965E-004$
4	$1.711515941653121E-006$
5	$1.302187697949098E-011$
6	$2.082752373344654E-016$

Pour cet exemple numéro 3 l'algorithme de Newton inexact est moins performant que l'ensemble des autres algorithmes utilisés jusqu'alors; ceux-ci donnaient certes une approximation de la solution avec 14 ou 15 chiffres exacts après la virgule et non 16 comme Newton Inexact, mais cette appro-

ximation était atteinte dès la troisième itération avec donc en général une vitesse de convergence plus rapide que pour Newton Inexact.

exemple 4

Le vecteur initial x_0 est dans cet exemple de coordonnées égales à $(2.5, 2.5, 2.5, 2.5)$, la solution du système étant le vecteur de $(3, 3, 3, 3)^T$.

it.	Newton Inexact
1	$2.499999999999116E-001$
2	$2.500000000001315E-002$
3	$3.048780487223297E-004$
4	$4.646136930719878E-008$
5	$1.343716804491635E-014$

La méthode Newton Lanczos donne ici, comme le NMMPE, le NMPE, et le NRRE, de moins bons résultats que les méthodes des deux premiers chapitres appliquées à ce même exemple ; en effet les méthodes iNTEA, iNCGS, iNCG-Ort, NCG-Adj, et NCG-Min donnaient une norme infinie de l'erreur en E-014 dès la seconde itération.

exemple 5

Si nous prenons le vecteur initial $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T$ utilisé pour la plupart des autres méthodes nous n'obtenons par Newton Inexact une bonne approximation de la solution qu'à l'itération 17. Nous allons donc prendre un vecteur initial plus proche de la solution; nous prenons pour le tableau de résultats ci-dessous le vecteur

$$x_0 = (-0.7, 0.7, -0.7, 0.7, -0.7, 0.7)^T$$

comme nous l'avions pris pour la méthode NCG-Adj.

it.	Newton Inexact
1	$2.679181732638075E-001$
2	$7.171679809518172E-002$
3	$7.401326510475759E-003$
4	$8.527804094327074E-008$
5	$2.220446049250313E-016$

Par rapport à NCG-Adj qui a été mis en oeuvre avec les mêmes conditions initiales, Newton Lanczos donne un résultat nettement meilleur : les autres méthodes utilisées précédemment donnaient elles un résultat moins bon quoique correct avec un vecteur initial plus aléatoire.

exemple 6

Nous rappelons que dans cet exemple la solution du système n'est pas connue, aussi nous affichons à chaque itération k la norme infinie de $F(x_k)$.

it.	Newton Inexact
1	$2.434700050380645E-001$
2	$2.878254029453869E-002$
3	$7.100310032827216E-004$
4	$5.689247790547378E-007$
5	$3.711035385239869E-013$
6	$4.510281037539698E-017$

La méthode de Newton inexacte nous fait ici gagner 1 à 3 chiffres exacts en fin d'algorithme par rapport aux méthodes utilisées dans les deux premiers chapitres sur cet exemple. Par contre il faut attendre la sixième itération pour avoir une bonne approximation de la solution. Les méthodes non linéaires issues de l'étude des polynômes biorthogonaux vues dans ce troisième chapitre restent les plus performantes sur cet exemple.

conclusion

Nous pouvons tout d'abord remarquer que nous avons utilisé ici l'algorithme Lanczos/Orthodir pour résoudre le système linéaire interne au problème non linéaire étudié. Nous aurions eu théoriquement les mêmes résultats en

utilisant Lanczos/Orthores, Lanczos/Orthomin, ou tout autre méthode de type Lanczos, ces méthodes calculant la solution exacte d'un système linéaire de dimension p en un nombre fini de pas inférieur à p [4, 34]. Cependant nous pouvons noter qu'avec les erreurs d'arrondis dues à l'arithmétique de l'ordinateur les résultats obtenus en pratique seraient différents.

Quant à la comparaison faite entre la méthode Newton Lanczos et les précédentes nous pouvons dire que, sur la plupart des exemples, la vitesse de convergence des algorithmes traités antérieurement parait plus rapide, le nombre d'itérations pour parvenir à la solution étant généralement inférieur. Par contre la précision finale semble meilleure pour les exemples traités par la méthode de Newton inexacte, les calculs ayant été également faits en double précision soit avec 16 chiffres significatifs pour l'algorithme Newton Inexact.

Dans certains exemples les conditions initiales ont dû être ajustées pour la méthode de Newton inexacte, en particulier le vecteur initial x_0 a dû être pris plus proche de la solution que pour d'autres algorithmes.

La différence importante entre les méthodes Newton Inexact et les méthodes étudiées dans ce travail est le nombre d'évaluations de fonction non linéaire par itération : nous avons une unique évaluation à faire par itération pour Newton Inexact contre $2p$ soit 2 fois la dimension du système pour les méthodes du deuxième chapitre hormis le NCG-Adj qui nécessite $2p+1$ évaluations, et contre $p+1$ évaluations à faire pour les méthodes NMMPE, NMPE, et NRRE.

Cependant les méthodes Newton Inexact obligent théoriquement au calcul de la matrice jacobienne du système soit au calcul de p^2 dérivées partielles, tandis que les autres méthodes étudiées ne nécessitent aucun calcul de dérivée.

5.1.2 Méthode d'Henrici

[18, 41]
Soit u_n une suite de vecteurs complexes à p composantes convergeant vers u. Pour accélérer la convergence d'une telle suite Henrici a proposé la transformation de cette suite en la suite h_n comme suit [29]:

$$h_n = u_n - \Delta U_n (\Delta^2 U_n)^{-1} \Delta u_n,$$

où $\Delta^i U_n$ est la matrice dont les colonnes sont $\Delta^i u_n, \ldots, \Delta^i u_{n+p-1}$
pour $i = 1, 2$. Nous pouvons remarquer que cette transformation est la

généralisation au cas vectoriel du procédé Δ^2 d'Aitken ; en effet pour $p = 1$ nous retrouvons la transformation scalaire d'Aitken.

h_n peut s'exprimer comme rapport de deux déterminants:

$$h_n = \frac{\begin{vmatrix} u_n & u_{n+1} & \cdots & u_{n+p} \\ (\Delta u_n)_1 & (\Delta u_{n+1})_1 & \cdots & (\Delta u_{n+p})_1 \\ (\Delta u_n)_2 & (\Delta u_{n+1})_2 & \cdots & (\Delta u_{n+p})_2 \\ \vdots & \vdots & & \vdots \\ (\Delta u_n)_p & (\Delta u_{n+1})_p & \cdots & (\Delta u_{n+p})_p \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ (\Delta u_n)_1 & (\Delta u_{n+1})_1 & \cdots & (\Delta u_{n+p})_1 \\ (\Delta u_n)_2 & (\Delta u_{n+1})_2 & \cdots & (\Delta u_{n+p})_2 \\ \vdots & \vdots & & \vdots \\ (\Delta u_n)_p & (\Delta u_{n+1})_p & \cdots & (\Delta u_{n+p})_p \end{vmatrix}},$$

où le déterminant du numérateur représente le vecteur obtenu en développant ce déterminant suivant sa première ligne d'après les règles habituelles et où $(\Delta u_j)_i$ désigne la ième composante du vecteur Δu_j.

Il existe différentes méthodes permettant d'implémenter cette transformation. Nous utiliserons ici le H-algorithme [7]. Le H-algorithme est un algorithme récursif permettant le calcul de quantités $H_k^{(n)}$ telles que

$$H_k^{(n)} = \frac{\begin{vmatrix} u_n & u_{n+1} & \cdots & u_{n+m} \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ g_2(n) & g_2(n+1) & \cdots & g_2(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ g_2(n) & g_2(n+1) & \cdots & g_2(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}.$$

En utilisant l'identité de Sylvester nous obtenons:

$$\begin{cases} H_0^{(n)} = u_n \text{ et } g_{0,i}^{(n)} = g_i(n) & \text{pour } n = 0,1,\ldots \text{ et } i = 1,2,\ldots \\ H_k^{(n)} = H_{k-1}^{(n)} - g_{k-1,k}^{(n)} \frac{\Delta H_{k-1}^{(n)}}{\Delta g_{k-1,k}^{(n)}} & \text{pour } n = 0,1,\ldots \text{ et } k = 1,2,\ldots \\ g_{k,i}^{(n)} = g_{k-1,i}^{(n)} - g_{k-1,k}^{(n)} \frac{\Delta g_{k-1,i}^{(n)}}{\Delta g_{k-1,k}^{(n)}} & \text{pour } n = 0,1,\ldots \quad k = 1,2,\ldots \text{ et } i > k. \end{cases}$$

où l'opérateur Δ agit sur l'indice supérieur n. Dans le cas où $g_i(n) = (\Delta u_n)_i$ pour $i = 1,\ldots,p$ nous avons

$$h_n = H_p^{(n)},$$

ce qui nous permet d'implémenter la transformation d'Henrici. L'une des applications de la méthode d'Henrici est la résolution de systèmes de point fixe du type $G(x) = x$. L'algorithme non linéaire issu de la transformation d'Henrici que nous allons utiliser s'écrit comme suit:

- choisir un point initial x_0,
- à l'itération k
  - poser $u_0 = x_k$,
  - calculer $u_j = G(u_{j-1})$ pour $j = 1,\ldots,p+1$,
  - calculer $x_{k+1}=h_0 = H_p^{(0)}$.

Il a été prouvé que cette méthode d'Henrici non linéaire est d'ordre au moins 2, sous l'hypothèse **H** utilisée dans les théorèmes ci-dessus [38].

Appliquons maintenant celle-ci aux exemples de résolution de point fixe précédents. nous pouvons remarquer que dans ces exemples nous avons pris en pratique $d_k = p$; or pour ce choix de d_k les méthodes NMPE, NRRE, et NMMPE ne sont autres que la méthode d'Henrici pour résoudre les systèmes non linéaires ; celle-ci étant programmée différemment il est cependant intéressant de voir ici quel algorithme semble le plus stable.

Sauf en cas de précision, les vecteurs initiaux des différents exemples sont ceux utilisés jusqu'à présent pour la plupart des méthodes.

exemple 1

it.	Henrici non linéaire
1	$1.608759730828515E-001$
2	$4.444171829838095E-002$
3	$3.446942074449066E-003$
4	$1.344094608524138E-005$
5	$5.457168050782002E-011$
6	$2.220445049250313E-016$

Dans cet exemple la méthode d'Henrici donne des résultats similaires à ceux obtenus jusqu'ici.

exemple 2

it.	Henrici non linéaire
1	$2.980872012403022E-001$
2	$1.089737539816200E-001$
3	$5.666530999048902E-005$
4	$3.865636921673854E-009$
5	$5.296327612591689E-017$

Ce tableau de résultats se classe parmis les meilleurs obtenus pour cet exemple par les méthodes testées à savoir les tableaux correspondant au iNTEA avec $i = 1$, au NTEA2, au NCG-Ort, et au NRRE.

exemple 3

Dans cet exemple si nous gardons le vecteur initial $x_0 = (1.2, 1.2, 1.2, 1.2)^T$ nous avons un dépassement de capacité dès la première itération de l'algorithme d'Henrici non linéaire. Pour que cet exemple marche par la méthode d'Henrici il faut prendre un vecteur initial plus proche de la solution. Nous prenons donc $x_0 = (1.2, 0.8, 1.2, 0.8)^T$. Nous obtenons alors le tableau de résultats suivant:

it.	Henrici non linéaire
1	$5.533758435115832E-003$
2	$4.457969744020573E-006$
3	$1.804112415015879E-012$
4	$1.154631945610163E-014$

Après changement du vecteur initial les résultats obtenus sont corrects. Ils approchent ceux obtenus par le NCG-Min.

exemple 4

Si nous conservons le vecteur initial $x_0 = (2.5, 2.5, 2.5, 2.5)^T$ nous n'obtenons qu'une première itération par l'algorithme d'Henrici ; la norme infinie de l'erreur à cette première itération est $2.482948545406893E-002$, puis il y a arrêt de l'algorithme. Par contre si nous prenons un vecteur initial plus proche de la solution soit $x_0 = (2.8, 2.9, 2.8, 2.9)^T$ l'algorithme tourne comme suit:

it.	Henrici non linéaire
1	$1.557654082161442E-002$
2	$3.423508304400968E-006$
3	$1.834088436680759E-013$

Malgré le rapprochement du vecteur initial les résultats sont moins bons que ceux donnés par les autres méthodes.

exemple 5

it.	Henrici non linéaire
1	$9.261858358035868E-002$
2	$3.205739059964463E-003$
3	$1.427194071701443E-006$
4	$1.728506227038906E-012$
5	$8.881784197001252E-015$

Les résultats sont ici dans la moyenne de ceux obtenus jusqu'alors.

exemple 6

it.	Henrici non linéaire
1	$1.713159046402568E-004$
2	$6.304956556846264E-013$
3	$1.110223024625157E-016$

Pour ce dernier exemple les méthodes étudiées dans le troisième chapitre soient le NMMPE, le NMPE, et le NRRE restent les meilleures.

conclusion

En conclusion de ces résultats numériques nous pouvons dire que sur certains exemples la méthode d'Henrici non linéaire est équivalente aux méthodes utilisées précédemment ; d'ailleurs quand nous prenons $d_k = p$ la méthode d'Henrici donne théoriquement le même vecteur résultat que les méthodes NMPE, NRRE, et NMMPE. Les algorithmes étant programmés différemment il est cependant intéressant de voir quel algorithme parait le plus stable. A ce sujet les exemples numéro 3 et 4 semblent montrer une moins bonne stabilité numérique du dernier algorithme par rapport aux précédents. la méthode d'Henrici non linéaire est d'ordre 2 comme les méthodes iNTEA, iNCGS, NCGM, NMMPE, NMPE, et NRRE ; cette méthode ne nécessite elle non plus aucune inversion de matrice et aucun calcul de dérivée ; de plus à chaque itération de l'algorithme d'Henrici non linéaire nous avons besoin du calcul de $p+1$ vecteurs u_j et non de d_k+1, $2d_k$ ou $2d_k+1$ comme pour les méthodes traitées dans cette thèse.

5.2 Bilan

Finalement nous pouvons dire que les méthodes étudiées dans ces trois chapitres se 'valent'. Nous ne pouvons en effet pas donner de choix d'une méthode résolvant *mieux* tous les types d'exemples, ce choix devant se faire suivant plusieurs critères.

Tout d'abord il est évident que pour des raisons de coût de calcul les méthodes iNTEA sont à préférer aux méthodes NCGM ; en effet rappelons que le calcul des vecteurs $G_i^{(0)}(d_k, 0)$, $CGMin_{d_k}$, $CGAdj_{d_k}$, et $CGOrt_{d_k}^i$ nécessite à chaque itération des méthodes NCGM, en plus du calcul mê-

me de ces quantités, le calcul d'au moins 2 vecteurs $\varepsilon_{i,2d_k}^{(0)}$, vecteur résultat de chaque itération d'une méthode iNTEA.
Par contre les méthodes NCGM donnent en somme une moyenne des méthodes iNTEA ce qui peut être un avantage quant aux résultats.

Au point de vue du nombre d'évaluations de fonction non linéaire les méthodes NMPE, NRRE, et NMMPE sont nettement moins coûteuses puisqu'elles ne nécessitent que $d_k + 1$ évaluations de fonction non linéaire par itération contre $2d_k + 1$ pour le NCG-Adj et $2d_k$ pour les autres méthodes NCGM et pour les méthodes iNTEA. Rappelons que ce nombre de calcul de vecteurs $u_{j+1} = G(u_j)$ par itération peut être réduit pour chaque méthode en étudiant les méthodes incomplètes correspondantes comme nous l'avons fait dans le chapitre 2.

Nous pouvons noter par ailleurs le nombre différent de vecteurs arbitraires à choisir suivant les méthodes ; aucun n'est nécessité pour le NMPE et le NRRE ; par contre un vecteur aléatoire intervient dans les méthodes iNTEA, NCG-Min, et NCG-Adj ; un second vecteur arbitraire est utilisé pour les méthodes iNCG-Ort ; la méthode ayant besoin du plus grand nombre de vecteurs arbitraires est le NMMPE avec d_k vecteurs à choisir en début de programme.

Les théorèmes de convergence des algorithmes émis dans ces trois chapitres donnaient tous sous certaines conditions une convergence quadratique locale des différents algorithmes à partir d'un certain rang. Nous pouvons cependant donner un *plus* quant aux résultats numériques obtenus par les méthodes iNCG-Ort et un *moins* pour la méthode NCG-Adj. Cette dernière approche en effet généralement moins la solution que les autres méthodes et nous pouvons aussi rappeler que pour avoir la convergence de cet algorithme nous avons dû, dans l'exemple 5, choisir le vecteur initial x_0 dans un voisinage plus restreint autour de la solution que dans les autres cas. Nous pouvons également souligner la bonne marche du NMMPE dans l'exemple 6 et ce malgré dix vecteurs à choisir en début d'algorithme.

Chapitre IV
Méthodes hybrides non linéaires

1 Théorie sur les méthodes hybrides

Soit $G : D \subset \mathbb{C}^p \to \mathbb{C}^p$ où D est un ouvert convexe de $\mathbb{C}^p$. Nous voulons résoudre le problème de point fixe $G(x) = x$. Nous supposons connues deux suites de vecteurs x'_k et x''_k convergeant vers une même solution x^* de ce système. Nous construisons alors une nouvelle suite $(y_k)_{k\in\mathbb{N}}$ comme suit :

$$y_k = \alpha_k x'_k + (1 - \alpha_k)x''_k.$$

Nous appelons fonction résiduelle toute application r de $\mathbb{C}^p$ dans $\mathbb{C}^p$ telle que

$$r(y) = 0 \Longleftrightarrow y = x^*.$$

Soient r' et r'' deux fonctions résiduelles. Nous posons $r'_k = r'(x'_k)$ et $r''_k = r''(x''_k)$ et nous choisissons alors le paramètre α_k tel que (ρ_k, ρ_k) soit minimal où

$$\rho_k = \alpha_k r'_k + (1 - \alpha_k)r''_k.$$

Nous obtenons alors

$$\alpha_k = \frac{(r''_k, r''_k - r'_k)}{\|r''_k - r'_k\|^2}.$$

La procédure ainsi obtenue est appelée procédure hybride non linéaire ou NLHP. Par construction même de α_k nous avons de plus l'inégalité

$$\|\rho_k\| \leq \min(\|r'_k\|, \|r''_k\|)$$

Une NLHP accélère sous certaines conditions les procédures construisant les suites (x'_k) et (x''_k). En particulier nous avons dans [12] le théorème suivant

Théorème 1 *Nous notons θ_n l'angle entre r''_n et $r''_n - r'_n$. Si $\exists\theta \neq \pi/2$ tel que $lim_{n\to\infty}\|\rho_n\|/\|r'_n\| = |\sin\theta| < 1$, alors $lim_{n\to\infty}\|\rho_n\|/\|r'_n\| = 0$ si et seulement si la suite (θ_n) tend vers 0 ou π quand n tend vers l'infini.*

Dans ce qui suit nous prendons

$$r'_k = x'_{k+1} - x'_k \quad \text{et} \quad r''_k = x''_{k+1} - x''_k.$$

En particulier nous prendrons pour les algorithmes appelés par la suite 'cas 6' et 'cas 9' : $(x_k)_{k\in\mathbb{N}}$ étant une suite de vecteurs convergeant vers x^*,

$x'_k = x_k$
$x'_{k+1} = x''_k = G(x_k)$
$x''_{k+1} = G(x''_k) = G(G(x_k))$
et dans ce cas nous construisons la suite hybride

$$y_k = \alpha_k x_k + (1 - \alpha_k) G(x_k)$$

avec $\alpha_k = \dfrac{(G(G(x_k)) - 2G(x_k) + x_k, G(G(x_k)) - G(x_k))}{\|G(G(x_k)) - 2G(x_k) + x_k\|^2}$;

ce coefficient est exactement le coefficient trouvé par Lemaréchal dans [36].

2 Différentes stratégies

Plusieurs stratégies peuvent être utilisées, comme dans [15] pour le cas linéaire, pour mettre en place la NLHP construisant $y_k = \alpha_k x'_k + (1 - \alpha_k) x''_k$

- cas 1
  Nous construisons deux suites (x'_k) et (x''_k) par deux méthodes différentes puis nous affichons la suite hybride (y_k).

- cas 2
  (x'_k) est construite par une méthode et $x''_k = x'_{k-1}$. La suite hybride (y_k) est donc telle que

  $y_0 = x'_0$ soit $\alpha_0 = 1$ et $y_k = \alpha_k x'_k + (1 - \alpha_k) x'_{k-1}$ pour $k \geq 1$.

- cas 3
  (x'_k) est obtenue par une méthode de résolution non linéaire et nous prenons $x''_k = y_{k-1}$. Nous initialisons α_0 à 1.

- cas 4
  (x'_k) est toujours calculée grâce à une méthode arbitraire mais cette fois en utilisant le vecteur hybride y_{k-1} comme nouveau point de départ à chaque itération k. Nous prenons de plus $x''_k = y_{k-1}$ en initialisant α_0 à 1 ce qui donne $y_0 = x'_0$.

- cas 5
  (x'_k) est construite par une première méthode. (x''_k) est calculée par une autre méthode en utilisant pour celle-ci y_{k-1} comme point de départ de l'itération k.

- cas 6
  Le calcul de la première suite (x'_k) par une méthode arbitraire sert de plus à construire la seconde suite (x''_k). En particulier nous prendrons ici $x''_k = G(x'_k)$ et nous utiliserons le coefficient α_k de Lemaréchal.

- cas 7
  (x'_k) et (x''_k) sont construites grâce à la même méthode mais avec des points de départ différents ou avec des vecteurs arbitraires à donner pour certaines méthodes (iNTEA, iNCGS, ...) différents.

- cas 8
  Nous avons deux méthodes qui donnent les suites (x'_k) et (x''_k) mais itérées à partir du vecteur hybride $y_{k-1} = \alpha_{k-1}x'_{k-1} + (1 - \alpha_{k-1})x''_{k-1}$ pour $k \geq 1$.

- cas 9
  Nous avons une méthode qui construit (x'_k) avec le vecteur hybride y_{k-1} en point de départ de chaque itération k et nous prenons $x''_k = G(x'_k)$. Ce cas est bien différent du cas 6 où nous n'avons pas utilisé le vecteur hybride pour itérer la méthode construisant (x'_k).

Appliquons maintenant ces différents cas aux exemples résolus précédemment par les méthodes NCGM. Pour chacun des exemples présentés nous choisissons en méthode de base la (ou les) méthode(s) NCGM qui avait le moins bien fonctionné sur l'exemple avec l'espoir d'améliorer les résultats par la méthode hybride. Nous rappelons à chaque fois le nombre de chiffres exacts obtenu par les méthodes NCGM choisies puis nous donnons le tableau des nombres obtenus en utilisant la suite $y_k = \alpha_k x'_k + (1-\alpha_k)x''_k$ comme décrit auparavant.

3 Exemples numériques

Exemple 1

it.	NCGS	cas 9 à NCGS
1	8.2E-001	8.9E-001
2	1.50	1.46
3	2.66	2.56
4	4.8	4.76
5	9.06	8.99
6	14.07	15.05

Après application de la méthode hybride ici choisie la méthode NCGS modifiée donne en le même nombre d'itérations un résultat ayant un chiffre significatif supplémentaire.

it.	NCG-Ort	cas 9 à NCG-Ort
1	8.4E-001	9.5E-001
2	1.51	1.68
3	2.56	2.98
4	4.59	5.53
5	8.65	10.63
6	14.54	17
7	15.35	

L'amélioration de l'algorithme NCG-Ort en utilisant le cas 9 des méthodes hybrides est ici nette. Nous obtenons en effet 17 chiffres significatifs grâce à la méthode hybride soit le maximum vu le type de déclaration utilisé (double precision) dans les programmes.

Exemple 2

it.	NCGS	NCGS modifiée par cas6.for
1	9.5E-001	1.55
2	1.79	2.57
3	3.18	4.57
4	7.15	8.54
5	11.42	16.26

La méthode hybride donne ici un résultat nettement meilleur que celui

obtenu en un même nombre d'itérations par la méthode initiale.

it.	1NCGS	cas 6 à 1NCGS	cas 9 à 1NCGS
1	1.65	1.65	1.65
2	1.01	2.90	2.78
3	1.97	4.46	6.33
4	3.19	6.96	15.35
5	5.01	10.63	16.01
6	7.39	15.35	
7	14.01	16.40	

Le cas 6 comme le cas 9 des méthodes hybrides se révèlent ici parfaitement efficaces. Rappelons que dans ces deux cas de méthodes hybrides nous construisons une première suite de vecteurs $(x_k)_{k\in\mathbb{N}}$ par une méthode, ici la méthode iNCGS avec i=1, et une seconde suite de vecteurs hybrides $(y_k)_{k\in\mathbb{N}}$ où $y_k = \alpha_k x_k + (1-\alpha_k)F(x_k)$; La différence entre les deux méthodes hybrides utilisées étant que le vecteur y_k est pris comme point de départ de l'itération $k+1$ dans le cas 9 et pas dans le cas 6. Nous pouvons noter qu'en plus d'une approximation très bonne de la solution le nombre d'itérations nécessaires dans le cas 9 est inférieur à celui de la méthode initiale. La méthode hybride due au cas 6 semble moins rapide que celle du cas 9 mais aboutit cependant à un résultat encore meilleur.

it.	NCG-Adj	cas 6 à NCG-Adj	cas 9 à NCG-Adj
1	2.1E-001	9.4E-001	9.4E-001
2	1.21	1.60	1.78
3	1.75	2.61	4.20
4	3.15	4.66	9.15
5	5.86	8.73	15.65
6	11.51	16.43	

Les cas 6 et cas 9 des méthodes hybrides appliqués à la méthode NCG-Adj sont également performants. Nous notons ici aussi un nombre d'itérations nécessaires plus grand pour le cas 6 que pour le cas 9 (quoique non supérieur au nombre utilisé pour la méthode initiale) mais pour arriver à une norme d'erreur encore plus petite.

Exemple 3

it.	NCG-Ort	cas 2 à NCG-Ort
1	3.77	3.77
2	10.73	6.88
3	14.70	14.70

Dans cet exemple la méthode hybride n'accélère pas la convergence de l'algorithme NCG-Ort qui était déjà satisfaisante. Il est cependant intéressant de noter que la convergence quadratique locale semble plus visible dans le tableau de résultats de la méthode hybride que dans celui de la méthode initiale.

it.	NMPE	cas 6 à NMPE	cas 9 à NMPE
1	2.06	3.58	3.58
2	4.29	5.13	8.83
3	9.11	9.74	15.00
4	13.40	14.26	

La méthode hybride issue du cas 6 appliquée à la méthode NMPE améliore les résultats obtenus initialement. Il faut noter que ce changement est encore plus net dans le cas 9.

it.	NCG-Adj	cas 6 à NCG-Adj	cas 9 à NCG-Adj
1	1.28	1.85	1.85
2	2.6	3.29	5.06
3	5.26	5.94	11.34
4	10.52	11.19	13.15
5	13.96	14.81	

Les méthodes 'cas 6' et 'cas 9' peuvent donc être vues comme des méthodes d'accélération de la convergence de la méthode initiale NCG-Adj sur cet exemple.

Exemple 4

it.	NMMPE	cas 3 à NMMPE	cas 9 à NMMPE
1	1.60	5.90	3.80
2	5.55	11.50	13.86
3	12.54	13.60	
4	13.82	13.82	

Le cas 9 appliqué au NMMPE donne ici une accélération de la convergence de la méthode initiale avec un meilleur résultat obtenu en moins d'itérations.

it.	NMPE	cas 6 à NMPE	cas 9 à NMPE
1	1E-002	1.12	1.12
2	2.26	5.13	12.60
3	4.61	9.83	13.89
4	10.96	13.87	
5	13.85		

Ici nous n'approchons pas la solution de plus près que par la méthode NMPE initiale mais nous avons cependant besoin d'un nombre moins important d'itérations pour y parvenir par les méthodes hybrides.

it.	cas 5 à NMMPE et NMPE
1	3.0E-001
2	3.87
3	7.90
4	13.82

Le cas 5 est appliqué ici en agissant sur deux méthodes : le NMPE et le NMMPE. Ces méthodes sont alors toutes deux améliorées par la méthode hybride.

Exemple 5

it.	NCG-Adj	cas 9 à NCG-Adj
1	7.4E-001	1.01
2	-5E-002	1.62
3	-1.0E-001	2.99
4	1.12	4.92
5	1.61	9.09
6	2.26	12.93
7	4.07	13.33
8	7.36	
9	13.07	

Nous avons pris ici un vecteur $x_0 = (-0.7, 0.7, -0.7, 0.7, -0.7, 0.7)^T$. La méthode hybride permet alors un gain de 2 itérations. Prenons maintenant un vecteur initial $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T$ pour lequel nous n'avions pas obtenu de convergence par NCG-Adj.

it.	cas9 à NCG-Adj
1	5.6E-001
2	8.9E-001
3	1.31
4	2.88
5	4.70
6	8.69
7	12.69
8	13.25

Etant donné que le vecteur hybride obtenu à chaque itération est utilisé en point de départ pour réitérer l'algorithme nous obtenons une méthode hybride convergente et ce contrairement à la méthode sur laquelle elle a agit.

it.	NCG-Ort	cas 8 à NCG-Ort et NCG-Adj
1	1.62	1.30
2	2.79	2.77
3	5.75	5.77
4	11.98	12.15
5	12.52	13.58
6	13.31	

Ici aussi le point de départ de la méthode NCG-Adj est $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T$. La méthode hybride couplant alors les méthodes NCG-Adj et NCG-Ort améliore les résultats obtenus par les deux méthodes initiales.

Exemple 6

it.	NCG-Adj	cas 6 à NCG-Adj	cas 9 à NCG-Adj
1	1.27	1.82	1.82
2	3.28	3.97	2.85
3	5.02	5.74	7.18
4	10.72	11.35	15.65
5	13.45	14.24	

La méthode NCG-Adj donne dans cet exemple de meilleurs résultats après application des deux méthodes hybrides 'cas 6' et 'cas 9'. La méthode hybride correspondant au cas 9 est dans cet exemple la meilleure.

it.	NTEA1	cas 9 à NTEA1
1	1.58	2.27
2	3.22	5.24
3	5.04	8.99
4	8.83	14.44
5	13.28	
6	13.178	
7	15.11	

Le 'cas 9' accélère nettement la convergence du NTEA1. Nous obtenons en effet un résultat très satisfaisant au bout de 4 itérations seulement.

Exemple 7

it.	NCGS	cas 9 à NCGS
1	2.61	3.21
2	1.63	5.35
3	4.70	13.37
4	9.752	14.75
5	14.45	

Dans ce dernier exemple l'utilisation du 'cas 9' nous permet d'obtenir une meilleure approche d'un vecteur propre de la matrice initiale en moins d'itérations.

Nous pouvons donc noter dans ces sept exemples une nette amélioration des résultats après usage de la méthode hybride. Cette amélioration est bien entendu moins flagrante quand nous utilisons au départ une méthode NCGM donnant déjà de très bons résultats. Nous pouvons remarquer que la mise en place de la méthode hybride par le cas 9 évoqué précédemment est de loin la meilleure ; ce cas nous donne en effet pour chaque exemple des améliorations très satisfaisantes des algorithmes initiaux ; elle permet même dans l'exemple 5 d'obtenir une convergence qui n'avait pas été obtenue sans changer le vecteur initial par l'algorithme de départ.

Notons encore qu'il est également possible de construire des NLHP à partir de plus de deux méthodes ; en effet si nous supposons connues $k+1$ suites $(x_n^{(0)})_{n\in I\!N}, \ldots, (x_n^{(k)})_{n\in I\!N}$ convergeant vers x^* et construites par différentes méthodes de recherche de point fixe alors nous pouvons construire la suite hybride comme suit :

$$y_k = x_n^{(0)} - \alpha_n \sum_{i=0}^{k} a_i^{(k,n)} x_n^{(i)}$$

où les $a_i^{(k,n)}$ sont des nombres tels que

$$\forall k, n, \quad \sum_{i=0}^{k} a_i^{(k,n)} = 0.$$

Nous avons toujours

$$\alpha_n = \frac{(r_n'', r_n'' - r_n')}{\|r_n'' - r_n'\|^2},$$

où $r''_n = r''(x_n^{(0)})$ et, comme expliqué dans [12]

$$r'_n = r'\left(\sum_{i=1}^{k} a_i^{(k,n)} x_n^{(i)}\right)$$

ou, $r^{(1)}, \ldots, r^{(k)}$ étant des fonctions résiduelles,

$$r'_n = \sum_{i=1}^{k} a_i^{(k,n)} r^{(i)}(x_n^{(i)}).$$

References

[1] M. Altman, *An iterative method for the eigenproblem of linear operators*, Bull. Pol. Acad. Sci., Math., 9 (1961) 751-755.

[2] C. Baheux, *Algorithmes d'Implémentation de la méthode de Lanczos*, Thèse, Université des Sciences et Technologies de Lille, 1994.

[3] C. Brezinski, *Accélération de la Convergence en Analyse Numérique*, LNM vol. 584, Springer-Verlag, Berlin, 1977.

[4] C. Brezinski, *The methods of Vorobyev and Lanczos*, Linear Alg. Appl., à paraître.

[5] C. Brezinski, *Padé-Type Approximation and General Orthogonal Polynomials*, ISNM vol.50, Birkhäuser, Basel, 1980.

[6] C. Brezinski, *Généralisation de la transformation de Shanks, de la table de Padé et de l'epsilon algorithme*, Calcolo, 12 (1975) 317-360.

[7] C. Brezinski, *About Henrici's method for non linear equations*, Symposium on Numerical Analysis and Computational Complex Analysis, Zurich, 15-17 (1983), non publié.

[8] C. Brezinski, *Biorthogonality and its Applications of Numerical Analysis*, Marcel Dekker, New York, 1992.

[9] C. Brezinski, Biorthogonality and conjugate gradient-type algorithms, dans *Contributions in Numerical Mathematics*, R.P. Agarwal, ed., World Scientific, Singapore, 1993, pp.55-70.

[10] C. Brezinski, *Composite sequence transformations*, Numer. Math., 46:311-321, 1985.

[11] C. Brezinski, *Numerical stability of a quadratic method for solving systems of nonlinear equations*, Computing, 14 (1975) 205-211.

[12] C. Brezinski, J.P. Chehab, *Nonlinear hybrid procedures and fixed point iterations*, ANO 354, février 1996.

[13] C. Brezinski, A.C. Matos, *Least-squares orthogonal polynomials*, J. Comput. Appl. Math., 46 (1993) 229-239.

[14] C. Brezinski, M. Redivo-Zaglia, *Look-Ahead in Bi-CGSTAB and other product-type methods for linear systems*, BIT, 35 (1995) 169-201.

[15] C. Brezinski, M. Redivo-Zaglia, *Hybrid procedures for solving linear sustems*, Numer. Math., 67 (1994) 1-19.

[16] C. Brezinski, H. Sadok, *Some vector sequence transformations with applications to systems of equations*, Appl. Numer. Math., 11(1993)443-473.

[17] C. Brezinski, H. Sadok, *Lanczos type algorithm for systems of linear equations*, Appl. Numer. Math., 11 (1993) 443-473.

[18] C. Brezinski, H. Sadok, *Vector sequence transformations and fixed point methods*, in: C. Taylor et al., eds, *Numerical Methods in Laminar and Turbulent Flows*, Pineridge, Swansea, 1987, pp.3-11.

[19] P.N. Brown, Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 n.3 (1990) 450-481.

[20] C.G. Broyden, *The convergence of an algorithm for solving sparse nonlinear systems*, Math. Comput., 25(1971) 285-294.

[21] S. Cabay and L.W. Jackson, *A polynomial extrapolation method for finding limits and antilimits of vector sequences*, SIAM J. Numer. Anal., 13 (1976) 734-752.

[22] A.T. Chronopoulos, *Nonlinear CG-like iterative methods*, J. Comp. Appl. Math., 40 (1992) pp.1-17.

[23] R.S. Dembo, S.C.Eisenstat, T.Steigaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982) 400-408.

[24] R.P. Eddy, Extrapolation to the limit of a vector sequence. In P.C.C. Wang, ed., *Information Linkage Between Applied Mathematics and Industry*, pp. 387-396, New York, 1979. Academic Press.

[25] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in *Numerical Analysis*, G.A. Watson ed., LNM vol. 506, Springer-Verlag, Berlin, 1976, pp.73-89.

[26] E. Gekeler, *On the solution of systems of equations by the epsilon algorithm of Wynn*, Math. Comput., 26 (1972) 427-436.

[27] B. Germain-Bonne, *Estimation de la limite de suites et formalisation de procédés d'accélération de la convergence*, Thèse d'état, Univ. Lille, 1978.

[28] M.H. Gutknecht, *Variants of BICGStab for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993) 1020-1033.

[29] P. Henrici, *Elements of Numerical Analysis*, Wiley, New York, 1964.

[30] K. Jbilou, *Méthodes d'Extrapolation et de Projection. Applications aux suites de vecteurs.* Thèse $3^{\text{ème}}$ cycle, Université des Sciences et Technologies de Lille, 1988.

[31] K. Jbilou, H. Sadok, *Some results about vector extrapolation methods and related fixed points iterations*, J. Comp. Appl. Math., 36 (1991) 385-398.

[32] s. Kaniel, J. Stein, *Least-square acceleration of iterative methods for linear equations*, J. Optim. Theory Appl. 14 (1974) 431-437.

[33] T. Kato, *Estimation of iterated matrices with application to the Von Newmann condition*, Numer. Math., 2 (1960) 22-29.

[34] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952) 33-53.

[35] H. Le Ferrand, *The quadratic convergence of the topological epsilon algorithm for systems of nonlinear equations*, Numerical Algorithms, 3 (1992) 273-284

[36] C. Lemaréchal, *Une méthode de résolution de certains systèmes bien posés*,(pour les hybrides) C. R. Acad. Sc. Paris,T.272, N^0 9, Mars 1971.

[37] M. Mesina, *Convergence acceleration for the iterative solution of the equations* $X = AX + f$, Comput. Methods Appl. Mech. Engrg. 10 (2) (1977) 165-173.

[38] Ortega, J. M. et Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New-York, 1970.

[39] B.P. Pugachev, *Acceleration if the convergence of iterative processes and a method of solving systems of nonlinear equations*, U.S.S.R. Comput. Math. and Math. Phys. 17 (1978) 199-207.

[40] W.C. Pye, T.A. Atchison, *An algorithm for the computation of higher order G-transformation*, SIAM J. Numer. Anal., 10 (1973) 1-7.

[41] H. Sadok, *About Henrici's transformation for accelerating vector sequences*, J. Comp. Appl. Math., 29 (1990) 101-110.

[42] A. Sidi. *Efficient implementation of minimal polynomial and reduced rank extrapolation methods*, J. Comput. Appl. Math., 36 (1991) 305-337.

[43] A. Sidi, W.F. Ford and D.A. Smith, *Acceleration of convergence of vector sequences*, SIAM J. Numer. Anal., 23 (1986) 178-196.

[44] P. Sonneveld, *CGS, A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989) 36-52.

[45] H.A. van der Vorst, Bi-CGSTAB: *A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992) 631-644.

[46] J. Van Iseghem, *Convergence of vectorial sequences. Applications*, Numer. Math., 68:549-562, 1994.

[47] Y.V. Vorobyev, *Method of Moments in Applied Mathematics Variables*, Gordon and Breach, New York, 1965.