UNIVERSITE DU BURUNDI

FACULTE DES SCIENCES

SECTION POLYTECHNIQUE

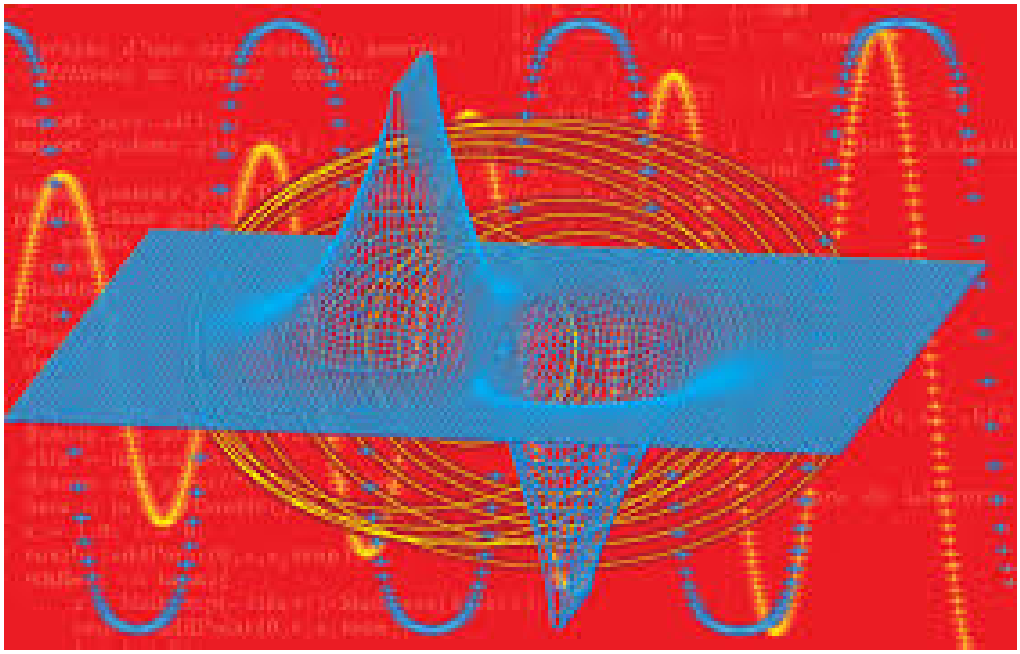
Master I maths & physique

Cours d'Analyse Numérique Avancée

(**Théorie** : 36 heures, **Travaux Dirigés** : 24 heures)

par

Prof. NYENGERI Hippolyte



Année académique 2025-2026

Table des matières

Descriptif du cours

Intitulé du cours : Analyse Numérique Avancée

Titulaire : Prof. NYENGERI Hippolyte

Objectifs du cours

Objectif général

L'objectif de ce cours est de présenter plusieurs méthodes numériques utilisées pour la résolution des systèmes linéaires et des équations non linéaires, la recherche des valeurs propres, l'intégration numérique d'équations différentielles ordinaires (EDO) et d'équations aux dérivées partielles (EDP). La présentation et l'analyse de ces méthodes sont suivies d'une implémentation et d'applications réalisées par les étudiants avec les langages de programmation **Fortran 2023**, **C++**, **Octave** et/ou **Python**.

Objectif spécifiques

A la fin de cet ECUE, les étudiants devraient être capables de :

- faire une analyse de la convergence et de la stabilité d'une méthode numérique,
- apprécier l'erreur d'une méthode numérique donnée,
- choisir une méthode appropriée à un problème donné,
- implémenter et appliquer des méthodes itératives dans la résolution des problèmes linéaires et de ceux non linéaires,
- résoudre numériquement des systèmes d'équations différentielles ordinaires, qu'ils soient « raides » ou non,
- résoudre numériques des équations aux dérivées partielles,
- calculer numériquement les valeurs et vecteurs propres d'une matrice carrée,
- mettre en oeuvre les différentes méthodes numériques présentées dans ce cours avec l'un ou l'autre des langages de programmation **Fortran 2023**, **python**, **Octave** et/ou **C++**,
- s'aider de représentations graphiques pour trouver la(les) solution(s) d'un problème.

Contenu du cours

- Introduction générale : histoire du calcul numérique, vocabulaire, exemples d'applications ;
- Résolution de systèmes linéaires par des méthodes itératives surtout : Position du problème ; résolution des systèmes tridiagonales par l'**algorithme de Thomas** ; inversion d'une matrice carrée par l'**algorithme de Faddev-Leverrier** ; les méthodes itératives (la méthode de Jacobi, la méthode de Gauss-Seidel, la méthode du gradient à pas optimal, la méthode du gradient conjugué, la méthode de la relaxation).
- Résolution des équations non linéaires : Position du problème ; localisation des zéros ; les méthodes d'encadrement comme celles de dichotomie et de la fausse position ; les méthodes des approximations successives ou méthodes du point fixe comme la méthode de Newton-Raphson, celle de la sécante et celle de Broyden ; normes matricielles et conditionnement.
- Recherche des valeurs propres : Méthode de Jacobi, méthode QR, transformation en matrice de Hessenberg, méthode de Lanczos, méthode de bissection, méthode de puissance, méthode de déflation de Wielandt, méthode de Givens, méthode de Rutishauser, méthode de Francis, calcul du polynôme caractéristique (méthode de Krylov, méthode de Leverrier, méthode de Faddev) ;
- Résolution numérique des systèmes d'équations différentielles ordinaires : Généralités, définitions, terminologies (problème de Cauchy, méthodes explicites, méthodes implicites, méthodes à un pas, méthodes multipas, convergence, consistance, stabilité, $\cdots$) ; Exemples de méthodes (Les méthodes d'Euler, la méthode de Heun, les méthodes de Runge-Kutta, les méthodes multipas de type prédicteur-correcteur, les méthodes d'Adams, les méthodes **BDF**-Backward Differentiation Formulas-), contrôle du pas.
- Introduction à l'approximation numérique des solutions d'équations aux dérivées partielles : histoire et enjeux de l'approximation des équations aux dérivées partielles ; méthode des différences finies ; méthode des éléments finis ; méthode des volumes finis ; applications de la méthode des différences finies combinée avec la méthode RK4 à l'équation d'advection-diffusion et à celle de réaction-diffusion.

Méthodologie du cours

L'enseignement magistral et des méthodes actives sont utilisés. Un syllabus servant de support de cours existe déjà. La recherche individuelle et/ou en équipe est vivement conseillée. On pourra faire recours à l'apprentissage individuel. Il est prévu des Travaux Dirigés sur ordinateur.

Mode d'évaluation

Des travaux d'évaluation continue seront donnés (40%). Un examen final (60%) en deux parties $\big[$une partie écrite (50%) et une partie pratique : programmation sur ordinateur (50%)$\big]$ sera organisé aussi bien en première session qu'en session de rattrapage.

Prérequis

Calcul Numérique et Programmation (Bac I polytechnique), Analyse Numérique et Programmation (Bac 3 maths & physique), Algorithmique I et II. Initiation à l'Informatique.

Conseils pour l'étude et l'examen

— Etudier au jour le jour et ne pas laisser la matière s'accumuler.

— Comprendre comment fonctionne la méthode utilisée pour résoudre un problème en s'aidant, quand c'est possible, d'une interprétation graphique de son mode opératoire.

— Décortiquer et exploiter pratiquement les programmes qui figurent dans les notes de cours.

— Résoudre effectivement les exercices proposés lors des séances en salle informatique et ne pas se contenter de lire les corrigés ou de faire fonctionner un programme écrit par une tierce personne.

— L'examen pratique (programmation) est à livre ouvert. Cette facilité peut se transformer en piège pour qui penserait que l'examen se réduirait à simplement appliquer des formules.

Références principales

1. Alfio Quarteroni, Riccardo Sacco and Fausto Saleri , **Méthodes Numériques. Algorithmes, analyse et applications** (Springer-Verlag Italia, Milano 2004).
2. Guillaume Legendre, **Méthodes numériques. Introduction à l'analyse numérique et au calcul scientifique**. Cours de Deuxième année de licence de Mathématiques et Informatique appliquées à l'Economie et à l'Entreprise (MI2E) à l'université de Paris-Dauphine, année académique 2020-2021.
`https://www.ceremade.dauphine.fr/~legendre/enseignement/methnum/cours_ananum_dauphine.pdf`
3. Jean-Marc Huré et Didier Pelat, **Méthodes numériques. Eléments d'un premier parcours**. Cours destiné aux étudiants de DEA Astrophysique & Méthodes associées aux université Paris 7 et 11, année académique 2002-2003
`https://media4.obspm.fr/public/M2R/supports/CoursMN.pdf`
4. Franck Jedrzejewski , **Introductions aux méthodes numériques,** 2^e **éd.** (Springer-Verlag France, Paris 2005).
5. Laurent Signac , **Introduction à l'algorithmique et à la programmation avec Python**, `https://deptinfo-ensip.univ-poitiers.fr`
6. C. Charles (2008) , **Introduction à Octave**,
`https://orbi.uliege.be/bitstream/2268/20084/1/Intro_octave.PDF`
7. Wikibooks contributors (2014) , **Octave Programming Tutorial/Text and file output**,
`https://en.wikibooks.org/w/index.php?title=Octave_Programming_Tutorial/Text_and_file_output&oldid=2721772`
8. Jean-Daniel BONJOUR (2020) , **Introduction à MATLAB et GNU Octave**
`https://enacit.epfl.ch/cours/matlab-octave/pdf/Matlab-Octave-Cours-JDBonjour-2020-09.pdf`
9. Jacques Lefrere , **Introduction au fortran 90/95/2003/2008**. Cours du Master de Sciences et Technologies (Université Pierre et Marie Curie, Paris VI, 2017),
`http://wwwens.aero.jussieu.fr/lefrere/master/mni/f90+c/polyf90.pdf`

Chapitre 0 Introduction générale

Ce document est un support au cours d'*Analyse Numérique Avancée* pour les étudiants de **Master 1** mathématique & physique. Il aborde :

- la résolution numérique de systèmes d'équations linéaires et non linéaires par des méthodes itératives surtout,
- la recherche des valeurs et vecteurs propres d'une matrice carrée,
- la résolution numérique des systèmes d'équations différentielles ordinaires,
- la résolution numérique d'équations aux dérivées partielles,

Les applications se feront avec les langages de programmation **Python**, **fortran 2023** [Jacques2017, Michael2023], **Octave** et/ou **C++**. Le traçage des courbes sera fait avec le logiciel **Octave**.

Octave n'est pas un logiciel de calcul algébrique ou symbolique, mais est un logiciel de calcul numérique, de visualisation et de programmation très performant. Ses données de base sont des matrices, pouvant évidemment se réduire à des vecteurs et des scalaires. Ceci veut dire que, pour être rapide, un programme doit privilégier les commandes matricielles plutôt que les boucles. Par exemple, si $\mathbf{A}$ est une matrice de nombres à 1000 lignes et 1000 colonnes, la commande $\mathbf{A} = \mathbf{A}/2$ divisera par deux chaque élément de la matrice $\mathbf{A}$ en 100 fois moins de temps que les deux boucles imbriquées

```
for i=1:1000 for j=1:1000 a(i,j)=a(i,j)/2;end,end
```

utilisées de façon similaire dans les autres langages de programmation.

Octave possède un langage interprété, ce qui veut dire qu'il n'y a pas besoin de compilation avant l'exécution d'un programme Octave. De plus, Octave ne demande pas de déclarations. Ces deux caractéristiques facilitent l'apprentissage d'Octave.

FORTRAN, quant à lui, est un langage de programmation développé par IBM (International Business Machines) vers 1955, et destiné à fournir aux scientifiques un moyen simple pour passer de leurs formules mathématiques jusqu'à un programme effectif (son nom est une abbréviation de **FORmula TRANslation**). Il est très efficace dans le domaine du calcul numérique, et offre de nombreuses bibliothèques de programmes (**librairies**, exemple : la librairie **LAPACK** - Linear Algebra Package) d'analyse numérique très utiles pour aussi bien les physiciens que les mathématiciens.

FORTRAN a fait l'objet de plusieurs normalisations : FORTRAN 77, FORTRAN 90 et 95, FORTRAN 2003, FORTRAN 2008, FORTRAN 2018 et plus récemment FORTRAN 2023. Tous les TP seront effectués dans un environnement **Linux** (distribution **UBUNTU**).

L'ordinateur est aujourd'hui un outil incontournable pour simuler et modéliser les systèmes, mais il faut encore savoir exprimer nos problèmes en langage formalisé des mathématiques pures. Nous sommes habitués à résoudre les problèmes de façon analytique, alors que l'ordinateur ne travaille que sur des suites de nombres. On verra dès lors qu'il existe souvent plusieurs approches pour résoudre un même problème, ce qui conduit à des algorithmes [1] différents. Un des objectifs de ce cours est de fournir des bases rigoureuses pour développer quelques algorithmes utiles dans la résolution de problèmes en physique et dans d'autres domaines.

Un algorithme, pour être utile, doit satisfaire un certain nombre de conditions. Il doit être :

1. Le mot algorithme vient du mathématicien arabe **Al-Khwarizmi** (VIIIè siècle) qui fut l'un des premiers à utiliser une séquence de calculs simples pour résoudre certaines équations quadratiques. Il est un des pionniers de l'al-jabr (algèbre).

- **rapide** : le nombre d'opérations de calcul pour arriver au résultat escompté doit être aussi réduit que possible.
- **précis** : l'algorithme doit savoir contenir les effets des erreurs qui sont inhérentes à tout calcul numérique. Ces erreurs peuvent être dues à la modélisation, à la représentation sur ordinateur ou encore à la troncature.
- **souple** : l'algorithme doit être facilement transposable à des problèmes différents.

Il importe de préciser que le principe d'un **modèle** est de remplacer un système complexe en un objet ou opérateur simple reproduisant les aspects ou comportements principaux de l'original (ex : modèle réduit, maquette, modèle mathématique ou numérique, modèle de pensée ou raisonnement).

Notons aussi que l'*aspect fini* des ordinateurs engendre nécessairement des erreurs. Considérons par exemple le cas d'un problème d'**EDO** ou d'**EDP**. La solution exacte de ce problème est une fonction continue. Les ordinateurs, quant à eux, ne connaissent que le *fini* et le *discret*. En effectuant un calcul numérique, un ordinateur ne peut retenir qu'un nombre fini de chiffres pour représenter les opérandes et les résultats des calculs intermédiaires. Les solutions approchées seront calculées comme des ensembles de valeurs discrètes sous la forme de composantes d'un vecteur solution d'un problème matriciel. La représentation des nombres dans un ordinateur introduit la notion d'**erreur d'arrondi** ou de **troncature**. Ces erreurs peuvent se cumuler sur un calcul et la solution numérique finale pourra s'avérer très éloignée de la solution exacte.

Exemple d'erreur d'arrondi : Considérons un ordinateur utilisant 4 chiffres pour représenter un nombre. Calculons la somme $1.348 + 9.999$. Le résultat exact est 11.347 et comporte 5 chiffres. Le calculateur va le représenter de manière approchée : 11.35. Il commet une **erreur d'arrondi** égale à $(11.35 - 11.347) = 0.003$.

0.1 Un exemple : le calcul de $\sqrt{x}$

Sur ordinateur, l'addition de deux entiers peut se faire de façon exacte mais non le calcul d'une racine carrée. On procède alors par approximations successives jusqu'à converger vers la solution souhaitée. Il existe pour cela divers algorithmes. Le suivant est connu depuis l'antiquité (mais ce n'est pas celui que les ordinateurs utilisent).

Soit x un nombre réel positif dont on cherche la racine carrée. Désignons par a_0 la première estimation de cette racine, et par ϵ_0 l'erreur associée :

$$\sqrt{x} = a_0 + \epsilon_0 \,. \tag{1}$$

Cherchons une approximation de ϵ_0 . Nous avons :

$$x = (a_0 + \epsilon_0)^2 = a_0^2 + 2a_0\epsilon_0 + \epsilon_0^2 \,. \tag{2}$$

Supposons que l'erreur soit petite face à a_0, ce qui permet de négliger le terme en ϵ_0^2. Alors nous avons :

$$x \approx a_0^2 + 2a_0\epsilon_0 \,. \tag{3}$$

Remplaçons l'erreur ϵ_0 par un ϵ_0' , qui en est une approximation, de telle sorte que

$$x = a_0^2 + 2a_0\epsilon_0' \,. \tag{4}$$

On en déduit que

$$\epsilon_0' = (x/a_0 - a_0)/2 \tag{5}$$

Le terme

$$a_1 = a_0 + \epsilon_0' = \frac{1}{2}\left(\frac{x}{a_0} + a_0\right) \tag{6}$$

constitue une meilleure approximation de la racine que a_0 , sous réserve que le développement soit convergent. Dans ce dernier cas, rien ne nous empêche de recommencer les calculs avec a_1 , puis a_2, etc., jusqu'à ce que la précision de la machine ne permette plus de distinguer le résultat final de la véritable solution. On peut donc définir une suite, qui à partir d'une estimation initiale a_0 devrait en principe converger vers la solution recherchée. Cette suite est :

$$a_{k+1} = \frac{1}{2}\left(\frac{x}{a_k} + a_k\right) , \qquad a_0 > 0 . \tag{7}$$

L'algorithme du calcul de la racine carrée devient donc

1. Démarrer avec une première approximation $a_0 > 0$ de $\sqrt{x}$.
2. A chaque itération k, calculer la nouvelle approximation $a_{k+1} = (x/a_k + a_k)/2$
3. Calculer l'erreur associée $\epsilon'_{k+1} = (x/a_{k+1} - a_{k+1})/2$
4. Tant que l'erreur est supérieure à un seuil fixé, recommencer en 2.

Le tableau ci-dessous illustre quelques itérations de cet algorithme pour le cas où $x = 4$.

i	a_i	ϵ'_i
0	4	-1.5
1	2.5	-0.45
2	2.05	-0.0494
3	2.00061	-0.000610
4	2.00000009	-0.000000093
etc		

TABLE 1 – Quelques itérations de l'algorithme du calcul de $\sqrt{x}$ dans le cas où $x = 4$ et $a_0 = 4$.

Nous voyons que l'algorithme converge très rapidement, et permet donc d'estimer la racine carrée d'un nombre moyennant un nombre limité d'opérations élémentaires (additions, soustractions, divisions, multiplications). Il reste encore à savoir si cet algorithme converge toujours et à déterminer la rapidité de sa convergence. L'analyse numérique est une discipline proche des mathématiques appliquées, qui a pour objectif de répondre à ces questions de façon rigoureuse.

Le script **Octave** suivant permet de générer les résultats du tableau ci-dessus. Si vous ne comprennez pas ces instructions pour le moment, la documentation sur **Octave** vous permettra d'en savoir plus. Durant la séance des TDs vous pourrez expérimenter l'effet du changement de la valeur initiale a_0 sur la rapidité de la convergence.

```
clear all
close all
format long
x=4;
a0=4;
e0p=(x/a0-a0)/2;
printf("%2s          %3s              %6s\n",'i','a_i','e0p_i')
printf("\n")
i=0;
printf("%2i    %12.8f    %13.9f\n",i,a0,e0p)
printf("\n")
for i=1:4
    a1=(x/a0+a0)/2;
    e1p=(x/a1-a1)/2;
    printf("%2i    %12.8f    %13.9f\n",i,a1,e1p)
    printf("\n")
    a0=a1;
end
```

Résultats du script ci-dessus

```
 i          a_i               e0p_i

 0      4.00000000     -1.500000000

 1      2.50000000     -0.450000000

 2      2.05000000     -0.049390244

 3      2.00060976     -0.000609663

 4      2.00000009     -0.000000093
```

0.2 Vocabulaire

0.2.1 Algorithme

La description précise des opérations successives à effectuer pour obtenir un certain résultat s'appelle un algorithme. Un algorithme peut donc être assimilé à un protocole ou une recette. Le mot, d'origine arabe, vient du nom **Al Khuwarizmi** d'un mathématicien perse qui vécut au IXème siècle. Les premiers algorithmes remontent à Euclide et même aux Babyloniens.

0.2.2 Calcul numérique

On appellera calcul numérique tout calcul (évaluation d'un nombre, d'une fonction, d'une matrice,...) qui est effectué au moyen d'une machine et/ou de tables de valeurs numériques.

0.2.3 Analyse numérique

On peut définir l'analyse numérique comme l'étude théorique des méthodes constructives de l'algèbre et de l'analyse mathématique. Par méthode constructive il faut comprendre une méthode qui fournit le moyen d'obtenir la solution d'un problème, ou, à tout le moins, une approximation de celle-ci. Ainsi, un théorème qui établit l'existence et l'unicité de la solution d'un problème de Cauchy (cf. cours d'analyse) ne sera d'aucune utilité au numéricien s'il n'indique pas comment construire effectivement la solution du problème.

0.2.4 Programme

Ensemble d'instructions destinées à être exécutées par l'ordinateur. Un programme est conservé sur le disque dur sous forme de fichiers exécutables (qui font appel les uns aux autres) et de différents fichiers de données contenant les paramètres. Pour s'exécuter, un programme doit être chargé en mémoire vive (mémoire de travail de l'ordinateur).

0.2.5 Application

Aussi appelée *Programme* ou *Logiciel*, elle désigne une entité informatique du domaine du « software » (pas d'existence solide) exécutée à la demande de l'utilisateur par le système d'exploitation dans un but précis. Ainsi parle-t-on d'*application de traitement de texte, de traitement d'images*, etc. Notons que les termes *logiciel* (ensemble des éléments informatiques qui permettent d'assurer une tâche ou une fonction) et *application* sont souvent utilisés de manière équivalente.

0.2.6 Langages de programmation

En *informatique*, un **langage de programmation** est une notation conventionnelle destinée à formuler des *algorithmes* et produire des *programmes informatiques* qui les appliquent[Wikipedia] . D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un *alphabet*, d'un *vocabulaire*, de règles de *grammaire* et de *significations*.

Les langages de programmation permettent de décrire d'une part les structures des données qui seront manipulées par l'appareil informatique, et d'autre part d'indiquer comment sont effectuées les manipulations, selon quels algorithmes. Ils servent de moyens de communication par lesquels le programmeur communique avec l'ordinateur en lui soumettant des instructions et en analysant les données matérielles fournies par ce dernier, mais aussi avec d'autres programmeurs ; les programmes étant d'ordinaire écrits, lus, compris et modifiés par une équipe de programmeurs.

Un langage de programmation est mis en œuvre par un traducteur automatique : *compilateur* ou *interprète*. Un compilateur est un programme informatique qui transforme dans un premier temps un *code source* écrit dans un langage de programmation donné en un code cible qui pourra être directement exécuté par un ordinateur, à savoir un programme en *langage machine* ou en *code intermédiaire*, tandis que l'interprète réalise cette traduction « à la volée ». L'activité de rédaction du **code source** d'un programme est commée **programmation**.

Les langages de programmation offrent différentes possibilités d'**abstraction**, et une notation proche de l'*algèbre*, permettant de décrire de manière concise et facile à saisir les opérations de manipulation de données et l'évolution du déroulement du programme en fonction des situations. La possibilité d'écriture abstraite libère l'esprit du programmeur d'un travail superflu, notamment de prise en compte des spécificités du matériel informatique, et lui permet ainsi de se concentrer sur des problèmes plus avancés.

Chaque langage de programmation supporte une ou plusieurs approches de la programmation –

paradigmes. Les notions induisant le paradigme font partie du langage de programmation et permettent au programmeur d'exprimer dans le langage une solution qui a été imaginée selon ce paradigme.

Les premiers langages de programmation ont été créés dans les années 1950 en même temps que l'avènement des ordinateurs. Cependant de nombreux concepts de programmation ont été initiés par un langage ou parfois plusieurs langages, avant d'être améliorés puis étendus dans les langages suivants. La plupart du temps la conception d'un langage de programmation a été fortement influencée par l'expérience acquise avec les langages précédents.

En résumé, un langage de programmation permet d'écrire un programme. Un programme, lorsqu'il est sous la forme de code source, est un texte qui exprime un algorithme, en vue de permettre l'exécution de ce dernier sur une machine.

Les langages utilisés pour programmer sont situés quelque part entre les séquences de 0 et 1 chères à la machine et le langage naturel cher à l'humain.

Voici un exemple de programme écrit en **Python** pour calculer les **nombres de Fibonacci** [2].

```
1 def fibonacci(n) :
2     f = [0, 1]
3     while len(f) < n + 1 :
4         f.append(f[-1] + f[-2])
5     return f
```

```
1 def main():
2     n = input("Entrez un nombre entier:")
3     n = int(n)
4     f = fibonacci(n)
5     print("suite de Fibonacci:U_{}={}\n".format(n,f))
6 main()
```

0.3 Exemples d'applications

Le calcul numérique est devenu indispensable dans de très nombreux domaines. La liste des disciplines qui ont recours au calcul et à l'analyse numériques est bien trop longue pour être dressée ici. Elle va de la physique à la démographie en passant par la chimie, la biologie, la médecine, le génie civil, l'archéologie, la gestion des stocks, la psychologie expérimentale, les jeux vidéos, etc.

Les sciences de l'ingénieur sont probablement celles qui font le plus appel au calcul numérique intensif. Les moyens informatiques modernes et les performances extraordinaires des microprocesseurs et des mémoires actuels ont permis un développement considérable des capacités de calcul des ordinateurs. A côté de ces matériels très performants, les chercheurs disposent de logiciels très élaborés (développés et mis au point par d'autres chercheurs). Il est devenu possible, par exemple, (ce n'était pas le cas avant les années 1990) de simuler numériquement des « crash-tests » incluant à la fois les déformations

2. Les **nombres de Fibonacci** sont les termes de la **suite de Fibonacci** [Fibonacci]. La **suite de Fibonacci** est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle est donc fondée sur la *formule de récurrence* suivante :

$$\mathcal{F}_{n+2} = \mathcal{F}_{n+1} + \mathcal{F}_n \qquad n \in \mathbb{N}$$

La suite de Fibonacci commence généralement par les termes 0 et 1 (parfois 1 et 1) et ses premiers termes sont : 0, 1, 1, 2, 3, 5, 8, 13, 21, etc. Elle doit son nom à **Leonardo Fibonacci** qui, dans un *problème récréatif* posé dans l'ouvrage **Liber abaci** publié en 1202, décrit la croissance d'une population de lapins : « Un homme met un couple de lapins dans un lieu isolé de tous les côtés par un mur. Combien de couples obtient-on en un an si chaque couple engendre tous les mois un nouveau couple à compter du troisième mois de son existence ? » . Notons $\mathcal{F}_n$ le nombre de couples de lapins au début du mois n. Jusqu'à la fin du deuxième mois, la population se limite à un couple, ce qu'on note : $F_1 = F_2 = 1$). Dès le début du troisième mois, le couple de lapins a deux mois et il engendre un autre couple de lapins ; on note alors F_3=2. On choisit alors de poser $F_0 = 0$, de manière que cette équation soit encore vérifiée pour $n = 0$.

des véhicules et les mouvements et chocs subis par les passagers (virtuels heureusement !). De telles simulations ont permis des avancées importantes en matière de sécurité automobile passive et active.

Un autre domaine qui a bénéficié des progrès du matériel et des logiciels informatiques est la météorologie. On peut, actuellement, lorsque les conditions de stabilité sont bonnes, prévoir l'évolution du temps sur une période de cinq à six jours. Ces prévisions sont effectuées dans des centres de calculs spécialisés (p.ex. Reading en Angleterre) et nécessitent de résoudre des millions de fois des systèmes de plusieurs millions d'équations à autant d'inconnues. Les ordinateurs utilisés dans ces centres peuvent effectuer plusieurs centaines de milliards d'opérations arithmétiques par seconde.

Une autre discipline dont le développement spectaculaire est lié au progrès des calculateurs numériques et de la modélisation mathématique est la bio-informatique. Certaines études, dans ce domaine, s'intéressent aux modifications conformationnelles de protéines qui sont à l'origine de maladies diverses telles la maladie d'Alzheimer et les encéphalopathies spongiformes. Ces études devraient permettre de mieux comprendre l'origine de ces maladies et par là, contribuer à les juguler.

Chapitre 1

Systèmes linéaires

On appelle système linéaire d'ordre n (n entier positif), une expression de la forme

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.1}$$

où $\mathbf{A} = (a_{ij})$, $1 \leq i, j \leq n$, désigne une matrice de taille $n \times n$ de nombres réels ou complexes, $\mathbf{b} = (b_i)$, $1 \leq i \leq n$, un vecteur colonne réel ou complexe et $\mathbf{x} = (x_i)$, $1 \leq i \leq n$, est le vecteur des inconnues du système. La relation précédente équivaut aux équations

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1, \cdots, n\,. \tag{1.2}$$

La matrice $\mathbf{A}$ est dite *régulière* (ou *non singulière*) si $det(\mathbf{A}) \neq 0$; on a existence et unicité de la solution $\mathbf{x}$ (pour n'importe quel vecteur $\mathbf{b}$ donné) si et seulement si la matrice associée au système linéaire est régulière. Théoriquement, si $\mathbf{A}$ est non singulière, la solution est donnée par *la formule de cramer* :

$$x_i = \frac{det\,(\mathbf{A}_i)}{det\,(\mathbf{A})}, \quad i = 1, \cdots, n \tag{1.3}$$

où $\mathbf{A}_i$ est la matrice obtenue en remplaçant la i-ème colonne de $\mathbf{A}$ par le vecteur $\mathbf{b}$. Cependant l'application de cette formule est inacceptable pour la résolution pratique des systèmes, car son coût est de l'ordre de $(n+1)!$ *floating-point operations (flops)*. En fait, le calcul de chaque déterminant par la formule

$$det(\mathbf{A}) = \sum_{\sigma} (-1)^{\epsilon(\sigma)} \prod_{i=1}^{n} a_{i,\sigma(i)} \tag{1.4}$$

(où la somme est étendue à toutes les permutations σ sur n objets et $\epsilon(\sigma)$ désigne la **signature** [1] de la permutation σ) requiert $n!$ flops. Par example, sur un ordinateur effectuant 10^9 flops par seconde, il faudrait $9,6.10^{47}$ années pour résoudre un système linéaire de seulement 50 équations. Il faut donc développer des algorithmes alternatives avec un coût raisonnable. Dans les sections suivantes, plusieurs méthodes sont analysées.

On appelle méthode de résolution **directe** d'un système linéaire un algorithme qui, si l'ordinateur faisait des calculs exacts, donnerait la solution en un nombre fini d'opérations. Il existe aussi des méthodes **itératives** qui consistent à construire une suite de vecteurs $\mathbf{x}_n$ convergeant vers la solution $\mathbf{x}$.

1. En mathématiques, une permutation est dite paire si elle présente un nombre pair d'inversions, impaire sinon. La signature d'une permutation vaut 1 si celle-ci est paire, –1 si elle est impaire.

1.1 Méthodes directes

Pour résoudre $\mathbf{Ax} = \mathbf{b}$, on cherche à écrire $\mathbf{A} = \mathbf{LU}$ où :
— $\mathbf{L}$ est une matrice triangulaire inférieure avec des 1 sur la diagonale,
— $\mathbf{U}$ est une matrice triangulaire supérieure.
La résolution de $\mathbf{Ax} = \mathbf{b}$ est alors ramenée aux résolutions successives des systèmes échelonnés $\mathbf{Ly} = \mathbf{b}$ et $\mathbf{Ux} = \mathbf{y}$.

1.1.1 Résolution des systèmes triangulaires

Une matrice $\mathbf{A} = (a_{ij})$ est *triangulaire supérieure* si

$$a_{ij} = 0 \quad \forall i\,,\ j\ :\ 1 \leq j < i \leq n \tag{1.5}$$

et *triangulaire inférieure* si

$$a_{ij} = 0 \quad \forall i\,,\ j\ :\ 1 \leq i < j \leq n\cdot \tag{1.6}$$

Suivant ces cas, le système à résoudre est dit *système triangulaire supérieur ou inférieur*. Si la matrice $\mathbf{A}$ est régulière et triangulaire alors, comme $det(\mathbf{A}) = \prod a_{ii}$, on déduit que $a_{ii} \neq 0$, pour tout $i = 1, \cdots, n$.

Si $\mathbf{A}$ est triangulaire inférieure, on a

$$x_1 = \frac{b_1}{a_{11}}\,, \tag{1.7}$$

et pour $i = 2,\ 3,\ \cdots,\ n$,

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j\right). \tag{1.8}$$

Cet algorithme est appelé *méthode de descente*.

Si $\mathbf{A}$ est triangulaire supérieure, on a

$$x_n = b_n/a_{nn} \tag{1.9}$$

et pour $i = n-1,\ n-2, \cdots,\ 2,\ 1$

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right). \tag{1.10}$$

Cet algorithme est appelé *méthode de remontée*.

Le nombre de multiplications et de divisions nécessaires dans cet algorithme est de $\frac{n(n+1)}{2}$ et le nombre d'additions et de soustractions est de $\frac{n(n-1)}{2}$, ce qui implique que l'algorithme nécessite n^2 flops.

1.1.2 Matrices tridiagonales

On considère le cas particulier où la matrice $\mathbf{A}$ est non singulière et tridiagonale, donnée par

$$\begin{bmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ 0 & & b_n & a_n \end{bmatrix} \tag{1.11}$$

Dans ce cas, les matrices **L** et **U** de factorisation **LU** de **A** sont des matrices bidiagonales de la forme :

$$\mathbf{L} = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} \alpha_1 & c_1 & & 0 \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & \alpha_n \end{bmatrix} \tag{1.12}$$

Les coefficients α_i et β_i peuvent être calculés par les relations :

$$\alpha_1 = a_1,\ \beta_2\alpha_1 = b_2 \Rightarrow \beta_2 = \frac{b_2}{\alpha_1},\ \beta_2 c_1 + \alpha_2 = a_2 \rightarrow \alpha_2 = a_2 - \beta_2 c_1, \cdots \tag{1.13}$$

Donc, on a

$$\alpha_1 = a_1,\ \beta_i = \frac{b_i}{\alpha_{i-1}},\ \alpha_i = a_i - \beta_i c_{i-1}, \quad i = 2, \cdots, n. \tag{1.14}$$

Cet algorithme est connu sous le nom de *algorithme de Thomas*, et le nombre d'opérations effectuées est de l'ordre de n. Avec l'algorithme de Thomas, résoudre un système tridiagonal $\mathbf{Ax} = \mathbf{f}$ revient à résoudre 2 systèmes bidiagonaux $\mathbf{Ly} = \mathbf{f}$ et $\mathbf{Ux} = \mathbf{y}$ avec

$$\mathbf{Ly} = \mathbf{f} \ \leftrightarrow\ y_1 = f_1,\ y_i = f_i - \beta_i y_{i-1}, \quad i = 2, \cdots, n. \tag{1.15}$$

$$\mathbf{Ux} = \mathbf{y} \ \leftrightarrow\ x_n = \frac{y_n}{\alpha_n},\ x_i = \frac{y_i - c_i x_{i+1}}{\alpha_i}, \quad i = n-1, \cdots, 1. \tag{1.16}$$

Le coût de calcul ici est de $3(n-1)$ opérations pour la factorisation et de $5n-4$ opérations pour la substitution, ce qui fait un total de $8n-7$ opérations.

1.2 Méthodes itératives

1.2.1 Introduction

L'idée des méthodes itératives est de construire une suite de vecteurs $\mathbf{x}^{(k)}$ qui converge vers le vecteur $\mathbf{x}$ solution du système $\mathbf{Ax} = \mathbf{b}$,

$$\mathbf{x} = \lim_{k\to\infty} \mathbf{x}^{(k)}\,. \tag{1.17}$$

L'intérêt des méthodes itératives, comparées aux méthodes directes, est d'être simples à programmer et de nécessiter moins de place en mémoire. En revanche le temps de calcul est souvent plus long.

Une stratégie est de considérer la relation de récurrence linéaire

$$\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{g}\,, \tag{1.18}$$

où **B** est la *matrice d'itération* de la méthode itérative (dépendant de **A**) et **g** est un vecteur (dépendant de **b**), tels que

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{g}. \tag{1.19}$$

Comme $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, on a $\mathbf{g} = (\mathbf{I} - \mathbf{B})\mathbf{A}^{-1}\mathbf{b}$; la méthode itérative (1.18) est donc complètement définie par la matrice **B**.

En définissant l'erreur au pas k comme

$$\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}, \tag{1.20}$$

on obtient la relation de récurrence

$$\mathbf{e}^{(k)} = \mathbf{B}\mathbf{e}^{(k-1)},\ \text{et donc}\ \mathbf{e}^{(k)} = \mathbf{B}^{(k)}\mathbf{e}^{(0)},\ k = 0,\ 1,\ \cdots \tag{1.21}$$

On démontre que $\lim_{k\to\infty} \mathbf{e}^{(k)} = 0$ pour tout $\mathbf{e}^{(0)}$(et donc pour tout $\mathbf{x}^{(0)}$) si et seulement si $\rho(\mathbf{B}) < 1$, où $\rho(\mathbf{B})$ est le *rayon spectral* de la matrice $\mathbf{B}$, défini comme

$$\rho(\mathbf{B}) = max\,|\lambda_i(\mathbf{B})|\ , \tag{1.22}$$

les $\lambda_i(\mathbf{B})$ étant les valeurs propres de la matrice $\mathbf{B}$.

Une technique générale pour construire des méthodes itératives est basée sur une décomposition *(splitting)* de la matrice $\mathbf{A}$ sous forme $\mathbf{A} = \mathbf{P} - \mathbf{N}$, où $\mathbf{P}$ et $\mathbf{N}$ sont des matrices à déterminer avec $\mathbf{P}$ non singulière . La matrice $\mathbf{P}$ est appelée *matrice de préconditionnement*. Plus précisément, $\mathbf{x}^{(0)}$ étant donné, on peut calculer $\mathbf{x}^{(k)}$, pour $k \geq 1$, en résolvant le système

$$\mathbf{P}\mathbf{x}^{(k+1)} = \mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}, \quad k \geq 0\,. \tag{1.23}$$

Clairement, la solution exacte $\mathbf{x}$ satisfait $\mathbf{P}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{b}$ et donc $\mathbf{A}\mathbf{x} = \mathbf{b}$. Le système (1.23) peut être écrit également sous la forme (1.18), avec $\mathbf{B} = \mathbf{P}^{-1}\mathbf{N}$, et $\mathbf{g} = \mathbf{P}^{-1}\mathbf{b}$.

Une relationn de récurrence équivalente à (1.23) est la suivante :

$$\mathbf{P}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = \mathbf{r}^{(k)}, \quad k \geq 0\,, \tag{1.24}$$

où $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ est le *résidu à l'itération k*. On peut généraliser la relation précédente comme

$$\mathbf{P}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = \alpha_k\mathbf{r}^{(k)}, \quad k \geq 0\,, \tag{1.25}$$

où on a introduit un paramètre α_k (qui peut être différent à chaque itération k) afin d'accélérer la convergence. Cette méthode est dite de **Richardson**. Les relations (1.23), (1.24) et (1.25) montrent qu'on doit résoudre un système linéaire de matrice $\mathbf{P}$ à chaque itération ; donc $\mathbf{P}$ doit être telle que la résolution du système ait un coût raisonnable. Par exemple, on pourra choisir $\mathbf{P}$ diagonale ou triangulaire.

1.2.2 La méthode de Jacobi

On remarque que si les éléments diagonaux de $\mathbf{A}$ sont non nuls, le système linéaire $\mathbf{A}\mathbf{x} = \mathbf{b}$ est équivalent à :

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right)\ , \quad i = 1,\ \cdots,\ n. \tag{1.26}$$

Pour une donnée initiale $\mathbf{x}^{(0)}$ choisie, on calcule $\mathbf{x}^{(k+1)}$ par

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \Big/ a_{ii}\,, \quad i = 1,\ \cdots,\ n. \tag{1.27}$$

Cela permet d'identifier le *splitting* suivant pour $\mathbf{A}$:

$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$

$\mathbf{D}$: diagonale de $\mathbf{A}$

$\mathbf{E}$: triangulaire inférieure avec des 0 sur la diagonale

$\mathbf{F}$: triangulaire supérieure avec des 0 sur la diagonale

La **matrice d'itération** de la méthode de Jacobi est donnée par :

$$\mathbf{B}_J = \mathbf{D}^{-1}\left(\mathbf{E} + \mathbf{F}\right) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}\cdot \tag{1.28}$$

L'algorithme de Jacobi nécessite le stockage des deux vecteurs $\mathbf{x}_j^{(k)}$ et $\mathbf{x}_j^{(k+1)}$. A chaque itération, on effectue $(n-1)$ multiplications, n additions et une division. Pour stocker $\mathbf{A}$ et les vecteurs $\mathbf{b}$, $\mathbf{x}^{(k)}$

et $\mathbf{x}^{(k+1)}$ on utilise (n^2+3n) mémoires. La méthode ne converge pas toujours.

Exemple. Considérons le système

$$\begin{pmatrix} 4 & 2 & 1 \\ -1 & 2 & 0 \\ 2 & 1 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 9 \end{pmatrix}$$

mis sous la forme

$$\begin{cases} x = 1 - y/2 - z/4 \\ y = 1 + x/2 \\ z = 9/4 - x/2 - y/4 \end{cases}$$

Soit $\mathbf{x}^{(0)} = (0,\, 0,\, 0)$ le vecteur initial. En calculant les itérations, on trouve :

$$\mathbf{x}^{(1)} = (1,\, 1,\, 9/4)$$
$$\mathbf{x}^{(2)} = (-1/16,\, 3/2,\, 3/2)$$
$$\mathbf{x}^{(3)} = (-1/8,\, -1/32,\, 61/32)$$
$$\mathbf{x}^{(4)} = (5/128,\, 15/16,\, 265/128)$$
$$\mathbf{x}^{(5)} = (7/512,\, 261/256,\, 511/256)$$

La suite $\mathbf{x}^{(k)}$ converge vers la solution du système $(0,\, 1,\, 2)$.

1.2.3 La méthode de Gauss-Seidel

Dans la méthode de Gauss-Seidel, publiée en 1874 par Ludwig Seidel (1821-1896), on a :

$$x_i^{(k+1)} = \frac{1}{a_{\,ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1,\, \cdots,\, n. \tag{1.29}$$

Il s'écrit aussi

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \mathbf{E})^{-1} \left(\mathbf{b} + \mathbf{F}\mathbf{x}^{(k)} \right). \tag{1.30}$$

Dans ce cas, le *splitting* de $\mathbf{A}$ est

$$\mathbf{P} = \mathbf{D} - \mathbf{E}, \qquad \mathbf{N} = \mathbf{F}, \tag{1.31}$$

et la matrice d'itération associée est

$$\mathbf{B}_{GS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{N} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}. \tag{1.32}$$

L'algorithme de Gauss-Seidel ne nécessite qu'un vecteur de stockage, $\mathbf{x}^{(k)}$ étant remplacé par $\mathbf{x}^{(k+1)}$ au cours de l'itération. Il est en général plus rapide que l'algorithme de Jacobi, donc préférable. En effet, les valeurs calculées sont utilisées au fur et à mesure du calcul et non à l'issue d'une itération comme dans la méthode de Jacobi. On améliore ainsi la vitesse de convergence.

Considérons un système à trois équations

$$\begin{cases} x = (b_1 - a_{12}y - a_{13}z)/a_{11} \\ y = (b_2 - a_{21}x - a_{23}z)/a_{22} \\ z = (b_3 - a_{31}x - a_{32}y)/a_{33} \end{cases} \tag{1.33}$$

A la première itération,on calcule à partir du vecteur initial

$$\mathbf{x}^{(0)} = \left(x^{(0)},\, y^{(0)},\, z^{(0)}\right) \tag{1.34}$$

la valeur $x^{(1)}$:

$$x^{(1)} = \left(b_1 - a_{12}y^{(0)} - a_{13}z^{(0)}\right) \Big/ a_{11}. \tag{1.35}$$

Cette valeur est réintroduite immédiatement dans le calcul de la deuxième composante (ce qui différencie cette méthode de la méthode de Jacobi, car on utilise ici la valeur $x^{(1)}$ et non $x^{(0)}$) :

$$y^{(1)} = \left(b_2 - a_{21}x^{(1)} - a_{23}z^{(0)}\right) \Big/ a_{22}. \tag{1.36}$$

De même, on porte $x^{(1)}$ et $y^{(1)}$ dans le calcul de $z^{(1)}$:

$$z^{(1)} = \left(b_3 - a_{31}x^{(1)} - a_{32}y^{(1)}\right) \Big/ a_{33} \tag{1.37}$$

A chaque itération, on effectue $(n-1)$ multiplications, n additions et une division. Pour stocker $\mathbf{A}$ et les vecteurs $\mathbf{b}$, $\mathbf{x}^{(k)}$ et $\mathbf{x}^{(k+1)}$, on utilise (n^2+2n) mémoires. Si $\mathbf{A}$ et $\mathbf{b}$ sont calculés, on emploie n mémoires. La méthode ne converge pas toujours.

Exemple. Considérons le système

$$\begin{cases} x = 1 - y/2 - z/4 \\ y = 1 + x/2 \\ z = 9/4 - x/2 - y/4 \end{cases}$$

Partant du point $\mathbf{x}^{(0)} = (0,\, 0,\, 0)$, on calcule successivement

$$\mathbf{x}^{(1)} = (1,\, 3/2,\, 11/8)$$
$$\mathbf{x}^{(2)} = (-3/32,\, 61/64,\, 527/256)$$
$$\mathbf{x}^{(3)} = (9/1024,\, 2047/2048,\, 16349/8192)$$

Cet ensemble de points converge vers la solution $(0,\, 1,\, 2)$.

La méthode de Gauss-Seidel est aussi utilisée pour résoudre des systèmes non linéaires.

Exemple : Soit à résoudre le système

$$\begin{cases} x = \sin(xy) - y/2\pi \\ y = 2\pi x - (\pi - 1/4)\left(\mathrm{e}^{2x-1} - 1\right) \end{cases}$$

Partant du point $(2/5,\, 3)$, on calcule successivement

$$\mathbf{x}^{(1)} = (0.455,\, 3.03)$$
$$\mathbf{x}^{(2)} = (0.499,\, 3.11)$$
$$\mathbf{x}^{(3)} = (0.505,\, 3.14)$$

qui converge vers la solution $x = 1/2$, $y = \pi$.

1.2.4 Convergence des méthodes de Jacobi et de Gauss-Seidel

On a les résultats suivants

■ Si **A** est une matrice à diagonale dominante stricte par lignes (ou par colonne), c'est-à-dire,

$$\sum_{j=1,j\neq i}^{n} |a_{ij}| < |a_{ii}| \quad i = 1,\, 2,\, 3\cdots,\, n \tag{1.38}$$

ou

$$\sum_{i=1,j\neq i}^{n} |a_{ij}| < |a_{jj}| \quad j = 1,\, 2,\, 3\cdots,\, n, \tag{1.39}$$

alors les méthodes de Jacobi et de Gauss-Seidel sont convergentes. Par conséquent, on peut avoir intérêt à réarranger les termes de **A** de façon à mettre **A** sous la forme d'une matrice dont les éléments diagonaux sont les plus grands possibles.

Démonstration. Nous prouvons la partie concernant la méthode de Jacobi. Soit **x** la solution du système linéaire ; on a

$$x_i^{(n+1)} - x_i = \frac{-1}{a_{ii}} \sum_{k\neq i} a_{ik} \left(x_k^{(n)} - x_k \right) \tag{1.40}$$

et donc

$$\begin{aligned} \max_i \left| x_i^{(n+1)} - x_i \right| &\leq \frac{1}{a_{ii}} \sum_{k\neq i} |a_{ik}| \max_k \left| x_k^{(n)} - x_k \right| && (1.41)\\ &\leq K \max_k \left| x_k^{(n)} - x_k \right| \text{ avec } 0 \leq K < 1 && (1.42) \end{aligned}$$

car **A** est à diagonale strictement dominante. De plus $\max_i |y_i| = ||\mathbf{y}||_\infty$ est une norme sur $\mathbb{R}^N$. On a ainsi montré que

$$||\mathbf{x}^{(n+1)} - \mathbf{x}||_\infty \leq K ||\mathbf{x}^{(n)} - \mathbf{x}||_\infty \,, \tag{1.43}$$

ce qui prouve la convergence.

Remarque : En pratique la stricte dominance n'est pas indispensable. L'inégalité large suffit pour la plupart des matrices inversibles.

■ Si **A** est une matrice symétrique définie positive, alors la méthode de Gauss-Seidel converge (la méthode de Jacobi pas forcément).

1.2.5 Méthode du gradient à pas optimal

Considérons la méthode itérative (1.25) avec $\alpha_k \neq 1$. Si **P** et **A** sont symétriques définies positives, alors on a un critère optimal pour le choix de α_k :

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)} \,,\, \mathbf{z}^{(k)} \right\rangle}{\left\langle \mathbf{A}\mathbf{z}^{(k)} \,,\, \mathbf{z}^{(k)} \right\rangle} \qquad k \geq 0 \,, \tag{1.44}$$

où $\mathbf{z}^{(k)} = \mathbf{P}^{-1}\mathbf{r}^{(k)}$. Cette méthode est appelée *du gradient préconditionné* (du gradient lorsque $\mathbf{P} = \mathbf{I}$). On a noté par $\langle \mathbf{x} \,,\, \mathbf{y} \rangle$ le produit scalaire entre les vecteurs **x** et **y**.

Démonstration. D'une part, on a

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} = \mathbf{A}\left(\mathbf{x} - \mathbf{x}^{(k)} \right) = \mathbf{A}\mathbf{e}^{(k)} \tag{1.45}$$

où $\mathbf{e}^{(k)}$ représente l'erreur à l'étape k, et d'autre part

$$\mathbf{e}^{(k+1)} = \mathbf{e}^{(k+1)}(\alpha) = \left(\mathbf{I} - \alpha \mathbf{P}^{-1}\mathbf{A} \right) \mathbf{e}^{(k)} \,, \tag{1.46}$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)}(\alpha) - \alpha \mathbf{A}\mathbf{z}^{(k)}. \tag{1.47}$$

Ainsi, en notant $||\cdot||_A$ la norme vectoriell issue du produit scalaire $\langle \mathbf{x}, \mathbf{y} \rangle_A = \langle \mathbf{A}\mathbf{x}, \mathbf{y} \rangle$, c'est-à-dire $||\mathbf{x}||_A = \langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle^{1/2}$, on tire que

$$\begin{aligned} ||\mathbf{e}^{(k+1)}||_A^2 &= \left\langle \mathbf{A}\mathbf{e}^{(k+1)}, \mathbf{e}^{(k+1)} \right\rangle = \left\langle \mathbf{r}^{(k+1)}, \mathbf{e}^{(k+1)} \right\rangle & (1.48) \\ &= \left\langle \mathbf{r}^{(k)}, \mathbf{e}^{(k)} \right\rangle - \alpha \left(\left\langle \mathbf{r}^{(k)}, \mathbf{P}^{-1}\mathbf{A}\mathbf{e}^{(k)} \right\rangle + \left\langle \mathbf{A}\mathbf{z}^{(k)}, \mathbf{e}^{(k)} \right\rangle \right) \\ &+ \alpha^2 \left\langle \mathbf{A}\mathbf{z}^{(k)}, \mathbf{P}^{-1}\mathbf{A}\mathbf{e}^{(k)} \right\rangle. & (1.49) \end{aligned}$$

Maintenant, on choisit α comme le α_k qui minimise $||\mathbf{e}^{(k+1)}||_A$, c'est-à-dire

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} ||\mathbf{e}^{(k+1)}||_A^2 \Big|_{\alpha=\alpha_k} = 0. \tag{1.50}$$

On obtient donc

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{z}^{(k)} \right\rangle}{\left\langle \mathbf{A}\mathbf{z}^{(k)}, \mathbf{z}^{(k)} \right\rangle}, \ \textbf{cqd}. \tag{1.51}$$

On peut démontrer que la suite $\left\{ \mathbf{x}^{(k)} \right\}$ donnée par la méthode du gradient converge vers $\mathbf{x}$ lorsque $k \to \infty$, et

$$||\mathbf{x}^{(k+1)} - \mathbf{x}||_A \leq \left(\frac{K_2(\mathbf{P}^{-1}\mathbf{A}) - 1}{K_2(\mathbf{P}^{-1}\mathbf{A}) + 1} \right)^k ||\mathbf{x}_0 - \mathbf{x}||_A, \quad k \geq 0. \tag{1.52}$$

En général, on choisit $\mathbf{P}$ de façon à avoir

$$K_2\left(\mathbf{P}^{-1}\mathbf{A}\right) << K_2(\mathbf{A}). \tag{1.53}$$

En conséquence,

$$\frac{K_2(\mathbf{P}^{-1}\mathbf{A}) - 1}{K_2(\mathbf{P}^{-1}\mathbf{A}) + 1} << \frac{K_2(\mathbf{A}) - 1}{K_2(\mathbf{A}) + 1}. \tag{1.54}$$

1.2.6 Méthode du gradient conjugué

Une méthode encore plus rapide dans le cas où $\mathbf{P}$ et $\mathbf{A}$ sont symmétriques définies positives est celle du *gradient conjugué préconditionné*. On pose cette fois-ci

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}, \quad k \geq 0 \tag{1.55}$$

avec $\mathbf{p}^{(k)} = \mathbf{z}^{(k)} - \beta_k \mathbf{p}^{(k-1)}$.

Soit $\mathbf{x}^{(0)}$ une donnée initiale. On calcule $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} = \mathbf{P}^{-1}\mathbf{r}^{(0)}$, $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$, puis pour $k \geq 0$,

$$\alpha_k = \frac{\left\langle \mathbf{r}^{(k)}, \mathbf{p}^{(k)} \right\rangle}{\left\langle \mathbf{A}\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle} \tag{1.56a}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)} \tag{1.56b}$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{p}^{(k)} \tag{1.56c}$$

$$\mathbf{z}^{(k+1)} = \mathbf{P}^{-1}\mathbf{r}^{(k+1)} \tag{1.56d}$$

$$\beta_{k+1} = \frac{\left\langle \mathbf{A}\mathbf{p}^{(k)}, \mathbf{z}^{(k+1)} \right\rangle}{\left\langle \mathbf{A}\mathbf{p}^{(k)}, \mathbf{p}^{(k)} \right\rangle} \tag{1.56e}$$

$$\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} - \beta_{k+1}\mathbf{p}^{(k)}. \tag{1.56f}$$

Dans ce cas, la formule (1.52) devient :

$$||\mathbf{x}^{(k+1)} - \mathbf{x}||_A \leq 2\left(\frac{\sqrt{K_2(\mathbf{P}^{-1}\mathbf{A})}-1}{\sqrt{K_2(\mathbf{P}^{-1}\mathbf{A})}+1}\right)^k ||\mathbf{x}^{(0)} - \mathbf{x}||_A\,, \quad k \geq 0\cdot \tag{1.57}$$

1.2.7 Méthodes de la relaxation

La convergence d'une méthode itérative ne dépend pas du choix du vecteur initial $\mathbf{x}^{(0)}$, mais la rapidité de concergence en dépend. D'où l'idée d'introduire un facteur de relaxation ω non nul. Les matrices $\mathbf{M}$ et $\mathbf{N}$ sont choisies comme dans la méthode de Gauss-Seidel mais pondérées par le facteur de relaxation :

$$\mathbf{M} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{E}\right) \quad \text{et} \quad \mathbf{N} = \frac{1-\omega}{\omega}\mathbf{D} + \mathbf{F}\cdot \tag{1.58}$$

La matrice

$$\mathbf{L} = \mathbf{M}^{-1}\mathbf{N} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{E}\right)^{-1}\left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{F}\right) \tag{1.59}$$

est appelée *matrice de relaxation*. L'algorithme est fondé sur le calcul des itérés

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right)\cdot \tag{1.60}$$

On démontre que si le facteur de relaxation dépasse 2, la méthode diverge. Pour $\omega = 1$, on retrouve la méthode de Gauss-Seidel. Lorsque $0 < \omega < 1$, on parle de *sous-relaxation* et lorsque $1 < \omega < 2$, on parle de *surrelaxation* (SOR : *Successive Over Relaxation*). La théorème d'ostrowski-Reich affirme que si $\mathbf{A}$ est une **matrice définie positive** et si le facteur de relaxation est tel que $0 < \omega < 2$, alors la méthode converge. Lorsque $\mathbf{A}$ est une matrice tridiagonale par blocs dont les blocs diagonaux sont inversibles, si l'on note $\mathbf{J}$ la matrice $\mathbf{J} = \mathbf{D}^{-1}\mathbf{M} = \mathbf{D}^{-1}(\mathbf{E}+\mathbf{F})$, et $\rho(\mathbf{J})$ son rayon spectral (c'est-à-dire le plus grand module des valeurs propres de $\mathbf{J}$), alors la valeur optimale du facteur de relaxation est donnée par

$$\omega_o = \frac{2}{1+\sqrt{1-\rho(\mathbf{J})^2}}\cdot \tag{1.61}$$

Dans certains cas, on utilise différents facteurs ω pour différents blocs de $\mathbf{A}$: c'est la méthode de relaxation par blocs.

Exemple. Pour un système de trois équations à trois inconnues, l'itération conduit à calculer :

$$\begin{cases} x^{(k+1)} = x^{(k)} + \omega\left(b_1 - a_{11}x^{(k)} - a_{12}y^{(k)} - a_{13}z^{(k)}\right)/a_{11} \\ y^{(k+1)} = y^{(k)} + \omega\left(b_2 - a_{21}x^{(k+1)} - a_{22}y^{(k)} - a_{23}z^{(k)}\right)/a_{22} \\ z^{(k+1)} = z^{(k)} + \omega\left(b_3 - a_{31}x^{(k+1)} - a_{32}y^{(k+1)} - a_{33}z^{(k)}\right)/a_{33} \end{cases} \tag{1.62}$$

La *surrelaxation successive symétrique* (SSOR : *Symetric Successive Over Relaxation*) consiste à faire jouer le même rôle aux matrices $\mathbf{E}$ et $\mathbf{F}$, en introduisant un vecteur intermédiaire $\mathbf{y}$ d'itérée $y^{(k)}$:

$$\begin{cases} \left(\frac{1}{\omega}\mathbf{D} - \mathbf{E}\right)y^{(k)} = \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{F}\right)x^{(k)} \\ \left(\frac{1}{\omega}\mathbf{D} - \mathbf{F}\right)x^{(k+1)} = \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{E}\right)y^{(k)} \end{cases} \tag{1.63}$$

1.2.8 Calcul de l'inverse d'une matrice carrée par l'algorithme de Faddev-Leverrier

Le calcul de l'inverse d'une matrice est une opération non seulement coûteuse, mais qui peut aussi s'avérer très peu stable.

Les **formules de Faddev ou de Leverrier** offrent une alternative pour le calcul de l'inverse de $\mathbf{A}$: posons $\mathbf{B}_0 = \mathbf{I}$, et calculons par récurrence

$$\alpha_k = \frac{1}{k}\mathrm{tr}(\mathbf{A}\mathbf{B}_{k-1})\,, \quad \mathbf{B}_k = -\mathbf{A}\mathbf{B}_{k-1} + \alpha_k\mathbf{I}\,, \quad k = 1\,,\ 2\,,\ \cdots\,,\ n. \tag{1.64}$$

Puisque $\mathbf{B}_n = \mathbf{0}$, si $\alpha_n \neq 0$ on obtient

$$\mathbf{A}^{-1} = \frac{1}{\alpha_n}\mathbf{B}_{n-1} \tag{1.65}$$

et le coût de la méthode pour une matrice pleine est de $(n-1)n^3$ flops. Pour plus de détails, voir **FF63** [2] et **Bar89** [3].

1.3 Exercices

Exercice 1. On veut résoudre le système linéaire $\mathbf{A}\mathbf{x} = \mathbf{b}$ où

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 6 & 8 \end{bmatrix} \qquad \text{et} \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

par la méthode d'élimination de Gauss.

a) Vérifier que l'algorithme de Gauss sans pivoting ne peut pas être exécuté jusqu'au bout.
b) Trouver une matrice de permutation $\mathbf{P}$ telle que la matrice $\mathbf{PA}$ soit factorisable.
c) Calculer la factorisation $\mathbf{LU}$ de la matrice $\mathbf{PA}$.
d) Résoudre le système linéaire $\mathbf{A}\mathbf{x} = \mathbf{b}$ en remplaçant $\mathbf{PA}$ par $\mathbf{LU}$ et en utilisant les algorithmes de substitution progressive et rétrograde.

Exercice 2.

1) Pour une matrice quelconque $\mathbf{A}$, montrer que $\rho(\mathbf{A}) \leq ||\mathbf{A}||$ pour toute norme matricielle. Donner l'expression de $||\mathbf{A}||_\infty$.
2) On suppose $\mathbf{A}$ symétrique définie positive. Montrer que $\rho(\mathbf{A}) = ||\mathbf{A}||_2$.
3) Soit

$$\mathbf{A} = \begin{bmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{bmatrix}.$$

Pour quelle valeur de a $\mathbf{A}$ est-elle définie positive ?

4) Ecrire la matrice $\mathbf{J}$ de l'itération de Jacobi.
5) Pour quelles valeurs de a la méthode de Jacobi converge-t-elle ?
6) Ecrire la matrice $\mathbf{G}$ de l'itération de Gauss-Seidel.
7) Calculer $\rho(\mathbf{G})$. Pour quelles valeurs de a cette méthode converge-t-elle plus vite que celle de Jacobi ?

2. Faddeev D. K. and Faddeeva V. N. (1963) *Computational Methods of Linear Algebra*. Freeman, San Francisco and London.
3. Barnett S. (1989) **Leverrier's Algorithm : A New Proof and Extensions**. *Numer. Math.* **7** : 338-352.

Exercice 3. Soit $\mathbf{A}$ une matrice décomposée en $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$

$$\mathbf{A} = \begin{pmatrix} \ddots & & -\mathbf{F} \\ & \mathbf{D} & \\ -\mathbf{E} & & \ddots \end{pmatrix}. \tag{1.66}$$

Pour résoudre $\mathbf{Ax} = \mathbf{b}$, on propose la méthode

$$\left(\frac{\mathbf{D}}{\omega} - \mathbf{E}\right)\mathbf{x}^{(k+1)} = \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{F}\right)\mathbf{x}^{(k)} + \mathbf{b} \qquad \omega \in \mathbb{R}^*. \tag{1.67}$$

a) Vérifier que si la méthode converge, elle converge vers une solution de $\mathbf{Ax} = \mathbf{b}$.

b) Donner la matrice d'itération $\mathbf{I}_\omega$ de cette méthode.
c) Calculer $\det(\mathbf{I}_\omega)$.
d) En déduire que $\rho(\mathbf{I}_\omega) \geq |1 - \omega|$. Conclusion ?

Exercice 4. On considère la matrice

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 3 \end{bmatrix}. \tag{1.68}$$

On se donne également un vecteur $\mathbf{b}$ de $\mathbb{R}^3$. On note $\bar{\mathbf{x}}$ l'unique solution du système $\mathbf{Ax} = \mathbf{b}$.

a) Montrer que la méthode de Jacobi pour résoudre le système linéaire $\mathbf{Ax} = \mathbf{b}$ est ici convergente.

b) Soit $\mathbf{x}_n$ le $n^{ième}$ itéré de la méthode de Jacobi en partant de $\mathbf{x}_0 = \mathbf{0}$. Déterminer la matrice d'itération $\mathbf{M}$ de la méthode de Jacobi (c'est la matrice qui vérifie $\mathbf{x}_{n+1} - \bar{\mathbf{x}} = \mathbf{M}(\mathbf{x}_n - \bar{\mathbf{x}})$).

c) Calculer les valeurs propres de $\mathbf{A}$. En déduire les valeurs propres de $\mathbf{M}$. Déterminer le module s de la plus petite valeur propre en module de $\mathbf{A}$, et S le module de la plus grande valeur propre en module de $\mathbf{M}$. (Vérifier que $S < 1$!)

d) Montrer que $|\bar{\mathbf{x}}| \geq \frac{1}{s}|\mathbf{b}|$ où $|\mathbf{u}|$ désigne la **norme euclidienne** usuelle du vecteur $\mathbf{u}$ dans $\mathbb{R}^3$.

e) Montrer que $|\mathbf{x}_n - \bar{\mathbf{x}}| \geq \frac{S^n}{s}|\mathbf{b}|$.

Chapitre 2

Résolution numérique d'équations non linéaires

Soit f : Ω fermé $\subset \mathbb{R}^p \to \mathbb{R}^p$. On cherche $\xi \in \Omega$ tel que $f(\xi) = 0$. Nous supposerons qu'un moyen (une étude graphique, une raison physique, un théorème, l'intuition ou ... la chance !) nous ont permis de voir que ξ était l'unique solution dans Ω . Nous supposerons aussi que f est au moins continue sur Ω.

Nous nous intéressons d'abord à l'approximation des zéros (ou racines dans le cas d'un polynôme) d'une fonction réelle d'une variable réelle, c'est-à-dire, étant donné un intervalle $I \subseteq \mathbb{R}$ et une application f de I dans $\mathbb{R}$, la résolution approchée du problème : *trouver* $\xi \in \mathbb{R}$ *(ou plus généralement* $\mathbb{C}$*) tel que* $f(\xi) = 0$.

Ce problème intervient notamment dans l'étude générale de fonctions d'une variable réelle, qu'elle soit motivée ou non par des applications [1] pour lesquelles des solutions exactes de ce type d'équation ne sont pas connues [2].

1. Essayons néanmoins de donner deux exemples, l'un issu de la physique, l'autre de l'économie. Supposons tout d'abord que l'on cherche à déterminer le volume V occupé par n molécules d'un **gaz de van der Waals** de température T et de pression p. L'*équation d'état* (c'est-à-dire l'équation liant les variables d'état que sont n, p, et V) d'un tel gaz s'écrit

$$\left(p + a\left(\frac{n}{V}\right)^2\right)(V - nb) = nk_\beta T,$$

où les coefficients a (pression de cohésion) et b (*covolume*) dépendent de la nature du gaz et k_β désigne la *constante de Boltzmann*. On est donc amené à résoudre une équation non linéaire d'inconnue V et de fonction

$$f(V) = \left(p + a\left(\frac{n}{V}\right)^2\right)(V - nb) - nk_\beta T = 0.$$

Admettons maintenant que l'on souhaite calculer le **taux de rendement annuel moyen** R d'un fonds de placement et que l'on se retrouve après n années avec un capital d'un montant de M euros. La relation liant M, n, R et V est

$$M = V\sum_{k=1}^{n}(1+R)^k = V\frac{1+R}{R}\big((1+R)^n - 1\big),$$

et on doit alors trouver R tel que

$$f(R) = M - V\sum_{k=1}^{n}(1+R)^k = M - V\frac{1+R}{R}\big((1+R)^n - 1\big) = 0.$$

2. Même dans le cas d'une **équation algébrique**, on rappelle qu'il n'existe pas de méthode de résolution générale à partir du degré cinq.

Toutes les méthodes que nous allons présenter sont *itératives* et consistent donc en la construction d'une suite réelle $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ qui, on l'espère, sera telle que

$$\lim_{k\to+\infty} x^{(k)} = \xi. \tag{2.1}$$

En effet, à la différence du cas des systèmes linéaires, la convergence de ces méthodes itératives dépend en général du choix de la donnée initiale $x^{(0)}$. On verra ainsi qu'on ne sait souvent qu'établir des résultats de *convergence locale*, valables lorsque $x^{(0)}$ appartient à un certain voisinage du zéro ξ.

Après avoir caractérisé la convergence des suites engendrées par des méthodes itératives, en introduisant notamment la notion d'*ordre de convergence*, nous présentons deux méthodes dites **d'encadrement**, à savoir la méthode de *dichotomie* et celle *de la fausse position*. Des méthodes adaptées au cas particulier des **équations algébriques** (c'est-à-dire polynomiales) sont brièvement abordées dans la sous-section 2.1.4. Pour terminer ce chapitre, nous considérerons, dans la section 2.2, le calcul des zéros d'une *fonction d'une variable complexe* et le cas des systèmes d'équations non linéaires.

Signalons que nous nous sommes surtout référés à [Guillaume2009] pour constituer les notes concernant le présent chapitre.

2.1 Cas d'une fonction réelle d'une variable réelle

2.1.1 Comptage et localisation des zéros : la méthode de Sturm

Dans le cas d'une fonction réelle d'une variable réelle, une bonne pratique est de commencer par faire un graphe de la fonction pour avoir une idée sur le nombre de zéros et leur *localisation*. On peut évidemment aussi étudier cette fonction par les moyens analytiques classiques pour *localiser* son(ses) zéro(s) éventuel(s). Dans le cas d'une fonction polynomiale d'une variable réelle, le comptage des racines peut aussi être réalisé au moyen de la **méthode de Sturm**. Celle-ci est fondée sur l'utilisation des **suites de Sturm** [Alain2009]. Notons en passant qu'une suite de polynômes P_0, P_1, $\cdots$, P_s est une **suite de Sturm** sur l'intervalle réel [a,b] lorsque :

1. les racines de P_0 sont simples (ou $P_0(a)\,P_0(b) \neq 0$) ;
2. si $P_0(c) = 0$ et $c \in]a,b[$, alors $\mathbf{sign}(P_1(c)) = -\mathbf{sign}\,(P_0'(c))$;
3. si $P_k(x) = 0$, alors $P_{k-1}(x)P_{k+1}(x) < 0$, et ce pour $1 \leq k \leq s-1$;
4. $P_s(x) \neq 0$ pour tout $x \in [a,b]$.

2.1.2 Généralités

2.1.2.1 Ordre de convergence d'une méthode itérative

Afin de pouvoir évaluer à quelle « vitesse » la suite construite par une méthode itérative converge vers sa limite (ce sera souvent un des critères discriminants lors du choix d'une méthode), il nous faut introduire quelques définitions.

Définition 2.1 (ordre d'une suite convergente)**.** *Soit une suite $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ de réels convergeant vers une limite ξ. On dit que cette suite convergente est d'ordre $r \geq 1$ s'il existe deux constantes C_1 et C_2 vérifiant*
$0 < C_1 \leq C_2 < +\infty$ telles que

$$C_1 \leq \frac{\left|x^{(k+1)} - \xi\right|}{\left|x^{(k)} - \xi\right|^r} \leq C_2 \quad \forall k \geq k_0, \tag{2.2}$$

où k_0 appartient à $\mathbb{N}$.

Par extension, une méthode itérative produisant une suite convergente vérifiant les relations (2.2) sera également dite *d'ordre r*. On notera que, dans plusieurs ouvrages, on trouve l'ordre d'une suite défini uniquement par le fait qu'il existe une constante $C \geq 0$ telle que pour tout $k \geq k_0 \geq 0$, $\left|x^{(k+1)} - \xi\right| \leq C \left|x^{(k)} - \xi\right|^r$. Il faut cependant observer que cette définition n'assure pas l'unicité de r, l'ordre de convergence pouvant éventuellement être plus grand que r. On préférera donc dire dans ce cas que la suite est d'ordre r *au moins*. On remarquera aussi que, si r est égal à 1, on a nécessairement $C_2 < 1$ dans (2.2), faute de quoi la suite ne pourrait pas converger.

La définition 2.1 est très générale et n'exige pas que la suite $\left(\frac{\left|x^{(k+1)}-\xi\right|}{\left|x^{(k)}-\xi\right|}\right)_{k\in\mathbb{N}}$ admette une limite quand k tend vers l'infini. Lorsque c'est le cas, on a coutume de se servir de la définition suivante.

Définition 2.2. *Soit une suite $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ de réels convergeant vers une limite ξ. On dit que cette suite est convergente d'ordre r, avec $r > 1$, vers ξ s'il existe un réel $\mu > 0$, appelé* ***constante asymptotique d'erreur****, tel que*

$$\lim_{k\to+\infty} \frac{\left|x^{(k+1)} - \xi\right|}{\left|x^{(k)} - \xi\right|^r} = \mu. \tag{2.3}$$

Dans le cas particulier où $r = 1$, on dit que la suite ***converge linéairement*** *si*

$$\lim_{k\to+\infty} \frac{\left|x^{(k+1)} - \xi\right|}{\left|x^{(k)} - \xi\right|} = \mu, \text{ avec } \mu \in]0, 1[, \tag{2.4}$$

et ***super linéairement*** *(resp.* ***sous-linéairement****) si l'égalité ci-dessus est verifié avec $\mu = 0$ (resp. $\mu = 1$).*

Ajoutons que la convergence d'ordre 2 est dite *quadratique*, celle d'ordre 3 *cubique*.

Finissons en indiquant que les notions d'ordre et de constante asymptotique d'erreur ne sont pas purement théoriques et sont en relation avec le nombre de chiffres exacts obtenus dans l'approximation de ξ. Posons en effet $\delta^{(k)} = -log_{10}\left(\left|x^{(k)} - \xi\right|\right)$; $\delta^{(k)}$ est alors le nombre de chiffres significatifs décimaux exacts de $x^{(k)}$. Pour k suffisamment grand, on a

$$\delta^{(k+1)} \approx r\delta^{(k)} - \log_{10}(\mu)\cdot \tag{2.5}$$

On voit donc que si r est égal à un, on ajoute environ $-\log_{10}(\mu)$ chiffres significatifs à chaque itération. Par exemple, si $\mu = 0.999$ alors $-\log_{10}(\mu) \approx 4,34 \times 10^{-4}$ et il faudra près de 2500 itérations pour gagner une seule décimale. Par contre, si r est strictement plus grand que un, on multiplie environ par r le nombre de chiffres significatifs à chaque itération. Ceci montre clairement l'intérêt des méthodes d'ordre plus grand que un.

2.1.2.2 Critères d'arrêt

En cas de convergence, la suite $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ construite par la méthode itérative tend vers le zéro ξ quand k tend vers l'infini. Pour l'utilisation pratique d'une telle méthode, il faut introduire un *critère d'arrêt* pour interrompre le processus itératif lorsque l'approximation courante de ξ est jugé « satisfaisante ». Pour cela, on a principalement le choix entre deux types de critères (imposer un nombre maximum d'itérations constituant une troisième possibilité) : l'un basé sur l'incrément et l'autre sur le résidu.

Soit $\varepsilon > 0$ la tolérance fixée pour le calcul approché de ξ. Dans le cas d'un *contrôle de l'incrément*, les itérations s'achèvent dès que

$$\left|x^{(k+1)} - x^{(k)}\right| < \varepsilon. \tag{2.6}$$

Si l'on choisit de *contrôler le résidu*, on met fin aux itérations dès que

$$\left|f\left(x^{(k)}\right)\right| < \varepsilon. \tag{2.7}$$

Selon le cas, chacun de ces critères peut s'avérer soit trop restrictif, soit trop optimiste.

2.1.3 Méthodes d'encadrement

Cette classe de méthodes repose sur la propriété fondamentale suivante, relative à l'existence de zéros d'une application d'une variable rélle à valeurs réelles.

Théorème 2.1 (existence d'un zéro d'une fonction continue)**.** *Soit un intervalle non vide [a,b] de* $\mathbb{R}$ *et f une application continue de [a,b] dans* $\mathbb{R}$ *vérifiant* $f(a)\,f(b) < 0$*. Alors il existe* $\xi \in]a,b[$ *tel que* $f(\xi) = 0$*.*

Démonstration. Si $f(a) < 0$, on a $0 \in]f(a), f(b)[$, sinon $f(a) > 0$ et alors $0 \in]f(b), f(a)[$. Dans ces deux cas, le résultat est une conséquence du théorème des valeurs intermédiaires [3]. Ce théorème donne dans certains cas l'existence de solutions d'équations et est à la base de techniques de résolutions approchées comme la recherche dichotomique ou la bissection.

2.1.3.1 Méthode de dichotomie

La *méthode de dichotomie* (ou *méthode de la bissection*) suppose que la fonction f est continue sur un intervalle [a,b], n'admet qu'un seul zéro $\xi \in]a,b[$ et vérifie $f(a)f(b) < 0$.

Son principe est le suivant. On pose $a^{(0)} = a$, $b^{(0)} = b$; on note $x^{(0)} = \frac{1}{2}\left(a^{(0)} + b^{(0)}\right)$ le milieu de l'intervalle de départ et on évalue la fonction f en ce point. Si $f\left(x^{(0)}\right) = 0$, le point $x^{(0)}$ est le zéro de f et le problème est résolu. Sinon, si $f\left(a^{(0)}\right) f\left(x^{(0)}\right) < 0$, alors le zéro ξ est contenu dans l'intervalle $]a^{(0)}, x^{(0)}[$ alors qu'il appartient à $]x^{(0)}, b^{(0)}[$ si $f\left(x^{(0)}\right) f\left(b^{(0)}\right) < 0$. On réitère ensuite ce processus sur l'intervalle $[a^{(1)}, b^{(1)}]$, avec $a^{(1)} = a^{(0)}$ et $b^{(1)} = x^{(0)}$ dans le premier cas, ou $a^{(1)} = x^{(0)}$ et $b^{(1)} = b^{(0)}$ dans le second, et ainsi de suite...

De cette façon, on construit de manière récurrente trois suites $\left(a^{(k)}\right)_{k\in\mathbb{N}}$, $\left(b^{(k)}\right)_{k\in\mathbb{N}}$ et $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ telles que $a^{(0)} = a$, $b^{(0)} = b$ et vérifiant, pour un entier naturel k :

- $x^{(k)} = \frac{a^{(k)}+b^{(k)}}{2}$,
- $a^{(k+1)} = a^{(k)}$ et $b^{(k+1)} = x^{(k)}$ si $f\left(a^{(k)}\right) f\left(x^{(k)}\right) < 0$,
- $a^{(k+1)} = x^{(k)}$ et $b^{(k+1)} = b^{(k)}$ si $f\left(x^{(k)}\right) f\left(b^{(k)}\right) < 0$.

La figure 2.1 illustre la mise en oeuvre de la méthode. Concernant la convergence de cette méthode, on a le résultat suivant, dont la preuve est laissée en exercice.

3. Le **théorème des valeurs intermédiaires** (**TVI**) ou **de Bolzano** est un théorème important en analyse et concerne des fonctions continues sur un intervalle. Il indique que si une fonction continue sur un intervalle prend deux valeurs m et n, alors elle prend toutes les valeurs intermédiaires entre m et n. Il est souvent énoncé comme suit : *Pour toute fonction* $f : [a,b] \to \mathbb{R}$ *et tout réel u compris entre* $f(a)$ *et* $f(b)$*, il existe au moins un réel c compris entre a et b tel que* $f(c) = u$*.*

Cas particulier (**Théorème de Bolzano**)

Si $f(a)\,f(b) \leq 0$*, il existe au moins un réel* $c \in [a,b]$ *tel que* $f(c) = 0$ *(car 0 est compris entre* $f(a)$ *et* $f(b)$*)*

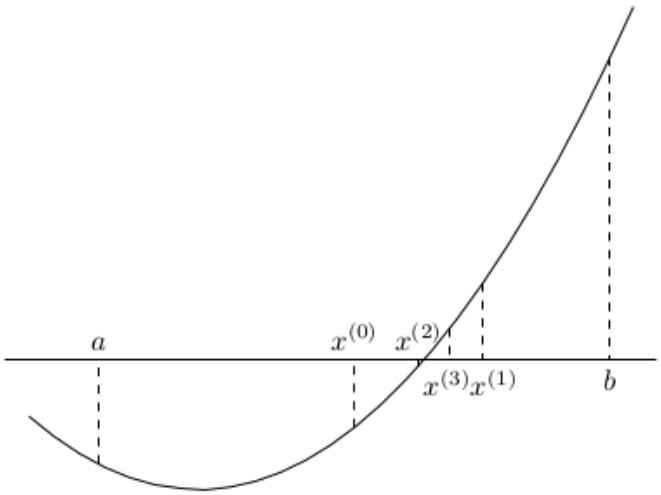


FIGURE 2.1 – Construction des premiers itérés de la méthode de dichotomie

.

Proposition 2.1. *Soit f une fonction continue sur un intervale [a,b], vérifiant* $f(a)f(b) < 0$*, et soit* $\xi \in]a, b[$ *l'unique solution de l'équation* $f(x) = 0$*. Alors, la suite* $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ *construite par la méthode de dichotomie converge vers* ξ *et on a l'estimation*

$$\left|x^{(k)} - \xi\right| \leq \frac{b-a}{2^{k+1}}, \ \forall k \in \mathbb{N}. \tag{2.8}$$

Il ressort de cette proposition que la méthode de dichotomie converge de manière certaine : c'est une méthode *globalement convergente*. L'estimation d'erreur (2.8) fournit par ailleurs directement un critère d'arrêt pour la méthode, puisque, à précision ε donnée, cette dernière permet d'approcher ξ en un nombre prévisible d'itérations. On voit en effet que, pour avoir $\left|x^{(k)} - \xi\right| \leq \varepsilon$, il faut que

$$\frac{b-a}{2^{k+1}} \leq \varepsilon \Leftrightarrow k \geq \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln(2)} - 1\cdot \tag{2.9}$$

Ainsi, pour améliorer la précision de l'approximation du zéro d'un ordre de grandeur, c'est-à-dire trouver $k > j$ tel que $\left|x^{(k)} - \xi\right| = \frac{1}{10}\left|x^{(j)} - \xi\right|$, il faut effectuer $k - j = \frac{\ln(10)}{\ln(2)} \simeq 3.32$ itérations. La convergence de cet algorithme est donc *lente*. Enfin, la méthode de dichotomie ne garantit pas une réduction monotone de l'erreur absolue d'une itération à l'autre, comme on le constate sur la figure 2.2 où la méthode de dichotomie est appliquée sur un polynôme de Legendre [4] de degré 5. Ce n'est donc pas une méthode d'ordre un au sens de la définition 2.1.

On gardera donc à l'esprit que la méthode de dichotomie est une méthode robuste permettant d'obtenir une approximation raisonable du zéro ξ pouvant servir à l'initialisation d'une méthode dont la convergence est plus rapide mais seulement *locale*.

4. Adrien-Marie Legendre (18 septembre 1752- 9 janvier 1833) était français. On lui doit d'importantes contributions en théorie des nombres, en statistique, en algèbre et en analyse, ainsi qu'en mécanique. Il est aussi célèbre pour être l'auteur des *Élément de géométrie*, un traité publié pour la première fois en 1794 reprenant et modernisant les *Éléménts* d'Euclide

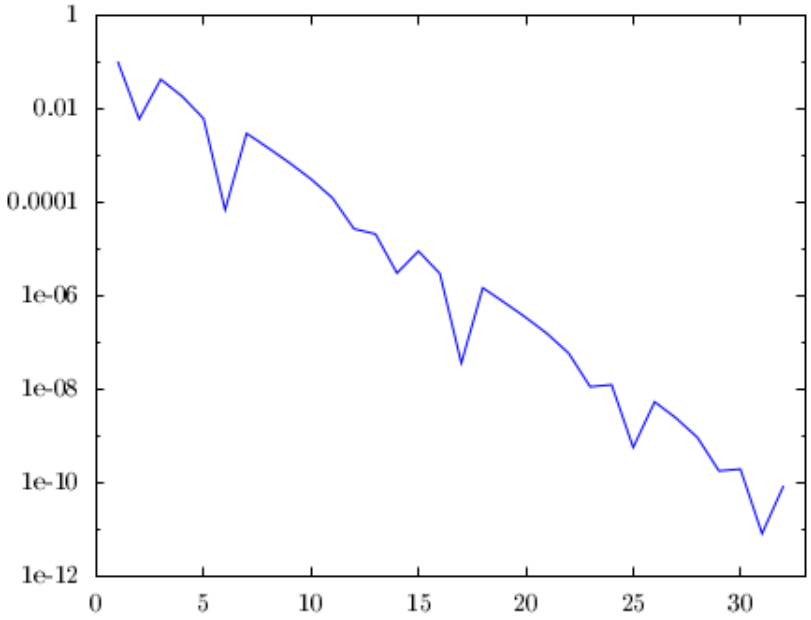


FIGURE 2.2 – Historique de la convergence, c'est-à-dire le tracé de l'erreur $\left|x^{(k)} - \xi\right|$ en fonction de k, de la méthode de dichotomie pour l'approximation de la racine $\xi = 0,9061798459...$ du polynôme de Legendre de degré 5, $P_5(x) = \frac{x}{8}\left(63x^4 - 70x^2 + 15\right)$, dont les racines se situent dans l'intevalle $]-1,1[$. On a choisi les bornes $a = 0,6$ et $b = 1$ pour l'intervalle d'encadrement initial et une précision de 10^{-10} pour le test d'arrêt, qui est atteinte après 31 itérations (à comparer à la valeur 30,89735... de l'estimation (2.9). On observe que l'erreur a un comportement oscillant, mais diminue néamoins en moyenne.

Exemple. On utilise la méthode de dichotomie pour approcher la racine du polynôme $f(x) = x^3 + 2x^2 - 3x - 1$ contenue dans l'intervalle $[1,2]$ (cette fonction est en effet continue et on a $f(1) = -1$ et $f(2) = 9$), avec une précision égale à 10^{-4}. Le tableau suivant donne les valeurs respectives des bornes $a^{(k)}$ et $b^{(k)}$ de l'intervalle d'encadrement, de l'approximation $x^{(k)}$ de la racine et de $f\left(x^{(k)}\right)$ en fonction du numéro k de l'itération.

2.1.3.2 Méthode de la fausse position

La *méthode de la fausse position*, dite encore méthode *regula falsi*, est une méthode d'encadrement combinant les possibilités de la méthode de dichotomie avec celles de la méthode de la sécante que nous ne traitons pas ici. L'idée est d'utiliser l'information fournie par les valeurs de la fonction f aux extrémités de l'intervalle d'encadrement pour remédier à la lente vitesse de convergence de la méthode de dichotomie (cette dernière ne tenant compte que du signe de la fonction). Sous des hypothèses raisonables de régularité sur f, on peut en effet montrer que la convergence de cette méthode est *linéaire*.

Comme précédemment, cette méthode suppose connus deux points a et b vérifiant $f(a)f(b) < 0$ et servant d'initialisation à la suite d'intervalles $[a^{(k)}, b^{(k)}]$, $k \geq 0$, contenant un zéro de la fonction f. Le procédé de construction des intervalles emboîtés est alors le même que pour la méthode de dichotomie, à l'exception du choix de $x^{(k)}$, qui est à présent donné par l'abscisse du point d'intersection de la droite passant par les points $\left(a^{(k)}, f\left(a^{(k)}\right)\right)$ et $\left(b^{(k)}, f\left(b^{(k)}\right)\right)$ avec l'axe des abscisses, c'est-à-dire

$$x^{(k)} = a^{(k)} - \frac{a^{(k)} - b^{(k)}}{f\left(a^{(k)}\right) - f\left(b^{(k)}\right)} f\left(a^{(k)}\right) = b^{(k)} - \frac{b^{(k)} - a^{(k)}}{f\left(b^{(k)}\right) - f\left(a^{(k)}\right)} f\left(b^{(k)}\right) = \frac{f\left(a^{(k)}\right)b^{(k)} - f\left(b^{(k)}\right)a^{(k)}}{f\left(a^{(k)}\right) - f\left(b^{(k)}\right)}. \qquad (2.10)$$

k	$a^{(k)}$	$b^{(k)}$	$x^{(k)}$	$f(x^{(k)})$
0	1	2	1,5	2,375
1	1	1,5	1,25	0,328125
2	1	1,25	1,125	−0,419922
3	1,125	1,25	1,1875	−0,067627
4	1,1875	1,25	1,21875	0,124725
5	1,1875	1,21875	1,203125	0,02718
6	1,1875	1,203125	1,195312	−0,020564
7	1,195312	1,203125	1,199219	0,003222
8	1,195312	1,199219	1,197266	−0,008692
9	1,197266	1,199219	1,198242	−0,00274
10	1,198242	1,199219	1,19873	0,000239
11	1,198242	1,19873	1,198486	−0,001251
12	1,198486	1,19873	1,198608	−0,000506
13	1,198608	1,19873	1,198669	−0,000133

FIGURE 2.3 – Recherche du zéro par la méthode de *dichotomie*.

On a représenté sur la figure 2.4 la construction des premières approximations $x^{(k)}$ ainsi trouvées. Cette méthode apparaît comme plus « flexible » que la méthode de dichotomie, le point $x^{(k)}$ ainsi construit étant plus proche de l'extrémité de l'intervalle $[a^{(k)}, b^{(k)}]$ en laquelle la valeur de la fonction $|f|$ est la plus petite. Par ailleurs, si f est une fonction linéaire, on voit que le zéro est obtenu après une itération plutôt qu'une infinité.

Indiquons que si la mesure de l'intervalle d'encadrement $[a^{(k)}, b^{(k)}]$ ainsi obtenu décroit bien lorsque k tend vers l'infini, elle ne tend pas nécessairement, à la difference de la méthode de dichotomie, vers zéro, comme l'illustre l'exemple ci-dessous.

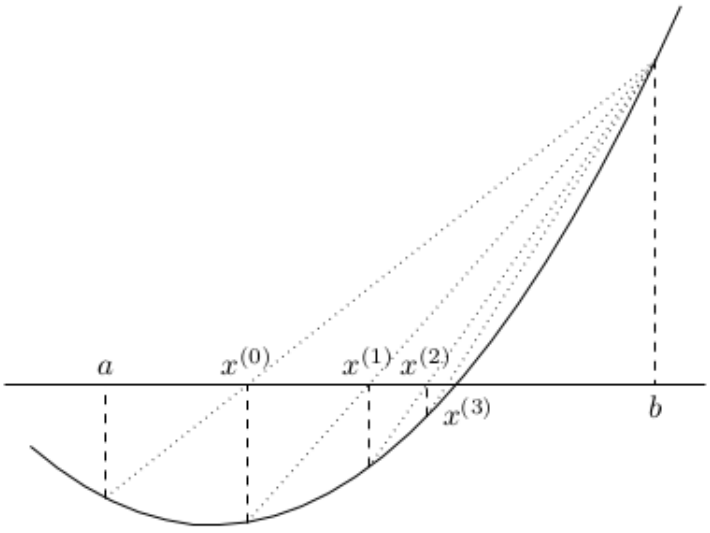


FIGURE 2.4 – Construction des premiers itérés de la méthode de la fausse position.

Exemple : On reprend l'exemple précédent en utilisant cette fois la méthode de la *fausse position*. Le tableau ci-après présente les résultats obtenus.

On observe que la borne de droite de l'intervalle d'encadrement initial est conservée tout au long

k	$a^{(k)}$	$b^{(k)}$	$x^{(k)}$	$f(x^{(k)})$
0	1	2	1,1	−0,549
1	1,1	2	1,151744	−0,274401
2	1,151744	2	1,176841	−0,130742
3	1,176841	2	1,188628	−0,060876
4	1,188628	2	1,194079	−0,028041
5	1,194079	2	1,196582	−0,012852
6	1,196582	2	1,197728	−0,005877
7	1,197728	2	1,198251	−0,002685
8	1,198251	2	1,19849	−0,001226
9	1,19849	2	1,1986	−0,00056
10	1,1986	2	1,198649	−0,000255

FIGURE 2.5 – Recherche du zéro par la méthode de la *fausse position*.

du calcul.

On notera pour terminer que le critère d'arrêt des itérations doit nécessairement être basé sur la valeur du résidu $f\left(x^{(k)}\right)$, puisque la longueur de l'intervalle d'encadrement du zéro de$f$ ne tend pas nécessairement vers zéro.

2.1.4 Méthodes pour les équations algébriques

2.1.4.1 Introduction

Dans la présente sous-section, nous considérons la résolution numérique d'équations algébriques, c'est-à-dire le cas pour lequel l'application f est un polynôme p_n de degré $n \geq 0$:

$$p_n(x) = \sum_{i=0}^{n} a_i x^i, \tag{2.11}$$

les coefficients $a_i, i = 0, \cdots, n$, étant des nombres réels données.

S'il est trivial de résoudre les équations algébriques du premier degré [5] et que la forme des solutions des équations de second dégré [6] est bien connue, il existe aussi des expressions analytiques pour les solutions des équations de degré trois et quatre, publiées par Cardano [7] en 1545 dans son *Artis Magnae, Sive de Regulis Algebraicis Liber Unus* (les formules étant respectivement dues à del Ferro [8] et Tartaglia [9] pour le troisième et à Ferrari [10] pour le quatrième). Par contre, le théorème d'Abel-Ruffini indique qu'il existe des polynômes de degré supérieur ou égal à cinq dont les racines ne s'expriment pas par radicaux. Le recours à une approche numérique se trouve par conséquent complètement motivé.

5. Ce sont les équations du type $ax + b = 0$, avec $a \neq 0$, dont la solution est donné par $x = -\frac{b}{a}$.

6. Ce sont des équations de la forme $ax^2 + bx + c = 0$, avec $a \neq 0$, dont les solutions sont données par $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

7. Girolamo Cardano (24 septembre 1501-21 septembre 1576) était un mathématicien, médecin et astrologue italien. Ses travaux en algèbre, et plus précisément ses contributions à la résolution des équations algébriques du troisième degré, eurent pour conséquence l'émergence des nombres imaginaires.

8. Scipione del Ferro (6 février 1465-5 novembre 1526) était un mathématicien italien. Il est célèbre pour avoir été le premier à trouver la méthode de résolution des équations algébriques du troisième degré sans terme quadratique.

9. Niccolò Fontana Tartaglia (vers 1499- 13 décembre 1557) était un mathématiicien italien. Il fût l'un des premiers à utiliser les mathématiques en balistique, pour l'étude des trajectoires de boulets de canon.

10. Lodovico Ferrari (2 février 1522- 5 octobre 1565) était un mathématicien italien. Élève de Cardano, il est à l'origine de la méthode de résolution des équations algébriques du quatrième degré.

2.1.4.2 Évaluation des polynômes et de leurs dérivées

Nous allons à présent décrire la *méhode de Horner*[11], qui permet l'évaluation efficace d'un polynôme et de sa dérivée en un point donné. Celle-ci repose sur le fait que tout polynôme $p_n \in \mathbb{P}_n$ peut s'écrire sous la forme

$$p_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x)\cdots)). \tag{2.12}$$

Si les formes (2.11) et (2.12) sont algébriquement équivalentes, la première nécessite n additions et $2n-1$ multiplications alors que la séconde ne requiert que n additions et n multiplications.

L'algorithme pour évaluer le polynôme p_n en un point z se résume au calcul de n constantes b_i, $i = 0, \cdots, n$, définies de la manière suivante :

$$b_n = a_n\,, \tag{2.13}$$
$$b_i = a_i + b_{i+1}z,\ i = n-1, n-2, \cdots, 0, \tag{2.14}$$

avec $b_0 = p_n(z)$.

Application. Évaluons le polynôme $7x^4 + 5x^3 - 2x^2 + 8$ au point $z = 0,5$ par la méthode de Horner. On a : $b_4 = 7$, $b_3 = 5 + 7 * 0,5 = 8,5$, $b_2 = -2 + 8,5 * 0,5 = 2,25$, $b_1 = 0 + 2,25 * 0,5 = 1,125$ et $b_0 = 8 + 1,125 * 0,5 = 8,5625$, d'où la valeur $8,5625$.

Il est à noter qu'on peut organiser ces calculs successifs de cet algorithme dans un tableau ayant pour première ligne les coefficients $a_i, i = n, n-1, \cdots, 0$, du polynôme à évaluer, et comme seconde ligne les coefficients $b_i,\ i = n, n-1, \cdots, 0$. Ainsi, chaque élément de la seconde ligne est obtenu en multipliant l'élément situé à sa gauche par z et en ajoutant au résultat l'élément situé au dessus.

Application. Pour l'exemple d'application précédent, on trouve le tableau suivant [12]

	7	5	-2	0	8
0	7	8,5	2,25	1,125	8,5625

Remarquons que les opérations employées par la méthode sont celles d'un procédé de *division synthétique*. En effet, si l'on réalise la division euclidienne de $p_n(x)$ par $(x - z)$, il vient

$$p_n(x) = (x - z)q_{n-1}(x) + r_0, \tag{2.15}$$

où le quotient $q_{n-1} \in \mathbb{P}_{n-1}$ est un polynôme dépendant de z par l'intermédiaire de ses coefficients, puisque

$$q_{n-1}(x) = \sum_{i=1}^{n} b_i x^{i-1}, \tag{2.16}$$

et le reste r_0 est une constante telle que $r_0 = b_0 = p_n(z)$. Ainsi, la méthode de Horner fournit un moyen simple d'effectuer très rapidement la division euclidienne d'un polynôme par un monôme de degré un.

Application. Effectuons la division euclidienne du polynôme $4x^3 - 7x^2 + 3x - 5$ par $x - 2$. En construisant un tableau comme précédemment, soit

	4	-7	3	-5
0	4	1	5	5

on obtient $4x^3 - 7x^2 + 3x - 5 = (x - 2)(4x^2 + x + 5) + 5$.

11. William George Horner (1786- 22 septembre 1837) était un mathématicien anglais. Il est connu pour sa méthode permettant l'approximation des racines d'un polynôme et pour l'invention en 1834 du *zootrope*, un appareil optique donnant l'illusion du mouvement.

12. Dans ce tableau, on a ajouté une première colonne contenant 0 à la deuxième ligne afin de pouvoir réaliser la même opération pour obtenir tous les coefficients $b_i, i = 0, \cdots, n$, y compris b_n.

Appliquons de nouveau la méthode pour effectuer la division du quotient q_{n-1} par $(x-z)$. On trouve

$$q_{n-1}(x) = (x-z)q_{n-2}(x) + r_1, \tag{2.17}$$

avec $q_{n-2} \in \mathbb{P}_{n-2}$ et r_1 une constante, avec

$$q_{n-2}(x) = \sum_{i=2}^{n} b_i x^{i-2} \text{ et } r_1 = c_1, \tag{2.18}$$

les coefficients $c_i, i = 1, \cdots, n$, étant définis par

$$c_n = b_n, \tag{2.19}$$
$$c_i = b_i + c_{i+1}z,\ i = n-1, n-2, \cdots, 1. \tag{2.20}$$

On a par ailleurs

$$p_n(x) = (x-z)^2 q_{n-2}(x) + r_1(x-z) + r_0, \tag{2.21}$$

et, en dérivant cette dernière égalité, on trouve que $r_1 = c_1 = p_n'(z)$. On en déduit un procédé itératif permettant d'évaluer toutes les dérivées du polynôme p_n au point z. On arrive en effet à

$$p_n(x) = r_n(x-z)^n + \cdots + r_1(x-z) + r_0, \tag{2.22}$$

après $n+1$ itérations de la méthode que l'on peut résumer dans un tableau synthétique comme on l'a déjà fait

	a_n	a_{n-1}	$\cdots$	a_2	a_1	a_0
0	b_n	b_{n-1}	$\cdots$	b_2	b_1	r_0
0	c_n	c_{n-1}	$\cdots$	c_2	r_1	
.	.	.	$\cdots$	r_2		
$\vdots$	$\vdots$	$\vdots$	$\iddots$			
.	.	r_{n-1}				
0	r_n					

(2.23)

dans lequel tous les élément n'appartenant pas à la première ligne (contenant les seuls coefficients connus initialement) ou à la première colonne sont obtenus en multipliant l'élément situé à gauche par z et en ajoutant le résultat de cette opération à l'élément situé au-dessus. Par dérivations successives de (2.22), on montre que

$$r_j = \frac{1}{j!} p_n^{(j)}(z),\ j = 0, \cdots, n, \tag{2.24}$$

où $p_n^{(j)}$ désigne la $j^{ième}$ dérivée du polynôme p_n.

Le calcul de l'ensemble du tableau (2.23) demande $\frac{1}{2}(n^2+n)$ additions et autant de multiplications.

2.1.4.3 Méthode de Newton-Horner

Compte tenu des remarques de la section précédente, on voit que la méthode de Newton peut judicieusement être adaptée pour la recherche des racines d'un polynôme, en exploitant la méthode de Horner étant donné que pour calculer le quotient apparaissant dans (2.28), c'est-à-dire

$$z^{(k+1)} = z^{(k)} - \frac{p\left(z^{(k)}\right)}{p'\left(z^{(k)}\right)}, \tag{2.25}$$

on a seulement besoin des deux premières lignes du tableau synthétique. En effet, si q_{n-1} est le polynôme associé à p_n, il vient en dérivant par rapport à x

$$p_n'(x) = q_{n-1}(x;z) + (x-z)q_{n-1}'(x;z), \tag{2.26}$$

d'où $p'_n(z) = q_{n-1}(z;z)$. Grâce à cette identité, la méthode de Newton-Horner pour l'approximation d'une racine r_j prend la forme suivante : étant donné une estimation initiale $r_j^{(0)}$ de la racine, calculer

$$r_j^{(k+1)} = r_j^{(k)} - \frac{p_n\left(r_j^{(k)}\right)}{p'_n\left(r_j^{(k)}\right)} = r_j^{(k)} - \frac{p_n\left(r_j^{(k)}\right)}{q_{n-1}\left(r_j^{(k)}; r_j^{(k)}\right)},\ k \geq 0. \tag{2.27}$$

Pour un polynôme de degré n, le coût de chaque itération de l'algorithme est égal à $4n$. Si la racine est complexe, il est nécessaire de travailler en arithmétique complexe et de prendre la donnée initiale dans $\mathbb{C}$.

2.2 Cas d'une variable complexe et des systèmes non linéaires

2.2.1 La méthode de Newton-Raphson

Rappelons que pour une équation réelle $f(x) = 0$, l'algorithme de la *méthode de Newton* peut s'écrire ainsi :

— On se donne une initialisation $x^{(0)}$, une tolérance d'arrêt ε sur la distance entre deux itérations successives et une itération maximale d'arrêt *itermax* pour les cas de convergence trop lente.

— Tant que $\Delta x > \varepsilon$ et $iter < itermax$:

- calculer $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$
- calculer $\Delta x = |x^{(n+1)} - x^{(n)}|$
- iter = iter +1

La *méthode de Newton* fonctionne tout aussi bien dans le **cadre complexe** que dans le cadre réel. La suite que l'on construit est donc formellement toujours de la forme $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}, \quad z_n \in \mathbb{C}\cdot \tag{2.28}$$

Exemple : Calculer les zéros de $f(z) = z^3 - 1$ par la *méthode de Newton.*

N.B : Les trois racines de l'unité sont 1, $j = e^{\frac{2i\pi}{3}}$ et j^2.

Pour un **système d'équations** de la forme $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ où $\mathbf{x}$ et $\mathbf{f}$ sont des vecteurs, la **méthode de Newton** peut s'écrire comme suit :

— On se donne une initialisation $\mathbf{x}_0$, une tolérance d'arrêt ε sur la distance entre deux itérations successives et une itération maximale d'arrêt *itermax* pour les cas de convergence trop lente.

— Tant que $\Delta x > \varepsilon$ et $iter < itermax$:

- Trouver $\mathbf{h}$ tel que $\mathbf{J}(\mathbf{x}_n)\mathbf{h} = \mathbf{f}(\mathbf{x}_n)$ où $\mathbf{J}$ désigne la *matrice jacobienne*
- calculer $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{h}$
- calculer $\Delta x = |\mathbf{x}_{n+1} - \mathbf{x}_n|$
- $iter = iter + 1$

Exemple : Résoudre, par la *méthode de Newton*, le système d'équations non linéaires

$$\begin{cases} x^3 + y = 1 \\ y^3 - x = -1 \end{cases}$$

Remarques :

- On vérifie que le couple (1, 0) est une solution de ce système.

- $\mathbf{J}(\mathbf{x}_n)\mathbf{h} = \mathbf{f}(\mathbf{x}_n)$ est un système d'équations algébriques en $\mathbf{h}$ pouvant être résolu à l'aide du logiciel **octave** (en tapant : $\mathbf{h} = \mathbf{J}\backslash\mathbf{f}$.

- **Matrice jacobienne**

  La matrice jacobienne est la matrice des dérivées partielles du premier ordre d'une fonction vectorielle.

  Soit $\mathbf{F}$ une fonction d'un ouvert $\mathbb{R}^n$ à valeurs dans $\mathbb{R}^m$. Une telle fonction est définie par ses m fonctions composantes à valeurs réelles :

$$\mathbf{F} : \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \longmapsto \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix}.$$

  Les dérivées partielles de ces fonctions en un point M, si elles existent, peuvent être rangées dans une matrice à m lignes et n colonnes, appelées **matrice jacobienne** de $\mathbf{F}$:

$$\mathbf{J}_F(\mathbf{M}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

- Dans le cas des systèmes de grande dimension, la méthode de Newton-Raphson s'avère assez coûteuse puisqu'il faut évaluer à chaque itération $N^2 + N$ fonctions (les N^2 dérivées partielles de la matrice jacobienne, plus les N fonctions coordonnées). De plus, il y a un système linéaire $N \times N$ (dont la matrice est en général pleine) à résoudre. Pour pallier le premier inconvénient, il est fréquent qu'on ne recalcule pas la matrice jacobienne à chaque itération, mais seulement de temps en temps. Bien sûr, à ce moment-là, la convergence n'est plus quadratique, mais en temps de calcule il arrive qu'on y gagne beaucoup, en particulier quand les dérivées de $\mathbf{f}$ sont très coûteuses à calculer.

2.2.2 La méthode de Broyden

La méthode de Broyden s'inspire directement de la remarque ci-dessus en poussant même la logique un peu plus loin. Puisqu'il est coûteux (voire même dans certains cas impossible) de calculer la matrice jacobienne de $\mathbf{f}$, on va se contenter d'en déterminer une valeur approchée à chaque itération $\mathbf{B}_n$. De plus la suite de matrices $\mathbf{B}_n$ se calculera simplement par récurrence.

On peut considérer que la **méthode de la sécante**, en dimension un, est basée un peu sur le même principe. En effet elle revient à approcher

$$f'(x_n) \quad \text{par} \quad \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

En dimension N, on va simplement imposer à la suite de matrices B_n de vérifier la même relation :

$$\mathbf{B}_n(\mathbf{x}_n - \mathbf{x}_{n-1}) = \mathbf{f}(\mathbf{x}_n) - \mathbf{f}(\mathbf{x}_{n-1}). \tag{2.29}$$

Bien sûr, la relation (2.29) ne définit pas la matrice $\mathbf{B}_n$. Elle n'impose que sa valeur dans une direction. Broyden a fait le choix simple de considérer qu'on pouvait passer de $\mathbf{B}_n$ à $\mathbf{B}_{n+1}$ en rajoutant simplement une matrice de rang 1. Ainsi, sa formule d'actualisation s'écrit,

$$\mathbf{B}_{n+1} = \mathbf{B}_n + \frac{(\delta\mathbf{f}_n - \mathbf{B}_n\delta\mathbf{x}_n)(\delta\mathbf{x}_n)^T}{||\delta\mathbf{x}_n||^2} \tag{2.30}$$

où on a noté $\delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$ et $\delta\mathbf{f}_n = \mathbf{f}(\mathbf{x}_{n+1}) - \mathbf{f}(\mathbf{x}_n)$. On vérifie immédiatement que la suite de matrices $\mathbf{B}_n$ définie par (2.30) satisfait (2.29). L'algorithme de Broyden s'écrit donc :

$$\left[\begin{array}{l} X_0 \text{ et } B_0 \text{ donnés, pour } n = 0, 1, \ldots, \text{ test d'arrêt, faire} \\ \text{Résolution du système linéaire } B_n \delta_n = -F(X_n) \\ X_{n+1} = X_n + \delta_n\,, \quad \delta F_n = F(X_{n+1}) - F(X_n) \\ B_{n+1} = B_n + \dfrac{(\delta F_n - B_n \delta_n)(\delta_n)^T}{\|\delta_n\|^2} \end{array}\right.$$

Remarques

- On peut prendre comme matrice initiale $\mathbf{B_0} = Id$; il faut évidemment attendre un certain temps avant d'avoir une approximation raisonnable de la matrice jacobienne.
- On peut démontrer qu'en général, comme pour la méthode de la sécante, la convergence est superlinéaire. La suite de matrice $\mathbf{B}_n$ ne converge pas forcément vers la matrice jacobienne de $\mathbf{f}$.

Chapitre 3

Recherche de valeurs propres

3.1 Introduction

Nous nous intéressons à la détermination des valeurs propres (le spectre) et/ou des vecteurs propres correspondants d'une matrice $\mathbf{A}$ de dimension n. Les n valeurs propres complexes λ sont les racines du polynôme caractéristique de degré n : $\det(\mathbf{A} - \lambda\mathbf{I})$. On trouve dans la littérature plusieurs méthodes de calcul du polynôme caractéristique d'une matrice carrée, entre autre la **méthode de Krylov**, la **méthode de Leverrier** et la **méthode de Faddev**.

La détermination des racines du polynôme caractéristique n'est pas efficace du point de vue numérique. Des méthodes plus adaptées sont utilisées. Celles-ci sont séparées entre deux types de méthodes :

- Les méthodes globales
  Ces méthodes permettent d'évaluer le spectre entier de la matrice. Elles utilisent une transformation de la matrice en une matrice semblable dont on calcule ensuite les éléments propres. Les principales méthodes sont :

  — La méthode de Jacobi
  — La transformation sous forme de matrice de Hessenberg
  — La méthode de Lanczos
  — La méthode de bissection
  — La méthode de Givens
  — La méthode de Rutishauser
  — La méthode de Francis

- Les méthodes partielles

  Ces méthodes visent à évaluer la plus grande ou la plus petite valeur propre ou encore la valeur propre la plus proche d'une valeur propre donnée. Les principales méthodes sont :

  — La méthode de la puissance
  — La méthode de déflation de Wielandt

Applications : Le problème aux valeurs propres émerge d'études d'oscillateurs physiques (systèmes masse-ressort, systèmes vibratoires, systèmes quantiques,$\cdots$), d'études de dynamique des structures, de flambage de poutres ainsi que d'études de stabilité des écoulements de fluides (transition laminaire/turbulent), $\cdots$.

3.2 Méthode de Jacobi

La *méthode de Jacobi* est une méthode itérative applicable à une matrice $\mathbf{A}$ symétrique. Elle revient à effectuer une suite de transformations de type rotation planaire qui permet d'annuler un élément (p, q), où p et q sont deux entiers, de la matrice $\mathbf{A}$. Chaque rotation élémentaire fait intervenir une matrice orthogonale $\mathbf{P}_{pq}$. On construit ainsi une suite de matrices symétriques $\mathbf{A}^{(k)}$ qui tend vers la matrice diagonale $\mathbf{D}$ semblable à la matrice $\mathbf{A}$, les éléments diagonaux de $\mathbf{D}$ étant les valeurs propres de $\mathbf{A}$. On pose $\mathbf{A}^{(1)} = \mathbf{A}$ et, à chaque transformation k, on construit la matrice $\mathbf{A}^{(k+1)}$ définie par :

$$\mathbf{A}^{(k+1)} = {}^t\mathbf{P}_{pq}^{(k)} \times \mathbf{A}^{(k)} \times \mathbf{P}_{pq}^{(k)}. \tag{3.1}$$

Les éléments de la matrice symétrique $\mathbf{A}^{(k+1)}$ sont donnés par :

$$\begin{cases} a_{ij}^{(k+1)} = a_{ij}^{(k)} \qquad \text{pour } i \neq p,\ q \text{ et } j \neq p,\ q \\ a_{ip}^{(k+1)} = a_{ip}^{(k)} \cos\alpha - a_{iq}^{(k)} \sin\alpha \qquad \text{pour } i \neq p,\ q \\ a_{iq}^{(k+1)} = a_{ip}^{(k)} \sin\alpha + a_{iq}^{(k)} \cos\alpha \qquad \text{pour } i \neq p,\ q \\ a_{pp}^{(k+1)} = a_{pp}^{(k)} \cos^2\alpha + a_{qq}^{(k)} \sin^2\alpha - 2a_{pq}^{(k)} \cos\alpha \sin\alpha \\ a_{qq}^{(k+1)} = a_{pp}^{(k)} \sin^2\alpha + a_{qq}^{(k)} \cos^2\alpha + 2a_{pq}^{(k)} \cos\alpha \sin\alpha \end{cases} \tag{3.2}$$

où l'angle $\alpha \in \left]-\frac{\pi}{4}, 0\right[\cup \left]0, \frac{\pi}{4}\right[$ vérifie :

$$\tan 2\alpha = \frac{2a_{pq}^{(k)}}{a_{qq}^{(k)} - a_{pp}^{(k)}} \equiv \theta \tag{3.3}$$

ou encore

$$\begin{cases} \cos(\alpha) = \sqrt{\frac{1}{2}\left(1 + \sqrt{\frac{1}{\sqrt{1+\theta^2}}}\right)} \\ \sin(\alpha) = sgn(\theta)\sqrt{\frac{1}{2}\left(1 - \sqrt{\frac{1}{\sqrt{1+\theta^2}}}\right)} \end{cases} \tag{3.4}$$

Si $a_{pp} = a_{qq}$, on choisira

$$\begin{cases} \cos(\alpha) = \frac{1}{\sqrt{2}} \\ \sin(\alpha) = \frac{sgn\left(a_{pq}^{(k)}\right)}{\sqrt{2}} \end{cases} \tag{3.5}$$

En pratique, on a le choix à chaque pas d'itération du couple (p, q). On définit différentes stratégies :

- Dans la *méthode de Jacobi classique*, on choisit (p, q) tels que
$$\left|a_{pq}^{(k)}\right| = \sup_{i \neq j} \left|a_{ij}^{(k)}\right| \tag{3.6}$$
- Dans la *méthode de Jacobi cyclique*, on effectue un balayage systématique en prenant pour (p, q) les couples $(1, 2)$, $(1, 3)$, $\cdots$, $(1, n)$ puis $(2, 3)$, $\cdots$, $(2, n)$, etc., jusqu'à $(n-1, n)$.
- Dans la *méthode de Jacobi cyclique avec seuil*, on effectue comme précédemment un balayage sur les éléments triangulaires supérieurs, chaque élément a_{ij} étant pris comme élément à annuler a_{pq}, mais on ne retient le couple (p, q) que si $|a_{ij}|$ est supérieur à un certain seuil qui peut être réajusté à chaque itération.

La méthode de Jacobi est stable, mais sa convergence est lente, ce qui en fait une méthode très peu utilisée.

3.3 Transformation en matrice de Hessenberg

Avec la transformation $\mathbf{X} = \mathbf{T} \cdot \mathbf{Y}$ où $\mathbf{T}$ est une matrice inversible, le problème $\mathbf{A} \cdot \mathbf{X} = \lambda \mathbf{X}$, $\mathbf{A}$ étant une matrice carrée quelconque, devient $(\mathbf{T}^{-1}\mathbf{AT}) \cdot \mathbf{Y} = \lambda \mathbf{Y}$. Les matrices $\mathbf{A}$ et $\mathbf{T}^{-1}\mathbf{AT}$ sont semblables. Le but est de trouver une matrice $\mathbf{T}$ telle que la matrice $\mathbf{T}^{(-1)}\mathbf{AT}$ devienne « plus simple » . Une possibilité est de chercher la matrice $\mathbf{T}$ telle que $\mathbf{T}^{-1}\mathbf{AT}$ soit une **matrice de Hessenberg H**, c'est-à-dire vérifiant $h_{ij} = 0$ pour $i > j + 1$:

$$\mathbf{T}^{-1}\mathbf{AT} = \mathbf{H} = \begin{bmatrix} * & * & \cdots & \cdots & * \\ * & * & \ddots & & \vdots \\ & * & \ddots & \ddots & * \\ & & \ddots & \ddots & * \\ & & & * & * \end{bmatrix} \tag{3.7}$$

Pour arriver à ce but, il existe plusieurs algorithmes :

- transformation élémentaires par élimination de Gauss
- transformations orthogonales (Householder, Schur, Lanczos, ...).

3.4 Méthode de Rutishauser (ou méthode LU)

La *méthode de Rutishauser* est fondée sur la décomposition LU où L est une matrice triangulaire inférieure dont les éléments diagonaux sont égaux à 1 et U une matrice triangulaire supérieure. Cette méthode a été dévéloppée dans les années 1950 par Heinz Rutishauser.

Soit $\mathbf{A}$ une matrice carrée quelconque d'ordre n. Dans la méthode de Rutishauser, une suite convergente de matrices triangulaires supérieures $\{\mathbf{A}_1\,,\ \mathbf{A}_2,\ ,\ \cdots\}$ est construite. Désignons par $\tilde{\mathbf{A}}$ la matrice vers laquelle cette suite converge. Alors, les éléments diagonaux de $\tilde{\mathbf{A}}$ tendent vers les valeurs propres de la matrice $\mathbf{A}$.

L'algorithme de Rutishauser est le suivant :

- On pose $\mathbf{A}_1 = \mathbf{A}$ puis on décompose $\mathbf{A}_1$ en $\mathbf{A}_1 = \mathbf{L}_1\mathbf{U}_1$ selon les principes de la décomposition LU.
- Connaissant les matrices $\mathbf{U}_1$ et $\mathbf{L}_1$, on forme le produit $\mathbf{A}_2 = \mathbf{U}_1\mathbf{L}_1$. Les matrices $\mathbf{A}_1$ et $\mathbf{A}_2$ sont semblables et ont donc mêmes valeurs propres car

  $$\mathbf{A}_2 = \mathbf{U}_1\mathbf{L}_1 = \mathbf{U}_1\mathbf{L}_1\mathbf{U}_1\mathbf{U}_1^{-1} = \mathbf{U}_1\mathbf{A}_1\mathbf{U}_1^{-1}\,. \tag{3.8}$$

  La matrice $\mathbf{A}_2$ est à son tour factorisée en écrivant $\mathbf{A}_2 = \mathbf{L}_2\mathbf{U}_2$, ce qui permet de construire la matrice $\mathbf{A}_3 = \mathbf{U}_2\mathbf{L}_2$ semblable à $\mathbf{A}_1$ et $\mathbf{A}_2$.
- On itère le processus, construisant ainsi une suite de matrices semblables $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\cdots$, $\mathbf{A}_k$, $\mathbf{A}_{k+1}$, $\cdots$. Cette suite converge vers une matrice triangulaire (ou quasi-triangulaire) $\tilde{\mathbf{A}}$. Si $\mathbf{A}$ a des valeurs propres réelles (ce qui est le cas si $\mathbf{A}$ est une matrice réelle symétrique ou une matrice hermitienne), alors toutes ses valeurs propres sont obtenues à partir de la diagonale principale de $\tilde{\mathbf{A}}$.

Il importe de préciser que la suite $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\cdots$, $\mathbf{A}_k$, $\mathbf{A}_{k+1}$, $\cdots$ vérifie la propriété suivante :

$$\mathbf{A}_{k+1} = \mathbf{\Lambda}_k^{-1}\mathbf{A}_1\mathbf{\Lambda}_k; \qquad \mathbf{A}_{k+1} = \mathbf{P}_k\mathbf{A}_1\mathbf{P}_k^{-1} \tag{3.9}$$

où

$$\mathbf{P}_k = \mathbf{U}_k \mathbf{U}_{k-1} \cdots \mathbf{U}_2 \mathbf{U}_1 \tag{3.10}$$

est une matrice triangulaire supérieure et

$$\mathbf{\Lambda}_k = \mathbf{L}_1 \mathbf{L}_2 \cdots \mathbf{L}_k \tag{3.11}$$

est une matrice triangulaire inférieure dont les éléments diagonaux sont égaux à 1.

En ffet,

$$\mathbf{A}_2 = \mathbf{U}_1 \mathbf{L}_1 = \mathbf{L}_1^{-1} \mathbf{L}_1 \mathbf{U}_1 \mathbf{L}_1 = \mathbf{L}_1^{-1} \mathbf{A}_1 \mathbf{L}_1 \,, \tag{3.12}$$

$$\mathbf{A}_3 = \mathbf{U}_2 \mathbf{L}_2 = \mathbf{L}_2^{-1} \mathbf{L}_2 \mathbf{U}_2 \mathbf{L}_2 = \mathbf{L}_2^{-1} \mathbf{A}_2 \mathbf{L}_2 = \mathbf{L}_2^{-1} \mathbf{L}_1^{-1} \mathbf{A}_1 \mathbf{L}_1 \mathbf{L}_2 = (\mathbf{L}_1 \mathbf{L}_2)^{-1} \mathbf{A}_1 \mathbf{L}_1 \mathbf{L}_2 \tag{3.13}$$

et ainsi de suite.

De même,

$$\mathbf{A}_2 = \mathbf{U}_1 \mathbf{L}_1 = \mathbf{U}_1 \mathbf{L}_1 \mathbf{U}_1 \mathbf{U}_1^{-1} = \mathbf{U}_1 \mathbf{A}_1 \mathbf{U}_1^{-1} \,, \tag{3.14}$$

$$\mathbf{A}_3 = \mathbf{U}_2 \mathbf{L}_2 = \mathbf{U}_2 \mathbf{L}_2 \mathbf{U}_2 \mathbf{U}_2^{-1} = \mathbf{U}_2 \mathbf{A}_2 \mathbf{U}_2^{-1} = \mathbf{U}_2 \mathbf{U}_1 \mathbf{A}_1 \mathbf{U}_1^{-1} \mathbf{U}_2^{-1} = \mathbf{U}_2 \mathbf{U}_1 \mathbf{A}_1 \left(\mathbf{U}_2 \mathbf{U}_1\right)^{-1} \tag{3.15}$$

et ainsi de suite.

Signalons, pour terminer cette section, que les vecteurs propres de $\mathbf{A}$ s'expriment en fonction des vecteurs propres de $\tilde{\mathbf{A}}$. Soit $\mathbf{V}$ la matrice des vecteurs de $\tilde{\mathbf{A}}$. La matrice $\mathbf{E}_k = \mathbf{\Lambda}_k \mathbf{V}$ converge vers la matrice des vecteurs propres de $\mathbf{A}$. Si $\mathbf{A}$ est une matrice symétrique définie positive, la méthode converge. Au-delà d'un certain indice d'itération, la convergence devient très lente : il faut un grand nombre d'itérations pour gagner en précision sur le calcul des valeurs propres. En particulier, lorsque les valeurs propres sont égales ou peu différentes, la convergence peut être très lente.

3.5 Méthode de Francis (ou méthode QR)

La *méthode de Francis* est identique à la méthode de Rutishauser à ceci près qu'elle utilise la décomposition **QR** (au lieu de la décomposition **LU**). Elle est fondée sur le théorème suivant concernant la décomposition QR d'une matrice carrée.

Théorème : Soit $\mathbf{A}$ une matrice à coefficients complexes (respectivement réels). Alors, il existe une matrice $\mathbf{Q}$ unitaire (respectivement orthogonale) et une matrice $\mathbf{R}$ triangulire supériure telle que $\mathbf{A} = \mathbf{QR}$.

La méthode QR a été introduite vers la fin des années 1950 par John G.F. Francis et par Vera N. Kublanovskaya, et ce indépendamment l'un de l'autre. Elle est probablement la méthode la plus couramment utilisée pour trouver toutes les valeurs propres d'une matrice quelconque, notamment non symétrique. L'ensemble des vecteurs propres peut être également obtenu.

A chaque itération k de la méthode de Francis (commençant par $k = 0$ si l'on pose $\mathbf{A}_0 = \mathbf{A}$), on décompose la matrice $\mathbf{A}_k$ en $\mathbf{A}_k = \mathbf{Q}_k \mathbf{R}_k$ selon les principes de la décomposition QR, avec $\mathbf{Q}^T = \mathbf{Q}_{-1}$ ou $\mathbf{Q}^\star = \mathbf{Q}^{-1}$ suivant que $\mathbf{A}$ est une matrice à coefficients réels ou à coefficients complexes respectivement. Nous construisons alors la matrice $\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k$. Comme

$$\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k = \mathbf{Q}_k^{-1} \mathbf{Q}_k \mathbf{R}_k \mathbf{Q}_k = \mathbf{Q}_k^{-1} \mathbf{A}_k \mathbf{Q}_k = \mathbf{Q}_k^T \mathbf{A}_k \mathbf{Q}_k \text{ (ou } \mathbf{Q}_k^\star \mathbf{A}_k \mathbf{Q}_k) \,, \tag{3.16}$$

tous les $\mathbf{A}_k$ sont semblables. Elles ont par conséquent mêmes valeurs propres.

L'agorithme de Francis est numériquement stable car il utilise des transformations orthogonales. Il converge plus rapidement que celui de Rutishauser et est donc préférable. Sous certaines conditions, les matrices $\mathbf{A}_k$ convergent vers une matrice triangulaire supérieure que nous notons $\tilde{\mathbf{A}}$ et qui constitue la

forme de Schur de la matrice $\mathbf{A}$. Les valeurs propres forment la diagonale principale de $\tilde{\mathbf{A}}$. La matrice $\mathbf{\Lambda}_k = \mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_k$ est la matrice des vecteurs propres de $\mathbf{A}$. Plus précisément, les colonnes de $\mathbf{\Lambda}_k$ tendent vers les vecteurs propres de $\mathbf{A}$.

3.6 Méthode de Lanczos

- Méthode itérative pour une matrice quelconque $\mathbf{A}$.
- La méthode consiste à calculer les puissances successives de $\mathbf{A}$ en construisant une suite de vecteurs orthogonaux $\mathbf{U}^{(k)}$ selon :

$$\beta_{k+1}\mathbf{U}^{(k)} = \mathbf{A}\mathbf{U}^{(k-1)} - \alpha_k\mathbf{U}^{(k-1)} - \beta_k\mathbf{U}^{(k-2)} \qquad \text{pour} k = 2, \ldots, n-1 \tag{3.17}$$

avec

$$\begin{cases} \alpha_k = {}^t\mathbf{U}^{(k-1)}\mathbf{A}\mathbf{U}^{(k-1)} \\ \beta_{k+1} = \sqrt{{}^t\mathbf{U}^{(k-1)}\mathbf{A}^2\mathbf{U}^{(k-1)} - \alpha_k^2 - \beta_k} \end{cases} \tag{3.18}$$

- Les vecteurs $U^{(0)}$, $\mathbf{U}^{(1)}$, …, $\mathbf{U}^{(n-1)}$ forment une base, appelée **base de Krylov**. Dans cette base, la matrice représentant $\mathbf{A}$, ayant les mêmes valeurs propres que $\mathbf{A}$, est tridiagonale :

$$\mathbf{A}' = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & 0 & \cdots & 0 \\ 0 & \beta_3 & \alpha_3 & \beta_4 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & \cdots & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix} \tag{3.19}$$

- La détermination des valeurs propres peut s'effectuer avec une méthode de type **QR** ou une **méthode de bissection**.
- Initialisation de l'algorithme :
  - Choix d'un vecteur initial $\mathbf{U}^{(0)}$ normalisé.
  - Calcul de α_1 :

$$\alpha_1 = {}^t\mathbf{U}^{(0)}\mathbf{A}\mathbf{U}^{(0)} \tag{3.20}$$

    et

$$\beta_2 = \sqrt{{}^t\mathbf{U}^{(0)}\mathbf{A}^2\mathbf{U}^{(0)} - \alpha_1^2} \tag{3.21}$$

  - Le vecteur $\mathbf{U}^{(1)}$ est défini par :

$$\beta_2\mathbf{U}^{(1)} = \mathbf{A}\mathbf{U}^{(0)} - \alpha_1\mathbf{U}^{(0)}. \tag{3.22}$$

3.7 Méthode de bissection

Méthode pour une matrice tridiagonale $\mathbf{A}$.

Appelons a_i, b_i et c_i respectivement les termes diagonaux, supérieurs et inférieurs de la matrice $\mathbf{A}$. La méthode consiste à calculer les valeurs propres à partir du polynôme caractéristique
$P_n(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}_n)$.

Un développement du déterminant par rapport à la dernière ligne permet d'écrire la relation de récurrence suivante :

$$P_n(\lambda) = (a_n - \lambda)P_{n-1}(\lambda) - c_{n-1}b_{n-1}P_{n-2}(\lambda) \tag{3.23}$$

avec l'initialisation $P_0(\lambda) = 1$ et $P_1(\lambda) = a_1 - \lambda$.

Le calcul des racines de $P_n(\lambda)$ s'opère de la manière suivante :

- Chercher un intervalle où $P_n(\lambda)$ change de signe
- Localiser une racine par **bissection**.

3.8 Méthode de Givens

La méthode de Givens (ou des **suites de Sturm**) permet de calculer les valeurs propres d'une matrice tridiagonale symétrique à coefficients réels.

Supposons que $\mathbf{A}$ soit mise sous la forme

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ c_1 & b_2 & c_2 & \ddots & \vdots \\ 0 & c_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & 0 & c_{n-1} & b_n \end{pmatrix} \tag{3.24}$$

et notons $\mathbf{B}_k$ la sous-matrice

$$\mathbf{B}_k = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ c_1 & b_2 & c_2 & \ddots & \vdots \\ 0 & c_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{k-1} \\ 0 & \cdots & 0 & c_{k-1} & b_k \end{pmatrix} \tag{3.25}$$

Les polynômes caractéristiques des matrices $\mathbf{B}_k$ vérifient les relations de récurrence

$$\begin{cases} p_0(\lambda) = 1\,,\ p_1(\lambda) = b_1 - \lambda \\ p_k(\lambda) = (b_k - \lambda)p_{k-1}(\lambda) - c_{k-1}^2 p_{k-2}(\lambda) \quad \text{pour } k = 2, \ldots, n \end{cases} \tag{3.26}$$

Ils vérifient les propriétés suivantes

$$\lim_{\lambda \to -\infty} p_k(\lambda) = +\infty\,. \tag{3.27}$$

Si $p_k(\lambda_0) = 0$, alors $p_{k-1}(\lambda_0)p_{k+1}(\lambda_0) < 0$ pour $k = 1, \ldots, n-1$. Le polynôme p_k a k racines réelles distinctes qui séparent les $(k+1)$ racines du polynôme p_{k+1} (i.e., $x < y < z$ avec $p_{k+1}(x) = p_{k+1}(z) = 0$ et $p_k(y) = 0$).

Soit a un réel quelconque. Si on pose

$$\text{Sgn}\,(p_k(a)) = \begin{cases} \text{sgn}\,(p_k(a)) & \text{si } p_k(a) \neq 0 \\ \text{sgn}\,(p_{k-1}(a)) & \text{si } p_k(a) = 0 \end{cases} \tag{3.28}$$

alors on démontre que le nombre $N(k, a)$ de changements de signes entre éléments de l'ensemble ordonné $\{+, \mathrm{Sgn}\,(p_1(a))\,, \ldots, \mathrm{Sgn}\,(p_k(a))\}$ est égal au nombre de racines du polynôme p_k qui sont strictement inférieurs à a.

Algorithme de Givens. Pour déterminer une valeur propre de la matrice $\mathbf{B}$, on se donne un intervalle arbitraire $[a_0,\, b_0]$ contenant λ_i. On prendra par exemple $a_0 = -b_0 = \|\mathbf{B}\|$. Soit c_0 le milieu de l'intervalle $[a_0,\, b_0]$.

$$\text{si } N(n, c_0) \geq i\,, \quad \lambda_i \in [a_0, c_0[$$

$$\text{si } N(n, c_0) < i\,, \quad \lambda_i \in [c_0, b_0]$$

On restreint alors l'intervalle de recherche à $[a_1, b_1]$ dans lequel on peut trouver λ_i. On détermine ainsi une suite d'intervalles emboîtés $[a_k, b_k]$ contenant λ_i et de longueur $(b_0 - a_0)/2^k$.

3.9 Méthode de puissance

Méthode itérative pour une matrice quelconque A qui permet d'évaluer son rayon spectral $\rho(A)$.

Le principe de la méthode repose sur le fait qu'en appliquant un grand nombre de fois la matrice sur un vecteur initial quelconque, les vecteurs successifs vont prendre une direction qui se rapproche du vecteur propre de la plus grande valeur propre (en valeur absolue).

L'algorithme :
1) Choisir un vecteur X_0.
2) Itérer : $X_{k+1} = \dfrac{AX_k}{\| AX_k \|}$ jusqu'à convergence $\| X_{k+1} - X_k \| < \varepsilon$.

La suite des vecteurs (X_k) convergent vers le vecteur propre de la plus grande valeur propre et la suite des normes $\| AX_k \|$ converge vers le rayon spectral $\rho(A)$.

Condition de convergence :
Si le vecteur de départ X_0 n'appartient pas au sous-espace vectoriel engendré par $n-1$ vecteurs propres de A alors la méthode converge.

3.10 Méthode de déflation de Wielandt

Méthode itérative pour une matrice quelconque A.

Supposons connu le rayon spectral de A, $\lambda_1 = \rho(A)$ et le vecteur propre correspondant X_1 (calculés par la méthode de la puissance). Il est possible de calculer λ_2, la valeur propre de module immédiatement inférieur à λ_1, et d'autres valeurs propres par la suite.
La méthode consiste à transformer la matrice A en une matrice $A^{(1)}$, ayant les mêmes valeurs propres que A excepté λ_1 remplacée par une valeur propre nulle, puis d'appliquer de nouveau à $A^{(1)}$ la méthode de la puissance.

L'algorithme :
1) Application de la méthode de la puissance pour déterminer λ_1 et X_1.
2) Choix d'un vecteur W tel que $< W, X_1 > = 1$.
3) Calcul de la matrice $A^{(1)} = A - \lambda_1 X_1 {}^t W$ qui a les mêmes valeurs propres que A sauf λ_1 remplacée par une valeur propre nulle.
4) Application de la méthode de la puissance à $A^{(1)}$ pour déterminer λ_2 et le vecteur propre associé Y_2 (vecteur propre de $A^{(1)}$). Le vecteur propre de A correspondant est donné par : $X_2 = Y_2 + \dfrac{\lambda_1}{\lambda_2 - \lambda_1} < W, Y_2 > X_1$

En itérant la déflation sur $A^{(1)}$, $A^{(2)}$,..., d'autres valeurs propres de la matrice A peuvent être déterminées.

3.11 Calcul du polynôme caractéristique

3.11.1 Méthode de Krylov

La méthode de Krylov utilise le théorème de Cayley-Hamilton pour calculer le polynôme caractéristique

$$P(\lambda) = (-1)^n \left(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \cdots + p_{n-1}\lambda + p_n\right). \tag{3.29}$$

Soit $\mathbf{A}$ la matrice associée au polynôme P et définie par

$$P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) = \begin{vmatrix} -(\lambda + p_1) & -p_2 & -p_3 & \cdots & -p_n \\ 1 & -\lambda & 0 & \cdots & 0 \\ 0 & 1 & -\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -\lambda \end{vmatrix} \tag{3.30}$$

si $\mathbf{b}$ est un vecteur arbitraire, on pose $\mathbf{b}_0 = \mathbf{A}^n\mathbf{b}$, $\mathbf{b}_1 = \mathbf{A}^{n-1}\mathbf{b}$,..., $\mathbf{b}_{n-1} = \mathbf{A}\mathbf{b}$, $\mathbf{b}_n = \mathbf{b}$. En appliquant $P(\mathbf{A}) = 0$, c'est-à-dire

$$\mathbf{A}^n + \sum_{i=1}^{n} p_i\mathbf{A}^{n-i} = \mathbf{0} \tag{3.31}$$

au vecteur $\mathbf{b}$, on obtient, en répétant l'opération, un système de n équations à n inconnues $(p_1, p_2, \ldots, p_n)$

$$p_1\mathbf{b}_1 + p_2\mathbf{b}_2 + \cdots + p_n\mathbf{b}_n = -\mathbf{b}_0 \tag{3.32}$$

qui se résout par une méthode de résolution de systèmes linéaires.

3.11.2 Méthode de Leverrier

La *méthode de Leverrier* utilise la trace matricielle pour calculer le polynôme caractéristique :

$$P(\lambda) = (-1)^n \left(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \cdots + p_{n-1}\lambda + p_n\right) \tag{3.33}$$

Notons $\lambda_1, \lambda_2, \ldots, \lambda_n$ les valeurs propres de P (non nécessairement distinctes).

$$P(\lambda) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)\ldots(\lambda_n - \lambda). \tag{3.34}$$

En posant, pour $k = 1, 2, \ldots, n$,

$$s_k = \mathrm{tr}\left(\mathbf{A}^k\right) = \sum_{i=1}^{n} \lambda_i^k, \tag{3.35}$$

les valeurs p_k des coefficients du polynôme caractéristique sont données par les formules de Newton :

$$\begin{cases} -p_1 = s_1 \\ -2p_2 = s_2 + p_1 s_1 \\ \cdots \\ -kp_k = s_k + p_1 s_{k-1} + \cdots + p_{k-1} s_1 \\ \cdots \\ -np_n = s_n + p_1 s_{n-1} + \cdots + p_{n-1} s_1 \end{cases} \tag{3.36}$$

qui est un système triangulaire qui se résout de proche en proche par la méthode de remontée.

Exemple : La matrice

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix} \tag{3.37}$$

a pour trace $s_1 = 6$. Les matrices

$$\mathbf{A}^2 = \begin{pmatrix} 1 & 3 & 0 \\ 0 & 4 & 0 \\ 0 & -5 & 9 \end{pmatrix}, \quad \mathbf{A}^3 = \begin{pmatrix} 1 & 7 & 0 \\ 0 & 8 & 0 \\ 0 & -19 & 27 \end{pmatrix} \tag{3.38}$$

ont pour traces $s_2 = 14$ et $s_3 = 36$. Les équations $p_1 = s_1 = -6$, $-2p_2 = s_2 + p_1 s_1 = 14 - 36$ et $-3p_3 = s_3 + p_1 s_2 + p_2 s_1 = 36 - 84 + 66$ conduisent à $p_1 = -6$, $p_2 = 11$ et $p_3 = -6$. Le polynôme caractéristique est donc

$$P(\lambda) = -\lambda^3 - p_1\lambda^2 - p_2\lambda - p_3 = -\lambda^3 + 6\lambda^2 - 11\lambda + 6. \tag{3.39}$$

3.11.3 Méthode de Faddeev

La *méthode de Faddeev*, aussi appelée *méthode de Souriau-Leverrier* utilise la trace matricielle pour calculer le polynôme caractéristique :

$$P(\lambda) = (-1)^n \left(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \cdots + p_{n-1}\lambda + p_n\right) \tag{3.40}$$

Posons

$$\begin{cases} A_1 = A \\ A_k = (A_{k-1} + p_{k-1} I)\, A \qquad k = 2,\, 3,\, \cdots,\, n\cdot \end{cases} \tag{3.41}$$

Le calcul des coefficients du polynôme caractéristique s'obtient par l'expression

$$p_k = -\frac{1}{k}\mathrm{tr}(\mathbf{A}_k). \tag{3.42}$$

Exemple : La matrice

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix}$$

a pour polynôme caractéristique

$$P(\lambda) = -\lambda^3 + 6\lambda^2 - 11\lambda + 6.$$

Retrouvons ce résultat en appliquant l'algorithme de Faddev. On calcule le coefficient p_1 à partir de la trace de la matrice $\mathbf{A}$: $p_1 = -\mathrm{tr}(\mathbf{A}) = -6$. Puis la matrice

$$\mathbf{A}_2 = \mathbf{A}^2 + p_1\mathbf{A} = \begin{pmatrix} -5 & -3 & 0 \\ 0 & -8 & 0 \\ 0 & 1 & -9 \end{pmatrix} \tag{3.43}$$

donne la valeur $p_2 = -\mathrm{tr}(\mathbf{A}_2)/2 = 11$. Le calcul de $\mathbf{A}_3$ grâce à l'expression

$$\mathbf{A}_3 = \mathbf{A}^3 + p_1\mathbf{A}^2 + p_2\mathbf{A} = \begin{pmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{pmatrix} \tag{3.44}$$

conduit à la valeur $p_3 = -\mathrm{tr}(\mathbf{A}_3)/3 = -6$. On retrouve bien l'expression du polynôme caractéristique.

3.12 Exercices

1. On considère la matrice
$$\mathbf{A} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 4 & 1 \\ 0 & -2 & 1 \end{pmatrix}$$
Calculer les valeurs propres et les vecteurs propres de cette matrice en appliquant la **méthode de Rutishauser**, puis celle de **Francis**.

2. Calculer le polynôme caractéristique de la matrice
$$\mathbf{A} = \begin{pmatrix} 1 & 3 & -1 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{pmatrix}$$
par la méthode de Krylov.

3. On considère la matrice
$$\mathbf{A} = \begin{pmatrix} -2 & 0 & 0 \\ 0 & 3 & -1 \\ 0 & -1 & 3 \end{pmatrix}$$
Calculer les valeurs propres de cette matrice par la *méthode de Givens*.

Chapitre 4

Résolution numérique de systèmes d'équations différentielles ordinaires

4.1 Présentation

Un système d'équations différentielles ordinaires (EDO) s'écrit sous forme canonique :

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{F}(\mathbf{X}, t)$$

où t est une variable réelle continue (souvent le temps, mais pas forcément), $\mathbf{X}$(t) est un vecteur de fonctions inconnues (dimension n), et $\mathbf{F}$ un vecteur de fonctions données. Pour que le problème soit « bien posé », il est nécessaire de lui adjoindre des « conditions initiales »sous la forme $\mathbf{X}(t = 0) = \mathbf{X}_0$. Ce système est du premier ordre. On peut toujours ramener un système d'ordre supérieur au premier ordre en augmentant le nombre de variables.

Si t intervient explicitement dans $\mathbf{F}$, le système est dit « non-autonome », ce qui est une difficulté numériquement. On peut là aussi éliminer t en ajoutant une variable supplémentaire $x_{n+1} = t$, ce qui donne une équation supplémentaire :

$$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}t} = 1\cdot$$

La forme la plus simple sur laquelle nous discuterons largement dans ce chapitre est

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{f}(\mathbf{y}, t)$$

où $\mathbf{f}$ englobe toute forme explicite de t et de $\mathbf{y}$.

Bien que $\mathbf{y}$ soit continue, on ne peut numériquement le connaître qu'en des points discrets. La résolution du système d'équations différentielles consiste donc à déterminer les meilleures approximations possibles d'un nombre fini de valeurs $(t_i, \mathbf{y}_i)$ où $\mathbf{y}_i = \mathbf{y}(t_i)$.

Il y a deux grandes catégories de méthodes, suivant que pour calculer $\mathbf{y}_{i+1}$ on utilise uniquement $\mathbf{y}_i$ (*méthodes à pas libre*), ou également les points $\mathbf{y}_{i-1}$, $\mathbf{y}_{i-2}$ $\cdots$ (*méthodes à pas liés*). Ces derniers ont été extrêmement populaires (classe des **méthodes d'Adams : Adams-Bashforth, Adams-Moulton**, etc...). Elles sont moins à la mode actuellement car difficiles à implémenter, lourdes à initialiser (il faut connaître $\mathbf{y}$ en plusieurs valeurs de x), et sujettes à des oscillations difficiles à contrôler dans un certain nombre de cas importants. Nous n'en parlerons pas ici en détail.

Nous insisterons donc sur les méthodes permettant de calculer $\mathbf{y}_{i+1}$ à partir de la seule connaissance de $\mathbf{y}_i$ et de $\mathbf{f}(\mathbf{y}_i, t_i)$. La plupart se regroupe sous le vaste chapeau de **Runge et Kutta**, dont la classique **RK4** n'est qu'un cas particulier.

4.2 Rappel – Solutions d'EDO simples

Nous rappelons ici les principes de résolution des équations différentiels simples. Dans cette section, la variable indépendante sera notée t et la variable dépendante $x(t)$.

ORDRE 1	SOLUTION GENERALE
$\dot{x} = ax + b$	$x(t) = C\mathrm{e}^{at} - \frac{b}{a}$
$\dot{x} = a(t)x + b(t)$	$x(t) = \mathrm{e}^{A(t)}(B(t) + C)$ avec $A(t) = \int a(t)\mathrm{d}t$ et $B(t) = \int \exp(-A(t))b(t)\mathrm{d}t$
$t\dot{x} - x = f(\dot{x})$	$x(t) = Ct + g(C)$
ORDRE 2	
Equation de Bernoulli $\dot{x} + a(t)x + b(t)x^{\alpha}$=0.	Effectuer le changement de fonction $x(t) = z(t)^{\frac{1}{1-\alpha}}$ L'équation devient $\dot{z} = (\alpha - 1)(a(t)z + b(t))$
Equation de Ricatti $\dot{x} + a(t)x + b(t)x^2 + c(t) = 0$	Effectuer le changement de fonction $z(t) = x(t) - x_p$ où x_p est une solution particulière de l'équation. On se ramène à une équation de Bernoulli avec avec $\alpha = 2$.
$\ddot{x} + a\dot{x} + bx = c$	r_1 et r_2 étant les racines de l'équation caractéristique $r^2 + ar + b = 0$, on a $x(t) = C_1\mathrm{e}^{r_1 t} + C_2\mathrm{e}^{r_2 t} + c/b$ Si $r = s + ip$ alors $x(t) = \mathrm{e}^{st}\Big(D_1\cos(pt) + D_2\sin(pt)\Big) + c/b$
Equation d'Euler $t^2\ddot{x} + at\dot{x} + bx = c(t)$	Changement de variable $t = \mathrm{e}^u$ et de fonction $x(t) = z(u)$. L'équation devient $\ddot{z} + (a-1)\dot{z} + bz = c\,(\mathrm{e}^u)$
ORDRE 3	
$\dddot{x} + a\ddot{x} + b\dot{x} + cx = 0$	r_1, r_2 et r_3 étant les racines de l'équation caractéristique $r^3 + ar^2 + br + c = 0$, on a $x(t) = C_1\mathrm{e}^{r_1 t} + C_2\mathrm{e}^{r_2 t} + C_3\mathrm{e}^{r_3 t}$.

4.3 Définitions, terminologies et exemples

4.3.1 Problème de Cauchy

■ Le problème de Cauchy [aussi appelé **problème aux valeurs initiales** ou « **initial value problem** (IVP) » en anglais] consiste à trouver la solution d'une EDO, scalaire ou vectorielle, satisfaisant les conditions initiales.

■ **Cas scalaire**

► Si I désigne un intervalle de $\mathbb{R}$ contenant le point t_0, le problème de Cauchy associé à une EDO du premier ordre s'écrit :

$$\begin{cases} y'(t) \equiv \dfrac{\mathrm{d}y}{\mathrm{d}t} = f(t, y(t)), \quad t \in \mathrm{I} \\ y(t_0) = y_0 \end{cases} \tag{4.1}$$

où $f(t, y)$ est une fonction donnée à valeur réelle définie sur le produit $S = \mathrm{I}\times]-\infty, +\infty[$ et continue par rapport aux deux variables. Si f ne dépend pas explicitement de t (i.e., $f(t, y) = f(y)$, l'équation est dite **autonome**.

► Bien que y soit continue, on ne peut numériquement le connaître qu'en des points discrets. La résolution du problème de Cauchy consiste donc à déterminer les meilleures approximations possibles d'un nombre fini de valeurs $(t_j, y(t_j))$.

► L'approximation au noeud t_j de la solution exacte $y(t_j)$ sera notée y_j ; f_j désignera la valeur $f(t_j, y_j)$.

■ **Cas vectoriel**

Soit $[t_0, t_0 + T]$ un intervalle fermé de $\mathbb{R}$, $\mathbf{f}$ une fonction continue de $[t_0, t_0 + T] \times \mathbb{R}^p$ dans $\mathbb{R}^p$, $\mathbf{y}_0$ un élément de $\mathbb{R}^p$; on cherche une fonction $\mathbf{y} \in C^1\big([t_0, t_0 + T]; \mathbb{R}^p\big)$ qui vérifie

$$\begin{cases} \mathbf{y}'(t) \equiv \dfrac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in [t_0, t_0 + T] \\ \mathbf{y}(t_0) = \mathbf{y}_0. \end{cases} \tag{4.2}$$

où $\mathbf{y}$ est un vecteur de dimension p. Ce problème est appelé problème de Cauchy pour l'équation différentielle vectorielle intervenant dans (4.2) et la condition $\mathbf{y}(t_0) = \mathbf{y}_0$ est une condition de Cauchy. Souvent, t représente le temps ; t_0 est alors appelé instant initial et $\mathbf{y}_0$ condition initiale. La donnée de $\mathbf{f}$ équivaut à la donnée de p fonctions f_1, f_2, $\cdots$, f_p continues de $[t_0, t_0+T]\times\mathbb{R}^p$ dans $\mathbb{R}$ et l'équation différentielle vectorielle est équivalente au système différentiel

$$\begin{cases} y_1'(t) = f_1(t, y_1(t), \cdots, y_p(t)) \\ \vdots \\ y_p'(t) = f_p(t, y_1(t), \cdots, y_p(t)) \end{cases} \tag{4.3}$$

Les équations précédentes sont du premier ordre car elles ne font intervenir que les dérivées premières en $\mathbf{y}$.

■ On peut plus généralement considérer des équations différentielles d'ordre supérieur, mais elles se ramènent aux équations du premier ordre. En effet, considérons l'équation différentielle

$$\mathbf{y}^{(q)} = \mathbf{f}\left(t, \mathbf{y}, \mathbf{y}'(t), \cdots, \mathbf{y}^{(q-1)}(t)\right) \tag{4.4}$$

où $\mathbf{f} \in C\big([t_0, t_0+T] \times (\mathbb{R}^p)^q; \mathbb{R}^p\big)$, et cherchons en une solution $\mathbf{y} \in C^q\big([t_0, t_0+T]; \mathbb{R}^p\big)$. Si on pose $\mathbf{z}_1 = \mathbf{y}$, $\mathbf{z}_2 = \mathbf{y}'$, $\cdots$, $\mathbf{z}_q = \mathbf{y}^{(q-1)}$, le problème (4.4) devient équivalent au système différentiel

$$\begin{cases} \mathbf{z}_1'(t) = \mathbf{z}_2(t) \\ \mathbf{z}_2'(t) = \mathbf{z}_3(t) \\ \vdots \\ \mathbf{z}_q'(t) = \mathbf{f}(t, \mathbf{z}_1(t), \cdots, \mathbf{z}_q(t)) \end{cases} \tag{4.5}$$

qui peut s'écrire sous la forme $\mathbf{z}'(t) = \mathbf{F}(t, \mathbf{z}(t))$ où $\mathbf{z} = {}^t(\mathbf{z1}, \cdots, \mathbf{z}_q)$ et
$\mathbf{F}(t, \mathbf{z}) = (\mathbf{z1}, \cdots, \mathbf{z}_q, \mathbf{f}(t, \mathbf{z1}, \cdots, \mathbf{z}_q))$,
la notation ${}^t(\cdot)$ désignant la transposition. La condition de Cauchy pour le problème (4.4) s'écrit alors

$$\mathbf{y}(t_0) = \mathbf{y}_0, \cdots, \mathbf{y}^{(q-1)}(t_0) = \mathbf{y}_0^{(q-1)}, \quad \left(\mathbf{y}_0, \cdots, \mathbf{y}_0^{(q-1)}\right) \in (\mathbb{R}^p)^q.$$

- Dans le problème de Cauchy, les conditions initiales sont données pour une seule valeur t_0 de t.
- Lorsque les conditions initiales sont données pour des valeurs distinctes de la variable indépendante t, par exemple :

  $\mathbf{y}(t_0) = \mathbf{y}_0$, $\mathbf{y}(t_1) = \mathbf{y}_1$, $\cdots$, $\mathbf{y}(t_{p-1}) = \mathbf{y}_{p-1}$,

  on obtient un **problème aux limites** (« boundary Value Problem » en Anglais).

  En analyse, un **problème aux limites** est constitué d'une *équation différentielle* (ou plus généralement *aux dérivées partielles*) dont on recherche une solution prenant de plus des valeurs imposées en des limites du domaine de résolution. Contrairement au problème analogue dit *de Cauchy*, où une ou plusieurs conditions en un même endroit sont imposées (typiquement la valeur de la solution et de ses dérivées successives en un point), auquel le *théorème de Cauchy-Lipschitz* apporte une réponse générale, les problèmes aux limites sont souvent des problèmes difficiles, et dont la résolution peut à chaque fois conduire à des considérations différentes.

  Dans le cadre des équations différentielles, une famille classique de problème aux limites est étudiée dans le cadre du **problème de Sturm-Liouville**.

  Exemple

  Dans le cas des équations aux dérivées partielles, de nombreux problèmes rentrent à la fois dans le cadre des problèmes de Cauchy du point de vue d'une variable, dans le cadre des problèmes aux limites par rapport à une autre variable. Par exemple :

  l'**équation d'une corde vibrante**, de la forme $\frac{\partial^2 y}{\partial x^2} = \frac{1}{v^2}\frac{\partial^2 y}{\partial t^2}$, où t est une variable de temps, x désigne l'abscisse des points de la corde, et y leur ordonnée dont on cherche la variation en fonction de x et t, s'accompagne d'une condition initiale, donnée par $y(x, t=0) = f(x)$ et d'une condition aux limites, la stabilité des deux points extrêmes de la corde, donnée par $y(a,t) = y(b,t) = 0$ si le domaine de définition est l'intervalle $[a, b]$.

■ Théorème de Cauchy-Lipschitz

On suppose que la fonction $\mathbf{f}$ *est continue sur* $[t_0, t_0+T] \times \mathbb{R}^p$ *et vérifie*
$\forall R > 0$, $\ni L_R$ *tel que* $\forall t \in [t_0, t_0+T]$, $\forall \mathbf{y}, \mathbf{z} \in B(\mathbf{y}_0, R)$, $||\mathbf{f}(t, \mathbf{y}) - \mathbf{f}(t, \mathbf{z})|| \leq L_R ||\mathbf{y} - \mathbf{z}||$,
où $||\cdot||$ *désigne une norme sur* $\mathbb{R}^p$ *et* $B(\mathbf{y}_0, R)$ *la boule de centre* $\mathbf{y}_0$ *et de rayon* R. *Alors, le problème de Cauchy (4.1) admet une unique solution maximale.*

- Le théorème de Cauchy-Lipschitz assure l'existence et l'unicité d'une solution au problème de Cauchy lorsque $\mathbf{f}$ est lipschitzienne par rapport à la deuxième variable $\mathbf{y}$.

4.3.2 Méthodes à un pas et méthodes multi-pas

■ Une méthode numérique pour l'approximation du problème aux valeurs initiales est dite à un pas si $\forall n \geq 0$, y_{n+1} ne dépend que de y_n. Autrement, on dit que le schéma est une méthode multi-pas (ou à pas multiples).

► Pour les **méthodes multi-pas**, la solution numérique au noeud t_{n+1} dépend de la solution aux noeuds t_k avec $k \leq n-1$.

► Une **méthode à q pas** ($q \geq 1$) est telle que, $\forall n \geq q-1$, y_{n+1} dépend directement de y_{n+1-q}, mais pas des valeurs de y_k avec $k < n+1-q$

4.3.3 Méthodes implicites et méthodes explicites

■ Une méthode est dite **explicite** si la valeur y_{n+1} peut être calculée directement à l'aide des valeurs précédentes y_k, $k \leq n$ (ou d'une partie d'entre elles).

► La Méthode d'Euler **progressive** (« forward » en anglais)

$$\begin{cases} y_0 = y(t_0) \\ y_{n+1} = y_n + h\, f(t_n, y_n), \quad n = 0,\ 1,\ \cdots,\ N_h - 1 \end{cases}$$

est une méthode explicite à un pas. C'est une méthode du premier ordre d'intérêt pédagogique, à éviter en pratique.

► La Méthode d'Adams-Bashforth d'ordre 2,

$$\begin{cases} y_0 = y(t_0) \\ y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1}), \quad n \geq 1 \end{cases}$$

est une méthode explicite à deux pas.

■ Pour une méthode **implicite**, y_{n+1} dépend aussi « de lui-même ».

► La Méthode d'Euler **rétrograde** (« backward » en anglais)

$$\begin{cases} y_0 = y(t_0) \\ y_{n+1} = y_n + h\, f(t_{n+1}, y_{n+1}), \quad n = 0,\ 1,\ \cdots,\ N_h - 1 \ (\bigstar) \end{cases}$$

est une méthode implicite à un pas. Elle est plus difficile à mettre en oeuvre, sauf si la forme de $f(t,y)$ permet le calcul analytique de y_{n+1} à partir de l'équation ($\bigstar$)

Avantage : meilleure **stabilité** que la méthode explicite.

► La Méthode d'Adams-Moulton d'ordre 4,

$$y_{n+1} = y_n + h\left(\frac{9}{24}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right),$$

est une méthode implicite à 3 pas.

4.4 Méthodes numériques à un pas

4.4.1 Formulation générale

■ Une méthode (ou schéma) **à un pas explicite** est une équation de récurrence de la forme

$$\begin{cases} \mathbf{y}_{n+1} = \mathbf{y}_n + h_n \Phi(t_n, y_n, h_n), & 0 \leq n \leq N-1, \\ t_{n+1} = t_n + h_n \end{cases}$$

■ Un schéma à un pas est dit **implicite** s'il est de la forme

$$\begin{cases} \mathbf{y}_{n+1} = \mathbf{y}_n + h_n \Phi(t_n, y_n, y_{n+1}, h_n), & 0 \leq n \leq N-1, \\ t_{n+1} = t_n + h_n \end{cases}$$

c'est-à-dire si Φ dépend non linéairement de y_{n+1}.

■ Pour les méthodes implicites, il s'agira le plus souvent de s'assurer que l'équation

$$\mathbf{y} = \mathbf{y}_n + h\Phi(t_n, \mathbf{y}_n, \mathbf{y}, h)$$

a une unique solution du moins pour h assez petit. Dans les cas les plus courants cela résultera du théorème des fonctions implicites.

4.4.2 Quelques exemples

■ Une façon d'obtenir une multitude de schémas est d'intégrer l'EDO $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$ sur l'intervalle $[t_n, t_{n+1}]$:

$$\mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) = \int_{t_n}^{t_{n+1}} \mathbf{y}'(t)\mathrm{d}t = \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\mathrm{d}t$$

et ensuite de remplacer l'intégrale définie par une approximation.

► **Intégration par la méthode des ractangles à gauche** :

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\mathrm{d}t \approx h_n \mathbf{f}(t_n, \mathbf{y}(t_n)),$$

ce qui donne la méthode d'Euler explicite

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_n, \mathbf{y}_n)$$

qui est d'**ordre 1**.

► **Intégration par la méthode des rectangles à droite** :

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\mathrm{d}t \approx h_n \mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})),$$

ce qui donne la méthode d'Euler implicite

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$$

qui est d'**ordre 2**.

▶ **Intégration par la méthode du point milieu** :

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(t,\mathbf{y}(t))\mathrm{d}t \approx h_n \mathbf{f}\left(t_n + \frac{h_n}{2}, \mathbf{y}\left(t_n + \frac{h_n}{2}\right)\right),$$

et remplacement de la solution au point $t_n + \frac{h_n}{2}$, $\mathbf{y}\left(t_n + \frac{h_n}{2}\right)$, par l'approximation obtenue au moyen de la méthode d'Euler explicite :

$$\mathbf{y}\left(t_n + \frac{h_n}{2}\right) \approx \mathbf{y}(t_n) + \frac{h_n}{2}\mathbf{f}(t_n, \mathbf{y}(t_n)),$$

ce qui donne la méthode d'Euler modifiée

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \left(t_n + \frac{h_n}{2}, \mathbf{y}_n + \frac{h_n}{2}\mathbf{f}(t_n, \mathbf{y}_n)\right)$$

qui est d'**ordre 2**.

▶ **Intégration par la méthode du trapèze** :

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(t,\mathbf{y}(t))\mathrm{d}t \approx \frac{h_n}{2}\Big(\mathbf{f}\left(t_n, \mathbf{y}(t_n)\right) + \mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1}))\Big),$$

ce qui donne la méthode du trapèze (ou **de Crank-Nicolson**

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h_n}{2}\Big(\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})\Big)$$

qui est d'**ordre 2**.

$\longrightarrow$ Eviter le caractère implicite de la méthode de Crank-Nicolson en remplaçant le $\mathbf{y}_{n+1}$ du membre de droite par l'approximation obtenue par la méthode d'Euler explicite :

$$\mathbf{y}_{n+1} \approx \mathbf{y}_n + h_n \mathbf{f}(t_n, \mathbf{y}_n),$$

ce qui donne la méthode de Heun (« improved Euler method »)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h_n}{2}\Big(\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}\left(t_{n+1}, \mathbf{y}_n + h_n \mathbf{f}(t_n, \mathbf{y}_n)\right)\Big)$$

4.4.3 Les méthodes d'Euler

4.4.3.1 Présentation des méthodes d'Euler

Les méthodes d'Euler sont les plus simples pour intégrer une équation différentielle. Cependant, ces méthodes sont rarement utilisées en raison de manque de précision et de leur instabilité. Nous avons au départ

$$\begin{cases} y'(t) = f(y(t), t) \\ y(t_0) = y_0 \end{cases} \tag{4.6}$$

Discrétisons d'abord le temps en prenant des pas de même durée h : $t_k = t_0 + kh$. Posons pour simplifier $y(t_k) = y_k$. Il s'agit désormais de trouver la suite $\{y_k\}$ définie par

$$\begin{cases} y'(t_k) = f(y_k, t_k) \\ y(t_0) = y_0 \\ t_k = t_0 + kh \end{cases} \tag{4.7}$$

Exprimons la dérivée de $y(t)$ par une différence finie (Cf. Analyse Numérique et Programmation, BAC 3 Maths et Physique, aunsi que la référence [Nyengeri2023]). Cela conduit à trois expressions

différentes suivant qu'on prend la différence avant, arrière ou centrée. Chacune possède des propriétés différentes.

$$\begin{cases} y'(t_k) = \frac{y_{k+1}-y_k}{h} + \mathcal{O}(h) & \text{différence avant} \\ y'(t_k) = \frac{y_k-y_{k-1}}{h} + \mathcal{O}(h) & \text{différence arrière} \\ y'(t_k) = \frac{y_{k+1}-y_{k-1}}{2h} + \mathcal{O}(h^2) & \text{différence centrée} \end{cases} \tag{4.8}$$

- La **méthode d'Euler explicite** s'obtient en prenant la différence avant. Cela donne une équation simple à résoudre en y_{k+1}

$$y_{k+1} = y_k + hf(y_k, t_k) + \mathcal{O}\left(h^2\right) \tag{4.9}$$

  Cette méthode, quoique simple, est peu utilisée. D'abord, elle est relativement peu précise. Mais surtout, elle est instable puisque l'erreur a généralement tendance à croître. Cette instabilité, illustrée dans la figure 4.1, peut survenir même si le pas h est très petit.

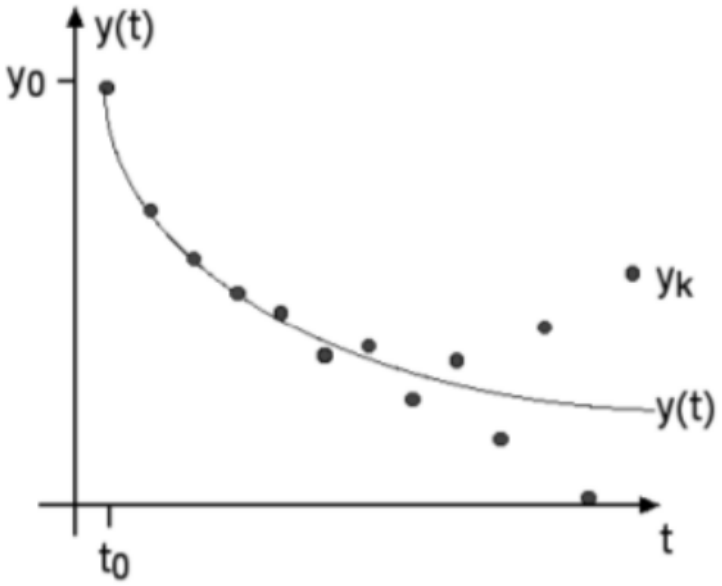


FIGURE 4.1 – Illustration d'un schéma d'intégration instable. L'erreur (=écart entre $y(t)$ et y_k croît progressivement et finit par envahir complètement.

- La **méthode d'Euler implicite** s'obtient en prenant la différence arrière. Cela donne une équation plus difficile à résoudre en y_k

$$y_k - hf(y_k, t_k) = y_{k-1} + \mathcal{O}\left(h^2\right) \tag{4.10}$$

  Cette équation n'admet pas toujours de solution analytique, ce qui est un sérieux handicap. En revanche, la méthode est stable puisque l'erreur n'a plus tendance à croître indéfiniment.

- La **méthode d'Euler centrée**, quoique plus précise, est peu utilisée car à chaque pas de temps il faut gérer à la fois y_{k+1}, y_k et y_{k-1} :

$$y_{k+1} = y_{k-1} + 2hf(y_k, t_k) + \mathcal{O}\left(h^2\right) \tag{4.11}$$

La figure 4.2 illustre l'intégration d'une équation différentielle par différentes méthodes, pour deux valeurs du pas h.

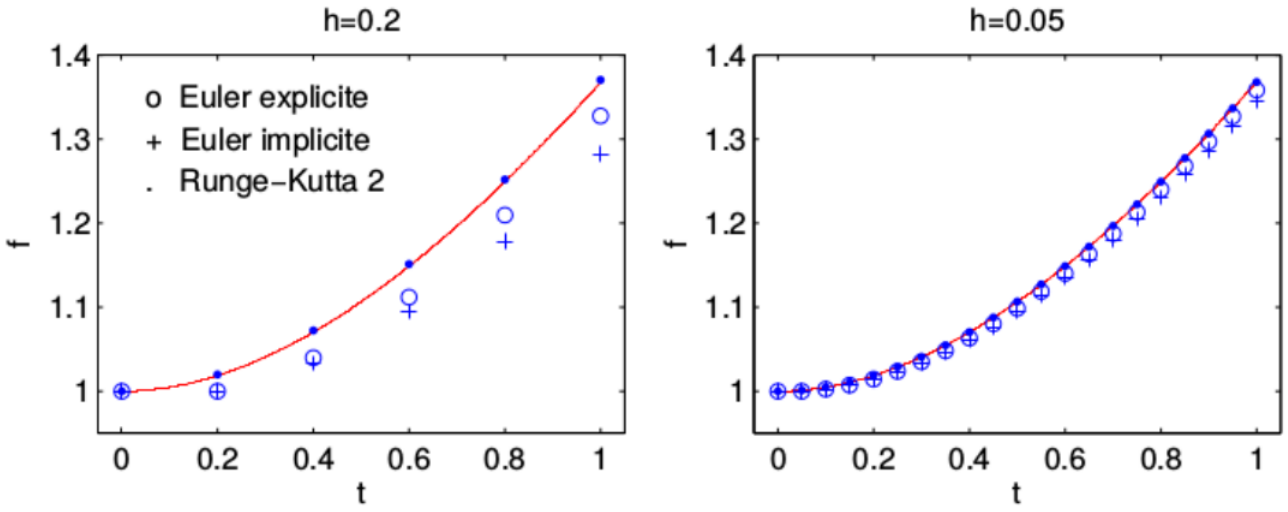


FIGURE 4.2 – Résultat de l'intégration de l'équation $y'(t) = 1 + t - y(t)$ avec $y(0) = 1$ par les méthodes d'Euler explicite, Euler implicite et de Runge-Kutta d'ordre 2. Les pas sont respectivement $h = 0.2$ (figure de gauche) et $h = 0.05$ (figure de droite). Le trait continu représente la solution exacte $y(t) = \mathrm{e}^{-t} + t$.

4.4.3.2 Stabilité des méthodes d'Euler

En plus de la précision, il est essentiel qu'une méthode d'intégration soit numériquement stable. Par stabilité on entend ici que l'intégration d'une équation dont on sait que la solution est bornée, doit elle aussi rester bornée.

L'étude de la stabilité d'une méthode numérique est une tâche complexe. L'exemple suivant permet cependant d'illustrer l'origine des instabilités. Intégrons l'équation $y' = -y$ avec la condition initiale $y_0 = 1$.

1. Avec la méthode d'Euler explicite, on obtient

$$y_{k+1} = y_k - hy_k \quad \Longrightarrow \quad y_k = (1-h)^k y_0 \tag{4.12}$$

   Cette suite converge à condition que $|1 - h| < 1$. Cela implique que soit $h > 0$ (toujours vrai) et $h < 2$. Il n'est donc pas possible de choisir des pas h arbitrairement grands, sous risque de voir la suite diverger.

2. Avec la méthode d'Euler implicite, on obtient

$$y_{k+1} = y_k - hy_{k+1} \quad \Longrightarrow \quad y_k = (1+h)^{-k} y_0. \tag{4.13}$$

   La stabilité est ici vérifiée pour tout $h > 0$. Plus généralement, on peut montrer que la méthode d'Euler implicite est toujours stable.

3. Avec la méthode d'Euler centrée, on obtient

$$y_{k+1} = y_{k-1} - 2hy_k \tag{4.14}$$

   et la condition de stabilité devient $\left|-h \pm \sqrt{1+h^2}\right| < 1$. Cette méthode peut elle aussi devenir instable.

4.4.4 Les méthodes de Runge-Kutta

■ Faire s évaluations de $\mathbf{f}$

- ► Une méthode d'ordre s si $s \leq 4$
- ► Pour atteindre l'ordre 5, six évaluations sont nécessaires

■ **Formulation générale :** Une **méthode de Runge-Kutta à s étapes** est donnée par :

$$\mathbf{k}_i = \mathbf{f}\left(t_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \quad i = 1,\ 2,\ \cdots,\ s,$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i.$$

■ **Représentation conventionnelle :** Tableau de Butcher

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \qquad c_i = \textstyle\sum_{j=1}^{s} a_{ij}\ ,\ i = 1, 2, \cdots, s.$$

- ► RK explicite si $a_{ij} = 0$ pour $j \geq i$; RK implicite dans le cas contraire
- ► méthode diagonalement implicite ou semi-implicite (DIRK) si $a_{ij} = 0$ pour $j > i$.

■ **Exemples de tableau de Butcher :**

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

Euler Explicite

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Euler I.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Heun

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Euler modifiée (RK2)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Trapèze

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

(RK3)

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

(RK4)

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Hammer & Hollingworth, ordre 4

- Les méthodes de Runge-Kutta sont couramment utilisées car elles allient précision, stabilité et simplicité. Toutes nécessitent plusieurs itérations pour effectuer un pas. La méthode la plus employée est la méthode dite **RK4** (Runge-Kutta d'ordre 4) pour son bon rapport précision/coût de calcul : En effet, l'ordre 4 est une charnière : au-delà, le nombre d'étages devient plus grand que l'ordre
- Les méthodes **RKI** ont moins de restrictions en ce qui concerne le nombre d'étages pour un ordre donné (voir la méthode à 2 étages et d'ordre 4). Mais il faut tenir compte des coûts en calcul qui sont plus importants. A cause de cela, les **MI** ne sont utilisées que lorsqu'elles apportent autre chose que la simple précision.

4.4.4.1 La méthode RK2

La *méthode de Runge-Kutta d'ordre 2* est la plus simple ; elle combine deux itérations successives de la méthode d'Euler explicite. Dans un premier temps la dérivée en (t_k, y_k) est évaluée pour faire une première estimation du point suivant (noté A dans la figure 4.3). L'estimation provisoire de y_{k+1} en ce point est ensuite utilisée pour affiner le calcul de la dérivée. Une nouvelle approximation de celle-ci est obtenue en prenant sa valeur à mi-parcours (prise au point B). C'est cette valeur de la dérivée qui sera ensuite utilisée pour estimer le prochain pas y_{k+1} (point C). La procédure d'intégration se résume à :

$$k_1 = f(t_k, y_k) \tag{4.15}$$

$$k_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}k_1\right) \tag{4.16}$$

$$y_{k+1} = y_k + hk_2 + \mathcal{O}\left(h^3\right) \tag{4.17}$$

Même si cette méthode demande deux fois plus d'opérations de calcul que la méthode d'Euler explicite pour effectuer un seul pas, le résultat est plus précis et plus stable.

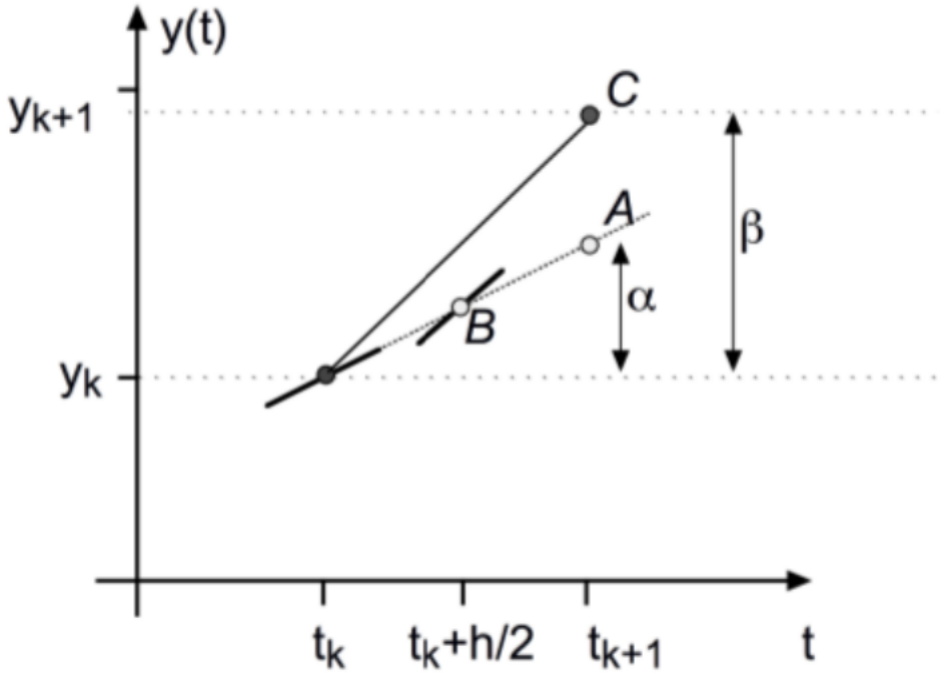


FIGURE 4.3 – Principe de fonctionnement de la méthode de Runge-Kutta du second ordre pour effectuer un seul pas. Les lettres réfèrent au texte.

4.4.4.2 La méthode RK4

La *méthode de Runge-Kutta d'ordre 4* combine quatre itérations successives. Sa précision est généralement encore meilleure. La procédure d'intégration devient

$$k_1 = f(t_k, y_k) \tag{4.18}$$

$$k_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}k_1\right) \tag{4.19}$$

$$k_3 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}k_2\right) \tag{4.20}$$

$$k_4 = f(t_k + h, y_k + hk_3) \tag{4.21}$$

$$y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}\left(h^5\right) \tag{4.22}$$

4.5 Les méthodes multi-pas

4.5.1 Formulation générale

■ Un schéma numérique à $r+1$ pas est de la forme

$$\begin{cases} \mathbf{y}_{n+1} = \Phi(\mathbf{y}_{n+1}, t_n, \mathbf{y}_n, h_n, t_{n-1}, \mathbf{y}_{n-1}, h_{n-1}, \cdots, t_{n-r}, \mathbf{y}_{n-r}, h_{n-r}), \quad r \le n \le N-1, \\ t_{n+1} = t_n + h_n \end{cases}$$

r est un entier ≥ 0.

► $\begin{cases} \mathbf{y}_{n+1} = \Phi(\mathbf{y}_{n+1}, t_n, y_n, t_{n-1}, \mathbf{y}_{n-1}, \cdots, t_{n-r}, \mathbf{y}_{n-r}, h), \quad r \le n \le N-1, \\ t_{n+1} = t_n + h \end{cases}$

lorsque le pas de discrétisation h_n est constant ($h_n = h$).

4.5.2 Exemples

■ **U**ne classe de méthode à $r+1$ pas dites linéaires à pas de discrétisation $h_n = h$ constant est définie par le schéma

$$y_{n+1} = \sum_{i=0}^{r} \alpha_i y_{n-i} + h \sum_{i=-1}^{r} \beta_i f_{n-i}, \quad 0 \le n \le N-r-1, \quad (\blacklozenge)$$

dans lequel les valeurs $f_{n-i} = f(t_{n-i}, y_{n-i})$ interviennent de manière linéaire.
$\beta_{-1} \neq 0 \implies$ méthode implicite.

■ La relation de récurrence ($\blacklozenge$) peut s'écrire sous une autre forme dite d'équation aux différences linéaire

$$\sum_{i=0}^{r+1} a_i y_{n+i} = h \sum_{i=0}^{r} b_i f_{n+i}, \quad 0 \le n \le N-r-1, \quad (\blacklozenge\blacklozenge),$$

en posant $\alpha_i = -\frac{a_{r-i}}{a_{r+i}}$ et $\beta_i = \frac{b_{r-i}}{a_{r+i}}$, où les constantes a_i, b_i sont indépendantes de n, $a_{r+1} \neq 0$ ($a_{r+1} = 1$ dans notre cas).

■ **Démarage de l'algorithme**

Si l'on considère la forme ($\blacklozenge$), le point initial (t_0, y_0) étant donné ; l'algorithme ne peut démarrer que si les valeurs $(y_1, f_1), \cdots, (y_r, f_r)$ ont déjà été calculées. Ce calcul ne peut être fait que par une méthode à un pas pour (y_1, f_1), à au plus 2 pas pour $(y_2, f_2), \cdots$, au plus r pas pour (y_r, f_r). On dit que les méthodes à pas multiples *ne sont pas auto-démarrantes*. L'initialisation des r premières valeurs (y_i, f_i), $1 \le i \le r$, sera généralement faite à l'aide d'une méthode de Runge-Kutta d'ordre supérieur ou égal à celui de la méthode ($\blacklozenge$).

■ **Remarques**

— Si $\beta_{-1} = 0$, nous pouvons obtenir directement la valeur de y_{n+1} en fonction de $y_n, \cdots, y_{n-r}$; la méthode est *explicite*.
— Si $\beta_{-1} \neq 0$, la méthode est *implicite* car elle définit y_{n+1} par une équation implicite. La méthode itérative définie dans le théorème ci-dessous permet dans ce cas d'approcher la solution de l'équation ($\blacklozenge$).

■ **Théorème**

Si $\beta_{-1} \neq 0$, l'équation (♦) admet une solution unique si $h < \frac{1}{k\beta_{-1}}$ où f est supposée k-**lipschitzienne** en y. La suite $\left(y_n^{(p)}\right)_p$ définie par la *méthode itérative*

$$y_{n+1}^{(p+1)} = h\beta_{-1} f\left(t_{n+1}, y_{n+1}^{(p)}\right) + \sum_{i=0}^{r} (\alpha_i y_{n-i} + h\beta_i f_{n-i}) \qquad (\clubsuit)$$

converge vers y_{n+1} lorsque p tend vers l'infini.

Preuve :

Si nous supposons connues les valeurs $y_{n-r}, \cdots, y_n$ et les valeurs $f_{n-r}, \cdots, f_n$, le deuxième terme du second membre de (♣) est une constante que nous noterons C. Donc y_{n+1} est une solution de l'équation

$$y = h\beta_{-1} f(t_{n+1}, y) + C.$$

En utilisant le *théorème du point fixe* et la *méthode des approximations successives* pour la fonction $\varphi \to h\beta_{-1} f(t_{n+1}, y) + C$ supposée contractante

$$\begin{aligned} |\varphi(y) - \varphi(y^\star)| &\leq h|\beta_{-1}||f(t_{n+1}, y) - f(t_{n+1}, y^\star)| \\ &\leq hk|\beta_{-1}||y - y^\star| = K|y - y^\star|, \quad K < 1 \end{aligned}$$

nous obtenos la condition $K = hk|\beta_{-1}| < 1$. En outre, on a l'estimation de l'erreur

$$\left|y_{n+1}^{(p)} - y_{n+1}\right| \leq (hk|\beta_{-1}|)^p \frac{1}{1 - hk|\beta_{-1}|} \left|y_{n+1}^{(1)} - y_{n+1}\right|$$

4.5.3 Les méthodes d'Adams

■ **Rappel :** Si y est une solution exacte de l'équation $y' = f(t, y(t))$, alors

$$y\left(t_{n+1}\right) = y\left(t_n\right) + \int_{t_n}^{t_{n+1}} f(t, y(t))\mathrm{d}t$$

■ Les **schémas d'Adams** approchent l'intégrale $\int_{t_n}^{t_{n+1}} f(t, y(t))\mathrm{d}t$ par l'intégrale d'un polynôme P interpolant f en des points donnés qui peuvent être à l'extérieur de l'intervalle $[t_n, t_{n+1}]$

■ On peut construire différents schémas selon les points d'interpolation choisis

■ Les méthodes d'Adams se divisent en deux familles :

- ► Les méthodes d'Adams-Bashforth qui sont explicites
- ► Les méthodes d'Adams-Moulton qui sont implicites

4.5.3.1 Les méthodes d'Adams-Bashforth

■ **Explicites :** pas de terme en $\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$

- **Principe :** Utiliser les k dernières valeurs $\mathbf{f}_{n-k+1}, \ , \cdots , \ \mathbf{f}_n$ afin de trouver un polynôme (de Lagrange de degré $k-1$) interpolant ces points :

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \beta h \sum_{i=1}^{k} \alpha_i \mathbf{f}\left(t_{n-i+1}, \mathbf{y}_{n-i+1}\right) \text{ avec } \beta \sum_{i=1}^{k} \alpha_i = 1$$

- **méthode d'ordre k, mais une seule évaluation du second membre à chaque pas**

k	β	α_1	α_2	α_3	α_4
1	1	1			
2	1/2	3	-1		
3	1/12	23	-16	5	
4	1/24	55	-59	37	-9

Coefficients des méthodes d'Adams-Bashforth à pas de temps fixe

4.5.3.2 Les méthodes d'Adams-Moulton

- **Implicites :** terme en $\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$

- **Utilise $k+1$ points dont t_{i+1}, donc implicite :**

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \beta h \sum_{i=0}^{k} \alpha_i \mathbf{f}\left(t_{n-i+1}, \mathbf{y}_{n-i+1}\right) \text{ avec } \beta \sum_{i=0}^{k} \alpha_i = 1$$

- **méthode d'ordre $k+1$ (nombre d'évaluations du second membre)**

k	β	α_0	α_1	α_2	α_3
0	1	1			
1	1/2	1	1		
2	1/12	5	8	-1	
3	1/24	9	19	-5	1

Coefficients des méthodes d'Adams-Moulton à pas de temps fixe

- Eviter les difficultés du caractère implicite de la méthode en utilisant un prédicteur explicite de $\mathbf{y}_{i+1}$. Injecter ensuite le résultat dans l'expression d'Adams-Moulton vue comme correcteur.

4.5.3.3 Les méthodes BDF (Backward Differentiation Formulas)

k=1	$\mathbf{y}_{n+1} - \mathbf{y}_n = h\mathbf{f}_{n+1}$ (Euler implicite)	: BDF1
k=2	$\mathbf{y}_{n+1} - \frac{4}{3}\mathbf{y}_n + \frac{1}{3}\mathbf{y}_{n-1} = \frac{2}{3}h\mathbf{f}_{n+1}$	: BDF2
k=3	$\mathbf{y}_{n+1} - \frac{18}{11}\mathbf{y}_n + \frac{9}{11}\mathbf{y}_{n-1} - \frac{2}{11}\mathbf{y}_{n-2} = \frac{6}{11}h\mathbf{f}_{n+1}$	: BDF3
k=4	$\mathbf{y}_{n+1} - \frac{48}{25}\mathbf{y}_n + \frac{36}{25}\mathbf{y}_{n-1} - \frac{16}{25}\mathbf{y}_{n-2} + \frac{3}{25}\mathbf{y}_{n-3} = \frac{12}{25}h\mathbf{f}_{n+1}$	: BDF4
k=5	$\mathbf{y}_{n+1} - \frac{300}{137}\mathbf{y}_n + \frac{300}{137}\mathbf{y}_{n-1} - \frac{200}{137}\mathbf{y}_{n-2} + \frac{75}{137}\mathbf{y}_{n-3} - \frac{12}{137}\mathbf{y}_{n-4} = \frac{60}{137}h\mathbf{f}_{n+1}$	: BDF5
k=6	$\mathbf{y}_{n+1} - \frac{360}{147}\mathbf{y}_n + \frac{450}{147}\mathbf{y}_{n-1} - \frac{400}{147}\mathbf{y}_{n-2} + \frac{225}{147}\mathbf{y}_{n-3} - \frac{72}{147}\mathbf{y}_{n-4} + \frac{10}{147}\mathbf{y}_{n-5} = \frac{60}{147}h\mathbf{f}_{n+1}$	: BDF6

- **Ces méthodes :**
  - **constituent une généralisation de la méthode d'Euler implicite,**
  - **sont implicites et d'ordre k,**
  - **ne sont stables que pour $k \leq 6$,**
  - **ont un très bon comportement avec les pbms dits « raides »,**
- L'utilisation d'un pas de temps variable complique un peu le calcul comme pour les méthodes d'Adams, mais ce calcul est bien plus simple avec les BDF.

4.5.4 Schéma prédicteur-correcteur

4.5.4.1 Principe

Il s'agit là d'une des méthodes les plus employées. Une méthode de prédiction-correction procède en deux étapes à chacune des itérations :

- **Prédiction** : On calcule une approximation de $\mathbf{y}(t_{n+1})$ notée $\bar{\mathbf{y}}^{[n+1]}$ à l'aide du schéma explicite
- **Correction** : On utilise le schéma implicite dans lequel les fonctions $\mathbf{f}$ utilisant $\mathbf{y}^{[n+1]}$ sont remplacées par les fonctions $\mathbf{f}$ utilisant $\bar{\mathbf{y}}^{[n+1]}$.

Pour $n \leftarrow 0$ à N **faire**

$\bar{\mathbf{y}}^{[n+1]} \leftarrow$ donné par un **schéma explicite**

$\mathbf{y}^{[n+1]} \leftarrow$ donné par un **schéma implicite**, inconnue $\mathbf{y}^{[n+1]}$ remplacée par $\bar{\mathbf{y}}^{[n+1]}$

Fin Pour

4.5.4.2 Exemple

Choisissons la méthode d'Euler explicite pour prédicteur et la méthode implicite des trapèzes comme correcteur.

Euler explicite : $\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + h\mathbf{f}\left(t^n, \mathbf{y}^{[n]}\right)$

Trapèze implicite : $\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{2}\left(\mathbf{f}\left(t^n, \mathbf{y}^{[n]}\right) + \mathbf{f}\left(t^{n+1}, \mathbf{y}^{[n+1]}\right)\right)$

On obtient :

$$\begin{cases} \bar{\mathbf{y}}^{[n+1]} = \mathbf{y}^{[n]} + h\mathbf{f}\left(t^n, \mathbf{y}^{[n]}\right) & \textbf{Prédiction} \\ \mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{2}\left(\mathbf{f}\left(t^n, \mathbf{y}^{[n]}\right) + \mathbf{f}\left(t^{n+1}, \bar{\mathbf{y}}^{[n+1]}\right)\right) & \textbf{Correction} \end{cases}$$

On retrouve ici, pour ce cas simple, une formule de Runge-Kutta d'ordre 2, appelée la **méthode de Heun**. En pratique, on peut utiliser un schéma d'Adams explicite pour la prédiction et un autre implicite pour la correction.

Exercice

On pose $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$. La **méthode de Adams-Bashforth d'ordre 4** explicite est donnée par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right)$$

et la **méthode de Adams-Moulton d'ordre 4** implicite par

$$\boldsymbol{y}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(9\boldsymbol{f}^{[n+1]} + 19\boldsymbol{f}^{[n]} - 5\boldsymbol{f}^{[n-1]} + \boldsymbol{f}^{[n-2]}\right)$$

avec $\boldsymbol{f}^{[n]} = \boldsymbol{f}(t^n, \boldsymbol{y}^{[n]})$.

Q. 1 *Ecrire la fonction algorithmique* REDPRECOR4VEC *permettant de résoudre un problème de Cauchy (vectoriel) par une méthode de prédiction-correction utilisant ces deux schémas. On minimisera le nombre d'appel à la fonction* $\boldsymbol{f}$ *dans la boucle principale.*

Correction de l'exercice Q.1

Q. 1 On utilise le schéma de Runge-Kutta d'ordre 4 pour initialiser les 4 premières valeurs.
Ensuite on utilise comme prédicteur le schéma explicite et comme correcteur le schéma implicite. Le principe est donc

- Calcul à l'aide du prédicteur :

$$\hat{\boldsymbol{y}}^{[n+1]} = \boldsymbol{y}^{[n]} + \frac{h}{24}\left(55\boldsymbol{f}^{[n]} - 59\boldsymbol{f}^{[n-1]} + 37\boldsymbol{f}^{[n-2]} - 9\boldsymbol{f}^{[n-3]}\right)$$

- Calcul à l'aide du correcteur :

$$\begin{aligned}\hat{\boldsymbol{f}}^{[n+1]} &= \boldsymbol{f}(t^{n+1}, \hat{\boldsymbol{y}}^{[n+1]})\\ \boldsymbol{y}^{[n+1]} &= \boldsymbol{y}^{[n]} + \tfrac{h}{24}\left(9\hat{\boldsymbol{f}}^{[n+1]} + 19\boldsymbol{f}^{[n]} - 5\boldsymbol{f}^{[n-1]} + \boldsymbol{f}^{[n-2]}\right)\end{aligned}$$

L'algorithme de la fonction REDPRECOR4VEC s'écrit alors :

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:,n) = \boldsymbol{y}^{(n-1)}$, $\forall n \in [\![1, N+1]\!]$

Fonction $[\boldsymbol{t}, \boldsymbol{Y}] \leftarrow$ REDPRECOR4VEC $(\ f, t^0, T, \boldsymbol{y}^0, N\ )$
  $\boldsymbol{t} \leftarrow DisReg(t^0, t^0+T, N)$
  $h \leftarrow (b-a)/N$
  $[\boldsymbol{t}_{ini}, \boldsymbol{Y}_{ini}] \leftarrow$ REDRK4VEC$(f, t^0, t^0 + 3*h, \boldsymbol{y}^0, 3)$
  Pour $n \leftarrow 1$ **à** 4 **faire**
    $\boldsymbol{Y}(:,n) \leftarrow \boldsymbol{Y}_{ini}(:,n)$
  Fin Pour
  $\boldsymbol{k}_1 \leftarrow \boldsymbol{f}(\boldsymbol{t}(3), \boldsymbol{Y}(:,3))$
  $\boldsymbol{k}_2 \leftarrow \boldsymbol{f}(\boldsymbol{t}(2), \boldsymbol{Y}(:,2))$
  $\boldsymbol{k}_3 \leftarrow \boldsymbol{f}(\boldsymbol{t}(1), \boldsymbol{Y}(:,1))$
  Pour $n \leftarrow 4$ **à** N **faire**
    $\boldsymbol{k}_0 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n), \boldsymbol{Y}(:,n))$
    $\hat{\boldsymbol{Y}} \leftarrow \boldsymbol{Y}(:,n) + (h/24) * (55 * \boldsymbol{k}_0 - 59 * \boldsymbol{k}_1 + 37 * \boldsymbol{k}_2 - 9 * \boldsymbol{k}_3)$
    $\hat{\boldsymbol{F}} \leftarrow \boldsymbol{f}(\boldsymbol{t}(n+1), \hat{\boldsymbol{Y}})$
    $\boldsymbol{Y}(:,n+1) \leftarrow \boldsymbol{Y}(:,n) + (h/24) * (9 * \hat{\boldsymbol{F}} + 19 * \boldsymbol{k}_0 - 5 * \boldsymbol{k}_1 + \boldsymbol{k}_2)$
    $\boldsymbol{k}_3 \leftarrow \boldsymbol{k}_2$
    $\boldsymbol{k}_2 \leftarrow \boldsymbol{k}_1$
    $\boldsymbol{k}_1 \leftarrow \boldsymbol{k}_0$
  Fin Pour
Fin Fonction

Il importe de préciser que ***DISREG*** est une fonction retournant une discrétisation régulière de l'intervalle d'étude $[a, b]$, tandis que la fonction ***REDRK4VEC*** concerne la résolution d'un problème de Cauchy par la méthode RK4. Les algorithmes en rapport avec ces fonctions sont donnés ci-bas.

Algorithme pour fonction DISREG retournant une discrétisation régulière de l'intervalle [a,b] (Cfr la référence [Cuvelier2019], page 31)

Données : a, b : deux réels, $a < b$
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$

Fonction $\boldsymbol{t} \leftarrow$ DisReg (a, b, N)
  $h \leftarrow (b-a)/N$
  Pour $n \leftarrow 0$ **à** N **faire**
    $\boldsymbol{t}(n+1) \leftarrow a + n * h$
  Fin Pour
Fin Fonction

Algorithme pour la fonction REDRK4VEC : résolution d'un problème de Cauchy par le schéma de RK4 (Cfr la référence [Cuvelier2019], page 41)

Données : $\boldsymbol{f}$: $\boldsymbol{f} : [t^0, t^0+T] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ fonction d'un problème de Cauchy (scalaire)
t^0 : réel, temps initial
T : réel > 0
$\boldsymbol{y}^0$: un vecteur de $\mathbb{R}^d$, donnée initiale
N : un entier non nul (nombre de pas de discrétisation).
Résultat : $\boldsymbol{t}$: vecteur de $\mathbb{R}^{N+1}$, $\boldsymbol{t}(n) = t^{n-1}$, $\forall n \in [\![1, N+1]\!]$
$\boldsymbol{Y}$: matrice réelle de dimension $d \times (N+1)$, $\boldsymbol{Y}(:,n) = \boldsymbol{y}^{(n-1)}$, $\forall n \in [\![1, N+1]\!]$

Fonction $[\boldsymbol{t},\boldsymbol{Y}] \leftarrow$ REDRK4Vec $(f, t^0, T, \boldsymbol{y}^0, N)$
  $\boldsymbol{t} \leftarrow DisReg(t^0, t^0+T, N)$
  $h \leftarrow (b-a)/N$
  $\boldsymbol{Y}(:,1) \leftarrow \boldsymbol{y}^0$
  Pour $n \leftarrow 1$ **à** N **faire**
    $\boldsymbol{k}_1 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n), \boldsymbol{Y}(:,n))$
    $\boldsymbol{k}_2 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h/2, \boldsymbol{Y}(:,n) + (h/2)\boldsymbol{k}_1)$
    $\boldsymbol{k}_3 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h/2, \boldsymbol{Y}(:,n) + (h/2)\boldsymbol{k}_2)$
    $\boldsymbol{k}_4 \leftarrow \boldsymbol{f}(\boldsymbol{t}(n) + h, \boldsymbol{Y}(:,n) + h\boldsymbol{k}_3)$
    $\boldsymbol{Y}(:,n+1) \leftarrow \boldsymbol{Y}(:,n) + (h/6) * (\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$
  Fin Pour
Fin Fonction

4.5.4.3 Comparaison avec Runge-Kutta

L'intérêt d'une méthode de résolution numérique d'équations différentielles se mesure principalement suivant deux critères :

- son coût pour obtenir une précision donnée (c'est-à-dire le nombre d'évaluations de fonctions par étapes).
- sa stabilité.

La caractéristique des méthodes de Runge-Kutta est que le pas est assez facile à adapter, la mise en oeuvre informatique plus aisée. Mais pour des méthodes du même ordre de consistance, les méthodes de Runge-Kutta exigent plus d'évaluations de fonctions. Quand on sait que la solution de l'équation n'est pas sujette à de brusques variations de la dérivée, on prendra une méthode de type prédicteur-correcteur mais, si le pas doit être adapté plus précisément, on préferera une méthode de Runge-Kutta.

4.6 Intégration dans la pratique

Le choix du pas est un point essentiel. S'il est trop grand, la méthode choisie, aussi bonne soit-elle, donnera des résultats erronés(Cf. figure 4.4). Si le pas est trop petit, on perdra du temps de calcul et on risquera d'être affecté par des erreurs d'arrondi. D'où les règles générales :

Le pas h doit être choisi nettement inférieur au temps caractéristique de la fonction $y(t)$ à intégrer. Par exemple, pour une fonction périodique de Période T, il faudra prendre $h << T$. Parmi les méthodes d'intégration qui existent, celles de Runge-Kutta offrent souvent un bon compromis entre précision et temps de calcul.

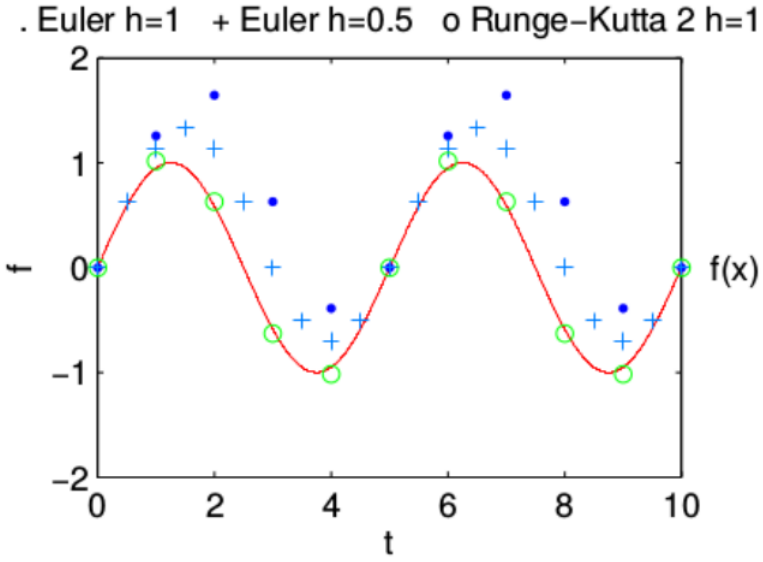


FIGURE 4.4 – Intégration d'une équation différentielle dont la solution est $y(t) = \sin(2\pi t/5)$. Cette équation a délibérément été intégrée avec des pas trop grands pour en montrer les conséquences. Les méthodes utilisées sont : Euler explicite avec un pas $h = 1$ (points), Euler explicite avec un pas $h = 0.5$ (croix), et Runge-Kutta d'ordre 2 avec un pas de $h = 1$.

Le problème de l'intégration des équations différentielles a fait et fait encore l'objet de recherches intensives. une solution intéressante consiste à adapter le pas h à l'allure de la fonction $y(t)$ pour gagner du temps. Lorsque la fonction $y(t)$ est régulière avec ses dérivées d'ordre >2 sont petites, alors on peut se contenter de choisir un pas élevé, qui sera progressivement réduit dans le voisinage de brusques variations de $y(t)$.

Les méthodes de Runge-Kutta peuvent aussi aisément se généraliser à la résolution d'équations différentielles d'ordre supérieur à 1.

4.7 Intégrer lorsque l'ordre est supérieur à 1

Les équations différentielles d'ordre supérieur à 1 peuvent aisément être transformées en des systèmes d'équations d'ordre 1, moyennant la définition de nouvelles variables.
Ainsi, le modèle de l'oscillateur harmonique amorti

$$\frac{d^2}{dt^2}f(t) + \lambda\frac{d}{dt}f(t) + \omega^2 f(t) = 0$$

peut s'écrire

$$\begin{aligned}\frac{d}{dt}f(t) &= u(t)\\ \frac{d}{dt}u(t) &= -\lambda u(t) - \omega^2 f(t)\end{aligned}$$

où $u(t)$ est la nouvelle variable, qui s'apparente à une vitesse. Cette paire d'équations couplées d'ordre 1 peut maintenant être intégrée à l'aide des méthodes discutées précédemment.

Plus généralement, toute équation différentielle d'ordre N

$$\frac{d^N f}{dx^N} + a_N(x)\frac{d^{N-1}f}{dx^{N-1}} + a_{N-1}(x)\frac{d^{N-2}f}{dx^{N-2}} + ... + a_2(x)\frac{df}{dx} + a_1(x)f(x) + a_0(x) = 0$$

peut être transformé en un système de N équations différentielles d'ordre 1, moyennant la création de $N-1$ nouvelles variables

$$\begin{aligned}\frac{d}{dx}f(x) &= u_1(x)\\ \frac{d^2}{dx^2}f(x) &= u_2(x)\\ \frac{d^3}{dx^3}f(x) &= u_3(x)\\ &\vdots\\ \frac{d^{N-1}}{dx^{N-1}}f(x) &= u_{N-1}(x)\\ \frac{d^N u_{N-1}}{dx^N} &= -a_N(x)u_{N-1}(x) - a_{N-1}(x)u_{N-2}(x) - \cdots - a_2(x)u_1(x) - a_1(x)f(x) - a_0(x)\end{aligned}$$

Les conditions initiales

$$f(x_0) = f_0, \quad \frac{df}{dx}|_{x=x_0} = f_0', \quad \frac{d^2 f}{dx^2}|_{x=x_0} = f_0'', \quad \text{etc.}$$

deviennent alors

$$f(x_0) = f_0, \quad u_1(x_0) = u_{1,0}, \quad u_2(x_0) = u_{2,0} \qquad \text{etc.}$$

Chapitre 5

Introduction à l'approximation des équations aux dérivées partielles

5.1 Introduction

L'approximation des équations aux dérivées partielles est un domaine mathématique directement lié à de nombreuses autres sciences, citons principalement : la physique et l'ingénierie. De plus, le développement de l'informatique a permis de donner un nouveau visage au calcul scientifique et donc à l'approximation des équations différentielles.

Les équations aux dérivées partielles interviennent dans de nombreux domaines de physique, qui comprennent les problèmes de diffusion, les phénomènes de propagation, ainsi que le domaine de la mécanique des fluides décrite par les équations hydrodynamiques comme celles de Navier-Stokes et l'équation de Schrödinger dépendante du temps pour la mécanique quantique. Ces équations différentielles n'ont généralement pas de solutions analytiques et une résolution numérique de ces équations est alors nécessaire.

Une équation aux dérivées partielles est une relation liant une fonction de n variables à ses dérivées partielles. L'ordre de l'équation est donné par l'ordre le plus élévé des dérivées partielles apparaissant dans l'équation. La forme générale d'une équation aux dérivées partielles linéaires est

$$\mathcal{L}[f(\mathbf{x})] = g(\mathbf{x}) \tag{5.1}$$

où $\mathbf{x}$ est un vecteur de composante $(x_1, x_2, \cdots, x_n)$ et $\mathcal{L}$ est un opérateur défini par

$$\mathcal{L} = p_0(\mathbf{x}) + \sum_{i=1}^{n} p_i(\mathbf{x})\partial_i + \sum_{i,j=1}^{n} p_{ij}(\mathbf{x})\partial_i\partial_j + \cdots \tag{5.2}$$

Si $g(\mathbf{x}) = 0$, on dit que l'équation est homogène.

5.2 Histoire et enjeux de l'approximation des équations aux dérivées partielles

5.2.1 Motivations : exemple de l'équation de la Chaleur et l'équation de Navier-Stokes

De nombreux phénomènes physiques sont représentés par des équations aux dérivées partielles. Ces équations permettent souvent le développement d'objets pour l'ingénierie : une aile d'avions, une centrale nucléaire, ... Le lien avec l'industrie est donc direct et l'obtention d'une solution précise est primordiale. Faute d'obtenir une solution analytique, une solution numérique est souvent satisfaisante.

5.2.1.1 Equation de la chaleur

L'équation de la chaleur (5.3) est une EDP parabolique bien connue. Elle est donnée par :

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = 0 \quad \text{sur } \Omega \times \mathbb{R}^+ \\ u(t=0) = u_0 \quad \text{sur } \Omega \end{cases} \tag{5.3}$$

où Ω est un domaine de $\mathbb{R}^N$.

Dans certains contextes, des solutions explicites sont données :

$$u(x,\, t) = \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^N} \exp\left(-\frac{\|x-y\|^2}{4t}\right) u_0(y) \mathrm{d}y \tag{5.4}$$

pour tous $t > 0$ et $x \in \mathbb{R}^N$.

Bien que ce théorème soit mathématiquement performant et intéressant, il ne permet que très rarement de donner une valeur numérique. La recherche d'une solution numérique par des méthodes d'approximation de l'équation aux dérivées partielles semble pouvoir répondre à cette problématique.

5.2.1.2 Equations de Navier-Stokes

En mécanique des fluides, les équations de Navier-Stokes sont des EDP non-linéaires couplées décrivant le mouvement des fluides newtoniens [1] dans l'approximation des milieux continus :

$$\frac{\partial}{\partial t} u_i + \sum_{j=1}^{N} u_j \frac{\partial u_i}{\partial x_j} = \frac{1}{R_e} \Delta u_i - \frac{\partial p}{\partial x_i} + f_i(x\,, t) \qquad \Big(x \in \mathbb{R}^N,\, t \geq 0\Big) \tag{5.5}$$

$$\mathrm{div}(u) = \sum_{i=1}^{N} \frac{\partial u_i}{\partial x_i} = 0 \qquad \Big(x \in \mathbb{R}^N,\, t \geq 0\Big) \tag{5.6}$$

$$u(x,\, 0) = u_0(x) \qquad \Big(x \in \mathbb{R}^N,\, t \geq 0\Big) \tag{5.7}$$

avec $1 \leq i \leq N$ et $R_e > 0$ [2].

La cohérence mathématique de ces équations n'est pas démontrée (voir [Fefferman1998]. Cependant, les outils numériques [3] permettent d'obtenir des solutions approchées et ainsi de modéliser des courants océaniques, des mouvements de masses d'air dans l'atmosphère, le comportement de gratte-ciels soumis au vent, ...

1. On appelle **fluide newtonien** un fluide dont la loi contrainte-vitesse de déformation est linéaire. La constante de proportionnalité est appelée viscosité. En termes usuels, cela signifie que le fluide continue de s'écouler indépendamment des forces extérieures qui agissent sur lui. Par exemple, l'eau est un fluide newtonien parce qu'elle continue d'exhiber les propriétés d'un fluide quelle que soit la vitesse à laquelle elle est agitée

2. R_e est le nombre de Reynolds ; il représente le rapport entre les forces d'inertie et les forces visqueuses et caractérise la nature du régime de l'écoulement (laminaire, transitoire et turbulent). Numériquement, un régime turbulent est difficile à manipuler (perte de stabilité).

3. Pour l'équation NSE, on utilise souvent l'exemple de la cavité carrée, régularisée ou non, pour tester un solveur numérique. O. Goyon propose une méthode dans [Goyon1996] par exemple, mais de très nombreux articles existent à ce sujet.

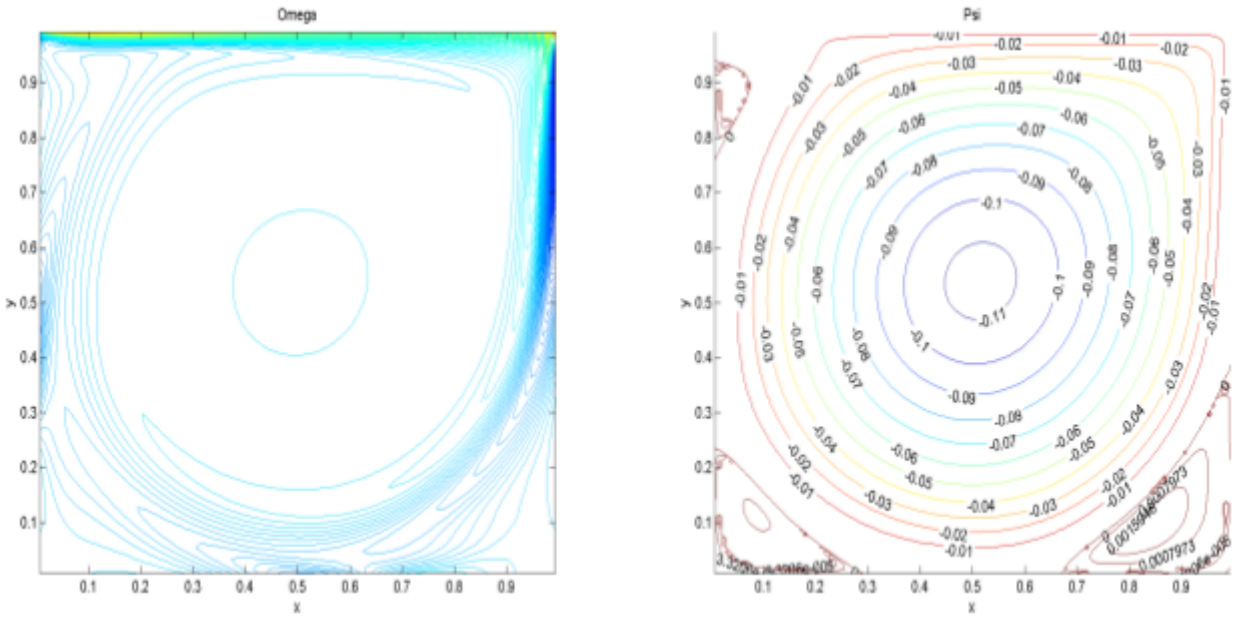


FIGURE 5.1 – Solution approchée de l'équation de Navier-Stokes incompressible $2D$ en vorticité/fonction de courant par le schéma $RSS - g \approx 1 - \tau = 1 - N = 127 - R_e = 3200 - \Delta t = 5 \times 10^{-4}$.

5.2.2 Histoire de l'approximation des EDP

5.2.2.1 Histoire des équations aux dérivées partielles

Cas des EDO

La notion d'équation différentielle apparaît chez les mathématiciens au $XVII^{\text{ième}}$ siècle. Encouragé par Huygens à étudier les mathématiques, Leibniz les invente en 1686, invention qu'il partage avec Newton. A l'origine les équations différentielles sont issues de problèmes mécaniques et géométriques.

Cependant, bien qu'en 1700 un grand nombre d'équations différentielles soient connues, il faut attendre 1739 pour qu'Euler résolve les équations linéaires du second ordre à coefficients constants. En 1866, Fuchs étudie les équations différentielles ordinaires linéaires à coefficients variables et s'intéresse aux solutions développables en séries au voisinage d'un point singulier. Briot et Bouquet, en 1855, s'attaquent aux EDO non linéaires d'ordre 1. Ces études se poursuivent par Emile Picard et Henri Poincaré ainsi que Painlevé.

En 1899, Painlevé donne des théorèmes sur les EDO non linéaires d'ordre 2. En 1902, en publiant son mémoire dans *Acta mathematica*, Painlevé indique 50 équations dont 44 intégrables explicitement. En 1909, Gambier corrige un cas oublié par Painlevé dans son mémoire.

Et pour les EDP...

La théorie de l'intégration des EDP du premier ordre date de 1734 et est écrite par Euler. En 1785, dans un mémoire de Lagrange résumant les connaissances de l'époque, on ne sait résoudre que 11 types d'EDP du premier ordre. Paul Charbit écrit un mémoire sur l'intégration de ces équations en 1784. Ce mémoire ne sera jamais publié et restera longtemps une énigme. Une copie de ce mémoire est découverte en 1928 ; on y découvre ce que l'on apelle aujourd'hui les équations différentielles caractéristiques.

En 1793, Lagrange prend connaissance du mémoire de Charbit et en 1797, il traite ces questions d'une manière plus compliquée dans *Théorie des fonctions analytiques*. En 1809, dans *Applications de*

l'analyse à la géométrie, Monge complète les travaux de Lagrange et Charbit et en 1815, Pfaff étend la méthode de Charbit ; ce procédé sera simplifié par Cauchy en 1819 puis par Jacobi en 1836. Ce sera la première méthode d'intégration d'une seule fonction inconnue à un nombre quelconque de variables indépendantes.

En ce qui concerne les EDP d'ordres supérieurs, peu de travaux existent. En 1743, D'Alembert écrit la première EDP de la mécanique :

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial u}{\partial x} - (l - s)\frac{\partial^2 u}{\partial s^2}, \tag{5.8}$$

puis suivent rapidement l'équation de Poisson suivie par celle des ondes. Laplace découvre l'équation des marées et Euler l'équation des vibrations des tiges (ordre 4 !). Cependant, les techniques du $18^{\text{ième}}$ siècle sont insuffisantes. Seule l'équation des cordes vibrantes sera résolue par D'Alembert par changement de variables.

5.2.2.2 Avant l'ordinateur : le calculateur analogique

Les calculateurs analogiques sont développés à partir des travaux de Lord Kelvin en 1870. Le principe de base est de réaliser des calculs sur une équation différentielle en substituant un ensemble de variable à un autre.

Un des premiers outils performant pour la résolution des équations différentielles est l'analyseur différentiel. Il est inventé en 1876 par James Thomson. Il s'agit d'un calculateur analogique conçu pour résoudre des équations différentielles par intégration en utilisant un système de roues et de volant.

Le premier analyseur analogique est construit par H.W. Nieman et Vannevar Bush en 1927 au **MIT** (Massachusetts Institute of Technology). Un rapport complet est publié en 1931. Une dizaine d'exemplaires sont par la suite construits en Angleterre et aux États-Unis. Au début des années 1940, ces derniers sont utilisés à la réalisation de tables balistiques d'artillerie avant l'invention de l'**ENIAC** (Electronic Numerical Integrator And Computer). Ils sont notamment utilisés au développement de la bombe rebondissante conçue pour détruire les barrages hydroélectriques allemands de la seconde guerre mondiale. Au milieu des années 1940, des analyseurs différentiels électroniques voient le jour.

En 1945, pour les besoins de l'armée, l'Université de Pennsylvanie met au point l'ENIAC qui n'est pas un ordinateur mais une calculatrice géante cadencée à 200 kHZ. Cependant ce dernier ne fut terminé que 3 mois après la fin de la guerre.

En 1945, John Von Neumann, en s'inspirant des travaux d'Alan Turing, invente l'architecture logique qui inspirera les premiers ordinateurs. En 1948, après deux ans de travail et la participation importante d'Alan Turing, le premier ordinateur est construit à Manchester : le **MARK1**. Ce dernier est réservé à des usages militaires et restera un exemplaire unique. Le premier ordinateur civil est commercialisé en 1951 et créé par Eckert et Mauchlu (chez **IBM**) le **UNIVAC1** (**UNIVersal Automatic Computer**).

5.2.2.3 Naissance de l'approximation numérique

Avec l'apparition de l'ordinateur, les méthodes numériques peuvent voir le jour. On en compte trois principales mais il en existe de nombreuses autres (on pensera notamment aux méthodes spectrales ou aux méthodes multi-grilles).

a) Les différences finies

La méthode consiste à remplacer les dérivées partielles par des différences divisées ou combinaisons de valeurs ponctuelles de la fonction en un nombre fini de points discrets ou noeuds du maillage.

Avantages : grande simplicité d'écriture et faible coût de calcul.

Inconvénients : limitation à des géométries simples, difficultés de prise en compte des *conditions aux limites de type Neumann.*

b) Les éléments finis

La méthode consiste à approcher, dans un sous-espace de dimension finie, un problème écrit sous format variationnelle (comme minimisation de l'énergie en général) dans un espace de dimension infinie. La solution approchée est dans ce cas une fonction déterminée par un nombre fini de paramètres comme, par exemple, ses valeurs en certains points ou noeuds du maillage.

Avantages : traitement possible de géométries complexes, nombreux résultats théoriques sur la convergence.

Inconvénient : complexité de mise en oeuvre et grand coût en temps de calcul et mémoire.

c) Les volumes finis

La méthode intègre, sur des volumes élémentaires de forme simple, les équations écrites sous forme de loi de conservation. Elle fournit ainsi de manière naturelle des approximations discrètes conservatives et est particulièrement bien adaptée aux équations de la mécanique des fluides. Sa mise en oeuvre est simple avec des volumes élémentaires rectangulaires.

Avantages : permet de traiter des géométries complexes avec des volumes de forme quelconque, détermination plus naturelle des *conditions aux limites de type Neumann.*

Inconvénient : peu de résultats théoriques de convergence.

5.3 Application de la méthode des différences finies à l'équation d'advection-diffusion : Utilisation du schéma semi-discret en espace et de la méthode RK4

On considère les processus d'advection et de diffusion appliqués à un champ scalaire $u(x,t)$, $x \in \mathbb{R}$, $t \in \mathbb{R}^+$. On obtient l'EDP d'ordre 2 homogène suivante :

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} - \beta \frac{\partial^2 u}{\partial x^2} = 0 \tag{5.9}$$

où $\alpha > 0$ est la vitesse (constante) d'advection et $\beta > 0$ est le coefficient (constante) de diffusion. On complète cette EDP en donnant une condition initiale sur le temps et les conditions aux limites en espace. Plus précisément, on considère le problème suivant :

$$\begin{cases} \frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} - \beta \frac{\partial^2 u}{\partial x^2} = 0, \quad x \in (-\pi, \pi), \quad t \in [0,2] \\ u(0,x) = \sin(x), \quad x \in (-\pi, \pi) \quad \text{(Condition initiale)} \\ u(t,-\pi) = \mathrm{e}^{-\beta t} \sin(\alpha t), \quad u(t,\pi) = \mathrm{e}^{-\beta t} \sin(\alpha t) \quad \text{(Conditions aux limites de Dirichlet)} \end{cases} \tag{5.10}$$

Indications :

- Discrétiser $u(t,x)$ dans l'espace en utilisant le pas Δx donné par l'équation $\Delta x = 2\pi/M$ où $M \in \mathbb{N}^+$ ($M = 600$). On propose d'utiliser la différence avant d'ordre 2 pour approximer la dérivée première et la différence centrée d'ordre 2 pour approximer la dérivée seconde.
- Tenir compte de la condition initiale et des conditions aux limites. Cela conduit à un système de $M-1$ EDO du premier ordre par rapport au temps.
- Résoudre le système d'EDO obtenu au moyen de la méthode RK4 avec un pas de discrétisation temporel $\Delta t = 1/1000$ en utilisant le logiciel Octave.
- Représenter graphiquement $u(t=2,x)$ dans le cas particulier où $\alpha = 0.1$ et $\beta = 0.05$ et comparer le résultat numérique et celui analytique. On précise que la solution analytique exacte est donnée par l'expression

$$u(t,x) = \mathrm{e}^{-\beta t}\sin(x-\alpha t)\cdot \tag{5.11}$$

Solution :

Discrétisons $u(t,x)$ dans l'espace avec la taille de la grille $\Delta x = \frac{2\pi}{M}$. Nous utilisons les schémas de différence avant d'ordre 2 et de différence centrée d'ordre 2 pour discrétiser le terme d'advection et le terme de diffusion. Nous obtenons alors un système de $M-1$ équations différentielles ordinaires d'ordre 1 dans le temps avec les conditions aux limites données par la solution exacte (5.11). Nous avons :

$$x_i = i\frac{2\pi}{M}, \quad \frac{-M}{2}+1 \le i \le \frac{M}{2}-1, \ \Delta x = \frac{2\pi}{M} \tag{5.12}$$

et nous obtenons à partir de (5.10) et (5.11) le système de dimension $(M-1)$ suivant :

$$\begin{cases} \frac{du_i}{dt} = -\alpha\frac{-u_{i+2}+4u_{i+1}-3u_i}{2\Delta x} + \beta\frac{u_{i+1}-2u_i+u_{i-1}}{(\Delta x)^2}, \ -N+1 \le i \le N-2 \\ \frac{du_{N-1}}{dt} = -\alpha\frac{u_N-u_{N-2}}{2\Delta x} + \beta\frac{u_N-2u_{N-1}+u_{N-2}}{(\Delta x)^2} \end{cases} \tag{5.13}$$

où $N = \frac{M}{2}$. Les conditions aux limites, qui sont périodiques, sont données par

$$\begin{cases} u_{-N}(t) = e^{-\beta t}sin(-\pi-\alpha t) = -e^{-\beta t}sin(\pi+\alpha t) = e^{-\beta t}sin(\alpha t) \\ u_N(t) = e^{-\beta t}sin(\pi-\alpha t) = e^{-\beta t}sin(\alpha t). \end{cases} \tag{5.14}$$

La condition initiale est $\quad u_i(0) = sin(x_i), \ -N+1 \le i \le N-1.$

Le système (5.13) peut etre réécrit comme suivant :

$$\frac{du_i}{dt} = \frac{\beta}{(\Delta x)^2}u_{i-1} + \frac{1}{\Delta x}\left(\frac{3}{2}\alpha - \frac{2\beta}{\Delta x}\right)u_i + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - 2\alpha\right)u_{i+1} + \frac{\alpha}{2\Delta x}u_{i+2}, -N+1 \le i \le N-2 \tag{5.15}$$

et

$$\frac{du_{N-1}}{dt} = \frac{1}{\Delta x}\left(\frac{\alpha}{2} + \frac{\beta}{\Delta x}\right)u_{N-2} - \frac{2\beta u_{N-1}}{(\Delta x)^2} + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - \frac{\alpha}{2}\right)u_N \tag{5.16}$$

En tenant compte des conditions aux limites, nous obtenons :

$$\begin{cases} \frac{du_{-N+1}}{dt} = \frac{\beta}{(\Delta x)^2}e^{-\beta t}sin(\alpha t) + \frac{1}{\Delta x}\left(\frac{3\alpha}{2} - \frac{2\beta}{\Delta x}\right)u_{-N+1} + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - 2\alpha\right)u_{-N+2} + \frac{\alpha}{2\Delta x}u_{-N+3} \\ \frac{du_i}{dt} = \frac{\beta}{(\Delta x)^2}u_{i-1} + \frac{1}{\Delta x}\left(\frac{3\alpha}{2} - \frac{2\beta}{\Delta x}\right)u_i + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - 2\alpha\right)u_{i+1} + \frac{\alpha}{2\Delta x}u_{i+2}, -N+2 \le i \le N-3 \\ \frac{du_{N-2}}{dt} = \frac{\beta}{(\Delta x)^2}u_{N-3} + \frac{1}{\Delta x}\left(\frac{3\alpha}{2} - \frac{2\beta}{\Delta x}\right)u_{N-2} + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - 2\alpha\right)u_{N-1} + \frac{\alpha}{2\Delta x}e^{-\beta t}sin(\alpha t) \\ \frac{du_{N-1}}{dt} = \frac{1}{\Delta x}\left(\frac{\alpha}{2} + \frac{\beta}{\Delta x}\right)u_{N-2} - \frac{2\beta}{(\Delta x)^2}u_{N-1} + \frac{1}{\Delta x}\left(\frac{\beta}{\Delta x} - \frac{\alpha}{2}\right)e^{-\beta t}sin(\alpha t). \end{cases} \tag{5.17}$$

5.4 Application de la méthode des différences finies au modèle du brusselator en présence de la diffusion : Utilisation du schéma semi-discret en espace et de la méthode RK4

On considère le système de deux équations aux dérivées partielles suivant :

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} = A + u^2 v - (B+1)u + \alpha \frac{\partial^2 u(t,x)}{\partial x^2} \\ \frac{\partial v(t,x)}{\partial t} = Bu - u^2 v + \alpha \frac{\partial^2 v(t,x)}{\partial x^2} \end{cases} \tag{5.18}$$

avec $0 \leq x \leq 1$, $A = 1$, $B = 3$, $\alpha = 1/50$ et les conditions aux limites $u(0,t) = u(1,t) = 1$, $v(0,t) = v(1,t) = 3$, $u(x,0) = 1 + \sin(2\pi x)$, $v(x,0) = 3$.

Pour résoudre numériquement ce système différentiel, nous discrétisons $u(t,x)$ et $v(t,x)$ dans l'espace avec la taille de la grille $\Delta x = 1/(N+1)$. Plus précisément, nous avons une grille de N points à savoir

$$x_i = i/(N+1) \quad (1 \leq i \leq N),\ \Delta x = 1/(N+1). \tag{5.19}$$

Nous obtenons alors, à partir de (5.18), les équations suivantes :

$$\begin{cases} \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = A + u_i^2 v_i - (B+1)u_i + \frac{\alpha}{(\Delta x)^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) \\ \frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = Bu_i - u_i^2 v_i + \frac{\alpha}{(\Delta x)^2}\left(v_{i-1} - 2v_i + v_{i+1}\right) \end{cases} \tag{5.20}$$

avec $u_0(t) = u_{N+1}(t) = 1$, $v_0(t) = v_{N+1}(t) = 3$, $v_i(0) = 1 + \sin(2\pi x_i)$, $v_i(0) = 3$, $i = 1,\ 2,\ \cdots,\ N$.

Indications :

- Considérez le système à $2N$ équations différentielles ordinaires obtenues à partir de (5.20) en remplaçant le couple (u,v) par le vecteur $\mathbf{y} = \begin{pmatrix} u \\ v \end{pmatrix}$.
- Résoudre ce système d'EDO au moyen de la méthode RK4 en prenant $N = 40$, $A = 1$, $B = 3$, $\alpha = 1/50$, $0 \leq t \leq 10$, $\Delta t = 1/10000$ et en utilisant le logiciel Octave.
- Représenter graphiquement, dans l'espace tridimensionnel, les résultats obtenus ((t,x,u) et (t,x,v)) au moyen du logiciel Octave.

5.5 Application de la méthode des différences finies à l'équation de la chaleur

Trouver $u : [0,T] \times [a,b] \ \to \mathbb{R}$ telle que

$$\frac{\partial u}{\partial t}(t,x) - D\frac{\partial^2 u}{\partial x^2}(t,x) = f(t,x), \qquad \forall (t,x) \in]0,T[\times]a,b[, \tag{5.21}$$

$$u(0,x) = u_0(x), \qquad \forall x \in [a,b] \tag{5.22}$$

$$-D\frac{\partial u}{\partial x}(t,a) = \alpha(t), \qquad \forall t \in [0,T] \tag{5.23}$$

$$u(t,b) = \beta(t), \qquad \forall t \in [0,T] \tag{5.24}$$

où $a < b$, $D > 0$ (coefficient de diffusivité), $\alpha : [0,T] \to \mathbb{R}$, $\beta : [0,T] \to \mathbb{R}$, $u_0 : [a,b] \to \mathbb{R}$ et $f : [0,T] \times [a,b] \to \mathbb{R}$ donnés.

Pour que le problème soit bien posé, il est nécessaire d'avoir compatibilité entre la condition initiale et la(les) condition(s) aux limites de type Dirichlet. Dans la présente application, (5.22) est la **condition initiale** et les **conditions aux limites** sont données par (5.23) (type Newmann) et (5.24) (type Dirichlet). Une **condition de compatibilité** n'apparaît alors qu'au point (t,x)=(0,b) et elle est donnée par

$$u_0(b) = \beta(0). \tag{5.25}$$

Application numérique :

$a = 0$, $b = 2\pi$, $N_x = 50$, $T = 10$, $N_t = 5000$, $D = 1$, $u(t,x) = \cos(t)\cos(x)$,

$f = D\cos(t)\cos(x) - sin(t)\cos(x)$, $\beta(t) = u(t,b)$, $\alpha(t) = D\cos(t)\sin(a)$ et $u_0(x) = \cos(x)$.

Bibliographie

[Alain2009] Alain Yger, Jacques-Arthur Weil et al., **Mathématiques L3 appliquées**. Cours avec 500 tests et exercices corrigés, (Pearson Education, Pans, 2009).

[Alfio2004] Alfio Quarteroni, Riccardo Sacco and Fausto Saleri , **Méthodes Numériques. Algorithmes, analyse et applications** (Springer-Verlag Italia, Milano 2004).

[Alfio2014] Alfio Quarteroni, Fausto Saleri and Paola Gervasio , **Scientific Computing with MATLAB and Octave**, Fourth Edition (Springer-Verlag, Berlin Heidelberg 2014). https://shannon.ir/Bookme/Scientific%20Computing%20with%20MATLAB%20and%20Octave.pdf

[Barnett1989] Barnett S., **Leverrier's Algorithm : A New Proof and Extensions**. *Numer. Math.* **7**, (1989), 338-352.

[Bose2008] Sujit Kumar Bose , **Numeric Computing in Fortran** (Alpha Science International Ltd, Oxford 2008).

[Curtiss1952] C. F. Curtiss C. F. et Hirschfelder J. O. , *Mathematics* , **38** , (1952) , 235-243.

[Cuvelier2019] François Cuvelier , **Méthodes numériques II.** Notes de cours de l'Université Paris XIII/Institut Galilée. https://www.math.univ-paris13.fr/~cuvelier/docs/Enseignements/Energetique/MethNumII/18-19/MethNumII_14janvier2019.pdf

[Ernest1996] Ernest Hairer et G. Wanner , **Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems ,** 2^e **éd.** (Springer-Verlag, Berlin 1996).

[Faddeev1963] Faddeev D. K. and Faddeeva V. N. , **Computational Methods of Linear Algebra** (Freeman, San Francisco and London 1993)

[Fatunula1988] Simeon Ola Fatunla , **Numerical Methods for Initial Value Problems in Ordinary Differential Equations** (Academic Press Inc., Harcourt Brace, Jovanovich Publishers, New York, 1988)

[Fefferman1998] Fefferman Charles L. , **Existence and Smoothness of the Navier-Stokes Equation**. rapport LAN, 1998

[Fibonacci] Nils Berglund (2005) , **Récréations mathématiques. La suite de Fibonacci** (Université du Sud Toulon–Var, novembre 2005), http://www.univ-orleans.fr/mapmo/membres/berglund/fibonacci.pdf

[Franck2005] Franck Jedrzejewski , **Introductions aux méthodes numériques,** 2^e **éd.** (Springer-Verlag France, Paris 2005).

[Goncalves2005] Goncalvès Eric, **Résolution numérique, discrétisation des EDP et EDO.** Notes de cours de l'Institut National Polytechnique de Grenoble. http://www.hach.ulg.ac.be/cms/system/files/Cours%20Grenoble%20EDP-EDO.pdf

[Goyon1996] Goyon Olivier S. O., **High-Reynolds number solutions of Navier-Stokes equations using incremental unknows**. *Comput. Methods Appl. Mech. Engrg.*, **130**, (1996), 319-335.

[Guillaume2009] Guillaume Legendre, **Méthodes numériques. Introduction à l'analyse numérique et au calcul scientifique**. Cours de Deuxième année de licence de Mathématiques et Informatique appliquées à l'Economie et à l'Entreprise (MI2E) à l'université de Paris-Dauphine, année académique 2009-2010, `http://users.metu.edu.tr/baver/Cours.NUM.pdf`

[Jacques2017] Jacques Lefrere, **Introduction au fortran 90/95/2003/2008**. Cours du Master de Sciences et Technologies (Université Pierre et Marie Curie, Paris VI, 2017), `http://wwwens.aero.jussieu.fr/lefrere/master/mni/f90+c/polyf90.pdf`

[Jean-Marc2002] Jean-Marc Huré et Didier Pelat, **Méthodes numériques. Elements d'un premier parcours**. Cours destiné aux étudiants de DEA Astrophysique & Méthodes associées aux université Paris 7 et 11, année académique 2002-2003 `https://media4.obspm.fr/public/M2R/supports/CoursMN.pdf`

[Laurent2015] Laurent Signac, **Introduction à l'algorithmique et à la programmation avec Python**, `https://deptinfo-ensip.univ-poitiers.fr`

[Michael2018] Michael Metcalf, John Reid and Malcolm Cohen, **Modern Fortran Explained : Incorporating Fortran 2018** (Numerical Mathematics and Scientific Computation), Fifth Edition, (Oxford University Press, Oxford 2018). `https://b-ok.africa/book/3712712/50cded`

[Michael2023] Michael Metcalf, John Reid and Malcolm Cohen, **Modern Fortran Explained : Incorporating Fortran 2023** (Numerical Mathematics and Scientific Computation), Sixth Edition, (Oxford University Press, Oxford 2023).

[Nyengeri2023] Nyengeri, H., Sinzinkayo, J. J., Dusaba, B. and Ndanzako, E. (2023) Effect of Asymmetric Finite Difference Formulas on the Orders of Central Difference Approximations for the Second Derivatgive of a Periodic Function. Open Access Library Journal, **10** :e10875. `https://doi.org/10.4236/oalib.1110875`

[Sandu1997] A. Sandu et al., *Benchmarking Stiff ODE Solvers for Atmospheric Chemistry Problems II : Rosenbrock Solvers. Atmospheric Environment*, **31**, (1997), 3459-3472.

[Wikipedia] Wikipedia, `https://fr.wikipedia.org/wiki/Langage_de_programmation`