Projets de licence en analyse numérique

I. Introduction

En guise d'introduction et de premier projet, on se fixe pour objectif de calculer le $n^{\text{ième}}$ terme d'une récurrence linéaire d'ordre k, par exemple pour la suite de Fibonacci (où $k = 2$)

$$\begin{aligned} f_{n+1} &= f_n + f_{n-1} \\ f_0 &= 0 \\ f_1 &= 1 \end{aligned}$$

La première approche consiste en l'algorithme très sobre

```
f:=proc(k)
if k<=1 then k else f(k-1)+f(k-2) fi
end;
```

et on vérifiera que cet algorithme n'est pas très bon quand n est assez grand : expliquer, et améliorer.

Une seconde approche est de résoudre explicitement la récurrence linéaire en utilisant le polynôme caractéristique, par exemple en Maple (un restart sera peut-être nécessaire !)

```
rsolve({f(n+1)=f(n)+f(n-1),f(0)=0,f(1)=1},f(n));
```

ce qui donne la solution en fonction de n après quoi il suffit de remplacer. Quel est l'inconvénient de cette méthode ?

Une troisième solution est de transformer la récurrence scalaire d'ordre k en une récurrence linéaire vectorielle de dimension k d'ordre 1, en posant dans le cas de la suite de Fibonacci

$$x_k = \begin{pmatrix} f_0 \\ f_1 \end{pmatrix}$$

après quoi il suffit de définir la matrice d'itération A telle que $x_{k+1} = Ax_k$. Ainsi, le calcul du $n^{\text{ième}}$ terme revient au calcul de A^n qu'il nous faut optimiser quand k est grand. En effet, on peut facilement calculer A^n avec beaucoup moins de n multiplications matricielles. Il suffit formellement d'écrire n en base 2, $n = n_0 + 2n_1 + 2^2 n_2 + 2^3 n_3 + \cdots + 2^m n_m$ puis

$$A^n = A^{n_0} \left(A^{n_1} (\cdots)^2 \right)^2$$

En déduire un algorithme d'exponentiation rapide et donner une estimation du nombre de multiplications scalaires nécessaires pour calculer A^n en fonction de n. Calculer effectivement la complexité pour de grandes valeurs de n. Appliquer l'algorithme d'exponentiation rapide au problème des suites récurrentes. Comparer avec la méthode la plus naïve.

II. Description

Pour tous les sujets, on cherchera à évaluer la complexité soit en comptant le nombre d'opérations théoriquement nécessaire quand c'est possible, soit en donnant une borne maximale ou encore une moyenne du nombre d'opérations en fonction de la taille du problème. On vérifiera la programmation des algorithmes en comptant les opérations (en utilisant par exemple des variables globales incrémentées à chaque opération élémentaire) et on tracera le temps d'exécution en fonction de la taille du problème pour vérifier l'ordre des méthodes. Par exemple, si une méthode nécessite $2n^2 + 3n$ multiplications pour un problème de taille n, on dit que sa complexité est de l'ordre de $O\left(n^2\right)$ et si on trace le temps d'exécution de l'algorithme en fonction de n, on doit observer (pour n assez grand) une branche parabolique.

On montrera les problèmes éventuels d'instabilité numérique sur des exemples particuliers et en fonction des méthodes. Dans la mesure du possible, on cherchera à caractériser *a priori* les problèmes dont la résolution numérique peut poser des problèmes.

Dans plusieurs cas, des méthodes standard d'accélération de la convergence (Δ^2 d'Aitken, Richardson) peuvent améliorer significativement la vitesse de convergence des algorithmes. Il faut les appliquer quand c'est possible.

Le travail demandé comporte une part importante de recherche bibliographique. Des méthodes non listées dans les sujets ci-dessous mais qui permettent de répondre aux problèmes soulevés sont les bienvenus.

III. Liste des sujets

A. Méthodes de résolution des systèmes linéaires

Méthodes directes : Méthode de Gauss et de Householder (factorisation QR) et méthodes itératives : méthode de Jacobi, de Gauss-Seidel et de relaxation. On demande une programmation de toutes ces méthodes avec comparaisons (stabilité numérique, complexité) et une discussion sur le choix du paramètre ω dans la méthode de relaxation.

B. Méthodes de calcul des valeurs propres et des vecteurs propres

Méthodes de Jacobi, Givens–Householder, QR ; Méthodes de la puissance itérée (Leverrier,...). Comparaisons, complexité, stabilité numérique.

C. Résolution d'équations non–linéaires

Dichotomie, Newton, quasi-Newton ; Comparaisons, complexité, stabilité numérique. Cas particulier des polynômes dans $\mathbb{R}$: recherche exhaustive des racines. Extension de la méthode de Newton dans $\mathbb{R}^n$; minimisation (méthode du gradient).

D. Interpolation, fonctions splines, courbes de Béziers

Interpolation de Lagrange (calcul direct et méthode de Newton), de Hermite, d'Abel–Gontcharov, trigonométrique ; choix optimal des points d'interpolation, phénomène de Runge. Fonction splines. Courbes de Béziers.

E. Calcul numérique d'intégrales

Méthodes de Gauss pour des fonctions de poids particulières : polynômes de Jacobi, Legendre, Chebyshev (1ère et 2nde espèce), Laguerre et Hermite. Méthodes de Newton–Côtes, calcul des noyaux de Peano. Méthode de Romberg. Comparaisons, complexité, stabilité numérique.

F. Intégration numérique des équations différentielles

Méthodes de Runge–Kutta. Discussion sur le calcul des coefficients en fonction de l'ordre. Estimation de l'erreur par des méthodes emboîtées, pas adaptatif. Comparaison avec d'autres méthodes (à pas multiple tel que la méthode d'Adams), complexité, stabilité numérique.

G. Intégration numérique des équations différentielles en présence d'intégrales premières, systèmes raides.

Méthode d'Euler et plus généralement méthodes de Runge–Kutta implicites et explicites. Méthode de pénalisation. Application aux systèmes Hamiltoniens.

H. Problèmes de tir

Système d'équations différentielles linéaire, calcul de l'exponentielle par la méthode des approximants de Padé (trouver deux polynômes $P(x)$ et $Q(x)$ de degré n tels que la fraction rationnelle $\frac{P(x)}{Q(x)}$ ait même développement limité que $\exp(x)$ à l'ordre $2n$, application matricielle). Équations différentielles du premier ordre avec condition terminale : résolvante et dérivée de $x(t)$ par rapport aux conditions initiales. Équations différentielles du second ordre avec conditions aux bords. Méthode des différences finies, méthode aux éléments finis (méthode de Galerkin) (cf [8], [10], [11]).

IV. Évaluation

Le sujet doit être bien compris et les programmes doivent fonctionner correctement dans tous les cas pour lesquels ils sont conçus. La note de TP sera calculée en fonction du travail effectué par rapport au travail demandé sur la base

– d'un rapport écrit remis par le binôme comportant une partie théorique (explication des algorithmes, analyse de la complexité) et une partie pratique (résultats des tests comparatifs et quelques résultats de simulation choisis pour leur aspect illustratif) ainsi qu'une conclusion et une bibliographie sur le sujet. Les algorithmes doivent être justifiés en citant les théorèmes qui en sont la base théorique (la démonstration de ces théorèmes n'est pas requise) ;

– d'un examen oral devant l'ordinateur pendant lequel le binôme défend son travail en faisant une démonstration des programmes écrits (chaque ligne de programme doit pouvoir être expliquée) et en répondant à quelques questions vaches posées par l'examinateur.

References

[1] *Notes de cours d'analyse numérique élémentaire : des équations algébriques aux équations différentielles.* Document de l'Université de Bourgogne, Licence de Mathématiques.

[2] J. Baranger, Analyse numérique, Hermann, 1991

[3] P. G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson/Dunod

[4] M. Crouzeix, A. L. Mignot, Analyse numérique des équations différentielles, Masson, 1992

[5] J.–P. Demailly, Analyse numérique et équations differentielles, Presses universitaires de Grenoble, 1991

[6] N. Gastinel, Analyse numérique linéaire, 1966

[7] A. S. Householder, The theory of matrices in numerical analysis, Dover, 1964

[8] P. Lascaux, R. Théodor, Analyse numérique matricielle appliquée à l'art de l'ingénieur (2 vol.), 1993

[9] B. Lucquin et O. Pironneau, *Introduction au calcul scientifique*, Masson, 1996

[10] J. P. Nougier, Méthodes de calcul numérique, Masson, 1983

[11] J. Rappaz, M. Picasso, Introduction à l'analyse numérique, Presses polytechniques et universitaires romande, 1998

[12] M. Schatzman, Analyse numérique, cours et exercices pour la licence, InterÉditions, 1991

[13] M. Sibony, J. Cl. Mardon, Cours d'analyse numérique (3 vol.), Hermann, 1982

[14] R. Théodor, Initiation à l'analyse numérique, cours du CNAM, Masson, 1986