Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de la Technologie
Ecole Supérieure de la Statistique et de l'Analyse de l'Information de Tunisie

Cours d'Analyse Numérique

Implémentations sur sage

Première Année du cycle d'Ingénieur

Ines Abdeljaoued-TEJ

Année Universitaire 2011/2012

ESSAI, 6 rue des métiers - BP 675, Tunis 1080
Tél : 71 703 717 / 71 704 309 - **Fax :** 71 704 329
Mél : Ines.Abdeljaoued@insat.rnu.tn ou inestej@gmail.com

Attention, malgré une re-lecture soigneuse, il est possible que ce cours contienne des fautes de frappe. Si vous en trouvez, merci de nous en avertir de préférence par e-mail : Ines.Abdeljaoued@insat.rnu.tn ou inestej@gmail.com. Si vous avez des questions de compréhension du cours, n'hésitez pas à nous poser les questions par email.

Table des matières

Introduction

La démarche du numéricien commence par la modélisation du problème ou sa mise en équation. Une analyse mathématique permet ensuite d'étudier le problème, de déterminer sous quelles conditions il existe une solution unique et sinon, choisir celle qui correspond au mieux à la réalité.

Lorsque l'ingénieur ne peut pas calculer toutes les valeurs, le mieux est d'approcher le problème en calculant des valeurs de certains points. C'est la discrétisation, qui doit être accompagnée d'une mesure de l'erreur commise, c'est-à-dire mesurer la distance entre la solution approchée et la véritable solution du problème. Une étape de simplification, comme la linéarisation pour des problèmes non linéaires, permet d'obtenir des problèmes pour lesquels on dispose de méthodes de résolution numérique performantes : résolution de systèmes linéaires, optimisation par une méthode de gradient, recherche de valeurs propres.

Comme suite aux résultats du calcul par ordinateur, il convient d'être circonspect et d'étudier les sources d'erreurs : les données elles-mêmes, l'arrondi, la méthode utilisée. La démarche qui vient d'être présentée est celle que suit le numéricien en charge de la simulation numérique d'un problème.

Le cours est structuré en trois grands chapitres :
– Interpolation et Approximation
– Racine d'équations non linéaires
– Résolution de systèmes d'équations linéaires.

Etant donné la valeur connue d'une certaine fonction en un certain nombre de points, quelle valeur prend cette fonction en un autre point quelconque situé entre deux points donnés ? Une méthode très simple est d'utiliser l'interpolation linéaire, qui suppose que la fonction inconnue évolue linéairement entre chaque paire de points successifs connus. Nous introduisons dans le premier chapitre l'interpolation polynomiale ainsi que d'autres méthodes d'interpolation utilisant des fonctions localisées telles que les splines.

Un autre problème fondamental est le calcul des solutions d'une équation donnée. Les algorithmes de recherche de racines d'une fonction sont utilisés pour rśoudre les équations

non linéaires. Si la fonction est différentiable et que sa dérivée est connue, alors nous choisirons la méthode de Newton sinon, d'autres méthodes seront détaillés dans le deuxième chapitre.

Le troisième chapitre porte sur l'étude des méthodes de calcul du vecteur x solution du système d'équations linéaires $Ax = b$. La première partie de ce chapitre sera consacrée aux méthodes directes et la deuxième aux méthodes itératives. Nous aborderons ensuite en troisième partie les méthodes de résolution de problèmes aux valeurs propres.

Le champ de l'analyse numérique est divisé en différentes disciplines suivant le type de problème à résoudre, et chaque discipline étudie diverses méthodes de résolution des problèmes correspondants. Il s'agit dans un premier temps d'étudier les principales méthodes en Analyse Numérique (Méthode de Lagrange, algorithmes du point fixe, de Newton, du gradient conjugué, la factorisation LU, etc...). Nous nous intéressons ensuite aux principes qui sous-tendent ces méthodes ainsi qu'à leur mise en oeuvre pratique en langage SCILAB et MAXIMA. Il s'agira enfin d'implémenter et de tester plusieurs algorithmes d'Analyse Numérique et plusieurs approches de résolution.

Nous donnons à la fin de chaque chapitre une liste d'exercices de difficultés variables. Ces exercices ne sont profitables que si l'élève-ingénieur les travaille. Qu'il garde à l'esprit ce proverbe chinois :

J'entends et j'oublie, (cours oral)
je vois et je retiens, (étude du cours)
je fais et je comprends (exercices).

Pour conclure, ce cours est une introduction à des approches plus complexes des mathématiques de l'ingénieur : d'autres thématiques seront développés dans les cours d'Optimisation Convexe, de Calcul de Variations, de Recherche Opérationnelle, etc...

Enfin, rappelez-vous la devise suivante :

That which is learned without desire is soon forgotten.

Sagemath : une plateforme mathématique complète et libre

`Python` est un langage de programmation adapté au calcul scientifique. Il est interprété, interactif et orienté objet. Il possède de nombreuses extensions pour le calcul, le traçage, le stockage de données. Combiné avec des fonctionnalités comme Tkinter, une interface graphique pour les utilisateurs, `Python` permet de développer de bonnes interfaces graphiques pour les codes. Par exemple, les principaux mots cles sous `Python` sont :

```
and else import assert except in    break
exec is class finally lambda  continue for
not def from or del global pass elif if print
raise return try while yield
```

L'aspect le plus intéressant est qu'il existe un ensemble de modules pour le calcul scientifique sous `Python`. Des modules pour pratiquement tout (vecteurs, tenseurs, transformations, dérivées, algèbre linéaire, transformées de Fourier, statistiques, groupes, etc) sont disponibles. Il est aussi possible d'utiliser des bibliothèques `C` et `Fortran` à partir de `Python`. Finalement, écrire un programme numérique est chose aisée en `Python`. Il est également possible de réaliser des algorithmes et des calculs parallèles : des interfaces existent vers netCDF (fichiers binaires portables), MPI et BSPlib.
Comme exemple, existe `Scipy`, la bibliothèque d'outils scientifiques libres pour `Python`. Elle inclut des modules pour créer des graphiques et faire du traçage, des modules d'optimisation, d'intégration, de fonctions spéciales, de manipulation de signaux et d'images, d'algorithmes génétiques, de solveurs d'équations différentielles ordinaires, etc.
Le responsable, K. Hinsen, propose un bon didacticiel intitulé Scientific Computing in `Python` (Calcul scientifique avec `Python`). Consuler aussi I. Lima dans [**?**].

`sage` par l'exemple

La plateforme `sage` est un ensemble de logiciels libres et payants (sous réserve d'avoir la licence) qui cohabitent ensemble grâce au langage `Python`.

Commandes de base

Afin de mieux illustrer les utilisations de base de `sage`, nous nous sommes largement inspirés de [**?**]. Mais avant tout, télécharger sage-4.4.1.tar depuis sagemath.org (280MB) et taper les commandes suivantes :

```
$ tar xf sage-4.3.1.tar
$ cd sage-4.3.1
$ make
```

Aller prendre un café (ou plusieurs). . .

1. Les opérations arithmétiques sont données par :

```
2+3
15 + 3 * 2 / 8 - 2^3
type(21)
--2
30/3
n(30/3, 4)
2^3
8%5
boite=7
```

   7

2. Les chaînes de caractères :

```
a="Bonjour, je suis une chaine de caractere"
print(a)
type(a)
```

   <type 'str'>.

3. Opérateurs conditionnels :

```
x = 2
y = 3
print x == y
print x <> y
print x, "!=", y, ":", x != y
print x < y
print x > y
print x >= y
```

   False.

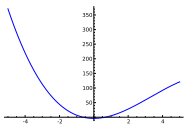
4. Utiliser ce code avec la variable `s` et obtenir la solution :

```
var('a,b,c')
eqn = [a+b*c==1, b-a*c==0, a+b==5]
s = solve(eqn, a,b,c)
```

La solution de eqn $= [bc + a = 1, -ac + b = 0, a + b = 5]$ est :

$$\left[a = \frac{25i\sqrt{79} + 25}{6i\sqrt{79} - 34}, b = \frac{5i\sqrt{79} + 5}{i\sqrt{79} + 11}, c = \frac{1}{10}i\sqrt{79} + \frac{1}{10}\right]$$

$$\left[a = \frac{25i\sqrt{79} - 25}{6i\sqrt{79} + 34}, b = \frac{5i\sqrt{79} - 5}{i\sqrt{79} - 11}, c = -\frac{1}{10}i\sqrt{79} + \frac{1}{10}\right]$$

5.
```
var('x')
f=-x^3+3*x^2+7*x^2-4
```
On construit l'objet courbe, stocké dans la variable `p` :
```
p = plot(f, x, -5, 5)
```


6. La commande if :
```
x = 3
print x > 5
if x > 5:
    print x
    print "plus grand"
    print "Fin du programme"
else:
    print "Introduire une autre valeur"
```

7. L'opérateur booléen and :
```
a = 7
b = 9
print a > 5 and b > 10
if a > 5 and b < 10:
    print "Les deux expressions sont vraies"
```

8. L'opérateur booléen or :
```
print a > 5 or b > 10
if a > 5 or b > 10:
    print "Au moins une des deux expressions est vraie"
```

9. L'opérateur booléen not :
```
a = 7
print a > 5
print not a > 5
```

10. La boucle while :

```
x = 1              #initialisation
while x <= 40:
    print x,
    x = x + 2      #incrementation de x par 2
```

11. Les ensembles :

```
x = Set([2,3,51,21])
print type(x)
print x[0]
print x
y = Set([51])
x.difference(y)
a = Set([1,3,2,5,5])
a.cardinality()
3 in a
a.union(x)
a.intersection(x)
```

12. Les listes :

```
y = [1,1/2,0.75,'Bonjour',x]
type(y)
y[3]
y[4]
add([1,2,3,4])
```

13. Les n-uplets :

```
z = (1,3,5,4)
type(z)
z[0]
a,b,c = (1,2,3)
a; b; c
```

14. Les opérateurs in et not in :

```
54 in [2,54,2]
30 not in [2,54,2]
```

15. La boucle for :

```
for i in [1,2,3,4,5]:
     i,
```

16. Les fonctions :

```
def salut(s):
    """
```

```
    Retourne Bonjour ou Au revoir.
    """
    if s>3:
        return "Bonjour"
    else:
        return "Au revoir"

html("<H1>Exemple de fonctions interactive</H1>")
html("<I> Juste pour voir</I>")
@interact
def _(p=1):
    salut(p)
```

17. Quelle est l'utilité des commandes suivantes :

```
denominator(12/23)
divisors(20)
gcd(40,116)
mul([2,3,4])
len([1,2,3,'hello'])
srange(11)
a=srange(4,step=1.3)
b=srange(-5,-1)
zip(a,b)
```

$[(0.000000000000000, -5), (1.30000000000000, -4), (2.60000000000000, -3), (3.90000000000000, -2)]$.

18. La boucle for :

```
for t in srange(6):
    t,

d=zip(a,b)
for (x,y) in d:
    x,y

[2*t for t in [0,1,2,3]]

[t^2 for t in range(20) if t%2==0]
```

19. Obtenir des résultats numériques :

```
a = pi
a
n(a)
a.n(digits=30)
a.n(prec=12)
```

20. Calcul symbolique :

```
var('a')
type(2*a)
m = 5 + a
n = 8 + a
y = m * n
z = y.expand()
y(5)
```

    130.

21. Equations symboliques : $[a = (-5)]$.

22. Divers :

```
A=matrix(3,3,[1,2,3,2,3,4,1,2,3])
latex(A)
```

23. Courbes en 2 ou 3 dimensions :

24. Calcul de $0 + 1 + \cdots + N - 1$:

```
def sum1(N):
 s = 0
for k in range(N):
s += k
return s
sage: time sum1(10^6)
```

    Cette fonction peut être sauvegardée dans un fichier (sum2.pyx par exemple). Les commandes suivantes permettent de lire ou d'exécuter le fichier :

```
sage: cat sum2.pyx
sage: load sum2.pyx
sage: time sum2(10^6)
```

Exemples divers

1. On considère les vecteurs et les matrices suivants

$$l = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \end{pmatrix}, v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}, A = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

    Créer et commenter le résultat des instructions $l * v$, $A + B$, $A * B$, B/A, ${}^t v$.

2. On appelle *méthode de Monte-Carlo* toute méthode visant à calculer une approximation numérique par utilisation de procédés aléatoires. Nous allons calculer une

valeur approchée de π par la méthode de Monte-Carlo [1].
On tire au hasard des points de coordonnées (x, y) dans un carré de longueur égale à 2. On vérifie s'ils appartiennent ou non à un disque de rayon 1 et de centre, le centre du carré. Notons que ces points peuvent être tirés avec la même probabilité dans l'ensemble du carré. D'après la loi des grands nombre, lorsque le nombre de tirage tend vers l'infini, le rapport entre le nombre de points tirés dans le disque et le nombre de points tirés au total tend vers le rapport des surfaces du cercle et du carré, soit $\frac{\pi}{4}$.

3. Etant donné $n > 0$, écrire en utilisant le moins de lignes possible, un script `sage` qui déclare la matrice de Cauchy de coefficients $a_{i,j} = \frac{1}{i+j}$.
4. On peut toujours transformer un problème du type $f(x) = 0$ en un problème de la forme $g(x) = x$ et ce d'une infinité de façons. Le théorème suivant caractérise les fonctions g qui donnent des suites convergentes : Si dans l'intervalle $[a, b]$, g vérifie les conditions suivantes :
   (a) $x \in [a, b]$ alors $g(x) \in [a, b]$,
   (b) l'application g est strictement contractante, c'est-à-dire $max_{x\in[a,b]} \mid g^{'}(x) \mid< 1$.

   Alors pour tout $x_0 \in [a, b]$, la suite définie par $x_{n+1} = g(x_n)$ converge vers l'unique point fixe x^* de g sur $[a, b]$.

   Ecrire une fonction prenant en entrée g, x_0, ϵ et un nombre d'itérations maximal N et qui rend une approximation du point fixe de g à ϵ près.

1. N. Metropolis et S. Ulam ont inventé en 1947 ce type de méthode. Le nom fait allusion aux jeux de hasard pratiqués dans les casinos de Monte-Carlo

Chapitre 1

Interpolation et Approximation

Soient $n+1$ couples : (x_0, f_0), (x_1, f_1), ..., $(x_n, f_n) \in \mathbf{R} \times \mathbf{R}$ tels que les x_i soient tous distincts. On cherche un polynôme $P : \mathbf{R} \longrightarrow \mathbf{R}$ tel que $P(x_i) = f_i$ pour $i = 0..n$.

Nous introduisons dans un premier temps la *formule d'interpolation de Lagrange* qui prouve l'existence et l'unicité du polynôme P de degré au plus n. Ensuite nous présenterons l'*algorithme d'Aitken* et la *méthode des différence divisées* qui utilise la *formule d'interpolation de Newton*. Puis, nous introduirons l'*algorithme de Neville* qui présente une méthode très pratique pour le calcul sur machine du polynôme P en un point. Nous introduirons enfin l'*interpolation Spline* qui est un outil fondamental du graphisme sur ordinateur.

1.1 Formule d'Interpolation de Lagrange

Généralement les réels f_i représentent le résultat d'expériences et sont l'expression d'une fonction f continue dans $\mathbf{R}$ en les points x_i. La formule d'interpolation de Lagrange est un moyen qui permet d'approcher la fonction f.

Théorème 1.1.1. *Soient (x_i, f_i) pour $i = 0..n$ des points réels ou complexes, avec $x_i \neq x_j$ pour $i \neq j$. Il existe un unique polynôme P de degré inférieur ou égal à n vérifiant $P(x_i) = f_i$ pour $i = 0..n$.*

La preuve du théorème 1.1.1 se base sur les *polynômes d'interpolation de Lagrange*

$$L_i(x) = \prod_{j=0, j\neq i}^{j=n} \frac{x - x_j}{x_i - x_j}$$

qui vérifient $L_i(x_i) = 1$ et pour $j \neq i$: $L_i(x_j) = 0$. Le polynôme P est alors égal à $P(x) = \sum_{i=0}^{i=n} f_i L_i(x)$ et il est appelé *polynôme ou formule de Lagrange.*

Démonstration : Soient P et Q deux polynômes de degré au plus égal à n et vérifiant $P(x_i) = Q(x_i) = f_i$ pour $i = 0..n$. Alors le polynôme $r = P - Q$ (qui est de degré au plus égal à n) s'annule en $x_0, x_1, \ldots, x_n$. C'est donc un polynôme qui a $n+1$ racines : il est nécessairement nul. Donc $P = Q$, d'où l'unicité.
Pour montrer l'existence d'un tel polynôme, il suffit de remarquer que la famille $(L_0, L_1, \ldots, L_n)$ forme une base de l'espace vectoriel des polynômes de degré inférieur ou égal à n (qui est de dimension $n+1$). En effet, pour tout x, $\sum_{i=0}^{i=n} \alpha_i L_i(x) = 0$ implique en particulier que $\alpha_k = 0$ pour $k = 0..n$ (on remplace x par x_k). On retrouve la formule de Lagrange en écrivant P dans cette base : $P = \sum_{i=0}^{i=n} f_i L_i(x)$ et $P(x_i) = f_i$ pour tout $i = ..n$.

Exemple 1.1.1. Le polynôme P de degré inférieur ou égal à 3 vérifiant

$$P(0) = 1, \qquad P(1) = 2, \qquad P(-1) = -1 \qquad et \qquad P(2) = 4$$

est $P(x) = 1.L_0(x) + 2.L_1(x) + (-1).L_{-1}(x) + 4.L_2(x) = \frac{1}{3}x^3 - \frac{1}{2}x^2 + \frac{7}{6}x + 1$ où :

$$\begin{array}{llll} L_0(x) = & \frac{(x-1)(x+1)(x-2)}{(0-1)(0+1)(0-2)} & , \quad L_1(x) = & \frac{(x-0)(x+1)(x-2)}{(1-0)(1+1)(1-2)}, \\ L_{-1}(x) = & \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)} & , \quad L_2(x) = & \frac{(x-0)(x-1)(x+1)}{(2-0)(2-1)(2+1)}. \end{array}$$

Exercice 1.1.1. Construire les polynômes d'interpolation de Lagrange de $f(x) = e^{-x^2}$ sur les entiers de $[-2, 2]$.

Exercice 1.1.2. Ecrire un algorithme qui calcule $L_i(x)$ pour n, i et x fixés.

Remarque 1.1.1. Si on utilise la formule de Lagrange, le nombre d'opérations nécessaires pour calculer $P(x)$ est de l'ordre de $4n^2$ opérations pour chaque valeur de x.

Sur ordinateur, on préfère utiliser la *formule barycentrique* :

$$P(x) = \frac{\sum_{k=0}^{k=n} \frac{a_k f_k}{x - x_k}}{\sum_{k=0}^{k=n} \frac{a_k}{x - x_k}}$$

avec $a_k = \frac{1}{v'(x_k)}$ et $v(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$. Cette formule nécessite pour une première valeur de x l'ordre de $2n^2$ opérations afin de déterminer $P(x)$. Pour d'autres valeurs de x, $2n$ opérations sont nécessaire au calcul de P a condition de conserver les valeurs a_k.

1.2 Algorithme d'Aitken

Soit la relation de récurrence suivante :

$$\begin{cases} F_{k,0}(x) & = f_k, \qquad k = 0..n \\ F_{k,j+1}(x) & = \frac{F_{j,j}(x)(x_k - x) - F_{k,j}(x)(x_j - x)}{x_k - x_j}, \qquad k = j+1..n \end{cases}$$

Théorème 1.2.1. *Le polynôme de Lagrange $P(x)$ sur les points $x_0, x_1, \ldots, x_n$ est égal à $F_{n,n}(x)$.*

La preuve du théorème 1.2.1 se base sur le fait que $F_{k,j}(x)$ avec $k \geq j$ est le polynôme de Lagrange sur les points $x_0, x_1, \ldots, x_{j-1}, x_k$.

Algorithme. On calcule $P(x) = F_{n,n}(x)$ à l'aide du tableau triangulaire suivant :

k/j		0	1	2	$\ldots$	$j-1$	j	$\ldots$	$n-1$	$x_k - x$
0	$F_{0,0}$									$x_0 - x$
1	$F_{1,0}$	$F_{1,1}$								$x_1 - x$
2	$F_{2,0}$	$F_{2,1}$	$F_{2,2}$							$x_2 - x$
3	$F_{3,0}$	$F_{3,1}$	$F_{3,2}$	$F_{3,3}$						$x_3 - x$
$\vdots$					$\ldots$					$\vdots$
k	$F_{k,0}$	$F_{k,1}$	$F_{k,2}$	$F_{k,3}$		$F_{k,j}$	$F_{k,j+1}$			$x_k - x$
$\vdots$										$\vdots$
n	$F_{n,0}$	$F_{n,1}$	$F_{n,2}$						$F_{n,n}$	$x_n - x$

Par exemple, pour calculer $F_{k,2}(x)$, on considère le produit croisé suivant : $F_{1,1}(x)(x_k - x) - F_{k,1}(x)(x_1 - x)$. Nous obtenons :

$$F_{k,2}(x) = \frac{F_{1,1}(x)(x_k - x) - F_{k,1}(x)(x_1 - x)}{x_k - x_1}.$$

Nous calcuons les éléments du tableau progressivement afin de déterminer $F_{n,n}(x) = P(x)$.

Démonstration du théorème 1.2.1 : Par récurrence sur j : $F_{k,0}(x) = f_k$ pour $k = 0..n$. Supposons que $F_{k,j}(x)$ est le polynôme de Lagrange sur les points $x_0, x_1, \ldots, x_{j-1}, x_k$ pour tout $k = j..n$.
En particulier, et d'après l'hypothèse de récurrence, $F_{j,j}(x)$ est le polynôme de Lagrange sur $x_0, x_1, \ldots, x_{j-1}, x_j$.
Montrons que $F_{k,j+1}(x)$ est le polynôme de Lagrange sur $x_0, x_1, \ldots, x_j, x_k$ pour $k = j+1..n$.
Pour $i = 0..j-1$:

$$F_{k,j+1}(x_i) = \frac{F_{j,j}(x_i)(x_k - x_i) - F_{k,j}(x_i)(x_j - x_i)}{x_k - x_j} = f_i$$

Nous utilisons pour cela l'hypothèse de récurrence : $F_{j,j}(x_i) = f_i$ et $F_{k,j}(x_i) = f_i$ pour $i = 0..j-1$). D'autre part, en utilisant la définition, on trouve que $F_{k,j+1}(x_j) = f_j$ et $F_{k,j+1}(x_k) = f_k$. En prenant $j = n$, on obtient le résultat du théorème.

Remarque 1.2.1. Le polynôme $F_{k,j}(x)$ avec $k \geq j$ est le polynôme de Lagrange sur les points $x_0, x_1, \ldots, x_{j-1}, x_k$ de degré inférieur ou égal à j.

Exemple 1.2.1. L'algorithme d'Aitken appliqué à l'exemple 1.1.1 donne :

k/j		0	1	2	$x_k - x$
0	1				$-x$
1	2	$F_{1,1} = 1 + x$			$1 - x$
2	-1	$F_{2,1} = 2x + 1$	$F_{2,2} = -\frac{x^2}{2} + \frac{3x}{2} + 1$		$-1 - x$
3	4	$F_{3,1} = 1 + \frac{3x}{2}$	$F_{3,2} = \frac{x^2}{2} + \frac{x}{2} + 1$	$F_{3,3} = \frac{x^3}{3} - \frac{x^2}{2} + \frac{7x}{6} + 1$	$2 - x$

1.2.1 Formule de Newton

Soient $n \in \mathbf{N}$ et $0 \leq i < j \leq n$ deux entiers distincts. Notons $D_{i,j}(x)$ le polynôme de Lagrange sur $x_i, x_{i+1}, \ldots, x_j$. C'est un polynôme de degré inférieur ou égal à $j-i$ vérifiant $D_{i,j}(x_k) = f_k$ pour $i \leq k \leq j$.

1.2.2 La méthode des différences divisées

Proposition 1.2.2 (relation des différences divisées)**.** *Notons $c_{i,j}$ le coefficient de x^{j-i} dans $D_{i,j}(x)$. Alors, on a la relation suivante :*

$$\begin{cases} c_{i,i} = f_i & \text{pour } i = 0..n, \\ c_{i,j} = \frac{c_{i+1,j} - c_{i,j-1}}{x_j - x_i} & \text{pour } 0 \leq i < j \leq n. \end{cases}$$

Démonstration : Soit $n \in \mathbf{N}$. Pour tout $0 \leq i < j \leq n$, on a

$$D_{i,j}(x) = \frac{D_{i+1,j}(x)(x - x_i) - D_{i,j-1}(x)(x - x_j)}{x_j - x_i} \quad . \tag{1.2.1}$$

En effet, le degré de $D_{i+1,j}(x)$ est $\leq j - i - 1$ et le degré de $D_{i,j-1}(x)$ est $\leq j - i - 1$ ce qui montre que le degré du polynôme $\frac{(x-x_i)D_{i+1,j}(x) - (x-x_j)D_{i,j-1}(x)}{x_j - x_i}$ est $\leq j - i$. De plus, $\frac{D_{i+1,j}(x_k)(x_k - x_i) - D_{i,j-1}(x_k)(x_k - x_j)}{x_j - x_i} = f_k$ (il faut distinguer les trois cas : $k = i$, $k = j$ et $k = i + 1..j - 1$ où $D_{i+1,j}(x_k) = D_{i,j-1}(x_k) = f_k$).

Le polynôme $\frac{D_{i+1,j}(x)(x-x_i) - D_{i,j-1}(x)(x-x_j)}{x_j - x_i}$ est alors le polynôme de Lagrange sur les points $x_i, \ldots, x_j$ et puisqu'il est unique (voir théorème 1.1.1) on a l'égalité (1.2.1) qui est équivalente à :

$$D_{i,j}(x) = \frac{D_{i+1,j}(x)(x-x_i) - D_{i,j-1}(x)(x-x_i)}{x_j - x_i} + D_{i,j-1}(x) \qquad .$$

On remarque que $\frac{D_{i+1,j}(x)(x-x_i)-D_{i,j-1}(x)(x-x_i)}{x_j-x_i}$ est un polynôme de degré $\leq j-i$ qui s'annule en $x_i, x_{i+1}, ..x_{j-1}$ alors :

$$D_{i,j}(x) = c_{i,j}(x-x_i)(x-x_{i+1})\dots(x-x_{j-1}) + D_{i,j-1}(x) \qquad . \tag{1.2.2}$$

En examinant le coefficient de x^{j-i} de part et d'autre des deux égalités (1.2.1) et (1.2.2), nous obtenons la relation $c_{i,j} = \frac{c_{i+1,j}-c_{i,j-1}}{x_j-x_i}$:

$$c_{i,j}(x-x_i)(x-x_{i+1})\dots(x-x_{j-1}) + D_{i,j-1}(x) = \frac{D_{i+1,j}(x)(x-x_i) - D_{i,j-1}(x)(x-x_j)}{x_j - x_i}.$$

En itérant le résultat de l'équation (1.2.2), on obtient :

$$D_{i,j}(x) = c_{i,j}(x-x_i)(x-x_{i+1})\dots(x-x_{j-1}) + c_{i,j-1}(x-x_i)\dots(x-x_{j-2}) + \dots + c_{i,i}$$

et pour $i = 0$ et $j = n$, on a :

Définition 1.2.1 (Formule de Newton)**.** Le polynôme de Lagrange $P(x)$ sur les points $x_0, x_1, \dots, x_n$ est donné par la *formule de Newton* :

$$P(x) = D_{0,n}(x) = f_0 + \sum_{k=1}^{k=n} c_{0,k}(x-x_0)(x-x_1)\dots(x-x_{k-1}) \qquad .$$

$c_{0,k}$ est le coefficient de x^k appelée *la différence divisée d'ordre k aux points* $x_0, x_1, \dots, x_k$.

La *formule de Newton* permet de déterminer le polynôme de Lagrange. Pour cela, nous utilisons un tableau analogue à la méthode d'Aitken : les coefficients de $P(x)$ sont sur la diagonale du tableau triangulaire.

Exemple 1.2.2. En utilisant la méthode des différences divisées et plus précisément la proposition 1.2.2 dans l'exemple 1.1.1 on obtient :

	x_j	$i = j$	$i = j-1$	$i = j-2$	$i = j-3$
$j = 0$	0	$c_{0,0} = 1$			
$j = 1$	1	$c_{1,1} = 2$	$c_{0,1} = 1$		
$j = 2$	-1	$c_{2,2} = -1$	$c_{1,2} = \frac{3}{2}$	$c_{0,2} = -\frac{1}{2}$	
$j = 3$	2	$c_{3,3} = 4$	$c_{2,3} = \frac{5}{3}$	$c_{1,3} = \frac{1}{6}$	$c_{0,3} = \frac{1}{3}$

Nous remplissons le tableau ligne par ligne avec $c_{i,i} = f_i$ pour $i = 0..n$ et $c_{i,j} = \frac{c_{i+1,j}-c_{i,j-1}}{x_j-x_i}$ pour $0 \leq i < j \leq 3$. Ainsi, $D_{0,3}(x) = \frac{1}{3}(x-0)(x-1)(x+1) + (-\frac{1}{2})(x-0)(x-1) + x + 1 = \frac{1}{3}x^3 - \frac{1}{2}x^2 + \frac{7}{6}x + 1$.

1.2.3 Algorithme de Neville

L'algorithme de calcul du polynôme de Lagrange $P(x) = D_{0,n}(x)$ pour x fixé est constitué des n étapes suivantes :

Etape 1 Pour i allant de 0 à $n-1$, calculer

$$D_{i,i+1}(x) = \frac{f_{i+1}(x - x_i) - f_i(x - x_{i+1})}{x_{i+1} - x_i}$$

Etape 2 Pour i allant de 0 à $n-2$, calculer

$$D_{i,i+2}(x) = \frac{D_{i+1,i+2}(x)(x - x_i) - D_{i,i+1}(x)(x - x_{i+2})}{x_{i+2} - x_i}$$

$\vdots$

Etape $n-1$ Pour i allant de 0 à 1, calculer

$$D_{i,i+n-1}(x) = \frac{D_{i+1,i+n-1}(x)(x - x_i) - D_{i,i+n-2}(x)(x - x_{i+n-1})}{x_{i+n-1} - x_i}$$

Etape n Pour $i = 0$ Calculer $D_{i,i+n}(x) = P(x)$.

1.3 Complexité d'algorithmes

Que devient le temps de calcul si on multiplie la taille des données par 2 ? Pour répondre à cette question, on considère le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par l'algorithme. Ce nombre est couramment appelé complexité algorithmique et il s'exprime en fonction de la taille n des données. On dit que la complexité de l'algorithme est $O(h(n))$ où h est généralement une combinaison de polynômes, de logarithmes ou d'exponentielles. Ceci reprend la notation mathématique classique, et signifie que le nombre d'opérations effectuées est borné par $ch(n)$, où c est une constante, lorsque n tend vers l'infini.

La croissance de cette complexité est en fonction de la taille des données. Les algorithmes usuels peuvent être classés en un certain nombre de grandes classes de complexité (source wikipédia) :

- Les algorithmes sub-linéaires, dont la complexité est en général en $O(logn)$. C'est le cas de la recherche d'un élément dans un ensemble ordonné fini de cardinal n.
- Les algorithmes linéaires en complexité $O(n)$ ou en $O(nlogn)$ sont considérés comme rapides, comme l'évaluation de la valeur d'une expression composée de n symboles ou les algorithmes optimaux de tri.
- Plus lents sont les algorithmes de complexité située entre $O(n^2)$ et $O(n^3)$, c'est le cas de la multiplication des matrices et du parcours dans les graphes.

– Au delà, les algorithmes polynomiaux en $O(n^k)$ pour $k > 3$ sont considérés comme lents, sans parler des algorithmes exponentiels (dont la complexité est supérieure à tout polynôme en n) que l'on s'accorde à dire impraticables dès que la taille des données est supérieure à quelques dizaines d'unités.

1.4 Estimation de l'erreur

Pour évaluer l'erreur d'interpolation de Lagrange, on a le théorème suivant :

Théorème 1.4.1. *Soient f une fonction de classe C^{n+1} sur un intervalle $[a, b]$ et P le polynôme de Lagrange de f en les points $x_0 < x_1 < \cdots < x_n \in [a, b]$. Alors*

$$f(x) - P(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_n)}{(n + 1)!}\ f^{(n+1)}(\zeta) \tag{1.4.1}$$

où $a \leq min(x, x_0) < \zeta < max(x, x_n) \leq b$.

Démonstration : La relation est triviale pour $x = x_i\ \forall i = 0..n$. Soit $x \neq x_i$, pour tout $i = 0..n$. On pose $k(x) = \frac{f(x)-P(x)}{(x-x_0)(x-x_1)\dots(x-x_n)}$ et $w(t)$ un polynôme qui s'annule en x_0, x_1, …, x_n, et x définit par : $w(t) = f(t) - P(t) - (t - x_0)(t - x_1)\dots(t - x_n)k(x)$. Il s'annule en $(n + 2)$ points. En appliquant le thórème de Rolle, on montre que $w^{n+1}(t)$ s'annule en au moins un point ζ :

$$w^{n+1}(\zeta) = f^{n+1}(\zeta) - (n + 1)!\ k(x) = 0 \qquad et \qquad k(x) = \frac{f^{n+1}(\zeta)}{(n + 1)!} \qquad .$$

On remarque que l'erreur dépend de la fonction $f^{n+1}(\zeta)$ et des points d'interpolation par $(x - x_0)(x - x_1)\dots(x - x_n)$. Considérons le cas où les points d'interpolation sont équidistants, c'est-à-dire, $x_i = a + hi$ pour $i = 0..n$ avec $h = \frac{b-a}{n}$. Si f est continue et que toutes ses dérivées sont continues jusqu'à l'ordre $n + 1$, alors

$$\mid f(x) - P(x) \mid \leq \frac{1}{2(n + 1)}\left(\frac{b - a}{n}\right)^{n+1}\ max_{a\leq x\leq b} \mid f^{(n+1)}(x) \mid \tag{1.4.2}$$

Cette quantité ne tend pas nécessairement vers 0 car les dérivées $f^{(n+1)}$ de f peuvent grandir très vite lorsque n croît. Dans l'exemple suivant, nous présentons deux cas sur la convergence de l'interpolation de Lagrange.

Exemple 1.4.1. Soit $f(x) = \sin(x)$, $\mid f^k(x) \mid \leq\ 1$: l'interpolé P converge vers f quelque soit le nombre de points d'interpolation et leur emplacement. Par contre, pour $f(x) = \frac{1}{1+25x^2}$ sur $[-1, 1]$ et bien que f soit indéfiniment continûment dérivable sur $[-1, 1]$, les grandeurs

$$max_{-1\leq x\leq 1} \mid f^k(x) \mid \qquad , \qquad k = 1, 2, 3, \dots$$

explosent très rapidement et l'inégalité (1.4.2) ne nous assure plus la convergence de l'interpolation.

Le choix des points x_i apporte une amélioration sensible de l'interpolation : Lorsque $a \leq x \leq b$, on choisit les abscisses d'interpolation

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$$

pour $i = 0..n$, on obtient :

$$\mid f(x) - P(x) \mid \leq \frac{(b-a)^{n+1}}{(n+1)!2^{2n+1}}\, max_{x\in[a,b]} \mid f^{(n+1)}(x) \mid \quad .$$

Les points $x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$ pour $i = 0..n$ sont calculés à partir des zéros des polynômes de Tchebycheff ($T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$).

Nous venons de voir que l'interpolant de Lagrange peut diverger lorsque le nombre de points d'interpolation n devient grand et lorsque les points d'interpolation sont équidistants. Une première solution consiste à découper l'intervalle $[a, b]$ en plusieurs sous-intervalles, c'est l'interpolation par intervalles. On peut également utiliser des points judicieusement placés (zéros des polynômes de Tchebychev par exemple).

1.5 Interpolation Spline

Nous rencontrons les splines dans la majorité des programmes de CAO et il est certainement bon pour un ingénieur d'en connaître l'existence et les principes.

L'interpolation spline est une approche locale en faisant de l'interpolation par morceaux : sur un sous-intervalle donné, on fait de l'interpolation polynomiale de degré faible (≤ 3). Les splines réalisent une interpolation polynomiale par morceaux et on imposera aux noeuds un degré de régularité suffisant. Avec les splines, on obtient une très bonne approximation, sans effets de bord, contrairement à l'interpolation de Lagrange.

Etant donnée une fonction f définie sur $[a, b]$ et un ensemble de noeuds $a \leq x_0 < x_1 < \cdots < x_n \leq b$. Nous appelons *spline d'ordre* 3 ou *spline cubique* interpolant f aux noeuds, une fonction S vérifiant les conditions suivantes :

1. S coïncide avec un polynôme P_i de degré 3 sur chacun des intervalles $[x_{i-1}, x_i]$ pour $i = 1..n$.
2. $S(x_i) = P_i(x_i) = f_i$ pour $i = 0..n$.
3. $P_i(x_i) = P_{i+1}(x_i)$ pour $i = 0..n-1$, (continuité de S).
4. $P_i^{'}(x_i) = P_{i+1}^{'}(x_i)$ pour $i = 0..n-1$, (continuité de $S^{'}$).
5. $P_i^{''}(x_i) = P_{i+1}^{''}(x_i)$ pour $i = 0..n-1$, (continuité de $S^{''}$).
6. S satisfait une des conditions au bord suivantes :
   $S^{''}(x_0) = S^{''}(x_n) = 0$ (spline libre) ou bien $S^{'}(x_0) = f_0^{'}$ et $S^{'}(x_n) = f_n^{'}$ (spline forcée lorsque f est dérivable).

Proposition 1.5.1. *Les polynômes $P_i(x)$ sont données par les formules suivantes :*

$$P_i(x) = M_{i-1}\frac{(x_i-x)((x_i-x)^2-h_i^2)}{6h_i} + M_i\frac{(x-x_{i-1})((x-x_{i-1})^2-h_i^2)}{6h_i} + \frac{f_{i-1}(x_i-x)}{h_i} + \frac{f_i(x-x_{i-1})}{h_i}$$

avec $h_i = x_i - x_{i-1}$ pour $i = 1..n$, $M_0 = M_n = 0$ et $M_1, M_2, \ldots, M_{n-1}$ sont solutions du système linéaire :

$$\frac{h_i}{6}M_{i-1} + \frac{h_i+h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{f_{i+1}-f_i}{h_{i+1}} - \frac{f_i-f_{i-1}}{h_i} \qquad \text{pour } i = 1..n-1.$$

Indications sur la preuve : Pour montrer ces égalités, on commence par exprimer $P_i^{''}$ en fonction de $f_i^{''}$ et $f_{i+1}^{''}$ par une interpolation linéaire : $P_i^{''}(x) = f_i^{''}\frac{x_{i+1}-x}{x_{i+1}-x_i} + f_{i+1}^{''}\frac{x-x_i}{x_{i+1}-x_i}$. En intègrant deux fois, on obtient : $P_i(x) = f_i^{“}\frac{(x_{i+1}-x)^3}{6h_i} - f_{i+1}^{“}\frac{(x_i-x)^3}{6h_i} + a_i(x_{i+1}-x) - b_i(x_i-x)$ où a_i et b_i sont obtenues par les conditions d'interpolation :

$$a_i = \frac{f_i}{h_i} - f_i^{''}\frac{h_i}{6}, \qquad b_i = \frac{f_{i+1}}{h_i} - f_{i+1}^{''}\frac{h_i}{6}$$

En tenant compte des raccords aux noeuds pour $P_i^{'}$ et $i = 1..n-1$, on obtient un système linéaire de $(n-1)$ équations et $(n+1)$ inconnues.

$$\frac{h_i}{6}M_{i-1} + \frac{h_i+h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{f_{i+1}-f_i}{h_{i+1}} - \frac{f_i-f_{i-1}}{h_i} \qquad \text{pour } i = 1..n-1.$$

Exemple 1.5.1. On reprend l'exemple 1.1.1 de la section précédente. On cherche la spline cubique $S(x)$ telle que $S(0) = 1$, $S(1) = 2$, $S(-1) = -1$ et $S(2) = 4$. On pose $x_0 = -1$, $x_1 = 0$, $x_2 = 1$ et $x_3 = 2$. Ils sont bien ordonnées. Alors $f_0 = -1$, $f_1 = 1$, $f_2 = 2$, $f_3 = 4$ et $h_i = 1$ pour $i = 1..3$. On prend $M_0 = M_3 = 0$; M_1 et M_2 sont solution du système :

$$\begin{cases} \frac{2}{3} & M_1 & + & \frac{1}{6} & M_2 & = & -1 \\ \\ \frac{1}{6} & M_1 & + & \frac{2}{3} & M_2 & = & 1 \end{cases}$$

Nous obtenons :

$$S(x) = \begin{cases} P_1(x) = & -\frac{1}{3}x^3 - x^2 + \frac{4}{3}x + 1 & x \in [-1,0] \\ P_2(x) = & \frac{2}{3}x^3 - x^2 + \frac{4}{3}x + 1 & x \in [0,1] \\ P_3(x) = & -\frac{1}{3}x^3 + 2x^2 - \frac{5}{3}x + 2 & x \in [1,2] \end{cases}$$

1.6 Introduction à l'Approximation

Soit f une fonction réelle définie soit de façon discrète, soit de façon continue. Le problème général de l'approximation consiste à déterminer une fonction g de $\mathbf{R}$ dans $\mathbf{R}$ de forme donnée, qui, en un certain sens, approche le mieux possible la fonction f. Un cas particulier est l'interpolation polynomiale (la fonction g est alors un polynôme). Mais l'interpolation est un outil de peu de valeur pour le traitement des données expérimentales. On l'utilise seulement lorsque les données sont très précises. Cette section traite particulièrement de la *méthode des moindres carrés discrets.*

1.6.1 Résultats fondamentaux

Il est relativement fréquent d'avoir à ajuster sur des données une loi empirique ou encore d'avoir à déterminer la valeur des paramètres d'une loi théorique de façon à reproduire les résultats d'une expérience. Une façon possible pour aborder de tels problèmes est de chercher les valeurs de paramètres minimisant un critère.

Soient u_0, u_1, ..., u_p, $p+1$ élémets d'un espace vectoriel normé E. Pour tout $y \in E$, le problème de la meilleure approximation de y admet une solution, c'est-à-dire que la fonction *norme de l'erreur* $\epsilon(a_0, \ldots, a_n) = \| y - \sum_{i=0}^{n} a_i u_i \|^2$ admet un minimum a dans E. Pour cela, on montre que cette fonction est continue sur un fermé borné : elle atteint donc son minimum.

Notons V le sous-espace vectoriel engendré par les $p+1$ points $u_0, u_1, \ldots, u_p$ de E deux à deux distincts. La meilleure approximation de y sur V au sens des moindres carrés est $\sum_{i=0}^{i=p} a_i u_i$ où les a_i, $i = 0..p$ sont solutions du système linéaire :

$$\sum_{i=0}^{i=p} a_i \ < u_i, u_j > = < y, u_j > \qquad pour \qquad j = 0..p. \tag{1.6.1}$$

1.6.2 Approximation polynomiale discrète

Il y a une manière assez simple de trouver l'équation d'une courbe. On positionne les données sur un graphique, on deplace la règle jusqu'à ce que la ligne se trouve équidistante de tous les points. L'analyse numérique change cette supposition à vue d'oeil en un processus plus scientifique c'est l'approximation au sens des moindres carrés.

Notons $E = \mathbf{R}[x]$ l'espace vectoriel des polynômes en la variable x et à coefficient dans $\mathbf{R}$. Soit V l'espace vectoriel des polynômes de degré $\leq p$. On pose $u_0 = x^p$, $u_1 = x^{p-1}, \ldots, u_{p-1} = x$ et $u_p = 1$.

On considère les abscisses $x_0, x_1, \ldots, x_n \in \mathbf{R}$ et f une fonction de $\mathbf{R}$ dans $\mathbf{R}$ telle que $f(x_i) = y_i$ pour $i = 0..n$. On veut approcher l'ensemble discret des points $(x_i, y_i)_{0 \leq i \leq n}$ par un polynôme de degré p à coefficient dans $\mathbf{R}$:

$$P_p(x) = a_0 x^p + a_1 x^{p-1} + \cdots + a_{p-1} x + a_p \qquad .$$

Pour tout $g_1, g_2 \in V$, on définit le produit scalaire sur V par

$$< g_1(x), g_2(x) > = \sum_{i=0}^{i=n} g_1(x_i) g_2(x_i) \ .$$

Le système (1.6.1) est alors équivalent au système suivant :

$$\begin{pmatrix} \sum_{i=0}^{i=n} x_i^{2p} & \sum_{i=0}^{i=n} x_i^{2p-1} & \sum_{i=0}^{i=n} x_i^{2p-2} & \dots & \dots & \sum_{i=0}^{i=n} x_i^{p} \\ \sum_{i=0}^{i=n} x_i^{2p-1} & \sum_{i=0}^{i=n} x_i^{2p-2} & \dots & \dots & \dots & \sum_{i=0}^{i=n} x_i^{p-1} \\ \sum_{i=0}^{i=n} x_i^{2p-2} & & & & & \vdots \\ \vdots & & & & & \vdots \\ \vdots & & & & & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i^{p} & \sum_{i=0}^{i=n} x_i^{p-1} & \dots & \dots & \sum_{i=0}^{i=n} x_i & p+1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} \sum y_i x_i^p \\ \sum y_i x_i^{p-1} \\ \vdots \\ \vdots \\ \sum y_i x_i \\ \sum y_i \end{pmatrix}$$

Lorsque $p = 1$, on parle de *régression* et dans ce cas, on cherche une droite $P_1(x) = a_0x+a_1$ tel que

$$\sum_{i=0}^{i=n}(y_i - a_0x_i - a_1)^2 \leq \sum_{i=0}^{i=n}(y_i - \alpha_0x_i - \alpha_1)^2 \qquad \forall \alpha_0, \alpha_1 \in \mathbf{R} \qquad .$$

Le système (1.6.1) est alors équivalent à :

$$\begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_iy_i \\ \sum_{i=0}^{i=n} y_i \end{pmatrix} \qquad . \tag{1.6.2}$$

En posant $X = (x_0, x_1, \dots, x_n)$ et $Y = (y_0, y_1, \dots, y_n)$, on obtient à l'aide des formules de la covariance et de la moyenne :

$$\begin{array}{rcll} a_0 & = & \frac{\sigma(X,Y)}{\sigma(X^2)} & avec \quad \sigma(X,Y) = \frac{1}{n+1}\sum_{i=0}^{i=n}(x_i - \bar{X})(y_i - \bar{Y}) \\ a_1 & = & \bar{Y} - a_0\bar{X} & et \qquad \bar{X} = \frac{1}{n+1}\sum_{i=0}^{i=n} x_i \end{array}$$

Le point moyen $(\bar{X}, \bar{Y})$ appartient à la droite d'équation $P_1(x) = a_0x + a_1$.

Exemple 1.6.1. On considère la fonction $f(x) = x^2 + 2x$ sur $[-1, 1]$. Calculer les valeurs de f aux points d'abscisses $-1, -\frac{1}{2}, 0, \frac{1}{2}$ et 1. Donner la droite de meilleure approximation discrète de f pour ces 5 points au sens des moindres carrés. On note $x_0 = -1$, $x_1 = -\frac{1}{2}$, $x_2 = 0$, $x_3 = \frac{1}{2}$ et $x_4 = 1$. Alors $y_0 = -1$, $y_1 = -\frac{3}{4}$, $y_2 = 0$, $y_3 = \frac{5}{4}$ et $y_4 = 3$. On calcule $\sum_{i=0}^{i=4} x_i^2 = \frac{5}{2}$, $\sum_{i=0}^{i=4} x_i = 0$, $\sum_{i=0}^{i=4} x_iy_x = 5$ et $\sum_{i=0}^{i=4} y_i = \frac{5}{2}$. On obtient le système suivant à partir du système (1.6.2) :

$$\begin{pmatrix} \sum_{i=0}^{i=4} x_i^2 = \frac{5}{2} & \sum_{i=0}^{i=n} x_i = 0 \\ \sum_{i=0}^{i=n} x_i = 0 & n+1 = 5 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_iy_i = 5 \\ \sum_{i=0}^{i=n} y_i = \frac{5}{2} \end{pmatrix} \qquad .$$

et $a_0 = 2$, $a_1 = \frac{1}{2}$. Donc $P_1(x) = 2x + \frac{1}{2}$.

1.7 Exercices d'Applications

1. Implémenter l'algorithme de Neville dans le logiciel de calcul sage pour x fixé.

2. Construire le polynôme de Lagrange de $f(x) = e^{-x^2}$ sur les entiers de $[-2, 2]$ par la méthode des différences divisées.
3. Soit la fonction f de $\mathbf{R}$ dans $\mathbf{R}$ définie par $f(x) = \frac{1}{1+x^2}$.
   (a) Construire le polynôme de Lagrange P de f par la méthode des différences divisées sur 0, 1, 3 et 5.
   (b) Comparer $P(4)$ et $f(4)$.
   (c) Que peut-on conclure ?
4. On recueille les données suivantes concernant la météo dans la région de Bizerte le 13 janvier 2003 :

temps (heures)	0h00	5h00	7H00	12h00
pluviométrie (millimètres)	10	3	5	2

   (a) Utiliser la méthode d'Aitken pour calculer le polynôme interpolant la fonction "pluviométrie".
   (b) Approximer la pluviométrie dans la région de Bizerte au temps t=10h00.
5. Soit $f(x) = x + exp(-x^2)$. Donner la formule du polynôme d'interpolation de Lagrange $L_i(x)$ sur les points $(i, f(i))$ pour i fixé.
6. Poser $f(x) = \exp(-x^2)$, $a = -1$, $b = 2$ et

$$\mathbf{x}_k = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2k+1}{2(n+1)}\pi)$$

   pour k allant de 1 à 3.
   (a) Tracer l'ensemble des zéros du polynômes de Tchebycheff sur un même graphe que la fonction f.
   (b) Calculer le polynôme de Lagrange en ces points et comparer avec la fonction f.
7. Déterminer la spline cubique des points $(-1, 0)$, $(0, 1)$, $(1, 3)$ et $(2, 2)$. Comparer cette spline avec le polynôme de Lagrange en ces même points.
8. L'évolution de la concentration c d'une espèce dans un mélange est donnée par une loi linéaire en fonction du temps : $c(t) = a_0 t + a_1$. Un expérimentateur fait une série de mesures pour déterminer les paramètres inconnus :

0	1	2	3	4	5	6	7	8	9	10	11
1,54	3,09	4,97	7,43	10,65	14,92	20,6	28,2	38,42	52,15	70,65	96,6

   (a) On pose $n = 11$. Calculer $\sum_{i=0}^{i=n} t_i^2$, $\sum_{i=0}^{i=n} t_i$, $\sum_{i=0}^{i=n} t_i c_i$ et $\sum_{i=0}^{i=n} c_i$ et écrire le système pour l'approximation au sens des moindres carrées.
   (b) Utiliser la méthode de régression des moindres carrés pour obtenir les paramètres a_0 et a_1. Rappelons que

$$\mathbf{a}_0 = \frac{\sigma(X,Y)}{\sigma(X^2)} \; et \; \mathbf{a}_1 = \bar{Y} - a_0\bar{X} \; .$$

9. (a) Trouver le polyôme P du second degré passant par $(0,0)$, $(1,1)$, $(2,8)$.
   (b) La fonction x^3 prend les valeurs de P. Utiliser la formule vue en cours pour exprimer l'erreur d'interpolation commise en remplaçant x^3 par le polynôme d'interpolation P.
10. (a) Interpoler e^x aux points 0, $\frac{1}{2}$ et 1 par un polynôme du second degré.
    (b) Trouver une expression pour l'erreur d'interpolation et une borne de l'erreur indépendante de ζ.
11. (a) Sachant que la température initiale d'un objet est de $0^{\mathbf{o}}\mathbf{C}$, puis de $5^{\mathbf{o}}\mathbf{C}$ après une minute et $3^{\mathbf{o}}\mathbf{C}$ après deux minutes, quelle a été la température maximale et quant a-t-elle été atteinte ?
    (b) Quelle est la température moyenne entre 1 et 2 minutes ?
12. Considérons l'ensemble des points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 7, & f_1 = 2, & f_2 = -1, & f_3 = -14. \end{array}$$

    (a) Calculer les polynômes d'interpolation de Lagrange L_0 et L_3.
    (b) Vérifier que $L_i(x_i) = 1$ et que $L_i(x_j) \neq 0$ pour tout i et $j \neq i$ de 0 à 3.
    (c) Citer deux méthodes permettant de calculer le polynôme de Lagrange.
    (d) Vérifier que le polynôme de Lagrange sur les points (x_0, f_0), (x_1, f_1), (x_2, f_2) et (x_3, f_3) est égal à :

$$P(x) = -2x^3 + x^2 - 2x + 2 \qquad .$$

    (e) Afin de déterminer une approximation du nuage de points (x_i, f_i) pour i de 0 à 3, calculer la droite de régression linéaire P_0 par la méthode des moindres carrés discrets.
    (f) Quelle est la différence entre P et P_0.

Méthode d'Hermite. Soit $x_0, x_1, \ldots, x_n$, $(n+1)$ points de a, b et f une fonction de classe C^1 sur a, b. Au lieu de chercher à faire coïncider f et P_n aux points x_i, nous pouvons aussi chercher à faire coïncider f et P_n ainsi que leurs dérivées jusqu'à un ordre donné aux points x_i.
De la même manière que dans l'interpolation de Lagrange, nous cherchons un polynôme P_n tel que $P_n(x_i) = f(x_i)$ et $P_n'(x_i) = f'(x_i)$.

Chapitre 2

Intégration numérique

2.1 Introduction

Soit f une fonction continue d'un intervalle $[a, b]$ dans $\mathbf{R}$. On désire approcher numériquement la quantité

$$\int_a^b f(x)\, dx \qquad .$$

Pour cela, on subdivise l'intervalle $[a, b]$ en N intervalles : $x_i = a + ih$, $i = 0..N-1$ et $h = \frac{b-a}{N}$. On peut alors approcher $\int_a^b f(x)\, dx$ par la somme des airs des rectangles de longueur $f(\frac{x_{i+1}-x_i}{2})$ et de largeur $x_{i+1} - x_i$. C'est la *méthode des rectangles.*

Exercice 2.1.1. Calculer l'aire comprise entre l'axe des abscisses, l'axe des ordonnées, la droite $x = 1$ et la courbe $f(x) = e^{x^2}$ avec la méthode du rectangle.

La méthode de Newton-Côtes utilise les polynômes de Lagrange $P(x)$ pour approcher la fonction $f(x)$ sur $[a, b]$; on considère comme valeur approchée de l'intégrale de f entre a et b :

$$\int_a^b P(x)\, dx \qquad .$$

2.2 Formules de Newton-Côtes

Soit $n \in \mathbf{N}^*$. On considère la subdivision uniforme $x_i = a + ih$ avec $i = 0..n$ et $h = \frac{b-a}{n}$. Soit P le polynôme d'interpolation de Lagrange de degré $\leq n$ vérifiant $P(x_i) = f(x_i)$, $i = 0..n$. On obtient alors $\int_a^b f(x)\, dx \simeq \int_a^b P(x)\, dx = \int_a^b \sum_{i=0}^{i=n} f(x_i)\, L_i(x)\, dx$. En utilisant le changement de variables $x = a + ht$ dans les $L_i(x)$, on obtient :

$$\int_a^b P(x)\, dx = \sum_{i=0}^{i=n} h\, f(x_i)\, \alpha_i \qquad \text{et} \qquad \alpha_i = \int_0^n \prod_{i=0, k\neq i}^{k=n} \frac{t-k}{i-k} dt \qquad .$$

Les nombres rationnels α_i sont indépendants de f et de $[a, b]$.

Remarque 2.2.1. Si $f = 1$, $a = 0$ et $b = 1$, on obtient que le polynôme de Lagrange associé à f soit égal à 1, d'où :

$$1 = \int_0^1 dx = \sum_{i=0}^{i=n} h\ f(x_i)\ \alpha_i = h \sum_{i=0}^{i=n} \alpha_i \qquad .$$

Ce qui donne $\sum_{i=0}^{i=n} \alpha_i = n$.

Remarque 2.2.2. Pour $n = 1$, on obtient $\alpha_0 = \alpha_1 = \frac{1}{2}$, d'où

$$\int_a^b f(x)\ dx = \frac{b-a}{2}(f(a) + f(b)) \qquad .$$

Pour $n = 2$, on obtient $\alpha_0 = \frac{1}{3}$, $\alpha_1 = \frac{4}{3}$ et $\alpha_2 = \frac{1}{3}$. D'où :

$$\int_a^b f(x)\ dx = \frac{b-a}{6}(f(a) + 4f(a+h) + f(b)) \qquad .$$

2.2.1 Méthode du trapèze

Soit $N \in \mathbf{N}^*$. On considère la subdivision uniforme $x_i = a + ih$, $i = 0..N$ avec $h = \frac{b-a}{N}$. La méthode du trapèze consiste à appliquer la méthode de Newton-Côtes avec $n = 1$ sur chaque intervalle $[x_i, x_{i+1}]$, $i = 0..N-1$. On obtient :

$$\int_a^b f(x)\ dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)\ dx \simeq T(h) \qquad \text{et} \qquad T(h) = \sum_{i=0}^{N-1} \frac{h}{2}(f(x_i) + f(x_{i+1}))$$

Théorème 2.2.1. *La méthode du trapèze est d'ordre deux. Plus précisément, si f est suffisamment dérivable et $h = \frac{b-a}{N}$, alors il existe $c \in]a, b[$ tel que*

$$\int_a^b f(x)\ dx - T(h) = -\frac{b-a}{12} h^2 f''(c) \qquad .$$

Exercice 2.2.1. Reprendre l'exemple 2.1.1 avec la méthode du trapèze.

2.2.2 Méthode de Simpson

Soit $N \in \mathbf{N}^*$. On considère la subdivision uniforme $x_i = a + ih$, $i = 0..2N$ avec $h = \frac{b-a}{2N}$. La méthode de Simpson consiste à appliquer la méthode de Newton-Côtes avec $n = 2$ sur chaque intervalle $[x_{2i}, x_{2i+2}]$, $i = 0..N-1$. On obtient :

$$\int_a^b f(x)\ dx = \sum_{i=0}^{N-1} \int_{x_{2i}}^{x_{2i+2}} f(x)\ dx \simeq S(h)$$

et

$$S(h) = \sum_{i=0}^{N-1} \frac{h}{3}(f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2}))$$

Théorème 2.2.2. *La méthode de Simpson est d'ordre quatre. Plus précisément, si f est suffisamment dérivable et $h = \frac{b-a}{2N}$, alors il existe $c \in]a, b[$ tel que*

$$\int_a^b f(x)\,dx - S(h) = -\frac{b-a}{180}h^4 f^{(4)}(c) \qquad .$$

Exercice 2.2.2. Reprendre l'exemple 2.1.1 avec la méthode de Simpson.

2.2.3 Remarque sur la Formule Composite

Les méthodes du rectangle, du trapèze et de simpson sont des cas particuliers de la formule composite définie par :

$$\sum_{i=0}^{N-1} \frac{x_{i+1}-x_i}{2} \sum_{j=0}^{j=n} \alpha_j f(x_i + (x_{i+1}-x_i)\frac{t_j+1}{2}) \qquad .$$

Les $n+1$ points $-1 \leq t_0 \leq \cdots \leq t_n \leq 1$ sont appelés *points d'intégration* et les $n+1$ points $\alpha_0, \ldots, \alpha_n$ sont appeés *poids de la formule de quadrature.*

Remarque 2.2.3. Pour la méthode du rectangle, on prends $n = 0$, $t_0 = 0$ et $\alpha_0 = 2$. La formule composite devient

$$\sum_{i=0}^{N-1} \frac{x_{i+1}-x_i}{2} f(\frac{x_i + x_{i+1}}{2}) \qquad .$$

Remarque 2.2.4. Pour la méthode du trapèze, on prends $n = 1$, $t_0 = -1$, $t_1 = 1$, $\alpha_0 = 1$ et $\alpha_1 = 1$. La formule composite devient

$$\sum_{i=0}^{N-1} \frac{x_{i+1}-x_i}{2}(f(x_i) + f(x_{i+1})) \qquad .$$

Remarque 2.2.5. Pour la méthode de Simpson, on prends $n = 2$, $t_0 = -1$, $t_1 = 0$, $t_2 = 1$, $\alpha_0 = \frac{1}{3}$, $\alpha_1 = \frac{4}{3}$ et $\alpha_2 = \frac{1}{3}$. La formule composite devient

$$\sum_{i=0}^{N-1} \frac{x_{i+1}-x_i}{6}(f(x_i) + 4f(\frac{x_i + x_{i+1}}{2}) + f(x_{i+1})) \qquad .$$

2.3 Exemples d'Implémentations

Consulter le support de cours :
http ://dmawww.epfl.ch/rappaz.mosaic/Support/support/node12.html
Un TP/TD aura lieu pour reprendre les méthodes de ce chapitre.

Chapitre 3

Racine d'équation non linéaire

Les méthodes analytiques de résolution d'équations algébriques polynomiales sont limités à certaines formes et à de faible degré (équations quadratiques, cubiques, quartiques, $ax^{2n} + bx^n + c = 0$, etc...). La résolution des équations polynomiales de degré 4 est déjà plus pénible et à partir du degré 5, E. Galois a montré qu'il n'est pas toujours possible de donner une forme exacte des racines au moyen des coefficients (on utilise alors le groupe de l'équation, l'ideal des relations entre les racines ou encore le corps de décomposition du polynôme : ce sont des méthodes formelles [**?**]).

Pour la résolution des équations polynomiales, nous sommes amenés à l'utilisation des méthodes numériques. Il en est de même pour les équations contenant des fonctions transcendantes (i.e. non-algébriques) telles que $\exp(x)$, $\mathrm{ch}(x)$, etc... En étudiant les méthodes de résolution d'une équation f, on se posera deux questions :

1. La méthode converge-t-elle vers x^* où $f(x^*) = 0$. En d'autre termes la suite $(x_n)_{n\in\mathbf{N}}$ générée par la méthode converge-t-elle vers la solution x^*.
2. Dans l'affirmative, avec quelle rapidité (ceci permet de comparer les méthodes entre-elles).

3.1 Quelques algorithmes classiques

On désigne par (x_n) une suite de nombres réels $x_0, x_1, \ldots, x_n, \ldots$. Si une telle suite converge vers un point x^*, on écrira $(x_n) \longrightarrow x^*$. En analyse numérique, on construit les suites à l'aide d'algorithmes.

3.1.1 Méthode de la bissection

On considère une fonction f continue et on cherche x^* solution de $f(x) = 0$. On suppose qu'on a localisé par tâtonnement un intervalle $[a, b]$ dans lequel la fonction f change de signe : $f(a).f(b) < 0$.

On est donc certain qu'il y a entre a et b au moins un zéro de f. Pour approcher de façon précise l'un de ces zéros, on utilise l'algorithme suivant :

Algorithme 1 (Dichotomie).

Entrée : les extrémitées **a** et **b**,
la fonction **f**, la précision ϵ souhaitée et
N le nombre d'itérations maximum.
Sortie : Une approximation c de la racine x^* de **f**
ou bien un message d'erreur.
1. **Si** $f(a).f(b) > 0$ **Alors** Imprimer(*pas de solution*).
Sinon
$n := 1$;
2. **Tant que** $n \leq N$ **Faire**
3. $c := \frac{a+b}{2}$;
4. **Si** $f(c) = 0$ **ou** $\frac{b-a}{2} \leq \epsilon$ **Alors**
5. Imprimer(c) ; $n := N + 2$;
Fin Si ;
6. $n := n + 1$;
7. **Si** $f(a).f(c) > 0$ **Alors** $a := c$; **Sinon** $b := c$;
Fin Tant que ;
Si $n = N + 2$ **Alors** Imprimer(c)
Sinon Imprimer(*Après N itérations l'approximation de x^* obtenue est c et l'erreur maximale est* $\frac{b-a}{2}$) ;
Fin Si ;
Fin Si ;
Fin.

A chaque itération, l'algorithme construit un nouvel intervalle, autour de x^*, qui est de longueur égale à la moitié de la longueur de l'intervalle précédent. Au bout de n itérations, la longueur de l'intervalle est alors de $\frac{b-a}{2^n}$.

Le nombre d'itérations nécessaire pour obtenir une précision de calcul égal à ϵ est définie par :

$$n \geq \frac{\log(b-a) - \log \epsilon}{\log 2} \quad .$$

Remarque 3.1.1. Les avantages de cet algorithme sont la convergence assurée et une marge d'erreur sûre. Par contre il est relativement lent. Il est souvent utilisé pour déterminer une approximation initiale à utiliser avec un algorithme plus rapide.

3.1.2 Méthode de Newton

Cet algorithme se base sur la suite suivante (donner une explication géométrique) :

$$\begin{cases} x_0 & \text{donné} \\ x_{n+1} = & x_n - \frac{f(x_n)}{f'(x_n)}. \end{cases}$$

La signification géométrique consiste à mener par le point $(x_n, f(x_n))$ la tangente à la courbe $f(x)$. La pente de cette tangente est

$$T(x) = f(x_n) + (x - x_n)f'(x_n) \qquad .$$

Si on cherche le point d'intersection de la tangente avec l'axe des x, i.e. si on résoud $T(x) = 0$, on retrouve x_{n+1} tel que défini par la suite.

Théorème 3.1.1. *[Convergence globale de la méthode de Newton] Soit f une fonction de Classe C^2 sur $[a, b]$ et $g(x) = x - \frac{f(x)}{f'(x)}$. Si f vérifie :*

1. *$f(a).f(b) < 0$*
2. *$\forall x \in [a, b]$, $f'(x) \neq 0$ (c'est la stricte monotonie),*
3. *$\forall x \in [a, b]$, $f''(x) \neq 0$ (concavité dans le même sens).*

Alors en choisissant $x_0 \in [a, b]$ tel que $f(x_0).f''(x_0) > 0$, la suite (x_n) définie par x_0 et $x_{n+1} = g(x_n)$ converge vers l'unique solution de $f(x) = 0$ dans $[a, b]$.

Preuve : Les deux premières conditions du théorème assurent l'existence et l'unicité d'une racine x^* unique de f dans $[a, b]$. D'autre part, comme $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ on obtient :

$$\begin{aligned} x_{n+1} - x^* &= x_n - x^* - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - x^* + \frac{f(x^*) - f(x_n)}{f'(x_n)} \\ &= x_n - x^* + \frac{(x^* - x_n)f'(x_n) + \frac{(x^* - x_n)^2}{2} f''(\zeta_n)}{f'(x_n)} \end{aligned}$$

Ainsi :

$$\begin{aligned} x_{n+1} - x^* &= (x_n - x^*)(1 - \frac{f'(x_n) + \frac{x^* - x_n}{2} f''(\zeta_n)}{f'(x_n)}) \\ &= \frac{(x_n - x^*)^2}{2} \frac{f''(\zeta_n)}{f'(x_n)} \end{aligned}$$

Si f'' et f' sont de même signe, alors $x_{n+1} - x^* > 0$ et la suite est alors minorée par x^* à partir du rang 1. Sinon, si f'' et f' sont de signes contraires, alors $x_{n+1} - x^* < 0$ et la suite est alors majorée. De plus, et selon les cas, la suite va être soit décroissante ($f'' > 0$ et $f' > 0$ ou bien $f'' < 0$ et $f' < 0$) donc convergente soit croissante ($f'' > 0$ et $f' < 0$ ou bien $f'' < 0$ et $f' > 0$) et donc convergente.

Algorithme 2 (Newton).

Entrée : Une approximation initiale x_0, la précision ϵ souhaitée et **N** le nombre maximum d'itérations.
Sortie : Une approximation c de la racine x^* de f ou bien un message d'erreur.
1. $n := 1$;
2. **Tant que** $n \leq N$ **Faire**
   $c := x_0 - \frac{f(x_0)}{f'(x_0)}$;
3. **Si** $\mid c - x_0 \mid \leq \epsilon$ **Alors** c ; $n := N + 2$ **Fin Si** ;
   $n := n + 1$;
   $x_0 := c$;
   Fin Tant que ;
   Si $n = N + 2$ **Alors** Imprimer(c) **Sinon**
   Imprimer(*La méthode a échoué après N itérations*) ;
   Fin Si ;
Fin.

Remarque 3.1.2. Il est essentiel de fixer une limite au nombre d'itérations car la convergence n'est pas assurée. On verra dans la suite qu'on peut construire des exemples pour lesquels cet algorithme ne converge pas. Cependant lorsque la convergence aura lieu, elle sera trés rapide comme le montre le théorème 3.1.2. D'autre part, le test d'arrêt n'assure pas que la racine x^* soit à une distance ϵ de x.

Définition 3.1.1. La méthode définie par $x_{n+1} = g(x_n)$ est dite *d'ordre p* si la limite de $\mid \frac{e_{n+1}}{e_n^p} \mid$ lorsque $n \longrightarrow +\infty$ est une constante réelle strictement positive.

Théorème 3.1.2. *Soit f une fonction de Classe C^2. Si x^* est une racine simple de f, alors la méthode de Newton est au moins d'ordre 2.*

Preuve : On remarque que $g'(x) = \frac{f(x).f''(x)}{(f'(x))^2}$ donc $g'(x^*) = 0$. D'autre part :

$$\begin{aligned} x_{n+1} - x^* &= g(x_n) - g(x^*) \\ &= (x_n - x^*)^2 \frac{g''(x^*)}{2} + (x_n - x^*)^2 \epsilon(e_n) \\ &= e_n^2 \frac{g''(x^*)}{2} + e_n^2 \epsilon(e_n) \end{aligned}$$

Comme $\epsilon(e_n)$ est négligeable au voisinage de $+\infty$ on obtient le résultat. A chaque itération, on multiplie par 2 le nombre de chiffres exacts de l'approximation.
D'une façon générale, si f est de Classe C^p alors : $e_{n+1} = e_n g'(x^*) + \frac{e_n^2}{2!} g''(x^*) + \cdots + \frac{e_n^p}{p!} g^{(p)}(x^*) + e_n^p \epsilon(e_n)$. La méthode est alors d'ordre p si $g'(x^*) = \cdots = g^{(p-1)}(x^*) = 0$ et $g^{(p)}(x^*) \neq 0$.

Il est parfois ennuyeux dans l'utilisation de la méthode de Newton-Raphson d'avoir à calculer $f'(x)$. Il se peut par exemple qu'on ne dispose pas du programme d'ordinateur pour en effectuer le calcul alors qu'on peut facilement calculer $f(x)$. L'algorithme suivant peut être considéré comme une approximation de la méthode de Newton.

3.1.3 Méthode de la sécante

Au lieu d'utiliser la tangente au point x_n, nous allons utiliser la sécante passant par les points d'abscisses x_n et x_{n-1} pour en déduire x_{n+1} (donner une explication géométrique). L'équation de la sécante s'écrit sous la forme :

$$S(x) = f(x_n) + (x - x_n)\tau_n \qquad avec \qquad \tau_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \quad .$$

On définit x_{n+1} comme étant l'intersection de la sécante $S(x)$ avec l'axe des x : $S(x_{n+1}) = 0$ et on obtient

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \quad .$$

Remarque 3.1.3. La méthode de la sécante est une méthode itérative à deux pas en ce sens que x_{n+1} dépend de x_n et de x_{n-1}. Dans la méthode de Newton-Raphson x_{n+1} ne dépend que de x_n. C'est une méthode à un pas.

Une façon commode de procéder est de choisir x_0 arbitrairement puis de choisir $x_1 = x_0 + h$ où h est un accroissement petit. Si x_n et x_{n+1} sont proches (on peut espérer que ce sera le cas si la méthode converge), on peut considérer que $\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$ est une approximation de $f'(x_n)$. Dans ce sens l'algorithme de la sécante est une approximation de l'algorithme de Newton.

Algorithme 3 (Algorithme de la sécante).

Entrée : Deux approximations initiale x_0 et x_1,
la précision ϵ souhaitée et **N** le nombre maximum d'itérations
Sortie : Une approximation c de la racine x^* de f ou bien un message d'erreur.

1. $n := 2$;
   $q_0 := f(x_0)$;
   $q_1 := f(x_1)$;
2. **Tant que** $n \leq N + 1$ **Faire**
   $c := x_1 - q_1 \frac{x_1 - x_0}{q_1 - q_0}$;
3. **Si** $\mid c - x_1 \mid \leq \epsilon$ **Alors** Imprimer(c) ; **Fin.**
4. $n := n + 1$;
   $x_0 := x_1$; $x_1 := c$;
   $q_0 := q_1$; $q_1 := f(c)$;
   Fin Tant que ;
   Imprimer(*La méthode a échoué après N itérations*) ;

Fin.

3.1.4 Méthode du point fixe

On peut toujours transformer un problème du type $f(x) = 0$ en un problème de la forme $g(x) = x$ et ce d'une infinité de façons. Par exemple, la méthode de Lagrange est donnée par $x_{n+1} = g(x_n)$ où $g(x) = \frac{af(x) - xf(a)}{f(x) - f(a)}$ (on pose alors $x_0 = b$). Il faut toutefois noter que de telles transformations introduisent parfois des solutions parasites. Par exemple, on peut écrire $f(x) = \frac{1}{x} - 3 = 0$ ou encore $x = 2x - 3x^2 = g(x)$ et 0 est solution de la seconde équation mais pas de la première.

Le théorème suivant caractérise les fonctions g qui donnent des suites convergentes :

Théorème 3.1.3. *Si dans l'intervalle $[a, b]$, g vérifie les conditions suivantes :*

1. *$x \in [a, b]$ alors $g(x) \in [a, b]$,*
2. *l'application g est strictement contractante, c'est-à-dire $max_{x \in [a,b]} \mid g'(x) \mid = L < 1$.*

Alors pour tout $x_0 \in [a, b]$, la suite définie par $x_{n+1} = g(x_n)$ converge vers l'unique point fixe x^* de g sur $[a, b]$.

Il est souvent délicat de déterminer un intervalle $[a, b]$ dans lequel les hypothèses du théorème 3.1.3 sont vérifiées. Si on peut estimer $\mid g'(x^*) \mid$, on a le résultat suivant :

Théorème 3.1.4. *Soit x^* une solution de l'équation $x^* = g(x^*)$. Si g' est continue et $\mid g'(x^*) \mid < 1$ alors il existe un intervalle $[a, b]$ contenant x^* pour lequel la suite définie par $x_0 \in [a, b]$ et $x_{n+1} = g(x_n)$ converge vers x^*.*

Proposition 3.1.5. *Soit x^* une solution de l'équation $x^* = g(x^*)$. Si $g^{'}$ est continue au voisinage de x^* et si $\mid g^{'}(x^*) \mid > 1$ alors pour tout $x_0 \in [a, b]$ différent de x^*, la suite $x_{n+1} = g(x_n)$ ne converge pas vers x^*.*

3.2 Méthodes d'accélération de la convergence

On veut savoir si la suite définie par $x_{n+1} = g(x_n)$ converge rapidement et comment se comporte l'erreur $e_n = x_n - x^*$ d'une itération à la suivante ?

3.2.1 Procédé Δ^2 d'Aitken

Dans les méthodes d'ordre 1 (i.e. $g'(x^* \neq 0)$), on considère la suite

$$y_n = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n} \quad .$$

C'est une meilleure approximation de x^* que x_n. Lorsque x_n est proche de la solution x^*, $(x_{n+1} - x_n)^2$ et $x_{n+2} - 2x_{n+1} + x_n$ sont voisins de 0. Le calcul de y_n est alors instable. On utilise alors une autre écriture équivalente à la précédente mais plus stable :

$$y_n = x_{n+1} + \frac{1}{\frac{1}{x_{n+2} - x_{n+1}} - \frac{1}{x_{n+1} - x_n}} \quad .$$

3.2.2 Méthode de Steffensen

On poursuit les itérations en remplaçant x_n par y_n et on calcule ainsi $g(y_n) = z_n$ et $g(g(y_n)) = g(z_n)$. On construit formellement la méthode de Steffensen $z_{n+1} = G(z_n)$ où $G(x) = \frac{xg(g(x)) - g(x)^2}{g(g(x)) - 2g(x) + x}$. La méthode de Steffensen est d'ordre au moins 2 si (x_n) est d'ordre 1 et si (x_n) est d'ordre p alors (z_n) est d'ordre $2p - 1$.

3.3 Exercices d'Applications

1. Soit $f(x) = x^2 - \exp(-1 - x^2)$ une fonction définie sur $[0, 5]$.
   (a) Tracer f, f' et f'' sur $[0, 5]$.
   (b) Montrer qu'il existe une racine unique de f sur $[0, 5]$.
   (c) Appliquer la méthode de Newton avec $x_0 = 5$.
   (d) Est-ce que la méthode de Newton converge pour $x_0 = 0$? justifier votre réponse.
   (e) Appliquer la méthode de la sécante à f avec $x_0 = 0$. Que remarquez-vous ?
   (f) Trouver une fonction g vérifiant $g(x) = x$ si et seulement si $f(x) = 0$ qui assure les conditions du théorème du point fixe. Appliquer la méthode des itérations successives.
   (g) Citer deux procédés pour accélérer la convergence des algorithmes.
2. Refaire l'exercice précédent avec $f(x) = x^3 - 4x + 1$ sur $[0, 0.5]$ et $g(x) = x - f(x)$.
3. (a) Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2.
   (b) Ecrire l'algorithme de Newton qui prend en entrée les points x_0, $a < b$ et une fonction f et rend la racine de $f(x) = 0$ sur $[a, b]$ ou bien un message d'erreur.
   (c) Calculer la racine de $f(x) = -2x^3 + x^2 - 2x + 2$ sur l'intervalle $[0.5, 1]$ avec $x_0 = 1$. La précision des calculs est à 10^{-6} près.
4. Soit f une fonction de $\mathbf{R}$ dans $\mathbf{R}$ telle que

$$f(x) = x^3 + 3x^2 - 1 \qquad \textit{pour} \qquad x \in \mathbf{R}.$$

   (a) Montrer que l'équation $f(x) = 0$ admet une unique solution sur $[0, 1]$.
   (b) Nous désirons déterminer la solution de l'équation $f(x) = 0$ avec la méthode de Newton sur $[0, 1]$. En quoi consiste cette méthode ?
   (c) Ecrire l'algorithme général de la méthode de Newton qui prend en entrée les points x_0, $a < b$ et une fonction f et rend la racine de $f(x) = 0$ sur $[a, b]$ ou bien un message d'erreur.
   (d) Est-ce que la fonction f vérifie les conditions du théorème de convergence globale de la méthode de Newton pour $x_0 = 1$. Donner le théorème.
   (e) Calculer la racine de f sur l'intervalle $[0, 1]$ avec une erreur de 10^{-6}.
5. On cherche le zéro réel de $f(x) = x^3 + 4x^2 - 10$.
   (a) En choisissant $x_0 = 1.5$, appliquer la méthode du point fixe aux transformations suivantes :
      i. $g_1(x) = x - x^3 - 4x^2 + 10$
      ii. $g_2(x) = \frac{1}{2}\sqrt{10 - x^3}$
      iii. $g_3(x) = \sqrt{\frac{10}{4+x}}$

On remarquera que l'algorithme ne converge pas pour g_1, alors qu'il converge mais à des vitesses différentes pour les autres.

(b) Approximer la racine de f avec la méthode de Newton et donner un nombre d'itérations n qui donne une précision de x^* à 10^{-9} près.

6. Implémenter l'ensemble des algorithmes de ce chapitre sur sage.

Chapitre 4

Résolution de systèmes d'équations linéaires

Un des points essentiels dans l'efficacité des méthodes envisagées concerne la taille des systèmes à résoudre. Entre 1980 et 2000, la taille de la mémoire des ordinateurs a augmenté. La taille des systèmes qu'on peut résoudre sur ordinateur a donc également augmenté, selon l'ordre de grandeur suivant :

1980	Matrice pleine	$n = 10^2$
	Matrice creuse	$n = 10^6$
2000	Matrice pleine	$n = 10^6$
	Matrice creuse	$n = 10^8$

Le développement des méthodes de résolution de systèmes linéaires est lié a l'évolution des machines informatiques. Un grand nombre de recherches sont d'ailleurs en cours pour profiter au mieux de l'architecture des machines (méthodes de décomposition en sous domaines pour profiter des architectures parallèlles, par exemple).

De nombreux efforts ont été consacrés au développement de méthodes de résolution de systèmes d'équations linéaires. Les méthodes standards incluent l'élimination de Gauss-Jordan, et la décomposition LU. Les méthodes itératives telles que la méthode du gradient conjugué sont généralement préférées sur les larges systèmes d'équations.

L'analyse numérique matricielle s'intéresse principalement à deux types de problèmes :
– Résoudre un système linéaire $Ax = b$;
– Calculer les valeurs propres et vecteurs propres d'une matrice donnée.
Nous étudierons pour cela des algorithmes destinés à des matrices de grande taille. Comme il s'agit de calculs volumineux sur machine, à chaque opération des erreurs systématiques dites d'arrondi son générées. Il convient donc de se préoccuper de l'influence de ces petites erreurs sur la solution du problème et donc d'étudier la stabilité. Nous nous intéresserons d'autre part, au coût de ces algorithmes en termes d'opérations élémentaires.

4.1 Rappels et compléments sur les matrices

Les matrices diagonales et triangulaires jouent un rôle particulier en analyse numérique matricielle pour la raison évidente que les systèmes linéaires avec de telles matrices sont de résolution très simple. Aussi nous nous intéresserons ici aux changement de base permettant d'accéder à des structures de ce type.

Nous considérons un espace vectoriel de dimension n sur $\mathbf{R}$ ou sur $\mathbf{C}$. Toutes les matrices considérées dans la suite sont des matrices carrées.

Rappelons qu'une matrice est dite *régulière* si elle est inversible, sinon elle est *singulière*. Une matrice A est dite *normale* si $AA^* = A^*A$ où A^* est la matrice adjointe définie par $< Au, v >=< u, A^*v >$ pour tout $u, v \in \mathbf{C}^n$. Le signe $<,>$ dénote le produit scalaire dans $\mathbf{C}^n$, soit $< u, v >= {}^t\bar{u}v$. Une matrice *unitaire* est une matrice vérifiant $A^*A = AA^{=}I_n$. Elle est *hermitienne* si $A^* = A$.
Une matrice est dite symétrique si elle est réelle et si ${}^tA = A$ (dans ce cas, ${}^tA = A^*$). La matrice A est orthogonale si ${}^tAA = A\,{}^tA = I_n$.

Théorème 4.1.1. *Etant donné une matrice carrée A, il existe une matrice unitaire U telle que $U^{-1}AU$ soit triangulaire. Etant donné une matrice normale A, il existe une matrice unitaire U telle que $U^{-1}AU$ soit diagonale.*

4.2 Normes matricielles

Soit V un espace vectoriel de dimension n sur $\mathbf{R}$ ou $\mathbf{C}$, on peut définir plusieurs normes équivalentes sur V dont :

$$\begin{array}{lcll} \| x \|_1 & = & \sum_{i=1}^{i=n} |x_i|, & x = {}^t(x_1, x_2, \ldots, x_n) \in V \\ \| x \|_\infty & = & max_{i=1..n}|x_i| & \\ \| x \|_2 & = & \sqrt{\sum_{i=1}^{i=n} |x_i|^2} = \sqrt{{}^t\bar{x}x}, & \text{norme de schur} \\ \| x \|_p & = & (\sum_{i=1}^{i=n} |x_i|^p)^{\frac{1}{p}}. & \end{array}$$

Si V est muni d'une norme $\| \, . \, \|$, on définit sur l'ensemble des matrices carrées d'ordre n la norme suivante :

$$\| A \| = sup_{x \neq 0} \frac{\| Ax \|}{\| x \|} = max_{\|x\|=1} \| Ax \|$$

Cette quantité définit bien une norme puisqu'elle vérifie

$$\begin{array}{lcl} \| A \| = 0 & \Rightarrow & A = 0 \\ \| \lambda A \| & = & |\lambda| \, \| A \| \\ \| A + B \| & \leq & \| A \| \, \| B \| \end{array}$$

Nous considérons un système de n équations linéaires à n inconnues

$$\left(\begin{array}{llll} a_{1,1}x_1+a_{1,2}x_2+\ldots & +a_{1,n}x_n & = & b_1 \\ a_{1,1}x_1+a_{1,2}x_2+\ldots & +a_{1,n}x_n & = & b_1 \\ \vdots \qquad\qquad \vdots & \vdots & \vdots & \vdots \\ a_{1,1}x_1+a_{1,2}x_2+\ldots & +a_{1,n}x_n & = & b_1 \end{array}\right)$$

Les $a_{i,j}$ et les b_i étant fixés.
Ce système est équivalent à $Ax = b$ avec

$$A = \left(\begin{array}{lllll} a_{1,1} & a_{1,2} & \ldots & a_{1,j} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,j} & \ldots & a_{2,n} \\ a_{i,1} & a_{i,2} & \ldots & a_{i,j} & \ldots & a_{i,n} \\ a_{n,1} & a_{n,2} & \ldots & a_{n,j} & \ldots & a_{n,n} \end{array}\right) \left(\begin{array}{l} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{array}\right) = \left(\begin{array}{l} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{array}\right)$$

4.2.1 Conditionnement d'une matrice

Un problème est *bien conditionné* si une petite variation des données entraîne une petite variation sur la solution, *mal conditionné* dans le cas contraire.

Définition 4.2.1. On appelle *conditionnement* d'une matrice le nombre le nombre

$$c(A) = || A || \, . \, || A^{-1} ||$$

Soit le système $Ax = b$ et le système perturbé $\bar{A}\bar{x} = \bar{b}$. Les erreurs relatives sur A et b sont : $\epsilon_A = \frac{||\bar{A}-A||}{||A||}$, $\epsilon_b = \frac{||\bar{b}-b||}{||b||}$.

Théorème 4.2.1. *Si* $\epsilon_A c(A) < 1 \qquad : \qquad \frac{||\bar{x}-x||}{||x||} \leq \frac{c(A)}{1-\epsilon_A c(A)}(\epsilon_A + \epsilon_b)$.

Le théorème 4.2.1 donne une borne de l'erreur relative sur x : la solution du système $Ax = b$ risque d'être très sensible aux variations sur les données si le conditionnement est élevé.
On se propose d'examiner la sensibilité d'une méthode de résolution :

1. aux erreurs d'arrondis
2. aux "petites perturbations" des données du problème.

On mesurera cette sensibilité par un nombre appelé conditionnement, qui dépend d'une norme sur K^n. Lorsqu'il est très élevé, cela peut mettre cause la pertinence du résultat numérique obtenu.
On part d'un système $Ax = b$, de taille n×n, et on se donne par ailleurs ΔA et Δb, présumés petits par rapport à A et b et tel que $A + \Delta A = \overline{A}$soit aussi inversible. On se propose de comparer les solutions x et $\overline{x}$ des systèmes :

$Ax = b$ et le système perturbé $\overline{A}\overline{x} = \overline{b}$
Un problème est bien conditionné si une petite variation des données entraîne une petite variation sur la solution, mal conditionné dans le cas contraire.

Définition On appelle conditionnement d'une matrice le nombre

$$c(A) = \|A\| \ . \left\|A^{-1}\right\|$$

Soit le système $A?? = b$ et le système perturbé $\overline{A}\overline{x} = \overline{b}$. Les erreurs relatives sur A et b sont donnée par :

$$?_A = \frac{\|\Delta A\|}{\|A\|} = \frac{\left\|\overline{A} - A\right\|}{\|A\|}, ?_b = \frac{\|\Delta b\|}{\|b\|} = \frac{\left\|\overline{b} - b\right\|}{\|b\|}$$

On écrit aussi $\overline{x} - x = \Delta x$ et on se demande si le taux d'erreur sur x mesuré par $\frac{\|\Delta x\|}{\|x\|} = \frac{\|\overline{x} - x\|}{\|x\|}$ s'obtient à partir de $\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}$ par un "facteur d'amplification" multiplicatif raisonnable.
Théorème.
Le théorème suivant donne une borne de l'erreur relative sur x : la solution du système $Ax = b$ risque d'être très sensible aux variations sur les données si le conditionnement est élevé.
Si $?_A c(A) < 1$: $\frac{\|\overline{x} - x\|}{\|x\|} < \frac{c(A)}{1 - ?_A c(A)} \ (?_A + ?_b)$
Exemple : On souhaite résoudre le système linéaire $Ax = b$, où A et b sont donnés par :

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \text{ et } b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

la solution du système est : $x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

on résout le système perturbé $\overline{A}\overline{x} = \overline{b}$ avec $b = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix}$

alors on trouve

$$\overline{x} = \begin{pmatrix} 9.2 \\ -12.6 \\ 4.5 \\ -11 \end{pmatrix}$$

Autrement dit, de très petites variations sur b ont conduit à de grandes variations sur x.
Le conditionnement de A est c(A)=4488 >>> 1.
Ce phénomène de mauvais conditionnement explique pour partie la difficulté de prévoir certains phénomènes. Les appareils de mesure ne sont jamais parfaits, et il est impossible de connaître exactement b. Cela peut entrainer une très grande imprécision sur la valeur de x.

Les conditionnements de matrice utilisés dans la pratique correspond aux trois normes usuelles $\|.\|_p$, p=1, 2, 8. On note $c(A) = \|A\|_p * \|A^{-1}\|_p$
Si le conditionnement de A est de l'ordre 1 alors on dit que A est bien conditionnée sinon, elle est mal conditionnée

4.2.2 La méthode de Gauss

Exercice 1. On considère le système $Ax = b$ suivant :

$$\begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix}$$

$$\left(\begin{array}{ccc|c} 3 & -1 & 0 & 5 \\ -2 & 1 & 1 & 0 \\ 2 & -1 & 4 & 15 \end{array}\right) \begin{array}{l} \\ l_2 \leftarrow l_2 + \frac{2}{3} l_1 \\ l_3 \leftarrow l_3 - \frac{2}{3} l_1 \end{array} \Leftrightarrow \left(\begin{array}{ccc|c} 3 & -1 & 0 & 5 \\ 0 & 1/3 & 1 & 10/3 \\ 0 & -1/3 & 4 & 35/3 \end{array}\right) \begin{array}{l} \\ \\ l_3 \leftarrow l_3 + l_2 \end{array}$$

$$\Leftrightarrow \left(\begin{array}{ccc|c} 3 & -1 & 0 & 5 \\ 0 & 1/3 & 1 & 10/3 \\ 0 & 0 & 5 & 5 \end{array}\right)$$

Nous effectuons une élimination ascendante (back-substitution) ou une remontée afin de résoudre le nouveau système triangulaire obtenu par la méthode de Gauss :

$$\begin{cases} x_3 = \frac{15}{5} = 3 \\ x_2 = \frac{\frac{10}{3} - 1 * x_3}{\frac{1}{3}} = 1 \\ x_1 = \frac{5 - (-1 * x_2 + 0 * x_3)}{3} = 2 \end{cases}$$

Nous considérons un système de n équations linéaires à n inconnues

$$A = \begin{pmatrix} a_{1,1}x_1 + a_{1,2} + \cdots + a_{1,j}x_j + \cdots + a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 + a_{2,2} + \cdots + a_{2,j}x_j + \cdots + a_{2,n}x_n & = & b_2 \\ \vdots & & \vdots \\ a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,j}x_j + \cdots + a_{i,n}x_n & = & b_i \\ \vdots & & \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,j}x_j + \cdots + a_{n,n}x_n & = & b_n \end{pmatrix}$$

les $a_{i,j}$ et les $b_{i,j}$ étant fixés. Ce système est équivalent à $Ax = b$ et par combinaisons linéaires successives des lignes du système, on triangularise supérieurement. La résolution du système $A^{(1)}x = b^{(1)}$ avec $A^{(1)} = A$ une matrice inversible telle que $a_{1,1}^{(1)} \neq 0$: Annuler

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots a_{1j}^{(1)} & \dots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \dots a_{2j}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1}^{(1)} & a_{i2}^{(1)} & \dots a_{ij}^{(1)} & \dots & a_{in}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \dots a_{nj}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} \text{ et } b^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_i^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$$

l'élément en position $(i, 1)$ pour $i = 1, \dots, n$ revient à soustraire de la $i$ème ligne, la première ligne multipliée par $m_{i,1} = a_{j,1}^{(1)}/a_{1,1}^{(1)}$.

Cet ensemble d'éliminations revient à prémultiplier la matrice $A^{(1)}$ par une matrice M_1 vérifiant $a_{1,1}$ est appelé *élément pivot de la transformation* M_1 et on obtient un nouveau

$$\mathbf{M}_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -m_{21} & 1 & 0 & \cdots & 0 \\ -m_{31} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

système équivalent au système de départ : $M_1 A^{(1)} x = M_1 b^{(1)}$ où

$$\mathbf{A}^{(2)} = \mathbf{M}_1\,\mathbf{A}^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}, \quad \mathbf{b}^{(2)} = \mathbf{M}_1\,\mathbf{b}^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

les seuls éléments à calculer sont les

$$a_{ij}^{(2)},\ i,j = 2,3,\ldots,n;\ \text{et}\ b_i^{(2)},\ i = 2,3,\ldots,n.$$

Ce calcul s'effectue grâce aux formules suivantes :

$$\begin{cases} m_{i1} = a_{i1}^{(1)} / a_{11}^{(1)}, \quad i = 2,3,\ldots,n \\ b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}, i = 2,3,\ldots,n \\ a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, i,j = 2,3,\ldots,n \end{cases} \quad (*)$$

De la même manière, on réitère la procédure de l'élimination de Gauss en supposant que $a_{2,2}^{(2)} \neq 0$ selon les formules par application de $n-1$ transformations de même type de

$$\begin{cases} m_{i2} = a_{i1}^{(2)} / a_{22}^{(2)}, \quad i = 3,4,\ldots,n \\ b_i^{(3)} = b_i^{(2)} - m_{i2} b_2^{(2)}, i = 3,4,\ldots,n \\ a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}^{(2)}, i,j = 3,4,\ldots,n \end{cases} \quad (**)$$

M_1 et M_2 on obtient finalement, pour autant que les pivots successifs soient non nuls, le système : $M_{n-1}M_{n-2}\ldots M_2M_1A^{(1)}x = M_{n-1}M_{n-2}\ldots M_2M_1b^{(1)}$ équivalent au système $A^{(1)}x = b^{(1)}$, mais dont la matrice est triangulaire supérieure.

Complexité de calcul

Le coût de la méthode de Gauss est donné par :
$\sum_{k=1}^{n-1}(2(n-k)(n-k+1)+n-k) = \sum_{i=1}^{n-1}(2i(i+1)+i)$ opérations soit $\frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n$ auxquelles il faut encore ajouter n^2 opérations pour l'élimination ascendante (algorithme de remontée). La méthode de Gauss exige donc finalement le calcul de $\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$ opérations en tout.
Tout ceci n'est évidemment valable que si les pivots successifs des transformations sont non nuls.

Remarques

Si, au cours des calculs, on rencontre un pivot nul, il faut permuter la ligne de ce pivot avec une ligne d'indice supérieur, présentant un pivot non nul dans la même colonne. Ceci sera toujours possible si le système donné est compatible ($\det\mathbf{A} \neq 0$). En pratique, on ne permute pas les lignes entre-elles, on change simplement le numéro des lignes à permuter en stockant dans un vecteur de travail les indices des pivots successifs. Pour la "back-substitution" on tient compte de l'ordre utilisé lors de la triangularisation.

4.2.3 La factorisation LU d'une matrice

On dit qu'une matrice A(n,n) possède une factorisation LU si A=LU avec L une matrice(n,n) triangulaire inférieure à diagonale unité et U une matrice triangulaire supérieure.

Proposition 1(unicité)

On suppose que A est régulière et admet une factorisation A= LU, alors U est régulière et la factorisation est unique.

Preuve :

1. det(A)=det(LU)=det(L)det(U)=det(U)

si A est régulière ($det(A) \neq 0$) alors U l'est aussi.

1. Supposons que A=L_1U_1=L_2U_2

alors $L_2^{-1}L_1 = U_2U_1^{-1} = D$

Comme L_2 est une matrice triangulaire inférieure à diagonale unité donc L_2^{-1} triangulaire inférieur à diagonale unité.

De même, comme L_1 est une matrice triangulaire inférieure à diagonale unité donc $D = L_2^{-1}L_1$est une matrice triangulaire inférieure à diagonale unité**.**

d'autre part

U_1 est une matrice triangulaire supérieure donc U_1^{-1} est une matrice triangulaire supérieure et U_2 est une matrice triangulaire supérieure donc $D = U_2U_1^{-1}$ est une matrice triangulaire supérieure.

Ainsi D est la matrice unité, $L_1 = L_2$ et $U_1 = U_2$

Proposition 2 (existence)

On suppose que l'élimination de Gauss est faisable sans permutation de lignes sur un système de matrice A supposée régulière. Alors A possède une factorisation LU avec U régulière.

La méthode de Gauss fournit un système à matrice triangulaire supérieure :

$\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1\mathbf{A}\boldsymbol{x} = \mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1\boldsymbol{b} = \boldsymbol{b}_1$

Posons $\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1\mathbf{A} = \mathbf{U}$

Comme les $\mathbf{M}_i$ sont triangulaires inférieures à diagonale unité le produit $\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1$ est aussi triangulaire inférieure à diagonale unité et l'on a :

$\mathbf{A} = (\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1)^{--1}\mathbf{U} = \mathbf{LU}$

où $\mathbf{L} = (\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1)^{--1}$ est une matrice triangulaire inférieure à diagonale unitaire.

Théorème 1

La matrice A(n,n) posséde une factorisation LU avec U régulière si et seulement si ses sous-matrices principales A^k k=1...n sont régulières.

Construction de A

Si on connaît une factorisation (on dit aussi décomposition) **LU** de la matrice **A** du système $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, sa solution s'obtient de la façon suivante :

1°) on calcule la solution du système $\mathbf{L}\boldsymbol{y} = \boldsymbol{b}$
2°) on résout ensuite $\mathbf{U}\boldsymbol{x} = \boldsymbol{y}$
En effet : $\mathbf{A}\boldsymbol{x} = \mathbf{LU}\boldsymbol{x} = \mathbf{L}\boldsymbol{y} = \boldsymbol{b}$
La connaissance de la décomposition **LU** permet de résoudre le système de départ au prix de $2n^2$ opérations arithmétiques (*supplémentaires*) puisqu'il faut résoudre deux systèmes triangulaires à n inconnues.

Remarques
Le coût de la factorisation **LU** est du même ordre que celui de la méthode de Gauss soit $2n^2/3$.
La méthode ne fonctionne qu'avec les matrices carrés régulières (det $\mathbf{A} \neq 0$).
La méthode fournit la matrice inverse de **A** grâce à la relation $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$.
La matrice **U** est identique à la matrice triangulaire fournie par la méthode de Gauss
La matrice $\mathbf{L} = (\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_2\mathbf{M}_1)^{--1}$ s'écrit :

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ m_{31} & m_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{pmatrix}$$

où les M_{ik} sont les facteurs multiplicatifs rencontrés lors de la triangularisation de Gauss.

Exemple
Effectuer la factorisation LU de la matrice :

$$\mathrm{A} = \begin{pmatrix} 4 & -9 & 2 \\ 2 & -4 & 4 \\ -1 & 2 & 2 \end{pmatrix}$$

D'après les formules : m_{21}= 0.5 et m_{31}= -0.25 ce qui nous donne

$$\mathrm{A} = \begin{pmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 3 \\ 0 & -0.25 & 2.5 \end{pmatrix}$$

De la même manière m_{32}= -0.5 et nous obtenons que

$$\mathrm{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.25 & -0.5 & 1 \end{pmatrix} \text{ et } \mathrm{U} = \begin{pmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 3 \\ 0 & 0 & 4 \end{pmatrix}$$

4.2.4 La factorisation de Cholesky

Cette méthode s'applique uniquement aux matrices **symétriques définies positifs** et permet comme déjà mentionné la factorisation A= RR^T où R est une matrice triangulaire inférieure.

Définition : Soit A une matrice symétrique, on dit que A est définie positive si :

$\begin{cases} \forall\, \mathrm{x} \in \mathrm{IR}^{\mathrm{n}} \quad \mathrm{x}^{\mathrm{T}} A\, x \;>\; 0 \\ \mathrm{x}^{\mathrm{T}} A\, x = \mathrm{x} A\, x^{T} \;=\; 0 \Leftrightarrow \mathrm{x} \;=\; 0 \end{cases}$ $\begin{cases} \forall\, \mathrm{x} \in \mathrm{IR}^{\mathrm{n}} \quad \mathrm{x}^{\mathrm{T}} A\, x \;>\; 0 \\ \mathrm{x}^{\mathrm{T}} A\, x = \mathrm{x} A\, x^{T} \;=\; 0 \Leftrightarrow \mathrm{x} \;=\; 0 \end{cases}$

Théorème 1 : Une matrice A symétrique est définie positive si tous ses déterminants sont positifs

Théorème 2 : Si les valeurs propres d'une matrice A symétrique sont toutes strictement positives alors A est définie positive. De même si A est définie positive alors toutes ses valeurs propres sont strictement positives.

Rappel : Les valeurs propres d'une matrice triangulaire sont les éléments de sa diagonale.

Théorème 3 : Si A est une matrice symétrique définie positive, alors il existe au moins une matrice réelle triangulaire inférieure R tel que A=RR^T
de plus si $R_{ii} > 0$ $\forall$ i=1,2,...,n alors la factorisation est unique.

Construction de R par la méthode de Cholesky :

$$r_{11} = \sqrt{a_{11}}$$

$r_{i1} = \frac{a_{i1}}{r_{11}}$ pour $i = 2 \dots n$

$r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ik}^2}$ pour $i = 2 \dots n$

$r_{ji} = \frac{1}{r_{ii}} \left[a_{ji} - \sum_{k=1}^{i-1} r_{jk} * r_{ik} \right]$ pour $i = 2 \dots n$, $j = i+1 \dots n$

Exemple

Soit A = $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 \\ 1 & 5 & 14 & 14 \\ 1 & 5 & 14 & 15 \end{pmatrix}$

La matrice symétrique A décomposée par la méthode de Cholesky tel que A=RR^T avec

R = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 1 \end{pmatrix}$

Remarques

Le coût du calcul est de l'ordre de $n^2/2$ opérations arithmétiques et n extractions de racine carré, c'est plus économique que la factorisation **LU** (*pour une matrice symètrique*).
Cette méthode fournit un test efficace pour déterminer si une matrice symétrique est définie positive ou non :
Dans de très nombreux problèmes concrets, les systèmes à résoudre ont des matrices symétriques définies positives en raison même des phénomènes qu'ils modélisent. La méthode de factorisation de Cholesky revêt donc une importance particulière.
Toutefois, comme toutes les méthodes directes, son coût de calcul et les erreurs d'arrondis deviennent un handicap quasiment insurmontable pour des systèmes comptant de l'ordre de 200 équations. En pratique on est surtout amené à résoudre des systèmes de plusieurs dizaines de milliers d'équations dont les matrices, fort heureusement, sont généralement

creuses. La rèsolution de tels systèmes nécessite l'emploi de méthodes itératives.

4.2.5 Valeurs propres et vecteurs propres

Aujourd'hui, le calcul des valeurs et vecteurs propres est indispensable dans toutes les branches de la science, en particulier pour la solution des systèmes des équations linéaires, en théorie de stabilité, pour les questions de convergence de processus itératifs, et en physique et chimie (mécanique, circuits, cinétique chimique, équation de Schrödinger).
Il existe plusieurs méthodes pour la détermination des valeurs et vecteurs propres d'une matrice :

1. Méthode de la puissance
2. Méthode de la puissance inverse de Wielandt
3. Transformation sous forme de Hessenberg (ou tridiagonale)
4. L'itération orthogonale
5. L'algorithme QR

4.2.6 La méthode de la puissance

Définition 3.3.1.1. Une matrice A admet une valeur propre Dominante lorsqu'elle est la plus grande valeur propre en module :

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$$

Le vecteur propre associé est dit dominant.
Remarque. La méthode de la puissance permet de calculer la valeur et le vecteur propre dominants de A.
Théorème 3.3.1.1. Supposons que λ et ν sont respectivement la valeur propre et le vecteur propre d'une matrice A. Pour toute constante α, on a (λ - α) est la valeur propre de (A - αI_n) et de vecteur propre A
Preuve immédiate
Théorème 3.3.1.2. si λ et ν sont respectivement la valeur propre et le vecteur propre d'une matrice A inversible, alors si λ ? 0, on a $1/\lambda$ et ν sont respectivement la valeur propre et le vecteur propre de A^{-1}.
Preuve On a Av=λv, λ A^{-1}v = λ $A^{-1}\lambda$v = A^{-1}Av = v et donc A^{-1}v = $1/\lambda$ v.
Définition 3.3.1.2. Un vecteur propre est dit normalisé si la plus grande de ses coordonnées est égale à 1

Remarque Il est facile de transformer un vecteur propre $v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ en formant un

nouveau vecteur $v' = \frac{1}{c}\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ avec $c = v_j$ et

$$|v_j| = \max_{i=1..n} |v_i| \ .$$

Théorème de la méthode de puissance Supposons que la matrice A admette une valeur propre dominante et un vecteur propre normalisé v. Ces deux quantités peuvent être déterminées à partir la méthode de la puissance.

On commence par $X_0 = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$; il est important que X_0soit différent des autres vecteurs propres.

$X_0 = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_i v_i = \alpha v + \sum_{\mathbf{i=2}}^{\mathbf{n}} \alpha_i v_i$ où v est le vecteur normalisé

On génère alors une suite (X_k) de manière récursive en utilisant la définition suivante :

$$\begin{cases} Y_k = AX_k \\ X_{k+1} = C_{k+1} Y_k \end{cases}$$

où C_{k+1}est la plus grande des coordonnées en valeur absolue de Y_k

alors la suite (X_k) converge vers v (vecteur normalisé) et la suite C_k converge vers (valeur propre dominante).

$$\begin{cases} \lim_{k\to+\infty} X_k = v \\ \lim_{k\to+\infty} C_k \ = \end{cases}$$

Preuve

Sachant que A admet n valeurs propres associées à n vecteurs propres distincts v_j $1 \leq j \leq n$, linéairement indépendants normalisées et formant une base de dimension n.

On peut alors écrire X_0 dans cette base :

$$X_0 = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n, \alpha_1 \neq 0$$

on suppose que pour $X_0 = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, $\max_{i=1..n} |x_i| \ = 1$

$$Y_0 = AX_0 = A\left[\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n\right]$$

$$= \alpha_1 A v_1 + \alpha_2 A v_2 + \cdots + \alpha_n A v_n$$

$$= \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2 + \cdots + \alpha_n \lambda_n v_n$$

$$= \lambda_1 \left[\alpha_1 \frac{\lambda_1}{\lambda_1} v_{\ 1} + \alpha_2 \frac{\lambda_2}{\lambda_1} v_{\ 2} + \cdots + \alpha_n \frac{\lambda_n}{\lambda_1} v_n \right]$$

$$= \lambda_1 \left[\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right) v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right) v_n \right]$$

en normalisant, on obtient :

$$X_1 = \frac{\lambda_1}{\mathbf{C}_1} \left[\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right) v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right) v_n \right]$$

après k itérations

$$Y_{k-1} = AX_{k-1}$$

$$= A \left[\frac{(\lambda_1)^{\mathrm{k}-1}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_{k-1}} \left[\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^{\mathrm{k}-1} v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^{\mathrm{k}-1} v_n \right] \right]$$

$$= \left[\frac{(\lambda_1)^{\mathrm{k}-1}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_{k-1}} \left[\alpha_1 \mathrm{A} v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^{\mathrm{k}-1} A v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^{\mathrm{k}-1} A v_n \right] \right]$$

$$= \left[\frac{(\lambda_1)^{\mathrm{k}-1}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_{k-1}} \left[\alpha_1 \lambda_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)_2^{\mathrm{k}-1} v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^{\mathrm{k}-1} {}_n v_n \right] \right]$$

$$= \left[\frac{(\lambda_1)^{\mathrm{k}}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_{k-1}} \left[\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^{\mathrm{k}} v_{\ 2} + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^{\mathrm{k}} v_n \right] \right]$$

or $\frac{\lambda_j}{\lambda_1} < 1 \ \forall\ j \in [2,\ \mathrm{n}]$ alors $\lim_{k \to +\infty} \alpha_j \left(\frac{\lambda_j}{\lambda_1} \right)^{\mathrm{k}} = 0 \quad \forall\ \mathrm{j} \in [2,\ \mathrm{n}]$
ainsi

$$\lim_{k \to +\infty} X_{\mathrm{k}} = \frac{\alpha_1 \, (_1)^{\mathrm{k}}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_k} \ v_1$$

comme X_{k} et v_1 sont normalisés, on a :
$\lim_{k \to +\infty} \frac{\alpha_1 (_1)^{\mathrm{k}}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_k} = 1$ (éq 3.3.1)
donc

$$\lim_{k \to +\infty} X_{\mathrm{k}} = v_1$$

on écrit l'équation 3.3.1 pour l'étape (k-1), on aura :
$\lim_{k \to +\infty} \frac{\alpha_1 (_1)^{\mathrm{k}-1}}{\mathbf{C}_1 \mathbf{C}_2 \ldots \mathbf{C}_{k-1}} = 1$ (eq 3.3.2)
en effectuant la division de l'eq 3.3.1 par l'eq 3.3.2, on aura :

$$\lim_{k \to +\infty} \frac{1}{\mathbf{C}_k} = 1$$

$$\lim_{k \to +\infty} \mathbf{C}_k = {}_1$$

Chapitre 5

Equations différentielles ordinaires

5.1 Rappel

On dit qu'une fonction g définie sur un intervalle $[a, b]$ est *lipschitzienne de constante* L si

$$\mid g(x) - g(x') \mid \leq L \mid x - x' \mid \qquad \forall\, x, x' \in [a, b]$$

En pratique, la fonction g est de classe C^1 sur $[a, b]$ et cette inégalité est une conséquence de la formule des accroissements finis :

$$g(x) - g(x') = g'(c)(x - x') \qquad c \in (x, x')$$

et on pose $L = max_{x\in[a,b]} \mid g'(x) \mid$.

Lemme 5.1.1. *1. On montre par récurrence que si $u_{n+1} \leq u_n\ \alpha + \beta$ pour tout n alors $u_n \leq u_0\ \alpha^n + \beta\frac{\alpha^n - 1}{\alpha - 1}$.*

2. On montre grâce à une étude de fonction que si $X > 0$ alors $(1 + X)^n \leq e^{nX}$.

5.2 Problème de Cauchy

Soient f une fonction de $[a, b] \times \mathbf{R}$ dans $\mathbf{R}$ continue et $\tau \in \mathbf{R}$. Le problème de Cauchy ou *problème de la condition initiale* est le suivant : existe-t-il une fonction $y(x)$ définie sur $[a, b]$ à valeurs dans $\mathbf{R}$ dérivable sur $[a, b]$ et vérifiant :

$$\left\{ \begin{array}{lcl} y(a) & = & \tau \\ y(x) & = & f(x, y(x)) \end{array} \right. \tag{5.2.1}$$

Théorème 5.2.1 (admis)**.** *On supprose que f est lipschitzienne par rapport à sa deuxième variable uniformément par rapport à sa première variable. Il existe donc $L > 0$ tel que*

$$\mid f(x, z_1) - f(x, z_2) \mid \leq L \mid z_1 - z_2 \mid \qquad , \ \forall x \in [a, b]\ , z_1, z_2 \in \boldsymbol{R}. \tag{5.2.2}$$

Alors le probmème de Cauchy admet une solution unique.

La condition (5.2.2) sera satisfaite si f admet une dérivée partielle par rapport à sa deuxième variable qui soit bornée dans $[a,b] \times \mathbf{R}$. On obtient grâce au Théorème des Accroissements Finis :

$$\begin{aligned} f(x,z_1) - f(x,z_2) &= \frac{\partial f}{\partial z}(x,z_c)(z_1 - z_2) \\ \Rightarrow \mid f(x,z_1) - f(x,z_2) \mid &\leq max_{x\in[a,b]} \mid \frac{\partial f}{\partial z}(x,z_c) \mid \mid z_1 - z_2 \mid \quad . \end{aligned}$$

On présente dans les sections suivantes des méthodes d'approximation de la solution $y(x)$ de (5.2.1). Pour cela, on subdivise l'intervalle $[a,b]$ uniformément et on approche la fonction y en ces points.
Soit n un entier naturel non nul. On pose $x_i = a + ih$ avec $h = \frac{b-a}{n}$ et $i = 0..n$. On va calculer une approximation y_i de $y(x_i)$ pour $i = 0..n$ et estimer l'erreur d'approximation $e_i = y_i - y(x_i)$.

5.3 Méthode d'Euler

Partant de $x_0 = a$, on connaît $y_0 = y(x_0) = \tau$. L'idée de la méthode d'Euler est de considérer que la courbe y sur $[x_0, x_1]$ n'est pas très éloignée de sa tangente en x_0 : une estimation rationnelle de $y(x_1)$ est $y_1 = y_0 + hf(x_0, y_0)$. En effet, pour x_1, l'équation de la tangente est $y'(x_0)(x - x_0) + y_0$. On estime alors que $y'(x_1)$ n'est pas très différent de $f(x_1, y_1)$. Sur $[x_1, x_2]$, on remplace la courbe par sa tangente approchée en x_1 ce qui donne l'estimation y_2 de $y(x_2)$: $y_2 = y_1 + hf(x_1, y_1)$ et on poursuit de la même manière.
Algorithme d'Euler :

Entrée : n, τ, a et b, et la fonction f.
Sortie : Une approximation de yx.
$x_0 := a$;
$y_0 := \tau$;
$h := \frac{b-a}{n}$;
Pour i **allant de** 0 **à** n **Faire**
$x_{i+1} := x_i + h$;
$y_{i+1} := y_i + hf(x_i, y_i)$;
Fin Pour ;
Imprimer(*La méthode d'Euler donne une approximation de y en* (x_i, y_i), $i = 0..n$).
Fin.

Théorème 5.3.1. *Si f est continue sur $[a,b] \times \boldsymbol{R}$, lipshitzienne par rapport à sa deuxième variable uniformément par rapport à sa première variable et si la soltion y de l'EDO est de Classe C^2 sur $[a,b]$, alors :*

$$\mid e_n \mid \leq \frac{(e^{L(b-a)} - 1)}{L} \frac{max_{x\in[a,b]} \mid y''(x) \mid}{2} h \quad .$$

Preuve : Voir la démonstration du théorème 5.5.1.

Définition 5.3.1. Une méthode de calcul numérique qui fournit une erreur $\mid e_n \mid \leq kh^p$ est dite *d'ordre p*. La méthode d'Euler est d'ordre 1.

Une méthode d'ordre 1 ne converge pas assez vite pour donner des résultats pratiques intéressants. On va étudier dans les deux sections suivantes deux méthodes d'ordre plus élevé.

5.4 Méthode de Taylor d'ordre 2

L'idée consiste à remplacer la courbe y sur un intervalle $[x_i, x_{i+1}]$ par une parabole (et non pas une droite comme dans la méthode d'Euler).
En utilisant la formule de Taylor sur y au voisinage de x_n, on obtient : $y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{6}y^{(3)}(c_n)$ avec $c_n \in [x_n, x_{n+1}]$. Comme $y'(x) = f(x, y(x))$, on a : $y''(x) = (\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial z})(x, y(x))$. On pose $y_{n+1} = y_n + hf(x_n, y_n) + \frac{h^2}{2}(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial z})(x_n, y_n)$.
Algorithme de Taylor d'ordre 2 :

Entrée : n, τ, a et b, et la fonction f.
Sortie : Une approximation de yx.

$x_0 := a$;
$y_0 := \tau$;
$h := \frac{b-a}{n}$;
Pour i **allant de** 0 **à** n **Faire**
  $x_{i+1} := x_i + h$;
  $y_{i+1} := y_i + hf(x_i, y_i) + \frac{h^2}{2}(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial z})(x_i, y_i)$;
Fin Pour ;
Imprimer(*La méthode de Taylor d'ordre 2 donne une approximation de y en* (x_i, y_i), $i = 0..n$).

Fin.

Théorème 5.4.1. *Si f est de Classe* C^2 *sur* $[a, b] \times \boldsymbol{R}$, *lipshitzienne par rapport à sa deuxième variable uniformément par rapport à sa première variable (avec la constante L) et si* $\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial z}$ *est lipshitzienne par rapport à sa deuxième variable uniformément par rapport à sa première variable (avec la constante* L_1*), alors pour* $h \in [0, h_0]$ *on a :*

$$\mid e_n \mid \leq \frac{(e^{(b-a)(L+\frac{h_0}{2}L_1)} - 1)}{L} \frac{max_{x\in[a,b]} \mid y^{(3)}(x) \mid}{6} h^2 \quad .$$

Preuve : Voir la démonstration du théorème 5.5.1.

5.5 Méthode de Runge-Kutta

Définition 5.5.1. Une méthode générale à un pas est définie par $y_{n+1} = y_n + h.\phi(x_n y_n, h)$ où ϕ est continue sur $[a, b] \times \mathbf{R} \times [0, h_0]$ lipshitzienne par rapport à sa seuxième variable uniformément par rapport à la première et à la troisième de constante L_ϕ.

Théorème 5.5.1. *Soit une méthode à un pas définie par* $y_{n+1} = y_n + h.\phi(x_n y_n, h)$ *où* ϕ *est continue sur* $[a,b] \times \boldsymbol{R} \times [0,h_0]$ *lipshitzienne par rapport à sa seuxième variable uniformément par rapport à la première et à la troisième de constante* L_ϕ *et telle que*

$$\frac{y(x+h) - y(x)}{h} - \phi(x, y(x), h) \leq kh^p$$

où y *est solution de l'EDO* $y'(x) = f(x, y(x))$ *et* f *de Classe* C^p *sur* $[a,b] \times \boldsymbol{R}$*. Alors*

$$| e_n | \leq K\ h^p \qquad .$$

Annexe A

Travaux Pratiques sur sage

Les codes suivants représentent des exercices corrigés ou des Projets de Fin d'années que nous avons encadrés en Analyse Numérique. Il s'agit de problèmes liés à l'interpolation polynomiale, à la recherche de solutions d'équations non linéaires et aux problèmes d'Analyse Numérique matricielle.

A.1 Interpolation

TP2 : Estimation de l'erreur

Théorème : Soient f une fonction de classe C^{n+1} sur un intervalle $[a, b]$ et P le polynôme de Lagrange de f en les points $x_0 < x_1 < \cdots < x_n \in [a, b]$. Alors

$$f(x) - P(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_n)}{(n + 1)!}\ f^{(n+1)}(\zeta) \tag{A.1.1}$$

où $a \leq min(x, x_0) < \zeta < max(x, x_n) \leq b$.

Démonstration : La relation est triviale pour $x = x_i\ \forall i = 0..n$. Soit $x \neq x_i$, pour tout $i = 0..n$. On pose $k(x) = \frac{f(x)-P(x)}{(x-x_0)(x-x_1)...(x-x_n)}$ et $w(t)$ un polynôme qui s'annule en x_0, x_1, ..., x_n, et x définit par : $w(t) = f(t) - P(t) - (t - x_0)(t - x_1)\dots(t - x_n)k(x)$. Il s'annule en $(n + 2)$ points. En appliquant le thórème de Rolle, on montre que $w^{n+1}(t)$ s'annule en au moins un point ζ :

$$w^{(n+1)}(\zeta) = f^{(n+1)}(\zeta) - (n + 1)!\ k(x) = 0 \qquad et \qquad k(x) = \frac{f^{(n+1)}(\zeta)}{(n + 1)!} \qquad .$$

Cette quantité ne tend pas nécessairement vers 0 car les dérivées $f^{(n+1)}$ de f peuvent grandir très vite lorsque n croît. Dans l'exemple suivant, nous présentons deux cas sur la convergence de l'interpolation de Lagrange.

Exemple : Soit $f(x) = \sin(x)$, $\mid f^k(x) \mid \leq 1$: l'interpolé P converge vers f quelque soit le nombre de points d'interpolation et leur emplacement. Par contre, pour $f(x) = \frac{1}{1+14x^2}$ sur $[-1, 1]$ et bien que f soit indéfiniment continûment dérivable sur $[-1, 1]$, les grandeurs

$$max_{-1\leq x\leq 1} \mid f^k(x) \mid \quad , \quad k = 1, 2, 3, \ldots$$

explosent très rapidement et ceci ne nous assure plus la convergence de l'interpolation.

Le choix des points x_i apporte une amélioration sensible de l'interpolation : Lorsque $a \leq x \leq b$, on choisit les abscisses d'interpolation

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$$

pour $i = 0..n$, on obtient :

$$\mid f(x) - P(x) \mid \leq \frac{(b-a)^{n+1}}{(n+1)!2^{2n+1}}\, max_{x\in[a,b]} \mid f^{(n+1)}(x) \mid \quad .$$

Les points $x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$ pour $i = 0..n$ sont calculés à partir des zéros des polynômes de Tchebychev ($T_n(x) =$ vérifiant $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$).

Exercice à faire sur SAGEmath :

1. Soit $f(x) = sin(x)$ et une subdivision de l'intervalle $[-1, 1]$ en $n = 5$ sous-intervalles equidistants. Posons $x_0 = -1$, $x_1 = -0.5$, $x_2 = 0$, $x_3 = 0.5$ et $x_4 = 1$.
   Implémenter l'algorithme de Lagrange se basant sur les polynômes d'interpolation de base. On prendra $f_i = f(x_i)$ pour $i = 0..n$.
   Dessiner dans un même graphique la fonction f, le polynôme P ainsi que les points (x_i, f_i). Reprendre les calculs pour $n = 13$.
2. Refaire le même travail avec la fonction $f(x) = \frac{1}{1+14x^2}$, puis avec $x_i = \cos(\frac{2i+1}{2n+2}\pi)$ pour $i = 0..n$ et $n = 5$ puis $n = 13$.

Correction sur SAGEmath :

1. Le code sous SAGEmath est donné ci-dessous :

```
sage: from pylab import arange
sage: xdata=arange(-1.0,1.1,0.5)
sage: f=sin
sage: P=lagrange(f, xdata)
sage: R=plot(f, -1,1, rgbcolor=(0,2,1))
sage: Q=plot(P, -1,1, rgbcolor=(0,1,2))
```

```
sage: n=len(xdata)-1
sage: s=Graphics()
sage: for i in range(n+1):
...           xi = xdata[i]
...           fi = f(xdata[i])
...           s = s + point((xi,fi), rgbcolor = (1,0,0))
sage: R+Q+s
```

Les graphiques suivants donnent la fonction *sin* et son polynoôme de Lagrange sur $n = 5$ puis sur $n = 13$ points :

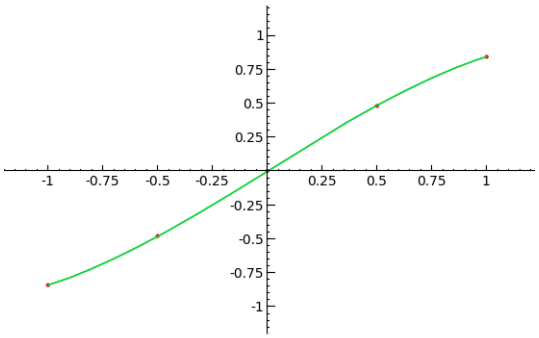


FIGURE A.1 – *Interpolation pour* $n = 5$

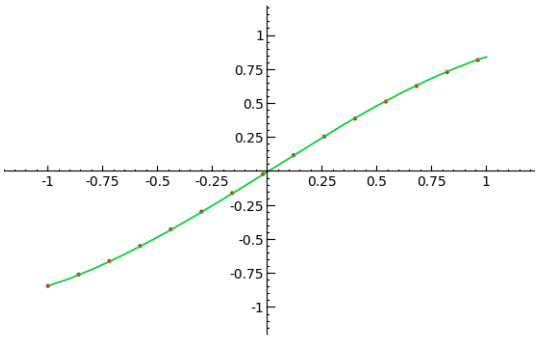


FIGURE A.2 – *Interpolation pour* $n = 13$

2. Si on calcule le polynôme de Lagrange sur la fonction $f = \frac{1}{1+14x^2}$, nous obtenons :

```
sage: from pylab import arange
sage: xdata=arange(-1.0,1.1,0.5)
sage: f=1/(1+14*x^2)
sage: P=lagrange(f, xdata)
sage: R=plot(f, -1,1, rgbcolor=(0,2,1))
sage: Q=plot(P, -1,1, rgbcolor=(0,1,2))
sage: n=len(xdata)-1
sage: s=Graphics()
sage: for i in range(n+1):
...           xi = xdata[i]
...           fi = f(xdata[i])
...           s = s + point((xi,fi), rgbcolor = (1,0,0))
...
sage: R+Q+s
```

Enfin, si on considère à la place des abscisses x_i équidistants, les zéros de Tchebychev, nous obtenons la figure 5 pour $n = 5$ points et ensuite la figure 6 pour $n = 13$:

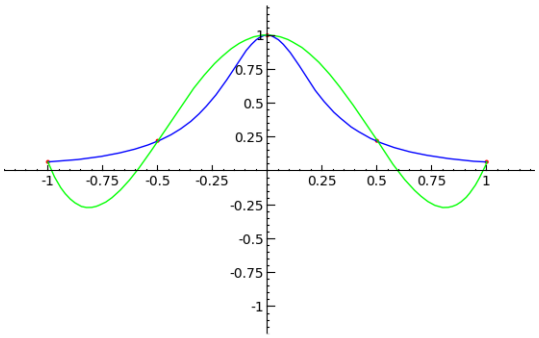


FIGURE A.3 – *Interpolation pour* $n = 5$

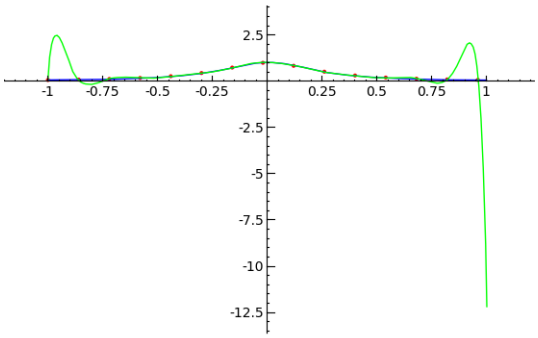


FIGURE A.4 – *Interpolation pour* $n = 13$

```
sage: a=-1; b=1; n=5;
sage: ydata=[simplify((a+b)/2+(b-a)/2*cos((2*k+1)*3.14/(2*n+2))) for k in range(n+1)]
sage: print(ydata)
[0.965960168538399, 0.707388269167200, 0.259459981914882, -0.257921542147459,
-0.706261644820005, -0.965546938710468]
sage: f=1/(1+14*x^2)
sage: P=lagrange(f, ydata)
sage: A=plot(f, -1,1, rgbcolor=(0,2,1))
sage: B=plot(P, -1,1, rgbcolor=(1,0,0))
sage: A+B
```

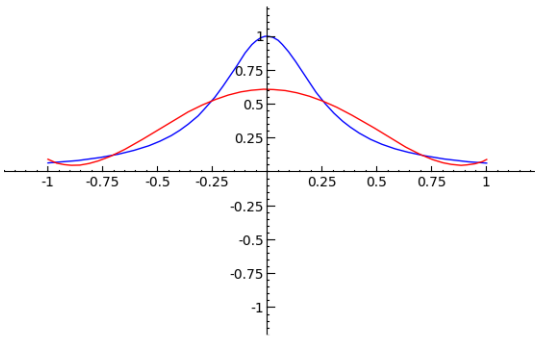


FIGURE A.5 – *Interpolation sur les zéros de Tchebychev pour* $n = 5$

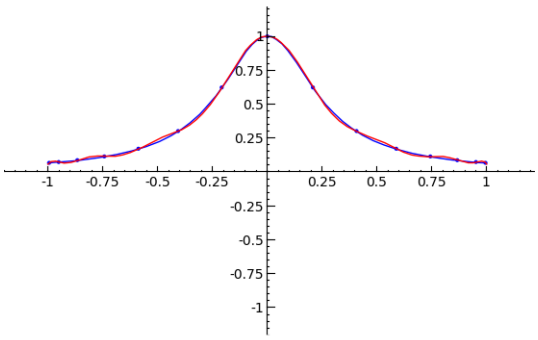


FIGURE A.6 – *Interpolation sur les zéros de Tchebychev pour* $n = 13$

```
sage: a=-1; b=1; n=13;
sage: ydata=[simplify((a+b)/2+(b-a)/2*cos((2*k+1)*3.14/(2*n+2))) for k in range(n+1)]
sage: print(ydata)
[0.993718576879459, 0.943939675865892, 0.846875476196380, 0.707388269167200,
```

```
 0.532465465533183, 0.330869571228815, 0.112699242252689, -0.111116592961247,
-0.329366202474389, -0.531116686408463, -0.706261644820005, -0.846027443250886,
-0.943412715311215, -0.993539086045688]
sage: f=1/(1+14*x^2)
sage: P=lagrange(f, ydata)
sage: A=plot(f, -1,1, rgbcolor=(0,2,1))
sage: B=plot(P, -1,1, rgbcolor=(1,0,0))
sage: A+B
```

A.2 Résolution d'équations

Considérons le polynôme $f(x) = x^4 + 6x^2 - 60x + 36$ sur $[0, 4]$. Il s'agira d'appliquer la méthode du point fixe pour les fonctions suivantes :

- $g_1(x) = \frac{x^4+6x^2+36}{60}$
- $g_2(x) = \frac{-36}{x^3+6x-60}$
- $g_3(x) = (-6x^2 + 60x - 36)^{\frac{1}{4}}$

1. Ecrire une procédure qui prend en entrée g, x_0 et N et qui calcule le Nème terme de la liste $x_{n+1} = g(x_n)$.
2. Vérifier ensuite la convergence de la méthode du point fixe pour chacun des g_i sur $[0, 4]$. Pour cela, il faut chercher à satisfaire toutes les conditions du théorème de convergence de la méthode du point fixe :
3. Ecrire un script qui permet d'accélérer la convergence de g_3.

A.3 Factorisation de Cholesky

On considère une matrice A tridiagonale, symétrique et définie-positive d'ordre N. On cherche à compléter un algorithme permettant de résoudre un système linéaire $Ax = b$ par la méthode de Cholesky. Notons a_i les éléments de la diagonale de A et c_j les éléments de la sur-diagonale et sous-diagonale de A pour $i = 1, \ldots, N$ et $j = 1 \ldots, N-1$:

$$A = \begin{pmatrix} a_1 & c_1 & & & & \\ c_1 & a_2 & c_2 & & & \\ & c_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & c_{N-2} & \\ & & & c_{N-2} & a_{N-1} & c_{N-1} \\ & & & & c_{N-1} & a_N \end{pmatrix}$$

Soit R une matrice triangulaire inférieure telle que $A = RR^t$. Alors R aura la forme suivante :

$$\begin{pmatrix} \ell_1 & & & & \\ m_1 & \ell_2 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ell_{N-1} & \\ & & & m_{N-1} & \ell_N \end{pmatrix}$$

1. Trouver le l et m en fonction de a et de c.
2. Compléter l'algorithme de décomposition de Cholesky suivant (l est mémorisé dans a et m est mémorisé dans c) :

$a(1) = \sqrt{a(1)}$
Faire $i = 1$ à $(N-1)$
  $c(i) = ?/?$
  $a(i+1) = \sqrt{?-?*?}$

3. La résolution de $Ax = b$ nécessite la résolution du système linéaire triangulaire inférieur $Ry = b$, puis du système triangulaire supérieur $R^t x = y$. Compléter l'algorithme de sorte à obtenir la solution x du système linéaire $Ax = b$:
   (a) par une méthode de descente de $Ry = b$:

$b(1) = b(1)/a(1)$
Faire $i = 1$ à $(N-1)$
  $b(i+1) = (b(i+1)-?*?)/?$

   (b) par une méthode de remontée de $R^t x = y$
4. Faire la décomposition de Cholesky de la matrice $N \times N$ suivante :
   Les éléments non nuls de la matrice R correspondent-ils aux éléments non nuls de la matrice A ?

Application en Probabilité - Simulation de la loi $\mathcal{N}(m, \Gamma)$). Ici $m \in \mathbf{R}^d$ et Γ est une matrice de $\mathcal{M}_d(\mathbf{R})$, symétrique et définie-positive. Soient C une matrice telle que $CC^t = \Gamma$ et Z un vecteur aléatoire gaussien de loi $\mathcal{N}(0, I_d)$. Alors $X = CZ + m$ a pour loi $\mathcal{N}(m, \Gamma)$).

Correction

$$b(N) = b(N)/a(N)$$
$$\text{Faire } i = (N-1) \text{ à } 1(\text{pas de } -1)$$
$$b(i) = (b(i) - ?*?)/?$$

$$A = \begin{pmatrix} 3 & -1 & 0 & \dots & 0 & -1 \\ -1 & 3 & -1 & & & 0 \\ 0 & -1 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & -1 & 0 \\ 0 & & & -1 & 3 & -1 \\ -1 & 0 & \dots & 0 & -1 & 3 \end{pmatrix},$$

```
sage: N =3
sage: a = [3.0 for i in range(N)]
sage: c = [-1.0 for i in range(N-1)]
sage: A = matrix(RR,N,N)
sage: for i in range(N):
...       A[i,i]=a[i]
...
sage: for i in range(N-1):
...       A[i,i+1]=c[i]
...
sage: for i in range(1,N):
...       A[i,i-1]=c[i-1]
...
sage: show(A)
```

$$\begin{pmatrix} 3.00000000000000 & -1.00000000000000 & 0.000000000000000 \\ -1.00000000000000 & 3.00000000000000 & -1.00000000000000 \\ 0.000000000000000 & -1.00000000000000 & 3.00000000000000 \end{pmatrix}$$

```
show(A.cholesky_decomposition())
```

$$\begin{pmatrix} 1.73205080756888 & 0.000000000000000 & 0.000000000000000 \\ -0.577350269189626 & 1.63299316185545 & 0.000000000000000 \\ 0.000000000000000 & -0.612372435695794 & 1.62018517460197 \end{pmatrix}$$

```
sage: def cholesky(a,c):
...       a[0]=sqrt(a[0]).n()
...       for i in range(N-1):
...           c[i]=c[i]/a[i].n()
...           a[i+1]=(sqrt(a[i+1]-c[i]^2)).n()
```

```
...       A = matrix(CC,N,N)
...       for i in range(N):
...           A[i,i]=a[i]
...       for i in range(1,N):
...           A[i,i-1]=c[i-1]
...       return(A)
sage: a = [3.0 for i in range(N)]
sage: c = [-1.0 for i in range(N-1)]
sage: show(cholesky(a,c))
```

$$\begin{pmatrix} 1.73205080756888 & 0.000000000000000 & 0.000000000000000 \\ -0.577350269189626 & 1.63299316185545 & 0.000000000000000 \\ 0.000000000000000 & -0.612372435695794 & 1.62018517460197 \end{pmatrix}$$

A.4 Méthode d'Euler

Partant de $x_0 = a$, on connaît $y_0 = y(x_0) = \tau$. L'idée de la méthode d'Euler est de considérer que la courbe y sur $[x_0, x_1]$ n'est pas très éloignée de sa tangente en x_0 : une estimation rationnelle de $y(x_1)$ est $y_1 = y_0 + hf(x_0, y_0)$. En effet, pour x_1, l'équation de la tangente est $y'(x_0)(x - x_0) + y_0$. On estime alors que $y'(x_1)$ n'est pas très différent de $f(x_1, y_1)$. Sur $[x_1, x_2]$, on remplace la courbe par sa tangente approchée en x_1 ce qui donne l'estimation y_2 de $y(x_2)$: $y_2 = y_1 + hf(x_1, y_1)$ et on poursuit de la même manière. La méthode d'Euler est d'ordre 1. Une telle méthode ne converge pas assez vite pour donner des résultats pratiques intéressants. On va étudier dans les deux sections suivantes deux méthodes d'ordre plus élevé. La méthode d'Euler est implémentée dans `sage` comme suit :

```
def euler(f,n,tau,a,b):
    h = (b-a)/n
    x = [x0]
    y = tau
    for i in range(n):
        x += [x[i]+h]
        y += [y[i]+h*f(x[i],y[i])]
    return x,y
```

Annexe B

Exerices de Révision et Applications

Exercice B.0.1. Citer deux méthodes d'interpolation polynomiale. Quelle est la différence entre ces méthodes et la méthode des moindres carrés discrets ?

Exercice B.0.2. Soit f une fonction de $\mathbf{R}$ dans $\mathbf{R}$ telle que

$$f(x) = x^3 + 3x^2 - 1 \qquad \text{pour} \qquad x \in \mathbf{R}.$$

1. Montrer que l'équation $f(x) = 0$ admet une unique solution sur $[0, 1]$.
2. Nous désirons déterminer la solution de l'équation $f(x) = 0$ avec la méthode de Newton sur $[0, 1]$.
   (a) Est-ce que la fonction f vérifie les conditions du théorème de convergence globale de la méthode de Newton pour $x_0 = 1$. Donner le théorème.
   (b) Calculer la racine de f sur l'intervalle $[0, 1]$ avec une précision de calculs égale à 10^{-6}.
3. Ecrire l'algorithme général de la méthode de Newton qui prend en entrée les points x_0, $a < b$ et une fonction f et rend la racine de $f(x) = 0$ sur $[a, b]$ ou bien un message d'erreur.

Exercice B.0.3 (10pt). Soit f une fonction de $\mathbf{R}$ dans $\mathbf{R}$ telle que

$$f(x) = x^3 + 3x^2 - 1 \qquad \text{pour} \qquad x \in \mathbf{R}.$$

1. Montrer que l'équation $f(x) = 0$ admet une unique solution sur $[0, 1]$.

2. Nous désirons déterminer la solution de l'équation $f(x) = 0$ avec la méthode de Newton sur $[0, 1]$. En quoi consiste cette méthode ?
3. Ecrire l'algorithme général de la méthode de Newton qui prend en entrée les points x_0, $a < b$ et une fonction f et rend la racine de $f(x) = 0$ sur $[a, b]$ ou bien un message d'erreur.
4. Est-ce que la fonction f vérifie les conditions du théorème de convergence globale de la méthode de Newton pour $x_0 = 1$. Donner le théorème.
5. Calculer la racine de f sur l'intervalle $[0, 1]$ avec une erreur de 10^{-6}.

Exercice B.0.4 (10pt). Soit A une matrice définie par :

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

1. Donner la définition de la décomposition LU d'une matrice carré.
2. Calculer la factorisation LU de A.
3. En déduire le déterminant de A.
4. Le système linéaire
$$Ax = \begin{pmatrix} 1 \\ 0 \\ -2 \\ 2 \end{pmatrix} \qquad \text{où} \qquad x \in \mathbf{R}^4$$
admet-il une solution ? est-elle unique ?
5. Résoudre le système en utilisant la factorisation LU de la matrice A.

Exercice B.0.5.
1. Soit f une fonction continue sur un intervalle $[a, b] \subset \mathbf{R}$ avec $f(a).f(b) < 0$. Expliquer géométriquement la méthode de Newton dans la recherche d'une solution de l'équation $f(x) = 0$ sur $[a, b]$.
2. Donner l'algorithme de la méthode de Newton pour la recherche d'une racine simple d'une fonction de classe C^2 sur $[a, b]$.

Exercice B.0.6. Nous considérons l'ensemble des points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 1, & f_1 = -1, & f_2 = 3, & f_3 = 19. \end{array}$$

1. Afin de déterminer une approximation du nuage de points (x_i, f_i) pour i de 0 à 3, calculer la droite de régression linéaire P_0 par la méthode des moindres carrés discrets.

2. Citer deux méthodes permettant de calculer le polynôme de Lagrange.
3. Montrer que les polynômes d'interpolation de Lagrange vérifient $L_i(x_i) = 1$ et $L_i(x_j) = 0$ pour tout i et $j \neq i$ de 0 à 3.
4. Calculer le polynôme de Lagrange P sur les points (x_0, f_0), (x_1, f_1), (x_2, f_2) et (x_3, f_3).
5. Quelle est la différence entre P et P_0.

6. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2.
7. Calculer la racine de $P(x) = 0$ sur l'intervalle $[0, 3]$ par la méthode de Newton avec $x_0 = 1$. La précision des calculs est à 10^{-6} près.

Exercice B.0.7.
1. Soit f une fonction continue sur un intervalle $[a, b] \subset \mathbf{R}$ avec $f(a).f(b) < 0$. Expliquer géométriquement la méthode de la sécante dans la recherche d'une solution de l'équation $f(x) = 0$ sur $[a, b]$.
2. Donner l'ordre de la méthode de Newton pour la recherche d'une racine simple d'une fonction de classe C^2.

Exercice B.0.8. Nous considérons l'ensemble des points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = -4, & f_1 = -1, & f_2 = -2, & f_3 = 5. \end{array}$$

1. Afin de déterminer une approximation du nuage de points (x_i, f_i) pour i de 0 à 3, calculer la droite de régression linéaire P_0 par la méthode des moindres carrés discrets.

2. Citer deux méthodes permettant de calculer le polynôme de Lagrange.
3. Montrer que les polynômes d'interpolation de Lagrange vérifient $L_i(x_i) = 1$ et $L_i(x_j) = 0$ pour tout i et $j \neq i$ de 0 à 3.
4. Calculer le polynôme de Lagrange P sur les points (x_0, f_0), (x_1, f_1), (x_2, f_2) et (x_3, f_3).
5. Quelle est la différence entre P et P_0.

6. Vérifier que $P(x_i) = f_i$ pour i de 0 à 3 et montrer qu'il existe une unique racine de P sur l'intervalle $[0, 3]$.
7. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2.
8. Calculer la racine de $P(x) = 0$ sur l'intervalle $[0, 3]$ par la méthode de Newton avec $x_0 = 3$. La précision des calculs est à 10^{-6} près.

9. Ecrire l'algorithme de Newton qui prend en entrée les points x_0, $a < b$ et une fonction f et rend la racine de $f(x) = 0$ sur $[a, b]$ ou bien un message d'erreur.

Exercice B.0.9.
1. Calculer par la méthode de la puissance, la valeur propre dominante à 10^{-2} près de

$$A = \begin{pmatrix} 4 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} \qquad \text{avec} \qquad x_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

2. Ecrire l'algorithme de la méthode de la puissance inversée.

Exercice B.0.10. Etant donnés une matrice A triangulaire inférieure de taille (n, n) avec des 1 sur la diagonale et un vecteur b de taille n, calculer la complexité en nombre d'opérations élémentaires de la méthode de descente appliquée au système $Ax = b$.

Exercice B.0.11. Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 1, & f_1 = 0, & f_2 = 1, & f_3 = 4. \end{array}$$

1. Afin de déterminer une approximation du nuage de points (x_i, f_i) par la droite de régression linéaire, nous appliquerons la méthode des moindres carrés discrets :
   (a) Soit A la matrice associée au système des moindres carrés :

$$A = \begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix}$$

   (b) Donner la décomposition LU de A
   (c) Résoudre le système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_i f_i \\ \sum_{i=0}^{i=n} f_i \end{pmatrix}$ par une descente puis une remontée.
2. Calculer le polynôme de Lagrange P sur les points (x_i, f_i) pour i entre 0 et 3.
3. Quelle est la différence entre P et la droite de régression linéaire ?

4. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2. Peut-on appliquer ce théorème au calcul de la racine de $P(x) = 0$ sur l'intervalle $[0, 3]$ avec $x_0 = 1$?

Exercice B.0.12. Etant donnés une matrice A triangulaire supérieure de taille (n, n) inversible et un vecteur b de taille n, calculer la complexité en nombre d'opérations élémentaires de la méthode de remontée appliquée au système $Ax = b$.

Annexe C

Tests, DS et Examens corrigés

C.1 2008-2009

Nom et Prénom : Groupe :
Ecole ou Institut de l'année dernière :

Test 1 : Interpolation et Approximation

Exercice : Soient (x_i, f_i) pour $i = 0..n$ des points réels ou complexes. Ecrire un algorithme permettant de calculer le polynôme de Lagrange se basant sur les polynômes d'interpolation $L_i(x)$ pour $i = 0..n$.
Déterminer la complexité de cet algorithme en fonction de n.

Correction :
Entrée : (x_i, f_i) pour $i = 0..n$
Sortie : Le polynôme de Lagrange P(x).

$P := 1$;
Pour i allant de 0 à n Faire
    L :=1 ;
    Pour j allant de 0 à n Faire
        Si $i <> j$ alors
            $L := L + (x - x_j)/(x - x_i)$;
        Fin Si ;
    Fin Pour ;
    $P := P + f_i + L$;
Fin Pour ;
Afficher(P) ;
Fin.

Il y a $o(4n^2)$ opérations élémentaires. Il s'agit de $o(2n^2)$ additions/soustractions, $o(n^2)$ multiplications et $o(n^2)$ divisions.

Examen du module Analyse Numérique

Exercice 1 (6pt).

1. Donner le théorème de Gershgorin appliqué à une matrice A de taille $n \times n$ quelconque.
2. Expliquer la méthode de la puissance appliqée à une matrice A en donnant l'algorithme.
3. Calculer les deux premières itérations de la méthode de la puissance apliquée à :

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \qquad \text{avec} \qquad y_0 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Exercice 2 (4pt). Exercice : Soient (x_i, f_i) pour $i = 0..n$ des points réels ou complexes. Ecrire un algorithme permettant de calculer le polynôme de Lagrange se basant sur les polynômes d'interpolation $L_i(x)$ pour $i = 0..n$.
Déterminer la complexité de cet algorithme en fonction de n.

Problème (10pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$x_0 = -1, \quad x_1 = 0, \quad x_2 = 1, \quad x_3 = 2,$$
$$f_0 = 1, \quad f_1 = 0, \quad f_2 = 1, \quad f_3 = 4.$$

1. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2. Peut-on appliquer ce théorème au calcul de la racine de $P(x) = 0$ sur l'intervalle $[0, 3]$ avec $x_0 = 1$?
2. Soit $A = \begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix}$. Donner la décomposition LU de A.
3. Résoudre le système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_i f_i \\ \sum_{i=0}^{i=n} f_i \end{pmatrix}$ par une descente puis une remontée.
4. En déduire une approximation du nuage de points (x_i, f_i) par la droite de régression linéaire.
5. Quelle est la différence entre le polynôme de Lagrange et la droite de régression linéaire ?

Correction de l'Examen d'Analyse Numérique

Exercice 1 (6pt).

1. Le théorème de Gershgorin appliqué à une matrice A de taille $n \times n$ quelconque donne une borne sur le spectre de A : toute valeur propre appartient à l'un au moins des disques de Gerschgorin :

$$| a_{ii} - \lambda | \leq \sum_{i \neq j} a_{ij}$$

2. La méthode de la puissance appliqée à une matrice A consiste à choisir au hasard un vecteur y_0 et à caluler la suite $y_{n+1} = \frac{Ay_n}{\|Ay_n\|}$.
3. Pour $y_0 = \begin{pmatrix} 0.71 \\ 0.71 \end{pmatrix}$, nous avons $y_1 = \begin{pmatrix} 0.45 \\ 0.89 \end{pmatrix}$ et $y_2 = \begin{pmatrix} 0.24 \\ 0.97 \end{pmatrix}$.

Exercice 2 (4pt). Entrée : (x_i, f_i) pour $i = 0..n$
Sortie : Le polynôme de Lagrange P(x).

$P := 1$;
Pour i allant de 0 à n Faire
    L :=1 ;
    Pour j allant de 0 à n Faire
        Si $i <> j$ alors
            $L := L + (x - x_j)/(x - x_i)$;
        Fin Si ;
    Fin Pour ;
    $P := P + f_i + L$;
Fin Pour ;
Afficher(P) ;
Fin.

Il y a $o(4n^2)$ opérations élémentaires. Il s'agit de $o(2n^2)$ additions/soustractions, $o(n^2)$ multiplications et $o(n^2)$ divisions.

Problème (10pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 1, & f_1 = 0, & f_2 = 1, & f_3 = 4. \end{array}$$

1. Voir le cours pour le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2.
   On ne peux appliquer ce thèorème au calcul de la racine de $P(x) = x^2$ sur l'intervalle $[0, 3]$: il faudra restreindre l'intervalle car $P'(0) = 0$ et $P''(0) = 0$).

2. Soit $A = \begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix}$. La décomposition LU de A est triviale :

$$L = \begin{pmatrix} 1 & 0 \\ \frac{1}{3} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 6 & 2 \\ 0 & \frac{10}{3} \end{pmatrix}.$$

3. La résolution du système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 8 \\ 6 \end{pmatrix}$ par une descente donne $\begin{pmatrix} 8 \\ \frac{10}{3} \end{pmatrix}$ puis la remontée donne $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

4. Une approximation du nuage de points (x_i, f_i) est donnée par la droite de régression linéaire d'équation $P_0(x) = x + 1$.

5. Conclure sur la variation en terme de degré, de complexité des calculs et en terme de différence entre interpolation et approximation.

Examen de contrôle du module Analyse Numérique

Problème 1 (10pt). Notons A une matrice symétique définie positive d'ordre n.

$r_{1,1}$ = $\sqrt{a_{1,1}}$
pour i de 2 à n faire :

$$r_{i,j} = \frac{a_{i,j}-\sum_{k=1}^{k=j-1} r_{ik}r_{jk}}{r_{j,j}} \quad \text{pour} \quad j = 1..i-1$$

et

$$r_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{k=i-1} r_{i,k}^2}.$$

1. Ecrire l'algorithme de la méthode de Cholesky
2. Supposons que la fonction racine suppose au plus 9 opérations élémentaires. Déterminer la complexité de la méthode de factorisation de Cholesky de A en fonction de n.
3. Donner les 2 premières itérations de cette méthode appliquée à la matrice

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

Problème 2 (10pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 1, & f_1 = 0, & f_2 = 1, & f_3 = 4. \end{array}$$

1. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2. Peut-on appliquer ce théorème au calcul de la racine de $P(x) = 0$ sur l'intervalle $[0, 3]$ avec $x_0 = 1$?
2. Soit $A = \begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix}$. Donner la décomposition LU de A.
3. Résoudre le système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_i f_i \\ \sum_{i=0}^{i=n} f_i \end{pmatrix}$ par une descente puis une remontée.
4. En déduire une approximation du nuage de points (x_i, f_i) par la droite de régression linéaire.

5. Quelle est la différence entre le polynôme de Lagrange et la droite de régression linéaire ?

Correction de l'Examen de contrôle du module Analyse Numérique

Problème 1 (10pt). Notons A une matrice symétique définie positive d'ordre n.

1. L'algorithme de la méthode de Cholesky est le suivant :
   Entrée : $a_{i,j}$
   Sortie : $r_{i,j}$

   $r_{1,1} = \sqrt{a_{1,1}}$
   Pour i allant de 2 à n Faire
   $r_{i,i} = a_{i,i}$
   Pour j allant de 1 à $i-1$ Faire
   $r_{i,j} = a_{i,j}$
   Pour k allant de 1 à $j-1$ Faire
   $r_{i,j} = r_{i,j} - r_{i,k}r_{j,k}$
   Fin Pour ;
   $r_{i,j} = \frac{r_{i,j}}{r_{j,j}}$
   Fin Pour
   Pour k allant de 1 à $i-1$ Faire
   $r_{i,i} = r_{i,i} - r_{i,k}r_{i,k}$
   Fin Pour ;
   $r_{i,i} = \sqrt{r_{i,i}}$
   Fin Pour ;
   Fin.
2. Sachant que la fonction racine suppose au plus 9 opérations élémentaires, la complexité de la méthode de factorisation de Cholesky de A en fonction de n est calculée comme suit :
   Il y a $\sum_{i=2}^{n}(\sum_{j=1}^{i-1}(j-1)) + \sum_{i=2}^{n}(i-1) = \frac{n(n-1)(n-2)}{6} + \frac{n(n-1)}{2}$ additions.
   Il y a $\sum_{i=2}^{n}(\sum_{j=1}^{i-1} j) + \sum_{i=2}^{n}(i-1) = \frac{n(n-1)(n-2)}{6} + \frac{n(n-1)}{2}$ multiplications.
   Et il y a $\sum_{i=2}^{n} 9 = 9(n-1)$. Donc $9(n-1)$ opérations élémentaires permettant le calcul des racines carrés.

   Nous obtenons une complexité de l'ordre de $o(\frac{1}{3}n3 + 9n)$.
3. La matrice R étant triangulaire inférieure, elle est calculée par la méthode de Cholesky appliquée à A. Nous obtenons $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

Problème 2 (10pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 1, & f_1 = 0, & f_2 = 1, & f_3 = 4. \end{array}$$

1. Voir le cours pour le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2.
   On ne peux appliquer ce thèorème au calcul de la racine de $P(x) = x^2$ sur l'intervalle $[0, 3]$: il faudra restreindre l'intervalle car $P'(0) = 0$ et $P''(0) = 0$).

2. Soit $A = \begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix}$. La décomposition $PA = LU$ est triviale :

   $$L = \begin{pmatrix} 1 & 0 \\ \frac{1}{3} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 6 & 2 \\ 0 & \frac{10}{3} \end{pmatrix}.$$

   Ici, la matrice de permutation P est égale à la matrice identité d'ordre 2.

3. La résolution du système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 8 \\ 6 \end{pmatrix}$ par une descente donne $\begin{pmatrix} 8 \\ \frac{10}{3} \end{pmatrix}$ puis la remontée donne $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

4. Une approximation du nuage de points (x_i, f_i) est donnée par la droite de régression linéaire d'équation $P_0(x) = x + 1$.

5. La variation entre ces deux approches se notent en terme de degré : le polynôme d'interpolation ayant un degré égal au nombre de points-1 alors que dans l'approximation, le degré en est indépendant. Dans le cas de l'approximation, le degré est largement plus petit que dans l'interpolation. La différence entre les deux approches est constatée dans la complexité des calculs : le cas étudié en cours, donne une approximation plus rapidement que l'interpolation de Lagrange par exemple.

C.2 2009-2010

Examen du module Analyse Numérique

Ines Abdeljaoued et Walid Miladi

Exercice 1 (6pt). Notons $A = (a_{ij})$ une matrice symétrique définie positive d'ordre n. La factorisation de la matrice A par la méthode de Cholesky est détaillée comme suit :

$r_{1,1}$ = $\sqrt{a_{1,1}}$
pour i de 2 à n faire :

$$r_{i,j} = \frac{a_{i,j}-\sum_{k=1}^{k=j-1} r_{ik}r_{jk}}{r_{j,j}} \quad \text{pour} \quad j = 1..i-1$$

et

$$r_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{k=i-1} r_{i,k}^2}.$$

1. Ecrire l'algorithme de la méthode de Cholesky qui prend en entrée A et qui calcule une matrice R triangulaire vérifiant $A = R^t R$.
2. Supposons que la fonction racine nécessite au plus 9 opérations élémentaires. Déterminer la complexité de la méthode de factorisation de Cholesky de A en fonction de n.
3. Donner la factorisation de Cholesky de la matrice

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

Exercice 2 (4pt). Calculer par la méthode de la puissance la valeur propre et le vecteur propre dominants de la matrice $A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ avec $X_0 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$.

Problème (10pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 0, & f_1 = -1, & f_2 = 0, & f_3 = 3. \end{array}$$

1. Calculer le poynôme de Lagrange P sur les points d'interpolation x_i, $i = 0..3$.
2. Donner le théorème de convergence de la méthode du point fixe pour une fonction de classe C^1 sur un intervalle $[a, b]$. Peut-on appliquer ce théorème au calcul de la racine de $P(x) = 0$ sur l'intervalle $[-2, 0]$ avec $g(x) = \frac{1}{2}P(x) + x$ et $X_0 = 0$? Si oui, calculer la racine de P sur $[-2, 0]$.
3. Soit $A = \begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix}$. Résoudre le système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_i f_i \\ \sum_{i=0}^{i=n} f_i \end{pmatrix}$ par une descente puis une remontée (suite à une factorisation LU).
4. En déduire une approximation du nuage de points (x_i, f_i) par la droite de régression linéaire $p_1(x) = a_0 x + a_1$.

Examen de contrôle du module Analyse Numérique

Exercice (8pt).

1. Etant donné une matrice $A = (a_{ij})$, λ une de ses valeurs propres associé au vecteur propre v et i l'indice de la coordonné v_i de v vérifiant $\mid v_i \mid = max_{1 \leq j \leq n} \mid v_j \mid$. Montrer que
$$\mid \lambda - a_{ii} \mid \leq \sum_{j=1, j \neq i}^{j=n} \mid a_{ij} \mid .$$
2. Calculer la valeur propre et le vecteur propre dominants de A par la méthode de la puissance :
$$A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \qquad \text{avec} \qquad X_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$
3. Que se passe-t-il lorsqu'on considère $X_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$?

Problème (12pt). Soit $n = 3$. Nous considérons l'ensemble de points

$$\begin{array}{llll} x_0 = -1, & x_1 = 0, & x_2 = 1, & x_3 = 2, \\ f_0 = 0, & f_1 = -1, & f_2 = 0, & f_3 = 3. \end{array}$$

1. Calculer le poynôme de Lagrange P sur les points d'interpolation x_i, $i = 0..3$.
2. Donner le théorème de convergence de la méthode du point fixe pour une fonction de classe C^1 sur un intervalle $[a, b]$. Peut-on appliquer ce théorème au calcul de la racine de $P(x) = 0$ sur l'intervalle $[0, 2]$ avec $g(x) = \frac{1}{2}P(x) + x$ et $X_0 = 0$?
3. Soit $A = \begin{pmatrix} \sum_{i=0}^{i=n} x_i^2 & \sum_{i=0}^{i=n} x_i \\ \sum_{i=0}^{i=n} x_i & n+1 \end{pmatrix}$. Donner la décomposition LU de A.
4. Résoudre le système $A \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{i=n} x_i f_i \\ \sum_{i=0}^{i=n} f_i \end{pmatrix}$ par une descente puis une remontée.
5. En déduire une approximation du nuage de points (x_i, f_i) par la droite de régression linéaire $p_1(x) = a_0 x + a_1$.
6. Quelle est la différence entre le polynôme de Lagrange et la droite de régression linéaire ?

C.3 2010-2011

DEVOIR SURVEILLÉ DU MODULE ANALYSE NUMÉRIQUE

Ines Abdeljaoued Tej

Interpolation.

1. Soient f une fonction de classe C^2 sur $[0,1]$ et $xdata = [x_0, x_1, \ldots, x_n]$ une liste de $n+1$ points deux à deux distincts de $[0,1]$. Ecrire l'algorithme de la méthode des différences divisées qui prend en entrée f et $xdata$ et qui calcule le polynôme de Lagrange de f aux points $x_0, x_1, \ldots, x_n$.

   Indication : la formule des différences divisées est la suivante :
   $$\begin{cases} c_{i,i} = f(x_i) & \text{pour } i = 0,..,n \\ c_{i,j} = \frac{c_{i+1,j} - c_{i,j-1}}{x_j - x_i} & \text{pour } 0 \leq i < j \leq n. \end{cases}$$

2. Répondre par Vrai ou Faux aux affirmations suivantes et justifier votre réponse :
   (a) Il existe une infinité de polynômes de degré $> n$ interpolant les points $(x_i, f(x_i))_{i=0,..,n}$.
   (b) Le polynôme de Lagrange de $sin(x)$ sur les points 0, $\frac{\pi}{2}$ et π est égal à $P(x) = x^3 + 3x^2$.

Intégration.

3. Calculer par la méthode des trapèzes une approximation $T(h)$ de $\int_0^2 x^2 dx$. Considérer pour cela une subdivision de l'intervalle $[0,2]$ en 2 sous intervalles de même amplitude.
4. Soient $h = \frac{b-a}{n}$ et $x_i = a + ih$. Sachant que $T(h) = \sum_{i=0}^{n-1} \frac{h}{2}(f(x_i) + f(x_{i+1}))$, vérifier que
   $$T(\frac{h}{2}) = \frac{T(h)}{2} + \frac{h}{2} \sum_{k=0}^{n-1} f(a + kh + \frac{h}{2}).$$
   Connaissant $T(h)$, donner le nombre d'opérations élémentaires nécessaires au calcul de $T(\frac{h}{2})$. Nous supposons qu'une évaluation de f nécessite α multiplications et β additions.
5. En déduire $\int_0^2 x^2 dx$ pour une subdivision de l'intervalle $[0,2]$ en 4 sous intervalles de même amplitude.

Résolution d'équations non linéaires.

6. Donner le théorème de convergence globale de la méthode de Newton pour une fonction de classe C^2 sur un intervalle $[a,b]$.
7. Peut-on appliquer ce théorème au calcul de la racine x^* de $f(x) = x^3 - 1$ sur l'intervalle $[0.5, 1.5]$? Si oui, déterminer x^* à 10^{-1} près.

Analyse Numérique - Correction du DS

Ines Abdeljaoued Tej

Interpolation.

1. Soient f une fonction de classe C^2 sur $[0,1]$ et $xdata = [x_0, x_1, \ldots, x_n]$ une liste de $n+1$ points deux à deux distincts de $[0,1]$. L'algorithme suivant reprend la méthode des différences divisées.
   Entrée : Une fonction f et $xdata == [x_0, x_1, \ldots, x_n]$
   Sortie : Le polynôme de Lagrange de f aux points $x_0, x_1, \ldots, x_n$

```
# Méthode des différences divisées (formule de Newton)
def newton(f,xdata):
    n = len(xdata)-1
    c = matrix(QQ,n+1,n+1)
    for i in range(n+1):
        c[i,i] = f(xdata[i])
    #print c
    #print("-------")
    for k in range(1,n+1):
        for i in range(n+1-k):
            c[i,i+k] = (c[i+1,i+k]-c[i,i+k-1])/(xdata[i+k]-xdata[i])
        #print c
        #print("-------")
    P = c[0,0]
    P += sum(c[0,k]*prod((x-xdata[j]) for j in range(k)) for k in range(1,n+1))
    return(expand(P))
```

2. Répondre par Vrai ou Faux aux affirmations suivantes et justifier votre réponse :
   (a) Vrai. Il existe une infinité de polynômes de degré supérieur strictement à n interpolant les points $(x_i, f(x_i))_{i=0,..,n}$. En particulier, par deux points passe une infinité de courbes, mais une seule est une droite.
   (b) Faux. Le polynôme de Lagrange de $sin(x)$ sur les points 0, $\frac{\pi}{2}$ et π n'est pas égal à $P(x) = x^3 + 3x^2$. Ici, P est un polynôme de degré supérieur au nombre de points + 1 et $P(x_i) \neq sin(x_i)$.

Intégration.

3. Une approximation $T(h)$ de $\int_0^2 x^2 dx$ par la méthode des trapèzes avec une subdivision de l'intervalle $[0,2]$ en 2 sous intervalles de même amplitude est égale à $T(1) = 3$.
4. Soient $\frac{h}{2} = \frac{b-a}{2n}$ et $x_i = a + i\frac{h}{2}$ pour $i = 0, \ldots, 2n$. Il suffit de regrouper les indices i dans $T(\frac{h}{2})$ en deux groupes : le premier constitué d'indices pairs (donnant $\frac{T(h)}{2}$)

et le second groupe d'indices impairs (donnant $\frac{h}{2}\sum_{k=0}^{n-1} f(a+kh+\frac{h}{2})$). Ainsi, nous obtenons que

$$T(\frac{h}{2}) = \frac{T(h)}{2} + \frac{h}{2}\sum_{k=0}^{n-1} f(a+kh+\frac{h}{2}). \qquad \text{(C.3.1)}$$

Connaissant $T(h)$, Le nombre d'opérations élémentaires nécessaires au calcul de $T(\frac{h}{2})$ est égal à

$n(\beta+2)+1$	additions	0	soustractions
$n(\alpha+1)$	multiplications	2	divisions

où α multiplications et β additions sont nécessaires pour une seule évaluation de f.

5. Remplacer $a=0$, $b=2$, $f(x)=x^2$, $n=2$ et $h=1$ dans (C.3.1). On obtient $\int_0^2 x^2dx$ pour une subdivision de l'intervalle $[0,2]$ en 4 sous intervalles de même amplitude, i.e. 2.75 :

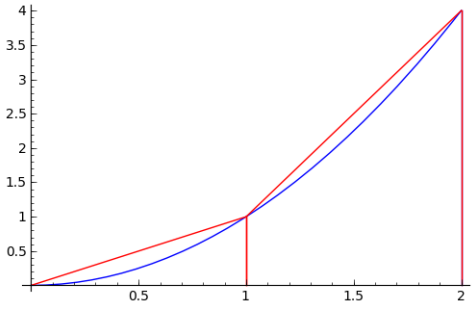


FIGURE C.1 – *Intégration pour* $h = 1$

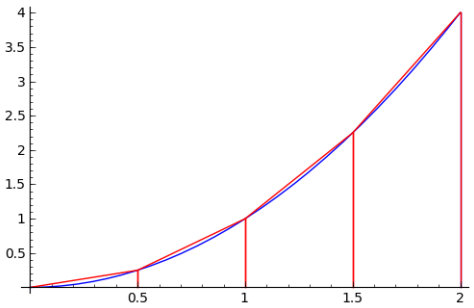


FIGURE C.2 – *Intégration pour* $h = \frac{1}{2}$

Résolution d'équations non linéaires.

6. Le théorème de convergence globale de la méthode de Newton. Soit f une fonction de classe C^2 sur un intervalle $[a,b]$ vérifiant $f(a).f(b) < 0$. Si pour tout $x \in [a,b]$, $f'(x) \neq 0$ et $f"(x) \neq 0$ alors $\forall x_0 \in [a,b]$ tel que $f(x_0).f"(x_0) > 0$, la suite $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ converge vers l'nique racine de f sur $[a,b]$.
7. La fonction $f(x) = x^3 - 1$ vérifie les conditions du théorème sur l'intervalle $[0.5, 1.5]$. Elle est strictement monotone et elle change de signe. Elle admet donc une unique racine obtenue, par la méthode de Newton, de la manière suivante : x^* à 10^{-1} près. Partant de $x_0 = 1.5$, nous obtenons $x_1 = 1.1481$, $x_2 = 1.0182$ et $x_3 = 1.0003$. La courbe rouge représente f' et la courbe bleue représente $f"$.

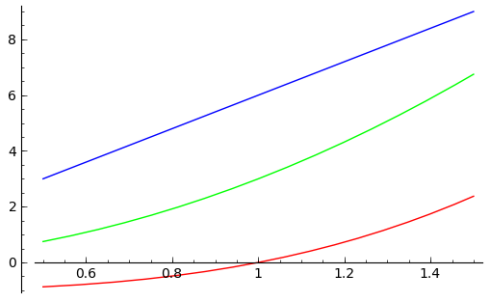


FIGURE C.3 – *Résolution de* $x^3 - 1$