ANALYSE NUMÉRIQUE

P. HELLUY

Table des matières

1. Résolution directe des systèmes linéaires

Dans la suite, $A = (a(i,j))_{1\leq i\leq n 1\leq j\leq n}$ désigne une matrice carré inversible de nombres réels ou complexes.

1.1. **Notions préliminaires.**

Une des tâches fondamentales du numéricien consiste à résoudre, à l'aide d'un ordinateur, des systèmes linéaires de la forme $Ax = b$ où x est l'inconnue et b un vecteur réel ou complexe. Souvent, il arrive que la matrice A est **creuse**, c'est à dire qu'elle contient beaucoup de zéros. Par opposition, une matrice possédant peu de zéros est dite **pleine**. Pour des raisons d'efficacité il faudra trouver des algorithmes permettant d'éviter au maximum le stockage des zéros afin d'économiser la mémoire et les temps de calcul.

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

1.1.1. *Stockage des matrices creuses.*

La méthode la plus économique pour stocker une matrice creuse consiste à mettre ses valeurs dans un tableau de réels $\text{atab}(k)_{1\leq k\leq N}$ où N est le nombre de valeurs non nulles de A. Les termes diagonaux de A seront toujours supposés **non nuls**. Il faut alors deux tableaux $\text{indi}(k)_{1\leq k\leq N}$ et $\text{indj}(k)_{1\leq k\leq N}$ pour repérer les numéros de lignes et de colonnes suivant la formule :

$$\text{atab}(k) = a(\text{indi}(k), \text{indj}(k))$$

Exemple : dans la suite nous ferons les calculs pour la matrice

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

ici, nous sommes conduits à

$$\begin{aligned} \text{atab} &= [2,-1,2,-1,-1,2,-1,-1,-1,2] \\ \text{indi} &= [1,1,2,2,3,3,3,4,4,4] \\ \text{indj} &= [1,4,2,3,2,3,4,1,3,4] \\ \text{N} &= 10 \end{aligned}$$

Bien qu'économique en mémoire, ce stockage (dit stockage **morse**) est peu adapté à la résolution directe des systèmes linéaires. Nous allons partir de cette représentation afin d'obtenir un stockage plus utile. Pour simplifier l'exposé, nous supposerons (quitte à stocker quelques zéros supplémentaires) que les valeurs non nulles de A sont réparties de manière symétrique ($a(i,j) \neq 0 \Rightarrow a(j,i) \neq 0$). A n'est pas nécessairement symétrique.

1.1.2. *Profil d'une matrice.*

Définition 1. – *On appelle* ***profil*** *de A le tableau $prof(j)_{1\leq i\leq n}$ défini par*

$$E_j = \{i, 1 \leq i \leq j, a(i,j) \neq 0\}$$
$$prof(j) = j - \max E_j$$

– *On dit qu'un couple (i,j) est dans le profil de A si*

$$j - prof(j) \leq i \leq j \text{ ou } i - prof(i) \leq j \leq i$$

Remarque : sur la colonne j, E_j contient en fait les indices de lignes i au dessus de la diagonale, pour lesquels $a(i,j) \neq 0$. $j - \text{prof}(j) + 1$ est alors le plus petit de ces indices.

Exemple : le profil est ici

$$\text{profil} = [0, 0, 1, 3]$$

1.1.3. *Diminution du profil d'une matrice, algorithme de Cuthill-Mac Kee.* Nous verrons par la suite qu'il est intéressant que la matrice A ait un profil le plus petit possible. L'algorithme de Cuthill-Mac Kee consiste à renuméroter les lignes et les colonnes de A afin de diminuer son profil, c'est à dire concentrer les valeurs non nulles autour de la diagonale. Nous dirons que deux indices i et j sont **voisins** si $a(i,j) \neq 0$ (rappelons d'ailleurs que par hypothèse $a(i,i) \neq 0$, donc i est toujours sont propre voisin), et nous commençons par compter les voisins $\text{nbvois}(i)$ de i grâce à l'algorithme suivant.

```
for k to N do
if indi[k] > indj[k] then
nbvois[indj[k]] :=nbvois[indj[k]]+1 :
nbvois[indi[k]] :=nbvois[indi[k]]+1 :fi :od :
for k to n do nbvois[k] :=nbvois[k]+1 :od :
```

Notons alors $\text{nvoismax} = \max_{1 \leq i \leq n} \text{nbvois}(i)$. nvoismax est le nombre maximum de voisins sur tous les indices. Nous sommes donc en mesure de construire le tableau $\text{listvois}(i,l)_{1 \leq i \leq n, 1 \leq l \leq \text{nvoismax}}$ où $\text{listvois}(i,l)$ contient le voisin numéro l de l'indice i :

```
indvois :=vector(n) :listvois :=matrix(n,nvoismax) :
for k to n do
indvois[k] :=1 :listvois[k,1] :=k :od :
for k to N do
if indi[k] > indj[k] then
i :=indi[k] :j :=indj[k] :
indvois[i] :=indvois[i]+1 :indvois[j] :=indvois[j]+1 :
listvois[i,indvois[i]] :=j :listvois[j,indvois[j]] :=i :
fi :od :
```

Exemple : on trouve

$$\text{listvois} = \begin{bmatrix} 1 & 4 & 0 \\ 2 & 3 & 0 \\ 3 & 2 & 4 \\ 4 & 1 & 3 \end{bmatrix}$$

Ces tableaux permettent d'ailleurs de calculer le profil de A :

```
prof :=vector(n,0) :
for i to n do :for j to indvois[i] do
if listvois[i,j] < i then
prof[i] :=max(prof[i],i-listvois[i,j]) :fi :od :od :
```

L'algorithme de Cuthill-MacKee consiste alors à partir d'un indice arbitraire $i_0 = \text{inum}(1)$. Ensuite, dans le tableau inum on ajoute les voisins de i_0 puis les voisins des voisins, etc. On peut donc écrire :

```
estcompte :=vector(n) :inum :=vector(n) :i0 :=1 :
for i to n do
estcompte[i] :=0 :od :
inum[1] :=i0 :
estcompte[i0] :=1 :
compte :=1 :pointeur :=1 :
while compte < n do
```

```
iactu :=inum[pointeur] :
for ii to nbvois[iactu] do
i :=listvois[iactu,ii] :
if estcompte[i]=0 then compte :=compte+1 :inum[compte] :=i :
estcompte[i] :=1 :fi :
od :
pointeur :=pointeur+1 :
od :
```

A la fin, le tableau inum contient la nouvelle numérotation.

Exemple :

$$\text{inum} = [1, 4, 3, 2]$$

Après avoir renuméroté, il est souvent intéressant de relancer plusieurs fois l'algorithme en prenant $i_0 := \text{inum}(n)$. Il est aussi préférable, dans la boucle sur ii de numéroter en priorité les indices ayant le moins de voisins. Si après l'algorithme, la diminution du profil est importante, il est préférable de remplacer la matrice A par la matrice A' définie par :

$$a'_{\text{inum}(i),\text{inum}(j)} = a_{i,j}$$

Exemple : A' est ici

$$A' = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

(dans la suite des exemples, nous continuons à travailler avec A et non avec A')

Nous sommes maintenant en mesure de construire un nouveau stockage de A appelé stockage profil.

1.1.4. *Stockage profil de A.* A sera stockée dans trois tableaux : ainf contiendra la partie inférieure de A. asup contiendra la partie supérieure et adiag la diagonale de A. La diagonale est facile à stocker

$$\text{adiag}(i) = a(i, i) \quad 1 \leq i \leq n$$

On décide de ranger dans asup les valeurs de la partie supérieure de A qui sont dans le profil, colonne après colonne, zéros compris. Le principe de stockage est le même pour ainf, en travaillant ligne par ligne. En notant NN le nombre de valeurs stockées dans ainf ou asup, on a donc

$$NN = \sum_{i=1}^{n} \text{prof}(i)$$

Pour le repérage, nous aurons besoin de savoir où commence chaque colonne dans asup (ou chaque ligne dans ainf). Pour cela, nous construisons un tableau $\text{debut}(i)_{1 \leq i \leq n}$ par l'algorithme

```
debut :=vector(n+1) :debut[1] :=1 :debut[2] :=1 :
for k from 3 to n+1 do
debut[k] :=debut[k-1]+prof[k-1] :od :
```

Il est alors possible de remplir asup et ainf par l'algorithme

```
asup :=vector(NN,0) :ainf :=vector(NN,0) :for k to N do
i :=indi[k] :j :=indj[k] :
if i < j then kk :=debut[j+1]-j+i :asup[kk] :=atab[k] :fi :
if i > j then kk :=debut[i+1]-i+j :ainf[kk] :=atab[k] :fi :
od :
```

Exemple :

$$\begin{aligned} NN &= 4 \\ \text{adiag} &= [2,2,2,2] \\ \text{ainf} &= [-1,-1,0,-1] \\ \text{asup} &= [-1,-1,0,-1]) \\ \text{debut} &= [1,1,1,2,5] \end{aligned}$$

Nous donnons enfin en exemple l'algorithme de calcul du produit de A par u avec stockage dans v

```
for i to n do
v[i] :=adiag[i]*u[i] :
for k from i-prof[i] to i-1 do
v[i] :=v[i]+ainf[debut[i+1]-i+k]*u[k] :od :
for k from i+1 to n do
if i > =k-prof[k] then
v[i] :=v[i]+asup[debut[k+1]-k+i]*u[k] :
fi :
od :od :
```

1.2. **Décomposition LU d'une matrice inversible.**

1.2.1. *Algorithme.* Pour résoudre $Ax = b$, on va chercher à écrire $A = LU$ où

- L est une matrice triangulaire inférieure avec des 1 sur la diagonale, telle que $L + L^T$ a le même profil que A.
- U est une matrice triangulaire supérieure, telle que $U + U^T$ a le même profil que A.

La résolution de $Ax = b$ est alors ramenée aux résolutions successives des systèmes **échelonnés** $Ly = b$ et $Ux = y$.

Nous allons procéder par identification :

$$l(i,j) = \begin{cases} 1 \text{ si } i = j \\ 0 \text{ si } i < j \\ l(i,j) \text{ si } i - \text{prof}(i) \leq j < i \\ 0 \text{ si } 1 \leq j < i - \text{prof}(i) \end{cases} \qquad u_{i,j} = \begin{cases} 0 \text{ si } j < i \\ u(i,j) \text{ si } j - \text{prof}(j) \leq i \leq j \\ 0 \text{ si } 1 \leq i < j - \text{prof}(j) \end{cases}$$

nous voulons donc que

$$a(i,j) = \sum_{k=1}^{n} l(i,k)\, u(k,j) \quad 1 \leq i \leq n$$

pour des (i,j) dans le profil de A, on a donc (par convention, $\sum_{k=a}^{b} = 0$ si $a > b$) :

$$\begin{aligned} a(i,j) &= \sum_{k=1}^{n} l(i,k)\, u(k,j) \\ &= \sum_{k=\max(i-\text{prof}(i),\, j-\text{prof}(j))}^{\min(i,j)} l(i,k)\, u(k,j) \end{aligned}$$

donc si $j < i$ alors

$$a(i,j) = \sum_{k=\max(i-\text{prof}(i),\, j-\text{prof}(j))}^{j} l(i,k)\, u(k,j)$$

et si $i \leq j$ alors

$$a(i,j) = \sum_{k=\max(i-\text{prof}(i),j-\text{prof}(j))}^{i} l(i,k)\,u(k,j)$$

et nous en déduisons que

$$u(1,j) = a(1,j) \quad 1 \leq j \leq 1+\text{prof}(j)$$

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

$$\begin{aligned} l(i,j) &= \frac{1}{u(j,j)}\left(a(i,j) - \sum_{k=\max(i-\text{prof}(i),j-\text{prof}(j))}^{j-1} l(i,k)\,u(k,j)\right) \\ &\quad \text{pour } i-\text{prof}(i) \leq j < i \\ u(i,j) &= a(i,j) - \sum_{k=\max(i-\text{prof}(i),j-\text{prof}(j))}^{i-1} l(i,k)\,u(k,j) \qquad \text{pour } i \leq j \leq i+\text{prof}(j) \end{aligned}$$

il se trouve, dans cet algorithme, que L et U peuvent être stockés dans les mêmes tableaux que A au cours de l'élimination, à condition de prendre les calculs dans le bon ordre ! L'algorithme final s'écrit alors :

```
for i from 2 to n do
for j from i-prof[i] to i-1 do
for k from max(i-prof[i],j-prof[j]) to j-1 do
ainf[debut[i+1]-i+j] :=ainf[debut[i+1]-i+j]
-ainf[debut[i+1]-i+k]*asup[debut[j+1]-j+k] :
asup[debut[i+1]-i+j] :=asup[debut[i+1]-i+j]
-ainf[debut[j+1]-j+k]*asup[debut[i+1]-i+k] :
od :
ainf[debut[i+1]-i+j] :=ainf[debut[i+1]-i+j]/adiag[j] :
od :
for k from i-prof[i] to i-1 do
adiag[i] :=adiag[i]-ainf[debut[i+1]-i+k]*asup[debut[i+1]-i+k]
od :od :
```

Exemple : On trouve

$$\begin{aligned} \text{adiag} &= [2, 2, 3/2, 5/6] \\ \text{asup} &= [-1, -1, 0, -1] \\ \text{ainf} &= [-1/2, -1/2, 0, -2/3] \end{aligned}$$

on a donc

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

$$U = \begin{bmatrix} 2 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 3/2 & -1 \\ 0 & 0 & 0 & 5/6 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1/2 & 1 & 0 \\ -1/2 & 0 & -2/3 & 1 \end{bmatrix}$$

1.2.2. *Descente-remontée.* L'algorithme de résolution de $Ly = b$ (descente) s'écrit

```
y :=vector(n,0) :x :=vector(n,0) :
y[1] :=b[1] :
for i from 2 to n do
y[i] :=b[i] :
for k from i-prof[i] to i-1 do
y[i] :=y[i]-ainf[debut[i+1]-i+k]*y[k] :
od :od :
```

L'algorithme de résolution de $Ux = y$ (remontée) s'écrit

```
x[n] :=y[n]/adiag[n] :
for i from n-1 to 1 by -1 do
x[i] :=y[i] :for k from i+1 to n do
if i > =k-prof[k] then
x[i] :=x[i]-asup[debut[k+1]-k+i]*x[k] :fi :
od :
x[i] :=x[i]/adiag[i] :
od :
```

1.2.3. *Algorithme avec pivotage.* Il arrive parfois au cours de l'algorithme que le **pivot**, qui est la valeur par laquelle on divise (ici adiag(i)), est nul ou très petit. À cause des erreurs d'arrondi, les résultats obtenus peuvent alors être très différents de la solution exacte. Il faut donc modifier l'algorithme. Pour cela, on amène à chaque étape en position (i, i), par échange de lignes, le plus grand élément en valeur absolue de la colonne i. Il faut mémoriser ces échanges de lignes dans un tableau (noté ici sig). De plus, la structure ligne de ciel étant détruite par ces échanges, nous présentons un algorithme pour les matrices pleines.

```
factolu :=proc(A,sig,n)
#initialisations
det :=1 :sig :=vector(n) :
for i to n do sig[i] :=i :od :
for k to n-1 do
p :=A[k,k] :kp :=k :
#recherche du pivot le plus grand
for i from k to n do
if abs(A[i,i]) > abs(p) then
kp :=i :p :=A[i,i] :fi :
od :
#calcul de la signature de la permutation
#pour calcul du determinant
if not(kp=k) then det :=-det :fi :
#echange des lignes kp et k
for j to n do
interm :=A[k,j] :A[k,j] :=A[kp,j] :A[kp,j] :=interm :
od :
interm :=sig[k] :sig[k] :=sig[kp] :sig[kp] :=interm :
#elimination
for i from k+1 to n do
g :=-A[i,k]/A[k,k] :A[i,k] :=-g :
for j from k+1 to n do
A[i,j] :=A[i,j]+g*A[k,j] :od :od :od :i :='i' :
#affichage du resultat : > #factorisation, permutation, determinant
op(A),op(sig),det*product(A[i,i],i=1..n) :end ;
```

L'algorithme de descente-remontée, en tenant compte de la permutation, est le suivant :

```
desrem :=proc(A,sig,b,x,n)
y :=vector(n) :
y[1] :=b[sig[1]] :
for i from 2 to n do
y[i] :=b[sig[i]]-sum(A[i,j]*y[j],j=1..i-1) :od :
x[n] :=y[n]/A[n,n] :i :='i' :j :='j' :
for i from n-1 by -1 to 1 do
x[i] :=(y[i]-sum(A[i,j]*x[j],j=i+1..n))/A[i,i] :
od :op(x) ;end ;
```

□□

1.3. **Travaux pratiques.**

(1) Réaliser un programme pour résoudre un système linéaire $Au = f$ où A est une matrice tridiagonale.

(2) Tester ce programme en comparant les résultats obtenus avec des résultats connus, dans un cas simple.

(3) En approchant le problème

$$\begin{aligned} -u''(x) &= f(x) \\ u(0) &= u(1) = 0 \end{aligned}$$

par un système linéaire, résoudre numériquement ce problème pour $f(x) = x(x-1)$, $f(x) = x(x-1/2)(x-1)$, $f(x) = x(x-1/3)(x-2/3)(x-1)$ et pour diverses dimensions du système linéaire. Conclusions ?

(4) Résoudre numériquement

$$\begin{aligned} -u''(x) + x(x-1)u(x) &= 1 \\ u(0) &= u(1) = 0 \end{aligned}$$

□□

2. Intégration numérique

2.1. **Préliminaires.** On souhaite disposer d'un moyen d'évaluer numériquement $I(f) = \int_A^B f(t)dt$ où f est une fonction continue sur un intervalle $[A, B]$ avec $A < B$. En effet, dans de nombreuses applications, cette intégrale ne se ramène pas à des fonctions simples, et même si c'est le cas, on cherche une méthode simple et utilisable dans le cas général. Pour cela, nous chercherons une **approximation** de $I(f)$, notée $J(f)$ sous la forme

$$J(f) = \sum_{i=0}^{N} \lambda_i f(x_i)$$

où les points x_i sont dans l'intervalle $[a, b]$. Les x_i sont appelés les **points d'intégration** et les coefficients λ_i les **poids d'intégration**.

2.2. **Intégration des polynômes.** Pour commencer, supposons que f est un **polynôme** de degré $\leq N$, et supposons que les points d'intégration x_i sont imposés avec

$$A \leq x_0 < \cdots < x_N \leq B$$

Dans ce cas, il est possible de trouver un unique jeu de λ_i tel que $I(f)$ soit exactement $J(f)$. En effet, par linéarité de I et J, il suffit que $I(f) = J(f)$ quand

$f = x \to x^i$ pour $0 \leq i \leq N$. Les poids d'intégration sont alors solutions du système linéaire

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_N \\ \vdots & & & \vdots \\ x_0^N & x_1^N & & x_N^N \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} B-A \\ \frac{B^2-A^2}{2} \\ \vdots \\ \frac{B^{N+1}-A^{N+1}}{N+1} \end{bmatrix} \Leftrightarrow A\Lambda = S$$

qui admet une solution et une seule car

$$\det A = (-1)^{E(N/2)+N} \prod_{i<j} (x_i - x_j) \neq 0$$

Exemples :

- $N = 0, x_0 = \frac{A+B}{2}$, on trouve $\lambda_0 = B - A$, soit

$$J(f) = (B-A)\, f(\frac{A+B}{2}) \tag{1}$$

  c'est la méthode des rectangles. On constate que la méthode est encore exacte pour des polynômes de degré ≤ 1.
- $N = 1, x_0 = A, x_1 = B$, on trouve $\lambda_0 = \lambda_1 = \frac{B-A}{2}$, soit

$$J(f) = (B-A)\, \frac{f(A)+f(B)}{2} \tag{2}$$

  c'est la méthode des trapèzes.
- $N = 2, x_0 = A, x_1 = \frac{A+B}{2}, x_2 = B$, on trouve $\lambda_0 = \lambda_2 = \frac{B-A}{6}, \lambda_1 = \frac{2(B-A)}{3}$, soit

$$J(f) = \frac{(B-A)}{6} \left(f(A) + f(B) + 4f(\frac{A+B}{2}) \right) \tag{3}$$

  c'est la méthode de Simpson. Cette méthode est encore exacte pour des polynômes de degré ≤ 3.

2.3. **Intégration des fonctions régulières.** Quand f n'est plus un polynôme, les formules (1), (2), et (3) peuvent encore être écrites. Cependant, on n'aura plus nécessairement $I(f) = J(f)$. Il faut alors se demander en quel sens $I(f)$ est une approximation de $J(f)$. À cette fin, posons $E(f) = J(f) - I(f)$. $E(f)$ est l'**erreur** de la méthode d'intégration. Nous allons évaluer $E(f)$ dans le cas de la méthode des rectangles.

Théorème 1. *Si f est de classe C^2 sur $[A, B]$ alors, $\exists c \in [A, B]$ tel que*

$$E(f) = -\frac{(b-a)^3}{24} f''(c)$$

Démonstration : commençons par écrire la formule de Taylor avec reste intégral

$$f(x) = f(A) + (x-A)f'(A) + \int_A^x (x-t)f''(t)dt$$

La formule des rectangles étant exacte pour des fonctions affines, on a (H est la fonction de Heaviside)

$$\begin{aligned} E(f) &= E(x \to \int_A^x (x-t)f''(t)dt) \\ &= (B-A)\int_A^B \max(0, \frac{A+B}{2}-t)f''(t)dt + \int_{x=A}^B \int_{t=A}^B \max(0,x-t)f''(t)dt \\ &= \int_{t=A}^B \left(\int_{x=A}^B \max(0,\frac{A+B}{2}-t) - \max(0,x-t)dx \right) f''(t)dt \\ &= \int_{t=A}^B \left(H(\frac{A+B}{2}-t)(B-A)(\frac{A+B}{2}-t) - \frac{(B-t)^2}{2} \right) f''(t)dt \end{aligned}$$

or $H(\frac{A+B}{2}-t)(B-A)(\frac{A+B}{2}-t) - \frac{(B-t)^2}{2}$ est une fonction négative sur $[A,B]$ donc $\exists c \in [A,B]$ tel que

$$\begin{aligned} E(f) &= f''(c)\int_{t=A}^B \left(H(\frac{A+B}{2}-t)(B-A)(\frac{A+B}{2}-t) - \frac{(B-t)^2}{2} \right) dt \\ &= -\frac{(B-A)^3}{24} f''(c) \end{aligned}$$

□

Nous admettons le théorème suivant (qui se démontre avec les mêmes techniques que le précédent)

Théorème 2. *Soit une méthode d'intégration J à N points, exacte pour des polynômes de degré $\leq P$ avec $P \geq N$. Alors, si f est de classe C^{P+1} sur $[A,B]$, $\exists c \in [A,B]$ tel que*

$$E(f) = K \cdot (B-A)^{P+2} f^{(P+1)}(c)$$

où $K = K(P)$ est une constante indépendante de f et de A et B.

Nous déduisons de ce théorème la majoration

$$|E(f)| \leq |K(P)| \, |(B-A)|^{P+2} \sup_{c\in[A,B]} \left| f^{(P+1)}(c) \right|$$

malheureusement, nous ne pouvons pas en déduire que $|E(f)| \to 0$ quand $P \to \infty$ car rien ne dit que $|K(P)| \, |(B-A)|^{P+2} \sup_{c\in[A,B]} \left| f^{(P+1)}(c) \right|$ tend vers 0. D'ailleurs, on peut exhiber des fonctions f pour lesquelles $|E(f)| \to \infty$ quand le nombre de points d'intégration augmente ! Nous sommes donc conduits à introduire les méthodes d'intégration **composites**.

2.4. **Méthodes composites.** On part d'une méthode d'intégration sur l'intervalle $[0,1]$ exacte pour des polynômes de degré $\leq N$

$$J(f) = \sum_{i=0}^N \mu_i f(\xi_i)$$

où $0 \leq \xi_0 < \cdots < \xi_N \leq 1$ sont les points d'intégration et les μ_i les poids d'intégration. Considérons maintenant un polynôme Q de degré $\leq N$ défini sur un intervalle $[a,b]$ un changement de variables donne :

$$\int_a^b Q(t)dt = (b-a)\int_0^1 Q((b-a)\xi + a)d\xi = (b-a)\sum_{i=0}^N \mu_i Q((b-a)\xi_i + a) \tag{4}$$

Une méthode d'intégration composite consiste alors à considérer une subdivision de l'intervalle $[A,B]$

$$A \leq x_0 < \cdots < x_M \leq B$$

et à appliquer la formule (4) sur chaque intervalle $[a,b]=[x_i,x_{i+1}]\,, 0\leq i\leq M-1$. Nous approcherons donc $I(f)$ par

$$H(f)=\sum_{k=1}^{M}(x_k-x_{k-1})\sum_{i=0}^{N}\mu_i f((x_k-x_{k-1})\xi_i+x_{k-1})$$

Dans le cas d'une subdivision régulière, $x_k=a+kh$ avec $h=\frac{B-A}{M}$ et

$$H(f)=h\sum_{k=1}^{M}\sum_{i=0}^{N}\mu_i f((k-1+\xi_i)h)$$

en appliquant le théorème (2) à chaque intervalle $[a,b]=[x_i,x_{i+1}]$, on obtient, dans le cas de la subdivision régulière :

Théorème 3.

$$|H(f)-I(f)|\leq|K(P)|\,(B-A)\sup_{c\in[A,B]}\left|f^{(P+1)}(c)\right|\,h^{P+1}$$

Cette fois-ci l'erreur tend bien vers 0 quand $h\to 0$ (c'est à dire quand $M\to\infty$) à P **fixé**.

2.5. **Ordre d'une méthode.** Continuons à étudier les méthodes composites dans le cas d'une subdivision régulière. Pour l'instant nous avons obtenu des **majorations** de l'erreur. Nous nous intéressons ici au cas où l'on dispose d'un équivalent de l'erreur quand $h\to 0$.

Définition 2. *On dit que la méthode est d'ordre L pour la fonction f ssi* $\exists C\neq 0$,

$$|H(f)-I(f)|\sim Ch^L \text{ quand } h\to 0$$

Il est évident que plus l'ordre de la méthode est élevé, plus la convergence de $H(f)$ vers $I(f)$ est rapide quand $h\to 0$. Néanmoins, la taille de la constante C n'est pas à négliger !

Exemples :

– On considère $f(x)=e^x$, et on calcule $\int_0^1 f(t)dt$ par la méthode des rectangles pour $10\leq M\leq 20$. On trace le logarithme de l'erreur en fonction du logarithme de M. Si M est assez grand, on s'attend à ce que la courbe soit presque une droite. Sa pente sera l'opposé de l'ordre de la méthode (car $M=\frac{b-a}{h}$). L'algorithme est le suivant

```
intrect :=proc(f,a,b,M)
h :=evalf((b-a)/M) :
h*sum(f((i+.5)*h),i=0..M-1) ;end ;
f :=x- > exp(x) ;
intrect(f,0,1,10) ;
graf := [seq([ln(1./j),
ln(abs(exp(1.)-1.-intrect(f,0,1,j)))],j=10..20)] ;
plot(graf) ;
```

La pente de la courbe obtenue est de -2. L'ordre de la méthode des rectangles est donc de 2 pour la fonction exponentielle.

– On considère $f(x)=e^x$, et on calcule $\int_0^1 f(t)dt$ par la méthode de Simpson pour $10\leq M\leq 20$. L'algorithme est le suivant

```
intsim :=proc(f,a,b,M)
h :=evalf((b-a)/M) :
h/6.*sum(f(i*h)+4.*f((i+.5)*h)+f((i+1.)*h),i=0..M-1) ;end ;
f :=x- > exp(x) ;
intsim(f,0,1,10) ;
graf := [seq([ln(1./j),
```

```
ln(abs(exp(1.)-1.-intrect(f,0,1,j)))],j=10..20)] ;
plot(graf) ;
```

L'opposé de la pente de la courbe obtenue est de 4. C'est l'ordre de la méthode de Simpson pour la fonction exponentielle.

– On considère $f(x) = \ln x$, et on calcule $\int_0^1 f(t)dt$ par la méthode des rectangles. On trouve que l'ordre est 1 ! Ceci montre que la régularité de la fonction est indispensable dans le théorème (2).

2.6. **Méthode de Gauss composite.** La méthode de Gauss consiste à déterminer les points d'intégration ξ_i sur l'intervalle $[0,1]$ de sorte que la méthode

$$J(f) = \sum_{i=0}^{N} \mu_i f(\xi_i)$$

déjà exacte pour des polynômes de degré $\leq N$ soit exacte pour des polynômes de degré le plus élevé possible. A priori, on dispose de $N+1$ degrés de liberté supplémentaires, on s'attend donc à ce que la méthode puisse être exacte pour des polynômes de degré $\leq 2N+1$.

Considérons alors sur l'ensemble des polynômes le produit scalaire

$$\langle P, Q\rangle = \int_0^1 P(t)Q(t)dt$$

En orthonormalisant la suite de polynômes $n \to x^n$, par le procédé de Gramm-Schmidt, on obtient

$$\begin{aligned}
L_0(x) &= 1 \\
L_1(x) &= 2\sqrt{3}\left(x - \frac{1}{2}\right) \\
L_2(x) &= 6\sqrt{5}\left(x^2 - x + \frac{1}{6}\right) \\
L_3(x) &= 20\sqrt{7}\left(x^3 - \frac{3}{2}\,x^2 + \frac{3}{5}\,x - \frac{1}{20}\right) \\
L_4(x) &= 210\left(x^4 - 2\,x^3 + \frac{9}{7}\,x^2 - \frac{2}{7}\,x + \frac{1}{70}\right) \\
&\vdots
\end{aligned}$$

nous admettrons que L_k admet k zéros distincts dans $]0,1[$. Nous allons alors montrer

Théorème 4. *Si les ξ_i sont les zéros de L_{N+1} alors la méthode*

$$J(f) = \sum_{i=0}^{N} \mu_i f(\xi_i)$$

est exacte pour des polynômes de degré $\leq 2N+1$.

Démonstration : soit P un polynôme de degré $\leq 2N+1$. La division euclidienne de P par L_{N+1}donne

$$P = qL_{N+1} + r$$

où r est un polynôme de degré $\leq N$ et q un polynôme de degré $\leq N$. Donc, par orthogonalité des polynômes L_k, et parce que les ξ_i sont racines de L_{N+1} :

$$\begin{aligned}\int_0^1 P(t)dt &= \int_0^1 q(t)L_{N+1}(t)dt + \int_0^1 r(t)dt = \int_0^1 r(t)dt \\ &= \sum_{i=0}^{N} \mu_i r(\xi_i) = \sum_{i=0}^{N} \mu_i (qL_{N+1}+r)(\xi_i) = \sum_{i=0}^{N} \mu_i P(\xi_i)\end{aligned}$$

□

Exemples :

- $N = 1 \Rightarrow \xi_0 = \frac{1}{2} - \frac{\sqrt{3}}{6}, \xi_1 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. La méthode est d'ordre 4 pour une fonction régulière.
- $N = 2 \Rightarrow \xi_0 = \frac{1}{2} - \frac{\sqrt{10}}{15}, \xi_1 = \frac{1}{2}, \xi_2 = \frac{1}{2} + \frac{\sqrt{10}}{15}$. La méthode est d'ordre 6 pour une fonction régulière.

□□

3. Interpolation des fonctions

3.1. **Interpolation de Lagrange.** On se donne une fonction continue, $f : [a, b] \to R$, et une subdivision de $[a, b]$

$$a = x_0 < \cdots < x_N = b$$

le problème de l'**interpolation** est alors de déterminer un polynôme P de degré $\leq N$ tel que

$$\forall i = 0 \cdots N,\ P(x_i) = f(x_i)$$

P est dit polynôme d'interpolation de f relatif à la subdivision x.

Théorème 5. *Pour toute subdivision x de points distincts, il existe un unique polynôme d'interpolation de f.*

Démonstration : si $P(t) = a_0 + a_1 x + \cdots + a_N x^N$, les coefficients de P sont solutions de

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^N \\ 1 & x_1 & \cdots & x_1^N \\ \vdots & & & \vdots \\ 1 & x_N & & x_N^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}$$

la matrice de ce système linéaire est une matrice de Vandermonde de déterminant $\neq 0$ car les points x_i sont distincts.□

Théorème 6. *Soit $f : [a, b] \to R$ de classe C^{N+1}. Soit P_N le polynôme d'interpolation associé à f relatif à la subdivision $a = x_0 < \cdots < x_N = b$ alors $\exists c \in [a, b]$*

$$f(x) - P_N(x) = (x - x_0) \cdots (x - x_N) \frac{f^{(N+1)}(c)}{(N+1)!}$$

Démonstration : voir les exercices□

On constate, comme pour l'intégration numérique, que si on souhaite **approcher** f il ne **faut pas**, en général, faire tendre N vers l'infini ! Il est préférable de **fixer** N, de subdiviser l'intervalle $[a, b]$ en M sous-intervalles égaux sur lesquels on interpole f avec N points. On peut alors faire tendre M vers l'infini. L'inconvénient de cette approche est que l'approximation de f est continue, polynômiale par morceaux, mais n'est pas dérivable, en général, aux extrémités des M sous-intervalles.

Exemple : l'interpolation P_N de $x \to \frac{1}{1+x^2}$ sur l'intervalle $[-1, 1]$ pour la subdivision $x_i = -1 + \frac{2i}{N}, i = 0 \cdots N$ ne converge pas vers f quand M tend vers l'infini (on observe des oscillations de P_N de plus en plus importantes. Ce comportement de P_N est appelé phénomène de Runge).

Mais il existe une autre difficulté que nous allons aborder sur un exemple

Exemple : soit $f(x) = x$ sur $[0,1]$. Soit la subdivision $x_i = \frac{i}{N}, i = 0 \cdots N$. La résolution de

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^N \\ 1 & x_1 & \cdots & x_1^N \\ \vdots & & & \vdots \\ 1 & x_N & & x_N^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}$$

conduit bien sûr à $(a_0, a_1, \cdots, a_N) = (0, 1, 0, \cdots, 0)$. Or si l'on tente de résoudre ce système avec $N = 20$, **numériquement** pour une précision de 10 chiffres, grâce à la méthode LU (avec ou sans pivotage), les résultats obtenus n'ont rien à voir avec la solution exacte. L'erreur peut atteindre, suivant la machine avec laquelle on travaille, plusieurs centaines de milliers ! Le système de Vandermonde appartient à la catégorie (heureusement plutôt rare) des systèmes linéaires **mal conditionnés**. On appelle ainsi un système pour lequel une petite erreur sur les coefficients ou le second membre entraîne une erreur importante sur la solution du système. Il est possible de savoir si un système $Ax = b$ est bien ou mal conditionné en connaissant sont **conditionnement**, défini par

$$\text{cond}(A) = \|A\| \left\|A^{-1}\right\|$$

où $\|\cdot\|$ est une norme matricielle quelconque. On a toujours $\text{cond}(A) \geq 1$, et plus ce nombre est grand plus la résolution du système est difficile. Dans notre exemple $\text{cond}(A) = 3565516259013508 80$. Par ailleurs, $\text{cond}(A) = 1$ ssi A est une matrice orthogonale ($A^{-1} = A^T$). Les systèmes linéaires les mieux conditionnés sont donc ceux dans lesquels intervient une matrice orthogonale.

Heureusement, il existe d'autres méthodes pour calculer le polynôme d'interpolation

3.2. Polynômes de Lagrange.

Définition 3. *Soit $a = x_0 < \cdots < x_N = b$ une subdivision de l'intervalle $[a,b]$. On appelle $i^{ème}$ polynôme de Lagrange associé à la subdivision et on note L_i le polynôme*

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

L_i est un polynôme de degré N, $(L_0, \cdots, L_N)$ constitue une base de l'ensemble des polynômes de degré $\leq N$, de plus

$$L_i(x_j) = \begin{cases} 1 \text{ si } j = i \\ 0 \text{ si } j \neq i \end{cases}$$

Théorème 7. *Soit $f : [a,b] \to R$. Soit P_N le polynôme d'interpolation associé à f relatif à la subdivision $a = x_0 < \cdots < x_N = b$. P_N est alors donné par*

$$P_N = \sum_{k=0}^{N} f(x_k) L_k$$

3.3. Interpolation par fonctions splines.

3.3.1. *Introduction.* Les inconvénients de l'interpolation de Lagrange sont :

- possibilités d'oscillations si le degré de l'interpolation est trop élevé.
- l'approximation n'est pas dérivable quand on utilise l'interpolation par morceaux.

Nous allons maintenant décrire une modification de l'interpolation de Lagrange qui permet d'éviter ces désavantages.

Soit $a \leq x_0 < \cdots < x_N \leq b$ une subdivision de l'intervalle $[a,b]$, et $f : [a,b] \to R$, une fonction. Nous noterons $f(x_i) = y_i$.

Définition 4. *Une fonction σ définie sur $[a,b]$ est appelée une spline d'interpolation de f relative à la subdivision x_i ssi*

- *σ est de classe C^1 sur $[a,b]$ (donc $\sigma'(x_i^+) = \sigma'(x_{i-1}^-)$ pour $i = 1 \cdots N-1$)*
- *σ est de classe C^∞ sur $[x_i, x_{i+1}], i = 0 \cdots N-1$ (σ'' n'est pas forcément continue sur $[a,b]$)*
- *$\sigma(x_i) = y_i$*

Remarque : spline signifie latte en Anglais. Une latte de bois souple était effectivement utilisée par les dessinateurs industriels pour tracer des courbes dérivables passant par des points donnés.

Il existe bien sûr une infinité de splines d'interpolation de f (dont le polynôme d'interpolation). En fait nous allons chercher parmi toutes les splines celles dont les oscillations sont minimales sur l'intervalle $[a,b]$ au sens où

$$\int_a^b \sigma''(t)^2 dt \text{ est minimale}$$

En termes physiques, la quantité $\int_a^b \sigma''(t)^2 dt$ est d'ailleurs l'énergie de flexion de la latte. Nous allons donc calculer la position effective d'une latte flexible qui passe par les points $(x_i, f(x_i))$.

3.3.2. *Caractérisation.*

Théorème 8. *Il existe une unique spline d'interpolation dont l'énergie est minimale. Cette spline σ vérifie :*

- *la restriction de σ à $[x_i, x_{i+1}]$ est un polynôme de degré ≤ 3 pour $i = 0 \cdots N-1$*
- *$\sigma(x_0) = y_0, \sigma(x_N) = y_N$*
- *$\sigma(x_i^+) = \sigma(x_i^-) = y_i$ pour $i = 1 \cdots N-1$*
- *$\sigma'(x_i^+) = \sigma'(x_i^-)$ pour $i = 1 \cdots N-1$*
- *$\sigma''(x_0) = \sigma''(x_N) = 0$*
- *$\sigma''(x_i^+) = \sigma''(x_i^-)$ pour $i = 1 \cdots N-1$*

Idée de la démonstration : Soit C l'ensemble des splines d'interpolation de f relativement à x_i

$$C = \left\{\sigma, \sigma \in C^1[a,b] \cap C^\infty[x_i, x_{i+1}], i = 0 \cdots N-1, \sigma(x_i) = y_i\right\}$$

C est un ensemble convexe. Soit V l'espace vectoriel

$$V = \left\{\varphi, \varphi \in C^1[a,b] \cap C^\infty[x_i, x_{i+1}], i = 0 \cdots N-1, \varphi(x_i) = 0\right\}$$

On a $C + tV \subset C$ pour tout réel t. Soit $E(\sigma) = \int_a^b \sigma''(t)^2 dt$. Pour tout φ dans V, et tout réel t, on a

$$E(\sigma) \leq E(\sigma + t\varphi)$$

par conséquent,

$$\frac{d}{dt} E(\sigma + t\varphi)\Big|_{t=0} = 0$$

ce qui se traduit (après calculs) par

$$\forall \varphi \in V, \quad \int_a^b \sigma''(t)\varphi''(t) = 0$$

On considère alors une fonction φ à support dans $]x_i, x_{i+1}[$ (le support d'une fonction est l'adhérence du plus grand ouvert sur lequel la fonction n'est pas nulle), nous avons alors, après intégrations par parties sur $[x_i, x_{i+1}]$

$$\int_a^b \sigma^{(4)}(t)\varphi(t) = 0$$

et ceci étant vrai pour toute fonction à support dans $]x_i, x_{i+1}[$, on en déduit que sur cet ouvert

$$\sigma^{(4)}(x) = 0$$

d'où la première assertion. Les assertions 2, 3 et 4 sont évidentes car $\sigma \in C$

Enfin, pour $\varphi \in C$, on a (en convenant de prolonger σ par 0 en dehors de $[a, b]$)

$$\begin{aligned}
\int_a^b \sigma''(t)\varphi''(t) &= -\int_a^b \sigma'''(t)\varphi'(t) + \sum_{i=0}^{N} \varphi'(x_i)\left(\sigma''(x_i^-) - \sigma''(x_i^+)\right) \\
&= \int_a^b \sigma^{(4)}(t)\varphi(t) + \sum_{i=0}^{N} \varphi'(x_i)\left(\sigma''(x_i^-) - \sigma''(x_i^+)\right) \\
&\quad + \sum_{i=0}^{N} \varphi(x_i)\left(\sigma'''(x_i^+) - \sigma'''(x_i^-)\right) \\
&= \sum_{i=0}^{N} \varphi'(x_i)\left(\sigma''(x_i^-) - \sigma''(x_i^+)\right) = 0
\end{aligned}$$

en choisissant convenablement φ, on en déduit les assertions 5 et 6.□

3.3.3. *Calcul de la spline.* Pour calculer pratiquement la spline minimale, on pose $M_i = \sigma''(x_i)$. En particulier, $M_0 = M_N = 0$. Sur l'intervalle $[x_i, x_{i+1}]$, σ est alors donné par

$$\begin{aligned}
\sigma(x) &= 1/6\,\frac{(x_{i+1}-x)\,(x-x_i)\,(x-2\,x_{i+1}+x_i)}{x_{i+1}-x_i} M_i \\
&\quad -1/6\,\frac{(x_{i+1}-x)\,(x-x_i)\,(x-2\,x_i+x_{i+1})}{x_{i+1}-x_i} M_{i+1} \\
&\quad + \frac{y_{i+1}-y_i}{x_{i+1}-x_i} x + \frac{y_i x_{i+1} - x_i y_{i+1}}{x_{i+1}-x_i}
\end{aligned}$$

de plus

$$\begin{aligned}
\sigma'(x_i) &= -1/3\,M_i\,(x_{i+1}-x_i) - 1/6\,M_{i+1}\,(x_{i+1}-x_i) + \frac{y_{i+1}-y_i}{x_{i+1}-x_i} \\
\sigma'(x_{i+1}) &= 1/6\,M_i\,(x_{i+1}-x_i) + 1/3\,M_{i+1}\,(x_{i+1}-x_i) + \frac{y_{i+1}-y_i}{x_{i+1}-x_i}
\end{aligned}$$

le raccord des dérivées au point x_i implique donc

$$\begin{aligned}
1/6\,M_{i+1}\,(x_{i+1}-x_i) + 1/6\,M_{i-1}\,(x_i - x_{i-1}) + & \quad 1/3\,M_i\,(x_{i+1}-x_{i-1}) \\
= & \quad \frac{y_{i+1}-y_i}{x_{i+1}-x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}
\end{aligned}$$

Pour simplifier, considérons une subdivision régulière ($x_{i+1} - x_i = h = \frac{b-a}{N}$), les M_i sont alors solutions de

$$\begin{bmatrix} 2/3 & 1/6 & & & 0 \\ 1/6 & 2/3 & 1/6 & & \\ & 1/6 & \ddots & \ddots & 0 \\ & & \ddots & \ddots & 1/6 \\ 0 & & & 1/6 & 2/3 \end{bmatrix} \begin{bmatrix} M_1 \\ \vdots \\ \\ \\ M_{N-1} \end{bmatrix} = \begin{bmatrix} \frac{y_2-2y_1+y_0}{h^2} \\ \vdots \\ \\ \\ \frac{y_N-2y_{N-1}+y_{N-2}}{h^2} \end{bmatrix}$$

il suffit donc de résoudre un système tridiagonal.

□□

4. Résolution numérique des équations non-linéaires

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

En général, les méthodes de résolution de $f(\overline{x}) = 0$ sont des méthodes itératives. Elles consistent à construire une suite x_n convergente (le plus rapidement possible) vers $\overline{x}$. Nous envisagerons le cas où cette suite est de la forme

$$\begin{aligned} x_0 &\in \Omega \\ x_{n+1} &= g(x_n) \end{aligned}$$

où g est une application de Ω dans Ω continue.

4.2. **Le théorème du point fixe.** Tout le monde a déjà entendu parler de

Théorème 9. *Soit g une application définie sur Ω fermé, qui vérifie*

(1) *$g(\Omega) \subset \Omega$ (cette condition est indispensable pour que la suite x_n soit définie !)*

(2) *g est $K-$Lipschitzienne sur Ω avec $K < 1$, c'est à dire*

$$\exists K, 0 < K < 1, \forall (x, y) \in \Omega \times \Omega, \|g(x) - g(y)\| \leq K \|x - y\|$$

Alors l'équation $g(\overline{x}) = \overline{x}$ admet une solution et une seule dans Ω. De plus pour toute suite de la forme

$$\begin{aligned} x_0 &\in \Omega \\ x_{n+1} &= g(x_n) \end{aligned}$$

on a $\lim_{n\to\infty} x_n = \overline{x}$.

Démonstration :

(1) unicité : soient x_1 et x_2, deux solutions. On a :

$$\|x_1 - x_2\| \leq K \|x_1 - x_2\|$$

et $K < 1 \Rightarrow x_1 = x_2$

(2) existence : soit la suite

$$\begin{aligned} x_0 &\in \Omega \\ x_{n+1} &= g(x_n) \end{aligned}$$

Puisque f est $K-$Lipschitzienne, on a

$$\|x_{n+1} - x_n\| \leq K \|x_n - x_{n-1}\|$$

donc

$$\|x_{n+1} - x_n\| \leq K^n \|x_1 - x_0\| \tag{5}$$

Calculons maintenant, pour $p > n$

$$\sum_{k=n+1}^{p} x_k - x_{k-1} = x_p - x_n$$

par (5), on obtient

$$\begin{aligned}\|x_p - x_n\| &\leq \sum_{k=n+1}^{p} K^{k-1} \|x_1 - x_0\| \leq K^n \|x_1 - x_0\| \sum_{k=0}^{\infty} K^k \\ &= \frac{K^n}{1-K} \|x_1 - x_0\| \to 0 \text{ quand } (n,p) \to \infty\end{aligned}$$

la suite (x_n) est donc de Cauchy dans Ω, donc convergente dans Ω car Ω est un fermé dans un complet donc complet. De plus, soit l, la limite de cette suite, par continuité de g, $g(l) = l$, d'où l'existence du point fixe.□

Il est important de disposer de critère pratique assurant qu'une fonction g est $K-$Lipschitzienne, avec $K < 1$.

Théorème 10. *Si $\|g'(x)\| \leq K$ pour tout x dans Ω convexe, alors, g est $K-$Lipschitzienne sur Ω.*

Démonstration : soient x et y dans Ω. Puisque le segment $[x, y]$ est inclus dans Ω, on a

$$g(y) - g(x) = \int_0^1 g'(x + t(x-y))(x-y)$$

donc

$$\begin{aligned}\|g(y) - g(x)\| &\leq \int_0^1 \|g'(x + t(x-y))(x-y)\| \, dt \\ &\leq \int_0^1 \|g'(x + t(x-y))\| \, \|(x-y)\| \, dt \\ &\leq K \|(x-y)\|\end{aligned}$$

□

En général, la méthode du point fixe, quand elle converge, est une méthode d'ordre 1. C'est à dire que, en gros, à chaque itération l'erreur est multipliée par un coefficient plus petit que un. Voici un énoncé plus précis

Théorème 11. *Supposons que la méthode du point fixe converge et que g est de classe C^2, alors*

$$x_{n+1} - \overline{x} = g'(\overline{x})(x_n - \overline{x}) + O(\|x_n - \overline{x}\|^2)$$

en particulier, en dimension un d'espace, si $x_n \neq \overline{x}$,

$$\frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} \to g'(\overline{x})$$

et donc, $|g'(\overline{x})| \leq 1$.

Nous sommes conduits à la définition suivante

Définition 5. *Une méthode du point fixe est d'ordre k ssi*

$$\frac{x_{n+1} - \overline{x}}{(x_n - \overline{x})^k} \to l \text{ avec } 0 < |l| < +\infty$$

4.3. **Méthode de Newton.** La méthode de Newton est un cas particulier de la méthode du point fixe. Quand elle converge, elle est en général d'ordre 2, d'où son intérêt. L'idée est la suivante :

Supposons que l'on connaisse à l'étape n une approximation x_n de la solution $\overline{x}$. Posons

$$x_{n+1} = x_n + dx_n$$

Nous allons chercher un accroissement dx_n qui améliore la précision de l'approximation. Un développement de Taylor donne

$$f(x_n + dx_n) = f(x_n) + f'(x_n)dx_n + O(\|dx_n\|^2)$$

On souhaite bien sûr que $f(x_n + dx_n)$ soit le plus proche possible de zéro. En négligeant les termes du second ordre, il est donc logique de résoudre

$$f(x_n) + f'(x_n)dx_n = 0$$

soit

$$x_{n+1} = x_n - f'(x_n)^{-1} f(x_n)$$

Bien sûr, la suite (x_n) n'est pas toujours définie. Il faut, en particulier, que $f'(x_n)$ soit une matrice inversible et que $x_{n+1} \in \Omega$. Pour étudier cette suite, on peut par exemple essayer d'appliquer le théorème du point fixe à la fonction $g : x \to g(x) = x - f'(x)^{-1} f(x)$.

Vérifions, dans le cas de la dimension 1, que la méthode est, sous certaines hypothèses, une méthode d'ordre 2.

Théorème 12. *Soit f de classe C^2 sur $[a, b]$ admettant une unique racine $\overline{x}$ dans $[a, b]$. On suppose que :*
- *$|f'(x)| \geq m > 0$ et $|f''(x)| \leq M$ sur $[a, b]$*
- *$x_0 \in [\overline{x} - \eta, \overline{x} + \eta]$ avec $\eta < \frac{2m}{M}$*

alors la méthode de Newton converge et

$$|x_n - \overline{x}| \leq |x_0 - \overline{x}|^{2^n} \left(\frac{M}{2m}\right)^{2^n - 1}$$

Démonstration : la suite x_n est définie par

$$x_{n+1} - \overline{x} = x_n - \overline{x} - f'(x_n)^{-1} \left(f(x_n) - f(\overline{x})\right)$$

par ailleurs,

$$f(x_n) - f(\overline{x}) = (x_n - \overline{x})\, f'(x_n) + \frac{(x_n - \overline{x})^2}{2} f''(\xi) \text{ avec } \xi \in]x_n, \overline{x}[$$

donc

$$x_{n+1} - \overline{x} = (x_n - \overline{x})^2 \frac{f''(\xi)}{2f'(x_n)}$$

et

$$|x_{n+1} - \overline{x}| \leq |x_n - \overline{x}|^2 \frac{M}{2m}$$

ce qui prouve l'existence de la suite et sa convergence. On a aussi

$$\frac{x_{n+1} - \overline{x}}{(x_n - \overline{x})^2} \to \frac{f''(\overline{x})}{2f'(\overline{x})}$$

Si $f''(\overline{x}) \neq 0$, la méthode est d'ordre 2.□

4.4. **Exercices.** 1) Montrer que l'équation $f(x) = x^3 - 2$ possède une solution unique α et qu'on peut obtenir celle-ci en utilisant la méthode de Newton à partir de $x_0 = 1$. Donner une minoration du nombre d'itérations à effectuer pour obtenir une précision $\varepsilon = 10^{-10}$

□□

5. Résolution numérique des équations différentielles

5.1. **Rappels.** Soient P un entier ≥ 1, T un réel > 0, α un élément de R^P et f une application continue de $[0,T]\times R^P$ à valeurs dans R^P. On cherche une solution $y:[0,T]\to R^P$ à l'équation différentielle

$$\begin{aligned} y'(t) &= f(t,y(t)) \quad \forall t\in[0,T] \\ y(0) &= \alpha \end{aligned} \tag{6}$$

Pour simplifier l'exposé, nous poserons $P=1$. Mais les résultats et méthodes décrits dans le cas scalaire s'étendent sans difficulté au cas vectoriel. Rappelons le théorème de Cauchy-Lipschitz

Théorème 13. *Si f est Lipschitzienne par rapport à y, c'est à dire s'il existe $K>0$ tel que*

$$\forall (t,y,y')\in[0,T]\times R^P\times R^P, |f(t,y)-f(t,y')|\leq K\,|y-y'|$$

alors le problème (6) admet une unique solution (pour T assez petit).

Il est souhaitable, avant d'aborder l'analyse numérique de ce type de problème, que ce théorème soit applicable. En effet, considérons l'exemple suivant

$$\begin{aligned} y' &= \sqrt{|y|} \\ y(0) &= 0 \end{aligned}$$

il est clair que $y(t)=0$ et $y(t)=\frac{t^2}{4}$ sont solutions. Il est difficile d'imaginer une méthode permettant d'approcher plusieurs solutions en même temps !

5.2. **Méthode d'Euler explicite.**
Principe. Posons $\Delta t = h = T/N$ où N est un entier ≥ 1. Si t représente le temps, Δt est souvent appelé le pas de temps. Soit $t_n = n\Delta t$ une subdivision régulière de l'intervalle $[0,T]$. Il est aussi possible (et même souhaitable dans les applications !) d'envisager des subdivisions irrégulières. Nous allons chercher un vecteur $(y_n)_{0\leq n\leq N}$ tel que $y_n\simeq y(t_n)$ où y est la solution exacte de (6).

Puisque

$$\begin{aligned} \frac{y(t_{n+1})-y(t_n)}{\Delta t} &= y'(t_n)+\varepsilon(\Delta t) \text{ avec } \lim_{\Delta t\to 0}\varepsilon(\Delta t)=0 \\ &= f(t_n,y(t_n))+\varepsilon(\Delta t) \end{aligned}$$

il est naturel de résoudre

$$\begin{aligned} \frac{y_{n+1}-y_n}{\Delta t} &= f(t_n,y_n) \quad 1\leq n\leq N \\ y_0 &= \alpha \end{aligned}$$

Le vecteur y peut ainsi se calculer par récurrence de proche en proche. On dit qu'on utilise une **méthode** ou un **schéma** de résolution **explicite**.
Exemple. Soit l'équation ($\lambda\neq 0$)

$$\begin{aligned} y' &= -\lambda y \\ y(0) &= 1 \end{aligned}$$

dont la solution est $y(t)=e^{-\lambda t}$. La méthode d'Euler explicite donne $y_n=(1-\lambda\Delta t)^n$. En posant $t=n\Delta t$, et en faisant $\Delta t\to 0$, on obtient

$$y_n=(1-\lambda\Delta t)^{\frac{t}{\Delta t}}\to e^{-\lambda t}=y(t)$$

Dans ce cas, la méthode d'Euler permet d'approcher $y(t)$ quand $\Delta t\to 0$. On dira que la méthode est convergente.

Supposons maintenant $\lambda > 0$. Il est alors clair que $y(t) \to 0$ quand $t \to \infty$. Il est bien évidemment souhaitable que ce soit aussi le cas pour y_n quand $n \to \infty$. Or ceci n'est vrai que si $|1 - \lambda\Delta t| < 1$ soit

$$\Delta t < \frac{2}{\lambda}$$

Cette condition (dite de stabilité) doit donc être vérifiée par Δt pour que la solution discrète y_n "ressemble" à la solution y. Elle est contraignante si λ est très grand en valeur absolue.

5.3. **Méthode d'Euler implicite.**
Principe. Puisque

$$\begin{aligned} \frac{y(t_{n+1}) - y(t_n)}{\Delta t} &= y'(t_{n+1}) + \varepsilon(\Delta t) \text{ avec } \lim_{\Delta t \to 0} \varepsilon(\Delta t) = 0 \\ &= f(t_{n+1}, y(t_{n+1})) + \varepsilon(\Delta t) \end{aligned}$$

il est également naturel de résoudre

$$\begin{aligned} \frac{y_{n+1} - y_n}{\Delta t} &= f(t_{n+1}, y_{n+1}) \quad 1 \leq n \leq N \\ y_0 &= \alpha \end{aligned}$$

Le vecteur y peut se calculer par récurrence de proche en proche, à condition de savoir résoudre à chaque étape une équation en y_{n+1}. On dit qu'on utilise une **méthode** ou un **schéma** de résolution **implicite**.
Exemple. Soit l'équation ($\lambda \neq 0$)

$$\begin{aligned} y' &= -\lambda y \\ y(0) &= 1 \end{aligned}$$

dont la solution est $y(t) = e^{-\lambda t}$. La méthode d'Euler implicite donne $y_n = (1 + \lambda\Delta t)^{-n}$. En posant $t = n\Delta t$, et en faisant $\Delta t \to 0$, on obtient

$$y_n = (1 + \lambda\Delta t)^{\frac{-t}{\Delta t}} \to e^{-\lambda t} = y(t) \tag{7}$$

Dans ce cas, la méthode d'Euler permet aussi d'approcher $y(t)$ quand $\Delta t \to 0$.

Supposons maintenant $\lambda > 0$. Il est alors clair que $y(t) \to 0$ quand $t \to \infty$. C'est aussi le cas pour y_n quand $n \to \infty$ indépendamment de Δt. Le schéma d'Euler implicite est dit **inconditionnellement stable**.

La méthode implicite est plus difficile à mettre en oeuvre car elle implique la résolution d'une équation à chaque étape. Cependant, elle permet d'obtenir, en général, des résultats plus précis, avec un pas de temps moins grand.

5.4. **Méthode explicite à un pas.** Nous allons dans la suite de ce cours nous attacher aux méthodes explicites à un pas. Ce sont des schémas qui s'écrivent

$$\begin{aligned} y_{n+1} &= y_n + \Delta t\, \Phi(t_n, y_n, \Delta t) \\ y_0 &= \alpha \end{aligned} \tag{8}$$

La méthode d'Euler explicite entre dans ce cadre, avec $\Phi(t_n, y_n, \Delta t) = f(t_n, y_n)$.

5.4.1. *Ordre d'une méthode.*

Définition 6. *Soit y la solution de (6). La méthode (8) sera dite consistante à l'ordre l ssi*

$$\sup_{t \in [0,T]} \left| \frac{y(t + \Delta t) - y(t)}{\Delta t} - \Phi(t, y(t), \Delta t) \right| = O(\Delta t^l)$$

nous admettrons qu'il suffit de vérifier que

$$\forall t \in [0, T] \quad \left| \frac{y(t + \Delta t) - y(t)}{\Delta t} - \Phi(t, y(t), \Delta t) \right| = O(\Delta t^l)$$

Il est facile de vérifier que la méthode d'Euler explicite est d'ordre 1

5.4.2. *Stabilité.* Nous allons définir une autre notion de stabilité, à ne pas confondre avec celle introduite en (7).

Définition 7. *La méthode (8) est stable ssi il existe deux constantes A et $B > 0$ telles que si y_n, z_n, ε_n sont solutions de*

$$\begin{aligned} y_{n+1} &= y_n + \Delta t\, \Phi(t_n, y_n, \Delta t) \\ y_0 &= \alpha \\ z_{n+1} &= z_n + \Delta t\, (\, \Phi(t_n, z_n, \Delta t) + \varepsilon_n) \\ z_0 &= \beta \end{aligned}$$

alors, pour tout $n \leq N$

$$|y_n - z_n| \leq A\,|\alpha - \beta| + B \max_{0\leq i\leq N} |\varepsilon_i|$$

Le résultat principal de stabilité est le suivant

Théorème 14. *Si Φ est lipschitzienne par rapport à y alors la méthode est stable.*

démonstration : soient

$$\begin{aligned} y_{n+1} &= y_n + \Delta t\, \Phi(t_n, y_n, \Delta t) \\ y_0 &= \alpha \\ z_{n+1} &= z_n + \Delta t\, (\, \Phi(t_n, z_n, \Delta t) + \varepsilon_n) \\ z_0 &= \beta \end{aligned}$$

on a

$$\begin{aligned} |y_{n+1} - z_{n+1}| &\leq |y_n - z_n| + \Delta t\,|\varepsilon_n| + \Delta t\,|\Phi(t_n, y_n, \Delta t) - \Phi(t_n, z_n, \Delta t)| \\ &\leq (1 + L\Delta t)\,|y_n - z_n| + \Delta t\,|\varepsilon_n| \end{aligned}$$

il est alors facile de montrer (par récurrence) que

$$|y_n - z_n| \leq (1 + L\Delta t)^n\,|y_0 - z_0| + \frac{(1 + L\Delta t)^n - 1}{L} \max_{p\leq n} |\varepsilon_p|$$

par ailleurs, si $k > 0$

$$1 + k \leq e^k$$

donc

$$(1 + L\Delta t)^n \leq e^{Ln\Delta t} \leq e^{LT} \text{ car } 0 \leq n\Delta t \leq N\Delta t = T$$

on obtient alors

$$|y_n - z_n| \leq e^{LT}\,|y_0 - z_0| + \frac{e^{LT} - 1}{L} \max_{p\leq n} |\varepsilon_p|$$

et la méthode est stable.□

5.4.3. *Convergence.*

Définition 8. *Une méthode est convergente à l'ordre l ssi*

$$\lim_{N\to\infty} \max_{0\leq i\leq N} |y_i - y(t_i)| = O(\Delta t^l)$$

Théorème 15. *Une méthode stable et consistante est convergente*

Démonstration : la méthode étant consistante, on a

$$y(t_{n+1}) = y(t_n) + \Delta t(\Phi(t_n, y(t_n), \Delta t) + \varepsilon_n)$$

avec

$$\max_n |\varepsilon_n| = O(\Delta t^l)$$

et

$$y_{n+1} = y_n + \Delta t \Phi(t_n, y_n, \Delta t)$$

la stabilité implique alors

$$|y_n - y(t_n)| \leq O(\Delta t^l)$$

d'où la convergence.□

5.4.4. *Exemples.*
Méthode d'Euler améliorée. Cette méthode s'écrit

$$\Phi(t, y, h) = f(t + \frac{h}{2}, y + \frac{h}{2} f(t, y))$$

elle est stable et consistante à l'ordre 2.
Méthode de Runge-Kutta d'ordre 4. Cette méthode s'écrit

$$\Phi(t, y, h) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

avec

$$\begin{aligned} k_1 &= f(t, y) \\ k_2 &= f(t + \frac{h}{2}, y + \frac{h}{2} k_1) \\ k_3 &= f(t + \frac{h}{2}, y + \frac{h}{2} k_2) \\ k_4 &= f(t + h, y + h k_3) \end{aligned}$$

elle est stable et consistante à l'ordre 4.
□□

6. Introduction à la résolution itérative des systèmes linéaires

6.1. **Introduction.** Soit le système linéaire

$$Ax = b$$

où A est une matrice $N \times N$ inversible. Une méthode itérative de résolution de ce système consiste à construire une suite $(x^{(n)})$ de vecteurs, convergente vers l'unique solution $\overline{x}$. L'intérêt des méthodes itératives, comparées aux méthodes directes, et d'être simple à programmer et de nécessiter moins de place en mémoire. En revanche le temps de calcul est souvent plus long.

En pratique, les méthodes les plus efficaces consistent à mélanger les techniques directes et itératives : on commence par calculer, de façon approchée, en économisant la mémoire, une factorisation LU de A. Puis on lance une méthode itérative basée sur cette factorisation incomplète. La convergence est alors très rapide.

6.2. **Méthodes de Jacobi et de Gauss-Seidel.**

6.2.1. *Méthode de Jacobi pour les matrices à diagonale dominante.* Les algorithmes que nous allons décrire s'applique à une famille de matrices que l'on rencontre souvent en pratique.

Définition 9. *La matrice $A = (a_{ij})_{1\leq i\leq N 1\leq j\leq N}$ est dite à diagonale strictement dominante ssi*

$$\forall i \quad |a_{ii}| > \sum_{k\neq i} |a_{ik}|$$

L'algorithme de Jacobi s'écrit alors, un vecteur $x^{(0)}$ étant donné

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

Il s'écrit aussi

$$x^{(n+1)} = D^{-1} \left(b - (L+U) x^{(n)} \right)$$

où l'on a posé

$$\begin{array}{rcl} A & = & L + D + U \\ L & : & \text{triangulaire inférieure} \\ D & : & \text{diagonale de } A \\ U & : & \text{triangulaire supérieure} \end{array}$$

L'algorithme de Jacobi est, on le voit, particulièrement simple. Il nécessite le stockage des deux vecteurs $x^{(n)}$ et $x^{(n+1)}$

6.2.2. *Méthode de Gauss-Seidel.* L'algorithme de Gauss-Seidel s'écrit, un vecteur $x^{(0)}$ étant donné

$$x_i^{(n+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{k=1}^{i-1} a_{ik} x_k^{(n+1)} - \sum_{k=i+1}^{N} a_{ik} x_k^{(n)} \right) \quad i = 1, 2, \cdots, N$$

Il s'écrit aussi

$$x^{(n+1)} = (D+L)^{-1} \left(b - U x^{(n)} \right)$$

(on aurait aussi pu envisager

$$x^{(n+1)} = (D+U)^{-1} \left(b - L x^{(n)} \right)$$

à condition de faire varier l'indice i de N à 1).

L'algorithme de Gauss-Seidel ne nécessite qu'un vecteur de stockage, $x^{(n)}$ étant remplacé par $x^{(n+1)}$ au cours de l'itération. Il est en général plus rapide que l'algorithme de Jacobi, donc préférable.

6.2.3. *Convergence de la méthode de Jacobi.*

Théorème 16. *Si A est à diagonale strictement dominante, alors l'algorithme de Jacobi converge.*

Démonstration : soit $\overline{x}$ la solution du système linéaire, on a

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

donc

$$\begin{array}{rcl} \max\limits_i \left| x_i^{(n+1)} - \overline{x_i} \right| & \leq & \dfrac{1}{a_{ii}} \displaystyle\sum_{k\neq i} |a_{ik}| \max_k \left| x_k^{(n)} - \overline{x_k} \right| \\ & \leq & K \max\limits_k \left| x_k^{(n)} - \overline{x_k} \right| \text{ avec } 0 \leq K < 1 \end{array}$$

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

$$\left\|x^{(n+1)} - \overline{x}\right\|_\infty \leq K \left\|x^{(n)} - \overline{x}\right\|_\infty$$

ce qui prouve la convergence.□

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

6.3. **Méthodes de gradient.** Si A est une matrice symétrique définie positive, il existe une méthode simple à programmer et très efficace : la méthode du gradient conjugué dont nous proposons la description en exercice.

6.3.1. *Méthode du gradient à pas optimal.* A est une matrice symétrique définie positive $N \times N$. On considère $\overline{x}$ l'unique solution du système linéaire $Ax = b$

1) Montrer que résoudre $Ax = b$ revient à trouver le minimum de

$$J(x) = \frac{1}{2} \langle Ax, x\rangle - \langle b, x\rangle$$

($\langle ., .\rangle$ désigne le produit scalaire usuel sur R^N)

2) On souhaite construire une suite de vecteurs $x^{(k)}$ convergente vers $\overline{x}$. Les notations suivantes auront cours dans la suite du problème :

$$\begin{aligned} r^{(k)} = b - Ax^{(k)} &\quad : \quad \text{"résidu" à l'étape } k \\ e^{(k)} = x^{(k)} - \overline{x} &\quad : \quad \text{erreur à l'étape } k \\ E(x) &\quad = \quad \langle A(x - \overline{x}), x - \overline{x}\rangle \end{aligned}$$

vérifier que

$$E(x) = -\langle r, e\rangle = \left\langle r, A^{-1}r\right\rangle$$

puis que

$$\nabla J(x) = -r$$

3) Posons

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$

$p^{(k)}$ est appelé direction de descente et $\alpha^{(k)}$ coefficient de descente. Justifier cette terminologie.

a) Montrer que le coefficient optimal de descente pour la direction $p^{(k)}$ est donné par

$$\alpha^{(k)} = \frac{\left\langle r^{(k)}, p^{(k)}\right\rangle}{\left\langle Ap^{(k)}, p^{(k)}\right\rangle}$$

puis que $\left\langle r^{(k)}, p^{(k+1)}\right\rangle = 0$

b) Montrer que

$$E(x^{(k+1)}) = E(x^{(k)})(1 - \gamma^{(k)})$$

où $\gamma^{(k)}$ est donné par

$$\gamma^{(k)} = \frac{\left\langle r^{(k)}, p^{(k)}\right\rangle^2}{\left\langle Ap^{(k)}, p^{(k)}\right\rangle \left\langle A^{-1}r^{(k)}, r^{(k)}\right\rangle}$$

c) On choisit comme direction de descente $p^{(k)} = r^{(k)}$ (méthode du gradient à pas optimal). Montrer que la suite $x^{(k)}$ converge vers $\overline{x}$.

6.3.2. *Méthode du gradient conjugué.* 4) a) On pose cette fois-ci $p^{(k)} = r^{(k)} + \beta^{(k)} p^{(k-1)}$. Calculer $\beta^{(k)}$ afin que $\gamma^{(k)}$ soit le plus grand possible (montrer que $\beta^{(k)} = -\frac{\langle Ap^{(k)}, r^{(k)} \rangle}{\langle Ap^{(k)}, p^{(k-1)} \rangle}$).

b) Déduire de ce qui précède l'algorithme du gradient conjugué. Décrire sa mise en oeuvre.

□□

7. Exercices et travaux pratiques

7.1. **TD TP n°1.**

7.1.1. *Exercices.* 1) Soient $f \in C^{n+1}([a,b])$, $a \leq x_0 < x_1 \cdots < x_n \leq b$, P_n le polynôme d'interpolation de f aux points (x_i). On se propose de montrer

$$f(x) - P_n(x) = \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi) \text{ avec } a \leq \xi \leq b$$

a) Soit $g : [a,b] \to R$ tel que g' existe. Montrer que si g a $(n+2)$ zéros distincts alors g' a au moins $(n+1)$ zéros distincts.

b) En considérant $W(t) = f(t) - P_n(t) - (t-x_0)\cdots(t-x_n)K(x)$ où K est tel que $W(x) = 0$ démontrer le résultat cherché.

2) On a donc

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

On cherche à majorer $|\prod_{i=0}^{n}(x-x_i)|$ dans le cas d'une subdivision régulière de pas h. On pose $\theta(x) = \prod_{i=0}^{n}(x-x_i)$, $h = 1$ et $x_0 = 0$ (pour simplifier).

a) Montrer que $\theta(x+1) = \theta(x)\frac{x+1}{x-n}$

b) En déduire que le max de $|\theta(x)|$est réalisé pour $x_0 \leq x \leq x_1$

c) Montrer que $\max_{x_0 \leq x \leq x_1} |\theta(x)| \leq \frac{n! h^{n+1}}{4}$

d) En déduire que

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{4(n+1)} h^{n+1}$$

3) On souhaite écrire une table de valeurs de $f(x) = \int_0^x e^{-t^2/2} dt$ pour une subdivision de pas h de l'intervalle $[0,1]$. Comment doit-on choisir h pour que l'interpolation de Lagrange à 3 points donne une approximation de f à 10^{-6} près ?

4) Soit arccos la détermination de la fonction inverse de cos définie par $\theta = \arccos x \Leftrightarrow x \in [0,\pi]$ et $x = \cos\theta$

a) On pose $Q_n(x) = \cos(n \arccos x)$. Montrer que les fonctions Q_n sont orthogonales sur l'intervalle $[-1,1]$ relativement au poids $w(x) = \sqrt{1-x^2}$

b) Montrer que $\langle Q_n, Q_n \rangle = \frac{\pi}{2}$ si $n \geq 1$ et que $\langle Q_0, Q_0 \rangle = \pi$

c) Montrer que Q_n est un polynôme de degré n vérifiant $Q_{n+1}(x) = 2xQ_n(x) - Q_{n-1}(x)$ (polynômes de Tchebychev)

5) Soit Q_n le $n^{\text{ième}}$ polynôme de Tchebychev

a) Montrer que Q_n a des zéros simples aux n points

$$x_k = \cos\frac{(2k-1)\pi}{2n} \quad k = 1 \cdots n$$

b) Montrer que Q_n atteint ses extrema sur l'intervalle $[-1,1]$ aux $n+1$ points $y_k = \cos\frac{k\pi}{n}$ $k = 0 \cdots n$ pour lesquels il prend alternativement les valeurs 1 et -1.

c) On considère $\overline{Q_n} = \frac{1}{2^{n-1}} Q_n$ (le coefficient de plus haut degré de $\overline{Q_n}$ est 1). Montrer que pour tout polynôme P de degré n, de coefficient de plus haut degré

égal à 1, on a

$$\frac{1}{2^{n-1}} = \max_{-1\le x\le 1} \left|\overline{Q_n}(x)\right| \le \max_{-1\le x\le 1} |P(x)|$$

d) On rappelle que

$$|f(x) - P(x)| \le \frac{M_{n+1}}{(n+1)!} \left|\prod_{i=0}^{n}(x - x_i)\right|$$

où P désigne le polynôme d'interpolation de f relativement à la subdivision (x_i). Comment choisir les (x_i) pour que l'erreur d'interpolation soit la plus petite possible ?

7.1.2. *Travaux pratiques.* Le but de ce TP est de mettre en évidence certains phénomènes liés à l'interpolation des fonctions. Pour les calculs, on s'aidera du logiciel Maple. Le compte-rendu (un par binôme) est à remettre à la fin de la seconde séance de 1h30. La qualité de la présentation et la précision de la rédaction (plus que la longueur du compte-rendu) auront un poids important dans la notation.

Considérons sur l'intervalle $[-1, 1]$ les fonctions suivantes :

$$\begin{aligned} f_1 &: x \to \sin(2\pi x) \\ f_2 &: x \to \frac{1}{\frac{1}{10} + x^2} \\ f_3 &: x \to \begin{cases} -1 \text{ si } x < 0 \\ +1 \text{ si } x \ge 0 \end{cases} \end{aligned}$$

Un entier $N > 0$ étant donné, on considère également la subdivision régulière

$$x_i = -1 + \frac{2i}{N} \quad 0 \le i \le N$$

1) a) Calculer, avec une précision de 5 chiffres significatifs, le polynôme d'interpolation P de f_1 en utilisant la matrice de Vandermonde et $N = 20$. Que constate-t-on ?

On pourra s'aider des commandes Maple suivantes, en les commentant.

```
> ? intro
> restart :
> with(linalg) :
> ? linalg
> ? Digits
> Digits :=5 ;
> N :=20 ;
> A :=matrix(N+1,N+1,(i,j)- > x[i-1]^(j-1)) ;
> x :=array(0..N,[seq(-1.+2.*i/N,i=0..N)]) ;
> A :=map(eval,A) ;
> f :=x- > sin(2*Pi*x) ;
> y :=[seq(evalf(f(x[i-1])),i=1..N+1)] ;
> c :=evalm(inverse(A)&*y) ;
> P :=sum(c[i]*t^(i-1),i=1..N+1) ;
> ? plot
> plot({P,f(t)},t=-1..1,-2..2) ;
```

b) Refaire le même calcul avec plus de précision. Conclusion ?

2) En utilisant la commande `interp` de Maple, calculer le polynôme d'interpolation de f_1, f_2 et f_3 pour $N = 5$, 10 et 20. Conclusion ?

Indication :

```
> restart ;N :=10 ;
>? seq
> x :=[seq(-1.+2.*i/N,i=0..N)] ;
> f :=x- > sin(2*evalf(Pi)*x) ;
> y :=[seq(f(-1.+2.*i/N),i=0..N)] ;
>? interp
> P :=interp(x,y,t) ;
> plot({P,f(t)},t=-1..1) ;
```

3) Refaire les calculs précédents en utilisant la subdivision de Tchebychev

$$x_i = \cos(i\pi/N) \quad 0 \leq i \leq N$$

4) Calculer les splines minimales d'interpolation de f_1, f_2 et f_3 pour une subdivision régulière et pour $N = 5$, 10 et 20. On pourra s'inspirer, en les commentant, des instructions suivantes :

```
> restart ;
> N :=10 ;
>?spline
> x :=[seq(-1.+2.*i/N,i=0..N)] ;
> f :=x- > 1./(.1+x^2) ;
> y :=[seq(f(-1.+2.*i/N),i=0..N)] ;
> readlib(spline) ;
> P :=spline(x,y,t) ;
> plot({P,f(t)},t=-1..1) ;
```

Vérifier, pour $N = 5$ que la solution donnée par Maple est correcte (calculer les dérivées secondes aux points d'interpolation et comparer avec les formules du cours).

□□

7.2. **TD TP n°2.**

7.2.1. *Exercices.* 1) Soient $f \in C^{n+1}([a,b])$, $a \leq x_0 < x_1 \cdots < x_n \leq b$, P_n le polynôme d'interpolation de f aux points (x_i). On se propose de montrer

$$f(x) - P_n(x) = \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi) \text{ avec } a \leq \xi \leq b$$

a) Soit $g : [a,b] \to R$ tel que g' existe. Montrer que si g a $(n+2)$ zéros distincts alors g' a au moins $(n+1)$ zéros distincts.

b) En considérant $W(t) = f(t) - P_n(t) - (t-x_0)\cdots(t-x_n)K(x)$ où K est tel que $W(x) = 0$ démontrer le résultat cherché.

2) On a donc

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

On cherche à majorer $|\prod_{i=0}^{n}(x-x_i)|$ dans le cas d'une subdivision régulière de pas h. On pose $\theta(x) = \prod_{i=0}^{n}(x-x_i)$, $h = 1$ et $x_0 = 0$ (pour simplifier).

a) Montrer que $\theta(x+1) = \theta(x)\frac{x+1}{x-n}$

b) En déduire que le max de $|\theta(x)|$est réalisé pour $x_0 \leq x \leq x_1$

c) Montrer que $\max_{x_0 \leq x \leq x_1} |\theta(x)| \leq \frac{n! h^{n+1}}{4}$

d) En déduire que

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{4(n+1)} h^{n+1}$$

3) On souhaite écrire une table de valeurs de $f(x) = \int_0^x e^{-t^2/2} dt$ pour une subdivision de pas h de l'intervalle $[0, 1]$. Comment doit-on choisir h pour que l'interpolation de Lagrange à 3 points donne une approximation de f à 10^{-6} près ?

4) Soit arccos la détermination de la fonction inverse de cos définie par $\theta = \arccos x \Leftrightarrow x \in [0, \pi]$ et $x = \cos\theta$

a) On pose $Q_n(x) = \cos(n \arccos x)$. Montrer que les fonctions Q_n sont orthogonales sur l'intervalle $[-1, 1]$ relativement au poids $w(x) = \sqrt{1 - x^2}$

b) Montrer que $\langle Q_n, Q_n \rangle = \frac{\pi}{2}$ si $n \geq 1$ et que $\langle Q_0, Q_0 \rangle = \pi$

c) Montrer que Q_n est un polynôme de degré n vérifiant $Q_{n+1}(x) = 2xQ_n(x) - Q_{n-1}(x)$ (polynômes de Tchebychev)

5) Soit Q_n le $n^{\text{ième}}$ polynôme de Tchebychev

a) Montrer que Q_n a des zéros simples aux n points

$$x_k = \cos\frac{(2k-1)\pi}{2n} \quad k = 1 \cdots n$$

b) Montrer que Q_n atteint ses extrema sur l'intervalle $[-1, 1]$ aux $n + 1$ points $y_k = \cos\frac{k\pi}{n}$ $k = 0 \cdots n$ pour lesquels il prend alternativement les valeurs 1 et -1.

c) On considère $\overline{Q_n} = \frac{1}{2^{n-1}} Q_n$ (le coefficient de plus haut degré de $\overline{Q_n}$ est 1). Montrer que pour tout polynôme P de degré n, de coefficient de plus haut degré égal à 1, on a

$$\frac{1}{2^{n-1}} = \max_{-1 \leq x \leq 1} \left|\overline{Q_n}(x)\right| \leq \max_{-1 \leq x \leq 1} |P(x)|$$

d) On rappelle que

$$|f(x) - P(x)| \leq \frac{M_{n+1}}{(n+1)!} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

où P désigne le polynôme d'interpolation de f relativement à la subdivision (x_i). Comment choisir les (x_i) pour que l'erreur d'interpolation soit la plus petite possible ?

7.2.2. *Travaux pratiques.* Le but de ce TP est de mettre en évidence certains phénomènes liés à l'interpolation des fonctions. Pour les calculs, on s'aidera du logiciel Maple. Le compte-rendu (un par binôme) est à remettre à la fin de la seconde séance de 1h30. La qualité de la présentation et la précision de la rédaction (plus que la longueur du compte-rendu) auront un poids important dans la notation.

Considérons sur l'intervalle $[-1, 1]$ les fonctions suivantes :

$$\begin{aligned} f_1 &: x \to \sin(2\pi x) \\ f_2 &: x \to \frac{1}{\frac{1}{10} + x^2} \\ f_3 &: x \to \begin{cases} -1 \text{ si } x < 0 \\ +1 \text{ si } x \geq 0 \end{cases} \end{aligned}$$

Un entier $N > 0$ étant donné, on considère également la subdivision régulière

$$x_i = -1 + \frac{2i}{N} \quad 0 \leq i \leq N$$

1) a) Calculer, avec une précision de 5 chiffres significatifs, le polynôme d'interpolation P de f_1 en utilisant la matrice de Vandermonde et $N = 20$. Que constate-t-on ?

On pourra s'aider des commandes Maple suivantes, en les commentant.

```
> ? intro
> restart :
```

```
> with(linalg) :
>? linalg
>? Digits
> Digits :=5 ;
> N :=20 ;
> A :=matrix(N+1,N+1,(i,j)- > x[i-1]^(j-1)) ;
> x :=array(0..N,[seq(-1.+2.*i/N,i=0..N)]) ;
> A :=map(eval,A) ;
> f :=x- > sin(2*Pi*x) ;
> y :=[seq(evalf(f(x[i-1])),i=1..N+1)] ;
> c :=evalm(inverse(A)&*y) ;
> P :=sum(c[i]*t^(i-1),i=1..N+1) ;
>? plot
> plot({P,f(t)},t=-1..1,-2..2) ;
```

b) Refaire le même calcul avec plus de précision. Conclusion ?

2) En utilisant la commande `interp` de Maple, calculer le polynôme d'interpolation de f_1, f_2 et f_3 pour $N = 5$, 10 et 20. Conclusion ?

Indication :

```
> restart ;N :=10 ;
>? seq
> x :=[seq(-1.+2.*i/N,i=0..N)] ;
> f :=x- > sin(2*evalf(Pi)*x) ;
> y :=[seq(f(-1.+2.*i/N),i=0..N)] ;
>? interp
> P :=interp(x,y,t) ;
> plot({P,f(t)},t=-1..1) ;
```

3) Refaire les calculs précédents en utilisant la subdivision de Tchebychev

$$x_i = \cos(i\pi/N) \quad 0 \leq i \leq N$$

4) Calculer les splines minimales d'interpolation de f_1, f_2 et f_3 pour une subdivision régulière et pour $N = 5$, 10 et 20. On pourra s'inspirer, en les commentant, des instructions suivantes :

```
> restart ;
> N :=10 ;
>?spline
> x :=[seq(-1.+2.*i/N,i=0..N)] ;
> f :=x- > 1./(.1+x^2) ;
> y :=[seq(f(-1.+2.*i/N),i=0..N)] ;
> readlib(spline) ;
> P :=spline(x,y,t) ;
> plot({P,f(t)},t=-1..1) ;
```

Vérifier, pour $N = 5$ que la solution donnée par Maple est correcte (calculer les dérivées secondes aux points d'interpolation et comparer avec les formules du cours).

□□

7.3. **TD TP n°3.**

7.3.1. *Exercices.*

Résolution itérative des équations non linéaires. 1) Montrer que l'équation $f(x) = x^3 - 2$ possède une solution unique α et qu'on peut obtenir celle-ci en utilisant la méthode de Newton à partir de $x_0 = 1$. Donner une minoration du nombre d'itérations à effectuer pour obtenir une précision $\varepsilon = 10^{-10}$

2) Soit la fonction $f(x) = \cos x - xe^x$ avec $0 \leq x \leq \frac{\pi}{2}$

a) Montrer que cette fonction a une et une seule racine l dans $[0, \frac{\pi}{2}]$

b) Expliciter la méthode de Newton. Donner une valeur de x_0 assurant la convergence vers l.

c) Soit la méthode suivante

$$\begin{aligned} x_0 &\in [a,b] \\ x_{n+1} &= \cos x_n e^{-x_n} = g(x_n) \end{aligned}$$

Montrer que l'on peut choisir un intervalle $[a,b] \subset [0, \frac{\pi}{2}]$ telle que la méthode converge vers l (montrer que $|x_n - l| \leq L^n |x_0 - l|$. Que vaut L ?)

d) Montrer que l'on peut prendre $[a,b] = [0, \frac{\pi}{2}]$

e) Calculer un nombre d'itérations suffisant pour obtenir une précision de 10^{-6} si $[a,b] = [0.45, \frac{\pi}{2}]$

Résolution des sytèmes linéaires. 1) On considère le problème : trouver $u \in C^2[0,1]$ solution de

$$\begin{aligned} -u''(x) + c^2(x)u(x) &= f(x) \\ u(0) &= u(1) = 0 \end{aligned} \tag{9}$$

c et f sont des fonctions continues sur $[0,1]$. $N \in N^*$ étant donné, on pose $h = \frac{1}{N}$ et $x_i = ih$ pour $0 \leq i \leq N$. Montrer que

$$u''(x_i) = \frac{u(x_{i+1}) + u(x_{i-1}) - 2u(x_i)}{h^2} + o(h)$$

2) On remplace (9) par un problème plus simple : l'inconnue devient un vecteur $(u_i)_{0 \leq i \leq N}$ (où a priori, $u_i \approx u(x_i)$). Proposer un système linéaire que pourrait vérifier les $(u_i)_{0 \leq i \leq N}$.

3) Une matrice A est dite tridiagonale si elle est de la forme

$$A = \begin{pmatrix} d_1 & u_1 & & 0 \\ l_1 & \ddots & \ddots & \\ & \ddots & \ddots & u_{n-1} \\ 0 & & l_{n-1} & d_n \end{pmatrix}$$

Proposer un algorithme (adapté de l'algorithme général) pour calculer la décomposition LU de A (on supposera que les pivots ne sont jamais nuls).

4) On pose $f(x) = \sin(\pi x)$, et $c(x) = 0$. Calculer la solution de (9) dans ce cas.

5) Calculer une base orthonormale de la matrice

$$B = \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix}$$

(on pourra essayer $e_i = \sin(\mu i)$ où μ est un nombre bien choisi).

6) Résoudre le système linéaire de la question 2 pour $f(x) = \sin(\pi x)$, et $c(x) = 0$.

7) On pose $i = E(x/h)$ à x fixé ($E(x)$ = partie entière de x). Soit $(u_i)_{0\leq i\leq N}$ trouvé à la question 6 et $u(x)$ la solution de la question 4. Montrer que $u_i \to u(x)$ quand h tend vers 0.

7.3.2. *TP : résolution itérative des équations non linéaires.* Le but de ce TP est de déterminer diverses méthodes pour résoudre :

$f(x) = 0$ avec $f(x) = \tan(x) - 1 + x$ sur l'intervalle $[0, 1]$

Le compte-rendu ne doit pas dépasser une copie double, courbes comprises.

1) Vérifier d'abord graphiquement que $f(x) = 0$ admet une unique racine $\overline{x}$ sur $]0, 1[$. Donnez un encadrement de $\overline{x}$ à 10^{-1} près.

Indications :

```
f :=tan(x)-1+x ;plot(f,x=0..1) ;
```

2) **Méthode du point fixe**

On pose :

$$g_\alpha(x) = \frac{f(x) + \alpha x}{\alpha}$$

et on considère la méthode :

$$\begin{aligned} x_0 &\in [0, 1] \\ x_{n+1} &= g_\alpha(x_n) \end{aligned}$$

Tracer $g_\alpha(x)$ sur $[0, 1]$ pour $\alpha = -3$. Montrer que $x_n \to \overline{x}$

Indications :

```
g :=(f+alpha*x)/alpha ;
alpha :=-3 ;
plot(g,x=0..1) ;
?diff ;
```

3) Dans cette question, $x_0 = 0$

On désire étudier l'ordre de convergence de la méthode, c'est dire déterminer γ tel que :

$$\frac{|x_{n+1} - \overline{x}|}{|x_n - \overline{x}|^\gamma} \to \lambda \text{ avec } 0 < \lambda < +\infty$$

Quelle méthode proposez-vous ? Quelles sont les difficultés de mise en oeuvre, comment s'en affranchir grâce à Maple ? Comparer les valeurs théoriques de λ et γ avec les valeurs numériques.

Indications :

```
xn :=0 ;N :=20 ;Digits :=100 ;
graf :=[] ;
xb :=fsolve(f=0,x=0..1) ;
for i to N do
xnp1 :=evalf(subs(x=xn,g)) ;
graf :=[op(graf),[ln(abs(xn-xb)),ln(abs(xnp1-xb))]] :
xn :=xnp1 :od :
plot(graf) ;
(graf[N][2]-graf[N-1][2])/(graf[N][1]-graf[N-1][1]) ;
```

4) **Méthode de Newton**

Réaliser la même étude pour la méthode de Newton :

$$\begin{aligned} x_0 &= 0 \\ x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \end{aligned}$$

5) **Méthode de la sécante**
Même étude pour la méthode de la sécante :

$$\begin{aligned} x_0 &= 0,\ x_1 = 1 \\ x_{n+1} &= x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \end{aligned}$$

Vérifier que $\gamma = \frac{1+\sqrt{5}}{2}$

□□

7.4. **TD TP n°4.**

7.4.1. *Exercices : Résolution itérative des équations linéaires.* 1)
a) Pour une matrice quelconque A montrer que $\rho(A) \leq \|A\|$ pour toute norme matricielle. Donner l'expression de $\|A\|_\infty$
b) On suppose A symétrique définie positive. Montrer que $\rho(A) = \|A\|_2$

2) Soit

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

a) Pour quelles valeurs de a A est-elle définie positive ?
b) Écrire la matrice J de l'itération de Jacobi
c) Pour quelles valeurs de a la méthode de Jacobi converge-t-elle ?
d) Écrire la matrice G de l'itération de Gauss-Seidel
e) Calculer $\rho(G)$. Pour quelles valeurs de a cette méthode converge-t-elle plus vite que celle de Jacobi ?

3) Soit A une matrice décomposée en $A = D - E - F$

$$A = \begin{pmatrix} \ddots & & -E \\ & D & \\ -F & & \ddots \end{pmatrix}$$

pour résoudre $Ax = b$ on propose la méthode

$$\left(\frac{D}{\omega} - E\right) x^{(k+1)} = \left(\frac{1-\omega}{\omega} D + F\right) x^{(k)} + b \quad \omega \in R^*$$

a) Vérifier que si la méthode converge, elle converge vers une solution de $Ax = b$
b) Donner la matrice d'itération L_ω de cette méthode
c) Calculer $\det(L_\omega)$
d) En déduire que $\rho(L_\omega) \geq |1 - \omega|$. Conclusion ?
□□

7.4.2. *TP : quelques algorithmes d'algèbre linéaire.* Le but de ce TP est de tester quelques méthodes numériques de calcul sur les systèmes linéaires.
Le compte-rendu ne doit pas dépasser une copie double, courbes comprises.

Dans la suite, on considère la matrice de taille $N - 1$:

$$B = \begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix}$$

on rappelle que les vecteurs propres $e(k)$ et les valeurs propres $\lambda(k)$ sont donnés par

$$\begin{aligned} e(k)_i &= \sin(\frac{ik\pi}{N}) \quad 1 \leq i \leq N-1 \quad 1 \leq k \leq N-1 \\ \lambda(k) &= 4\sin^2(\frac{k\pi}{2N}) \end{aligned}$$

1) En vous aidant des instructions suivantes, vérifiez l'intérêt de la méthode LU pour les matrices tridiagonales. Pour cela, comparer, en fonction de N, le temps de calcul de résolution d'un système linéaire plein ou tridiagonal.

```
restart :with(linalg) :N :=40 ;
A :=evalm(1e-6*randmatrix(N-1,N-1)) :
for i to N-1 do A[i,i] :=2. :od :
for i to N-2 do A[i,i+1] :=-1. :A[i+1,i] :=-1. :od :
# utiliser la ligne suivante à la place des 3 précédentes :
#A :=band([-1.,2.,-1.],N-1) ;
s :=randvector(N-1) ;
# calcul du temps. Ne pas oublier de faire restart entre chaque calcul!!!
tt :=time() :linsolve(A,s) :time()-tt ;
```

2) Programmer dans Maple les méthodes de Jacobi et de Gauss-Seidel. Comparer sur quelques exemples leurs vitesses de convergence (compter le nombre d'itérations nécessaires pour atteindre une précision donnée).

3) Vérifier que l'algorithme suivant permet de calculer la valeur propre de plus grand module de B. Justifier votre réponse. Comment calculer la valeur propre de plus petit module ?

```
> restart :with(linalg) :N :=20 ;
> A :=band([-1.,2.,-1.],N-1) ;
> x :=evalf(randvector(N-1)) ;
> iter :=30 ;
> for i to iter do
> y :=evalm(A&*x) :x :=evalm(1./norm(y,2)*y) :od :
> seq(y[i]/x[i],i=1..N-1) ;
> k :=N-1 :lambda :=evalf(4.*sin(k*Pi/2/N)^2) ;
> s :=vector(N-1,[seq(evalf(sin(i*k*Pi/N)),i=1..N-1)]) ;
> s :=evalm(1./norm(s,2)*s) ;
> op(x) ;
```

4) Expliquer à quoi servent les lignes suivantes.

```
> k :=N/2-2 ;
> lambda :=evalf(4.*sin(k*Pi/2/N)^2)+1e-5 ;
> x :=evalf(randvector(N-1)) ;iter :=30 ;Id :=array(1..N-1,1..N-1,identity) ;
> for i to iter do
> y :=linsolve(evalm(A-lambda*Id),x) :x :=evalm(1./norm(y,2)*y) :od :
> seq(y[i]/x[i],i=1..N-1) ;
> s :=vector(N-1,[seq(evalf(sin(i*k*Pi/N)),i=1..N-1)]) ;
> s :=evalm(1./norm(s,2)*s) ;
> op(x) ;
> y :=evalm(A&*x) :
> seq(y[i]/x[i],i=1..N-1) ;
```

□□

8. Examen d'analyse numérique n°1

8.1. **Enoncé.**

Exercice 1. On cherche une méthode d'intégration numérique de la forme

$$\int_0^1 f(t)dt \simeq \alpha f(0) + \beta f'(\gamma) \tag{10}$$

où γ est un point de l'intervalle $]0,1[$

1) On suppose que γ est donné. Déterminer α et β pour que la méthode soit exacte pour des polynômes de degré ≤ 1.

2) Comment choisir γ pour que la méthode soit exacte pour des polynômes de degré ≤ 2 ?

3) Que devient la formule (10) si l'intervalle $[0,1]$ est remplacé par un intervalle $[a,b]$ quelconque ? (utiliser le changement de variable $t = (b-a)t' + a$, $t' \in [0,1]$).

4) Décrire la méthode composite associée à (10) pour calculer $\int_A^B f(t)dt$ (on se limitera au cas d'une subdivision régulière de l'intervalle $[A,B]$ de pas $h = \frac{B-A}{N}$).

Exercice 2. Soient les polynômes suivants

$$\begin{aligned} G_0(x) &= (2x+1)(x-1)^2 \\ G_1(x) &= x(x-1)^2 \\ D_0(x) &= x^2(3-2x) \\ D_1(x) &= x^2(x-1) \end{aligned}$$

on vérifie facilement qu'ils constituent une base de l'ensemble des polynômes de degré ≤ 3 et que

$$\begin{array}{llll} G_0(0)=1 & G_0(1)=0 & G_0'(0)=0 & G_0'(1)=0 \\ G_1(0)=0 & G_1(1)=0 & G_1'(0)=1 & G_1'(1)=0 \\ D_0(0)=0 & D_0(1)=1 & D_0'(0)=0 & D_0'(1)=0 \\ D_1(0)=0 & D_1(1)=0 & D_1'(0)=0 & D_1'(1)=1 \end{array}$$

(**la démonstration de ces résultats n'est pas demandée**).

1) Soit f une fonction C^1 sur $[0,1]$. Déterminer l'unique polynôme P de degré ≤ 3 tel que

$$\begin{aligned} f(0) &= P(0) \quad f'(0) = P'(0) \\ f(1) &= P(1) \quad f'(1) = P'(1) \end{aligned}$$

2) Soit f une fonction C^1 sur un intervalle $[a,b]$ quelconque. Déterminer l'unique polynôme P de degré ≤ 3 tel que

$$\begin{aligned} f(a) &= P(a) \quad f'(a) = P'(a) \\ f(b) &= P(b) \quad f'(b) = P'(b) \end{aligned}$$

3) Soit f une fonction C^1 sur un intervalle $[A,B]$ quelconque, et soit $A = x_0 < \cdots < x_N = B$ une subdivision de $[A,B]$. Déterminer l'unique fonction F polynômiale de degré ≤ 3 par morceaux, de classe C^1sur $[A,B]$ telle que

$$f(x_i) = F(x_i) \quad f'(x_i) = F'(x_i) \quad 0 \leq i \leq N$$

4) A-t-on convergence de F vers f quand le pas de la subdivision tend vers 0 ?

Exercice 3. Soit Q le polynôme

$$Q(t) = t^2 - St^2 + P$$

on suppose que Q admet 2 racines réelles $x_0 \neq y_0$.

1) Montrer que trouver x_0 et y_0 revient à résoudre

$$\begin{aligned} xy - P &= 0 \\ (x+y) - S &= 0 \end{aligned}$$

La suite de l'exercice a pour but de décrire la méthode de Newton permettant de résoudre ce système non linéaire de deux équations à deux inconnues

2) Comment s'écrit la méthode de Newton pour résoudre $F(x,y)=0$ quand F est une application de $R^2 \to R^2$?

3) Désormais F est l'application de $R^2 \to R^2$ définie par

$$F(x,y) = \begin{bmatrix} xy - P \\ (x+y) - S \end{bmatrix}$$

calculer $J(x,y)$ la matrice jacobienne de $F(x,y)$.

4) Vérifier que la méthode de Newton pour résoudre $F(x,y)=(0,0)$ s'écrit ici

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} - \begin{bmatrix} \frac{Q(x_n)}{x_n - y_n} \\ \frac{Q(y_n)}{y_n - x_n} \end{bmatrix} \tag{11}$$

5) Décrire la mise en oeuvre de l'algorithme (11) pour trouver les racines d'un polynôme de degré 2. Pouvez-vous généraliser à des polynômes de degré > 2 ? Quels sont les avantages de cette méthode comparée à d'autres méthodes ? Que se passe-t-il si Q a des racines multiples ou complexes ?

Exercice 4. 1) Soit y une fonction de classe C^∞ sur R. On souhaite approcher y' par une formule de la forme

$$y'(t) \simeq \alpha y(t) + \beta y(t-h) + \gamma y(t-2h) \tag{12}$$

Grâce à un développement de Taylor au voisinage de $h=0$, déterminer (α,β,γ) de façon que la méthode soit la plus "précise" possible.

2) En utilisant la formule (12), proposer une méthode pour résoudre l'équation différentielle

$$\begin{aligned} y'(t) &= f(y(t)) \quad t \geq 0 \\ y(0) &= y_0 \end{aligned}$$

par une formule de la forme

$$H(y_{n+1}, y_n, y_{n-1}, h) = 0$$

3) Comment mettre en oeuvre une telle méthode ? Comment calculer y_1 ?

□□

8.2. **Corrigé.**

Exercice 1. 1) La formule étant exacte pour $x \to 1$ et $x \to x$, on trouve $\alpha = 1$ et $\beta = 1/2$

2) La formule étant aussi exacte pour $x \to x^2$, on trouve $\gamma = 1/3$

3) On a $\int_a^b f(t)dt = (b-a)\int_0^1 f((b-a)t'+a)dt'$, d'où

$$\int_a^b f(t)dt \simeq (b-a)f(a) + \frac{(b-a)^2}{2} f'(\frac{2}{3}a + \frac{1}{3}b)$$

4) $\int_A^B f(t)dt \simeq \sum_{i=0}^{N-1} \int_{A+ih}^{A+(i+1)h} f(t)dt = \sum_{i=0}^{N-1} hf(A+ih) + \frac{h^2}{2} f'(A+(i+\frac{1}{3})h)$

Exercice 2. 1) P est donné par $P = f(0)G_0 + f'(0)G_1 + f(1)D_0 + f'(1)D_1$

2) $g(t') = f((b-a)t'+a)$ est définie sur $[0,1]$. On pose $Q(t') = f(a)G_0(t') + (b-a)f'(a)G_1(t') + f(b)D_0(t') + (b-a)f'(b)D_1(t')$ et donc

$$\begin{aligned} P(t) &= Q(\frac{t-a}{b-a}) \\ &= f(a)G_0(\frac{t-a}{b-a}) + (b-a)f'(a)G_1(\frac{t-a}{b-a}) + f(b)D_0(\frac{t-a}{b-a}) + (b-a)f'(b)D_1(\frac{t-a}{b-a}) \end{aligned}$$

3) Sur l'intervalle $[x_{i-1}, x_i]$, F est donnée par la formule de la question 2) avec $a = x_{i-1}$ et $b = x_i$

4) Il y a convergence de F vers f quand le pas de la subdivision tend vers 0 car le degré d des polynômes d'interpolation est **fixé** ($d \leq 3$).

Exercice 3. 1) $Q(t) = (t-x_0)(t-y_0) = t^2 - (x_0+y_0)t^2 + x_0y_0$ donc

$$\begin{aligned} x_0y_0 - P &= 0 \\ (x_0+y_0) - S &= 0 \end{aligned}$$

2) Si $F(x,y) = \begin{pmatrix} F_1(x,y) \\ F_2(x,y) \end{pmatrix}$, alors $F'(x,y) = \begin{pmatrix} \frac{\partial F_1(x,y)}{\partial x} & \frac{\partial F_1(x,y)}{\partial y} \\ \frac{\partial F_2(x,y)}{\partial x} & \frac{\partial F_2(x,y)}{\partial y} \end{pmatrix}$ et la méthode de Newton s'écrit

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} - F'(x_n,y_n)^{-1} F(x_n,y_n)$$

3) $J(x,y) = \begin{pmatrix} y & x \\ 1 & 1 \end{pmatrix}$

4) $J(x,y)^{-1} = \frac{1}{y-x}\begin{pmatrix} 1 & -x \\ -1 & y \end{pmatrix}$, par conséquent

$$\begin{aligned} \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} &= \begin{pmatrix} x_n \\ y_n \end{pmatrix} - \frac{1}{y_n - x_n}\begin{pmatrix} 1 & -x_n \\ -1 & y_n \end{pmatrix}\begin{bmatrix} x_ny_n - P \\ x_n + y_n - S \end{bmatrix} \\ \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} &= \begin{pmatrix} x_n \\ y_n \end{pmatrix} - \begin{pmatrix} \frac{Q(x_n)}{x_n - y_n} \\ \frac{Q(y_n)}{y_n - x_n} \end{pmatrix} \end{aligned}$$

5) L'algorithme se généralise sans difficulté pour des polynômes de degré d plus grand. Ainsi, pour $d = 3$, on trouve

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} - \begin{pmatrix} \frac{Q(x_n)}{(x_n-y_n)(x_n-y_n)} \\ \frac{Q(y_n)}{(y_n-x_n)(y_n-z_n)} \\ \frac{Q(z_n)}{(z_n-x_n)(z_n-y_n)} \end{pmatrix}$$

L'algorithme n'est pas très intéressant pour $d = 2$. En revanche, pour $d \geq 3$, il est très utile : il converge en général de manière quadratique, indépendamment de l'initialisation, et permet d'obtenir toutes les racines en même temps. Pour évaluer $Q(t)$, il est conseillé d'utiliser l'algorithme de Hörner. Si Q a des racines multiples, la convergence de la méthode de Newton n'est plus quadratique ($J(x_0,y_0)$ n'est pas inversible). Si Q a des racines complexes, la méthode est encore utilisable, à condition de l'initialiser avec un vecteur complexe.

Exercice 4. 1) Un développement de Taylor donne

$$\begin{aligned} & \alpha y(t) + \beta y(t-h) + \gamma y(t-2h) \\ = \; & (\alpha+\beta+\gamma)y(t) + (-\beta-2\gamma)hy'(t) + (\frac{\beta}{2}+2\gamma)h^2y''(t) + O(h^3) \end{aligned}$$

il faut donc prendre

$$\begin{aligned} \alpha+\beta+\gamma &= 0 \\ -\beta-2\gamma &= \frac{1}{h} \\ \frac{\beta}{2}+2\gamma &= 0 \end{aligned}$$

soit

$$\alpha = \frac{3}{2h} \quad \beta = -\frac{2}{h} \quad \gamma = \frac{1}{2h}$$

2) D'après ce qui précède, $y'(t) = \frac{3y(t)-4y(t-h)+y(t-2h)}{2h} + O(h^2)$. Si $t_n = nh$ et si on cherche $y_n \simeq y(t_n)$, on peut choisir de résoudre

$$\frac{3y_{n+1} - 4y_n + y_{n-1}}{2h} = f(y_{n+1})$$

soit

$$H(y_{n+1,}y_n, y_{n-1}, h) = 3y_{n+1} - 2hf(y_{n+1}) - 4y_n + y_{n-1} = 0$$

3) Pour mettre en oeuvre cette méthode implicite à **deux pas**, il faut résoudre à chaque pas de temps l'équation d'inconnue y_{n+1} (on peut utiliser la méthode de Newton). De plus, pour amorcer l'algorithme, y_1 sera évalué à l'aide d'une formule explicite à **un pas** du type $y_1 = y_0 + h\Phi(y_0, h)$.

□

9. Examen d'analyse numérique n°2

9.1. **Enoncé.**

Exercice 1 : étude de la méthode des rectangles composite

. 1) Soit f une application de classe C^2 sur l'intervalle $[0, 1]$. On rappelle qu'il existe alors un $t_0 \in [0, 1]$ (dépendant de f) tel que

$$\int_0^1 f(t)dt = f(\frac{1}{2}) + \frac{1}{24}f''(t_0)$$

Déduire de cette propriété que si g est une application de classe C^2 sur l'intervalle $[a, b]$ alors on peut trouver $x_0 \in [a, b]$ tel que

$$\int_a^b g(x)dx = (b-a)g(\frac{a+b}{2}) + \frac{(b-a)^3}{24}g''(x_0)$$

(faire le changement de variable $x = t(b-a) + a$)

2) On se donne maintenant une application f de classe C^2 sur un intervalle $[a, b]$. On pose $I(f) = \int_a^b f(x)dx$. Soit $J_N(f)$ l'approximation de $I(f)$ par la méthode des rectangles composite à N points :

$$J_N(f) = \frac{b-a}{N}\sum_{i=1}^{N} f(\frac{i-1/2}{N}(b-a) + a)$$

montrer qu'il existe des points $x_i \in [(b-a)\frac{i-1}{N} + a, (b-a)\frac{i}{N} + a]$ $(1 \leq i \leq N)$ tels que

$$\int_a^b f(x)dx = J_N(f) + \frac{(b-a)^3}{24N^2} \cdot \frac{1}{N}\sum_{i=1}^{N} f''(x_i)$$

3) En déduire qu'il existe $c \in [a, b]$ tel que

$$\int_a^b f(x)dx = J_N(f) + \frac{(b-a)^3}{24N^2}f''(c)$$

4) Comment choisir N pour être sûr que $J_N(f)$ soit une approximation à 10^{-6} près de $I(f)$ si $f(x) = \exp(x)$ et $[a, b] = [0, 2]$?

5) Que se passe-t-il selon vous si l'on applique la même méthode d'intégration numérique à $f(x) = \ln(x)$ sur l'intervalle $[0, 2]$?

Exercice 2 : méthode de Jacobi pour une matrice 3x3
. 1) On considère la matrice

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

On se donne également un vecteur b de R^3. On note $\overline{x}$ l'unique solution du système $Ax = b$. Montrer que la méthode de Jacobi pour résoudre le système linéaire $Ax = b$ est ici convergente.

2) Soit x_n le $n^{\text{ième}}$ itéré de la méthode de Jacobi en partant de $x_0 = 0$. Déterminer la matrice d'itération M de la méthode de Jacobi (c'est la matrice qui vérifie $x_{n+1} - \overline{x} = M(x_n - \overline{x})$).

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

4) Montrer que $|\overline{x}| \leq \frac{1}{s} |b|$ où $|u|$ désigne la norme euclidienne usuelle du vecteur u dans R^3

5) Montrer que $|x_n - \overline{x}| \leq \frac{S^n}{s} |b|$

Exercice 3 : Méthode à deux pas pour résoudre une équation différentielle
. 1) On considère la méthode d'intégration numérique suivante sur l'intervalle $[0, 1]$:

$$\int_0^1 y(x)dx \simeq \alpha y(0) + \beta y(\frac{1}{2})$$

Déterminer α et β pour que cette méthode soit exacte pour des polynôme de degré le plus élevé possible.

2) Que devient cette méthode sur un intervalle $[a, b]$ quelconque ?

3) Soit T un réel > 0, f une application de $R^+ \times R \to R$ de classe C^1 et soit y la solution de l'équation différentielle

$$\begin{aligned} y'(t) &= f(t, y(t)) \text{ pour } t > 0 \\ y(0) &= y_0 \end{aligned}$$

soit h un réel > 0. Montrer que

$$y(2h) = y(0) + \int_0^{2h} f(t, y(t))dt$$

4) Déduire des questions précédentes une méthode numérique permettant de calculer une approximation y_n de $y(nh)$ de la forme

$$y_{n+2} = G(y_n, y_{n+1})$$

5) Comment proposez-vous de calculer y_1 ?
□