Résumé du cours d'Analyse Numérique

1) *Convergence*

- *Linéaire*: on gagne la même quantité de précision à chaque itéation.
- *Quadratique*: on gagne le double de précision à chaque itéation.
- *Cubique*: on gagne le triple de précision à chaque itéation.
- etc.

2) *Résolution d'une équation f(x)=0*

a) Estimer graphiquement la racine. Tracer le graphe de la fonction et zoomer sur les racines.

b) **Méthode de la dichotomie**: [suite d'intervalles encadrant la racine]

i) $[a_0,b_0]=[a,b]$

ii) $[a_n,b_n](connu) => w=\frac{(a_n+b_n)}{2}$

$\circ f(a_n).f(w)<0 \Rightarrow [a_{n+1},b_{n+1}]=[a_n,w]$

iii) $\circ f(a_n).f(w)>0 \Rightarrow [a_{n+1},b_{n+1}]=[w,b_n]$

$\circ f(a_n).f(w)=0 \Rightarrow w=racine$

iv) Avantages : - Convergence certaine vers la racine (si elle est unique sur l'intervalle), si f est continue.

- Pour une précision e, il faut faire n itération avec $n > \frac{\ln\frac{b-a}{e}}{\ln 2}$

(car l'intervalle après n itérations est de longueur $\frac{b-a}{2^n}$).

v) Inconvénients : - Convergence linéaire (lente).
- Nécessité du changement de signe sur l'intervalle.

c) **Méthode de la sécante**: [suite de valeurs tendant vers racine]

i) Fixer x_0 et x_1

ii) Calculer $x_{n+2}=x_{n+1}-f(x_{n+1})\frac{x_{n+1}-x_n}{f(x_{n+1})-f(x_n)}$ (n=0,1,2,...)

iii) Avantages: - Pour une fonction à un seul zéro peu d'hypothèses de départ.
- Convergenge rapide (pour racines simples, avec bon choix de x_0 et x_1)

iv) Inconvénients: - Lente convergence.
- Manque de précision pour une racine multiple.
- Risque d'une division par zéro.

d) **Méthode du point fixe/d'itération**:

i) Choisir x_1

ii) Calculer $x_{n+1}=g(x_n)$ (n=1,2,...)

iii) Choisir g(x) tel que x=g(x).

iv) Avantages: - Facile à mettre en oeuvre, g(x) qui converge rapidement n'est souvent pas trop difficile trouver.

v) Inconvénients: - Bien choisir g(x), tel que g(x)=x. Il faut parfois différents g(x) pour trouver différentes racines d'une même équation.
- Convergence si $|g'(x_0)|<1$ (vérifier graphiquement si graphe de g'(x)<pente y=x). (divergence si g(x) pas bonne)
- Convergence lente (si g(x) est valable).

e) **Méthode de Newton(-Raphson)/de la tangente**: (remplacer la fonction par sa tangente: approximation. Zéro de la tangente est proche de celui de la fonction si la tangente est très semblable à la fonction) [génère une suite de tangentes au graphe de F(x) et une suite de valeurs x_n]
   i) Choisir x_1 proche du zéro cherché.
   ii) Calculer: $x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}$ (n=1,2,..) [avec F(x)=0 équation à résoudre].
   Ou approximation: $x_{n+1} = x_n - h\frac{F(x_n)}{F(x_n+h)-F(x_n)}$ (n=1,2,... et h très petit).
   iii) Avantages: - Converge quadratiquement si racine simple.
   - Méthode valable pour trouver zéros complexes.
   iv) Inconvénients: - Converge linéairement si racine multiple.
   - Si racine multiple: F et F' très petites => convergence très lente.
   - Besoin de calculer la dérivée de la fonction.

f) Arrêter l'algorithme quand l'écart entre deux solution x_n et x_{n+1} est assez faible (de l'ordre de 10^{-k}, avec k que l'on veut).

3) ***Résolution de systèmes algébriques linéaires (problème aux CL, etc.) : méthodes directes***

$\sum_{j=1}^{n} a_{ij}x_j = b_i$ (i=1,2,...,n) ou Ax=b avec A: matrice carrée des a_{ij} (nxn), x et b: vecteurs colonnes x_j et b_j (nx1). "n" équations à "n" inconnues.

a) **Méthode du déterminent**:
$Ax = b \Rightarrow x = A^{-1}b = (\det A)^{-1}(A^*)^t b$ (Matlab: x=A\b)

b) Système à matrice triangulaire inférieure / supérieure
$x_k = \frac{b_k - \sum_{j=1}^{k-1} a_{kj}x_j}{a_{kk}}$ (si $a_{kk}\neq 0$, detA≠0) [inférieure: k=1,2,3,...n | supérieure: k=n,n-1,n-2,...,1]

c) Méthode de Gauss:
(! Si pivot s'annule: permuter la ligne de ce pivot avec une ligne d'un indice supérieur)
   i) Définir la matrice A=(a_{ij}) du système à résoudre, le vecteur b=[...] (1xn).
   ii) Pour k=1,...,n-1:
   $\begin{cases} m = \frac{A_{i,k}}{A_{k,k}} \\ b_i = b_i - m.b_k \end{cases}$ (i=k+1,...,n) , $A_{i,j} = A_{i,j} - m.A_{k,j}$ (j=k,...,n).
   iii) $x_n = \frac{b_n}{A_{n,n}}$, S=0, pour i=n-1,...,1 : $S = S + (A_{i,j}.x_j)$ (j=i+1,...,n), $x_i = \frac{(b_i - S)}{A_{i,i}}$

d) Méthode de Newton-Raphson:
   i) Evaluer graphiquement (tracer f et g) l'intersection des courbes (solution du système).
   ii) Choisir un couple (x_1,y_1) proche d'une solution du système.
   iii) Calculer $x_{n+1} = x_n + \Delta x_n$ (n=1,2,3,...)

Avec $x_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix}$, $J(x_n)\Delta x_n = -F(x_n)$ où J : Jacobienne $J(x_n) = \begin{pmatrix} \frac{\partial f}{\partial x}(x_n) & \frac{\partial f}{\partial y}(x_n) \\ \frac{\partial g}{\partial x}(x_n) & \frac{\partial g}{\partial y}(x_n) \end{pmatrix}$,

$F(x_n) = \begin{pmatrix} f(x_n) \\ g(x_n) \end{pmatrix}$ [N.B. si on a f(x,y) et g(x,y), remplace (x_n) par (x_n,y_n).

4) ***Systèmes algébriques: méthodes indirectes***

a) **Méthode Jacobi** (méthode itérative):

i) Définir N, le nombre d'intervalles (n: nombre d'équation = N-1), K, le nombre d'itérations.

ii) Construire A=(a_{ij}) matrice du système Ax=b à résoudre (diaonale ou tridiagonale par exemple), b=[...] vecteur colonne des termes indépendants.

iii) Soit le système équivalent: x=Fx+c.

iv) Décomposer A en A=L+D+U (D: diagonale principale, U: triangulaire supérieure, L: triangulaire inférieure). $A = \begin{pmatrix} D_1 & U_1 & 0 & 0 \\ L_1 & D_2 & U_2 & 0 \\ 0 & L_2 & \dots & \dots \\ 0 & 0 & \dots & D_n \end{pmatrix}$

v) Calculer $F=I-D^{-1}A$ (I: matrice identité) ou encore $F=-D^{-1}(L+U)$ et $c=D^{-1}b$.

vi) Itération: $x_0 = x_n$, $x_n = F.x_0 + c$ (itérer pour k=1,...,K).

5) ***Systèmes mal conditionnés et incompatibles***

- ...
- Méthode des moindres carrés.

6) ***Interpolation et lissage***

a) Interpolation polynomiale de Vandermonde:

i) Trouver un polynôme p de degré $n \geq 0$, qui prend en $t_0,t_1,...t_n$, les valeurs $p_0,p_1,...,p_n$. Cad: $p(t_j)=p_j$ pour $0 \leq j \leq n$.

ii) On a les équations : $p_j=a_0+a_1t_j+a_2t^2_j+...+a_nt^n_j$.

iii) Résoudre le système Va=p (trouver les coefficients a_k) avec $V = \begin{pmatrix} 1 & t_0 & t_0^2 & \dots & t_0^n \\ 1 & t_1 & t_1^2 & \dots & t_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & t_n & t_n^2 & \dots & t_n^n \end{pmatrix}$. (matrice (n+1)x(n+1), on peut le résoudre sur Matlab avec: a=V\b).

iv) Pas conseillée, car elle pose des problèmes numériques.

b) *Polyfit* sur Matlab : p=polyfit(x,y,n)

i) p=polyfit(x,y,n) : donne dans p les coefficients d'un polynôme *P(x)* de degré n approchant les *y(i)* par *P(x(i))* (au sens des moindres carrés). $p=[p_1\ p_2\ ...\ p_{n+1}]$ où $P(x)= p_1x^n + p_2x^n + ... + p_{n+1}$.

ii) Procédure: - 1° Générer un vecteur contenant les abscisses x (uniformément réparties): *x=(a:b:c)'* [de a à c par pas de b. N.B. le " ' " indique qu'on doit prendre la transposée].

- 2° évaluer *y=f(x)* (où f est la fonction à approcher).
- 3° évaluer les coefficients: *p=polyfit(x,y,n)* (degré n).

- 4° évaluer le polynôme aux abscisses décidés précédemment : *g=polyval (p,x)* (pour le représenter et estimer la qualité de l'approximation).

c) **Interpolation linéaire**:
   i) Deux points successifs sont reliés par un segment de droite.

d) **Interpolation parabolique/quadratique**:
   i) Faire une boucle qui avance par pas de 2 (*for i=a:2:b*): On prend les abscisses $x_{pi}=[t_i\ t_{i+1}\ t_{i+2}]$ et les ordonnées $y_{pi}=[y_i\ y_{i+1}\ y_{i+2}]$.
   ii) On évalue $p_i=polyft(x_{pi},y_{pi},2)$.
   iii) On évalue un plus grand nombre de points avec: $u_i=t_i:0.1=t_{i+2}$ et $h_i=polyval(p_i,u_i)$ pour tracer h_i avec une plus grande précision.

e) **Splines cubiques**:
   i) Créer des abscisses (quelques points): *x=a:1:b*.
   ii) Evaluer *y=f(x)*.
   iii) Créer les abscisses (plus nombreuses): *t=a:0.01:x(end).*
   iv) Evaluer *cs_nat=spline(x,y)* (spline naturelle).
   v) Evaluer *u=ppval(cs_nat,t)* (pour la tracer).

7) <u>Intégration numérique de</u> $\int_a^b f(x)dx$

a) Formule des **trapèzes** [assimiler l'aire entre le graphe de f et l'axe des abscisses à la somme des aires de n trapèzes]
   i) Subdiviser [a,b] en sous-intervalles uniformes: $x_i=a+ih$ avec $h=\frac{b-a}{n}$, $f(x_i)=f_i$, i=0,1,2,...,n.
   ii) Sur l'intervalle $[x_{i-1},x_i]$: l'aire à sommer vaut $\frac{h(f_i+f_{i+1})}{2}$.
   iii) Formule des trapèzes: $\int_a^b f(x)dx \approx \frac{h}{2}(f_0+2f_1+2f_2+\ldots+2f_{n-1}+f_n)$
   iv) Méthode d'ordre 2 (erreur proportionnelle à h^2). Formule exacte pour les fonctions de degré ≤ 1.
   v) Formule des trapèzes améliorée:
   $\int_a^b f(x)dx \approx \frac{h}{2}[f_0+2f_1+2f_2+\ldots+2f_{n-1}+f_n]-\frac{h^2}{12}[f'(a)-f'(b)]$ (erreur $\propto h^4$).

b) Formule de **Simpson** [remplacer la fonction à intégrer par des segments de parabole]
   i) Subdiviser [a,b] en n sous-intervalles uniformes $[x_{i-1},x_i]$: $x_i=a+ih$ avec $h=\frac{b-a}{n}$, $f(x_i)=f_i$, i=0,1,2,...,n.
   ii) Sur chaque intervalle $[x_{i-1},x_i]$: l'aire à sommer vaut $\frac{h(f_{i-1}+4f_i+f_{i+1})}{3}$ (le répéter pour les n/2 intervalles).
   iii) Formule de Simpson:
   $\int_a^b f(x)dx \approx \frac{h}{3}(f_0+4f_1+2f_2+4f_{31}+\ldots+2f_{n-2}+4f_{n-1}+f_n)$
   iv) Méthode d'ordre 4, formule exacte pour les polynômes f(x) de degré ≤ 3.

c) Méthode de **Romberg**

i) Pour n=1,2,4,8,... calculer: $S_0(n) = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{n-1} + f_n), h = \frac{(b-a)}{n}$

ii) Pour n=2^{m-1},2^m,2^{m+1}...; m=1,2,3,... calculer: $S_m(2n) = \frac{4^m S_{m-1}(2n) - S_{m-1}(n)}{4^m - 1}$

iii) Très bonne méthode.

d) Fonction *quad*

i) Voir dans Matlab "*help quad*".

8) Equations différentielles ordinaires

Résoudre: $\begin{cases} \frac{dy}{dt}(t) = y'(t) = f(t, y(t)) \\ y(a) = y_0 \end{cases} \quad t \in [a,b]$

Données: h (pas), x_0,x_f, $N = \frac{(x_f - x_0)}{h}$,$y_0$, f(x,y)=..., dfdy=... (dérivée de f(x,y) p/r à y).

a) Méthode d'**Euler explicite** [règle du rectangle]

i) y_0=const (donnée par Pb Cauchy).

ii) Pour k=0,1,2,...,N-1 : $y_{k+1} = y_k + h.f(t_k, y_k)$, $t_{k+1} = t_k + h$.

b) Méthode d'**Euler implicite** [règle du rectangle]

i) y_0=const (donnée par Pb Cauchy).

ii) Pour n=1,...,N : - Poser z=y_n.

- Pour k=1,2,3,4. : • $g = z - y_n - h.f(x_{n+1}, z)$

• $g' = \frac{dg}{dz} = 1 - h.\frac{df}{dy}(x_{n+1}, z)$

• $z = z - \frac{g}{g'}$

- y_{n+1}=z.

c) Méthode de **Crank-Nicholson** [règle du trapèze]

i) y_0=const (donnée par Pb Cauchy).

ii) Pour n=1,...,N : - Poser z=y_n.

- Pour k=1,2,3,4. : • $g = z - y_n - \frac{h}{2}.[f(x_n, y_n) + f(x_{n+1}, z)]$

• $g' = \frac{dg}{dz} = 1 - \frac{h}{2}.\frac{df}{dy}(x_{n+1}, z)$ (df/dy peut être indépendante de x_{n+1}, donc on aurait: df(z)/dy).

• $z = z - \frac{g}{g'}$

- y_{n+1}=z.

d) Méthode de **Heun / Runge-Kutta à 2 paramètres**

i) y_0=const (donnée par Pb Cauchy).

ii) Pour n=1,...,N : - k1=f(x_n,y_n).

- $k2 = f(x_{n+1}, (y_n + h.f(x_n, y_n)))$

- $y_{n+1} = y_n + \frac{h}{2}.(k1 + k2)$

e) Notes:

i) Pour des pas (h) très petites, Euler explicite et implicite sont très précis. Ils sont d'ordre 1 en h => diviser le pas par 2 implique que l'erreur obtenue pour une valeur de t fixée est aussi divisée par deux!
ii) Euler explicite peut poser des problèmes de stabilité (problème avec des suites alternées: divergence au lieu de convergence) contrairement à Euler implicite.

9) ***Problèmes aux conditions aux limites***

Résoudre: $\begin{cases} -u''(x) + c(x).u(x) = f(x) & x \in [a,b] \\ u(a) = u_0 \\ u(b) = u_f \end{cases}$

a) Méthode des **différenes finies**

i) Divisons l'intervalle en N+1 sous-intervalles égaux, de longueur $h = \frac{(b-a)}{N+1}$, avec des abscisses: x_0=a, x_1=h, ..., x_j=jh, x_{N+1}=b.

ii) Discrétiser l'équation avec $\begin{cases} y(x_n) \approx y_n \\ y''_n \approx \frac{y_{n-1} - 2y_n + y_{n+1}}{h_N^2} \end{cases}$

iii) Résoudre le système tridiagonal NxN obtenu.

b) Méthode du **tir au but**

i) Soit le problème

$$\begin{cases} y''(x) + a(x,y(x),y'(x)).y'(x) + b(x,y(x),y'(x)).y(x) + c(x,y(x),y'(x)) = 0 & x \in [a,b] \\ y(a) = y_0 \\ y(b) = y_f \end{cases}$$

ii) Posons: $\begin{cases} y(x) = u_1(x) \\ y'(x) = u_2(x) \\ y'(a) = C \end{cases}$ (C à déterminer pour avoir y(b)=y_f).

iii) Trouver C_k en itérant:

$$\begin{cases} u'_1(x) = u_2(x) \\ u'_2(x) = -a(x,u_1(x),u_2(x)).u_2(x) - b(x,u_1(x),u_2(x)).u_1(x) - c(x,u_1(x),u_2(x)) \\ u_1(a) = y_0 \quad CI_1 \\ u_2(a) = C_k \quad CI_2 \end{cases}$$

iv) Ex: si problème linéaire par rapport à la fonction qu'on cherche (et sa dérivée).

- Choisir un C_1, trouver $y(b)_{C1}$ avec C_1 (continuer si $y(b)_{C1} \neq y_f$)
- Choisir un C_2, trouver $y(b)_{C2}$ avec C_2 (continuer si $y(b)_{C2} \neq y_f$)
- Prendre $C_3 = C_2.(\frac{y_f - y_{C1}}{y_{C2} - y_{C1}}) - C_1.(\frac{y_f - y_{C2}}{y_{C2} - y_{C1}})$, itérer pour trouver $y(b)_{C3}=y_f$.

N.B: pour résoudre le problème de Cauchy: utiliser ode45 et intégrer la fonction au point iii) (avec les CI à envoyer à ode45).

c) Méthode de **collocation**

i) Pas d'idées...

10) ***Convergence***

- exponentielle: tracer la graphe de l'écart maximum (en prendre son logarithme en base 10) entre le résultat que l'on obtient et le résultat qu'il faudrait obtenir (par exemple: si on a f(x)=0, tracer pour les a décidés, l'écart entre f(a) et 0), et voir s'il est linéaire ou non.
- Etudier l'évolution des erreurs (relatives et absolues):
  - Erreur absolue: écart entre la courbe obtenue et la courbe étudiée (de référence).
  - Erreur relative: Erreur absolue nomralisée (divisée par la valeur de f(a) où f est la fonction et a le point d'abscisse où on calcule l'erreur commise).

11) Commentaires

Joyeux Noël et bonne année! Et surtout... Joyeux blocus!