TP MT00

Printemps 2024

1 Déroulement

Les TPs de MT00 ont pour objectif de présenter des applications directes de notions vues en cours. Le langage de programmation utilisé est Scilab, dont un tutoriel est à votre disposition. Voici le planning des TP :

1. Semaine 1 : Introduction à Scilab et calculs de base d'algèbre linéaire.
2. Semaines 2 et 3 : Systèmes linéaires : élimination de Gauss, factorisations, applications.
3. Semaines 4 et 5 : Méthodes itératives : Extraction de racine, méthode de Newton.
4. Semaines 6 et 7 : Interpolation et intégration numérique.
5. Semaines 8 et 9 : Méthodes d'Euler, schémas de Runge Kutta et applications.
6. Semaine 10 : Finalisation des TP

Quelques conseils :

— Prendre connaissance de l'énoncé du TP avant la séance de TP.

— Créer un répertoire personnel où vous pouvez stocker l'ensemble des scripts Scilab.

— Préférer l'utilisation de scripts (ou fonctions) à un travail uniquement en lignes de commande.

2 TP 1

2.1 Pour commencer

Saisissez dans la console de Scilab les éléments suivants et observez les résultats obtenus (y compris les messages d'erreurs) :

$A = [1\ 2\ 3; 4\ 5\ 6]$	$A * C$	$I = eye(5, 5)$	$A(1, 2 : 3)$	$z = 0 : 5$
$B = [1, 2, 3; 4, 5, 6]$	$C * A$	$M = zeros(2, 4)$	$I(1 : 2, 2 : 5)$	$t = 0 : 0.5 : 5$
$C = [1\ 2; 3\ 4; 5\ 6]$	$A * A$	$N = ones(4, 3)$	$size(A)$	$p = 5 : 0$
$D = C'$	$A * A'$	$A(2, 3)$	$[m\ n] = size(A)$	$p = 5 : -1 : 0$
$x = [7\ 8\ 9]$	$x * C$	$A(1, :)$	$size(A, 1)$	$q = linspace(0, 10, 3)$
$y = [7; 8; 9]$	$y' * C$	$A(:, 1)$	$size(A, 2)$	$clear$

2.2 Transformations du Plan

• On se place dans le plan et le repère $(O, \vec{i}, \vec{j})$. Ecrire la matrice R de rotation d'angle $\pi/4$ et de centre 0. Ecrire la matrice S de symétrie par rapport à la droite $y = x$. Ecrire la matrice P de la projection sur l'axe (Ox) parallèlement à (Oy). Ecrire la matrice H qui multiplie tous les vecteurs par une constante positive $k = 4$ (on parle de changement d'échelle ou homothétie de rapport k et de centre 0). Pour rappel, l'écriture matricielle M d'une application linéaire f est sous la forme :

$$M = \begin{array}{cc} & \begin{array}{cc} f(\vec{i}) & f(\vec{j}) \end{array} \\ & \left(\begin{array}{cc} * & * \\ * & * \end{array} \right) \begin{array}{c} \vec{i} \\ \vec{j} \end{array} \end{array}$$

• Effectuez les produits matriciels RP, PR, RH, HR, S^2 et vérifiez que le produit de matrices correspond à la composition des applications. Le produit est-il commutatif ?

2.3 Tracé de fonctions

• Représenter les fonctions $f(x) = sin(x)$ et $g(x) = x - \frac{x^3}{6}$ sur l'intervalle $[0, \pi/2]$ sur une même figure.

2.4 Algèbre linéaire

• Calculer le déterminant de la matrice d'ordre $n \in \{3, 5\}$ pour les matrices carrées de terme général $a_{i,j} = max(i, j)$, puis $a_{i,j} = i^j$.

• Calculer l'inverse de la matrice d'ordre 5 et de terme général $a_{i,j} = C_i^j$ si $i \geq j$, 0 sinon.

• Ecrire une fonction ayant en entrée une matrice X et en sortie la somme partielle suivante :

$$S(X) = \sum_{k=0}^{20} \frac{(X^T X)^k}{k!}$$

3 TP 2 et 3

3.1 Algorithmes de base

On rappelle que si la matrice $A \in \mathcal{M}_{n,n}$ est inversible et triangulaire inférieure, alors la solution de $Ax = b$ est donnée par $x_1 = \frac{b_1}{a_{1,1}}$ et pour $i > 1$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j} x_j}{a_{i,i}}.$$

Voici le script Scilab `TP_Ch2_partie1.sci` :

```
function x=solveLowerTri(A,b)
  n=size(A,1);
  x=zeros(n,1);
  x(1)=b(1)/A(1,1);
  for i=2:n
      x(i)=(b(i)-A(i,1:i-1)*x(1:i-1))/A(i,i);
  end
endfunction

L=[1 0 0;2 1 0;3 2 1];
b=[1 -1 1]';

x=solveLowerTri(L,b);
disp(x)
```

Il définit une fonction `solveLowerTri` permettant de faire cette résolution, puis l'utilise pour résoudre un système d'équations particulier. Vous noterez que dans cette fonction la somme $\sum_{j=1}^{i-1} a_{i,j} x_j$ est directement calculée comme le produit d'un vecteur ligne par un vecteur colonne (le code Scilab prend en compte le cas i=1, dans ce cas les deux vecteurs sont vides et la somme est nulle)

3.2 Travail à réaliser

Exécutez ce script et vérifiez que la solution donnée par le programme est correcte. Dans le même script Scilab, écrivez trois nouvelles fonctions :

1. une fonction nommée `solveUpperTri` résolvant un système linéaire $Ax = b$ dont la matrice A est triangulaire supérieure :

   ```
   function x=solveUpperTri(A,b)
   ...
   endfunction
   ```

2. une fonction nommée `EliminationGauss` transformant un système de type $AX = b$ en un système de type $\widehat{A}X = \widehat{b}$ avec $\widehat{A}$ triangulaire supérieure. Cet algorithme renvoie un message d'erreur (flag à 0) si un pivot est nul.

   ```
   function [Achap,bchap,flag]=EliminationGauss(A,b)
   ...
   endfunction
   ```

3. une fonction nommée `factorizeLU` calculant la factorisation $A = LU$ d'une matrice A (sans permutation de lignes) :

```
function [L,U]=factorizeLU(A)
...
endfunction
```

Vérifiez vos fonctions à l'aide de la matrice et du vecteur

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \ b = \begin{pmatrix} 4 \\ 5 \\ 3 \end{pmatrix}$$

3.3 Application

Nous vous proposons de calculer une approximation de la solution $u(x)$ de l'équation aux dérivées partielles suivante

$$\left\{ \begin{array}{rcl} -u''(x) & = & f(x) \ , \quad x \in]0,1[\\ u(0) & = & 0 \\ u(1) & = & 0 \end{array} \right. \tag{1}$$

Vous pouvez imaginer $x \to u(x)$ comme la distribution de température d'une barre de longeur 1, chauffée de l'interieur par une quantité de chaleur $f(x)$. On suppose que les échanges de chaleur avec l'extérieur ne s'effectuent que par les extremités de la barre, qui sont maintenues à une température nulle. Dans certains cas particuliers il est possible d'obtenir la solution exacte (par exemple quand f est une fonction constante). Dans le cas général on cherche une approximation de $u(x)$ en certains points (x_i), avec $i = 0 \dots N$. Pour simplifier, on choisit des points x_i équidistants ; on choisit donc $h = \frac{1}{N}$, et on a

$$x_i = ih$$

En ces points, l'équation se récrit :

$$\begin{array}{rcl} -u''(x_i) & = & f(x_i) \quad , \quad i = 1, \dots, N-1 \\ u(x_0) = u(x_N) & = & 0 \end{array} \tag{2}$$

On remarque par (2) que la valeur de u est connue sur les bords du domaine $[0, 1]$. Les dérivées apparaissant dans l'équation sont approchées par des formules n'utilisant que la valeur de u aux points (x_i). On peut approcher la dérivée seconde de u à l'aide de la formule classique :

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

Si on néglige le reste $\mathcal{O}(h^2)$, on peut obtenir une approximation $v_i \approx u(x_i)$ en résolvant le système linéaire suivant :

$$-\frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \end{pmatrix}$$

et $v_0 = v_N = 0$.

3.4 Travail à réaliser

1. Déterminer la solution exacte de (1) pour les deux second membres suivants :
   — $f(x) = 1$,
   — $f(x) = \sin(\pi x)$.
2. Écrivez un programme qui donne l'approximation $[v_i]_{i\leq 0\leq N}$ de $[u(x_i)]_{i\leq 0\leq N}$ pour les deux second membres proposés ci-dessus (vous prendrez successivement $N = 3, 5, 10$), en utilisant les fonctions `factorizeLU`, `solveLowerTri` et `solveUpperTri`. Utilisez les possibilités graphiques de Scilab pour représenter à l'écran vos résultats : vous pouvez par exemple superposer sur un même graphe l'approximation obtenue $[v_i]_{i\leq 0\leq N}$ avec les valeurs théoriques $[u(x_i)]_{i\leq 0\leq N}$.
3. Quelle autre autre factorisation de la matrice A pourrait être utilisée ? Écrivez une nouvelle fonction calculant cette factorisation.

4 TP 4 et 5

4.1 La méthode de Gauss-Seidel

• Implémenter la méthode de Gauss Seidel associée à la résolution de $Ax = b$. On choisira comme vecteur initial un vecteur aléatoire de taille n. Le critère d'arrêt est obtenu lorsque $\left\| Ax^{(k)} - b \right\| < \epsilon = 10^{-8}$.

Vérifier la justesse de l'algorithme à l'aide des exemples suivants

— $A = \begin{bmatrix} 8 & 2 & 0 \\ 0 & 5 & 3 \\ 3 & 1 & 6 \end{bmatrix}$ et $b = \begin{bmatrix} 10 \\ 11 \\ 16 \end{bmatrix}$

— A la matrice d'ordre n et de terme général $a_{i,j} = min(i, j)$ et b vecteur colonne formé de 1.

4.2 Méthode de Newton

Le critère d'arrêt dans cette section correspond à $\left\| x^{(k+1)} - x^{(k)} \right\| < \epsilon = 10^{-8}$.

• Implémenter la méthode de Newton afin d'obtenir la fonction racine-p-ème ($p \geq 1$) d'un réel positif N. L'algorithme est le suivant :

$$x^{(0)} = 1 \qquad x^{(k+1)} = \frac{1}{p}((p-1)x^{(k)} + \frac{N}{\left(x^{(k)}\right)^{p-1}})$$

• Implémenter la méthode de Newton pour obtenir les racines du polynôme $P(x) = x^3 - 10x + 2$ et une valeur initiale $x_0 \in [-10, 10]$. Vérifiez que l'algorithme converge (pratiquement toujours) vers une des trois racines de P. Vérifier que l'algoritme converge aussi (pratiquement toujours) si x_0 est un nombre complexe, par exemple $2 + i$.

• Implémenter la méthode de Newton pour obtenir les racines de la fonction $\sqrt{|x|}$ avec une valeur initiale quelconque entre 1 et 10. Expliquer le résultat.

• Implémenter la méthode de Newton pour obtenir les racines réelles (et complexes) du polynôme $P(x) = x^7 - 2x^3 + 5$. Pour cela, on peut montrer facilement que toutes les racines sont de module inférieur à 2 et ne s'intéresser qu'aux points x_0 de module plus petit que 2.

4.3 Dichotomie

• Implémenter la méthode par dichotomie afin d'obtenir la fonction racine-p-ème d'un réel positif N. On choisit comme valeur initiale $a = 0$ et $b = N + 1$. Le critère d'arrêt correspond à $\left\| x^{(k+1)} - x^{(k)} \right\| < \epsilon = 10^{-8}$.

• Pour une précision identique et une fonction donnée, comparer le temps d'excution de l'algorithme de Newton et celui par dichotomie.

5 TP 6 et 7

• Ecrire un algorithme permettant d'obtenir la matrice de VanDerMonde puis de déterminer les coefficients du polynôme d'interpolation.

ENTREE : Vecteur ti (taille n), vecteur yi=f(ti) (taille n)
SORTIE : coefficients ai du polynôme

• Déterminer à partir du programme le polynôme P d'interpolation pour $t = [1, 2, 3, 4]$ et $y = [4, 3, 2, 1]$. Déterminer P(10^{10}). Déterminer ensuite P théoriquement et calculer P(10^{10}). On pourra s'aider des fonctions poly et horner.

• Ecrire un algorithme permettant d'obtenir les coefficients c_i dans la base de Newton.

ENTREE : Vecteur ti (taille n), vecteur yi=f(ti) (taille n)
SORTIE : la matrice des différences divisées OU les ci uniquement

• Ecrire un algorithme permettant d'évaluer un polynôme dans la base de Newton avec le schéma de Horner. Reprendre la question 2.

ENTREE : Vecteur des ci (taille n), x (réel ou complexe ou vecteur)
SORTIE : P(x)

• On considère la fonction f définie sur $[0, \frac{\pi}{2}]$ par $f(x) = x^3 sin(x)$. A partir de la méthode composite des rectangles (à gauche, centrée ou à droite), évaluer $\int_0^{\frac{\pi}{2}} f(t)dt$ pour différents tailles de subdivision. Comparer à la valeur exacte $\frac{3}{4}(\pi^2 - 8)$.

• On considère la fonction f définie sur $[0, \frac{\pi}{2}[$ par $f(x) = ln(cos(x))$. A partir de la méthode composite des rectangles (à gauche, centrée ou à droite), évaluer $\int_0^{\frac{\pi}{2}} f(t)dt$ pour différents tailles de subdivision. Comparer à la valeur exacte $\frac{-\pi ln(2)}{2}$. Pourquoi l'utilisation de la méthode des rectangles n'est pas complètement justifiée ici ?

• On considère la fonction f définie sur [-1,1] par $f(x) = 1/(1+(7+4\sqrt{3})*x^2)$

a) On choisit des noeuds réguliers (6). Tracer la fonction f et son polynôme d'interpolation.

b) On choisit des noeuds de Tchebychev (6). Tracer la fonction f et son polynôme d'interpolation.

c) On choisit une formule de quadrature pour calculer $\int_{-1}^{1} f(t)dt$. Ecrire une fonction pour obtenir les poids wi par système linéaire (VanDerMonde).

ENTREE : Vecteur ti (taille n)
SORTIE : Vecteur wi (taille n)

d) Pour différentes valeurs de n, comparer les calculs d'intégrale entre les noeuds réguliers et les noeuds de Tchebychev et les comparer à la valeur exacte $5\pi(2 - \sqrt{3})/6$.

e) Implémenter l'algorithme d'intégration numérique composée basée sur la Formule de Simpson et des noeuds réguliers (17).

6 TP 8 et 9

6.1 Équation différentielle linéaire d'ordre un

On considère l'équation différentielle suivante :

$$\left\{ \begin{array}{rcl} y'(t) & = & \mathbf{i}y(t)\ , \quad t \in]0,10[\\ y(0) & = & 1 \end{array} \right. \tag{3}$$

• Déterminer la solution théorique ($i^2 = -1$).

• Implémenter la méthode d'Euler explicite associée à cette équation différentielle. On pourra par exemple écrire une fonction acceptant comme argument $y0$, une valeur de temps maximale T et un nombre d'observations N (en particulier le pas $h = \frac{T}{N}$) et la fonction associée au schéma numérique f. En sortie, nous pouvons obtenir deux vecteurs, un vecteur $(t_n)_{n=0..N}$ associé à la subdivision régulière entre 0 et T, et un vecteur $(y_n)_{n=0..N}$ donnant l'approximation de la solution par le Schéma d'Euler explicite :

```
function y=myeuler(y0,T,N,f)
h=T/N;
t(1)=0;
y(1)=y0;
...
endfunction
```

• De manière similaire, implémenter la méthode d'Euler implicite.

• De manière similaire, implémenter la méthode d'Euler-Cauchy (predicteur correcteur).

6.2 Équation différentielle linéaire d'ordre deux

Rappel de cours sur les schémas de Runge Kutta :

1. Schéma d'Euler

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ y_{n+1} &= y_n + hk_1. \end{aligned}$$

2. Schéma du point milieu (k_1 est défini comme pour le schéma précédent)

$$\begin{aligned} k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \\ y_{n+1} &= y_n + hk_2. \end{aligned}$$

3. Schéma de Runge et Kutta d'ordre 4 (k_1 et k_2 sont définis comme pour le schéma précédent)

$$\begin{aligned} k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2), \quad k_4 = f(t_n + h, y_n + hk_3), \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

On considère l'équation différentielle suivante :

$$\left\{\begin{array}{rcl} y''(t) & = & 3y(t) - 2y'(t) - 8e^{-t}\ , \quad t \in]0,10[\\ y(0) & = & 1 \\ y'(0) & = & 1 \end{array}\right. \tag{4}$$

• Vérifier que la solution de cette équation est $y(t) = 2e^{-t} - e^{-3t}$.
• Implémenter la méthode d'Euler explicite.

• Implémenter le schéma du point milieu (RK2).

• Implémenter le schéma RK4.

• Vérifier que pour une valeur de N donnée, RK4 est plus performante que RK2, elle-même meilleure qu'Euler explicite.

• Comparer les résultats avec la solution donnée par la fonction *ode* de Scilab.

6.3 Le pendule

On considère un pendule initialement au repos et présentant une déviation θ_0 par rapport à la verticale. L'équation différentielle régissant la déviation $\theta(t)$ est la suivante :

$$\left\{\begin{array}{rcl} \ddot{\theta}(t) & = & -\frac{g}{L}\sin\theta(t), \\ \theta(0) & = & \theta_0, \\ \dot{\theta}(0) & = & 0. \end{array}\right. \tag{5}$$

Lorsque θ_0 est petit, on peut se permettre de faire une approximation de $\sin\theta(t)$ au premier ordre, ce qui conduit à poser

$$\sin\theta(t) \approx \theta(t),$$

et dans ce cas on montre aisément que l'on a

$$\theta(t) = \theta_0 \cos\omega t, \quad \omega = \sqrt{g/L} \tag{6}$$

• Pour différentes valeurs de θ_0, tracer sur un même graphique la solution linéarisée et l'approximation obtenue par Scilab.