Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séances 1 et 2 : prise en main de MATLAB et GNU OCTAVE

Le symbole ⋄ indique un exercice optionnel. Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Exercice 1 (manipulation et opérations sur les tableaux).

1. On considère les vecteurs et matrices suivants

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

   a. Créer des tableaux correspondants à $\boldsymbol{l}$, $\boldsymbol{v}$, A et B.

   b. Extraire la première ligne, la deuxième colonne et la sous-matrice $(a_{ij})_{1\leq i\leq 3, 1\leq j\leq 5}$ de la matrice A, les termes diagonaux de la matrice B.

   c. Commenter le résultat produit par les instructions suivantes : `l*v`, `v*l`, `l./v'`, `A*B`, `B/A`, `A.*B`, `A*A'`, `sin(l)` et `exp(A)`.

   d. Comment déterminer le format (nombres de lignes et de colonnes) de ces tableaux ?

2. MATLAB et GNU OCTAVE permettent également la génération « automatique » de tableaux particuliers. À titre d'illustration, analyser [1] et commenter le résultat produit par chacune des instructions suivantes :

```
r=[1.3:15.8]
s=[1.3:0.4:15.8]
t=linspace(1.3,15.8,5)
u=ones(size(t))
v=3*ones(1,5)
w=sin([0:pi/6:pi])
x=eye(3,3)
y=rand(1,5)
z=zeros(5,1)
```

3. On considère les vecteurs de $\mathbb{R}^3$ suivants

$$\boldsymbol{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \ \boldsymbol{v} = \begin{pmatrix} -5 \\ 2 \\ 1 \end{pmatrix} \text{ et } \boldsymbol{w} = \begin{pmatrix} -1 \\ -3 \\ 7 \end{pmatrix}.$$

   a. Créer des tableaux correspondants à ces vecteurs.

   b. Calculer $\boldsymbol{u} + \boldsymbol{v}$, $\boldsymbol{u} + 3\,\boldsymbol{v} - 5\,\boldsymbol{w}$, $\frac{1}{5}\,\boldsymbol{w}$.

   c. En utilisant les commandes appropriées [2], calculer $\|\boldsymbol{u}\|_2$, $\|\boldsymbol{v}\|_1$, $\|\boldsymbol{w}\|_\infty$ et le cosinus de l'angle formé par les vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$.

1. Ne pas hésiter à faire appel à l'aide en ligne via l'instruction `help` suivie du nom de la commande concernée pour comprendre son fonctionnement.
2. On pourra notamment consulter les aides en ligne des fonctions `norm` et `dot`.

4. On considère les matrices suivantes :

$$A = \begin{pmatrix} 2 & 3 \\ 6 & 5 \end{pmatrix} \text{ et } B = \begin{pmatrix} 2 & 3 & 4 \\ 7 & 6 & 5 \\ 2 & 8 & 7 \end{pmatrix}.$$

   a. Créer des tableaux correspondants à A et B.

   b. En utilisant les commandes appropriées, calculer leurs déterminants, inverses et valeurs propres et vecteurs propres associés [3].

5. Créer des tableaux correspondants à la matrice identité d'ordre 6 et à la matrice nulle d'ordre 3.

6. Pour $n \in \mathbb{N}$, $n \geq 2$, on considère la matrice tridiagonale d'ordre n définie par

$$A_n = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

   a. Que fait la suite d'instructions suivante ?

```
S=[eye(n) zeros(n,1)];
S=S(:,2:n+1);
A=2*eye(n)-S-S'
```

   b. Répondre à la même question avec la suite d'instructions ci-dessous.

```
D=diag(ones(n,1));
SD=diag(ones(n-1,1),1);
A=2*D-SD-SD'
```

7. On considère la matrice et les vecteurs

$$A = \frac{1}{8}\begin{pmatrix} 5 & -2 & 1 \\ 2 & 0 & 2 \\ 1 & -2 & 5 \end{pmatrix},\ \boldsymbol{b} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \text{ et } \boldsymbol{u}^{(0)} = \begin{pmatrix} 5 \\ 2 \\ -4 \end{pmatrix},$$

   et la suite de vecteurs définie par

$$\boldsymbol{u}^{(n+1)} = A\,\boldsymbol{u}^{(n)} + \boldsymbol{b},\ \forall n \in \mathbb{N}.$$

   a. Créer des tableaux correspondants à A, $\boldsymbol{b}$ et $\boldsymbol{u}^{(0)}$.

   b. Calculer les premiers termes de la suite $(\boldsymbol{u}^{(n)})_{n\in\mathbb{N}}$. Qu'observe-t-on ?

   c. Répondre à la question précédente avec

$$A = \begin{pmatrix} 5 & 6 & 3 \\ -1 & 5 & -1 \\ 1 & 2 & 0 \end{pmatrix},\ \boldsymbol{b} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \text{ et } \boldsymbol{u}^{(0)} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}.$$

   d. Interpréter ces résultats.

Exercice 2 (écriture de fonctions). Les fonctions de MATLAB (ou de GNU OCTAVE) sont implémentées dans des fichiers texte de type `.m`, crées avec un simple éditeur de texte. Un tel fichier débute par le mot `function`, suivi du nom de la fonction et des paramètres d'entrée et de sortie de cette dernière. Cette déclaration est de la forme :

3. On consultera l'aide en ligne de la commande `eig`, les termes « valeur propre » et « vecteur propre » se traduisant respectivement par *"eigenvalue"* et *"eigenvector"* en anglais.

```
function [y_1,y_2,...] = mafonction(x_1,x_2,...)
 % Les commentaires situes juste apres la declaration de la fonction
 % constituent l'aide obtenue en entrant la commande
 % >> help mafonction
```

Le nom de la fonction et le préfixe du fichier `.m` la contenant doivent impérativement être identiques [4]. Dans l'exemple ci-dessus, le fichier doit s'appeler `mafonction.m`. Elle sera appelée depuis la ligne de commande, un script ou une autre fonction de la manière suivante :

```
>> [x,y]=mafonction(a,b)
```

On notera qu'il est possible de récupérer le code source de toute fonction pré-existante au moyen de la commande `type`.

1. Écrire une fonction nommée `polaire`, prenant comme arguments d'entrée les coordonées cartésiennes (x, y) d'un point de $\mathbb{R}^2$ et renvoyant en sortie les coordonnées polaires (r, θ) de ce point [5]. Penser à commenter le code source de manière à ce qu'un utilisateur puisse utiliser l'aide en ligne pour s'informer sur cette nouvelle fonction.
2. En utilisant une structure de contrôle de type `if ... else ...`, écrire une fonction nommée `profil` qui associe à toute matrice A à m lignes et n colonnes son *profil*, c'est-à-dire la suite d'entiers $\{\phi(i)\}_{i=1,\dots,m}$ avec ϕ une application de $\{1, \dots, m\}$ dans $\mathbb{N}$ telle que

$$\phi(i) = \begin{cases} \inf\{j \in \{1, \dots, n\} \,|\, a_{ij} \neq 0\} & \text{si la } i^{\text{ème}} \text{ ligne de } A \text{ est non nulle,} \\ n + i & \text{sinon.} \end{cases}$$

Exercice 3 (représentation graphique avec la commande `plot`). On cherche à obtenir une représentation graphique de la fonction $f(x) = \exp(-x)\sin(4\,x)$ sur l'intervalle $[0, 2\,\pi]$.

1. Que contiennent les tableaux créés via les commandes suivantes ?

```
x=linspace(0,2*pi,101);
y=exp(-x).*sin('*x);
```

2. Tracer alors la représentation graphique de la fonction f associées aux tableaux `x` et `y`. En utilisant le zoom, déterminer une valeur approchée du maximum de f sur $[0, 2\,\pi]$. Comment affiner le tracé pour préciser ce maximum ?
3. En utilisant l'instruction `hold on`, tracer sur une même figure une représentation graphique des fonctions $x \mapsto x^2$ et $x \mapsto x^2 \sin(x) \exp(-x)$ sur l'intervalle $[-1, 1]$ en utilisant des couleurs différentes pour chacune d'entre elles.

Exercice 4 ◇ (nombres complexes). On note u et v les nombres complexes

$$u = 11 - 7\mathrm{i} \text{ et } v = -1 + 3\mathrm{i}.$$

Calculer les modules de u et de v, les produits $u\overline{v}$ et $\overline{u}v$, les parties réelle et imaginaire de $u^3 + v^2$.

---

4. Pour éviter d'éventuels conflits dûs à l'emploi de noms de fonctions utilisés par MATLAB, on prendra soin de vérifier qu'une fonction de même nom n'existe pas déjà (grâce à la commande `help` par exemple) et/ou de mettre des majuscules dans le nom (MATLAB étant sensible à la casse).

5. On rappelle que, en vertu du théorème de Pythagore, on a $r = \sqrt{x^2 + y^2}$ et que, pour obtenir l'angle θ dans l'intervalle $[0, 2\pi[$, on utilise les formules suivantes :

$$\theta = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{si } x > 0 \text{ et } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) + 2\,\pi & \text{si } x > 0 \text{ et } y < 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{si } x < 0, \\ \frac{\pi}{2} & \text{si } x = 0 \text{ et } y < 0, \\ \frac{3\,\pi}{2} & \text{si } x = 0 \text{ et } y < 0. \end{cases}$$

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séance 3 : boucles et récursivité

Le symbole ⋄ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Exercice 1 (suite de Fibonacci [1]).

1. Calculer les valeurs des vingt premiers termes de la suite de Fibonacci définie par :
$$u^{(0)} = 0,\ u^{(1)} = 1 \text{ et } u^{(n+2)} = u^{(n+1)} + u^{(n)},\ \forall n \in \mathbb{N},$$
et garder ces valeurs dans un vecteur.
2. Calculer les termes successifs de la suite tant que $u_n \leq 100$ et afficher le dernier terme de la suite inférieur à 100.
3. Écrire enfin une fonction `fibonacci(n)` calculant de manière itérative le $n^{\text{ième}}$ terme de la suite de Fibonacci, sans toutefois conserver les valeurs de tous les termes de la suite.

Exercice 2 (suites adajacentes). On définit deux suites $(u^{(n)})_{n\in\mathbb{N}}$ et $(v^{(n)})_{n\in\mathbb{N}}$ par
$$u^{(0)} = 1,\ v^{(0)} = 2,\ u^{(n+1)} = \frac{u^{(n)} + v^{(n)}}{2},\ v^{(n+1)} = \sqrt{u^{(n+1)}v^{(n)}},\ \forall n \in \mathbb{N}.$$
On admet que ces suites sont adjacentes, de limite $\dfrac{\sqrt{27}}{\pi}$.

1. Écrire un programme lisant un entier k et affichant l'approximation du nombre π obtenue à partir de la valeur de $v^{(k)}$.
2. Écrire un programme lisant un réel ε et affichant l'approximation du nombre π obtenue à partir de la valeur de $v^{(k)}$, premier terme de la suite $(v^{(n)})_{n\in\mathbb{N}}$ à satisfaire la condition
$$\left|\frac{u^{(k)} - v^{(k)}}{u^{(k)} + v^{(k)}}\right| \leq \varepsilon.$$

Exercice 3 ⋄ (développement en série de cos et sin). Écrire une fonction `cosn(n,x)`, prenant comme argument un entier naturel non nul n et un réel x, calculant une approximation de la valeur de $\cos(x)$ obtenue en ne conservant que les n premiers termes du développement en série
$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$
Comparer les résultats de cette fonction avec ceux de `cos(x)` pour différentes valeurs de n. Écrire de la même manière une fonction `sinn(n,x)` basée sur le développement
$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

1. Leonardo Pisano (v. 1175 - v. 1250), dit Fibonacci, était un mathématicien italien. Il reste connu de nos jours pour un problème conduisant aux nombres et à la suite qui portent son nom, mais, à son époque, ce sont surtout les applications de l'arithmétique au calcul commercial (calcul du profit des transactions, conversion entre monnaies de différents pays) qui l'ont rendu célèbre.

Exercice 4 (programmation récursive). En informatique, une fonction dite *récursive* lorsqu'elle s'appelle elle-même. En pratique, une telle fonction aura toujours au moins une instruction conditionnelle, afin que, dans certains cas au moins, il n'y ait pas d'appel récursif (sans quoi la fonction s'appellerait indéfiniment jusqu'à la saturation de la pile, provoquant ainsi un plantage du programme). Le concept de fonction récursive est généralement opposé à celui de fonction *itérative*, qui s'exécute sans s'invoquer ou s'appeler explicitement.
Bien que cette forme de programmation aboutisse à des programmes concis et proches des formulations mathématiques qui en sont à l'origine, il peut parfois être mal indiqué ou même catastrophique d'employer la récursivité (toute fonction récursive pouvant être remplacée par une fonction itérative), comme on le vérifiera à la troisième question du présent exercice.

1. Écrire une fonction récursive `rfactorielle(n)` calculant $n!$.
2. Écrire une fonction récursive `PGCD(a,b)` renvoyant le plus grand commun diviseur[2] des entiers relatifs a et b calculé par l'*algorithme*[3] *d'Euclide*[4].
3. Écrire une fonction récursive `rfibonacci(n)` calculant les n premiers termes de la suite de Fibonacci et comparer son temps d'exécution avec celui de la fonction `fibonacci(n)` de l'exercice 1.
4. Écrire une fonction récursive `syracuse(n)` renvoyant la valeur 1 si la conjecture de Syracuse[5] est vérifiée pour l'entier $n > 0$.
5. Écrire une fonction récursive `rcosn(n,x)` de calcul de l'approximation de $\cos(x)$ vue dans l'exercice 3 utilisant la relation :

$$u^{(0)} = 1 \text{ et } u^{(n)} = -\frac{u^{(n-2)}}{n(n-1)}\, x^2,\ \forall n \geq 2.$$

2. C'est-à-dire le plus grand entier naturel qui divise simultanément ces deux entiers.

3. Cet algorithme est basé sur la propriété suivante : *on suppose que $a \geq b$ et on note r le reste de la division euclidienne de a par b ; alors le pgcd de a et b est le pgcd de b et r.* En pratique, il suffit donc de faire des divisions euclidiennes successives jusqu'à trouver un reste nul.

4. Euclide (Ευκλείδης, v. 325 avant J.-C. - v. 265 avant J.-C.) était un mathématicien de la Grèce antique ayant probablement vécu en Afrique. Il est l'auteur des *Éléments*, un traité de mathématiques et de géométrie qui est considéré comme l'un des textes fondateurs des mathématiques modernes.

5. On appelle *suite de Syracuse* une suite d'entiers naturels définie de la manière suivante : *on part d'un nombre entier plus grand que zéro ; s'il est pair, on le divise par deux ; s'il est impair, on le multiplie par trois et on ajoute un au résultat.* La suite est alors obtenue en répétant cette opération. Après que le nombre 1 a été atteint, la suite devient périodique, les valeurs $(1, 4, 2)$ se répétant indéfiniment en un cycle appelé *cycle trivial*. La *conjecture de Syracuse*, ou *de Collatz*, est l'hypothèse mathématique selon laquelle les suites de Syracuse de tous les nombres entiers strictement positifs atteignent le cycle trivial.

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séance 4 : quelques premières applications du calcul scientifique

Le symbole $\diamond$ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Exercice 1 (calcul d'une valeur approchée de π par la méthode de Monte-Carlo [1]). Pour obtenir une valeur approchée du nombre π par la méthode de Monte-Carlo, on tire « au hasard » [2], dans un carré de côté 2, des points de coordonnées (x, y) et l'on vérifie s'ils appartiennent ou non au disque de rayon 1 et de centre le centre du carré. Les points pouvant être tirés avec la même probabilité dans tout le carré, le rapport du nombre de points tirés dans le disque sur le nombre de points tirés au total tend avec le nombre de tirages vers le rapport des surfaces du cercle et du carré, soit $\frac{\pi}{4}$.

1. Au moyen de la commande `rand`, qui renvoie un nombre réel positif tiré de manière pseudo-aléatoire dans l'intervalle $[0, 1]$, écrire une fonction prenant comme argument le nombre de tirages à réaliser et renvoyant la valeur approchée de π obtenue par la méthode de Monte-Carlo correspondante (pour simplifier, on pourra se restreindre au quart de carré contenu dans l'orthant positif de $\mathbb{R}^2$).
2. Donner un ordre du nombre de tirages nécessaires pour obtenir plus que les deux premières décimales de π. Que dire de l'efficacité de cette méthode ?

Exercice 2 (applications linéaires entre espaces de polynômes). Soit n un entier naturel non nul. On note $\mathbb{R}_n[X]$ l'espace vectoriel des fonctions polynomiales de degré inférieur ou égal à n.

1. Pour P et Q deux éléments de $\mathbb{R}_n[X]$, on note $R(P, Q)$ le *reste de la division euclidienne de P par Q*. On considére l'application linéaire :

$$\begin{array}{rccc} \mathcal{R} & : & \mathbb{R}_n[X] & \to & \mathbb{R}_n[X] \\ & & P & \mapsto & R(P, X^2). \end{array}$$

   a. Écrire une fonction prenant comme paramètre un entier n et renvoie la matrice $M_{\mathcal{R}}$ de l'application $\mathcal{R}$ dans la base canonique $\{1, X, X^2, \ldots, X^n\}$ de $\mathbb{R}_n[X]$.

   b. À l'aide de cette fonction, calculer le reste de la division par x^2 pour les polynômes suivants :

   (i) $7\,x^8 + 411\,x^7 - 231\,x^5 + 31\,x^4 + 451\,x^3 - 231\,x - 42$,

   (ii) $x^7 + \dfrac{5}{21}\,x^5 + 0.432\,x^4 - 22\,x^3 + 51\,x^2 - \dfrac{1}{39}\,x + 4.431$.

   c. Calculer pour $n = 6, 7$ et 8 le noyau de $\mathcal{R}$. Que constate-t-on ?

1. On appelle *méthode de Monte-Carlo* toute méthode visant à calculer une approximation numérique par utilisation de procédés aléatoires. Le nom de ce type de méthode, qui fait allusion aux jeux de hasard pratiqués dans les casinos de Monte-Carlo, a été inventé en 1947 par Nicholas Metropolis et publié pour la première fois en 1949 dans un article co-écrit avec Stanislas Ulam.
2. C'est-à-dire avec une probabilité uniforme.

2. On considère à présent l'application linéaire *dérivée* :

$$\begin{array}{rccc} \mathcal{D} & : & \mathbb{R}_n[X] & \to & \mathbb{R}_n[X] \\ & & P & \mapsto & P', \end{array}$$

associant à tout polynôme $P(x) = \sum_{k=1}^{n} a_k\, x^k$ sa dérivée $P'(x) = \sum_{k=1}^{n} k\, a_k\, x^k$. Réprendre la question précédente avec $\mathcal{D}$ en place de $\mathcal{R}$.

3. On considère enfin les deux applications composées $\mathcal{D}\circ\mathcal{R}$ et $\mathcal{R}\circ\mathcal{D}$. Que font-elles ? Sont-elles identiques ? Quel est leur lien avec les matrices $M_{\mathcal{D}}M_{\mathcal{R}}$ et $M_{\mathcal{R}}M_{\mathcal{D}}$?

Exercice 3 $\diamond$ **(procédé d'orthonormalisation de Gram[3]–Schmidt[4]).** On rappelle que, partant d'une famille $\mathcal{B} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ de vecteurs linéairement indépendants de $\mathbb{R}^n$, avec m et n des entiers tels que $2 \leq m \leq n$, le *procédé d'orthonormalisation de Gram–Schmidt* permet de construire une famille $\mathcal{B}' = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$ de vecteur orthonormaux donnés par :

$$\boldsymbol{q}_1 = \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|},$$

$$\widetilde{\boldsymbol{q}}_{k+1} = \boldsymbol{x}_{k+1} - \sum_{i=1}^{k} (\boldsymbol{x}_{k+1}, \boldsymbol{q}_i)\, \boldsymbol{q}_i,\ \boldsymbol{q}_{k+1} = \frac{\widetilde{\boldsymbol{q}}_{k+1}}{\|\widetilde{\boldsymbol{q}}_{k+1}\|},\ k = 1, \ldots, m-1.$$

1. Écrire une fonction nommée `gramschmidt`, prenant comme paramètre d'entrée une matrice ayant pour colonnes les m vecteurs de la famille $\mathcal{B}$ et retournant en sortie une matrice ayant pour colonnes les m vecteurs de la famille $\mathcal{B}'$, obtenue en appliquant à $\mathcal{B}$ le procédé d'orthonormalisation de Gram–Schmidt. Penser à inclure une procédure vérifiant que la famille $\mathcal{B}$ fournie lors de l'appel de la fonction est bien libre.
2. On pose $\varepsilon = 10^{-8}$. Tester la fonction `gramschmidt` avec la famille

$$\mathcal{B} = \left\{ \begin{pmatrix} 1 \\ \varepsilon \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ \varepsilon \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ \varepsilon \end{pmatrix} \right\},$$

puis vérifier que les vecteurs obtenus sont bien orthogonaux deux à deux. Que constate-t-on ?

3. Pour pallier aux défauts d'orthogonalité observés des vecteurs de la famille $\mathcal{B}'$ (dûs aux erreurs d'arrondi en arithmétique à virgule flottante), on utilise en pratique une version plus stable de cette méthode, appelée *procédé de Gram–Schmidt modifié*. On procède alors comme suit :

$$q_1 = \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|},$$

$$\boldsymbol{q}_{k+1}^{(0)} = \boldsymbol{x}_{k+1},\ \boldsymbol{q}_{k+1}^{(i)} = \boldsymbol{q}_{k+1}^{(i-1)} - \left(\boldsymbol{q}_{k+1}^{(i-1)}, \boldsymbol{q}_i\right) \boldsymbol{q}_i,\ i = 1, \ldots, k,\ \boldsymbol{q}_{k+1} = \frac{\boldsymbol{q}_{k+1}^{(k)}}{\|\boldsymbol{q}_{k+1}^{(k)}\|},\ k = 1, \ldots, m-1.$$

Implémenter, en modifiant la fonction déjà existante, cette variante pour obtenir une nouvelle fonction qu'on nommera `modgramschmidt`. Reprendre alors l'orthonormalisation de la famille $\mathcal{B}$ donnée dans la question précédente.

---

3. Jørgen Pedersen Gram (27 juin 1850 - 29 avril 1916) était un actuaire et mathématicien danois. Il fit d'importantes contributions dans les domaines des probabilités, de la l'analyse numérique et de la théorie des nombres. Le procédé qui porte aujourd'hui son nom fut publié en 1883 dans un article intitulé *On series expansions determined by the methods of least squares.*

4. Erhard Schmidt (13 janvier 1876 - 6 décembre 1959) était un mathématicien allemand. Il est considéré comme l'un des fondateurs de l'analyse fonctionnelle abstraite moderne.

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séance 5 : résolution numérique de systèmes linéaires

Le symbole ⋄ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Avant de commencer, télécharger l'archive contenant les fichiers nécessaires à la séance de travaux pratiques à l'adresse

`http://www.ceremade.dauphine.fr/~legendre/enseignement/tp/tpsystemes.tgz`

puis extraire les fichiers en question.

Exercice 1 (utilisation des méthodes de Gauss [1]–Seidel [2] et de Jacobi [3], *d'après A. Quarteroni*). Soient n un entier naturel non nul et ε un réel appartenant à l'intervalle $[0,1]$. On considère le système linéaire $A_\epsilon \boldsymbol{x} = \boldsymbol{b}_\epsilon$ d'ordre n, où

$$A_\varepsilon = \begin{pmatrix} 1 & \varepsilon & \varepsilon^2 & 0 & \cdots & 0 \\ \varepsilon & 1 & \varepsilon & \ddots & \ddots & \vdots \\ \varepsilon^2 & \varepsilon & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \varepsilon^2 \\ \vdots & \ddots & \ddots & \ddots & 1 & \varepsilon \\ 0 & \cdots & 0 & \varepsilon^2 & \varepsilon & 1 \end{pmatrix} \text{ et } \boldsymbol{b}_\varepsilon = A_\varepsilon \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

La commande `[A,b]=matrix(n,epsilon)`, faisant appel à une fonction présente dans l'archive téléchargée, permet de construire les tableaux associés à la matrice A_ε et au vecteur $\boldsymbol{b}_\varepsilon$ pour des valeurs de l'entier n et du réel ε données. On pose dans un premier temps $n = 5$.

1. On sait que si la matrice A_ε est *à diagonale strictement dominante*, alors la méthode de Gauss–Seidel, appliquée à la résolution du système linéaire $A_\epsilon \boldsymbol{x} = \boldsymbol{b}_\epsilon$, converge.
   a. Vérifier que A_ε est à diagonale strictement dominante lorsque l'on fixe $\varepsilon = 0,3$.
   b. Après avoir complété et renommé le fichier `methiter.m` de l'archive pour obtenir une fonction nommée `gaussseidel` implémentant la méthode de Gauss–Seidel, calculer une solution approchée du système, avec une tolérance pour le critère d'arrêt égale à 10^{-10} et le vecteur $\boldsymbol{x}^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$ pour initialisation. Noter le nombre d'itérations nécessaires pour obtenir cette solution.
   c. De la même manière, compléter et renommer le fichier `methiter.m` pour obtenir une fonction nommée `jacobi` implémentant la méthode de Jacobi.

1. Johann Carl Friedrich Gauß (30 avril 1777 - 23 février 1855) était un mathématicien, astronome et physicien allemand. Surnommé par ses pairs *« le prince des mathématiciens »*, il étudia tous les domaines des mathématiques et contribua à développer la plupart des branches des sciences.

2. Philipp Ludwig von Seidel (24 octobre 1821 - 13 août 1896) était un mathématicien, physicien de l'optique et astronome allemand. Il a étudié l'aberration optique en astronomie en la décomposant en cinq phénomènes constitutifs, appelés *« les cinq aberrations de Seidel »*, et reste aussi connu pour la méthode de résolution numérique de systèmes linéaires portant son nom.

3. Carl Gustav Jacob Jacobi (10 décembre 1804 - 18 février 1851) était un mathématicien allemand. Ses travaux portèrent essentiellement sur l'étude des fonctions elliptiques, les équations différentielles et aux dérivées partielles, les systèmes d'équations linéaires, la théorie des déterminants. Un très grand nombre de résultats d'algèbre et d'analyse portent ou utilisent son nom.

2. a. Tracer le graphe des valeurs des rayons spectraux respectifs des matrices d'itération des méthodes de Jacobi et de Gauss–Seidel associées à A_ε en fonction de celle du paramètre ε, pour $\varepsilon = 0,\ 0,1,\ 0,2,\ldots,\ 1$.
   b. Que dire de la convergence des deux méthodes en fonction de la valeur de ε ?
   c. Quelle méthode choisir pour résoudre le système dans le cas où $\varepsilon = 0,5$? Utiliser la méthode sélectionnée et comparer le nombre d'itérations nécessaires à celui observé en 2.b.. Quelle explication donner à la différence constatée ?
3. On pose à présent $n = 100$. Pour $\varepsilon = 0,3$ et $\varepsilon = 0,35$, tracer (en utilisant la commande `semilogy`) et comparer les graphes de la norme du résidu $\boldsymbol{r}^{(k)} = \boldsymbol{b}_\varepsilon - A_\varepsilon \boldsymbol{x}^{(k)}$ en fonction du numéro de l'itération $1 \leq k \leq 50$ pour la méthode de Jacobi. Commenter en particulier la pente des courbes. Dans le cas où $\varepsilon = 0,35$, estimer à partir du graphe le nombre d'itérations nécessaires pour que la norme du résidu soit plus petite que 10^{-10}.

Exercice 2 (influence des erreurs d'arrondi). Le but de cet exercice est de mettre en évidence les problèmes, liés aux erreurs d'arrondi dans les calculs, apparaissant dans certaines méthodes numériques, notamment pour la résolution de systèmes linéaires. On considère pour cela H, la matrice de Hilbert d'ordre n (obtenue en entrant la commande `H=hilb(n)`).

1. Poser $n = 10$ et choisir un vecteur non nul $\boldsymbol{x}_0$ de $\mathbb{R}^n$ (par exemple en entrant la commande `x0=ones(10,1)`). Calculer ensuite le vecteur $\boldsymbol{b} = H\boldsymbol{x}_0$.
2. Résoudre alors le système $H\boldsymbol{x} = \boldsymbol{b}$ avec MATLAB. Que constate-t-on ? Comparer précisément la solution obtenue $\boldsymbol{x}$ avec $\boldsymbol{x}_0$ en calculant la quantité
$$\frac{\|\boldsymbol{x} - \boldsymbol{x}_0\|_2}{\|\boldsymbol{x}_0\|_2}$$
appelée *erreur relative* sur la valeur de $\boldsymbol{x}$.
3. L'erreur relative dépend-t-elle fortement du choix du vecteur $\boldsymbol{x}_0$?
4. Écrire une fonction nommée `errrel` prenant comme argument d'entrée l'ordre n de la matrice de Hilbert et renvoie la valeur correspondante de l'erreur relative. Tracer ensuite son graphe.

Exercice 3 ⋄ (analyse de la perturbation des données). On cherche à résoudre le système linéaire $A\boldsymbol{x} = \boldsymbol{b}$, où
$$A = \begin{pmatrix} 1 & 0,1 & 0,01 & 0,001 & 0,0001 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1,5 & 2.25 & 3,375 & 5,0625 \\ 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 9 & 27 & 81 \end{pmatrix} \text{ et } \boldsymbol{b} = \begin{pmatrix} 0,01 \\ 1 \\ 2,25 \\ 4 \\ 9 \end{pmatrix}.$$

1. Écrire une fonction nommée `gauss`, prenant comme arguments d'entrée la matrice A et le vecteur $\boldsymbol{b}$, implémentant la méthode de Gauss pour la résolution du système $A\boldsymbol{x} = \boldsymbol{b}$. Elle renverra en sortie le vecteur solution $\boldsymbol{x}$.
2. Utiliser cette fonction pour calculer la solution $\boldsymbol{y}$ du système perturbé $(A + \delta A)\boldsymbol{y} = \boldsymbol{b}$, où δA est une matrice définie aléatoirement au moyen de la commande `random`, et évaluer la quantité
$$\frac{\|\boldsymbol{y} - \boldsymbol{x}\|_2}{\|\boldsymbol{y}\|_2} \frac{\|A\|_2}{\|\delta A\|_2}.$$
3. Utiliser cette fonction pour calculer la solution z du système perturbé $A\boldsymbol{z} = \boldsymbol{b} + \delta\boldsymbol{b}$, où $\delta\boldsymbol{b}$ est un vecteur défini aléatoirement, et évaluer la quantité
$$\frac{\|\boldsymbol{z} - \boldsymbol{x}\|_2}{\|\boldsymbol{z}\|_2} \frac{\|\boldsymbol{b}\|_2}{\|\delta\boldsymbol{b}\|_2}.$$
4. Soit D la matrice diagonale obtenue à partir de la matrice A en entrant la commande `D=diag(diag(A))`. On pose $B = D^{-1}A$, $\boldsymbol{c} = D^{-1}\boldsymbol{b}$, et on considère le système linéaire $B\boldsymbol{x} = \boldsymbol{c}$. Reprendre les questions 2. et 3. avec $\delta B = D^{-1}\delta A$ et $\delta\boldsymbol{c} = D^{-1}\delta\boldsymbol{b}$. Qu'en conclure ?

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séance 6 : calcul de valeurs et vecteurs propres

Le symbole $\diamond$ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Exercice 1 (méthode de la puissance). La *méthode de la puissance* est une méthode itérative très simple de calcul de la[1] valeur propre de plus grand module d'une matrice (on parle de valeur propre *dominante*) et d'un vecteur propre associé.

Soit A une matrice d'ordre n que l'on suppose diagonalisable. On note $(\lambda_i)_{1\leq i\leq n}$ ses valeurs propres répétées avec leurs multiplicités respectives et ordonnées par ordre croissant de leurs modules, c'est-à-dire que

$$|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_n|,$$

et $(\boldsymbol{u}_i)_{1\leq i\leq n}$ une base de vecteurs propres associés. On suppose de plus que λ_n a une multiplicité algébrique égale à un. Dans toute la suite, on note $\|\cdot\|_2$ la norme euclidienne sur $\mathbb{C}^n$.

La méthode de la puissance pour calculer λ_n se résume dans l'algorithme suivant

- **Initialisation :**
  choisir $\boldsymbol{v}^{(0)} \in \mathbb{C}^n$ tel que $\|\boldsymbol{v}^{(0)}\|_2 = 1$,
- **Itérations :**
  pour $k \geq 1$
  $\boldsymbol{z}^{(k)} = A\boldsymbol{v}^{(k-1)}$,
  $\boldsymbol{v}^{(k)} = \dfrac{\boldsymbol{z}^{(k)}}{\|\boldsymbol{z}^{(k)}\|_2}$,
  $\nu^{(k)} = (\boldsymbol{v}^{(k)})^* A\boldsymbol{v}^{(k)}$.

On a alors le résultat suivant.

Théorème 1 *On suppose que, dans la base des vecteurs propres, le vecteur initial $\boldsymbol{v}^{(0)}$ s'écrit $\boldsymbol{v}^{(0)} = \sum_{i=1}^n \alpha_i \boldsymbol{u}_i$ avec $\alpha_n \neq 0$. Alors il existe une constante $C > 0$ telle que*

$$\|\tilde{\boldsymbol{v}}^{(k)} - \boldsymbol{u}_n\|_2 \leq C \left|\frac{\lambda_{n-1}}{\lambda_n}\right|^k, \ k \geq 1,$$

où $\tilde{\boldsymbol{v}}^{(k)} = \dfrac{\|A^k\boldsymbol{v}^{(0)}\|_2 \boldsymbol{v}^{(k)}}{\alpha_n {\lambda_n}^k} = \boldsymbol{u}_n + \displaystyle\sum_{i=1}^{n-1} \frac{\alpha_i}{\alpha_n}\left(\frac{\lambda_i}{\lambda_n}\right)^k \boldsymbol{u}_i.$

L'estimation ci-dessus exprime la convergence de la suite $\left(\tilde{\boldsymbol{v}}^{(k)}\right)_{k\in\mathbb{N}}$ (et donc de $\left(\boldsymbol{v}^{(k)}\right)_{k\in\mathbb{N}}$) vers $\boldsymbol{u}_n$, la suite $\nu^{(k)}$ convergeant alors vers λ_n, d'autant plus rapidement que le quotient $|\lambda_{n-1}/\lambda_n|$ est petit.

1. Proposer un critère d'arrêt pour les itérations de l'algorithme de la méthode de la puissance introduit plus haut.

1. On ne considèrera pas ici de matrices possédant plusieurs valeurs propres dominantes et pour lesquelles un traitement spécifique est requis.

2. Écrire une fonction `[lambda,u,iter]=puissance(A,v0,tol,nmax)` implémentant l'algorithme ainsi obtenu, les paramètres d'entrée `tol` et `nmax` représentant respectivement la tolérance pour le critère d'arrêt et le nombre maximum d'itérations, les paramètres en sortie `lambda`, `u` et `iter` contenant respectivement la suite des valeurs $\nu^{(k)}$, l'approximation finale du vecteur propre associé à la valeur propre dominante et le nombre d'itérations nécessaires à la convergence de l'algorithme.
3. On considère la matrice
$$A = \begin{pmatrix} 15 & -2 & 2 \\ 1 & 10 & -3 \\ -2 & 1 & 0 \end{pmatrix}.$$
   a. Utiliser la fonction `puissance` pour rechercher la valeur propre dominante de A ainsi qu'un vecteur propre associé, en choisissant $\boldsymbol{v}^{(0)} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T/\sqrt{3}$ comme vecteur initial et en prenant une tolérance égale à 10^{-8}.
   b. Valider le résultat obtenu en utilisant la commande `eig`.
4. On souhaite à présent évaluer l'influence de la condition initiale sur la convergence de la méthode sur un exemple. On considère pour cela la matrice réelle symétrique suivante
$$B = \begin{pmatrix} 0,5172 & 0,5473 & -1,224 & 0,8012 \\ 0,5473 & 1,388 & 1,353 & -1,112 \\ -1,224 & 1,353 & 0,03642 & 2,893 \\ 0,8012 & -1,112 & 2,893 & 0,05827 \end{pmatrix}.$$
   a. Pour chacune des conditions initiales suivantes, tracer sur un même graphe les termes de la suite $\|\boldsymbol{z}^{(k)}\|_2$ en fonction de k : $\boldsymbol{v}^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T$, $\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}^T/\sqrt{4}$ et $\begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}^T/\sqrt{2}$.
   b. Utiliser la commande `eig` pour obtenir les valeurs propres de B. Commenter alors les trois courbes obtenues à la question précédente en tentant de donner des explications.

Exercice 2 ◇ (méthode de la puissance inverse). La *méthode de la puissance inverse* permet d'obtenir une approximation de la valeur propre d'une matrice A la *plus proche* d'un nombre $\mu \in \mathbb{C}$ donné n'appartenant pas au spectre de A, en appliquant la méthode de la puissance à la matrice ${M_\mu}^{-1} = (A-\mu I)^{-1}$. Bien que plus coûteuse que la méthode de la puissance (elle nécessite en effet d'inverser un système linéaire à chaque itération de l'algorithme [2]), la méthode de la puissance inverse a l'avantage de pouvoir converger vers n'importe quelle valeur propre de A et se prête donc bien au raffinement d'une approximation d'une valeur propre, obtenue par exemple par une technique de localisation (comme celles basées sur les disques de Gershgorin [3]).

1. Sur le modèle de la fonction `puissance` de l'exercice 1, écrire une fonction `[lambda,u,iter]=puissanceinv(A,v0,mu,tol,nmax)` implémentant la méthode de la puissance inverse, le paramètre d'entrée `mu` étant l'approximation initiale de la valeur propre que l'on souhaite approcher. On utilisera la commande `lu` pour calculer la factorisation LU de M_μ et, pour résoudre les systèmes triangulaires, les fonctions `forwardcol` et `backwardcol` fournies dans l'archive disponible à l'adresse suivante : `http://www.ceremade.dauphine.fr/~legendre/enseignement/tp/tpeigenvalues.tgz`.
2. Utiliser la fonction `puissanceinv` pour calculer la valeur propre de plus petit module de la matrice A de l'exercice précédent, avec $\boldsymbol{v}^{(0)} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T/\sqrt{3}$ et une tolérance égale à 10^{-8}.

2. En pratique, on calcule une fois pour toute la factorisation LU de M_μ, ce qui permet de n'avoir à résoudre à chaque itération que deux systèmes triangulaires.

3. Semyon Aranovitch Gershgorin (Семён Аранович Гершгорин, 24 août 1901 - 30 mai 1933) était un mathématicien biélorusse (soviétique) qui travailla en algèbre et en théorie des fonctions d'une variable complexe. Dans son article *Über die Abgrenzung der Eigenwerte einer Matrix* publié en 1931, il donna des estimations permettant de localiser dans le plan complexe les valeurs propres d'une matrice carrée.

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séances 7 et 8 : résolution d'équations non linéaires

Le symbole ⋄ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Avant de commencer, télécharger l'archive contenant les fichiers nécessaires à la séance de travaux pratiques à l'adresse

`http://www.ceremade.dauphine.fr/~legendre/enseignement/tp/tpnonlineaire.tgz`

puis extraire les fichiers en question.

Exercice 1 (méthodes de dichotomie et de Newton[1]–Raphson[2], *d'après A. Quarteroni*). Dans cet exercice, on souhaite utiliser sur des exemples les différentes méthodes d'approximation d'un zéro d'une fonction vues en cours.

1. On considère la fonction $f(x) = \frac{x}{2} - \sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2}$ sur l'intervalle $\left[-\frac{\pi}{2}, \pi\right]$, en observant qu'elle y possède deux zéros.
   a. Définir une fonction *inline* pour f en entrant la commande : `f=inline('x/2-sin(x)+pi/6 -sqrt(3)/2','x')`.
   b. À l'aide d'un graphe de la fonction f, expliquer pourquoi la méthode de dichotomie ne peut être utilisée que pour calculer l'un de ces deux zéros, que l'on notera ξ dans la suite.
   c. Compléter les lignes manquantes du fichier `dichotomie.m` afin d'obtenir une fonction `[zero,iter,res,inc]=dichotomie(f,a,b,tol,nmax)` implémentant la méthode de dichotomie pour l'approximation un zéro de la fonction f compris dans un intervalle $[a, b]$ tel que $f(a)f(b) < 0$. Les autres paramètres d'entrée `tol` et `nmax` représentent respectivement la tolérance pour le critère d'arrêt et le nombre maximum d'itérations, les paramètres de sortie `zero`, `iter`, `res` et `inc` étant pour leur part l'approximation du zéro obtenue, le nombre d'itérations nécessaire au calcul de cette approximation, la valeur de la fonction `func` en ce point et un vecteur contenant la suite des valeurs absolues des différences entre deux approximations successives (dite suite des incréments).
   d. Utiliser la fonction `dichotomie` pour calculer une approximation de ξ avec une tolérance égale à 10^{-10} pour le critère d'arrêt et en choisissant un intervalle $[a, b]$ convenable.
   e. Au moyen de la commande `semilogy`, tracer le graphe de la suite des incréments $|x^{(k+1)} - x^{(k)}|$ avec une échelle semilogarithmique et déterminer la loi selon laquelle ces quantités tendent vers 0 quand k tend vers l'infini.
   f. Compléter les lignes manquantes du fichier `newton.m` afin d'obtenir une fonction `[zero, iter,res,inc]=newton(f,df,x0,tol,nmax)` implémentant la méthode de Newton pour l'approximation un zéro de la fonction f. Les paramètres d'entrée `df`, `x0`, `tol` et `nmax` représentent respectivement le nom de la fonction *inline* correspondant à f', le point de départ de la méthode, la tolérance pour le critère d'arrêt et le nombre maximum d'itérations. En sortie, les paramètres sont identiques à ceux de la fonction `dichotomie`.

1. Sir Isaac Newton (4 janvier 1643 - 31 mars 1727) était un philosophe, mathématicien, physicien et astronome anglais. Figure emblématique des sciences, il est surtout reconnu pour sa théorie de la gravitation universelle et la création du calcul infinitésimal.

2. Joseph Raphson (v. 1648 - v. 1715) était un mathématicien anglais. Son travail le plus notable est son *Analysis Aequationum Universalis*, publié en 1690 et contenant une méthode pour l'approximation d'un zéro d'une fonction d'une variable réelle à valeurs réelles.

g. Calculer des approximations des deux zéros ξ et ζ de la fonction f avec la méthode de Newton, en prenant une tolérance égale à 10^{-10} pour le critère d'arrêt et comme point de départ π pour ξ et $-\frac{\pi}{2}$ pour ζ. Comparer les nombres d'itérations effectuées pour chacun des zéros. Pourquoi sont-ils très différents ? Comparer également les graphes des suites des incréments obtenus avec la commande `semilogy`.

h. On cherche à réduire le nombre d'itérations nécessaires pour obtenir une approximation du zéro négatif ζ de la fonction f. La *méthode de Newton modifiée*, basée sur la modification suivante de la relation de récurrence de l'algorithme

$$x^{(k+1)} = x^{(k)} - 2\frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)},$$

a une convergence quadratique si $f'(\zeta) = 0$. Programmer cette méthode dans une fonction `modnewton` et observer combien d'itérations sont nécessaires pour obtenir une approximation de ζ avec une tolérance égale à 10^{-10} pour le critère d'arrêt.

2. On considère à présent la fonction $g(x) = x + e^{-20\,x^2}\cos(x)$, dont on veut approcher les zéros par la méthode de Newton.

   a. Définir une première fonction *inline* pour g en entrant la commande : `g=inline('x +exp(-20*x.^2).*cos(x)','x')`, puis une seconde pour sa dérivée g'.

   b. Utiliser la fonction `newton` pour essayer de calculer une approximation d'un zéro de g en prenant $x^{(0)} = 0$ pour initialisation et une tolérance égale à 10^{-10} pour le critère d'arrêt.

   c. Tracer le graphe de g sur l'intervalle $[-1, 1]$ et tenter de donner une explication qualitative du fait la méthode de Newton ne converge pas avec $x^{(0)} = 0$ comme point de départ.

   d. Appliquer cinq intérations de la méthode de dichotomie à la fonction g sur l'intervalle $[-1, 1]$ pour obtenir une initialisation convenable pour la méthode de Newton et relancer la recherche à partir de ce point.

3. Renommer et modifier le fichier `dichotomie.m` pour obtenir, sur le même modèle, une fonction `regulafalsi` implémentant la *méthode de la fausse position*[3]. De la même manière, écrire une fonction qui implémente la *méthode de la sécante*[4] à partir du fichier `newton.m`.

Exercice 2 (calcul de $\sqrt{2}$). Dans cet exercice, on cherche à calculer une approximation de $\sqrt{2}$ de diverses façons.

1. On peut tout d'abord obtenir une valeur approchée de $\sqrt{2}$ en cherchant la racine positive du polynôme $f(x) = x^2 - 2$. Appliquer successivement à f les méthodes de dichotomie, de la fausse position, de Newton et de la sécante sur l'intervalle $[1, 2]$ et déterminer celles qui convergent.

2. On peut également se servir de *méthodes de point fixe*, définies à partir des applications suivantes :

$$g_1(x) = 2 + x - x^2,\ g_2(x) = \frac{2}{x} \text{ et } g_3(x) = \frac{x+2}{x+1}.$$

   a. Parmi les trois fonctions ci-dessus, lesquelles conduisent à une méthode de point fixe convergente ?

3. On rappelle que la méthode de la fausse position est un algorithme de recherche d'un zéro d'une fonction f qui combine les possibilités de la méthode de dichotomie et de la méthode de la sécante. Ceci se traduit par le fait que, à l'itération k, le point intermédiaire c_k n'est pas défini comme étant le milieu de l'intervalle $[a_k, b_k]$ mais comme l'abscisse de l'intersection de la droite passant par les points $(a_k, f(a_k))$ et $(b_k, f(b_k))$ avec l'axe des abscisses, c'est-à-dire

$$c_k = a_k - \frac{a_k - b_k}{f(a_k) - f(b_k)} f(a_k).$$

4. On rappelle que la méthode de la sécante est une méthode dérivée de celle de Newton, dans laquelle la quantité $f'\left(x^{(k)}\right)$ apparaissant dans la relation de récurrence est remplacée par le quotient $\frac{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}{x^{(k)} - x^{(k)-1}}$. L'initialisation de cette méthode nécessite donc deux points $x^{(0)}$ et $x^{(1)}$ proches, si possible, du zéro recherché.

b. Vérifier ces affirmations en calculant les 20 premiers termes des suites définies par les relations de récurrence

$$x^{(k+1)} = g_i(x^{(k)}),\ k \in \mathbb{N},\ i \in \{1, 2, 3\}.$$

Exercice 3 ◇ (ordres de convergence). On considère les fonctions

$$f(x) = 3\cos(x) - 2\ln(x+1) - 1,\ g(x) = 2x - 1,\ h(x) = (2x-1)^3 \text{ et } k(x) = (2x-1)^5.$$

1. Calculer les dérivées de chacune de ces fonctions.
2. En utilisant les fonctions écrites à l'exercice 1, comparer le nombre d'itérations nécessaires aux méthodes de dichotomie, de la fausse position, de Newton et de la sécante pour obtenir le zéro, compris entre 0 et 1, de ces fonctions, en prenant un tolérance égale à 10^{-5}.
3. Pour chacune des fonctions et méthodes, représenter graphiquement les premiers termes de la suite $\left(\frac{\ln|x^{(n+1)}-\xi|}{\ln|x^{(n)}-\xi|}\right)_{n\in\mathbb{N}}$, où ξ est le zéro de la fonction considérée[5] et $\left(x^{(n)}\right)_{n\in\mathbb{N}}$ la suite des approximations de ξ produites par la méthode considérée. Déterminer sur ces graphes les ordres de convergence respectifs des méthodes.

Exercice 4 ◇ (bassins d'attraction de la méthode de Newton). On s'intéresse à la recherche des solutions complexes de l'équation $z^3 = 1$ par la méthode de Newton. On considère pour cela la fonction d'une variable complexe $f(z) = z^3 - 1$, qui s'annule en chaque point z du plan complexe tel que $z^3 = 1$.

1. Écrire deux fonctions *inline* `f` et `df` renvoyant respectivement les valeurs de f et de f' en un point quelconque de $\mathbb{C}$.
2. Pour tout entier $n \geq 2$, on définit une grille de pas $h = \frac{3}{n-1}$ couvrant le carré $[-1,5, 1,5] \times [-1,5\mathrm{i}, 1,5\mathrm{i}]$. Écrire une programme résolvant, pour une valeur donnée de n, l'équation $f(z) = 0$ par la méthode de Newton[6] initialisée successivement en chaque point de la grille $z_{ij} = -1,5(1+\mathrm{i}) + (i+j\mathrm{i})h$, $0 \leq i, j \leq n$, avec une tolérance égale à 10^{-4}. Pour chaque couple (i, j), stocker dans le tableau à deux dimensions `nrac` le numéro k ($k = 1$, 2 ou 3) de la racine cubique $e^{i\frac{2k\pi}{3}}$ vers laquelle la méthode aura convergée à partir de z_{ij} (en posant $k = 4$ lorsque la méthode n'a pas convergé après `nmax=100` itérations) et dans le tableau `niter` le nombre d'itérations nécessaires pour atteindre la convergence (en stockant le nombre maximal d'itérations autorisées `nmax` en l'absence de convergence).
   Pour automatiser le processus de reconnaissance de la racine approchée par la valeur `zero` retournée, on pourra utiliser les intructions suivantes (ci-dessous, `racines` désigne un tableau contenant les trois racines cubiques complexes de l'unité et `tol` est la tolérance du critère d'arrêt de la méthode de Newton) :

```
d=racines-[zero zero zero];
[m,k]=min(d);
if (abs(m)>tol)
   k=4;
end
```

3. À l'aide de la commande `imagesc`, afficher une représentation de chacun des tableaux `nrac` et `niter` obtenus pour $n = 100$.
4. Refaire des tracés pour des pas de grille plus petits (*i.e.*, de plus grandes valeurs de n). Que dire des « frontières » des trois bassins de convergence de la méthode ?

5. On pourra éventuellement obtenir une approximation de ξ en utilisant la commande `fsolve`.
6. On pourra pour cela utiliser la fonction `newton` écrite à l'exercice 1.

Université Paris-Dauphine
Département MIDO
DUMI2E deuxième année

Méthodes numériques
année 2009/2010
Guillaume Legendre

feuille de travaux pratiques

Séance 9 : interpolation polynomiale et intégration numérique

Le symbole ⋄ indique un exercice optionnel. Les notes biographiques sont pour partie tirées de WIKIPEDIA (`http://www.wikipedia.org/`). Le travail demandé peut être effectué indifféremment avec MATLAB ou le logiciel libre GNU OCTAVE (`http://www.gnu.org/software/octave/`).

Avant de commencer, télécharger l'archive contenant le fichier nécessaire à la séance de travaux pratiques à l'adresse
`http://www.ceremade.dauphine.fr/~legendre/enseignement/tp/tpinterpolation.tgz`
puis extraire le fichier en question.

Exercice 1 (interpolation de Lagrange [1], *d'après A. Quarteroni)*). Dans MATLAB et GNU OCTAVE, tout polynôme de degré $n \in \mathbb{N}$, $p(x) = a_0 + a_1\,x + a_2\,x^2 + \cdots + a_n\,x^n$, est représenté par un vecteur `p=[an,...,a2,a1,a0]` contenant [2] les $n+1$ coefficients de p. La commande `polyval(p,x)` (voir `help polyval`) permet d'évaluer n'importe quel polynôme p en un point x donné. La commande `polyfit(x,y,n)` (voir `help polyfit`) calcule les coefficients du polynôme de degré n interpolant (au sens des moindres carrés) les valeurs contenues dans le tableau `y` aux points donnés dans le tableau `x`. La fonction `p=lagrange(f,a,b,n)` de l'archive utilise la commande `polyfit` pour construire le polynôme d'interpolation de Lagrange de degré n d'une fonction f pour $n+1$ points équidistribués dans l'intervalle $[a,b]$.
On considère maintenant la fonction $\sin(x)$ sur l'intervalle $[0, 3\pi]$.

1. Utiliser la fonction `lagrange` pour calculer le polynôme d'interpolation de Lagrange $\Pi_n \sin$ pour une distribution uniforme de points dans l'intervalle $[0, 3\pi]$ et $n = 1, \ldots, 5$. Comparer graphiquement les polynômes obtenus avec la fonction donnée.
2. Évaluer l'erreur
$$E_n(\sin) = \max_{x\in[0,3\pi]} |\sin(x) - \Pi_n \sin(x)|$$
et tracer sur un graphe quelques valeurs de $E_n(\sin)$ en fonction de n.
3. En observant que $|\sin^{(n)}(x)| \leq 1$, $\forall n \in \mathbb{N}$ et $\forall x \in \mathbb{R}$, comparer les erreurs obtenues à la question précédente avec l'estimation théorique
$$E_n(f) \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n}\right)^{n+1} \max_{x\in[a,b]} |f^{(n+1)}(x)|.$$

Exercice 2 (phénomène de Runge [3] et points de Tchebychev [4]). On considère, sur l'intervalle $[-5, 5]$, la fonction
$$f(x) = \frac{1}{1+x^2}.$$

1. Joseph Louis Lagrange (Giuseppe Lodovico Lagrangia en italien, 25 janvier 1736 - 10 avril 1813) était un mathématicien et astronome franco-italien. Fondateur du calcul des variations avec Euler, il a également produit d'importantes contributions tant en analyse, qu'en géométrie, en théorie des groupes et en mécanique.
2. Attention : le premier élément du vecteur correspond au coefficient du terme de plus haut degré.
3. Carl David Tolmé Runge (30 août 1856 - 3 janvier 1927) était un mathématicien et physicien allemand. Il a co-développé la méthode de Runge-Kutta, qui est l'une des plus utilisées pour la résolution numérique d'équations différentielles.
4. Pafnouti Lvovitch Tchebychev (Пафну́тий Льво́вич Чебышёв, 4 mai 1821 - 26 novembre 1894) était un mathématicien russe. Il est connu pour ses travaux dans le domaine des probabilités et des statistiques.

1. Utiliser la fonction `lagrange` pour construire le polynôme d'interpolation de Lagrange $\Pi_n f$ de degré n, avec $n = 2, 4, 8$ et 12, et comparer graphiquement les polynômes obtenus avec la fonction donnée.
2. Évaluer l'erreur
$$E_n(f) = \max_{x\in[-5,5]} |f(x) - \Pi_n f(x)|$$
et tracer sur un graphe quelques valeurs de $E_n(f)$ en fonction de n. Qu'observe-t-on ?
3. Interpoler une fonction en des points équidistribués sur un intervalle $[a, b]$ n'est pas forcément le meilleur choix, comme le montre l'absence de convergence des interpolations de Lagrange constatée à la question précédente. Pour une interpolation de degré n, on peut montrer que l'erreur sera minimale si les points d'interpolation $(x_k)_{0\leq k\leq n}$ sont (à une transformation affine près) les racines du polynôme de Tchebychev T_{n+1}, c'est-à-dire si
$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k+1}{2(n+1)}\,\pi\right),\ \forall k \in \{0, \ldots, n\}.$$
   a. Modifier la fonction `lagrange` pour que l'interpolation se fasse aux *points de Tchebychev* définis ci-dessus.
   b. Reprendre alors les questions 1. et 2..

Exercice 3 (ordre de convergence de méthodes de quadrature). Soit f une fonction continue sur l'intervalle $[0, 1]$. Le but de cet exercice est d'illustrer l'efficacité des méthodes de quadrature classiques pour l'approximation de l'intégrale
$$I(f) = \int_0^1 f(x)\,\mathrm{d}x,$$
pour divers choix de fonctions, notamment l'effet l'influence de la régularité de la fonction sur la vitesse de convergence de la méthode.

1. Écrire trois fonctions ayant chacune pour paramètres d'entrée la fonction f et un nombre n de subdivisions de l'intervalle $[0, 1]$ et calculant une approximation $I_n(f)$ de $I(f)$ respectivement par les méthodes des rectangles, des trapèzes et de Simpson [5].
2. Appliquer ces fonctions au calcul de $I_n(f)$ pour $f(x) = e^x$ et diverses valeurs bien choisies de n.
3. Au moyen de la commande `semilogy`, tracer le graphe des courbes d'erreurs $|I(f) - I_n(f)|$ en fonction du nombre de sous-intervalles n et commenter les résultats.
4. Reprendre la question précédente avec $f(x) = |3x^4 - 1|$. Que constate-t-on ? Comment procéder pour retrouver les ordres de convergence théoriques des méthodes dans ce cas ?
Indication : décomposer l'intégrale en deux parties afin d'intégrer sur des intervalles sur lesquels la fonction est suffisament régulière.

5. Thomas Simpson (20 août 1710 - 14 mai 1761) était un inventeur et mathématicien anglais, connu principalement pour la méthode d'intégration numérique portant son nom.