Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

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

Le symbole $\diamond$ 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 « automatisée » 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 $(\boldsymbol{u}^{(k)})_{k\in\mathbb{N}}$ définie par

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

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

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

   c. Reprendre les questions précédentes 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). On rappelle que l'on peut définir une nouvelle fonction pour MATLAB ou GNU OCTAVE en l'implémentant, au moyen d'un simple éditeur de texte, dans un fichier ayant pour extension [4] `.m`. Un tel fichier de fonction débute par le mot-clé `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.

4. Ce type de fichier permet aussi d'enregistrer une séquence d'instructions à faire exécuter par le logiciel ; on parle dans ce cas de *fichier de script*.

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

Le nom de la fonction et le préfixe du nom du fichier la contenant doivent impérativement être identiques [5]. Ainsi, dans l'exemple ci-dessus, le fichier de fonction doit être nommé `mafonction.m`. La fonction elle-même est appelée depuis la ligne de commande, un fichier de script ou une autre fonction de la manière suivante :

```
[y_1,y_2,...]=mafonction(x_1,x_2,...)
```

On notera qu'il est possible d'afficher le code source d'un fichier de fonction préexistant grâce à 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 [6]. 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(4*x);
```

2. Tracer 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 ⋄ (manipulation des nombres complexes). Soit u et v les nombres complexes

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

5. Pour éviter d'éventuels conflits dûs à l'emploi de noms déjà affectés à des commandes de MATLAB ou de GNU OCTAVE, 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 (les logiciels étant sensibles à la casse).

6. 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 :

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

Créer des variables correspondant à nombres et calculer les modules et les arguments de u et de v, les produits $u\overline{v}$ et $\overline{u}v$, ainsi que les parties réelle et imaginaire de u^3+v^2 en utilisant les commandes appropriées.

Exercice 5 ◇ (formats d'affichage). MATLAB et GNU OCTAVE utilisent par défaut le format de représentation *double précision* de la norme IEEE 754 pour les calculs arithmétiques et toute variable est *a priori* stockée dans ce format. On peut afficher des (tableaux de) de valeurs numériques ou de chaînes de caractères en tapant simplement le nom des variables correspondantes ou en utilisant la commande `disp`, cette dernière se contentant de réaliser l'affichage sans écrire le nom de la variable concerné.

Il existe différents formats d'affichage prédéfinis, sélectionnables grâce à la commande `format`.

1. Tester ces formats[7] en exécutant la suite d'instructions suivantes :

```
x=pi^5;
disp(x) % affichage avec la notation par défaut
format long % notation à séparateur fixe et 15 chiffres significatifs
disp(x)
format short e % notation "scientifique" et 5 chiffres significatifs
disp(x)
format long e % notation "scientifique" et 15 chiffres significatifs
disp(x)
format short % retour à la notation par défaut
```

Il est aussi possible de contrôler très précisement la mise en forme de l'affichage avec la commande `fprintf`, qui permet de spécifier explicitement le format via une chaîne de caractères. Dans cette dernière, le format voulu pour chaque variable débute par le signe `%`, suivi de la *longueur minimal du champ* (c'est-à-dire le nombre minimal de caractères utilisés pour l'affichage) et d'un *caractère de conversion* (une lettre) indiquant que la valeur à afficher est un unique caractère (`%c`), une chaîne de caractères (`%s`), un entier signé en base 10 (`%d` ou `%i`), un entier non signé en base 10 (`%u`) ou un réel en notation à séparateur fixe (`%f`), en notation « scientifique » (`%e`) ou « compacte » (`%g`). Dans les derniers cas, on peut préciser le nombre maximal de caractères utilisés (pour `%s`), le nombre de chiffres situés à droite du séparateur (pour `%f` ou `%e`) ou le nombre de chiffres significatifs (pour `%g`), par un point suivi d'un chiffre. Indiquons que plusieurs autres opérateurs optionnels existent (`'`, `-`, `+` ou le caractère « espace »), ainsi que des caractères déchappement, tels que `\n` pour un saut de ligne , `\t` pour une tabulation horizontale, etc... Le tableau de variables à afficher donné en argument de la commande doit contenir au moins autant d'éléments qu'il y en a de prévu dans la chaîne de caractère donnant le format.

2. Tester les possibilités offertes par `fprintf` en exécutant les commandes suivantes puis en les modifiant :

```
fprintf('%c\n','abcdefghijklmnopqrstuvwxyz')
fprintf('%s\n','abcdefghijklmnopqrstuvwxyz')
fprintf('%.13s\n','abcdefghijklmnopqrstuvwxyz')
fprintf('%i %i %i\n',[-10:10])
fprintf('%+.2i %+.2i %+.2i\n',[-10:10])
fprintf('%.0f\t%.1f\t%.10f\n',pi,pi,pi)
fprintf('%.0e\t%.1e\t%.10e\n',pi,pi,pi)
fprintf('%.0g\t%.1g\t%.10g\n',pi,pi,pi)
```

3. Choisir une mise en forme et afficher le tableau de variables créé par la commande

7. Il faut y ajouter deux formats, un court (`format short g`) et un long (`format long g`), utilisant une notation « compacte », pour laquelle le logiciel choisit entre les notations à séparateur fixe et scientifique en fonction de la valeur à afficher.

Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séance 3 : boucles et récursivité

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 (suite de Fibonacci).

1. Écrire une boucle calculant les valeurs des vingt premiers termes de la suite de Fibonacci définie par
$$u^{(0)} = 0,\ u^{(1)} = 1 \text{ et } u^{(k+2)} = u^{(k+1)} + u^{(k)},\ \forall k \in \mathbb{N},$$
et conservant ces valeurs dans un vecteur.
2. Écrire une boucle calculant les termes successifs de la suite de Fibonacci dont la valeur est inférieure ou égale à 50000 et afficher le dernier de ces termes.
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 $(u^{(k)})_{k\in\mathbb{N}}$ et $(v^{(k)})_{k\in\mathbb{N}}$ par

$$u^{(0)} = 1,\ v^{(0)} = 2,\ u^{(k+1)} = \frac{u^{(k)} + v^{(k)}}{2},\ v^{(k+1)} = \sqrt{u^{(k+1)}v^{(k)}},\ \forall n \in \mathbb{N}.$$

On admet que ces suites sont adjacentes, de limite $\dfrac{\sqrt{27}}{\pi}$.

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

Exercice 3 ◇ (développements en série entière 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 la fonction cosinus en x obtenue en ne conservant que les n premiers termes du développement en série entière

$$\cos(x) = 1 - \frac{x^2}{2!} + \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + \ldots \tag{1}$$

Comparer les résultats de cette fonction avec ceux de la commande `cos(x)` pour différentes valeurs de n et de x. Écrire de la même manière une fonction `sinn(n,x)` basée sur le développement en série entière de la fonction sinus

$$\sin(x) = x - \frac{x^3}{3!} + \cdots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \ldots$$

1. Utiliser pour cela la fonction `input`.

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 une interruption 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, en utilisant la fonction `rem` donnant le reste de la division euclidienne de deux entiers, une fonction récursive `PGCD(a,b)` renvoyant le plus grand commun diviseur [2] des entiers naturels a et b calculé par l'*algorithme d'Euclide* [3].
3. Écrire une fonction récursive `rfibonacci(n)` calculant le $n^{\text{ième}}$ terme 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 `collatz(n)` renvoyant la valeur 1 si la conjecture de Collatz [4] est vérifiée pour l'entier $n > 0$.
5. Écrire une fonction récursive `rcosn(n,x)` de calcul d'une approximation de $\cos(x)$ vue dans l'exercice 3 utilisant la relation entre les termes de la somme (1), c'est-à-dire

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

Exercice 5 ⋄ (procédé Δ^2 d'Aitken). On peut obtenir une valeur approchée du réel π en sommant un nombre fini de termes de la *série de Madhava–Gregory–Leibniz*,

$$\sum_{k=0}^{+\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4}.$$

La convergence de cette série est malheureusement lente et, pour l'accélérer, on se propose d'utiliser la *procédé Δ^2 d'Aitken*. Cette technique consiste, à partir de la suite $(s^{(k)})_{k\in\mathbb{N}}$ des sommes partielles de la série de Madhava–Gregory–Leibniz, en la construction d'une suite $(u^{(k)})_{k\in\mathbb{N}}$ définie par

$$u^{(k)} = s^{(k)} - \frac{\left(s^{(k+1)} - s^{(k)}\right)^2}{s^{(k)} - 2\,s^{(k+1)} + s^{(k+2)}},$$

ayant même limite que $(s^{(k)})_{k\in\mathbb{N}}$ et convergeant plus rapidement.

1. Écrire une boucle calculant les termes de la suite $(s^{(k)})_{k\in\mathbb{N}}$ et s'arrêtant lorsque la condition $\left|s^{(k)} - \frac{\pi}{4}\right| \leq \varepsilon$ est vérifiée, avec ε un réel strictement positif fixé. En prenant $\varepsilon = 10^{-6}$, combien faut-il calculer de termes pour satisfaire le critère ?
2. Modifier la boucle de façon à calculer les termes de la suite $(u^{(k)})_{k\in\mathbb{N}}$. Pour quelle valeur de l'entier k a-t-on $\left|u^{(k)} - \frac{\pi}{4}\right| \leq \varepsilon$, avec $\varepsilon = 10^{-6}$?
3. Reprendre les questions précédentes avec $\varepsilon = 10^{-8}$.

---

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. On appelle *suite de Syracuse* toute 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 étant 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 Collatz* affirme que les suites de Syracuse de tous les nombres entiers strictement positifs atteignent le cycle trivial.

Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séance 4 : quelques premières applications du calcul scientifique

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 (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 de longueur égale à 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. Ces points pouvant être tirés avec la même probabilité dans l'ensemble du carré, le rapport entre le nombre de points tirés dans le disque et le nombre de points tirés au total tend, lorsque le nombre de tirages tend vers l'infini, vers le rapport des surfaces du cercle et du carré, soit $\frac{\pi}{4}$, en vertu de la loi des grands nombres.

1. Au moyen de la commande `rand`, qui génère une suite de nombres réels jouant le rôle d'une réalisation d'une suite de variables aléatoires continues, indépendantes et identiquement distribuées selon la loi uniforme sur 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 de deux décimales exactes 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 polynômes à coefficients réels 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 considère 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. En utilisant la commande `null`, déterminer le noyau de $\mathcal{R}$ pour $n = 6$, 7 et 8. Que constate-t-on ?

1. On appelle *méthode de Monte-Carlo* toute méthode visant à calculer une approximation numérique par utilisation d'un procédé aléatoire. Le nom de ce type de méthode, qui fait allusion aux jeux de hasard pratiqués dans le célèbre casino d'un des quartiers de la cité-État de la principauté de Monaco, 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 selon une loi de 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 de $\mathbb{R}_n[X]$ ayant pour coefficients a_k, $k = 0, \ldots, n$, le polynôme P' ayant pour fonction polynomiale $P'(x) = \sum_{k=1}^{n} k\, a_k\, x^{k-1}$. Reprendre 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 produits de matrices $M_{\mathcal{D}} M_{\mathcal{R}}$ et $M_{\mathcal{R}} M_{\mathcal{D}}$?

Exercice 3 (procédé d'orthonormalisation de Gram–Schmidt). 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 vecteurs 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 des vecteurs de la famille $\mathcal{B}'$ observés (qui sont dûs aux erreurs d'arrondi en arithmétique à virgule flottante), il faut utiliser une version plus stable de la méthode, appelée procédé de Gram–Schmidt modifié. On procède alors de la manière suivante :

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

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

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

Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séance 5 : résolution numérique de systèmes linéaires

Le symbole $\diamond$ 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/`).

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/tp_systemes.tgz`

puis extraire les fichiers en question.

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

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

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 par lignes, 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 par lignes lorsque l'on fixe $\varepsilon = 0,3$.
   b. Après avoir complété et renommé le fichier `methiter.m` contenu dans 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.
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 $H\boldsymbol{x} = \boldsymbol{b}$ en inversant la matrice H. 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-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 fonction `rand`, 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 $\boldsymbol{z}$ 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
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séance 6 : calcul de valeurs et vecteurs propres

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 (méthode de la puissance). La méthode de la puissance est une méthode itérative très simple fournissant des approximations 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 λ_i, $1 \leq i \leq n$ ses valeurs propres, comptées avec leurs multiplicités respectives et ordonnées de la manière suivante

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

et l'on désigne par $\{\boldsymbol{v}_i\}_{i=1,\ldots,n}$ une base de vecteurs propres associés. On suppose de plus que λ_n a une multiplicité algébrique égale à un et qu'elle est la seule valeur propre de plus grand module. Dans toute la suite, on note $\|\cdot\|_2$ la norme euclidienne sur $\mathbb{C}^n$.

La méthode de la puissance pour calculer λ_n consiste en l'algorithme suivant.

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

En supposant que le vecteur $\boldsymbol{q}^{(0)}$ n'est pas contenu dans le sous-espace engendré par les vecteurs propres associés aux valeurs propres autres que la valeur propre dominante λ_n, on montre que la suite des vecteurs $(\boldsymbol{q}^{(k)})_{k\in\mathbb{N}}$ et la suite des quotients de Rayleigh $(\nu^{(k)})_{k\in\mathbb{N}}$ convergent respectivement vers un vecteur colinéaire à $\boldsymbol{v}_n$ et vers λ_n lorsque k tend vers l'infini, la convergence étant d'autant plus rapide 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.
2. Écrire une fonction `[lambda,v,iter]=puissance(A,q0,tol,nmax)` implémentant l'algorithme de la méthode de la puissance ainsi obtenu, les paramètres d'entrée `tol` et `nmax` représentant respectivement la tolérance pour le critère d'arrêt et le nombre maximum d'itérations à effectuer, les paramètres en sortie `lambda`, `v` et `iter` contenant respectivement la suite des valeurs $\nu^{(k)}$ calculées, l'approximation d'un vecteur propre associé à la valeur propre dominante et le nombre d'itérations qui ont été nécessaires pour satisfaire le critère d'arrêt en cas de convergence.
3. On considère la matrice
$$A = \begin{pmatrix} 15 & -2 & 2 \\ 1 & 10 & -3 \\ -2 & 1 & 0 \end{pmatrix}.$$

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

a. Utiliser la fonction `puissance` pour rechercher la valeur propre dominante de A, ainsi qu'un vecteur propre associé, en prenant le vecteur $\boldsymbol{q}^{(0)} = \begin{pmatrix}1 & 1 & 1\end{pmatrix}^T/\sqrt{3}$ comme initialisation et une tolérance égale à 10^{-8} pour le critère d'arrêt.

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

$$B = \begin{pmatrix} 0,5172 & 0,5473 & -1,224 & 0,8012 \\ 0,5473 & 1,388 & 1,353 & -1,112 \\ -1,224 & 1,353 & 0,03642 & 2,893 \\ 0,8012 & -1,112 & 2,893 & 0,05827 \end{pmatrix}.$$

a. Pour chacune des trois initialisations qui suivent, tracer sur une même figure les termes de la suite $\|\boldsymbol{z}^{(k)}\|_2$ en fonction de k : $\boldsymbol{q}^{(0)} = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}^T$, $\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}^T/\sqrt{4}$ 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 de résoudre 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.

1. Sur le modèle de la fonction `puissance` de l'exercice 1, écrire une fonction `[lambda,v,iter]= puissanceinv(A,q0,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 la matrice M_μ et, pour résoudre les systèmes triangulaires associés à cette factorisation, les fonctions `forwardcol` et `backwardcol` fournies dans l'archive disponible à l'adresse suivante : `http://www.ceremade.dauphine.fr/~legendre/enseignement/tp/tp_eigenvalues.tgz`.
2. Utiliser la fonction `puissanceinv` pour calculer la valeur propre de plus petit module de la matrice A de l'exercice précédent, en prenant $\boldsymbol{q}^{(0)} = \begin{pmatrix}1 & 1 & 1\end{pmatrix}^T/\sqrt{3}$ comme initialisation et une tolérance égale à 10^{-8} pour le critère d'arrêt.

2. En pratique, on calcule une fois pour toute la factorisation LU de la matrice M_μ, ce qui permet de ne plus avoir ensuite qu'à résoudre deux systèmes triangulaires à chaque itération de la méthode.

Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séances 7 et 8 : résolution d'équations non linéaires

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/`).

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/tp_nonlineaire.tgz`

puis extraire les fichiers en question.

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

1. On considère tout d'abord 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 dite *inline* pour f en entrant la commande : `f=inline('x/2-sin(x)+pi/6-sqrt(3)/2','x')`.
   b. À l'aide du graphe de la fonction f sur $\left[-\frac{\pi}{2}, \pi\right]$, expliquer pourquoi la méthode de dichotomie ne peut être utilisée que pour approcher l'un des deux zéros de f, 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 d'un zéro d'une fonction f donnée, compris dans un intervalle $[a, b]$ tel que $f(a)f(b) < 0$. Les autres paramètres d'entrée `tol` et `nmax` de la fonction `dichotomie` sont respectivement la tolérance pour le critère d'arrêt de la méthode et le nombre maximum d'itérations à effectuer, 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 f 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 à partir du choix d'un intervalle $[a, b]$ convenable.
   e. Au moyen de la commande `semilogy`, tracer le graphe de la suite des incréments $|x^{(k+1)} - x^{(k)}|$ (en fonction de 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)` qui implémente la méthode de Newton–Raphson pour l'approximation d'un zéro d'une fonction dérivable f donnée. Les paramètres d'entrée `df`, `x0`, `tol` et `nmax` représentent respectivement le nom de la fonction *inline* correspondant à la fonction f', l'initialisation de la méthode, la tolérance pour le critère d'arrêt de la méthode et le nombre maximum d'itérations à effectuer. En sortie, les paramètres sont identiques à ceux de la fonction `dichotomie`.
   g. Calculer des approximations des deux zéros ξ et ζ de la fonction f avec la méthode de Newton–Raphson, en prenant une tolérance égale à 10^{-10} pour le critère d'arrêt et comme

initialisation le point π pour ξ et $-\frac{\pi}{2}$ pour ζ. Comparer les nombres d'itérations effectuées pour obtenir une approximation de 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 à l'obtention d'une approximation du zéro négatif ζ de la fonction f. La méthode de Newton–Raphson modifiée, basée sur la modification suivante de la relation de récurrence de la méthode de Newton–Raphson

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

a une convergence quadratique si $f'(\zeta) = 0$. Implémenter cette méthode dans une fonction `modnewton` et observer combien d'itérations sont nécessaires pour qu'elle fournisse 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–Raphson.

   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 d'approcher 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–Raphson ne converge pas avec l'initialisation précédente.

   d. Appliquer cinq intérations de la méthode de dichotomie à la fonction g sur l'intervalle $[-1, 1]$ et utiliser le point obtenu comme initialisation de la méthode de Newton–Raphson pour la recherche d'un zéro de g.

3. Renommer et modifier le fichier `dichotomie.m` pour obtenir, sur le même modèle, une fonction `regulafalsi` implémentant la méthode de la fausse position[1]. De la même manière, écrire une fonction qui implémente la méthode de la sécante[2] 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 $f(x) = x^2 - 2$. Pour cela, appliquer successivement à f les méthodes de dichotomie, de la fausse position, de Newton–Raphson 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

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

considérées sur l'intervalle $[1, 2]$.

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

1. On rappelle que, pour cette méthode, l'approximation du zéro à l'itération k $x^{(k)}$ est donnée par 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

$$x^{(k)} = a^{(k)} - \frac{a^{(k)} - b^{(k)}}{f(a^{(k)}) - f(b^{(k)})} f(a^{(k)}).$$

2. On rappelle que la méthode de la sécante peut être obtenue à partir de la méthode de Newton–Raphson en remplaçant la quantité $f'\left(x^{(k)}\right)$ apparaissant dans la relation de récurrence 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 distincts, $x^{(-1)}$ et $x^{(0)}$, si possible proches du zéro recherché.

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

$$x^{(0)} = \frac{1}{2},\ x^{(k+1)} = g_i(x^{(k)}),\ k \in \mathbb{N},\ i \in \{1, 2, 3\}.$$

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. Comparer le nombre d'itérations nécessaires aux méthodes de dichotomie, de la fausse position, de Newton–Raphson et de la sécante pour obtenir le zéro, compris entre 0 et 1, de ces fonctions, en prenant une tolérance égale à 10^{-5} pour les critères d'arrêt des méthodes.
3. Pour chacune des fonctions f, g et h et pour chacune des méthodes utilisées dans la question précédente, représenter graphiquement les premiers termes de la suite $\left(\frac{\ln|x^{(k+1)}-\xi|}{\ln|x^{(k)}-\xi|}\right)_{k\in\mathbb{N}}$ en fonction de k, où ξ est le zéro de la fonction considérée[3] et $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ est la suite des approximations de ξ produites par la méthode considérée. Déterminer à partir de ces graphes les ordres de convergence respectifs des méthodes.

Exercice 4 ◇ (bassins de convergence de la méthode de Newton–Raphson). On s'intéresse à la recherche des solutions complexes de l'équation $z^3 = 1$ par la méthode de Newton–Raphson. 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. Définir 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 $f(z) = 0$ avec une tolérance égale à 10^{-4} par la méthode de Newton–Raphson[4] initialisée successivement en chaque point de la grille $z_{ij} = -1,5(1+\mathrm{i}) + (i + j\mathrm{i})h$, $0 \leq i, j \leq n$. 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 complexe de l'unité $e^{i\frac{2k\pi}{3}}$ 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 instructions 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–Raphson) :

```
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 ?

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

Université Paris-Dauphine
Département MIDO
DE MI2E deuxième année

Méthodes numériques
année 2012/2013

feuille de travaux pratiques

Séance 9 : interpolation polynomiale et formules de quadrature

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 (interpolation de Lagrange à l'aide des matrices de Vandermonde). Dans MATLAB et GNU OCTAVE, toute fonction polynomiale de degré $n \geq 0$, $p(x) = a_0 + a_1\,x + a_2\,x^2 + \cdots + a_n\,x^n$, est représentée par un tableau `p=[an,...,a2,a1,a0]` contenant [1] les $n+1$ coefficients du polynôme qui lui est associé. La commande `polyval(p,x)` permet alors d'évaluer n'importe quelle fonction polynomiale p en un point (ou groupe de points) x donné. Enfin, la commande `vander(x)` renvoie la matrice de Vandermonde associée à un ensemble de nœuds contenu dans le tableau `x`.

1. Écrire une fonction `lagrange`, ayant pour arguments une fonction continue f, les bornes d'un intervalle $[a, b]$ et un entier positif n, construisant le polynôme d'interpolation de Lagrange $\Pi_n f$ associé à une distribution uniforme de $n+1$ points dans l'intervalle $[a, b]$ par résolution du système linéaire de Vandermonde correspondant.
2. Utiliser cette fonction pour construire les polynômes d'interpolation de Lagrange $\Pi_n \sin$ de la fonction sinus à nœuds équirépartis sur l'intervalle $[0, 3\pi]$, avec $n = 1, \ldots, 5$. Comparer les graphes des polynômes obtenus avec celui de la fonction donnée sur ce même intervalle.
3. Évaluer de manière approchée l'erreur
$$E_n(\sin) = \max_{x\in[0,3\pi]} |\sin(x) - \Pi_n \sin(x)|$$
et représenter sur un graphe les valeurs obtenues en fonction de n, pour $n = 1, \ldots, 12$. Que se passe-t-il pour $n > 12$?
4. En observant que $|\sin^{(n)}(x)| \leq 1$, $\forall n \in \mathbb{N}$ et $\forall x \in \mathbb{R}$, comparer les valeurs de l'erreur obtenues à la question précédente avec celles fournies par la majoration 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 et points de Tchebychev). On considère, sur l'intervalle $[-5, 5]$, la fonction
$$f(x) = \frac{1}{1+x^2}.$$

1. Utiliser la fonction `lagrange` de l'exercice précédent pour construire le polynôme d'interpolation de Lagrange $\Pi_n f$ de degré n, avec $n = 2, 4, 8$ et 12, de f en des nœuds équirépartis sur $[-5, 5]$ et comparer graphiquement les polynômes obtenus avec la fonction donnée.
2. Évaluer de manière approchée l'erreur
$$E_n(f) = \max_{x\in[-5,5]} |f(x) - \Pi_n f(x)|$$
et représenter sur un graphe les valeurs obtenues en fonction de n, pour $n = 1, \ldots, 12$. Qu'observe-t-on ?

1. On notera que le premier élément du vecteur correspond au coefficient du terme de plus haut degré.

Interpoler une fonction en des nœuds équidistribués sur un intervalle $[a, b]$ n'est pas forcément le meilleur choix, comme le montre l'absence de convergence de l'interpolation de Lagrange constatée à la question précédente. Pour une interpolation de degré n, on peut montrer que l'erreur sera minimale si les nœuds 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

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k+1}{2(n+1)}\,\pi\right),\ \forall k \in \{0, \ldots, n\}.$$

3. Modifier la fonction `lagrange` pour que l'interpolation soit faite aux points de Tchebychev définis ci-dessus.
4. Reprendre alors les questions 1 et 2.

Exercice 3 ◇ (ordre de convergence de formules de quadrature composées). Soit f une fonction réelle continue sur l'intervalle $[0, 1]$. Le but de cet exercice est de mesurer l'efficacité de formules de quadrature de Newton–Cotes composées classiques pour l'approximation de l'intégrale définie

$$I(f) = \int_a^b f(x)\,\mathrm{d}x,$$

pour différents choix de fonction, et 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, les bornes a et b, et un nombre $m \geq 1$ de subdivisions de l'intervalle $[a, b]$, calculant une approximation $I_{m,n}(f)$ de $I(f)$ respectivement par les formules de quadrature de la règle du point milieu (formule ouverte, $n = 0$), de la règle du trapèze (formule fermée, $n = 1$) et de la règle de Simpson (formule fermée, $n = 2$) composées.
2. Au moyen de la commande `semilogy`, tracer le graphe des courbes de l'erreur $|I(f) - I_{m,n}(f)|$ de chacune de ces trois formules en fonction du nombre de sous-intervalles pour $f(x) = e^x$, $a = 0$ et $b = 1$. Commenter les résultats.
3. Reprendre la question précédente pour $f(x) = \left|3\,x^4 - 1\right|$. Que constate-t-on ? Comment procéder pour retrouver les ordres de convergence théoriques des formules dans ce cas ?

Exercice 4 ◇ (méthode de Romberg). On considère l'utilisation de la règle du trapèze composée associée à une subdivision dyadique de l'intervalle $[a, b]$ pour le calcul de l'intégrale $I(f)$ de l'exercice précédent. En supposant la fonction f suffisamment régulière et en posant $H = \frac{b-a}{2^k}$, $k \geq 0$, on peut montrer à partir de la formule d'Euler–Maclaurin que

$$I_{2^k,1}(f) = I(f) + b_2\,H^2 + b_4\,H^4 + \ldots$$

À partir d'une estimation de l'intégrale pour une subdivision de taille $\frac{H}{2}$, on obtient, grâce au procédé d'extrapolation de Richardson, une meilleure approximation de $I(f)$, fournie par la quantité

$$R(k+1,1) = R(k+1,0) + \frac{1}{3}\,(R(k+1,0) - R(k,0)) = \frac{1}{3}\,(4\,R(k+1,0) - R(k,0)),$$

où l'on a posé $R(k,0) = I_{2^k,1}(f)$ et $R(k+1,0) = I_{2^{k+1},1}(f)$, et qui est d'ordre quatre en H. La méthode de Romberg pour l'évaluation approchée de $I(f)$ consiste simplement en l'application répétée de cette opération à partir de la valeur $k = 0$.

Écrire une fonction `romberg`, ayant pour arguments d'entrée une fonction f, les bornes a et b, un nombre d'extrapolations maximal N et une tolérance ε, renvoyant la valeur de l'approximation $R(k,k)$ telle que $|R(k,k) - R(k-1,k-1)| \leq \varepsilon$, avec $0 \leq k \leq N$, ou bien $k = N$. Pour cela, on construira un tableau des valeurs extrapolées $R(k,m)$, $0 \leq m \leq k \leq N$, dont les éléments satisfont la relation de récurrence

$$R(k,m) = \frac{1}{4^m - 1}\,(4^m\,R(k,m-1) - R(k-1,m-1)),\ 1 \leq m \leq k \leq N,$$

et la première colonne telle que $R(k,0) = I_{2^k,1}(f)$, $k = 0, \ldots, N$.