

SÉANCE D'INITIATION

1. Quelques rappels sur les tableaux

Exercice 1.1. afficher les vecteurs $[1^2, 2^2, \dots, 10^2]$, $[\sin(1), \dots, \sin(10)]$, $[\sin(1) - 1^3, \dots, \sin(10) - 10^3]$, et calculer la somme $\sum_{k=1}^{10} \sin(k) - k^3$.

Exercice 1.2. Calculer par la méthode des rectangle $\int_0^{\frac{\pi}{2}} \sin(x)dx$ en prenant 10, 100 puis 10000 points et comparez la à sa valeur exacte.

Exercice 1.3. Comparer avec la méthode des trapèze en utilisant l'instruction *trapz* :

```
>> n=10;
>> x=0 :pi/(2*n) :pi/2;
>> y=sin(x);
>> trapz(x,y)
>> (pi/(2*n))*trapz(y)
```

2. Quelques rappels sur les fonctions

Exercice 2.1. Quelles sont les différences entre les trois fonctions dont les fichiers .m sont les suivants :

```
function res = dudu(a)
res(1)=sqrt((a(1))^4+(a(2))^6)-a(3);
res(2)=a(1)*a(2)*a(3);
```

```
function [x,y] = dudu(a)
x=sqrt((a(1))^4+(a(2))^6)-a(3);
y=a(1)*a(2)*a(3);
```

```
function [x,y] = dudu(a1,a2,a3)
x=sqrt((a1)^4+(a2)^6)-a3;
y=a1*a2*a3;
```

Exercice 2.2. Quelle est la différence entre les deux fonctions suivantes :

```
function res = dudu1(x)
res=sin(x^2);
```

```
function res = dudu2(x)
res=sin(x.^2);
```

On pourra, après les avoir rentrées, essayer les commandes suivantes :

```
>> dudu1([1 2 3])
>> dudu2([1 2 3])
```

Que se passe-t-il et que peut-on en conclure ?

Exercice 2.3. Soit la fonction de \mathbb{R} dans \mathbb{R} , définie par morceaux par :

$$f(x) = \begin{cases} 0 & \text{si } x < 0, \\ 1 & \text{si } 0 \leq x \leq 2, \\ 2 & \text{si } 2 < x < 18, \\ 3 & \text{si } x \geq 18. \end{cases}$$

Écrire cette fonction en matlab de façon vectorielle, c'est-à-dire, de telle sorte que l'image d'un tableau soit le tableau des images.

Une autre possibilité permet de définir une fonction sans passer par un fichier (on réservera cette solution à une fonction simple et utilisée un nombre restreint de fois) en utilisant la fonction très pratique, la fonction inline, dont voici un exemple d'utilisation :

```
>> g=inline('x.^2+sin(x)');
```

Pour calculer $g(2)$, il suffira de taper

```
>> g(2);
```

Cette fonction est en mémoire vive (le vérifier en tapant *whos*) et le demeure tant qu'on ne fait pas *clear* et qu'on reste dans la même session de matlab. On pourra consulter l'aide à propos de cette fonction.

3. Les complexes

Matlab accepte les nombres complexes sous la forme $a+ib$ ou $\rho\exp(i\theta)$ (on peut aussi noter j à la place de i).

Toutes les opérations algébriques sur les complexes sont reconnues ($*, +, /$). On dispose aussi des fonctions *real*, *imag*, *abs*, *angle*, *conj*.

On peut éléver un complexe à une puissance réelle grâce à la fonction : n . On dispose aussi de *sqrt*, *log*, *exp*.

Toutes les opérations définies pour les matrices à coefficients réels demeurent valables pour les matrices à coefficients complexes.

attention : ne pas dans utiliser un compteur noté i dans un programme où l'on utilise des complexes ; dans ce cas, dans l'expression $3 + 5i$, la variable i désignera le nombre complexe i et dans l'expression $3 + 5 * i$, la variable i désignera le compteur i !

Exercice 3.1. Écrire une fonction qui calcule les n racines n -ème d'un nombre complexe.

4. Matlab et la récursivité

Matlab reconnaît la récursivité ; elle est à utiliser avec modération comme le montre l'exercice suivant :

Exercice 4.1. Déterminez la fonction factorielle en utilisant sa définition récursive :

$$\forall n \in \mathbb{N}, \quad n! = \begin{cases} n \times (n-1)! & \text{si } n \geq 1, \\ 1 & \text{si } n = 0. \end{cases}$$

Calculer, avec cette fonction, $0!$, $10!$, $510!$ (que se passe-t-il dans ce cas ?). Calculer $(1,5)!$, $(0,5)!$. Comment peut on palier le problème mis en évidence ?

Utiliser les fonctions de matlab *prod*, *factorial*, et *gamma* pour calculer $n!$. Comparer, en terme de temps et de nombres d'opérations. Qu'en conclure ?

Parfois, en revanche, seule la récursivité est utilisable, comme le montrent les deux exercices suivants :

Exercice 4.2. Déterminez la fonction de Mc Carthy en utilisant sa définition récursive :

$$\forall n \in \mathbb{N}, \quad f(n) = \begin{cases} n - 10 & \text{si } n > 100, \\ f(f(n + 11)) & \text{sinon.} \end{cases}$$

Calculer $f(n)$, pour $0 \leq n \leq 120$. Que constate-t-on ? Calculer, pour $0 \leq n \leq 120$, le nombre d'itérations nécessaires pour calculer $f(n)$. La fonction f est elle définie si n n'est pas entier ? Faire un graphique mettant en évidence ces résultats.

Exercice 4.3. Faire la même étude pour la fonction de Collatz définie par :

$$\forall n \in \mathbb{N}, \quad \begin{cases} \text{si } n \leq 1, & f(n) = 1, \\ \text{si } n \geq 2, & f(n) = \begin{cases} f(n/2) & \text{si } n \text{ est pair,} \\ f(3n + 1) & \text{si } n \text{ est impair.} \end{cases} \end{cases}$$

La preuve que le calcul de cette fonction se finit toujours est encore un problème ouvert !

5. Matlab et les boucles

Matlab est un interpréteur ; il n'y a pas de compilation et, parce qu'elles ne sont pas optimisées, les boucles peuvent rallonger notablement le temps de calcul :

Exercice 5.1. Écrire une fonction pour calculer les sommes

$$T_1(n) = \sum_{i=1}^n \log(i) \text{ et } T_2(n) = \sum_{i=1}^n (\log(i))^2.$$

Pour chacune des ces deux sommes, on essayera deux façons différentes :

- on utilisera dans un premier temps l'instruction *for* ;
- on utilisera dans un second temps l'instruction *sum* ;

On comparera, pour différentes valeurs de n , les deux méthodes, notamment en terme de nombres d'opérations élémentaires et de temps de calcul. Conclure.

Exercice 5.2. Entrer les trois fonctions suivantes les analyser et les comparer en terme de nombres d'opérations élémentaires et de temps de calcul. Conclure.

```
function x=somme1(n,m)
for i = 1:n
    xtemp = 0;
    for j = 1:m
        xtemp = xtemp+log(i)*exp(-j^2);
    end
    x(i) = xtemp;
end
```

```
function x=somme2(n,m)
x=sum(((log(1:n)).*ones(1,m)).*(ones(n,1)*exp(-(1:m).^2)))';
```

```
function x=somme3(n,m)
x=sum(exp(-(1:m).^2).*log(1:n));
```

6. Les polynômes

Le polynôme

$$P(X) = a_n X^n + a_{n-1} X^{n-1} + \dots + a_1 X + a_0,$$

est représenté en matlab par le vecteur ligne $[a_n \ a_{n-1} \dots \ a_1 \ a_0]$ (de longueur $n + 1$).

Outre les opérations sur les matrices qui s'appliquent donc aux polynômes, il existe un certain nombre de fonctions spécifiques aux polynômes.

$>> conv(A,B)$ multiplie les deux polynômes A et B.

$>> [Q,R]=deconv(A,B)$ renvoie les deux polynômes Q et R tels que $A = BQ + R$ avec degré de R strictement inférieur à celui de B.

$>> roots(P)$ renvoie les racines du polynôme P (dans \mathbb{C}). Réciproquement, $poly$ calcule le polynôme à partir de ces racines.

```
>> P=[1 -6 11 -6];
>> S=roots(P)
>> poly(S)
>> poly(S')
```

$>> polyval(f,X)$ renvoie les valeurs de la fonction f sur la matrice X.

$>> polyder(f)$ dérive le polynôme f.

$>> [q,d]=polyder(a,b)$ calcule la dérivée de a/b (où a et b sont deux polynômes) sous la forme q/d.

$>> polyval(f,A)$ renvoie $f(A) = p_1 A^n + p_2 A^{n-1} + \dots + p_n A + p_{n+1} I$, où A est une matrice et f un polynôme.

Exercice 6.1. Faire une fonction qui enlève les zéros superflus des polynômes (par exemple, elle renverra [0 1 2] sur [1 2]).

Exercice 6.2. Faire une fonction qui permette de faire la somme de deux polynômes de degrés différents (ce qui n'est pas possible avec matlab, car on ne peut additionner deux vecteurs de tailles différentes).

Exercice 6.3. Calculer le reste et le quotient de la division de A par B , où

$$\begin{aligned} A &= 3X^6 - 8X^5 + X^4 + 16X^3 - 19X^2 + 9X + 6, \\ B &= 3X^2 + 4X - 1. \end{aligned}$$

Que remarquez-vous ? Comment pourrez on y remédier ? Faites en une fonction et testez la sur les calculs de la division euclidienne de A par B où

$$\begin{aligned} A &= X^5 - X^4 + 6X^3 + 2X^2 - 6X - 9, \\ B &= X^3 - X^2 + 7X + 1, \end{aligned}$$

et

$$\begin{aligned} A &= X^5 - X^4 + 6X^3 + 2X^2 - 7X - 1, \\ B &= X^3 - X^2 + 7X + 1. \end{aligned}$$

Matlab permet aussi de déterminer la décomposition en éléments simples dans \mathbb{C} . Si A et B sont deux polynômes, et si A/B n'admet que des pôles simples, on peut calculer, avec matlab, les résidus r_k , les pôles p_k et le polynôme K tels que

$$\frac{A}{B} = K + \frac{r_1}{X - p_1} + \frac{r_2}{X - p_2} + \dots + \frac{r_n}{X - p_n} + K.$$

Pour cela on tape, $>> [r,p,k] = residue(A,B)$.

Les résidus r_k sont stockés dans le vecteur r , les pôles p_k sont stockés dans le vecteur p et K dans le vecteur k .

Si la fraction A/B admet pour le pôle p_j une multiplicité m , matlab remplacera la fraction $r_j/(X-p_j)$ par

$$\frac{r_j}{X - p_j} + \frac{r_{j+1}}{(X - p_j)^2} + \dots + \frac{r_{j+m-1}}{(X - p_j)^m}.$$

Si on cherche la décomposition en éléments simples dans \mathbb{R} , on utilisera la décomposition dans \mathbb{C} et on regroupera les termes conjugués deux à deux.

Exercice 6.4. Calculer la décomposition dans \mathbb{C} de

$$\frac{17X^2 + 36X + 9}{2X^3 + 4X^2 + 2X + 4}.$$

et en déduire sa décomposition dans \mathbb{R} .

Exercice 6.5. Calculer la décomposition de

$$\frac{2X^6 - 32X^5 + 188X^4 - 516X^3 + 709X^2 - 479X^2 - 479X + 134}{X^5 - 8X^4 + 24X^3 - 34X^2 + 23X - 6}.$$

7. Les chaînes et autres types de données

7.1. Les chaînes

Une chaîne est un vecteur ligne dont le nombre de composantes est égal à la longueur de la chaîne.

Exemple :

```
>> ch='une chaîne'
>> size(ch)
>> length(ch)
```

Pour concaténer deux chaînes, on écrit celle-ci comme un vecteur ligne :

```
>> ch=[ch, ' c''est pratique ','et agréable'];
>> ch
>> disp(ch)
>> disp('ch')
```

Pour rentrer une chaîne sans saisir les apostrophes, on tape

```
>> ch= input('entrez la chaîne : ', 's');
```

Les fonctions *num2str* et *int2str* transforment respectivement un réel et un entier en une chaîne (cf. exercice suivant).

Exercice 7.1. Écrire une fonction qui transforme un polynôme (stocké sous forme de tableau) en une chaîne de caractère ; par exemple, le polynôme [3 8 -7 0 1] aura pour image la chaîne '3X^4 +8X^3-7X^2+X' .

Il existe un certain nombre de fonctions sur les chaînes dont voici quelques exemples :

```
>> abs
>> setstr
>> isstr
>>
>> str2num
>> sprintf
>> upper
>> lower
>> eval
>> blanks
>> deblank
```

7.2. Les entiers

Il existe un certain nombre de fonctions spécifiques aux entiers :

```
>> mod(a,b) : renvoie le quotient de la division entière de a par b (cf. rem).
>> factor(a) : renvoie les facteurs premiers de a.
>> primes(a) : renvoie un vecteur ligne avec les nombres premiers plus petit que a.
>> isprime(a) : retourne 1 si a est un nombre premier.
```

$\gg \text{nextpow2}(a)$: renvoie n tel que $2^{n-1} \leq a < 2^n$.
 $\gg \text{perms}(c)$: renvoie toutes les permutations possibles du vecteur c.
 $\gg \text{nchoosek}(v,k)$: v est un vecteur de longueur supérieure ou égale à k ; cette fonction renvoie toute les permutations possibles de k éléments parmi les composantes de v.

7.3. Les ensembles

Les vecteurs lignes de matlab peuvent être aussi considérés comme des ensembles, dont on donne quelques fonctions :

intersect
ismember
setdiff
sextor
unique
union

8. Analyse de fonctions

Représenter la fonction g de \mathbb{R} dans \mathbb{R} telle que

$$g(x) = \frac{5x - 6,4}{(x - 1,3)^2 + 0,002} + \frac{9x}{x^3 + 0,03} - \frac{x - 0,4}{(x - 0,92)^2 + 0,005}.$$

En trouver tous les zéros en utilisant la fonction $x=fzer('exozero',alpha)$ où alpha désigne la valeur initiale de recherche. Pour la première racine, calculer $x=fzer('exozero',alpha,tol)$ où tol est l'erreur admise. On prendra tol dans l'ensemble $\{10^{-6}, 10^{-10}, 10^{-15}, \text{eps}, 10^{-20}\}$. Que remarquez vous ?

Trouver le minimum de g grâce à la fonction $fmin(g,a,b)$.

9. L'utilisation du debugger

Avec matlab, il est possible de debugger agréablement ses script, en suivant l'évolution de l'exécution d'un script, pas à pas, ou en s'arrêtant à des endroits précis, tout en contrôlant ou modifiant les différentes variables mises en jeu.

Pour cela, une fois que l'on a ouvert les différents scripts avec l'éditeur de matlab, il faut placer, en déplaçant le curseur, des "points d'arrêt" là où l'on désirer voir le programme s'arrêter provisoirement.

On lance ensuite le programme de la fenêtre matlab (ou on appelle la fonction) ; il apparaît une invite "K>>". On contrôle ensuite l'exécution du programme, à partir de l'éditeur de plusieurs façons :

- on fait debug/continu pour aller d'un point d'arrêt à l'autre.

- on fait F10 pour avancer pas à pas.

- on fait F11 pour avancer pas à pas en entrant dans chacune des sous-procédures appelées . **Attention**, dans ce cas, on pourra éventuellement entrer dans des sources de fonctions déjà programmées de matlab, qu'il ne faut pas modifier.

Dans tous les cas, apparaît une petite flèche dans l'éditeur qui indique à quel endroit du programme on se trouve. Parallèlement, dans la fenêtre matlab, on peut contrôler les variables et même les modifier. Ces variables sont spécifiques à un espace de travail, indiqués dans le menu **stack**

Quand le programme a tourné ou est en train de tourner avec le debugger, on peut accéder aux valeurs des variables en déplaçant la souris sur chacune des variables dans l'éditeur matlab : sa valeur apparaît alors.

Exercice 9.1. Ecrire un script faisant appel à plusieurs sous fonctions et tester le debugger.

10. Calcul d'intégrales et résolution d'équations différentielles

Outre les méthodes d'intégration numériques déjà vues (rectangles et trapèzes), on peut aussi citer la règle de Simpson ; matlab l'utilise avec l'instruction *quad(f,a,b)*.

Calculer l'intégrale double

$$\int_0^1 \int_0^1 e^{-x^2-y^2} dx dy,$$

en utilisant *dblquad(f,a,b,c,d)*. **attention**, dans la définition de f, il faudra prendre garde à taper $f(x,y)=\exp(-x.^2-y.^2)$.

Matlab permet aussi d'intégrer des équations différentielles ordinaires du type :

$$\begin{cases} X'(t) = F(t, X(t)), \\ X(t_0) = X_0. \end{cases}$$

en utilisant *ode45*.

Par exemple, on intégrera numériquement le problème sur [01]

$$\begin{cases} x'(t) = -x^2(t), \\ x(0) = 1, \end{cases}$$

en utilisant la fonction xprim définie par le fichier suivant :

```
function res=xprim(t,x);
% intégration de l'edo
% y'(t)=xprim(t,y(t)) et y(t0)=y0

res=-x.^2
```

et en tapant

```
>> [T,X]=ode45('xprim',[0,1],1);
>> plot(T,X);
```

On pourra aussi regarder les fonctions *ode23*, *ode113*, *ode23t*... qui correspondent à d'autres méthodes d'intégration.