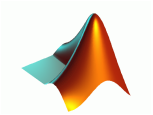

TRAVAUX PRATIQUES DE GÉNIE INFORMATIQUE

1 • But de la séance.

Cette première séance a pour but de vous faire découvrir les principes de base de la programmation sous Matlab. Cette découverte est dirigée mais nous vous conseillons de largement utiliser les modules d'aide de MATLAB. Le but de ces séances de travaux pratiques est de vous rendre autonomes, donc tout le temps que vous passerez à découvrir les fonctionnalités de ce logiciel est du temps de gagné.

1.1 • Un peu d'historique.

MATLAB est une abréviation de Matrix LABoratory. Écrit à l'origine, en Fortran, MATLAB était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle, écrite en C par the MathWorks Inc., existe en version professionnelle et en version étudiant. Sa disponibilité est assurée sur plusieurs plates-formes : Sun, Bull, HP, IBM, compatibles PC (DOS, Unix ou Windows), Macintoch, iMac et plusieurs machines parallèles.

MATLAB est un environnement puissant, complet et facile à utiliser destiné au calcul scientifique. Il apporte aux ingénieurs, chercheurs et à tout scientifique un système interactif intégrant calcul numérique et visualisation. C'est un environnement performant, ouvert et programmable qui permet de remarquables gains de productivité et de créativité.

MATLAB est un environnement complet, ouvert et extensible pour le calcul et la visualisation. Il dispose de plusieurs centaines (voire milliers, selon les versions et les modules optionnels autour du noyau Matlab) de fonctions mathématiques, scientifiques et techniques. L'approche matricielle de MATLAB permet de traiter les données sans aucune limitation de taille et de réaliser des calculs numériques et symboliques de façon fiable et rapide. Grâce aux fonctions graphiques de MATLAB, il devient très facile de modifier interactivement les différents paramètres des graphiques pour les adapter selon nos souhaits.

L'approche ouverte de MATLAB permet de construire un outil sur mesure. On peut inspecter le code source et les algorithmes des bibliothèques de fonctions (Toolboxes), modifier des fonctions existant et ajouter d'autres.

MATLAB possède son propre langage, intuitif et naturel. Il permet des gains de temps de développement spectaculaires par rapport à des langages comme le C, le TurboPascal et le Fortran. Cependant, comme ce langage est interprété, ce gain de temps se paye en temps CPU. Il est alors possible de faire des liaisons de façon dynamique, à des programmes C ou Fortran, échanger des données avec d'autres applications (via la DDE : MATLAB serveur ou client) ou utiliser MATLAB comme moteur d'analyse et de visualisation.

MATLAB comprend aussi un ensemble d'outils spécifiques à des domaines, appelés Toolboxes (ou Boîtes à Outils). Indispensables à la plupart des utilisateurs, les Boîtes à

Outils sont des collections de fonctions qui étendent l'environnement MATLAB pour résoudre des catégories spécifiques de problèmes. Les domaines couverts sont très variés et comprennent notamment le traitement du signal, l'automatique, l'identification de systèmes, les réseaux de neurones, la logique floue, le calcul de structure, les statistiques, etc.
MATLAB fait également partie d'un ensemble d'outils intégrés dédiés au Traitement du Signal. En complément du noyau de calcul MATLAB, l'environnement comprend des modules optionnels qui sont parfaitement intégrés à l'ensemble :

- Une vaste gamme de bibliothèques de fonctions spécialisées (Toolboxes)
- Simulink, un environnement puissant de modélisation basée sur les schémas-blocs et de simulation de systèmes dynamiques linéaires et non linéaires
- Des bibliothèques de blocs Simulink spécialisés (Blocksets)
- D'autres modules dont un Compilateur, un générateur de code C, un accélérateur,...
- Un ensemble d'outils intégrés dédiés au Traitement du Signal : le DSP Workshop.

MATLAB permet le travail interactif soit en mode commande, soit en mode programmation ; tout en ayant toujours la possibilité de faire des visualisations graphiques. Considéré comme un des meilleurs langages de programmations (C ou Fortran), MATLAB possède les particularités suivantes par rapport à ces langages :

- la programmation facile,
- la continuité parmi les valeurs entières, réelles et complexes,
- la gamme étendue des nombres et leurs précisions,
- la bibliothèque mathématique très compréhensive,
- l'outil graphique qui inclus les fonctions d'interface graphique et les utilitaires,
- la possibilité de liaison avec les autres langages classiques de programmations (C ou Fortran).

Dans MATLAB, aucune déclaration n'est à effectuer sur les nombres. En effet, il n'existe pas de distinction entre les nombres entiers, les nombres réels, les nombres complexes et la simple ou double précision. Cette caractéristique rend le mode de programmation très facile et très rapide. En Fortran par exemple, une subroutine est presque nécessaire pour chaque variable simple ou double précision, entière, réelle ou complexe. Dans MATLAB, aucune nécessité n'est demandée pour la séparation de ces variables.
La bibliothèque des fonctions mathématiques dans MATLAB donne des analyses mathématiques très simples. En effet, l'utilisateur peut exécuter dans le mode commande n'importe quelle fonction mathématique se trouvant dans la bibliothèque sans avoir à recourir à la programmation.
Pour l'interface graphique, des représentations scientifiques et même artistiques des objets peuvent être créées sur l'écran en utilisant les expressions mathématiques. Les graphiques sur MATLAB sont simples et attirent l'attention des utilisateurs, vu les possibilités importantes offertes par ce logiciel.

1.2 • Algèbre linéaire.

Comme vous l'avez compris, MATLAB fonctionne à la base sur la représentation matricielle et les opérations d'algèbre linéaire. Nous vous conseillons donc d'effectuer une petite remise à niveau dans ce domaine à l'aide du document de révision joint à ce TP.

2 • Découverte de MATLAB®.

2.1 • Principe.

Le principe de l'ensemble de ce module est de vous rendre autonome sur l'utilisation d'un logiciel de calcul (MATLAB ou un autre). Nous vous conseillons donc de solliciter votre curiosité.

2.2 • Quelques principes de base.

Après avoir lancé le logiciel Matlab, vous devez impérativement vous positionner dans un répertoire vous appartenant. Vous pouvez soit le faire à l'écran avec la commande `cd` :

```
>> cd \MonRepertoire
```

soit le faire en utilisant le bandeau supérieur de l'interface de Matlab. Votre meilleur ami : la commande `help` qui fait appel à la documentation de toute fonction du logiciel. Vous pouvez essayer :

```
>> help cd
```

N'hésitez pas à consulter l'aide complète en cliquant sur le logo HELP du bandeau.
Contrairement au C, les variables utilisées sous Matlab ne sont pas typées et ne nécessitent pas de déclaration. Pour créer une variable, il suffit de l'affecter. Par exemple :

```
>> x = [1,2,3] ;
```

crée la variable x en tant que tableau à une ligne et 3 colonnes.
Vous pouvez visualiser x en tapant simplement son nom :

```
>> x
```

Vous pouvez lister l'ensemble des variables utilisées dans l'espace courant en tapant la commande `who` :

```
>> who
```

Pour créer un vecteur de 1 colonne et quatre lignes il faut taper :

```
>> y = [1;2;3;4]
```

Remarquez que si vous ne mettez par de point-virgule après la commande, le résultat est affiché. De façon générale il faut séparer les commandes soit par une virgule, soit par un point-virgule (pour afficher ou ne pas afficher le résultat).
Essayez ces différentes commandes :

```
>> clock
>> date
>> pi
>> 0 / 0
>> 5^3
```

Expliquez.
Pour faire du calcul sous matlab, il suffit d'écrire la formule. C'est très simple. Par

exemple essayez de calculer : $x = \frac{2\left(512\cos\left(\frac{\pi}{3}\right) + 13^2\right)}{127}$.

Comme vous pouvez le voir, sous Matlab, un clou chasse l'autre car le fait de donner une nouvelle valeur à x vous a fait perdre son ancienne valeur.

Essayez les opérateurs suivants : `< > <= >= == ~= && ||` (inférieur, supérieur, inférieur ou égal, supérieur ou égal, différent, et, ou). Notez le symbole ~ qui signifie la négation et qui est l'équivalent du `!` en C.

Essayez d'utiliser des fonctions telles que racine carrée (`sqrt`), exponentielle (`exp`), logarithme (`log`), sinus, cosinus, tangente, arctangente. Découvrez la fonction atan2.

-> • Quelques variables spéciales :

vous avez dejà découvert `pi`, `inf` représente l'infini et `NaN` (not a number) est la valeur attribuée à un calcul qui ne peut aboutir (comme $\frac{0}{0}$ par exemple).

2.3 • Calcul matriciel.

Le principe de base de Matlab est de permettre d'effectuer des calculs matriciels avec la même facilité que vous venez d'effectuer des calculs sur des réels. Les opérations portent les mêmes noms ... mais il faut bien sûr faire attention aux dimensions des matrices. A titre d'exemple :

```
>> A = [1,2,3;4,5,6] ; B = [1,2;3,4;5,6] ;
>> C = A * B ;
>> D = B * A ;
>> E = B + A ;
```

Toutes les opérations sont utilisables avec les matrices : sinus, cosinus, exponentielle, multiplication, division, ... que signifierait une division pour une matrice ?

Pour inverser une matrice `A` vous pouvez écrire soit `inv(A)` soit `A^-1`.

Si les opérations classiques sont à interpréter au sens des matrices, vous pouvez aussi effectuer une opération terme a terme en faisant précéder votre opération d'un point :

```
>> A = [1,2;3,4] ; B = [5,6;7,8] ;
>> C = A * B ;
>> D = A .* B ;
```

Constatez la différence entre ces deux opérations. Refaites-le avec d'autres opérations.

Pour transposer une matrice :

```
>> D = A' ;
```

Il vous est possible de créer des matrice spéciales telles que des matrices identité, des matrices ne contenant que des zéros et des matrices remplies de nombres aléatoires (tirages uniformes ou gaussiens). Découvrez ces fonctions : `ones`, `zeros`, `rand` et `randn`.

2.4 • Affichage.

Matlab vous donne la possibilité d'afficher graphiquement les résultats d'expérimentations. Pour afficher une figure :

```
>> figure(10) ;
```

Comme vous pouvez le voir, cette commande fait apparaître une fenêtre d'affichage. Chaque fois que vous voudrez afficher sur cette fenêtre, il vous suffira de taper la même commande.

-> • Exercice : trouvez et affichez une droite au sens des moindres carrés.
On va supposer une droite de pente a=2 et d'origine b=-3. Son équation est donc $y = ax + b$. Vous allez créer un vecteur de 50 valeurs comprises entre 0 et 10 par :

```
>> x = (0:50) * 10 / 50 ;
```

Vous auriez aussi pu écrire :

```
>> x = 0:0.2:10 ;
```

Ensuite vous créez le vecteur des 50 valeurs de y correspondant à ces 50 valeurs d'abscisse :

```
>> a = 2 ; b = -3 ; y = a*x + b ;
```

Pour visualiser la droite :

```
>> figure(10) ; plot(x, y, 'r-') ;
```

Il y a de nombreuses options pour la fonction plot. Découvrez les par vous-même.
Par exemple changez la couleur de la droite, affichez la en points-tillés etc.
Nous allons bruiter cette correspondance linéaire entre x et y par un bruit gaussien centré de variance 2 :

```
>> y = y + ( randn(size(y)) * sqrt(2) ) ; figure(11) ; plot(x, y, 'b*') ;
```

La suite, c'est à vous de le faire avec cette petite aide mathématique :
Soient n points d'abscisses $x_1 \dots x_n$ et d'ordonnées $y_1 \dots y_n$, la droite de coordonnées (a,b) minimisant l'erreur $\|ax + b - y\|$ est donnée par :

$$\begin{bmatrix} a \\ b \end{bmatrix} = (A^T A)^{-1} A^T B \text{ avec } A = \begin{bmatrix} x_1 & 1 \\ \dots & \dots \\ x_n & 1 \end{bmatrix} \text{ et } B = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}$$

, l'opérateur $.^T$ signifiant la transposition et l'opérateur $.^{-1}$ l'inversion.
Calculez les coefficients de cette droite, tracez-la sur une autre figure.
Pour tracer deux courbes sur la même figure il faut utiliser la commande :

```
>> hold on ;
```

qui est naturellement invalidée par

```
>> hold off ;
```

Affichez la courbe et calculez la variance de l'erreur d'approximation, c'est à dire $\frac{1}{n}\sum_{i=1}^{n} (ax_i + b - y_i)^2$ en faisant des opérations matricielles (en remarquant que la norme d'un vecteur E est égale à $E^T E$). Que constatez vous ? Vous pouvez aussi calculer la moyenne de cette erreur en utilisant la fonction `sum`.

Vous pouvez aussi visualiser des données tridimensionnelles. Un exercice classique sous Matlab et qui en a été le logo pendant longtemps est le tracer du sinus cardinal en 2D.

```
>> x = -4:0.2:4 ; y = x ; z = sinc(x)'*(sinc(y)) ;
>> figure(7) ; surf(x,y,z) ;
>> figure(8) ; mesh(x,y,z) ;
```

Expliquez le résultat obtenu. Essayer de le refaire avec une Gaussienne par exemple $\left(z = \exp\left(-\frac{(x^2+y^2)}{\sigma^2}\right)\right)$. Regardez toutes les options des fonctions `surf` et `mesh`. Essayez de modifier les propriétés des images avec la fonction `shading` :

```
>> shading interp ; shading flat ;
```

Essayez aussi d'utiliser les fonctions `rotate` et `rotate3D`.

2.5 • Quelques fonctions de Matlab.

Découvrez et tracez les fonctions suivantes :

`asin, acos, atan, abs, round, floor, ceil.`

Essayez les fonctions suivantes sur des echantillons aléatoires :

`mod, mean, sum, max, min, std.`

2.6 • Les boucles.

Les boucles sous Matlab sont très simples, elles commencent par un ordre de debut de boucle comme `for`, `while`, `if` ou `switch` et **doivent toujours se terminer pas un ordre de fin de boucle** à savoir `end`.

Exemples :

```
>> for i=1:100
>> x(i) = ( 2.3 * i ) + ( 5 * randn(1,1) ) ;
>> end
>> for i=1:100
>> if ( x(i) > 5 )
>> y(i) = 1 ;
>> else
>> y(i) = -1 ;
>> end
```

Vous pouvez bien sûr regrouper tous ces ordres sur la même ligne.

Attention ! Matlab ne fait aucune différence entre les entiers et les réels, sauf que vous ne pouvez pas indicer un tableau avec des réels. Essayez par exemple :

```
>> for i=1:0.1:10
>> x(i) = ( 2.3 * i ) + ( 5 * randn(1,1) ) ;
>> end
```

-> • Exercice

A titre d'exercice, essayez de faire une petite routine qui vous donnerait la plus petite valeur réelle représentable sur votre machine (ca dépend des machines).