TP1 : Introduction à Matlab

1 Lancer Matlab

1. Créer un répertoire spécifique pour les TP de Matlab (par exemple *TPmatlab*).
2. Lancer Matlab à l'aide de la commande `matlab`. Se placer dans le répertoire de travail *TPmatlab*.
3. Quitter Matlab à l'aide de la commande `exit` ou `quit`.

   Attention! Quand vous arrêterez votre session de travail, il est nécessaire de quitter Matlab avant de vous délogguer (autrement, il est possible que Matlab reste ouvert et qu'une licence soit verrouillée).
4. Relancer Matlab. Êtes-vous encore dans le bon répertoire?

2 Aide en ligne - sauvegarde

Pour obtenir de l'aide sur une commande :

- taper `help <nom de la commande>` dans la fenêtre de Matlab, par exemple `help help`. Le résultat est affiché dans la fenêtre de travail.
- ou taper `doc <nom de la commande>` dans la fenêtre Matlab. Le résultat est affiché dans la fenêtre Help de Matlab.
- ou ouvrir le menu *Help* de la fenêtre Matlab et aller dans la rubrique *index*.

Pour obtenir de l'aide sur un sujet :

- taper `lookfor <nom du sujet>` dans la fenêtre de travail de Matlab.
- ou ouvrir le menu *Help* de la fenêtre Matlab et aller dans la rubrique *Search*.

Utiliser l'aide si besoin pour effectuer les opérations suivantes :

1. Calculer `0.5*realmax`, `2*realmax`. Que donne le résultat $0/1$? $0/0$?
2. Créer le vecteur ligne $v_1 = (1, 2, 4)$ et le vecteur colonne $v_2 = (3, 5, 6)$.
3. Sauvegarder v_1 et v_2 dans le fichier Tpinit.mat, puis dans le fichier Tpinit.txt (fichier texte).
4. A quoi servent les commandes `whos`, `clear` ?
5. Effacer v_1, v_2, et les restaurer.

3 Vecteurs et Matrices

1. Calculer si c'est possible (et si ce n'est pas le cas, analyser ce que dit Matlab) le produit scalaire des vecteurs suivants, en utilisant uniquement les opérations de base (pas de `for` ni de `while`) :

   (a) $v = (45, 3, -7, 38)$ et $w = (0, 1)$

   (b) $v = (-1... - 90)$ et $w = (1..90)$

   (c) $v = (0.01, 0.02, ..., 1)$ et $w = (-1, -0.99, ..., 2)$.

2. Trouver deux matrices inversibles (2×2) telles que $A. * B = 0$.

3. Soit A une matrice carrée. En une ligne, construire une matrice diagonale B ayant la même diagonale que A.

4. Considérons les matrices $A = \begin{pmatrix} 1 & -1 \\ 4 & 3 \\ 3 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 2 & -8 & 1 \\ 0 & 0 & -2 \end{pmatrix}$ et le vecteur $v = \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix}$. On désigne par C le produit $C = A * B$. Calculer les vecteurs $x = Bv$, $y = Ax$, $z = Cv$. Peut-on calculer l'inverse de C? Que donne Matlab comme réponse ?

5. Voici quelques exemples de matrices prédéfinies dans Matlab : `eye(n)`, `ones(n)`, `pascal(n)`, `magic(n)` où n est un entier positif.

   - Calculer `ones(n)^m` pour différentes valeurs de m et n et interpréter le résultat.
   - Calculer `ones(3)*magic(3)` et interpréter le résultat.

6. Examiner les aides en ligne des fonctions `sparse`, `full`, `spy`. Quel est l'intérêt d'utiliser ces fonctions ? Sous quelles conditions sont-elles intéressantes ?
   Tester les commandes `A = sparse(10,10,2.1)` puis `B = full(A)` et `spy(A)`.

7. On considère la matrice tridiagonale d'ordre n définie par

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

   Que fait la séquence d'instructions suivante ?

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

   De même que fait la séquence suivante ?

```
D=diag(ones(n,1));
```

`SD=diag(diag(D),-1);`
`A=2*D-SD-SD'` Comment corriger les erreurs ?

Tester enfin les commandes
`e=ones(n,1);`
`b=2*e;`
`A=spdiags([-e b -e],-1:1,n,n)`

A votre avis, laquelle est la plus pratique ?

Recommencer les mêmes opérations sur les matrices A_{10} et A_{1000}

8. Calculer les déterminants, valeurs propres et vecteurs propres, inverses s'ils existent des matrices suivantes :

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

4 Graphiques

4.1 Polygones

1. Tracer le polygone passant par les points : $A_1 = [0, 0]$, $A_2 = [0.2, 0.5]$, $A_3 = [0.4, 0.8]$, $A_4 = [0.7, 0.9]$, $A_5 = [1, 1]$, $A_6 = [0.8, 0.6]$, $A_7 = [0.5, 0.1]$.
2. Tracer un polygone régulier à 6 côtés.

4.2 Courbes

1. Tracer la courbe définie par $y = \cos(x+1)*\sin(100*x+0.5)$ en ayant échantillonné la fonction sur 50 intervalles entre 0 et 1. Que remarquez-vous ? Comment améliorer le tracé ?
2. Tracer la même courbe sans échantillonage.
3. Donner un nom aux axes et une légende en employant deux méthodes différentes.
4. Ajouter à la courbe tracée celle définie par $y = \cos(x+1)$, la tracer avec des symboles ("+" ou "×", pas en trait plein en tout cas) et d'une couleur différente.
5. Dans une autre fenêtre, tracer la fonction $f_1 : x \mapsto \frac{x-5}{x-1}$ sur $[0, 1[$ en créant un fichier `fonction1.m` avec un commentaire descriptif (précédé du signe % dans le programme).
6. Calculer $f_1(2)$, $f_1(\exp(i\pi/2))$. Que se passe-t-il si on tape `help fonction1` dans la fenêtre Matlab ?

   Prenez l'habitude de commenter les fonctions que vous écrivez.
7. Comment feriez-vous pour mettre les deux graphes précédents dans une même fenêtre de visualisation ?
8. Tracez le graphe de la courbe $t \in [0; 2\pi] \mapsto (\cos(2\pi t), \sin(2\pi t), t)$.

4.3 Histogrammes

Générer un 1000 échantillon de variables aléatoires suivant $N(0,1)$. En tracer l'histogramme. Le comparer à la densité.

4.4 Surfaces

Tracer la surface correspondant à la fonction $f(x,y) = \sqrt{xy}$ avec et sans échantillonage.

5 Programmes et procédures

Prenez l'habitude de, systématiquement, commenter au début d'un programme et d'effacer toutes les variables pour ne pas utiliser des variables déjà affectées.

Écrire un programme avec une boucle pour construire le vecteur $V = (1, , 9, ..., (10^4)^2)$. Comparer (à l'aide des fonctions `tic` et `toc` judicieusement utilisées ou encore de la fonction `cputime`) le temps de calcul de ce programme avec celui du calcul direct de ce vecteur. Conclusions ?