Initiation à Matlab

Disponible en ligne depuis

https://niess.github.io/matlab-instru/

Session 1

V. Niess

Novembre 2016

Objectifs de ce TD

- Matlab a une **syntaxe intuitive** qui se prête bien à un apprentissage par l'usage. Les objectifs de cette première session de TD sont de :
  - se familiariser avec la *manipulation de tableaux* de données, objet de base en Matlab.
  - s'habituer à utiliser l'aide en ligne.
  - lire et sauvegarder des données.
- Ce document est jallonné de **questions**, `Q%:`, vous proposant des pistes de reflexion. Les réponses sont disponibles à la fin.
- Des **exercices** vous sont également proposé afin d'illustrer les différentes notions discutées.

Q1: C'est quoi Matlab?

1. **MAT**rix **LAB**oratory : un laboratoire financé par Holywood pour la réalisation du film Matrix ?

2. Un **langage informatique** interprété de haut niveau.

    - *Interprété* par opposition à compilé. Le langage n'est pas traduit directement en instructions machine, mais lu par un programme `interprète` à la volée.
    - *De haut niveau*, dans le sens où Matlab utilise un langage mathématique, *en anglais* proche de nos concepts, par opposition à un langage proche des instructions élémentaires de la machine, comme l'Assembleur.

3. Matlab c'est **une boîte à outils** informatique. Les problèmes de bas niveau, tel que la gestion de la mémoire, sont cachés à l'utilisateur. On met bout à bout des fonctionnalités existantes. Le principal problème étant de trouver le bon outil dans sa boîte.

    - *Si un code Matlab devient complexe c'est que ce n'est probablement pas la langage le mieux adapté pour ce cas de figure.*

La Command Window

Oh! Une calculatrice!

- La **fenêtre de commande** est une console permettant d'interagir dynamiquement avec l'interpréteur de MATLAB.
  - Elle peut être utilisée comme une *calculatrice* évoluée:

```
>> 1 + 1
```

  - Le retour à la ligne, ↵, conduit à *l'évaluation* par l'interpréteur:

```
ans =
  2
```

- Les *operateurs arithmétiques* standards sont: `+`, `-` , `*`, `/` et `^` pour la puissance, i.e. `(x^y)`. La plupart des fonctions mathématiques élémentaires sont également connues, e.g. : `exp` , `log`, `sqrt` , `abs`.

Exercice 1 : les nombres heureux

- un **nombre heureux** est un nombre entier qui, lorsqu'on ajoute les carrés de chacun de ses chiffres, puis les carrés des chiffres de ce résultat et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre, donne 1. Par exemple :

  19141 :

```
>> 1^2 + 9^2 + 1^2 + 4^2 + 1^2 ↵

ans =
  100

>> 1^2 + 0^2 + 0^2 ↵

ans =
  1
```

- Votre *jour* de naissance, e.g. 13, est-il heureux? On se limitera à une 20^{aine} d'essais au plus . Comment vérifier si ce nombre est malheureux ou non?

Variables et affectation

- Le *résultat* d'une opération est affiché par Matlab comme:

```
ans =
  2
```

  Par défaut Matlab enregistre dans la variable nommée `ans` pour *answer*, i.e. réponse en Anglais, le résultat de la dernière opération . Essayez la commande suivante:

```
>> 2 * ans ↵
```

  Q2: Qu'obtenez-vous? Que vaux `ans` maintenant? Tapez simplement `ans`, puis ↵, pour vérifier.

- Pour *enregistrer* le résultat dans une autre variable il suffit d'utiliser **l'opérateur d'affectation**, `=`. Par exemple:

```
>> somme = 1 + 1 ↵

somme =
  2
```

Le Workspace

Vie et mort des variables

- L'ensemble des variables connues est visible dans la fenêtre de l'espace de travail, i.e. le *Workspace*. Vous pouvez visualiser et éditer le contenu de ces variables en cliquant avec la souris.
- La commande `clear` permet de **supprimer** une variable de l'espace de travail. Par exemple :

  ```
  >> clear somme ↵
  ```

  Pour supprimer *toutes* les variables:

  ```
  >> clear all ↵
  ```

  Vous pouvez également supprimer une variable avec la souris+*[Delete]*.

Exercice 2 : le nombre d'or

- Deux longueurs strictement positives `a` et `b` respectent la «*proportion d'or*» si et seulement si, le rapport de `a` sur `b` est égal au rapport de `a + b` sur `a`. Soit $\frac{a}{b} = \frac{a+b}{a} = \phi$. En particulier on en déduit que la suite définie par la relation de récurrence:

$$\phi_{n+1} = 1 + \frac{1}{\phi_n}$$

  converge vers le nombre d'or φ.

- Estimez le nombre d'or φ a partir de la relation de récurrence donnée par l'équation précedente.
  - On définira une variable `phi` initialisée à `1` puis contenant le résultat des itérations successives.
  - Vous pouvez naviguer dans *l'historique* des commandes avec les flèches du clavier.
  - Comparez au résultat algébrique : $\phi = \frac{1+\sqrt{5}}{2}$. A partir de combien d'itérations φ est-il connu à 3 décimales près?

Déclaration explicite d'un tableau

- Un tableau de valeurs est délimité par les opérateurs `[` et `]`. Le séparateur de colonne est l'espace ou la virgule. Ainsi :

```
>> v = [1, 2, 3]
```

ou

```
v = [1 2 3]
```

déclare explicitement un vecteur de 3 éléments de dimension `1x3`, soit `1` ligne et `3` colonnes. n.b. Matlab utilise la convention *Lin-coln*.

- Une matrice, e.g. `3x3`, peut être déclarée selon :

```
>> M = [1 2 3 ↵
  4 5 6 ↵
  7 8 9]
```

ou en utilisant le point virgule comme séparateur de ligne, selon:

```
>> M = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

Déclaration d'un tableau par blocs

- Un tableau peut être déclaré *par blocs* à partir d'autres tableaux ou de lui même. Ainsi:

```
>> A = [v; M]
```

  est un tableau de taille `4x3`.

  Q3: Que renvoie `[v, M]`? Pourquoi?

- Il est parfois utile de déclarer un tableau vide, auquel on ajoutera des éléments par la suite. La syntaxe est:

```
>> A = []
>> A = [A, v]
```

  par exemple, pour ajouter les éléments du vecteur `v` au tableau `A`.

Accès à un élément d'un tableau

- L'accès à un élément d'un tableau se fait avec l'opérateur `()`, selon :

```
>> v(1)

>> M(2, 1)
```

  pour accéder au 1er élément du vecteur `v` ou à la 2ème ligne 1ère colonne de `M` (convention *Lin-coln*).

- Pour accéder au dernier élément d'un tableau on dispose du mot-clef `end`. Par exemple :

```
>> v(end)

>> M(end, end-1)
```

  pour accéder au dernier élément de `v` ou à la dernière ligne, avant dernière colonne de `M`.

Exercice 2b: retour sur le nombre d'or

Générez un tableau contenant les vingt 1$^{\text{ères}}$ itérations de la série donnée par la relation de récurrence :

$$\phi_{n+1} = 1 + \frac{1}{\phi_n}$$

et convergeant vers le nombre d'or, φ.

- On utilisera la définition d'un *tableau par blocs* pour la relation de récurrence ainsi que le mot clef `end`.
- Vous pouvez naviguer dans l'historique des commandes avec les flèches du clavier.
- Comparez au résultat algébrique : $\phi = \frac{1+\sqrt{5}}{2}$. A partir de combien d'itérations la précision relative sur φ est-elle meilleure que `0.1 %`?

Arithmétique avec des tableaux

- La quasi totalité des fonctions de Matlab sont conçues pour *opérer sur des tableaux*. Ainsi les fonctions mathématiques tel que : `exp` , `log`, `sqrt` et `abs` s'appliquent élément par élément.

  Q4: Que renvoie Matlab pour sqrt([1, 4, 9])?

- par contre, les opérateurs élémentaires `+`, `-` , `*`, `/` et `^` s'appliquent au **sens des matrices**. Les dimensions des tableaux devant être cohérentes et la matrice à droite du `/` inversible dans le cas de la division.

- Pour appliquer les opérateurs `*`, `/` et `^` *élément par élément* il faut les préfixer avec un point. Par exemple :

```
>> [1, 2, 3] .^ 2

ans =
  1   4   9

>> [1, 2; 3, 4] .* [4, 5; 6, 7]

ans =
   4   10
  18   28
```

Aide en ligne de commande

Help! I need some function!

- Matlab intègre une **documentation exhaustive** accessible depuis la ligne de commande avec le mot clef `help` ou via le menu `Help` de la fenêtre principale. Par exemple :

  ```
  >> help sum
  ```

  renvoie une description de la fonction `sum`. Ces descriptions incluent des *exemples détaillés* à la fin, explicitant l'utilisation de la commande. On trouvera également des suggestions d'autres commandes similaires.

- Pour faire une recherche par mot-clef on peut utiliser la commande `lookfor`. Par exemple :

  ```
  >> lookfor sum
  ```

  renvoie toutes les commandes dont le descriptif contient `sum`.

Exercice 3: calcul de π ... #1

- Voici 3 formules *historiques* ayant été proposées pour estimer π:
  - Série de Leibniz Allemagne, 1676.

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

  - Produit de Wallis, Angleterre, 1655.

$$\pi = 2 \prod_{k=1}^{+\infty} \frac{(2k)^2}{(2k)^2 - 1}$$

  - Série de Madhava, Inde, 1400.

$$\pi = \sqrt{12} \sum_{k=0}^{+\infty} \frac{(-1/3)^k}{2k+1}$$

Exercice 3: calcul de π ... #2

- Estimez π à partir des 3 relations précedentes. Comparez les vitesses de convergence en vous limitant au cinq 1er termes. Pour vous aider :
  - On introduira un vecteur d'indices `k = [0, 1, 2, 3, 4]` pour générer un tableau contenant les termes d'indice k de la série.
  - Les fonctions `sum(v)` et `prod(v)` donnent la somme et le produit des termes du vecteur `v`.
  - La constante `pi` est prédéfinie dans matlab. Attention à ne pas la masquer en définissant une variable du même nom.

Types de données et chaînes de caractères

- Les variables utilisées jusqu'ici représentent des nombres réels ou entiers. Elles sont décrites dans la colonne `Class` de l'espace de travail comme étant de **type** `double array`.
- Pour représenter une séquence de texte, Matlab se sert de tableaux ou *chaînes de caractères*. La syntaxe utilise le caractère `'` comme délimiteur. Ainsi :

```
>> message = 'rhododendron'
```

  définie une variable `message` contenant le texte `rhododendron`. Sa classe est `char array`.

- La *chaîne de caractères* est un tableau de d'entiers codant pour les différents caractères du texte. La correspondance est donnée par une table ASCII. La seule différence faite par Matlab avec un tableau de réels ou d'entiers, est la **représentation** qui se fait sous forme de texte. C'est à dire que ce ne sont pas les valeurs de code qui sont affichées à l'écran, mais les caractères codés.

Conversion de type et manipulation des chaînes de caractères

- Pour passer d'un type à l'autre on peut utiliser des **fonctions de conversion** tel que `double` ou `char`. Par exemple :

```
>> double(message)

>> char(v+64)
```

  affiche les codes entiers correspondants à `message` et `char` affiche les caractères codés par le vecteur `v+64`.

- Les opérations et fonctions mathématiques définies pour des entiers ou des réels sont également applicables aux chaînes de caractères. Le résultat étant un tableau de réels.

  Q5: Que renvoient les operations suivantes? pourquoi?

```
>> '1' + 1

>> char('1' + 1)
```

Exercice 1b : retour sur les nombres heureux

- Le nombre formé par votre **date de naissance** est-il heureux? Par exemple, le 14 juillet 1789 donne le nombre `14071789`.
  - Utilisez la fonction `num2str` pour convertir un entier en un tableau de caractères contenant ses différents chiffres.
  - Vous pourrez remarquer que *les codes ASCII pour les chiffres de 0 a 9 se suivent*. Par exemple, quel est le résultat de :

    ```
    >> '0123456789' - '0'
    ```

  - Utilisez la fonction `sum` vue précédemment pour calculer en *une ligne de commande* la somme des carrés des chiffres d'un nombre.

Génération automatique de tableaux remarquables

- Pour générer automatiquement des suites arithmétiques allant de `x1` à `x2` par pas de `dx`, on dispose d'une notation abrégée: `x1:dx:x2`. Lorsque le pas vaut 1 on peut l'omettre. Par exemple :

```
>> v = 1:3

>> v = 1:1:3
```

  génèrent tous deux le vecteur `[1, 2, 3]`.

- L'on dispose également de fonctions dédiées pour générer certains tableaux remarquables. Par exemple, pour générer un tableau de zéros de 4 lignes et 3 colonnes :

```
>> A = zeros(4 , 3)
```

  Voici quelques autres fonctions utiles : `ones`, `eye`, `rand` et `randn`.

Selection d'un sous-tableau ... #1

- L'on peut accéder simultanément à plusieurs éléments d'un tableau en utilisant un **vecteur d'indices**. Par exemple :

```
>> message([1:3])
```

ou plus simplement :

```
>> message(1:3)
```

renvoîent tous deux les trois 1er éléments de message, soit `rho` dans le cas ou `message = 'rhododendron'`. Rappelons également que l'on peut utiliser le mot-clef `end`. Soit par exemple :

```
>> v(end-1:end)
```

renvoît les deux *derniers éléments* du vecteur `v`.

Q6: Comparez `v([1, end])` *et* `v(1:end)`. *Quelle est la différence? Pourquoi?*

Selection d'un sous-tableau ... #2

- Pour les matrices la notation est la même. Par exemple :

```
>> M(1:2, 1:2)

ans =
  1   2
  4   5
```

  renvoie la sous matrice `2x2` de la matrice M, `3x3`, définie précédemment.

- Dans le cas particulier où l'on souhaite sélectionner une ligne ou une colonne entière on peut *simplifier la notation* en remplaçant `1:end` par `:`. Soit par exemple :

```
>> M(1, 1:end)

>> M(1,:)
```

  sélectionnent tous deux la 1^{ere} ligne de la matrice `M`.

Propositions logiques et indexation ... # 1

- Une **proposition logique** est une expression qui ne peut prendre que 2 valeurs : `vrai` ou `faux`. En Matlab le résultat d'une proposition logique est codé dans un tableau de type logique par `0` pour `faux` et tout autre nombre pour vrai, géneralement `1`.
- Un exemple de proposition logique est la comparaison du contenu de deux tableaux. Le **test d'égalité** se fait avec *l'opérateur de comparaison* `==`. Attention à ne pas le confondre avec l'operateur d'affectation `=`.

  Q7: que renvoie :

  ```
  >> 1:3 == [2, 2, 2]
  ```

  Quel est le type du résultat? Notez que la comparaison est faite terme à terme selon l'ordre des indices.

Propositions logiques et indexation ... # 2

- Pour une liste complète des des opérateurs logiques l'on pourra consulter l'aide en ligne suivante :

```
>> help ops
```

  La syntaxe est la même qu'en langage `C` sauf pour l'operateur de **négation** qui est `~` en Matlab. Ainsi le test de la différence se fait avec `~=`.

- L'accès aux éléments d'un tableau peut se faire simplement à partir d'un **tableau logique**. Seuls les éléments qui satisfont à la condition logique sont sélectionnés. Par exemple, pour un tableau `A` quelconque :

```
>> A(A >= 0)
```

  sélectionne tous les éléments de `A` qui sont positifs ou nuls.

- La fonction `find` renvoie les indices non nuls d'un tableau. Elle permet ainsi de *convertir un tableau logique en un tableau d'indices*.

Exercice 3b: calcul de π par Monte-Carlo #1

- On se propose d'estimer la valeur de π par **Monte-Carlo**. Pour cela on mesure la surface du cercle de rayon unité rapportée à celle du carré dans lequel il est inscrit. L'algorithme est le suivant :
  - Générez aléatoirement une série de `N` points distribués uniformément dans `[0, 1]x[0, 1]`.
  - Comptez le nombre `m` de points inscrits dans le cercle de rayon unité.

  Le rapport m/N tend vers $\pi/4$ lorsque `N` tend vers l'infini.
- Une estimation de *l'erreur Monte-Carlo* est donnée par :

$$\sigma = 2\sqrt{\frac{m(N-m)}{N^3}}$$

Exercice 3b: calcul de π par Monte-Carlo #2

- Estimez π et l'incertitude Monte-Carlo associée pour $N = 10^4, 10^5$ et 10^6. *Comparez les performances* obtenues aux méthodes précédentes, utilisant des séries.
- Pour vous aider :
  - La fonction `rand` permet de générer une séquence de **nombres pseudo aléatoires** distribués uniformément dans `[0,1]`.
  - L'opérateur de comparaison *inferieur ou égal* est `<=`.
  - Pour *compter le nombre d'éléments* d'un vecteur vous pouvez utiliser la fonction `length`.

Sauvegarder son travail

- Si vous fermez Matlab *le contenu de l'espace de travail sera perdu*. Vous pouvez cependant **sauvegarder** ùne partie ou l'integralite de vos variables sur le disque avec la commande `save`. Par exemple :

  ```
  >> save mon_travail
  ```

  La sauvegarde se fait sous la forme d'un fichier compressé binaire au format `.mat`, e.g. `mon_travail.mat` dans le cas précedent.

- Notez que sauf si vous avez spécifié le chemin complet, le fichier sera sauvegardé dans le *répertoire courant*. Pour changer le répertoire courant vous pouvez utiliser la commande `cd` ou l'onglet `Current Directory` de la fenêtre Matlab.

  ```
  >> cd C:\mon\repertoire\
  ```

- Pour les personnes habituées à *DOS* ou *Linux* les commandes système `dir`, `ls`, `pwd`, `mkdir`, ... sont reconnues par Matlab.

Charger un tableau ... #1

- Pour *recharger* dans votre espace de travail les *variables sauvées* sur le disque, il suffit d'utiliser la commande `load`, selon :

```
>> load mon_travail
```

  Attention toutefois à vous trouver dans le bon répertoire ou alors à spécifier le chemin complet.

- La commande `load` permet également de charger dans l'espace de travail un tableau sauvé dans un *fichier au format texte*. La syntaxe demande maintenant de spécifier explicitement l'extension du fichier, selon :

```
>> load data.txt

>> A = load ('data.txt')
```

  La seconde forme est dite *fonctionelle*. Elle permet ici une assignation de la table chargée à la variable `A`. Si aucune extension n'est spécifiée matlab considère qu'il s'agit d'un fichier au format `.mat`.

Charger un tableau … #2

- Notez que que pour *charger automatiquement* un fichier texte avec la fonction `load` les *restrictions* suivantes s'appliquent :
    - Le fichier texte doit être *régulier*, c’est à dire sans entête et avec des lignes comportant toutes le même nombre de colonnes.
    - Le *séparateur de colonnes* peut être un espace, une tabulation, une virgule ou un point virgule.

  En l’absence d’assignation, une table contenant les données et portant le nom du fichier sans l’extension, sera créé dans l’espace de travail.

Réponses aux questions

Réponses aux questions

- Q1: C’est quoi Matlab?
  - Il fallait répondre 2 et 3. Matlab n’a pas de rapport avec le film Matrix. Par contre l'acronyme signifie bien **MAT**rix **LAB**oratory.
- Q2: Variables et affectation
  - `ans * 2` renvoie 4. En l’absence d’affectation explicite, le résultat est sauvé dans `ans` de nouveau . Donc `ans` vaut 4 maintenant.
- Q3: Déclaration d’un tableau par blocs
  - `[v, M]` donne une erreur de concaténation car v et M n’ont pas le même nombre de lignes .
- Q4: Arithmétique avec des tableaux
  - `sqrt ([1, 4, 9])` donne le tableau `[1, 2, 3]`. La racine carrée est appliquée à chaque élément du tableau de départ.

Réponses aux questions

- Q5: Conversion de type et manipulation des chaînes de caractères
  - `'1' + 1` renvoie le nombre 50 car le code ASCII correspondant au caractère `'1'` est 49. Par contre `char('1'+1)` renvoie le caractère de code ASCII 50, soit le caractère `'2'`. Les caractères `'0'`, `'1'`, ... à `'9'` se suivent. Il en est de même pour les caractères de l'alphabet.
- Q6: Selection d'un sous-tableau ... #1
  - `v([1, end])` renvoie le 1er et le dernier élément du vecteur `v` alors que `v(1:end)` renvoie tous les éléments de `v`. Sauf pour un vecteur de 2 éléments le résultat est différent.
- Q7: Propositions logiques et indexation ... # 1
  - Le résultat de `1:3 == [2, 2, 2]` est le tableau logique `[0, 1, 0]`. Seul le 2ème élément est le même dans les deux tableaux comparés.