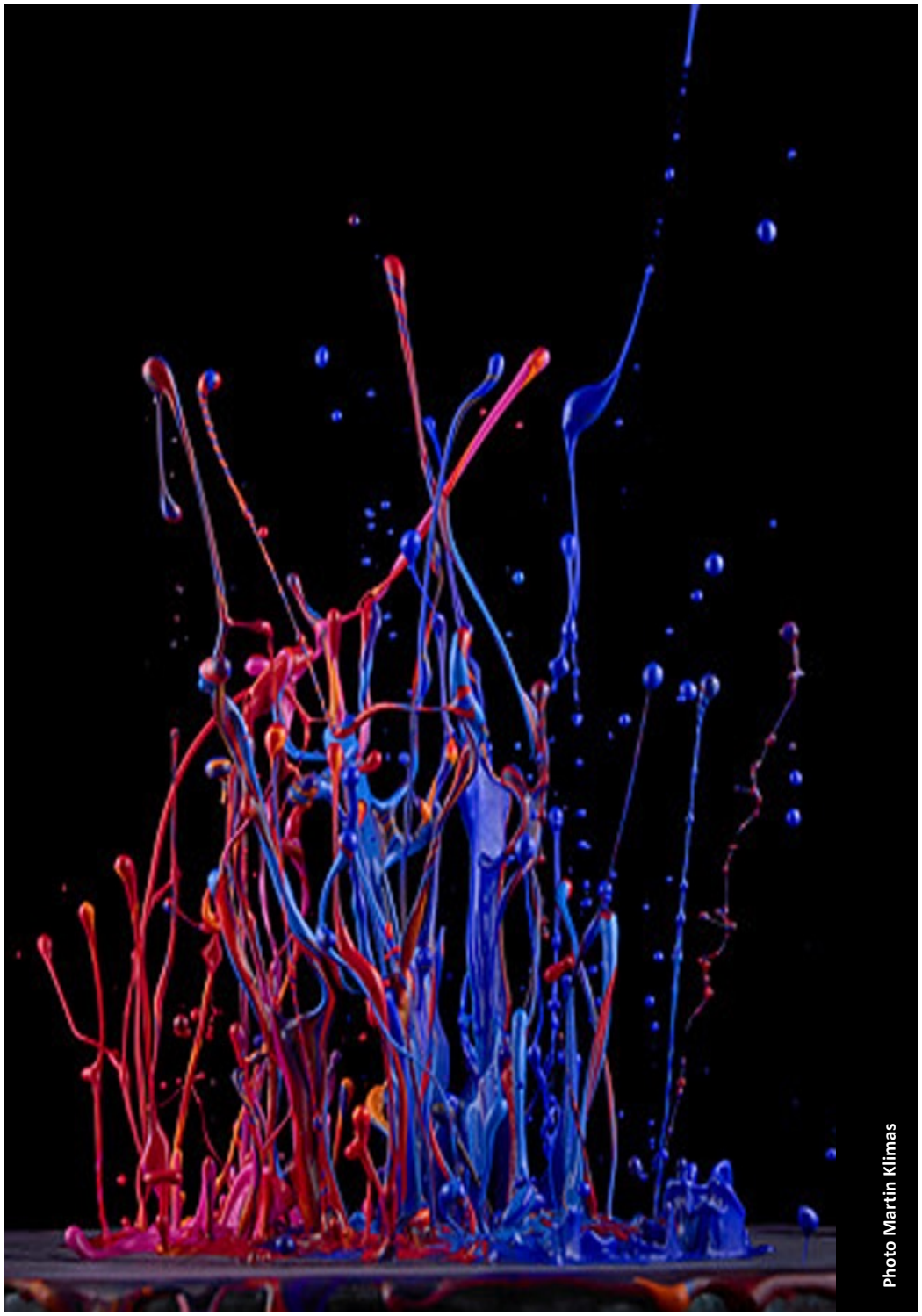


Photo Martin Klimas

EPFL

Programmation pour Ingénieur

Matlab I

ME 3e semestre

rev. 2025.r1

Christophe Salzmann

Laboratoire d'Automatique

Today

- Matlab introduction
- Environnement
- Matlab interactif
  - vecteurs, matrices et fonctions associées
- Sauvegarde/chargement de données

Why using MATLAB?

Demo

graph, simul, optim

Matlab

mat*rix* ***lab****oratory*

- Matlab est un langage de haut niveau et un environnement de développement spécialisé dans le calcul numérique
- MATLAB a été conçu par Cleve Moler ~ 1970-80
- Le but était de permettre l'accès aux librairies numériques LINPACK et EISPACK sans avoir à programmer du Fortan
- Le développement initial est porté en C et commercialisé en 1984 par la société The MathWorks
- Fonctions mathématiques pour l'algèbre linéaire, statistiques, analyse, filtrage, optimisation, intégration numérique, etc.

Matlab

Calculs Numériques

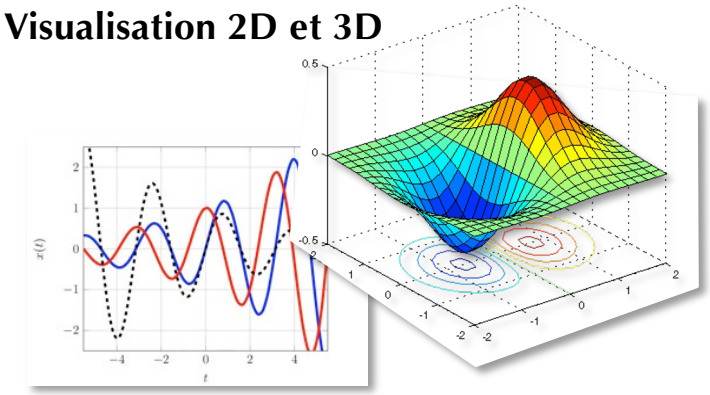
Calculs Symboliques

$$f(x) = A\sin(wx)$$

$$\int f(x)dx = -\frac{A}{w}\cos(wx) + C$$

$$\frac{df(x)}{dx} = Aw\cos(wx)$$

Visualisation 2D et 3D



Programmation (interpreter/compiler)

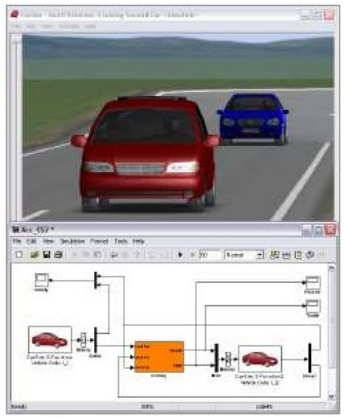
```
% We must go through all the under diagonal elements of M
for i = 2:n
    for j = 1:i-1
        G = eye(n,n);
        if R(i,j) ~= 0

            % Compute the next rotation sin and cosin
            r = sqrt(R(j,j)^2 + R(i,j)^2);
            c = R(j,j)/r;
            s = R(i,j)/r;

            % Define the Givens matrix
            G(i,j) = -s;
            G(j,i) = s;
            G(j,j) = c;
            G(i,i) = c;

            % Rotate the Q and R matrices
            Q = Q*G';
            R = G*R;
        end
    end
end

return;
```

Calculs Matriciels et Vectoriels

$$Ax = b \Rightarrow x = A^{-1}b$$

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Simulations Dynamiques



Matlab

- Matlab a des centaines de fonctions intégrées
- Et des milliers de librairies *toolbox* pour la plus part des domaines scientifiques
- Simulink: environnement de simulation (pas couvert dans ce cours)

Parallel Computing
- Parallel Computing Toolbox
- MATLAB Distributed Computing Server

Math, Statistics, and Optimization
- Symbolic Math Toolbox
- Partial Differential Equation Toolbox
- Statistics Toolbox
- Curve Fitting Toolbox
- Optimization Toolbox
- Global Optimization Toolbox
- Neural Network Toolbox
- Model-Based Calibration Toolbox

Control System Design and Analysis
- Control System Toolbox
- System Identification Toolbox
- Fuzzy Logic Toolbox
- Robust Control Toolbox
- Model Predictive Control Toolbox
- Aerospace Toolbox

Signal Processing and Communications
- Signal Processing Toolbox
- DSP System Toolbox
- Communications System Toolbox
- Wavelet Toolbox
- Fixed-Point Toolbox
- RF Toolbox
- Phased Array System Toolbox

Image Processing and Computer Vision
- Image Processing Toolbox
- Computer Vision System Toolbox
- Image Acquisition Toolbox
- Mapping Toolbox

Test and Measurement
- Data Acquisition Toolbox
- Instrument Control Toolbox
- Image Acquisition Toolbox
- OPC Toolbox
- Vehicle Network Toolbox

Computational Finance
- Financial Toolbox
- Econometrics Toolbox
- Datafeed Toolbox
- Database Toolbox
- Spreadsheet Link EX (for Microsoft Excel)
- Fixed-Income Toolbox
- Financial Derivatives Toolbox

Computational Biology
- Bioinformatics Toolbox
- SimBiology

Code Generation and Verification
- MATLAB Coder
- HDL Coder
- HDL Verifier
- Filter Design HDL Coder

Application Deployment
- MATLAB Compiler
- MATLAB Builder NE (for Microsoft .NET Framework)
- MATLAB Builder JA (for Java language)
- MATLAB Builder EX (for Microsoft Excel)
- Spreadsheet Link EX (for Microsoft Excel)

Database Connectivity and Reporting
- Database Toolbox
- MATLAB Report Generator

Matlab

- Matlab peut être employé de manière interactive (mode *calculette*) ou programmé à l'aide de scripts ou de fonctions externes.
- Matlab possède un environnement complet de programmation (éditeur, debugger, gestion de fichiers, etc.)
- Matlab a des fonctionnalités et structures (boucles, if, etc.) à un autre langage de haut niveau (ex. C/C++)
- L'entier de la documentation se trouve dans le help!

En tant qu'étudiants de l'EPFL vous avez le droit d'installer la version étudiant sur votre machine personnelle! (en VPN) ici:

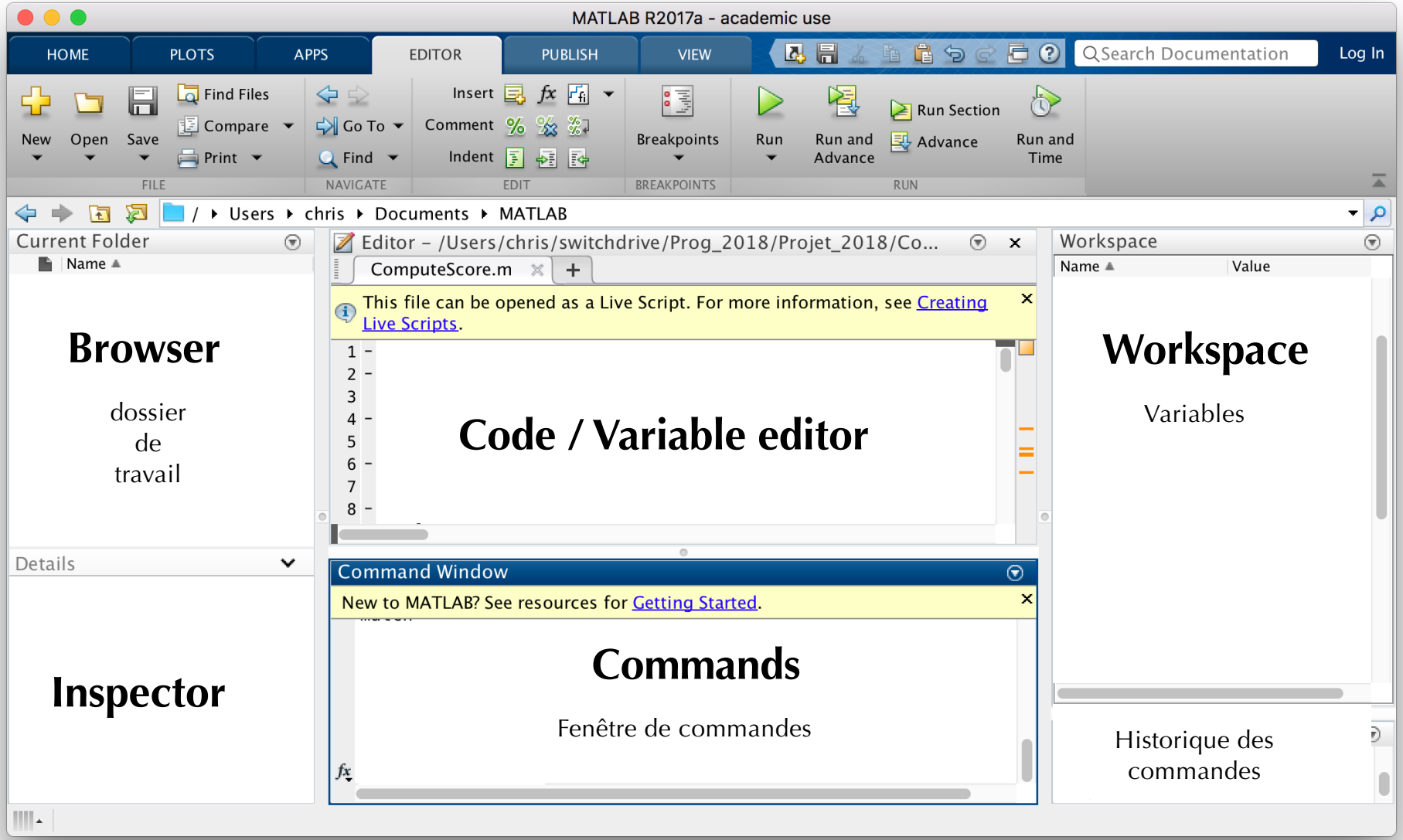
https://distrilog-etudiants.epfl.ch/

Existe aussi en application web (utiliser votre email EPFL pour créer un compte), ici: https://matlab.mathworks.com

Demo

intro

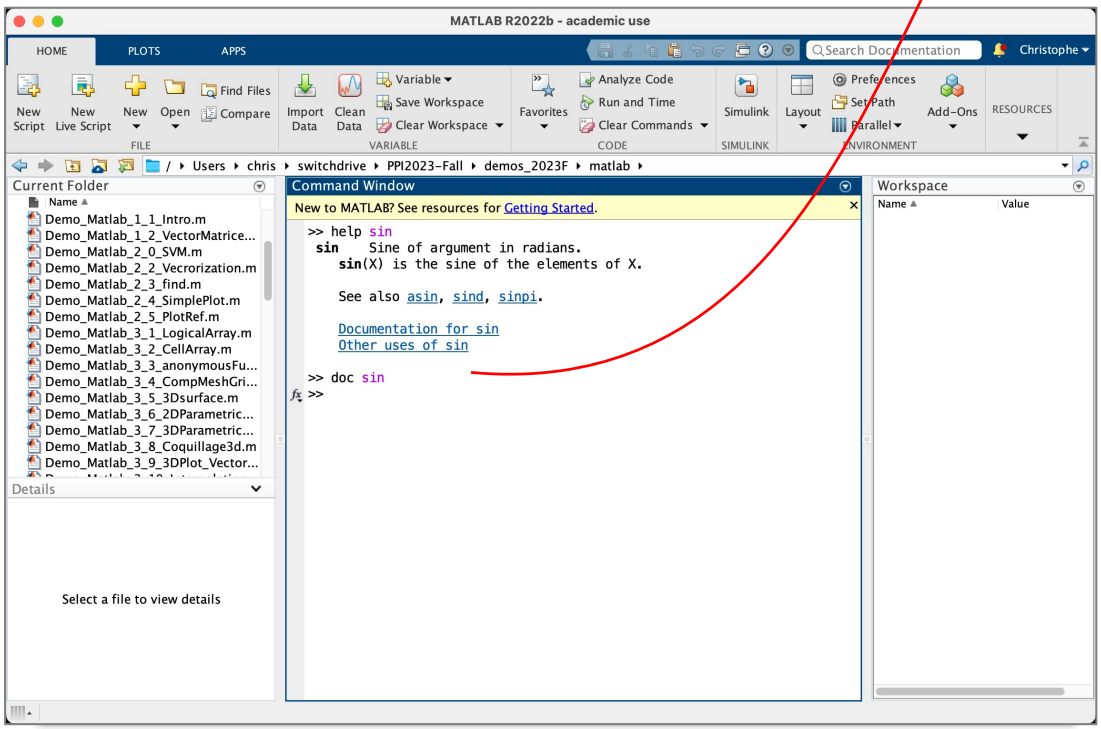
Environnement

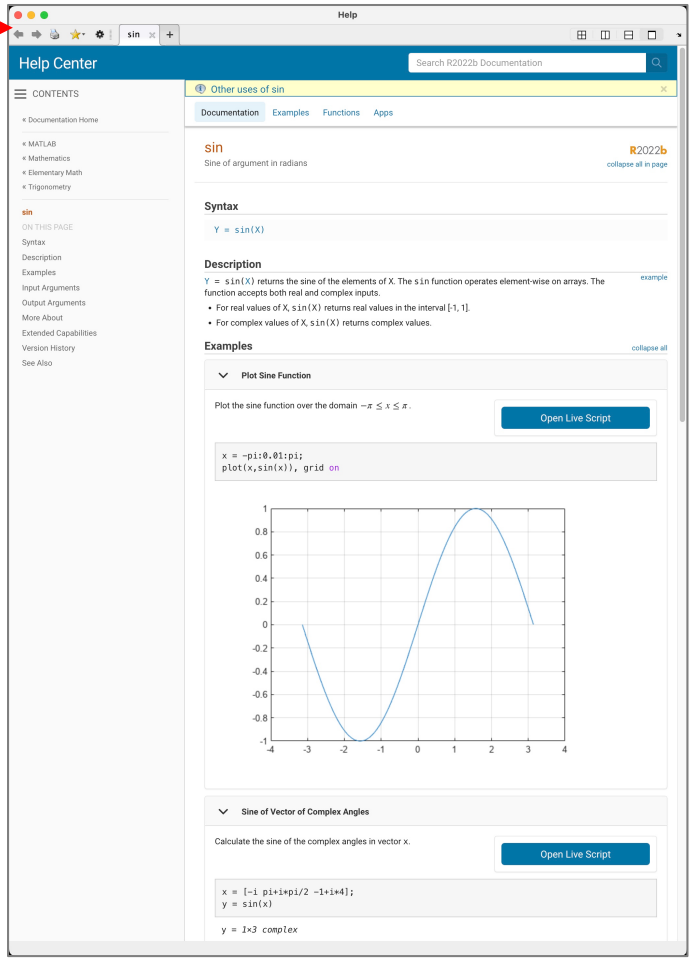


Matlab interactif

La fenêtre de commandes vous permet d'entrer des commandes de manière interactive.
Ex:

```
>> help sin
>> doc sin
>> demo
```





Matlab interactif

- Matlab est un *interpréteur*, ce qui veut dire qu'il traite une ligne à la fois. Il n'y a pas d'étape de compilation, le code s'exécute jusqu'à la fin ou s'arrête à la première erreur survenue.

- Les variables n'ont pas besoin d'être créées à l'avance, elles le sont à la volée. Les variables *vivent* dans le **Workspace**.

- Matlab travaille principalement avec des nombres à virgule flottante (float - 64 bits), les entiers, les char et les strings. Il existe d'autres types/structures de données: complexes, structures, cell arrays, maps, java object, functions handlers.

- Ex.

```
>> 3*sin(0.5)
ans =
       1.4383
```

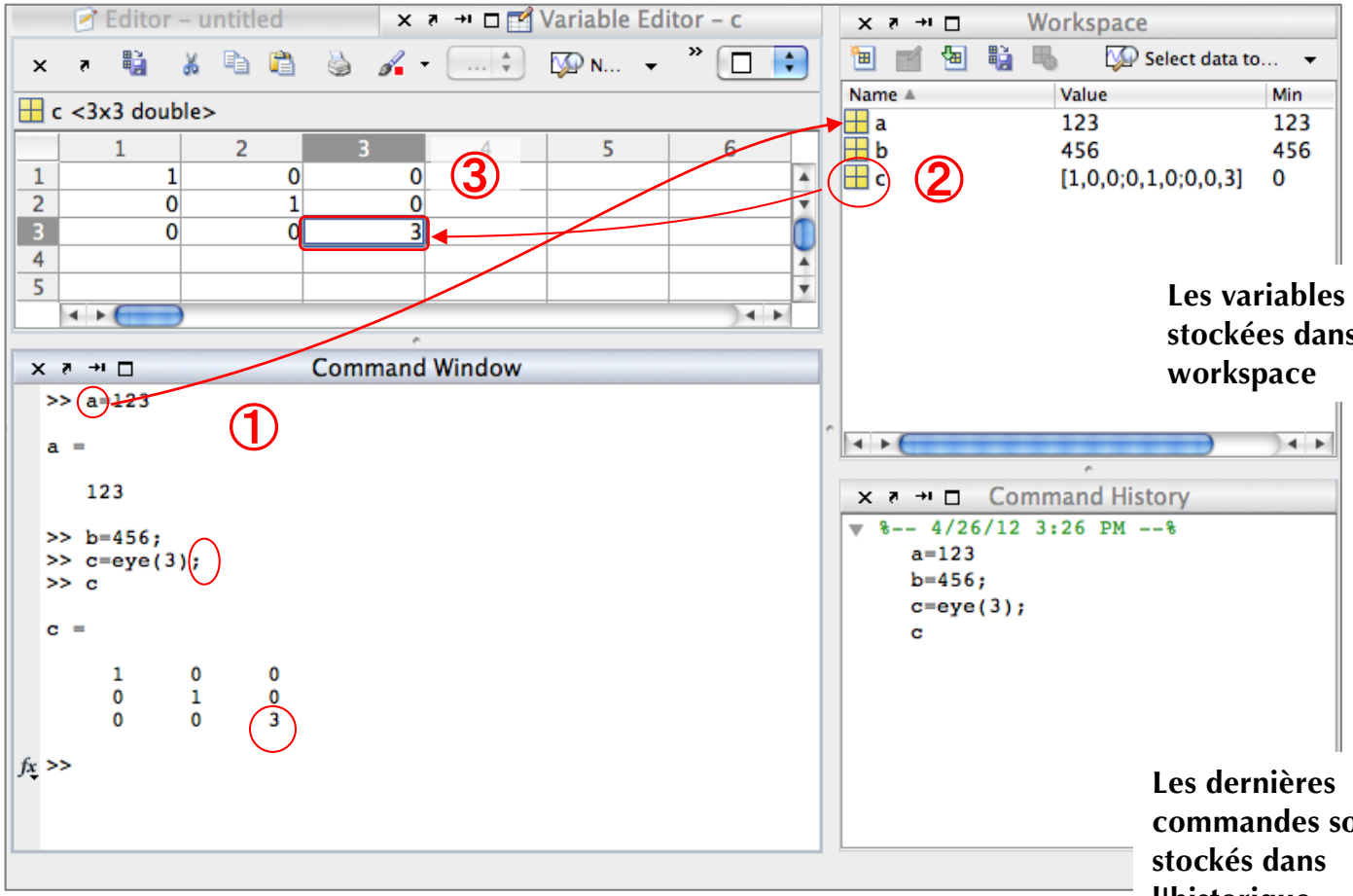
Matlab interactif

- Pour créer une variable il suffit de lui assigner une valeur
- Les noms des variables doivent **commencer par une lettre**, après lettres, chiffres et '_'
- Les noms des variables est ***case sensitive*** (`myVar` ≠ `MyVar`)
- Il existe des noms de variables/constantes prédéfinis, ex: `i, j, pi, ans, Inf, -Inf, NaN,` etc..
- Ex.

```
>> myVar = 1.234
>> myStr = 'abc'
>> 1_var = 3
??? 1_var = 3
       |
   Error: The input character is not valid in
MATLAB
   statements or expressions.
```

Matlab interactif

Il est possible d'éditer les variables du workspace



Mettre ; à la fin de la ligne supprime le retour dans la fenêtre de commande

Les variables sont stockées dans le workspace

Les dernières commandes sont stockés dans l'historique

① création de la variable dans la fenêtre de commande
② la variable et sa valeur sont alors affiché dans le *workspace* = mémoire de Matlab
③ il est possible de **visualiser** et de **modifier** le contenu de la variable

Matlab interactif

- Il existe un grand nombre de fonctions mathématiques élémentaires qui s'appliquent aussi bien aux scalaires qu'aux éléments de vecteurs ou de matrices
- Les angles sont exprimés en radians

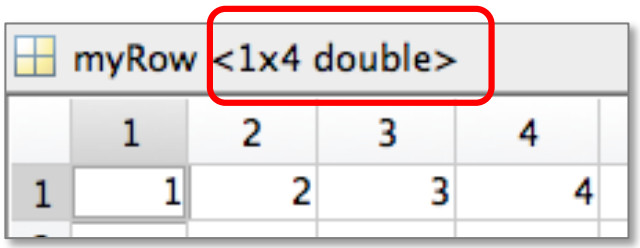
```
>> A = pi;
>> cos(A)
ans =
    -1
```

```
>> sin(A)
>> cos(A)
>> tan(A)
>> asin(A)
>> acos(A)
>> atan(A)
>> abs(A)
>> sqrt(A)
>> real(A)   % partie réelle
>> imag(A)   % partie imaginaire
>> conj(A)   % complexe conjugué
>> exp(A)
>> log(A)    % log naturel
>> log10(A)  % log base 10
```

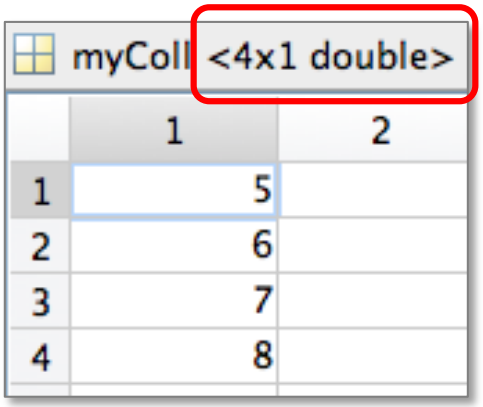
Vectors

- L'intérêt de Matlab est sa grande facilité à travailler avec des *arrays* ou *matrix* et des *vectors*.
- Matlab différencie les vecteurs collones des vecteurs ligne !
- **Vecteur ligne (rows)**, éléments séparés par ' ' (espace) ou ','
- **Vecteur collone (column)** éléments séparés par ';'
- Ex:

```
>> myRow = [1 2 3 4]

myRow =

     1     2     3     4
```




```
>> myColl=[5; 6; 7; 8]

myColl =

     5
     6
     7
     8
```



Vectors access

```
>> A = [1 2 3 4];    % '%' indique un commentaire
>> A(1)              % A(n): accède au nᵉ élément de A

ans =                % ans: nom de la variable de réponse si non assignée
     1

>> A(0)
??? Subscript indices must either be real positive integers or logicals.
```

En Matlab les indexes commencent à 1!

Vectors access

```
>> A = [1 2 3 4];
>> A(7) = 8;          % Etend A[] à 7 éléments, met la 7e valeur à 8
                      % Les valeurs non-définies sont mises à 0
A =
     1     2     3     4     0     0     8

>> V=(0: 0.2 :1)      % Défini un vecteur (X0: pas :Xmax)
V =
      0    0.2000    0.4000    0.6000    0.8000    1.0000

>> W=(0:1)            % Si omis, le pas est de 1 (X0:Xmax)
W =
     0     1

>> Z=(0:2:1)          % matlab tries its best…
Z =
     0
```

Vectors access

```
>> B = [1 2 3 4 5 6];
>> c = [2 4 6];
>> B([1 3])      % l'index peut etre un vecteur
ans =           % les valeurs de la position 1 et 3 de B, [B(1) B(3)]
     1   3

>> B(c)          % l'index est le vecteur c
ans =
     2   4   6

>> B(2:5)        % l'index est le vecteur [2 3 4 5]
ans =
     2   3   4   5

>> B(2:end-3)    % l'index est le vecteur [2 (6-3)] -> [2 3]
ans =
     2   3
```

Vectors opérations

```
>> A = rand(2)    % génère une matrice 2x2 de nombres aléatoires
A =
    0.4427    0.9619
    0.1067    0.0046

>> B = rand(1,5) % génère une matrice 1x5 de nombres aléatoires
B =
    0.7482  0.4505  0.0838  0.9133  0.2290

>> m = max(B)
m=
     0.9133
```

Paramètres de retour multiple -> dans un vecteur

```
>> [m p] = max(B)   % retourne
m=                  %  - valeur
     0.9133
p=                  %  - indice
     4
```

```
>> [~, p] = max(B) %  ignoring 1st param
p=                 %  - indice
     4
```

Vectors opérations

```
>> length(B);       % taille de B
>> sum(B);          % somme des elem. de B if vect, for matrices sum(B, 'all') see help
>> mean(B);         % valeur moyenne de B, = sum(B)/length(B)

>> B=sort(B)
B = 0.0838   0.2290   0.4505    0.7482   0.9133
```

Polynomes

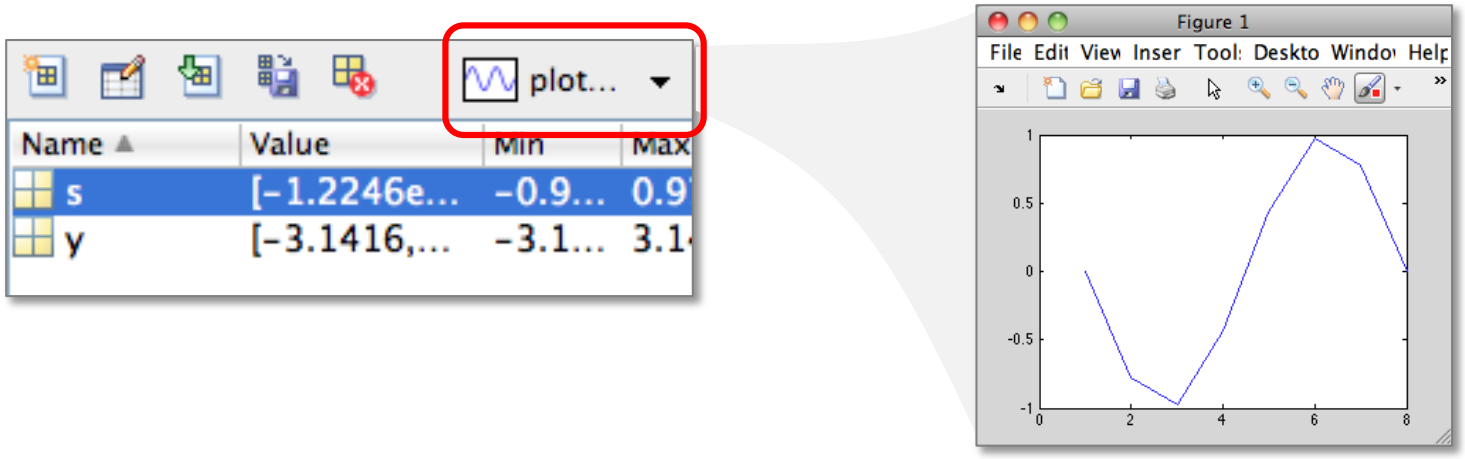
Matlab défini les coéfficiants d'un polynome sous la forme d'un vecteur

```
>> p = [1 -6 -72 -27]; % x^3-6x^2-72x-27
>> r = roots(p)        % racines du polynome P
r =
   12.1229
   -5.7345
   -0.3884

>> poly(r)            % polynome à partir des racines r
ans =
    1.0000   -6.0000  -72.0000  -27.0000  % val. identiques ☺
```

Vectors

Matlab permet d'éviter l'utilisation de boucles pour créer ou itérer sur les éléments d'un vecteur ou d'une matrice.

```
>> y = linspace(-pi,pi,8)        % y = linspace(Min, Max, nbrValeurs)
                                 % crée un vecteur de nbrValeurs allant de Min à Max
y =
   -3.1416   -2.2440   -1.3464   -0.4488    0.4488    1.3464    2.2440    3.1416A

>> s=sin(y)  % sin est appliqué à chaque élément du vecteur, pas besoin de boucles
☺
s =
   -0.0000   -0.7818   -0.9749   -0.4339    0.4339    0.9749    0.7818    0.0000
```



Matrices

- Les matrices se construisent de la même manière que les vecteurs: ' ' et ',' pour séparer les éléments sur la même ligne et ';' pour séparer les colonnes

```
>> m = [1 2 3 4;  5 6 7 8]

m =

     1     2     3     4
     5     6     7     8
```

m <2x4 double>

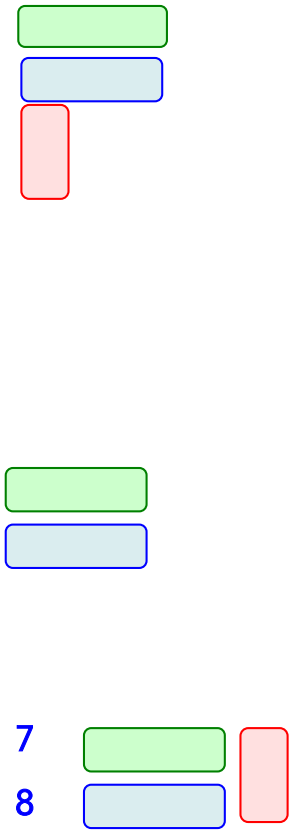
	1	2	3	4
1	1	2	3	4
2	5	6	7	8

```
>> v = []  % empty matrix
v =
     []
```

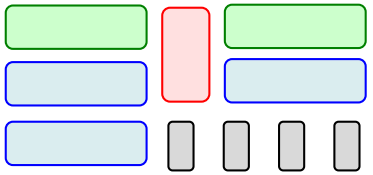
Matrices

- Les matrices se construisent par concathénation de vecteurs et/ou matrices (attention aux dimensions)

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = [7;8];

>> m =[a;b]
m =
     1   2   3
     4   5   6

>> n=[m c]
n =
     1   2   3   7
     4   5   6   8
```

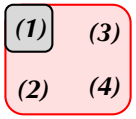


```
>> p = [[[a;b] c [a;b]]; [b 9 0 2 1]]
p =
     1   2   3   7   1   2   3
     4   5   6   8   4   5   6
     4   5   6   9   0   2   1

>> size(p)
ans =
     3   7

>> size(a)
ans =
     1   3
```

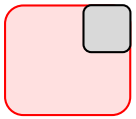


Matrices access

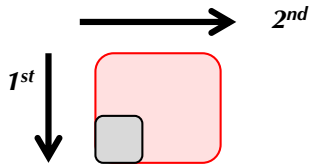
```
>> M = [1 2 ; 3 4]
>> M(1) % pos:1
ans =
     1
```




```
>> M(3)
ans =
     2
```



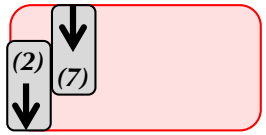
```
>> M(2,1) % row:2, col: 1
ans =
     3
```



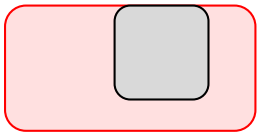

```
>> M(1,2)
ans =
     2
```

'...' -> continue à la ligne suivante

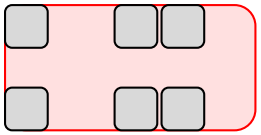
```
>> N=[ 1  2  3  4  5; ...
       6  7  8  9 10; ...
      11 12 13 14 15];

>> N(2:5)  % selectionne pos 2 à 5
ans =
     6  11   2   7
```




```
>> N(1:2,3:4) % selectionne sub-matix
              % rows,cols
ans =
     3     4
     8     9
```



```
>> N([1 3], [1 3 4])
```



Matrices access

```
>> M = [1 2 ; 3 4]
>> M(1,:)    % M(row,col)
             % ':'-> en entier
ans =
     1     2

>> M(:,1) % read as
          % all lines of col 1
ans =
     1
     3

>> M(:,:)
ans =
     1     2
     3     4
```

```
>> N = [1 2 3; 4 5 6]
N =
     1     2     3
     4     5     6

% remplace 2e colonne
>> N(:,2)=[7 8]
N =
     1     7     3
     4     8     6

% Efface 2e colonne
>> N(:,2)=[]
N =
     1     3
     4     6
```

Matrices création

Fonctions prédéfinies pour créer des matrices/vecteurs.

```
% matrice de 0
>> Z = zeros(2,3)
z =
     0     0     0
     0     0     0

% matrice de 1, carrée par défaut
O = ones(2)
O =
     1     1
     1     1

% matrice identité
E = eye(2,3)
E =
     1     0     0
     0     1     0
```

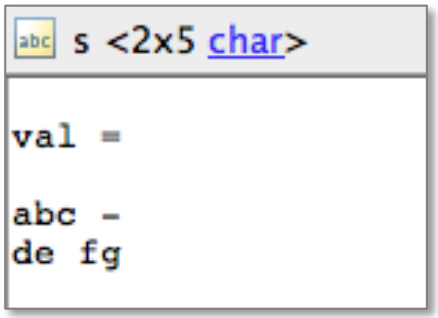
```
% Matrice diagonale
>> D = diag([1 2 3])
D =
     1     0     0
     0     2     0
     0     0     3

% Elements sur la diagonale
>> diag(E)
ans =
     1
     1
```

Vector & matrix demos

String

- Les strings sont des matrices comme les autres, les éléments étants des caractères.

```
>> s=['ab' 'c -';'de fg']
s =
abc -
de fg
```



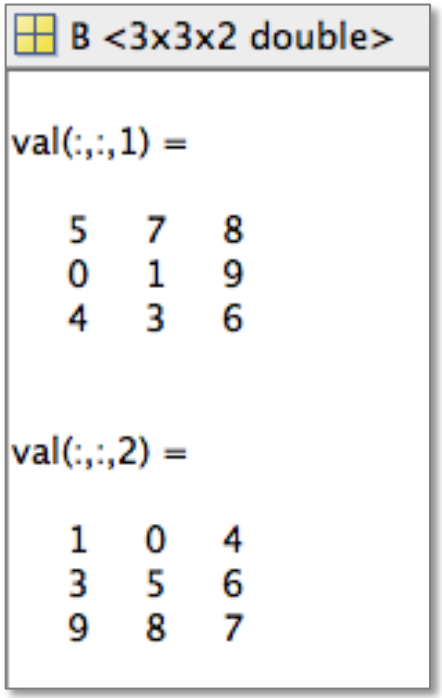
- Des fonctions sur les chaines de caractères similaires au C sont disponibles, ex. sprintf(), num2str(), sscanf()

```
>>
sprintf('%0.5g',(1+sqrt(5))/2)

ans =
    '1.618'
```

Matrices 3D *and more*

- Matlab travaille principalement avec des matrices 2D mais supporte les matrices de plus de 2 dimensions.

```
>> B = [5 7 8;
        0 1 9;
        4 3 6];
% ajoute une dimension
>> B(:,:,2) = [1 0 4;
               3 5 6;
               9 8 7];
>> B
B(:,:,1) =
     5     7     8
     0     1     9
     4     3     6
B(:,:,2) =
     1     0     4
     3     5     6
     9     8     7
```



Matrices 3D *and more*

- Matlab travaille principalement avec des matrices 2D et supporte les matrices de plus de 2 dimensions. La fonction **CAT()** concatène les arguments de la fonction.

C = CAT(DIM, A1, A2,..)

```
>> C = cat(3, [2 8; 0 5], ...
           [1 3; 7 9], [2 3; 4 6])
C(:,:,1) =
     2     8
     0     5
C(:,:,2) =
     1     3
     7     9
C(:,:,3) =
     2     3
     4     6
```

```
>> D = cat(4, C, ...
          cat(3, ...
            [2 8; 0 5], ...
            [1 3; 7 9], ...
            [2 3; 4 6]));
```

CAT(2,A,B) ≡ [A,B]

CAT(1,A,B) ≡ [A;B]

D <4-D double>

```
val(:,:,1,1) =
   2   8
   0   5

val(:,:,2,1) =
   1   3
   7   9

val(:,:,3,1) =
   2   3
   4   6

val(:,:,1,2) =
   2   8
   0   5

val(:,:,2,2) =
   1   3
   7   9

val(:,:,3,2) =
   2   3
   4   6
```

Sparse Matrices

- Matlab support les matrices creuses. C'est à dire des matrices où il y a principalement des 0
- Pour économiser de la place mémoire, matlab stocke uniquement les coordonnées et les valeurs pour les éléments différents de 0

```
>> A = [ 0   0   0   5
         0   2   0   0
         1   3   0   0
         0   0   4   0];
S = sparse(A)
S =

   (3,1)         1
   (2,2)         2
   (3,2)         3
   (4,3)         4
   (1,4)         5
```

S = sparse(idx_raw,idx_col,value,nbr_raw,nbr_col)

```
>> S = sparse([3 2 3 4 1],[1 2 2 3 4],[1 2 3 4
5],4,4)

S =

   (3,1)         1
   (2,2)         2
   (3,2)         3
   (4,3)         4
   (1,4)         5
```

Matrices opérations

Transposé

```
>> x = [-1 0 2];
>> x'
ans =
    -1
     0
     2
```

```
>> A = [1 2 3 ; ...
        4 5 6 ;...
        7 8 0];

>> A'
ans =
     1     4     7
     2     5     8
     3     6     0
```

```
>> Z = [1 + 5i,2 + 6i;...
        3 + 7i,4 + 8i];

>> Z'   % transposée complexe conjuguée
ans =
   1.0000 - 5.0000i   3.0000 - 7.0000i
   2.0000 - 6.0000i   4.0000 - 8.0000i

>> Z.'  % transposée
ans =

   1.0000 + 5.0000i   3.0000 + 7.0000i
   2.0000 + 6.0000i   4.0000 + 8.0000i
```

Matrices opérations

```
>> A = [1 2 3 ; ...
        4 5 6 ;...
        7 8 0];

% Attention aux
dimensions
>> A + A'
ans =
     2     6    10
     6    10    14
    10    14     0

% -1 élém. by élém.
>> A-1
ans =
     0     1     2
     3     4     5
     6     7    -1
```

.operation indique une opération élément par élément, ex: **.*** **./**

```
% mul. élém. by élém.
>> A .* A
ans =
     1     4     9
    16    25    36
    49    64     0

>> A./A
ans =
     1     1     1
     1     1     1        % NaN: Not a Number
     1     1   NaN        % 0/0 indéfini -> NaN
                          % 1/0: Inf, -1/0: -Inf
```

Sauvegarder des données

Il est possible de sauver le contenu ou une partie du *workspace* dans un fichier. Par défaut au format **binaire**, et extension *.mat*

```
% sauve l'entier du workspace dans a.mat
>> save a
```

```
% sauve v1 v2 v3 dans data.mat
>> save data.mat v1 v2 v3
```

data.mat

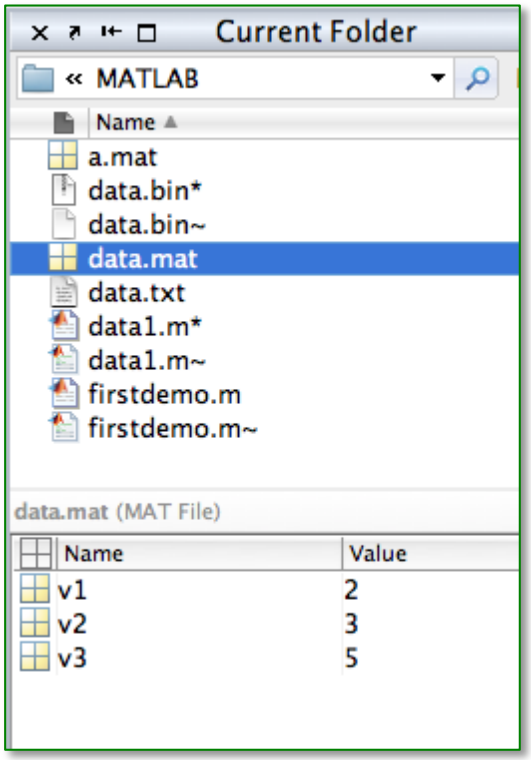
```
4D41 544C 4142 2035 2E30 204D 4154 2D66
696C 652C 2050 6C61 7466 6F72 6D3A
204D 4143 4936 342C 2043 7265 6174 6564
206F 6E3A 2053 756E 2041 7072 2032 3920
3233 3A31 343A 3439 2032 3031 3220 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 0000 0000 0000 0000 0001
494D 0F00 0000 2400 0000 789C E363 6060
3000 6236 20E6 80D2 20C0 0AE5 33C2 3113
4399 2103 9004 B118 1800 1A11 0111 0F00
0000 2500 0000 789C E363 6060 3000 6236
20E6 80D2 20C0 0AE5 33C2 3113 4399 1103
9064 6460 06F2 011A 2001 130F 0000 0024
0000 0078 9CE3 6360 6030 0062 3620 E680
D220 C00A E533 C231 1343 9931 0390 6404
CB01 001A 3301 16
```

```
% sauve v1 v2 v3 dans data.txt au format ascii
(texte)
% le nom et la structure des variables sont perdus!
>> save data.txt v1 v2 v3 -ascii
```

data.txt

```
2.0000000e+00
3.0000000e+00
5.0000000e+00
```

Inspect le contenur de fichiers .mat

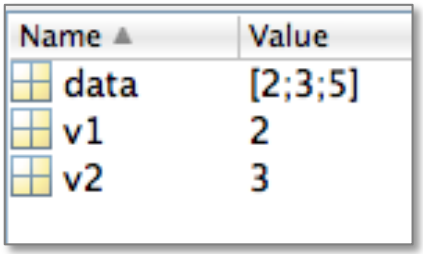


Charger des données

De manière simétrique à la sauvegarde de données, il est possible de charger des données dans le *workspace*

Attention les variables du *workspace* seront écrasées par le contenu du fichier chargé!

```
% charge l'entier du contenu de data.mat dans le workspace
>> load ('data.mat')

% charge 'v1' et 'v2' de data.mat dans le workspace
>> load ('data.mat','v1','v2')

% charge l'entier du contenu de data.txt dans le workspace
% sous le nom 'data'
>>load ('data.txt' '-ascii')
```



Charger des données

Il est aussi possible de chargez des données dans le *workspace* en exécutant un scritp (.m)

Attention les variables du *workspace* ayant le même nom que les variables du script seront écrasées!

```
% Exécute le script 'myScript.m' le workspace peut être modifié
>> run('/path/to/myScript.m')
```

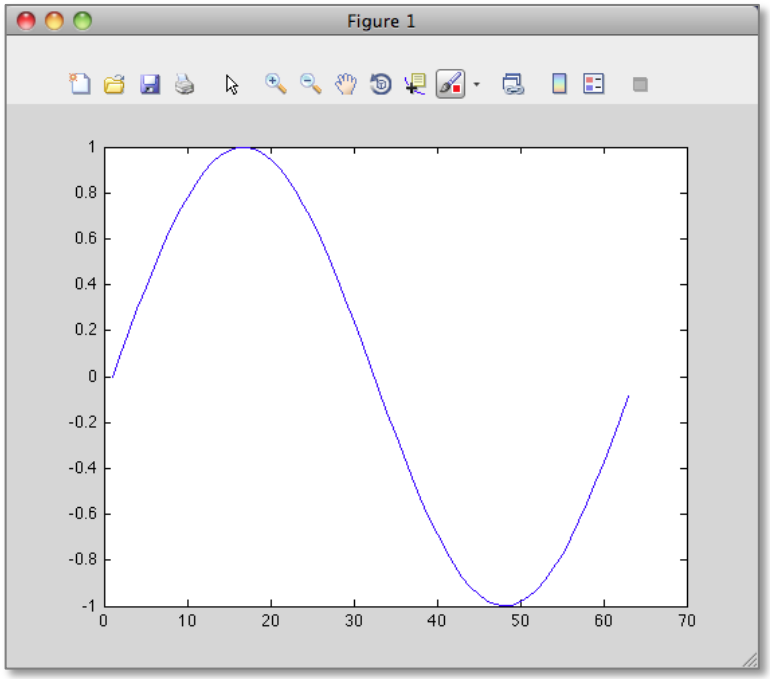
Si le script se trouve dans le dossier courant, il est possible de simplement taper le nom de celui-ci dans la ligne de commande.

```
>> myScript
```

Very simple plot

La commande `plot` permet de créer une figure à partir de données nuériques.

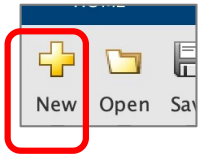
```
>> a = [0:0.1:2*pi]
>> b = sin(a);
>> plot(b);
```



Une fois le graphique affiché dans la figure, il est possible de modifier ses caratéristiques de manière interactive.

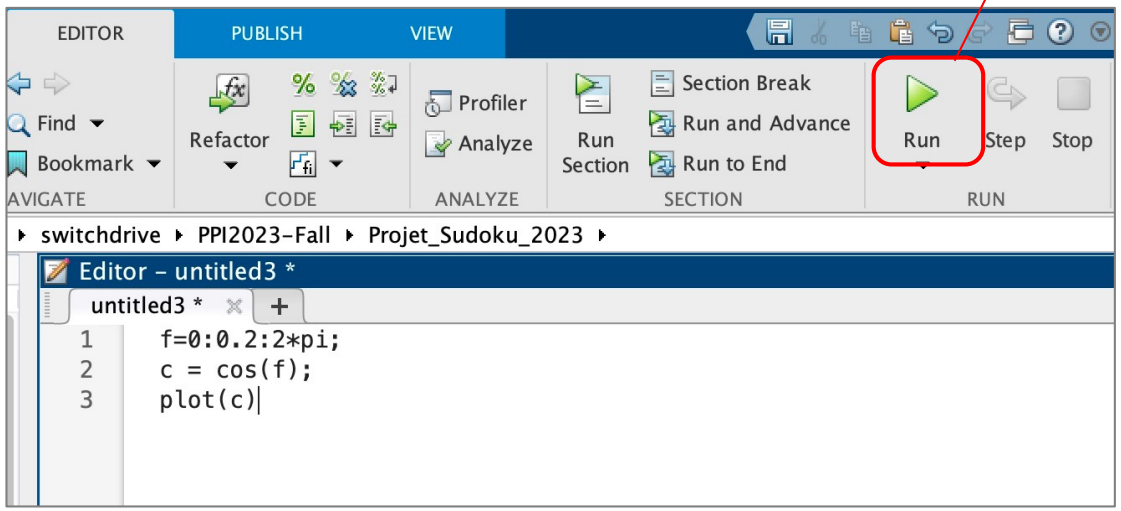
Programme Matlab

Les opérations entrées de manière séquentielle dans la fenêtre de commandes de Matlab peuvent être lue depuis un fichier avec l'extension `.m`



Crée un nouveau fichier

Exécute le fichier



Les variables du fichier sont sauvées dans le *workspace*!

Matlab 1 recap

- Matlab mode interactif
- Repenser les problèmes en fonction de matrices/vecteurs et éviter les boucles.
- Les fonctions sont polymorphiques
- **Les indices commencent à 1**
- Vecteur ligne [1 2 3]
- Vecteur colonne [1; 2; 3]
- 1^{e} ligne d'une matrice M(1,:)
- 1^{e} colonne d'une matrice M(:,1)

Exam Q.

$$\sin(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

Calculez *sin(x)* en employant le développement en série ci-dessus. Votre code calcule les *n* premiers éléments de la série, avec *n* = 100. **Votre code ne doit pas contenir de boucle**. Décomposez votre code en plusieurs séries qui seront combinées pour obtenir le résultat final. Le calcul de factorielle se fait à l'aide de la fonction matlab *factorial(x)*.

Exam Q.

$$\sin(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{(2i+1)}}{(2i+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

```
n=100;
x=pi/2;
idx = 0:n;
s=(-1).^idx;
p= 2*idx+1;
num =s.* x.^p
den = factorial(p);
mySIN = sum(num./den)
```