Chapitre 6 : Matlab comme Solveur Numérique et Symbolique

Université Alger1, Dept MI

2° année Maths, Semestre 3, 2016

Matière : Outils de Programmation 2

Contact : fodil.laib@hotmail.com

Fonctions Temporaires

On peut définir des fonctions temporaires à la ligne de commande à l'aide des commandes **inline** **ou** **@**

```
>> f1 = inline('2*x + 5')
f1 =
     Inline function:
     f1(x) = 2*x + 5
>> f1(2)
ans =
     9
>> t = [ 2 0  1];
>> f1(t)
ans =
     9     5     7
>> f2 = inline('sin(x.^2) + exp(y)')
f2 =
     Inline function:
     f2(x,y) = sin(x.^2) + exp(y)
>> f2(5,3)
ans =
   19.9532
```

On peut spécifier l'ordre des variables :

```
>> f3 = inline('theta^2 * x ')
f3 =
     Inline function:
     f3(theta,x) = theta^2 * x
>> f3(5,2)
ans =
    50
>> f4 = inline('theta^2 * x ', 'x', 'theta')
f4 =
     Inline function:
     f4(x,theta) = theta^2 * x
>> f4(5,2)
ans =
    20
```

...

Une fonction anonyme, définie avec @ peut utiliser les valables globales

```
>> a = 2;
>> b = 5;

>> g1 = @(x,y) a*x - b*y
g1 =
    @(x,y)a*x-b*y

>> g1(2,3)
ans =
   -11
```

Essayons la même fonction avec inline pour voir la diffrence :

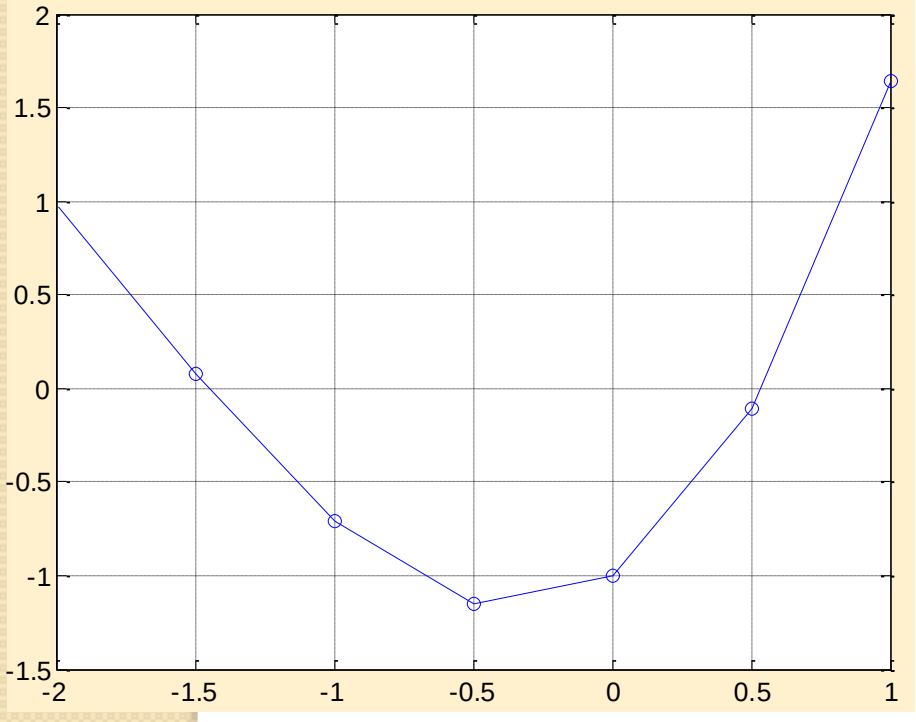
```
>> f5 = inline('a*x - b*y', 'x','y')
f5 =
     Inline function:
     f5(x,y) = a*x - b*y

>> f5(2,3)
Error in inline expression ==> a*x - b*y
 Undefined function or variable 'a'.
```

Zéro d'une Fonction

Soit f(x) une fonction définie sur un intervalle donné.

La fonction Matlab *fzero* retrouve la valeur x*, si elle existe, telle que f(x*) = 0

```
>> f = inline('exp(x) - 2 * cos(x)');
>> x = -2 : 0.5 : 1;
>> plot(x,f(x),'-o'), grid
```



Recherche de la 1° racine :

On spécifie une valeur de départ (x0 = 0,5)

```
>> fzero(f, 0.5 )
ans =
    0.5398
```

Ou bien, on spécifie un intervalle de recherche [0,1]

```
>> fzero(f, 0, 1)
ans =
    0.5398
```

Recherche de la 2° racine :

On specifie une valeur de départ x0 = -2 :

```
>> fzero(f,-2)
ans =
   -1.4537
```

Ou bien, on specifie l'intervalle [-2, -1] :

```
>> fzero(f,-2,-1)
ans =
   -1.4537
```

Résolution d'un Système d'Equations Nonlineaires

Soir à résoudre le système

$$\begin{cases} 4x^2 + 6y^2 - 10x + 5y - 15 = 0 \\ x^2 - 8y^2 + x + 3y - 4 = 0 \end{cases}$$

D'abord, on définit la fonction correspondante dans un fichier f2.m :

```
function z = f2(t)

x = t(1);
y = t(2);

z(1) = 4*x .^2 + 6*y .^2 - 10*x + 5*y - 15;
z(2) = x.^2 - 8*y.^2 + x + 3*y -4;
```

Puis, on choisit une solution de départ :

```
>> S0= [2 3 ];
```

Finalement, on fait appel à la fonction *fsolve* pour résoudre le système d'équations :

```
>> S = fsolve(@f2, S0)
S =
    2.7287    1.0858
```

Optimisation Nonlinéaire

La fonction *fminbnd*, resp *fminsearch*, recherche le minimum d'une fonction à une seule variable, resp plusieurs variables.

Exemple 1 :

$$\min f(x) = x^2 - 5x + 1$$

dans l'intervalle [1, 4] :

```
>> fminbnd( @(x)  x^2 - 5*x + 1 ,  1,  4)
ans =
    2.5000
```

Exemple 2 :

$$\min f(x,y) = x^2 + y^2 + 2xy$$

avec une solution de départ $[x_0,y_0] = [0, 1]$:

```
>> f2 = @(x) x(1)^2 + x(2)^2 + 2*x(1)*x(2) ;
>> fminsearch (f2 , [0;1])
ans =
  -0.47968
   0.47968
```

fmincon(f, x0, A, b) permet de résoudre, entre autres, le programme suivant :

$$\begin{cases} \min f(x) \\ \text{avec} \\ Ax \leq b \end{cases}$$

Exemple 3 :

$$\begin{cases} \min\ f(x) = -2x_1x_2 - 5{x_3}^2 \\ \text{avec} \\ 3x_1 + x_2 + 4x_3 \leq 30 \\ 5x_1 \quad + \quad x_3 \leq 25 \end{cases}$$

```
>> fun1 = @(x) - 2*x(1)*x(2) - 5*x(3)^2
>> A = [3 1 4; 5 0 1];
>> b = [30 25 ];
>> x0 = [1 1 1]  % solution de départ
>> x = fmincon(fun1,x0,A, b)
x =
  1.0e+139 *
   -1.0711   -1.8518    1.2663
```

Ajustement d'un Nuage de Points

La fonction *p = polyfit(x,y,n)* determine le polynome p d'ordre n qui ajuste au mieu le nuage de points (x,y)

Soit le nuage de points suivant :

```
>> x = [ -3.14   -2.44   -1.74   -1.04   -0.34
0.34    1.04    1.74    2.44    3.14 ];
>> y =[-0.00   -0.64   -0.98   -0.86   -0.34
0.34    0.86    0.98   0.64    0.00];
```

Déterminons le polynome d'ordre 3 qui represente au mieux ce nuage

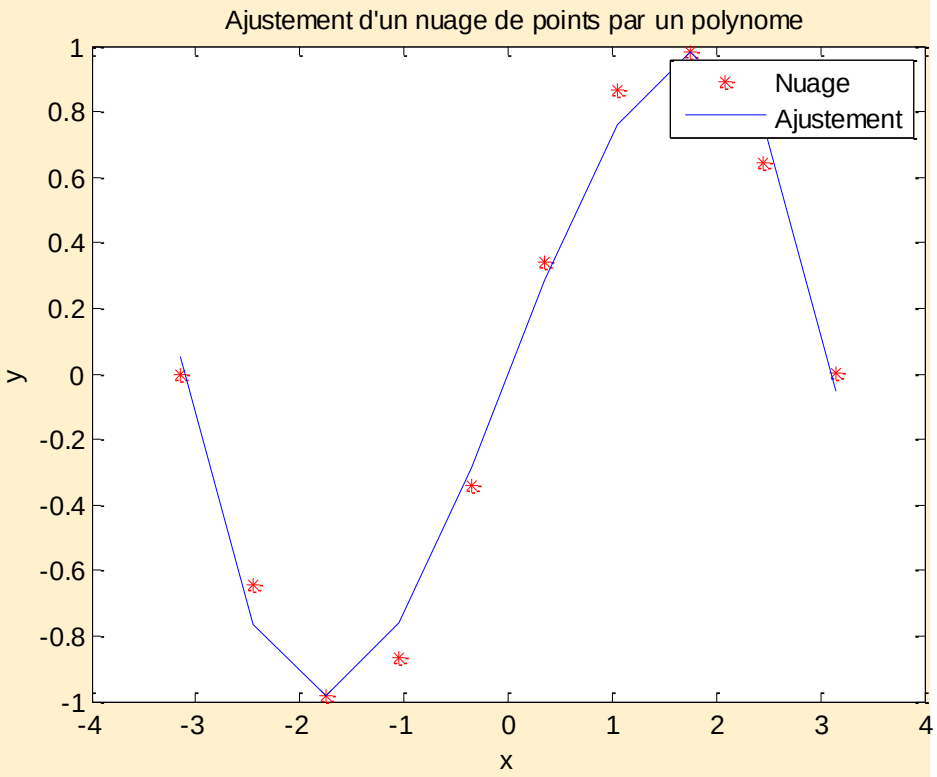
```
>> p = polyfit(x,y,3)
p =
   -0.0850    0.0000    0.8214   -0.0000
```

donc le polynome trouvé est :

$$P(x) = -0.085\,x^3 + 0.821\,x$$

Vérification : Dessinons le nuage de points (x,y) ainsi que le polynôme d'ajustement p :

```
>> plot(x,y,'r*')
>> hold on
>> plot(x,y1,'b')
>> legend('Nuage','Ajustement')
```



Variables Symboliques

Pour connaitre la version utilisée de Matlab et les toolbox installés, on tape la commande *ver* :

```
>> ver
-----------------------------------------------
MATLAB Version: 7.14.0.739 (R2012a)
…
-----------------------------------------------
MATLAB                          Version 7.14
Simulink                        Version 7.9
…
Symbolic Math Toolbox           Version 5.8
```

La commande *syms* permet de définir des variables symboliques. Elle est définie dans le toolbox *Symbolic Math Toolbox* :

```
>> syms x
>> a = 2;
>> whos
Name    Size    Bytes    Class
a       1x1     8        double
x       1x1     60       sym
>> a * a
ans =
     4
>> x * x
ans =
x^2
>> x / x^2
ans =
1/x
```

Fonctions Symboliques

Pour créer une fonction symbolique :

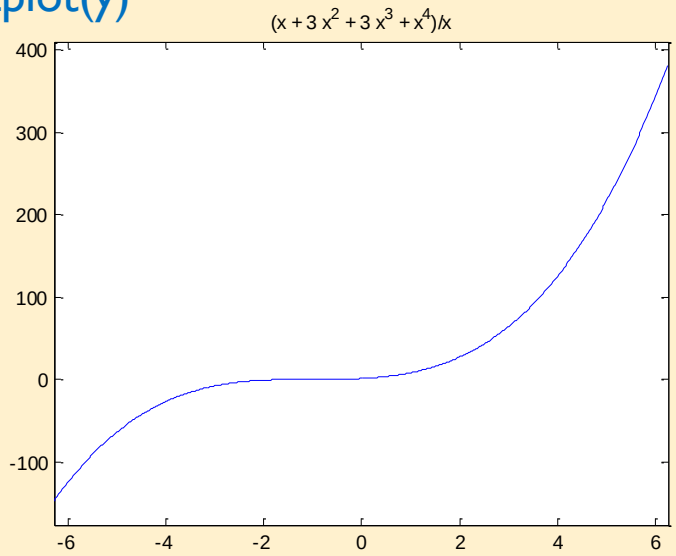
```
>> syms x
>> y = (x^4 + 3 * x^3 + 3* x^2 + x)/x
y =
(x^4 + 3*x^3 + 3*x^2 + x)/x
```

Pour évaluer cette fonction au point x = 2 :

```
>> subs(y,2)
ans =
    18
```

Pour déssiner la courbe de y :

```
>> ezplot(y)
```



Par defaut, *ezplot* prend l'intervalle , $[-2\pi, 2\pi]$.

Afin d'avoir un autre intervalle, eg [-10, 25], on tape : `>> ezplot(y, [-10, 25])`

Pour afficher y sous une forme plus lisible :

```
>> pretty(y)
```

$$\frac{x^4 + 3x^3 + 3x^2 + x}{x}$$

Pour simplifier la la forme de y :

```
>> simple(y)
ans =
(x + 1)^3
```

D'autres fonctions symboliques :

factor, *expand*, *finverse*, *simplify*, …

Calcul des Limites

Soit à calculer la limite

$$\lim_{n\to 3} x^2 + 5$$

```
>> syms x
>> f = x^2 +5;
>> limit(f, 3)
ans =
 14
```

Si x → 0, pas besoin de le spécifier :

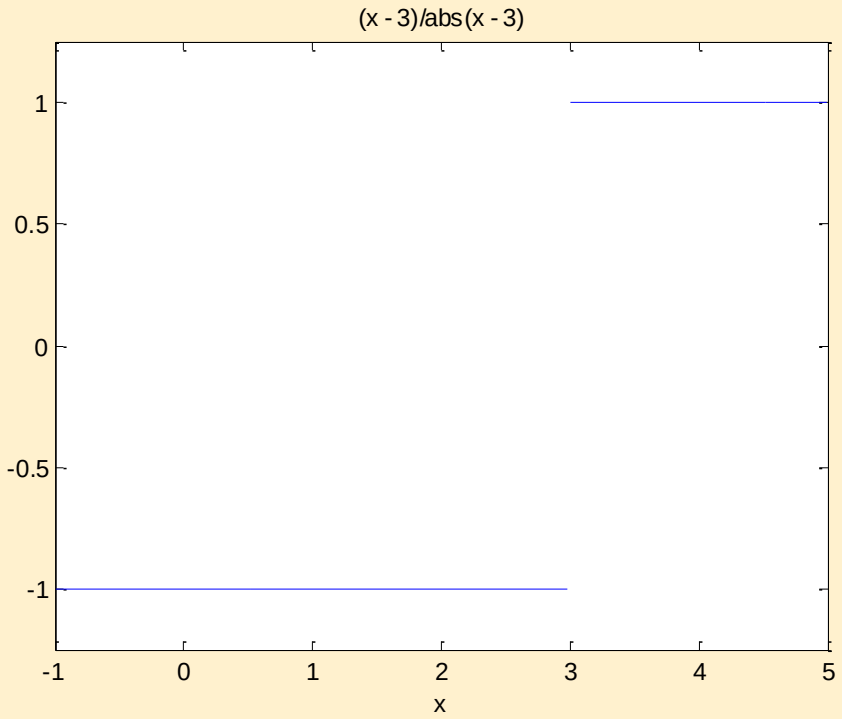
$$\lim_{n\to 0} \frac{x^3 + 5}{x^4 + 7}$$

```
>> g = (x^3+5) / (x^4 + 7)
>> limit(g)
ans =
 5/7
```

Equivalent à `>> limit(g,0)`

Limites à droite et à gauche :

```
>> f = (x - 3)/abs(x-3);
>> ezplot(f,[-1,5])
```




```
>> limit(f,x,3,'left')
ans =
-1
>> limit(f,x,3,'right')
 ans =
  1
```

Calcul des Dérivées

Soit la fonction : $f(t) = 3t^2 + \frac{2}{t^2}$

```
>> syms t
>> f = 3*t^2 + 2*t^(-2);
```

La dérivée de f par rapport à t est calculée par:

```
>> diff(f)
ans =
6*t - 4/t^3
```

On utilise la syntaxe *diff(f,n)* pour avoir les dérivées d'orders superieurs.

La dérivée seconde de f est :

```
>> diff(diff(f))
ans =
12/t^4 + 6
```

ou bien, de manière directe :

```
>> diff(f,2)
 ans =
 12/t^4 + 6
```

Dérivées de fonctions à plusieurs variables :

```
>> syms x y
>> f = sin( x * y )
f =
sin(x*y)
```

La dérivée de f par rapport à y est

```
>> diff(f,y)
ans =
x*cos(x*y)
```

La dérivée par rapport à x est :

```
>> diff(f,x)
ans =
y*cos(x*y)
```

Pour calculer la dérivée 2nd de f par rapport y, on entre :

```
>> diff(f,y,2)
 ans =
 -x^2*sin(x*y)
```

Calcul des Intégrales

Intégrale Indéfinie :

$$F(x) = \int f(x)dx$$

La fonction *int* calcule l'intégrale indéfinie :

```
>> syms x
>> f = 2*x^2 + 3*x + 1;
>> F = int(f)
F =
(x*(4*x^2 + 9*x + 6))/6

>> expand(F)
ans =
(2*x^3)/3 + (3*x^2)/2 + x

>> pretty(F)
       2
 x (4 x  + 9 x + 6)
 ------------------
        6
```

Intégrale définie :

$$I = \int_a^b f(x)dx$$

se calcule par *int(f,a,b)* :

```
>> f = x^3 - 2*x +5;
>> I = int(f, 1, 2)
 I =
 23/4
```

Equations Différentielles

dsolve permet de résoudre des équations différentielles.

y' sera désigné par Dy,

y" par D2y, …

Par exemple, l'équation:

$$y' = t + 1$$

sera notée : Dy = t + 1

```
>> y = dsolve('Dy = t+1')
 y =
 C2 + (t*(t + 2))/2
 >> pretty(y)
            t (t + 2)
  C2 +  ---------
                2
```

Vérification :

```
>> diff(y)
ans =
t + 1
```

S'il y a une condition initiale y(0) = 2 :

```
>> y = dsolve('Dy = t+1','y(0)=2')
y =
(t*(t + 2))/2 + 2
```

Equation d'ordre 2 :

Soit à résoudre l'équation $y'' = t + 1$

```
>> y = dsolve('D2y = t + 1', 'y(0)= 5','y(1)=10')
y =
(t*(t^2 + 3*t + 26))/6 + 5

>> pretty(y)
       2
  t (t  + 3 t + 26)
  ----------------- + 5
          6
```

Résolution d'Equations

La fonction *solve* résout des équations ou systèmes d'équations algébriques :

Soit à résoudre l'équation $ax - b = 0$

On peut procéder de plusieurs façons :

```
>> clear all
>> solve('a*x + b = 0')
ans =
-b/a
```

ou bien

```
>> syms x a b
>> f = a*x + b ;
>> x0 = solve(f==0)
 x0 =
 -b/a
```

Vérification :

```
>> subs(f,x,x0)
 ans =
 0
```

Système d'équations :

Soit à résoudre le système :

$$\begin{cases} ax + 9y - 5 = 0 \\ bx - 6y - 4 = 0 \end{cases}$$

```
>> syms x y a b
>> f1 = a*x+9*y-5 ;    f2 = b*x -6*y – 4 ;
>> [x0,y0] = solve(f1,f2)
x0 =
22/(2*a + 3*b)
y0 =
-(4*a - 5*b)/(3*(2*a + 3*b))
```

Vérification :

```
>> subs(f1,{x,y},{x0,y0})
 ans =
 (22*a)/(2*a + 3*b) - (3*(4*a - 5*b))/(2*a + 3*b) - 5
 >> simplify(ans)
 ans =
 0
 >> simplify( subs(f2,{x,y},{x0,y0}) )
 ans =
 0
```