Modélisation et résolutions numérique et symbolique de problèmes via les logiciels Maple et MATLAB (MODEL)

Cours n°6 : Introduction à MATLAB

Stef Graillat

Université Pierre et Marie Curie (Paris 6)

Objectifs du cours

Objectifs :

- Concept mathématique : définition mathématique de concepts et de quantités
- Algorithme : comment calculer efficacement ces quantités sur ordinateur (via l'utilisation de MATLAB et Maple) ?
- Résolution de problème : utiliser les concepts et les algorithmes pour résoudre des problèmes concrets

Références principales

- Scientific Computing with Case Studies, Dianne P. O'Leary, SIAM, 2009
- Numerical Computing with MATLAB, Cleve Moler, SIAM, 2004
- MATLAB Guide, Desmond J. Higham, Nicholas J. Higham, 2nd édition, SIAM, 2005
- Solving Problems in Scientific Computing Using Maple and MATLAB, Walter Gander, Jiri Hrebicek, 4e édition, Springer, 2004
- Numerical Recipes. The Art of Scientific Computing, William Press, Saul Teukolsky, William Vetterling et Brian Flannery, 3rd Edition, Cambridge University Press, 2007

Champs d'application

Les notions vues dans ce cours interviennent dans :

- la robotique
- le traitement du signal
- le traitement d'image
- la géométrie algorithmique
- la biologie
- etc.

Plan du cours

- Introduction à l'arithmétique à virgule flottante et présentation de MATLAB
- Décomposition en valeurs singulières, application à la compression d'image
- Calcul de vecteurs propres, valeurs propres, application à l'algorithme PageRank de Google
- Transformée de Fourier discrète, application en traitement du signal et en calcul formel
- Méthode de Monte-Carlo, application au pricing doption en finance

Calcul formel / calcul numérique

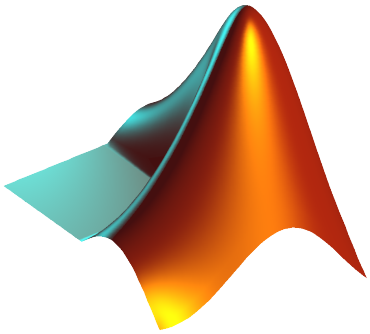
- Algorithmes symboliques : fournissent une représentation exacte du résultat mais peuvent être lents
- Algorithmes numériques : exécutés en précision finie, ils sont plus rapides en général mais fournissent des résultats approchés

$\rightarrow$ calcul symbolique-numérique : utiliser quand cela est possible des algorithmes numériques (au sein d'algorithmes symboliques) en contrôlant la précision des résultats

Logiciels utilisés

Calcul numérique

- MATLAB
- SciLab
- Octave
- FreeMat



Calcul formel

- Maple
- Sage
- Maxima
- Mathematica
- Magma

1. Arithmétique à virgule flottante

Peut-on compter jusqu'à 6 avec un ordinateur ?

$2-1$		1.000000000000000
$\left(\frac{1}{\cos(100\pi+\pi/4)}\right)^2$		2.000000000000111
$3\frac{\tan(\arctan(10000))}{10000}$		2.999999999997162
$\left(\left(\cdots\left(\sqrt{\sqrt{\cdots\sqrt{4}}}\right)^2\cdots\right)^2\right)^2$	(20 fois)	4.000000000629434
$5\times\left\{\frac{(1+e^{-100})-1)}{(1+e^{-100})-1)}\right\}$		`NaN`
$\frac{\log(e^{6000})}{1000}/$		`Inf`

Encore plus grave

- 1994 : Bug de la division du Pentium, 4195835.0/3145727.0 donnait 1.333739068902037589 au lieu de 1.333820449136241002
- Excel, version 3.0 à 7.0, entrez 1.40737488355328 vous obtiendrez 0.64
- Excel 2007 (premières versions), calculez $65535-2^{-37}$, vous obtiendrez 100000

Nombres à virgule flottante

Un nombre flottant normalisé $x \in \mathbb{F}$ est un nombre qui s'écrit sous la forme

$$x = \pm \underbrace{x_0.x_1 \ldots x_{p-1}}_{\textit{mantisse}} \times b^e, \quad 0 \leq x_i \leq b-1, \quad x_0 \neq 0$$

b : la base, p : précision, e : exposant vérifiant $e_{\min} \leq e \leq e_{\max}$

Précision machine $\epsilon = b^{1-p}$, $|1^+ - 1| = \epsilon$

Approximation de $\mathbb{R}$ par $\mathbb{F}$, arrondi $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$
Soit $x \in \mathbb{R}$ alors

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \leq \mathbf{u}.$$

L'unité d'arrondie $\mathbf{u}$ vaut $\mathbf{u} = \epsilon/2$ pour l'arrondi au plus près.

Modèle standard de l'arithmétique à virgule flottante

Norme IEEE 754 (1985)

- Les opérations arithmétique ops $(+, -, \times, /, \sqrt{})$ sont effectuées comme si elles étaient calculées en précisions infinies puis arrondies ensuite
- Par défaut : arrondi au plus près

Type	Taille	Mantisse	Exposant	Unité d'arrondi	Intervalle
Simple	32 bits	23+1 bits	8 bits	$\mathbf{u} = 2^{-24} \approx 5,96 \times 10^{-8}$	$\approx 10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{-53} \approx 1,11 \times 10^{-16}$	$\approx 10^{\pm 308}$

Soient $x, y \in \mathbb{F}$,

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta), \quad |\delta| \leq \mathbf{u}, \quad \circ \in \{+, -, \cdot, /\}$$

Exceptions

L'arithmétique est "fermée" : chaque opération retourne un résultat.

Exception	Résultats
Invalid operation	NaN (Not a Number)
Overflow	$\pm\infty$
Divide by zero	$\pm\infty$
Underflow	Nombres dénormalisés
Inexact	Résultat arrondi correctement

NaN est généré par les opérations telles que $0/0$, $0 \times \infty$, ∞/∞, $(+\infty) + (-\infty)$ et $\sqrt{-1}$.
Les symbôles infinis vérifient $\infty + \infty = \infty$, $(-1) \times \infty = -\infty$ et $(\mathit{fini})/\infty = 0$.

Précision des calculs

À chaque arrondi, on perd a priori un peu de précision, on parle d'erreur d'arrondi.

Même si une opération isolée retourne le meilleur résultat possible (l'arrondi du résultat exact), une suite de calculs peut conduire à d'importantes erreurs du fait du cumul des erreurs d'arrondi.

Les deux sources principales d'erreur d'arrondis au cours des calculs sont l'élimination et l'absorption.

Exemple du phénomène d'élimination

Soit $f(x) = (1 - \cos(x))/x^2$, alors $0 \leq f(x) < 1/2$ pour tout $x \neq 0$.
Avec $x = 1.2 \times 10^{-5}$, le cosinus arrondi à 10 chiffres significatifs vaut

$$c = 0.9999\ 9999\ 99,$$

donc

$$1 - c = 0.0000\ 0000\ 01$$

Par conséquent $(1 - c)/x^2 = 10^{-10}/1.44 \times 10^{-10} =$ 0.6944 ...!!!!

Pourtant la soustraction $1 - c$ est exacte.

Pour éviter l'élimination, réécrire f sous la forme

$$f(x) = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2$$

Exemple du phénomène d'absorption

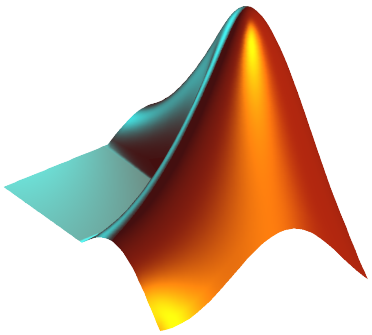
On calcule numériquement, pour de grandes valeurs de N, la somme :

$$\sum_{i=1}^{N} \frac{1}{i}$$

Résultats d'un programme C (flottant SP) sur un processeur Pentium4 :

ordre	N			
	10^5	10^6	10^7	10^8
exacte	1.209015e+01	1.439273e+01	1.669531e+01	1.899790e+01
$1 \to N$	1.209085e+01	1.435736e+01	1.540368e+01	1.540368e+01
$N \to 1$	1.209015e+01	1.439265e+01	1.668603e+01	1.880792e+01

2. Introduction à MATLAB



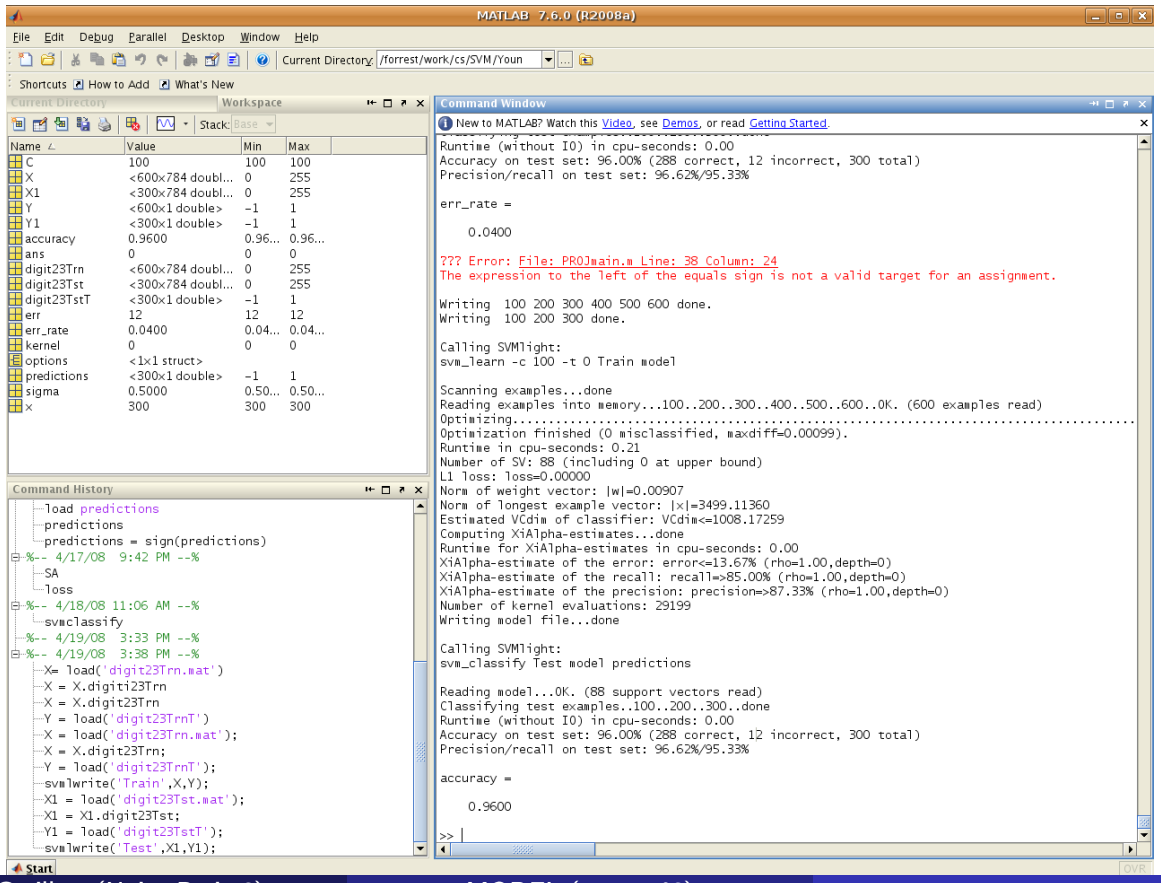
MATLAB

- MATLAB = MATrix LABoratory
- un langage de programmation et un environnement de développement
- MATLAB a été conçu par Cleve Moler à la fin des années 1970
- MATLAB est complété par de multiples boites à outils (toolboxes)
- le langage MATLAB supporte la POO
- Interaction possible avec les langages C et Fortran

- pour l'aide sur une commande `command`, utiliser `help command` ou `doc command`
- pour écrire des commentaires %

Référence : MATLAB Guide, Desmond J. Higham, Nicholas J. Higham, 2nd édition, SIAM, 2005

MATLAB (suite)



MATLAB (suite)

- MATLAB a été créé dans les années 70 par Cleve Moler alors professeur de mathématiques à l'Université du Nouveau-Mexique
- MATLAB a été crée à partir des bibliothèques Fortran, LINPACK et EISPACK
- MATLAB a ensuite évolué, en intégrant la bibliothèque LAPACK en 2000
- Il y a des alternatives libres à MATLAB telles que GNU Octave, FreeMat et Scilab
- La version actuelle de MATLAB est MATLAB R2012b (version 8.00)

Vecteurs et matrices en MATLAB

Les variables de MATLAB sont principalement des vecteurs et des matrices

- Les vecteurs :
  - vecteur ligne

```
>> format compact
>> x = [1.1 10.1 100.1]
x =
   1.1000 10.1000 100.1000
```

  - vecteur colonne

```
>> x = [1.1; 10.1; 100.1]
x =
   1.1000
  10.1000
 100.1000
```


Vecteurs en MATLAB (suite)

- Affichage et affectation

```
>> x = [1.1; 10.1; 100.1];
>> x
x =
   1.1000
  10.1000
 100.1000
>> x(3) = -1.1
x =
   1.1000
  10.1000
  -1.1000
```

Les vecteurs sont indicés à partir de 1 (et pas 0 comme en C)

Vecteurs en MATLAB (suite)

- Transposition d'un vecteur

```
>> x = [1.1 10.1 100.1]'
x =
    1.1000
   10.1000
  100.1000
```

- Pour entrer un vecteur ou une commande occupant plus d'une ligne

```
>> x = [0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 ...
    .55 .60 .65 .70 .75 .80 .85 .90 .95 1];
```

- Longueur d'un vecteur

```
>> length(x)
ans =
    21
```


Vecteurs en MATLAB (suite)

- Vecteur de taille quelconque (par défaut les vecteurs sont en ligne)

```
n = 20;
h = 1/n;
for k=1:n
   x(k) = k*h;
end
```

- Initialisation d'un vecteur colonne

```
x = zeros(n,1);
```

- les deux-points. La notation `a:b` dénote le vecteur ligne allant de `a` à `b` par pas de 1 tandis que pour `a:s:b` le pas est `s`

```
>> x = 1:5
x =
    1 2 3 4 5
>> x = 4:-1:0
x =
    4 3 2 1 0
```

Vecteurs en MATLAB (suite)

- `>> x = 0:0.05:1;  % vecteur à 21 composantes.`
  `>> x = 0.05*(0:20) % autre façon de générer le même vecteur.`
- Accéder à des parties d'un vecteur
  `x(1:4)` extrait les 4 premiers éléments de `x`
- `linspace(a,b,n)` produit un vecteur ligne avec n composantes qui divise [a,b] en $n-1$ intervalles égaux.

  ```
  x = linspace(0,1,21);
  ```


Polynômes en MATLAB

Un polynôme de degré n en MATLAB est représenté par un vecteur de taille $n+1$
Le polynôme $1+2x+3x^2$ est représenté par :

```
>> p = [3 2 1]
p =
     3 2 1
```

La fonction `polyval` permet d'évaluer un polynôme

```
>> polyval(p, 0)
ans =
     1
```

Polynômes en MATLAB

Un polynôme de degré n en MATLAB est représenté par un vecteur de taille $n+1$
Le polynôme $1 + 2x + 3x^2$ est représenté par :

```
>> p = [3 2 1]
p =
     3 2 1
```

La fonction `polyval` permet d'évaluer un polynôme

```
>> polyval(p, 0)
ans =
      1
```


Les matrices en MATLAB

La puissance de MATLAB provient de ses opérations matricielles

- rapides
- et précises (qualité numérique)

2 façons d'entrer des matrices

```
>> A = [1 2 3
        2 4 7
       -1 0 5]
A =
   1 2 3
   2 4 7
  -1 0 5
>> a = [1 2 3; 1 4 8; 3 -1 0]
a =
   1 2 3
   1 4 8
   3 -1 0
```

Les matrices en MATLAB

La puissance de MATLAB provient de ses opérations matricielles

- rapides
- et précises (qualité numérique)

2 façons d'entrer des matrices

```
>> A = [1 2 3
        2 4 7
       -1 0 5]
A =
   1 2 3
   2 4 7
  -1 0 5
>> a = [1 2 3; 1 4 8; 3 -1 0]
a =
   1 2 3
   1 4 8
   3 -1 0
```


Les matrices en MATLAB (suite)

Fonctions spéciales pour créer des matrices

- `zeros(m,n)` produit une matrice de 0 de taille $m \times n$

```
>> A = zeros(2,4)
A =
   0 0 0 0
   0 0 0 0
```

- `ones(m,n)` produit une matrice de 1 de taille $m \times n$

```
>> A = ones(3)
A =
   1 1 1
   1 1 1
   1 1 1
```

Les matrices en MATLAB (suite)

- `eye(n)` produit la matrice identité de taille n

```
>> A = eye(3)
A =
   1 0 0
   0 1 0
   0 0 1
```

- Résolution d'un système linéaire $Ax = b$

```
>> A = [1 2 3; 2 4 7; -1 0 5];
>> b = [1 1 1]';
>> x =A\b
x =
    -6
     5
    -1
```


Les matrices en MATLAB (suite)

- Autre utilisation des deux-points :

```
>> C = A([1 3],:)
C =
     1     2     3
    -1     0     5
```

- Fonctions « vectorisées »
  On veut évaluer une fonction sur un vecteur de valeurs x_i :
  $a = x_1 < x_2 < \cdots < x_n = b$.
  Les fonctions standards acceptent des vecteurs en arguments et renvoient un vecteur.

```
n=21;
x = linspace(0,2*pi,n);
y = cos(x);
```

Sauvegarder votre travail

- En général, on écrit la liste des commandes tapées dans un fichier texte (via l'éditeur `edit` intégré à MATLAB) ou via votre éditeur de texte préféré

- Pour sauvegarder des variables, utilisez la fonction `save`. Sauvegarder les variables `A`, `b` dans le fichier `svar.mat` se fait par :

  ```
  save svar A b
  ```

  On les recharge par la commande :

  ```
  load svar
  ```


Aide

- taper `help command`.

  ```
  >> help length
   LENGTH   Length of vector.
      LENGTH(X) returns the length of vector X.  It is
      equivalent to MAX(SIZE(X)) for non-empty arrays
      and 0 for empty ones.
  ```

- pour l'avoir en mode graphique, taper `doc`

  ```
  >> doc length
  ```

M-files

2 types de fichier (qui portent l'extension `.m`) :

- script M-files : ni entrée, ni sortie et utilise les variables de l'espace de travail
- fonction M-files : contient une fonction qui accepte des arguments en entrée et renvoie des arguments en sortie et les variables internes sont locales à la fonction

Exemple de fonction (à stocker dans un fichier `sumprod.m`) :

```
function [s,p]= sumprod(x)
   n = length(x);
   s=0;
   p=1;
   for i=1:n
      s = s + x(i);
      p = p*x(i);
   end
```


M-files (suite)

Structure d'une fonction M-files

1. le mot-clé `function`
2. liste des arguments en sortie (entre crochets `[ ]` s'il y en a plusieurs)
3. le symbole =
4. le nom de la fonction (qui doit être le même que le nom du fichier `.m`)
5. la liste entre parenthèse des entrées
6. le corps de la fonction

Pour éditer les fichiers, taper `edit`

Commandes utiles : `dir`, `ls`, `cd`, `type`, `lookfor`, `path`

Affichage

- Format numérique : commande `format` pour afficher des nombres en virgules fixes ou flottantes

```
>> format short, pi^4 % fixe, 5 chiffres
ans =
   97.4091
>> format short e, pi^4 % flottante, 5 chiffres
ans =
   9.7409e+001
>> format long, pi^4 % fixe, 15 chiffres
ans =
   97.40909103400242
>> format long e, pi^4 % flottante, 15 chiffres
ans =
   9.740909103400242e+001
```


Affichage (suite)

- Affichage de chaînes de caractère
  - la commande `disp` permet d'afficher une chaîne de caractère ou une variable

```
>> x=3;
>> disp(x);
     3
>> disp('test');
test
```

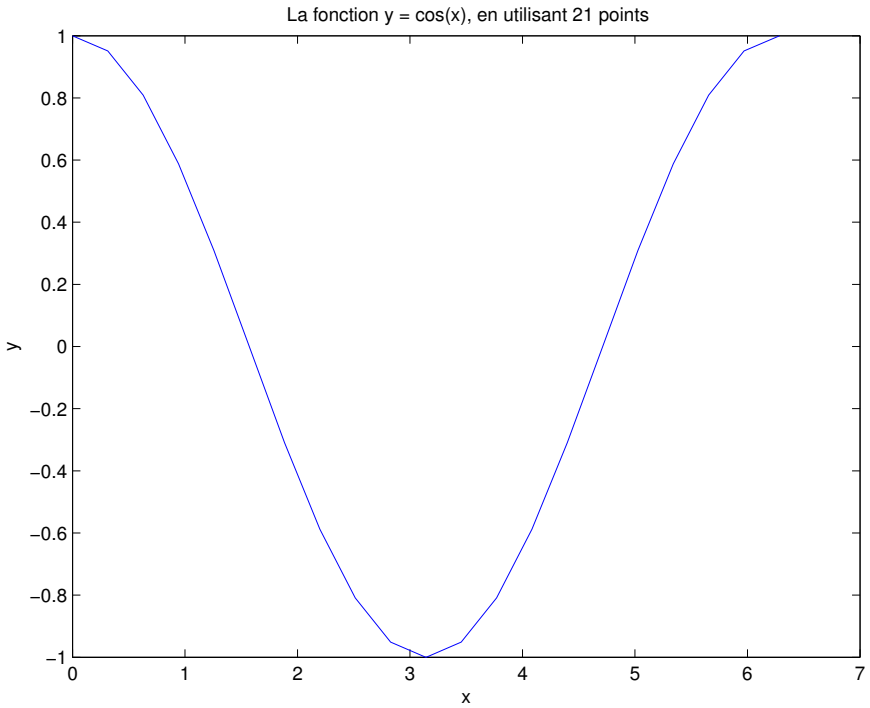
  - la commande `fprintf` permet d'afficher les chaînes de caractère et des chiffres (similaire à la commande `printf` de C)

Graphiques

Tracé d'une fonction $y = f(x)$ sur un intervalle $[a, b]$

```
n = 21;
x = linspace(0,2*pi,n);
y = cos(x);
plot(x,y)
title('La fonction y = cos(x), en utilisant 21 points')
xlabel('x')
ylabel('y')
```


Graphiques



Structures de contrôle et tests

- boucle `for`

```
for variable = expression
   instructions
end
```

- boucle `while`

```
while expression
   instructions
end
```

- test `if`

```
if expression
   instructions
end
```

```
if expression
   instructions
else
   instructions
end
```


Structures de contrôle et tests

- opérations relationnelles

==	égal
~=	différent
<	strictement inférieur
>	strictement supérieur
<=	inférieur ou égal
>=	supérieur ou égal

- opérations logiques

&	et
|	ou
~	non

Symbolic Math Toolbox

- La toolbox contient le noyau de MuPAD et des fonctions MATLAB qui communiquent avec ce noyau
- Nouveau type d'objet : les objets `sym` créés par les commandes `sym` et `syms`
- Créer une variable symbolique `x`

```
>> syms x
```


Symbolic Math Toolbox (suite)

- Manipulation symbolique d'expression en `x`

```
>> f = 1/(1+x^2)
f =
   1/(1+x^2)

>> g = int(f) % intégration
g =
   atan(x)

>> diff(g) % dérivation
ans =
   1/(1+x^2)

>> syms a
>> y = solve(f-a) % résoud f(x)-a=0
y = [ 1/a*(-a*(-1+a))^(1/2)]
    [ -1/a*(-a*(-1+a))^(1/2)]
```

Symbolic Math Toolbox (suite)

- Arithmétique multiprécision : fonction `vpa`

```
>> digits % par défaut
Digits = 32
>> vpa('sqrt(2)')
ans =
1.4142135623730950488016887242097
>> digits(50)
>> vpa('sqrt(2)')
ans =
1.4142135623730950488016887242096980785696718753769
```

- Arithmétique exacte : on travaille avec des expressions symboliques

```
>> z = sym('sqrt(2)')
z =
sqrt(2)
>> z^2-2
ans =
0
```


Symbolic Math Toolbox (suite)

Conversion polynôme symbolique polynôme MATLAB

```
>> syms x
>> sym2poly(2+x)
ans =
     1 2
>> poly2sym([1 2], x)
ans =
     2+x
```

Possibilité de conversion d'un symbolique en fonction MATLAB

```
>> syms x y z
r = x^2 + y^2 + z^2;
f = matlabFunction(log(r)+r^(-1/2),'file','myfile');
```

De cette façon on crée le fichier `myfile.m` qui contient la fonction MATLAB qui prend en entrée x, y, z et renvoie $\log(r) + r^{-1/2}$

Symbolic Math Toolbox (suite)

Conversion polynôme symbolique polynôme MATLAB

```
>> syms x
>> sym2poly(2+x)
ans =
     1 2
>> poly2sym([1 2], x)
ans =
     2+x
```

Possibilité de conversion d'un symbolique en fonction MATLAB

```
>> syms x y z
r = x^2 + y^2 + z^2;
f = matlabFunction(log(r)+r^(-1/2),'file','myfile');
```

De cette façon on crée le fichier `myfile.m` qui contient la fonction MATLAB qui prend en entrée x, y, z et renvoie $\log(r) + r^{-1/2}$

MATLAB à l'ARI

- Il y a 16 licences MATLAB à l'ARI. Il s'agit de MATLAB R2009a (version 7.8) avec la Symbolic Math Toolbox.

  Pour lancer MATLAB, il suffit de taper `matlab` dans un terminal

- Pour avoir un aperçu des fonctionnalités de MATLAB, taper `demo`