Formation Matlab et Calcul Scientifique

Cours 3: Résolution des equations et Interpolation

Moncef Mahjoub

ENIT-LAMSIN, BP 37, 1002 Tunis Belvedere, Tunisia

Plan

Plan

Système linéaire

- Soit système linéaire suivant

$$\begin{cases} x + 2y - 3z &= 5 \\ -3x - y + z &= -8 \\ x - y + z &= 0 \end{cases}$$

- Ecrire le système matriciel associé

```
» A=[1 2 -3;-3 -1 1;1 -1 1];
» b=[5;-8;0];
```

- Résoudre le système en une seule ligne

```
» x=A\b;
```

- Le symbole \ s'applique à des système carré et rectangulaire
- Pour les systèmes rectangulaires, donne la solution des moindres carrés

Algèbre linéaire matricielle

- Introduire la matrice suivante

```
» mat=[1 2 -3;-3 -1 1;1 -1 1];
```

- Calculer le rang de cette matrice

```
» r=rank(mat);
```

- Calculer le déterminant (matrice carrée)

```
» d=det(mat);
```

- Calculer l'inverse

```
» E=inv(mat);
```

- x = A \ b est la même que x = inv(A) * b

Décomposition des matrices

- MATLAB a intégré les méthodes de décomposition des matrices
- Les plus courants sont :
  - Décomposition en valeurs propres

```
» [V,D]=eig(X)
```

  - Décomposition en valeurs singulières

```
» [U,S,V]=svd(X)
```

  - Décomposition QR

```
» [Q,R]=qr(X)
```

Exercices

- Résoudre les systèmes suivants

$$\begin{cases} x + 4y &= 34 \\ -3x + y &= 2 \end{cases}$$

$$\begin{cases} 2x - 2y &= 4 \\ x + y &= 3 \\ 3x + 4y &= 2 \end{cases}$$

Exercices

- Résoudre les systèmes suivants

$$\begin{cases} x + 4y &= 34 \\ -3x + y &= 2 \end{cases}$$

```
» A=[1 4;-3 1];
» b=[34;2];
» rank(A)
» x=inv(A)*b;
```

$$\begin{cases} 2x - 2y &= 4 \\ x + y &= 3 \\ 3x + 4y &= 2 \end{cases}$$

```
» A=[2 -2;-1 1;3 4];
» b=[4;3;2];
» rank(A)
```

- rectangular matrix

```
» x1=A\b;
```

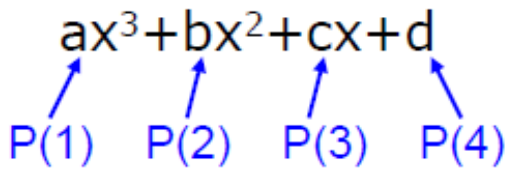
- gives least squares solution

```
» error=abs(A*x1-b)
```

Plan

Polynômes

- Plusieurs fonctions peuvent être décrits par un polynôme d'ordre élevé
- MATLAB représente un polynôme par un vecteur des coefficients. Si P est un polynôme, alors



- P = [1 0 -2] représente le polynôme $x^2 - 2$
- P = [2 0 0 0] représente le polynôme $2x^3$

Opérations sur les polynômes

- Un polynôme de longueur N+1 est de degré N
- Racines d'un polynôme
  `>> r = roots(P).`
  r est un vecteur de longueur N
- On peut reconstruire un polynôme à partir de ses racines
  `>> P = poly(r)`
  r est un vecteur de longueur N
- Evaluer un polynôme en un point
  `>> y0 = polyval(P,x0)`
  x0 et y0 sont des scalaires
- Evaluer un polynôme sur un ensemble de points
  `>> y = polyval(P,x)`
  x est un vecteur ; y est un vecteur de même taille

Interpolation Polynômiale

- MATLAB interpole des données par un polynôme
- Soit les vecteurs X = [-1 0 2] et Y = [0 -1 3]
  >> p2 = polyfit(X,Y,2) ;
  • détermine la meilleur interpolation polynômiale de degré 2 aux points (-1,0), (0,-1) et (2,3)
- Voir help polyfit pour plus d'informations

```
» plot(X,Y,'o', 'MarkerSize', 10);
» hold on;
» x = -3:.01:3;
» plot(x,polyval(p2,x), 'r--');
```

Exercice

- Evaluer $y = x^2$ pour x=-4 :0.1 :4
- Ajouter un bruit aux données. Utiliser randn. Tracer les données bruitées avec le maqueur $\cdot$
- Interpoler les données bruitées par un polynôme de degré 2
- Tracer le résultat et la donnée initiale dans une même figure avec des couleurs différentes

Exercice

- Evaluer $y = x^2$ pour x=-4 :0.1 :4

```
» x=-4:0.1:4;
» y=x.^2;
```

- Ajouter un bruit aux données. Utiliser randn. Tracer les données bruitées avec le maqueur ·

```
» y=y+randn(size(y));
» plot(x,y,'.');
```

- Interpoler les données bruitées par un polynôme de degré 2

```
» p=polyfit(x,y,2);
```

- Tracer le résultat et la donnée initiale dans une même figure avec des couleurs différentes

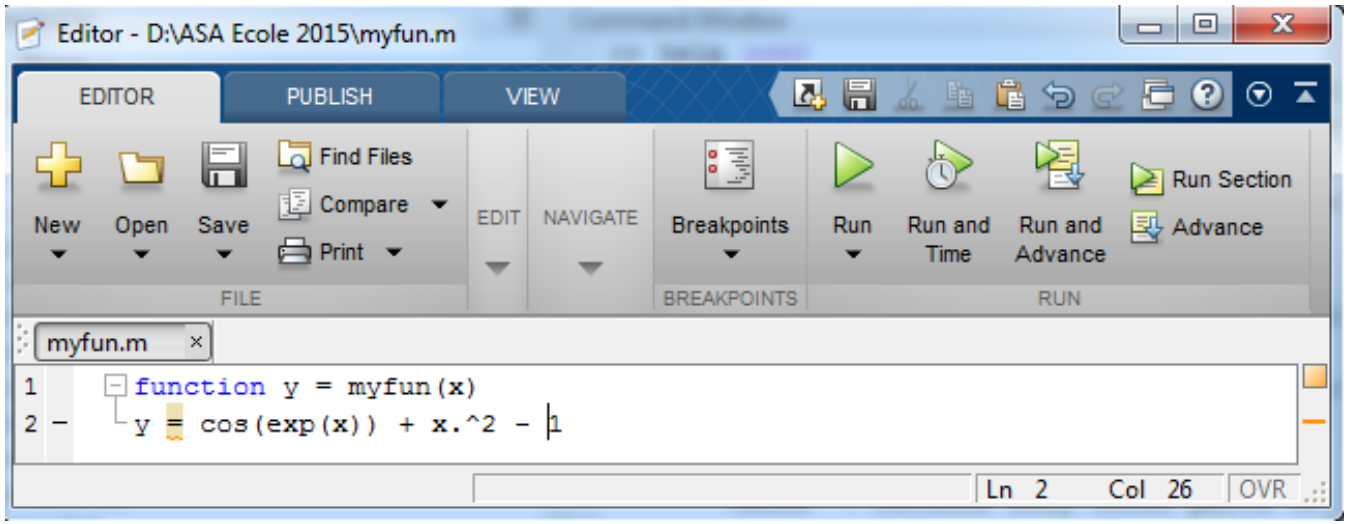
```
» hold on;
» plot(x,polyval(p,x),'r')
```

Plan

Equation non linéaire

- Beaucoup de problèmes nécessitent de résoudre $f(x) = 0$
- fzero calculer les zéros de n'importe quelle fonction
- fzero a besoin d'une fonction comme input
  - 1 indique la recherche du zéro au voisinage de 1

```
» x=fzero('myfun',1)
» x=fzero(@myfun,1)
```



Minimisant une fonction

- fminbnd : minimise une fonction sur un intervalle borné

```
» x=fminbnd('myfun',-1,2);
```

- myfun prend une entrée scalaire et renvoie une sortie scalaire
- myfun(x) sera le minimum de myfun pour $-1 \leq x \leq 2$

- fminsearch : sans contrainte d'intervalle

```
» x=fminsearch('myfun',.5)
```

- cherche un minimum local de myfun avec un point initiale x=0.5

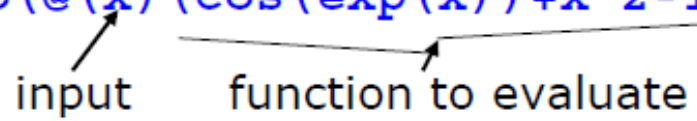
Fonctions anonymes

- On a pas besoin de créé un fichier pour définir une fonction

```
» x=fzero(@myfun,1)
```

- Au lieu de cela, vous pouvez faire une fonction anonyme

```
» x=fzero(@(x)(cos(exp(x))+x^2-1),   1   );
```




```
» x=fminbnd(@(x)  (cos(exp(x))+x^2-1),-1,2);
```

- Si vous êtes familier avec les méthodes d'optimisation, utiliser la boîte d'optimisation
  - » `linprog`
    - linear programming using interior point methods
  - » `quadprog`
    - quadratic programming solver
  - » `fmincon`
    - constrained nonlinear optimization

Exercice

- Déterminer le minimum de la fonction $f(x) = \cos(4x)\sin(10x)e^{-|x|}$ dans l'intervalle $[-\pi, \pi]$. Utiliser fminbnd.
- Tracer cette fonction et vérifier l'existence d'un minimum
- Introduire la fonction suivante :

```
» function y=myFun(x)
» y=cos(4*x).*sin(10*x).*exp(-abs(x));
```

- Trouvez le minimum dans command window :

```
» x0=fminbnd('myFun',-pi,pi);
```

- Tracer la fonction et vérifier le résultat

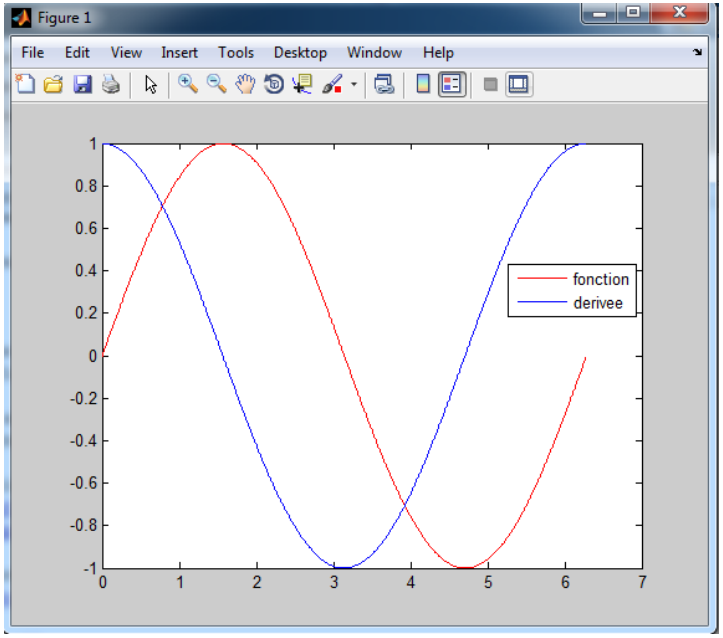
```
» figure; x=-pi:.01:pi; plot(x,myFun(x));
```

Plan

Dérivation numérique

- MATLAB peut calculer des dérivées numériquement

```
»x=0:0.01:2*pi;
»y=sin(x);
»dydx=diff(y)./diff(x);
```



- On peut aussi opérer sur les matrices

```
» mat=[1 3 5;4 8 6];
» dm=diff(mat,1,2)
```

- Gradient en 2D

```
» [dx,dy]=gradient(mat);
```

Intégrtion Numérique

- MATLAB contient des méthodes d'intégration
- Adaptive Simpson's quadrature (input est une fonction)

```
» q=quad('myFun',0,10);
```

  - q est l'intégrale de la fonction myfun de 0 à 10

```
» q2=quad(@(x) sin(x)*x,0,pi)
```

  - q2 est l'intégrale de sin(x)*x de 0 à pi

- Méthode des trapèzes (input est un vecteur)

```
» x=0:0.01:pi;
» z=trapz(x,sin(x));
```

  - z est l'intégrale de sin(x) de 0 à pi

```
» z2=trapz(x,sqrt(exp(x))./x)
```

  - z2 est l'intégrale de $\sqrt{e^x}/x$ de 0 à pi

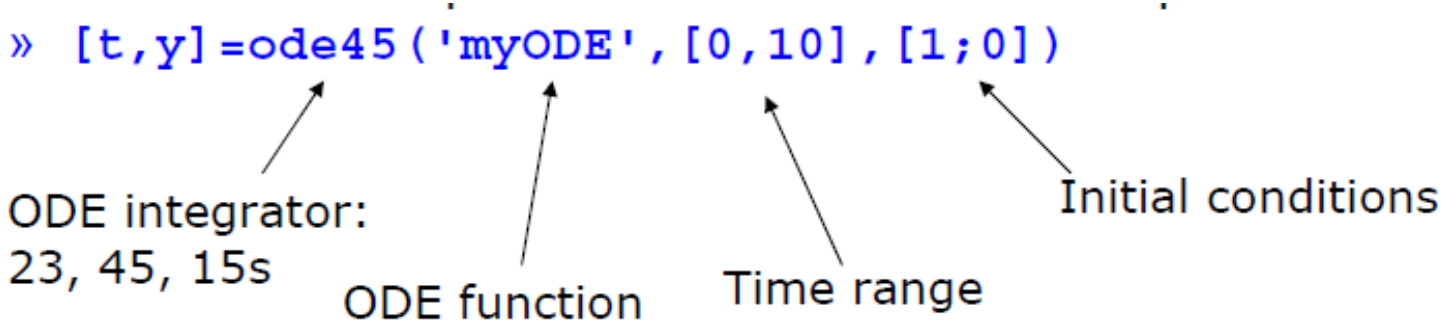
Plan

Solveurs ODE : MATLAB

- MATLAB intègre des différentes méthodes d'intégration
- ode23 : algorithme de Runge-Kutta du 2ème et 3ème ordres
- ode45 : algorithme de Runge-Kutta du 4ème et 5ème ordres (Haute précision et vitesse raisonnable. Le plus couramment utilisé.)
- ode113 : utilise une méthode de Adams-Bashforth-Moulton. C'est un solveur multi-pas
- Les solveurs pour problèmes "raides" sont au nombre de quatre : ode15s, ode23s, ode23t, ode23tb

Serveurs EDO : Syntaxe Standard

- La syntaxe d'appel du serveur est



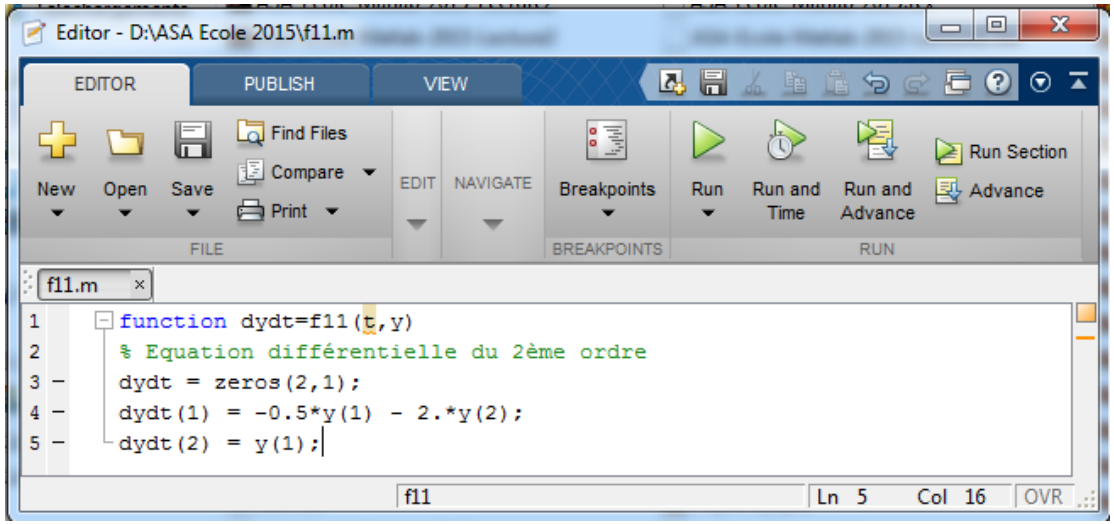
- Inputs :
  - Nom de la fonction qui définie l'EDO. Elle renvoie dy/dt
  - Intervalle du temps : un vecteur à deux éléments $[t_0, t_1]$
  - Conditions intiale : un vecteur colonne avec la condition initiale pour l'EDO
- Outputs :
  - t : un vecteur colonne contenant les noeuds de l'intervalle $[t_0, t_1]$ où a été calculée la solution
  - y : une matrice contenant les valeurs de la solution et de ses dérivées aux noeuds de l'intervalle $[t_0, t_1]$. La i^e ligne de y contient les valeurs de la dérivées $i - 1^e$ de la solution aux noeuds de l'intervalle $[t_0, t_1]$. La 1^e ligne contient la solution y de l'EDO

Fonction EDO

- La fonction EDO doit retourner la valeur de la dérivée à un temps donné
- Exemple

$$\begin{cases} \dfrac{dA}{dt} = -0.5A - 2B \\ \dfrac{dB}{dt} = A \end{cases}$$

- y = [A ; B]
- dydt = [dA/dt ; dB/dt]



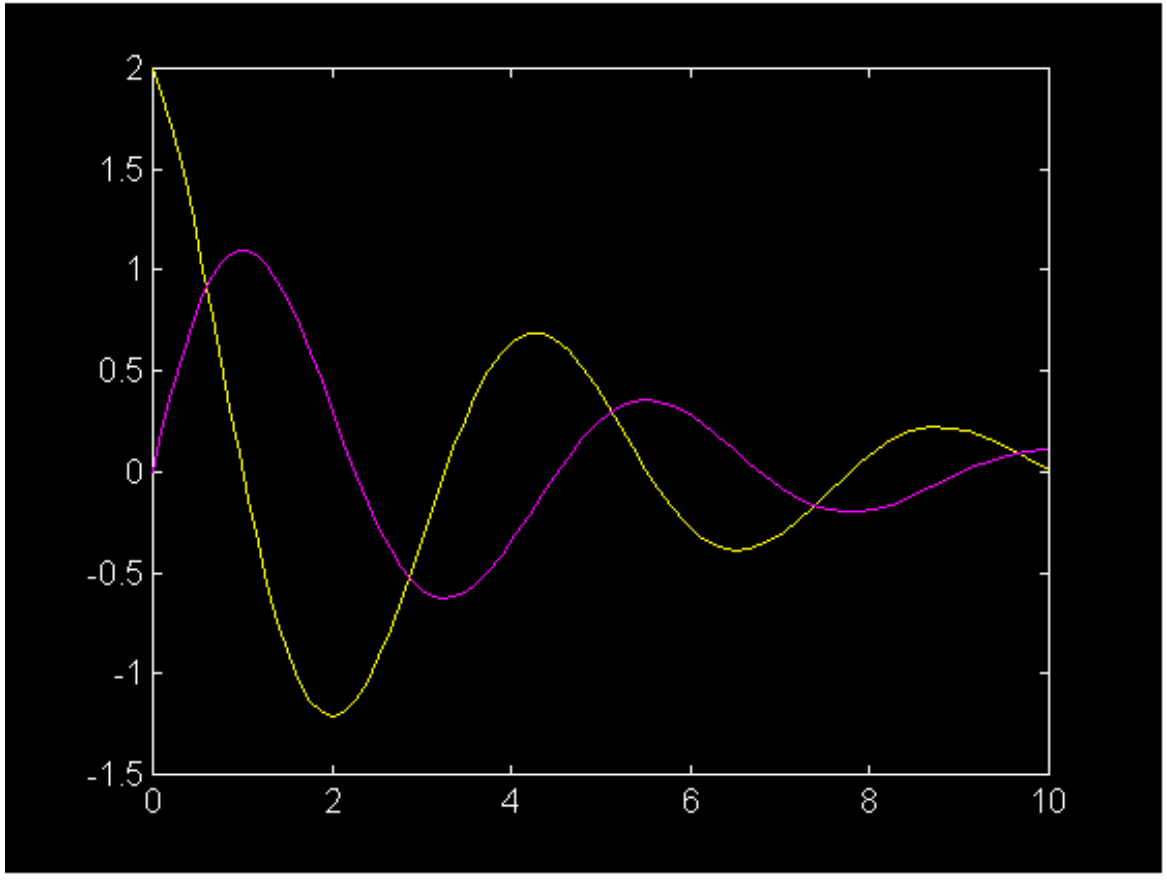
Affichage des résultatss

- Pour résoudre et tracer les équations différentielles

```
>> [ t , y ]=ode23 ( 'f11' , 0 , 10 , [ 0 2 ] )
» plot(t,y(:,1),'k','LineWidth',1.5);
» hold on;
» plot(t,y(:,2),'r','LineWidth',1.5);
» legend('A','B');
» xlabel('Time (s)');
» ylabel('Amount of chemical (g)');
» title('Chem reaction');
```

Résultats

- Le code donne le résultat suivant



Exercice : EDO

- Utiliser ode45 pour résoudre $dy/dt - -ty/10$ avec $t \in [0, 10]$ et $y(0) = 10$;
- Tracer le résultat

Exercice : EDO

- Utiliser ode45 pour résoudre $dy/dt - -ty/10$ avec $t \in [0, 10]$ et $y(0) = 10$;
- Tracer le résultat

- Introduire la fonction suivante
- Intégrer la fonction EDO et tracer le résultat

```
» [t,y]=ode45('odefun',[0 10],10);
```

- Vous pouvez également utiliser une fonction anonyme

```
» [t,y]=ode45(@(t,y) -t*y/10,[0 10],10);
```

- Tracer le résultat

```
» plot(t,y);xlabel('Time');ylabel('y(t)');
```

Exercice : EDO

- La fonction intégrée ressemble à ceci :