Option MAE 2007 – Master DSSC

Calculs et
Essais en
Vibration

Plan des TP

- Rappels de base d'UNIX
- Introduction à Matlab
- Simulation modèle 1 DDL sous Matlab
- Mise en œuvre d'outils de traitement du signal
- Simulation numérique de la réponse vibratoire
- Essais marteau & illustration des propriétés dynamiques des structures (EdF Clamart)

Rappels de base d'UNIX

- *Operating System* (OS, càd la *langue de l'ordinateur*) le plus répandu sur les stations de travail et ordinateurs de calcul

- A donné naissance à un nombre considérable de versions dérivées, avec des améliorations spécifiques,...

  ➔ par exemple LINUX, Mac-OS X, Windows (DOS, CYGWIN pour implémentation LINUX)

- C'est LE système pour le calcul intensif, existant sur tous les super-ordinateurs, quel que soit l'origine et le fabricant

- UNIX est sensible à la casse ! (majuscules / minuscules)

Compte UNIX

- Première action : se connecter sur l'ordinateur grâce à son **`login`** et à son **`password`**
- Utilisez le compte maevo ou DEA
- Mot de passe actuel : maevo05
- Connectez vous, ouvrez un terminal (xterm)
- Créez un répertoire pour chaque binôme

  mkdir NomRepertoire

- Vous accédez toujours au même compte, quelle que soit la machine sur laquelle vous travaillez à MSSMat
- Ce n'est bien sûr pas le même compte qu'au CTI

*** Lire un fichier, **more**

- Permet de consulter rapidement le contenu d'un fichier
- Pas de possibilité de modifications
- Taper **more nomdufichier** (p.ex. **more .login**)
- Si le fichier est plus long qu'une page d 'écran, taper **<RTN>** pour afficher une ligne de plus, **<ESPACE>** pour accéder à la page suivante, **b** (pour back) pour revenir une page en arrière
- Recherche d 'une chaîne de caractères donnée: **/chaine<RTN>**
- Quitter : **q**
- Autres **cat nomdufichier** (tout le fichier)
  sous MATLAB c'est type

Aide en ligne, **man**

- **man** (pour manuel) permet d'afficher l'aide en ligne associée à une commande UNIX donnée
- Rentrer par exemple **man more** ou **man passwd**
- Le comportement de la fenêtre d 'aide est le même que celui obtenu pour la commande **more**
- Chaque fois que vous voulez plus de détails sur une commande

  → **man commande**

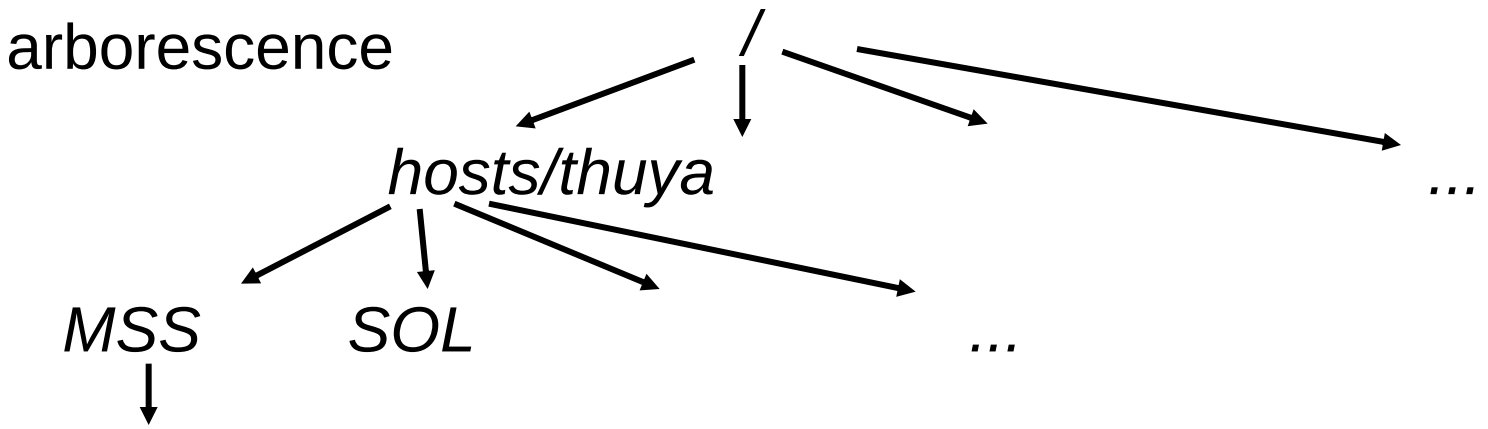
- Permet aussi de récupérer la liste des arguments et des options d'une commande

Liste de fichiers, `ls` (`dir` pour win/matlab)

- `ls` (pour liste) permet d'afficher l'ensemble des fichiers et des répertoires disponibles en local
- Taper `ls` dans le répertoire actuel
- Cette commande dispose de nombreuses options complémentaires (derrière le signe -), par exemple
  - ➔ `ls -l` : taille, propriété des fichiers et autorisations
  - ➔ `ls -1` : affichage sur une seule colonne
- Les options sont combinables, par exemple
  - ➔ `ls -ltr` : affichage par ordre chronologique
- Pour tous les détails sur les options existantes
  - ➔ `man ls`

Gestion de répertoire, **mkdir** (unix/win/matlab)

- **mkdir** (pour make directory) permet de créer un nouveau répertoire
- Taper `cd ~;mkdir VotreNom`
- La commande **rmdir** (remove directory) permet de supprimer un répertoire



/hosts/thuya/MSS/STR4/ENSE/maevo/VotreNom

Se déplacer, **cd**, **pwd** **(unix/win/matlab)**

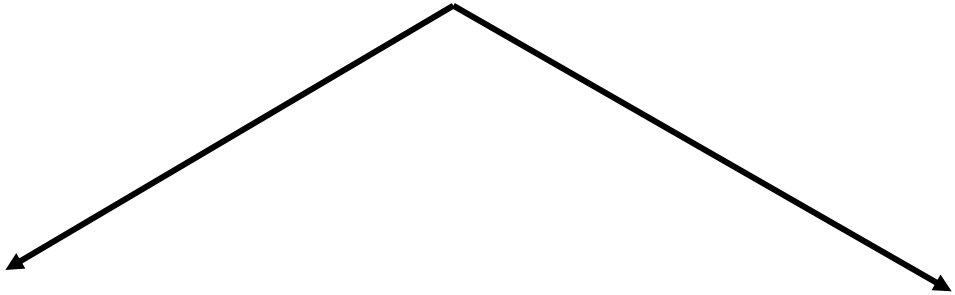
- **cd** (pour change directory) permet de se déplacer parmi les répertoires existants
- pwd (present working directory) permet de savoir où on se trouve dans l'arborescence
- Chaque utilisateur possède un répertoire par défaut
  - cd ~ (revenir au home, même chose que cd sans argument, pas sous DOS)
  - cd . (ne fait rien car . = pwd)
  - cd .. (remonte d'un niveau)
  - cd /tmp (déplacement absolu dans /tmp)

*** Autorisations, `chmod` (win: cacls)

- Vous ne pouvez bien sûr vous déplacer que là où vous possédez les autorisations nécessaires...
- Par `ls -l`, vous pouvez visualiser les autorisations locales en lecture (**`r`** pour read), écriture (**`w`** pour write) et exécution (**`x`** pour execute)
- Ces autorisations peuvent être différentes pour le propriétaire du fichier (également donné par `ls -l`), le groupe, et le reste du monde
  - ➔ p. ex. **`rwxr-x---`**
- Il est possible de modifier ces autorisations par la commande `chmod` au besoin (voir `man chmod`). A priori inutile pour vous...

Editer un fichier (1/2)

- Rendez vous dans votre répertoire **`MATLAB`** personnel
- Deux logiciels différents (au moins !) sont disponibles pour créer et modifier un fichier



- Editeur Matlab
- Simple, mais suffisant pour toutes vos applications CEV
- Dans MATLAB, menu open ou taper

```
open FileName.m <RTN>
```

- Logiciel générique : emacs
- Beaucoup plus puissant, et plus complexe
- Pour connaisseurs seulement
- Taper **`emacs &`** **`<RTN>`**

Editer un fichier (2/2)

- Remarque : le caractère & derrière une commande (emacs &) permet de récupérer la main immédiatement, même si la commande met du temps à s'exécuter. L'exécution continue en arrière-plan. Sans &, la fenêtre est bloquée jusqu'à la fin de l'exécution de la commande... (c'est aussi vrai sous win)
- Taper une ligne de texte quelconque
- Sauvegarder le fichier sous le nom **`fichier.txt`**, en allant dans le menu **`File/Save as`**
- Quitter l'éditeur de texte (Menu **`File/Quit`**)

Gestion de fichiers **`cp mv rm`**

- Commencer par **`ls -l`** pour vérifier l'existence du fichier
- Pour dupliquer un fichier, utiliser **`cp`** (copy sous win, copyfile sous matlab)

  ➔ **`cp fichier.txt copie.txt`**
- Pour renommer un fichier, utiliser **`mv`** (move sous win)

  ➔ **`mv copie.txt nouveau.txt`**
- Finir par **`ls -l`** : les propriétés (autorisations, propriétaire,...) sont conservées par **`cp`** et **`mv`**
- Pour supprimer un fichier, utiliser **`rm`** (del sous win, delete sous matlab)

  ➔ **`rm nouveau.txt`** (confirmation demandée)

Gestion de répertoires, **cp mv rm**

- Rendez-vous dans votre **home** par **cd**
- Pour dupliquer un répertoire et son contenu, utiliser **cp -r**

  ➔ **cp -r MATLAB COPIE**

  ➔ **ls COPIE** : les fichiers contenus dans **MATLAB** ont été recopiés tels quels dans **COPIE**
- Pour renommer un répertoire (contenu non modifié), **mv**

  ➔ **mv COPIE NOUVEAU**
- Pour supprimer un répertoire avec son contenu, utiliser **rm -r**

  ➔ **rm -r NOUVEAU** (confirmation demandée)

  ☹ **rm -rf NOUVEAU** (pas de confirmation demandée !)

*** Archivage **tar, gzip, zip**

- **tar** (pour Tape Archive) permet d'assembler une grande quantité de fichiers en une seule archive (= un seul fichier)

- zip permet d'assembler et de compresser des fichiers, très utile pour transférer des fichiers vers un PC

- extraction avec unzip

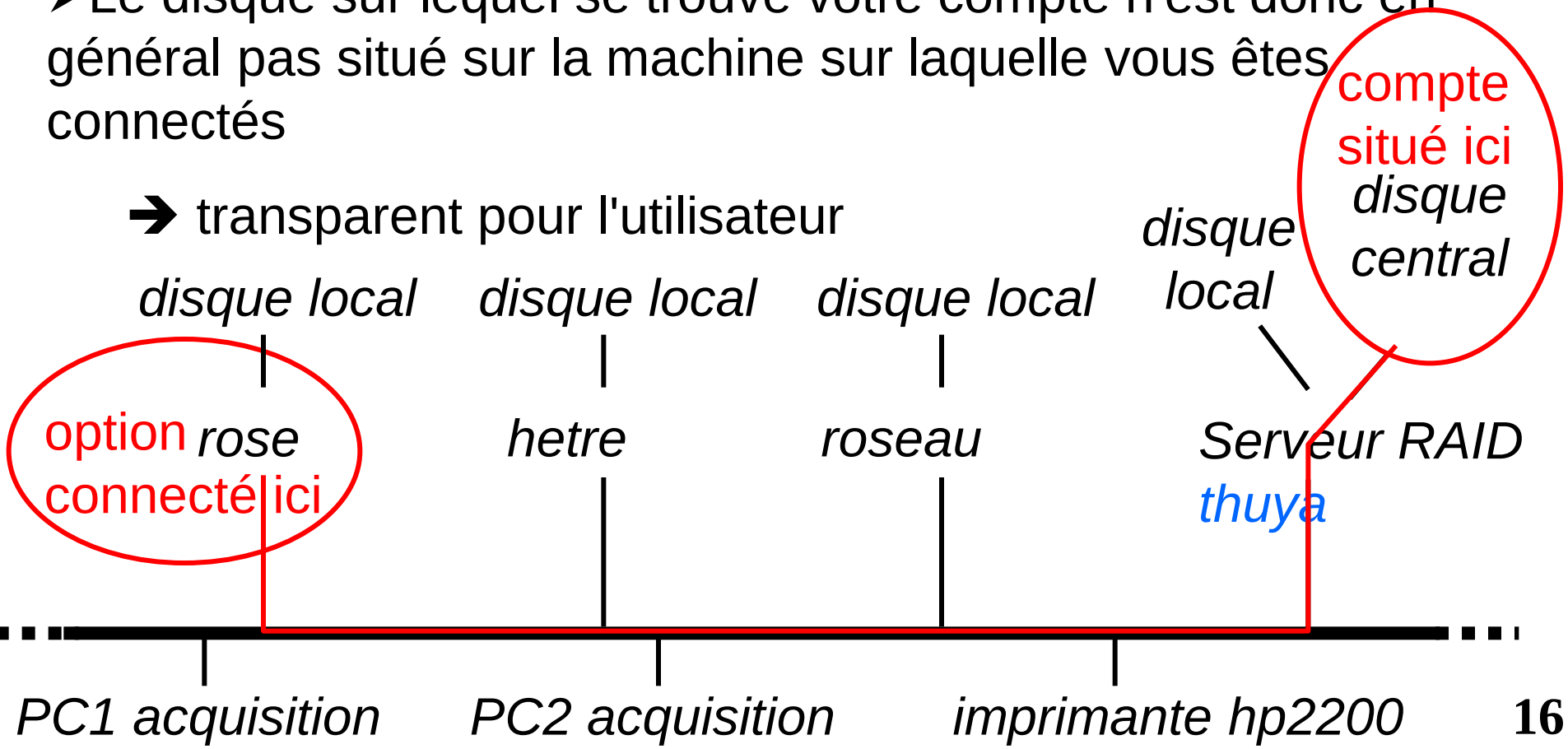
- gzip,gunzip : version de zip ne s'appliquant qu'a un seul fichier

*** Réseau Ethernet

- Toutes les machines sont connectées entre elles par réseau Ethernet

- Le disque sur lequel se trouve votre compte n'est donc en général pas situé sur la machine sur laquelle vous êtes connectés

➔ transparent pour l'utilisateur



Adresses Internet

- Toutes les machines sont connectées au réseau international Internet
- Chacun de ces ordinateurs dispose d'un identifiant unique, permettant un accès depuis n'importe quelle autre machine. Cet identifiant est constitué par une série de 4 nombres :
  - ➔ p.ex. **`138.195.68.73`**
  - ➔ France-Ecole Centrale-Labo EM2C-machine numero 73
- Ces numéros peuvent aussi être remplacés par des noms fixés par convention (serveurs de noms pour traduction) :
  - ➔ **`frescobaldi.em2c.ecp.fr`** , **`puma.cti.ecp.fr`**
  - ➔ id. système précédent mais à l'envers...

Transfert de fichier, **`ftp`** (1/3)

- **`ftp`** (pour File Transfer Protocol) permet de transférer des fichiers entre deux machines
- Connectez vous sur une machine du CTI par FTP :
  - ➔ p.ex. **`ftp puma.cti.ecp.fr`**
- Entrez votre **`login`** et votre **`password`** (du CTI !)
- Vous avez maintenant accès à tous les fichiers de votre compte CTI
  - ☹ **`put fichier.txt`** envoie ce fichier d'MSS vers le CTI
  - ☹ **`get fichierCTI.txt`** récupère ce fichier (s'il existe !) depuis le CTI vers MSS (attention : écrasement)

Transfert de fichier, **`ftp`** (2/3)

- Deux modes de transfert

  ➔ **`asc`** pour des fichiers standard ASCII (texte, listing,Postscript)

  ➔ **`bin`** pour des fichiers binaires (exécutables, fichiers comprimés, GIF, JPEG,...)

- Pour des transferts multiples de fichiers, rajouter **`m`** (multiple)

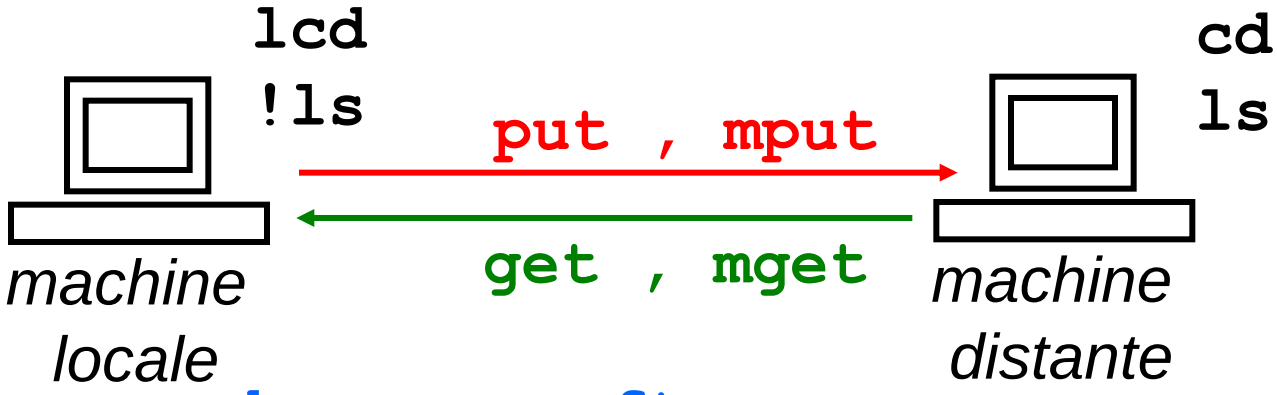
  ➔ **`mput *.txt`** envoie tous les fichiers d'MSS se terminant par **`.txt`** vers le CTI

  ➔ **`mget *fic*`** récupère tous les fichiers du CTI contenant **`fic`** vers MSS

  ➔ **`prompt no/yes`** : désactive/active la confirmation

Transfert de fichier, **ftp** (3/3)

- Pour se déplacer entre les répertoires, commandes précédents **cd** (machine distante), **lcd** (pour local cd, sur la machine locale)
- Pour visualiser les fichiers accessibles, utiliser **ls** (machine distante), resp. **!ls** (machine locale)
- Pour sortir du prompt **ftp>**, utiliser **quit**



Versions modernes : sftp

- scp toto.m user@distante:toto.m : copie sur le compte distant
- rsync

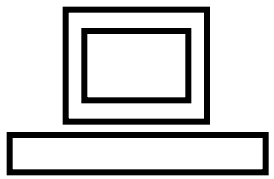
Travail a distance : telnet, ssh

Ouverture d'une `shell` sur une machine distante avec retour graphique Xwindows (X11)

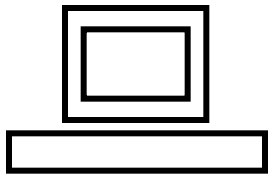
- **`ssh -X user@distante`** : version moderne
- La technique traditionnelle

```
xhost distante
telnet distante
```

```
setenv DISPLAY locale:0.0
```



machine locale



machine distante

Plan des TP

- Rappels de base d'UNIX
- Formation à Matlab
- Simulation 1 DDL sous Matlab
- Mise en œuvre d'outils de traitement du signal
- Simulation de la réponse vibratoire
- Essais marteau sur un couvre culasse

Introduction à Matlab (1/3)

- Matlab : MATrix LABoratory
- A l'origine, logiciel de calcul matriciel
- Depuis, étendu pour devenir le logiciel le plus vendu de calcul numérique pour l'ingénierie

  ➔ différent de Mathematica, Maple : logiciels de calcul formel
- Produit par **http://www.mathworks.com**

- Vous l'utiliserez pour beaucoup d'autres projets !

Introduction à Matlab (2/3)

- Un coeur central : le logiciel MATLAB à proprement parler

  ➔Un langage de programmation, analogue au C

  ➔Un gestionnaire de mémoire. Par défaut matrice de flottant double précision. Autres variables structures de données (struct), objets, matrices typées, …

  ➔Une distribution des meilleures librairies numériques (BLAS, LAPACK, FFTW, …), graphiques (OPEN-GL, menus, boutons, etc.), et entrées/sortie (binaires indépendants machine, …)

  ➔ Environnement de développement (debugger, profiler, lien dynamique avec C, Fortran, Java, COM (ActiveX), …)

Introduction à Matlab (3/3)

- Complété par une quantité de librairies annexes (*toolboxes*, avec financement séparé !)

  ➔ traitement temps réel, traitement d'images, finances, chimie, contrôle de systèmes, …

  ➔ simulation de systèmes dynamiques (Simulink), complété par des extensions appelés blockset (codage micro-controlleurs, …)

  ➔ Dynamique des Structures (www.sdtools.com) utilisé pour les TP

Aspects positifs

- **Portabilité** : MATLAB est une machine virtuelle identique sur toutes les plateformes (comme JAVA).
- **Facilités de développement** : pas de déclaration des variables, pré-interprétation automatique, existence d'un débugger interactif
- Qualité des **librairies** numériques et graphiques
- Existence de **Toolboxes** adaptées à de très nombreux problèmes
- Présent partout (quasi-monopole)
- De nombreux sources fournis

Aspects négatifs

- MATLAB est assez cher (pour un industriel licence individuelle PC 2500 Euros)

  Concurrent gratuit www.scilab.org .

- MATLAB est un langage de programmation
- Il est facile d'écrire du code sous-performant.(Pertes de temps en boucles et allocation mémoire)

```
» clear all
» tic; t=linspace(0,10,10000); for j1=1:10000
b(j1)=sin(t(j1));end;toc
elapsed_time = 4.3960
» tic; t=linspace(0,10,10000); b=sin(t); toc
elapsed_time = 0.0200
```

*** Historique

- Matlab 4 (1992) : matrices creuses, graphiques objets (*handle graphics*)
- Matlab 5 (1995) : typage des variables, programmation objet, extensions graphiques
- Matlab 6 (2000) : intégration JAVA, LAPACK au lieu de EISPACK
- Matlab 7 : généralisation du typage

Lancer Matlab

- Pour lancer Matlab, taper
  - ➔ **`matlabcti`** ou `m65` dans une fenêtre UNIX
- Apparition fugitive d'une fenêtre graphique
  - ➔ tout va bien concernant le graphisme
- Obtention du prompt **`>>`**
  - ➔ Matlab est prêt à recevoir les commandes

Informations

- Version de Matlab dont vous disposez

  ➔ **`version`**

- Toolboxes complémentaires disponibles en local

  ➔ **`ver`**

- Aide disponible : manuel papier. En ligne :

  ➔**`help`** ou **`help nomdecommande`** : aide en ligne directe sous le prompt Matlab

  ➔**`doc`** ou **`doc nomdecommande`** : documentation complète au format HTML

  ➔web(fullfile(matlabroot,'help','helpdesk.html'),'-browser')

Création/manipulation de données

- Fonctions implicites
  - ➔ `A = sqrt(3)` racine carrée
  - ➔ `A = 3^4` puissance
  - ➔ `A = exp(2*log(3))` exp./logarithme
  - ➔ `a = sin(pi/4)` trigon., pi eps i j préprogrammé
  - ➔ `B = abs(a-5)` valeur absolue ou norme d'un complexe
  - ➔ **+ - * / (voir help /)**
  - ➔ Pour Matlab, `A` différent de `a`

Création/manipulation de données

- Création de matrices
  - `A = [1 2 3; 4 5 6]`
  - `;` séparation de lignes (le saut de ligne marche aussi)
  - `,` séparation sur la même ligne
  - `. . .` saut de ligne texte sur même ligne de matrice
  - `[]` indique explicitement une matrice
  - `A2 = [A; [7 8], 9]`

Création/manipulation de données

- Création d'un vecteur **A** de dimension 4 par pas de 1
  - ➔ `A = 1:4`
  - ➔ résultat affiché directement à l'écran
  - ➔ incrément de 1 implicite
  - ➔ premier indice mis à 1 par défaut
  - ➔ pas de dimensionnement préalable nécessaire
- Liste de variables
  - ➔ `who` liste des variables existantes (`ans` pour answer)
  - ➔ `whos` idem `who` avec en plus la taille des matrices (cf. `size`)

Création/manipulation de données

- Vecteur à incrément explicite
  - ➔ `A = -1:0.1:1`
  - ➔ première valeur:incrément:dernière valeur
  - ➔ pas de virgule, des `.`
  - ➔ redimensionnement et/ou écrasement automatique

Création/manipulation de données

- Matrice initialement remplie de 0 ou de 1 au choix
  - ➔ `Z = zeros(8)` matrice carrée 8x8 de 0
  - ➔ `U = ones(size(Z))` matrice carrée de 1 de la même taille que **Z**
  - ➔ `U2 = U` boucle sur les indices inutiles (même néfaste !)
  - ➔ `V = U(:,1)` vecteur formé à partir de la première colonne
- Matrice unitaire
  - ➔ `I = eye(5)` matrice unité 5
  - ➔ help elmat

Création/manipulation de données

- Redimensionnement automatique

  ➔ `B = 0:0.5:3`

  ➔ `size(B)` : taille de la matrice `B = 7`

  ➔ `B(10)= B(1)-5`

  ➔ `size(B)` : taille de la matrice `B` modifiée

  ➔ matrice complétée par défaut par des 0 de manière automatique et redimensionnée

  ➔ attention aux fautes d'indice ! Contrôle par `size` ...

Opérations matricielles (1/3)

- Calcul de déterminant
  - → `d = det(A2)`
- Transposition (matrice complexe `a'` **`hermitien,`** `a.'` **`transpose)`**
  - → `B = A2'`
- Inversion
  - → `C = inv(A2)` déterminant nul !
  - → `A2(3,3) = 0`
  - → `d = det(A2)`
  - → `C = inv(A2)`

Opérations matricielles (2/3)

- Somme/produit de matrices
  - `C = C-1` opération répétée sur chaque élément
  - `D = A2 + C` opération matricielle
  - `P = A2*(C+4)` opération matricielle
  - `M = A2^2` opération matricielle

Opérations matricielles (3/3)

- Division et pseudo-division
  - ➔ `C = C/2` opération répétée sur chaque élément
  - ➔ `D = C/A2` solution de `D*A2 = C`
  - ➔ `D = C\A2` solution de `C*D = A2`

- Ménage général
  - ➔ `clear all` toutes variables réinitialisées

Opérations par éléments (1/2)

- Différence entre opération matricielle et opération portant sur les éléments individuels
- Opérateur `.` placé devant un autre opérateur
  - ➔ `A = [1/5 2 3; 4 5 6; sqrt(7) 8 9^2]`
  - ➔ `B = A^2` opération matricielle `B = A*A`
  - ➔ `C = A.^2` chaque élément de **C** est calculé comme étant le carré de l'élément correspondant de **A**
- Opérateur `.*` `./` `.^` etc

Opérations par éléments (2/2)

- Une opération scalaire appliquée sur une matrice prend automatiquement effet sur chacun de ses éléments séparément

  - `A = [1/5 2 3; 4 5 6; sqrt(7) 8 9^2]`
  - `B = sqrt(A)` opération par élément
  - `C = sin(A)` idem
  - `D = sin(B)./B` deux opérations élémentaires...

Indices

- Vous pouvez facilement sélectionner un partie de matrice

  - `A=linspace(0,10,11); A=[A;A];`
  - `A(1,2)`
  - `A(:,3)`
  - `i1=1:3; A(:,i1)`
  - `A(:,4:end)`
  - `A(:)`

Minimum/maximum

- Différence entre vecteurs et matrices
  - ➔ valeur unique ou valeur de la ligne du maximum
- Vecteur
  - ➔ `V = [1/5 2 3 sqrt(8)]`
  - ➔ `m1 = max(V)` valeur seule
- Matrice
  - ➔ `A = [1/5 2 3; 4 5 6; sqrt(7) 8 9^2]`
  - ➔ `m2 = max(A)` vecteur des max. par colonnes
  - ➔ `m3 = min(min(A))` valeur seule car **`min`**(vecteur)

Sauvegarde de données

- Fichiers binaires, relecture seulement possible par Matlab (terminaison en **`.mat`**)
  - ➔ **`save allvar.mat`** sauvegarde toutes les variables de Matlab actuellement en service
  - ➔ **`save A.mat A`** sauvegarde seulement **`A`** dans le fichier nommé **`A.mat`** (créé pour l'occasion)
  - ➔ **`clear all`** : ménage complet
- Recharger les valeurs dans Matlab
  - ➔ **`load A.mat`** pour **`A`** seulement
  - ➔ **`load allvar.mat`** pour tout recharger comme avant

Faire du ménage

➔ `close all` , fermeture de toutes les figures précédentes, si graphisme

➔ `clear all` , effacement de toutes les variables existantes

➔ `clear NomDeLaVariable global` , effacement d'une variable spécifique et des variables globales

*** Temps d'exécution

- **`profile`** pour identifier les parties les plus lentes d'un programme
- Utilisation d'une horloge interne de Matlab
  - ➔ **`tic`** démarrage de l'horloge
  - ➔ **`toc`** arrêt de l'horloge et affichage du temps écoulé depuis le dernier **`tic`**
  - ➔ possibilité de sauvegarder et d' additionner les temps résultants
  - ➔ **`cputime`** : analogue mais permet de s'affranchir des autres utilisateurs de l'ordinateur, donc plus précis

; Affichage ou non des résultats

➢Pour les opérations longues, il est nettement préférable de ne pas afficher automatiquement les résultats

➔ `A = ones(12)` création d'une "grosse" matrice

➔ `B = A^2`

➔ `C = A^2;`

➔ le `;` derrière une commande permet de supprimer l'affichage automatique des résultats

➔ même résultat `diff = max(max(abs(B-C)))`

➔ beaucoup plus rapide, car l'affichage est lent ! (vérifier au besoin)

Création de scripts Matlab

- L'énorme intérêt de Matlab ne devient évident qu'avec la création de scripts et fonctions
- Il s'agit de programmes en langage de programmation intégré dans Matlab (quasi-C)
- Lancer votre éditeur de texte préféré. Depuis MATLAB
  - ➔ `open nomdufichier.m`
  - ➔ `!emacs & % appel unix`

Script Matlab : commentaires

- Elément essentiel, bien que n'ayant aucune action
  - ➔ le commentaire, commençant par `%`
- Permet d'identifier le but du script, ses entrées-sorties, sa date de création, son créateur
- Et surtout de décrire en détail le déroulement des opérations....
- Un script non-commenté est pratiquement sans valeur. Pour un programmeur professionnel, entre 30 et 50% des lignes sont des lignes de commentaires.

Script Matlab : conditions (1/2)

- Les ordres conditionnels sont bien sûr aussi très importants

```
if (condition logique)
        bloc d'instructions
else                    [optionnel]
     bloc d'instructions alternatives
end
```

- La condition logique peut combiner différents éléments en utilisant & (et), | (ou), == (égalité), ~= (différence), par exemple

```
if ((A > B) & ((B~=0) | (A <0)))
```

Script Matlab : conditions (2/2)

- D'autres ordres conditionnels existent aussi
  - **while** **(condition logique)**
  - **bloc d'instructions**
  - **end**
- Permet la répétition d'instructions jusqu'au respect d'une condition (interruption préalable possible par **break**)
- **switch** permet de faire des branchements multiples (avec **case**) en fonction de la valeur d'une expression initiale)

Script Matlab : boucles

- Les boucles constituent bien sûr un élément essentiel des scripts

```
for j1 = 1:10
            bloc d'instructions
end
```

- Variation d'indice :

➔ ind= 1:2:20; %début:pas:fin (idem vecteurs implicites)

```
for j1 = ind
            bloc d'instructions
end
```

Script Matlab : structure

- Un script Matlab doit toujours posséder un nom se terminant par `.m` (pour Matlab)
  - ➔ par exemple **`graphisme.m`**
  - ➔ au début, bloc de commentaires (%) : but, date de création, entrées/sorties, structure,...
  - ➔coeur du programme principal, structuré et commenté
- Premier exemple, à taper dans votre fenêtre d'éditeur...

- which graphisme permet de savoir ou est le fichier.

Premier script Matlab

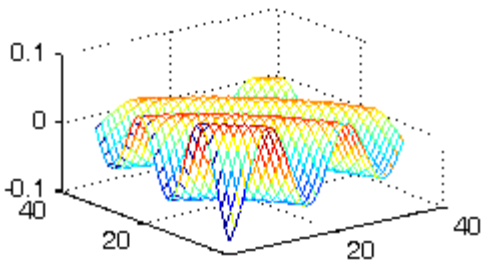
```
%But : representation graphique de la fonction
%(sin r)/r en 2D
%
% V2 : 11 Septembre 2002, E. Balmes
%
close all
clear all
% Limites du domaine selon directions X et Y
Xmin = -8;
Xmax = 8;
Ymin = -8;
Ymax = 8;
```

Premier script Matlab (suite)

```
% Pas de discretisation selon X et Y
Xstep = 0.5;
Ystep = 0.5;
% Generation des vecteurs de discretisation
X=Xmin:Xstep:Xmax;
Y=Ymin:Ystep:Ymax;
% Calcul du carre de la distance a l'origine D
for j = 1:size(Y,2)
     D(:,j) = X(:).^2 + Y(j)^2;
end
```

Premier script Matlab (fin)

```
% Calcul de la fonction a tracer (distance)
FS = sqrt(D)+eps; %eps predefinie="epsilon"
F = sin(FS)./FS;
% Ouverture de fenetre et trace graphique
figure(1);
% Cette derniere ligne subira de nombreuses
%modifications par la suite pour tester
%d'autres commandes
mesh(F)
%
%Fin de programme
```



Script Matlab : exécution

- Sauver **`graphisme.m`** (par exemple)
- Exécuter en tappant **`graphisme`** sur la ligne de commande
- Ordre de recherche des fichiers .m

  dans . (local) puis path
- Edition du path ... (voir **`path, rmpath, addpath`**)
- Les types de fichiers reconnus par Matlab

  .m : fichiers source en langage Matlab

  .p : fichiers objet directement executable par Matlab

  .mex* : librairie dynamique écrite en C, C++ ou Fortran

  les classes java.

path Matlab : répertoires visibles

- Copier votre script par

  ➔ `copyfile graphisme.m ../graph2.m` (au niveau supérieur, c'est à dire dans votre *home directory*) ou

  ➔ `! cp graphisme.m ../graph2.m` (appel Unix)

- Essayer d'exécuter sous Matlab en tapant `graph2`

  ➔ ce script est introuvable...

- Rajouter votre répertoire de base au chemin d'accès en tapant sous Matlab

  ➔ `cd ..;addpath(pwd)`

- Maintenant, vous pouvez exécuter `graph2` sous Matlab...

Mauvais script Matlab (1/2)

➢Conseil : changer de nom lors de la sauvegarde pour éviter d'écraser le bon script…

```
for j1=1:33
for j2=1:33
D(j1,j2) = ((j1-1)*0.5+8)^2+((j2-1)*0.5+8)^2;
end
end
F = sin(sqrt(D)+1.e-5)./(sqrt(D)+1.e-5);
mesh(F)
```

Mauvais script Matlab (2/2)

- Résultat équivalent, mais :
  - ➔ pas facilement modifiable ! (min, max, step,...)
  - ➔ difficilement compréhensible par un autre
  - ➔ difficilement compréhensible par le rédacteur lui-même dans deux mois...
  - ➔ en pratique non réutilisable pour un programme plus gros
  - ➔ moins efficace du point de vue de la vectorisation
  - ➔ très difficile à déboguer

*** Divers...

- `!` Appel au système d'exploitation

`eval` : permet de transformer une chaîne de caractères en commande exécutée

➔ `eval['!gzip ','nomfic',num2str(i)]`

`return` : pour sortir d'un script avant la fin (debugging). Très utile pour tester pas à pas un script !

Fonctions

- Un fichier toto.m débutant par

  function [a,b,c]=toto(in1,in2,in3)

  Sorties a,b,c. Entrees in1,in2,in3
- Il peut y avoir des sous-fonctions qui ne sont visibles que par la fonction elle même

  function [a,b,c]=toto2(in1,in2,in3)
- les variables sont locales aux fonctions (ajoutez function toto au debut de votre script)

*** Effet de la vectorisation (1/2)

➢Pour voir l'effet de la "vectorisation", on va supprimer la représentation graphique dans le script. Remplacer l'ancien **`figure(1)`** et **`mesh(F)`** par

```
tic
for j1=1,size(F,2)
      for j2=1,size(F,1)
            F2(j1,j2) = F(j1,j2)+2;
      end
end
toc
```

*** Effet de la vectorisation (2/2)

➢Pour comparer, rajouter juste derrière dans le script un ordre parfaitement équivalent :

➔ `tic`

➔ `F3 = F+2;`

➔ `toc`

➔ les temps d'exécution varient. Répéter plusieurs fois !

➔ dans le deuxième cas, un traitement vectorisé des calculs permet d'accélérer considérablement (facteur ≈10) l'obtention des résultats. Et pourtant la matrice reste ici petite...

➔ on s'efforcera toujours de coder les scripts Matlab de manière vectorielle, surtout si les calculs sont longs !

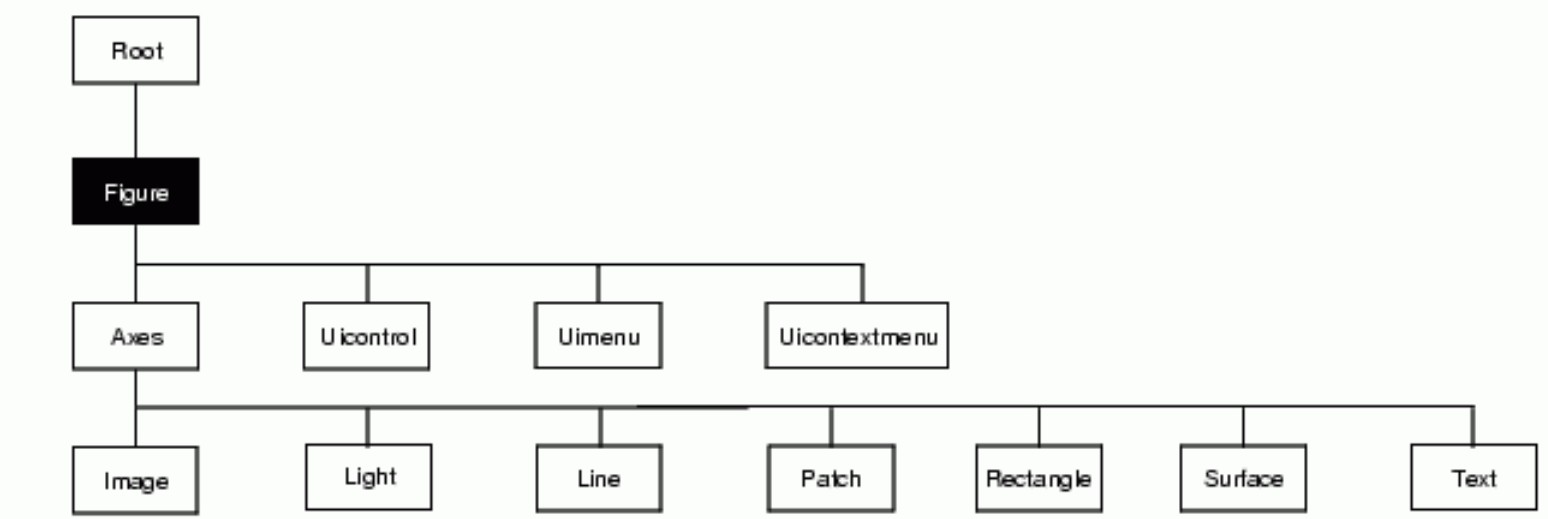
figure : fenêtres graphiques

- Ouverture et fermeture de figures

  ➔ **`figure(3)`** ouvre une nouvelle figure appellée **`Figure No.3`**. Possibilité de donner des arguments supplémentaires en passant par **`gcf`** (voir **`help figure`**).

  ➔ **`close(3)`** ferme la figure

  ➔ **`close all`** ferme toutes les figures ouvertes



2D (help graph2d)

- Représentation graphique d'une fonction monodimensionnelle

  ➔ utiliser par exemple une ligne de la fonction 2D **`F`**

  ➔ **`plot(F(1,:))`** représentation de la première ligne de **`F`**

  ➔ par défaut, représentation par une ligne continue par morceaux joignant les points

  ➔ possibilité de choisir la couleur de ligne (cf. **`contour`**)

  ➔ possibilité de représenter par des symboles (p.ex. **`+`**, **`x`**,**`o`**,**`.`**)

  ➔ **`plot(F(1,:),'o')`**

  ➔ combinaison avec choix des couleurs

  ➔ **`plot (F(1,:),'rx')`**

Superposition de graphes (1/2)

- A chaque appel d'une fonction graphique, Matlab réutilise la même figure, en effaçant le graphe précédent, à moins de faire appel à la commande **`figure`** qui, par défaut, ouvre une nouvelle figure et la rend active

- Dans le script

  ➔ `plot(F(1,:),'o')`

  ➔ `plot(F(3,:),'+')`

  ➔ le deuxième graphe efface et remplace le premier (si vous arrivez à le voir…)

Superposition de graphes (2/2)

➢Pour obtenir la superposition de plusieurs dessins dans la même figure, utiliser la commande **`hold`** suivie de **`on`** ou **`off`**

➔ **`plot(F(1,:),'o')`**

➔ **`hold on`**

➔ **`plot(F(3,:),'+')`**

➔ **`hold off`**

➔ le deuxième et le premier graphe sont maintenant sur la même figure. Tout nouveau graphe effacera et remplacera ce graphe, puisque **`hold off`** a été précisé...

Sous-graphes

➢Il est aussi possible d'obtenir plusieurs graphes les uns à côté des autres dans la même figure graphique grâce à la commande **`subplot`**

➔ `subplot(2,1,1)`

➔ `plot(F(1,:),'o')`

➔ `subplot(2,1,2)`

➔ `plot(F(3,:),'bx')`

➔ chacun des graphes individuels est accessible et modifiable de manière standard...

Tracé de champs de vecteurs

➢Pour représenter à l'écran des champs de vecteurs en 2D, Matlab utilise deux tableaux différents, qui contiennent respectivement les composantes des vecteurs. Remplacer **`mesh(F)`** par

```
F2 = ones(size(F)) % utilise arbitrairement
%une valeur fixe a 1 pour la composante 2

quiver(F,F2)
```

➔ deux arguments optionnels : au début pour les positions réelles des points de maillage, à la fin pour la taille des vecteurs

➔ **`quiver(Y,X,F,F2,3)`**

*** Couleurs

`colorbar` permet de représenter à côté du graphe la palette de couleurs correspondante

`colormap` permet de changer la palette de couleurs :
`colormap(jet), colormap(gray), colormap(hsv),...`

➔ un argument optionnel supplémentaire permet de changer le nombre de niveaux de couleurs

➔ **`colormap(jet(8))`**

➔ **`colormap(jet(256))`**

Labels, limites, …

➢**`xlabel, ylabel, zlabel, title`** permettent d'introduire des titres sur les axes

➔ p.ex. **`xlabel('x coordinate (m)')`**

`axis` permet au besoin de changer les bornes supérieure et inférieure des différents axes

axis equal : unités égales propriété DataAspectRatio

➢**`caxis`** permet de changer les limites de la palette de couleurs

*** Impression d'une figure

- On imprime la fenêtre active (voir gcf ou faire figure(2) pour activer ou créer la figure 2)
- Il est possible de changer la fenêtre active par **`figure`**

  `print -deps figure.eps` crée un fichier Postscript encapsulé Noir et Blanc appellé **`figure.eps`**

  `print -depsc figcouleur.eps` crée un fichier Postscript couleur appellé **`figcouleur.eps`**

  `!lp figure.eps` % appel a l'UNIX

  ☹ Rappel : l'imprimante est limitée au Noir et Blanc...

  Pour du bitmap **`print -dpng figcouleur.png`**

3D (help graph3d)

- Représentation graphique d'une variable

  `mesh(F)` représentation de la fonction bidimensionnelle **`F`** dans la troisième direction, en chaque point du maillage

  `surf(F)` analogue à **`mesh`**, mais en créant une surface continue

  `contour(F)` représentation des isocontours de la fonction bidimensionnelle **`F`** dans le plan

  `contour(F,20)` deuxième argument optionnel = nombre d'isocontours

  `contour(F,5,'k')` troisième argument optionnel = couleur (fixe) des isocontours

Graphiques 3D

- Représentation graphique d'une variable

  couleur codée sur une lettre : par exemple **k**=black, **w**=white, **b**=blue, **r**=red, **g**=green, **y**=yellow,... (cf. **help**)

*** Graphiques 3D

- Représentation graphique d'une variable

  ➔ **`pcolor(F)`** représentation par des zones de couleur continues de la fonction bidimensionnelle **`F`** dans le plan

  ➔ possibilité de modifier les détails de la représentation graphique par une deuxième ligne de commande

  ➔ **`shading flat`** chaque bloc de maillage est associé à une couleur unique

  ➔ **`shading interp`** interpolation de la couleur par dégradé entre les 4 coins du bloc de maillage

  ➔ plus joli, mais ne change rien à la quantité d'informations réellement disponible !

Graphiques 3D axes x,y

`mesh(Y,X,F)` [noter les modifs sur les axes]

`mesh(-2*Y,X,F)`

➔noter l'orientation de la fenêtre

possibilité d'échanger les directions X et Y par simple transposition de la fonction : **`F'`**

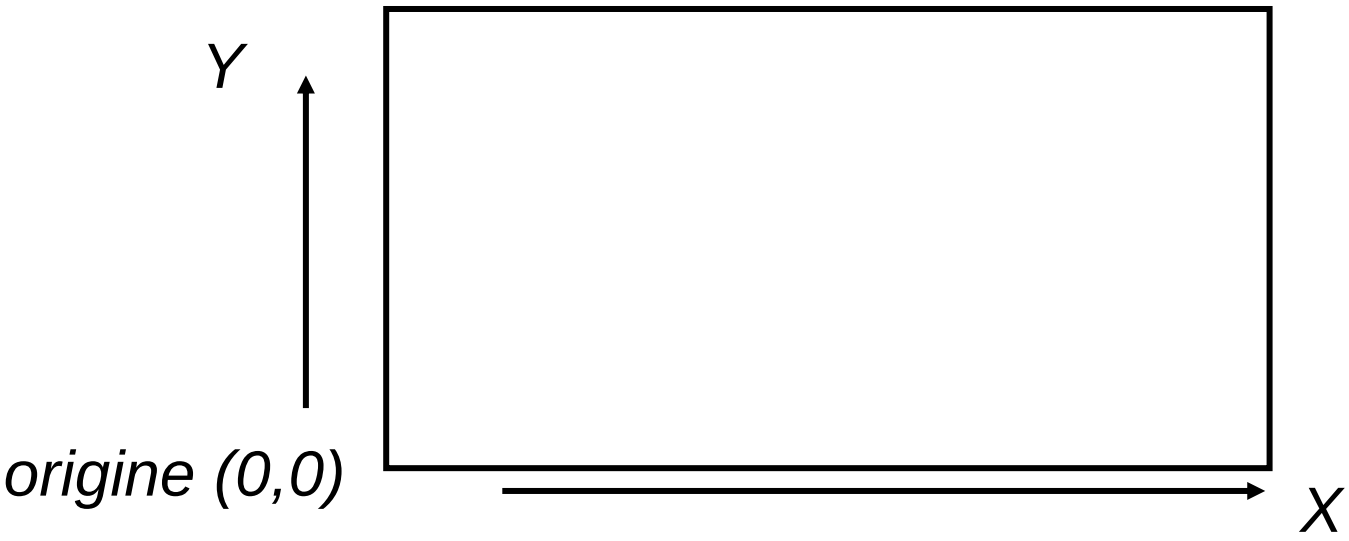
Graphiques 3D, view

- Orientation usuelle pour la fenêtre graphique

  `pcolor(X,Y,F')`

  permet de se replacer dans une vision standard pour les lignes et les colonnes de la matrice visualisée

- Changement de vue view(0,0), view(3), view(0,90)

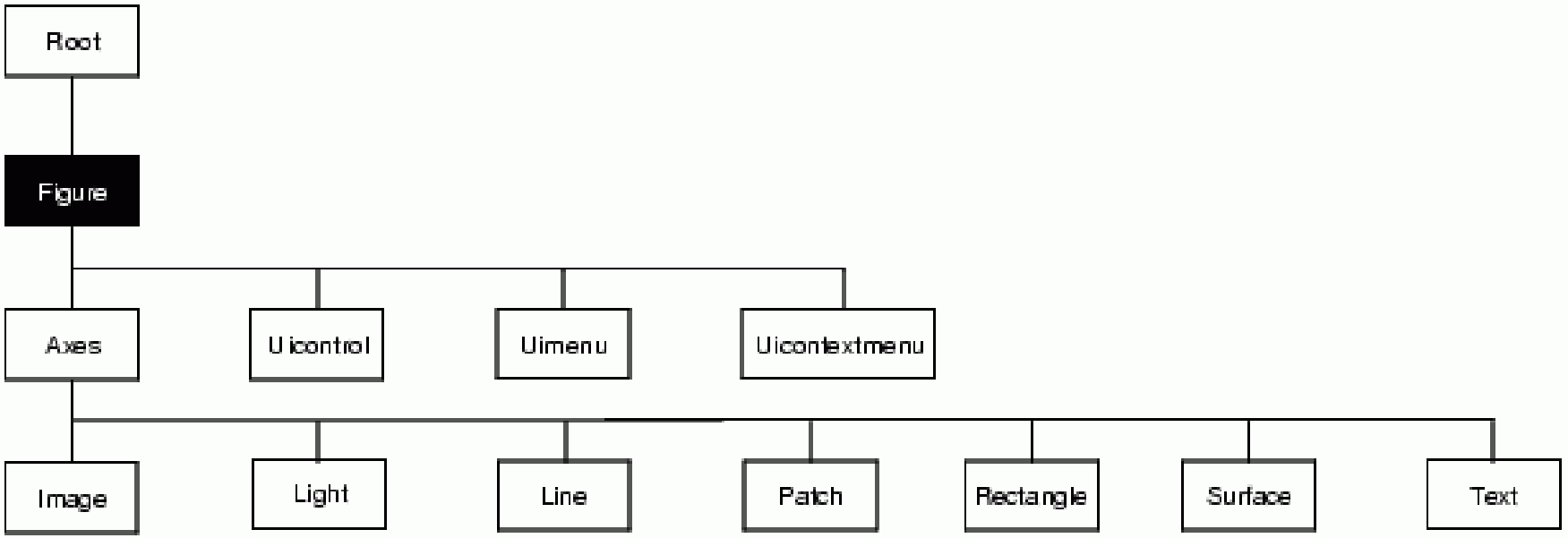


*** set(…,'prop','value'), get()

`get(gca)` %obtenir les propriétés de l'axe courrant

h= get(gca,'title');get(h)

set(h,'fontsize',30) % modifier la taille d'un titre



*** Tracé immédiat ou différé (1/2)

➢Pour accélérer l'exécution, Matlab ne trace pas à l'écran les graphes quand l'intervalle de temps entre deux tracés est trop bref. Remplacer **`mesh(F)`** par

➔ **`for i=1:size(F,1)`**

➔ **`plot(F(i,:),'r-')`**

➔ **`end`**

➔ les figures intermédiaires ne sont pas tracées par défaut...

*** Tracé immédiat ou différé (2/2)

➢Pour changer ce comportement, utiliser l'ordre additionnel **drawnow**

➔ `for i1=1,size(F,1)`

➔ `plot(F(i1,:),'r-')`

➔ `drawnow`

➔ `end`

➔ défilement de toutes les figures intermédiaires, comme pour un film

*** Réalisation de films à l'écran

➢Pour réaliser un film à l'écran, il faut d'abord stocker les différentes images dans une matrice unique (défilement de **F**)

```
nimage = size(F,1)
for i=1,nimage
        F2(1:nimage-i+1,:) = F(i:nimage,:);
        if (i>1)
         F2(nimage-i+2:nimage,:) = F(1:i-1,:);
        end
        surf(F2)
        Mfilm(:,i) = getframe; %acquisition
end % maintenant, on peut jouer le film
```

*** Réalisation de films à l'écran

- On peut maintenant faire jouer le film par la commande **movie**

```
fps = 4; % vitesse (frames per second)

movie(Mfilm,4,fps) % deuxieme argument =
%nombre de repetitions du defilement
```

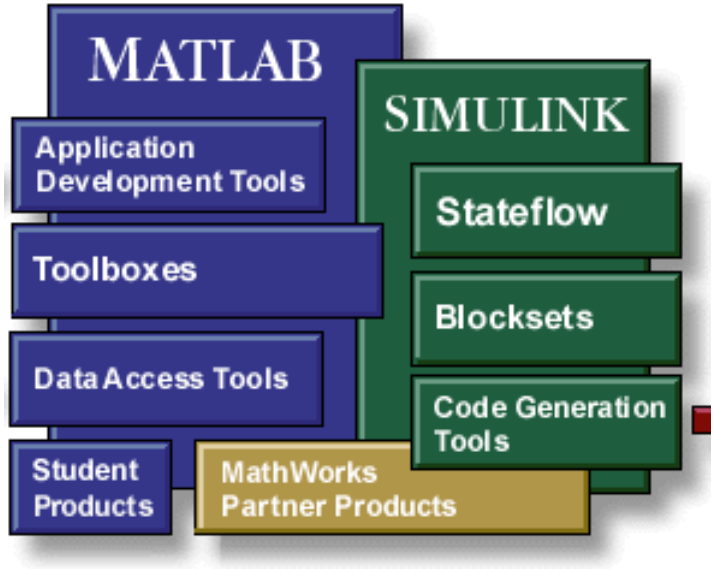
Pour en savoir plus

doc, doc NomDeLaFonction

help, help NomDeLaFonction

demo

*** Matlab et Simulink



Simulink

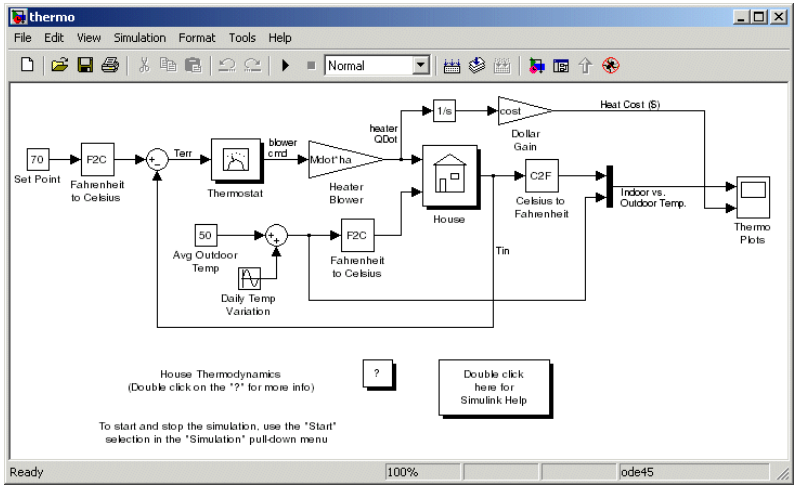
Programmation graphique

Outils d'intégration numérique

Simulation événementielle

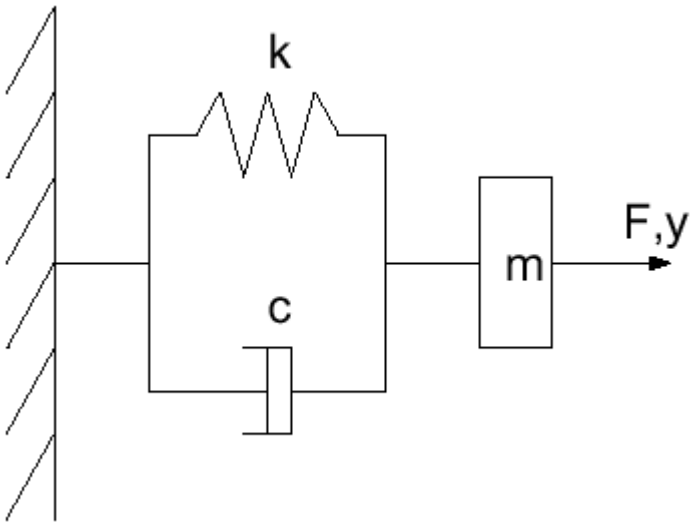
Liens avec les microcontrôleurs

Plate-forme de référence pour la conception d automatismes



Simulation 1 DDL

- Modèle d'état associé à M = 1 K=1e2
- Choissez C pour avoir un taux d'amortissement de l'ordre de 1 \% (indice : il faut calculer les valeurs propres de la matrice d'état A et regarder le rapport partie réelle sur amplitude des pôles)



1 DDL réponse libre

- Utilisez les fonctions d'intégration de Matlab (help ode45) pour calculer la réponse à une condition initiale non nulle et tracez cette réponse.
- Calculez la transformée de Fourier de la réponse (avec fft) retrouvez la fréquence de résonance dans cette FFT (calculez les fréquences à partir du vecteur temps).
- Illustrez l'influence du taux d'amortissement en superposant les fft pour trois valeurs de C différentes.

1 DDL réponse forcée

- Appliquez une force sinusoïdale dont vous choisirez l'amplitude et la fréquence de façon à illustrer l'apparition de la raie correspondante dans la fft de la réponse.
- Illustrez la décroissance du transitoire lié aux conditions initiales. (Indication : comparez les fft d'un signal complet et d'un signal tronqué (valeurs t>0).