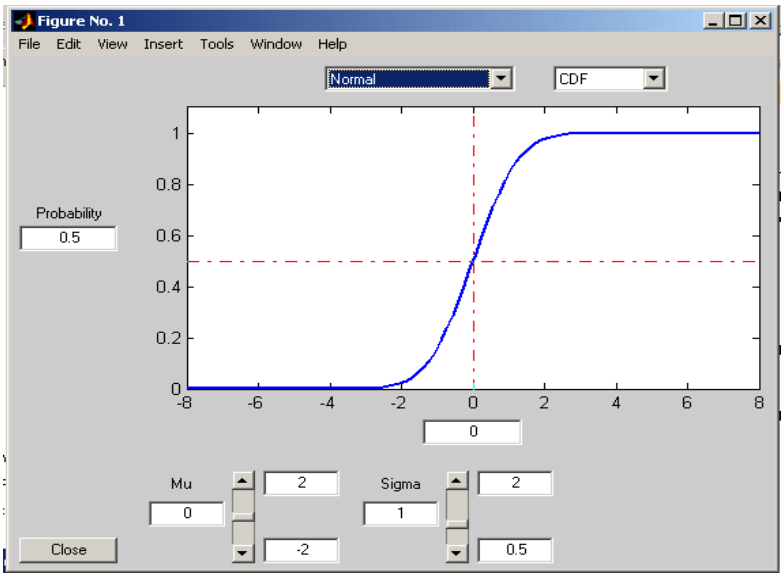
Création d'une interface graphique

Une interface graphique permet de contrôler une application interactivement avec la souris , plutôt que par lancement des commandes au clavier.

Une interface graphique comprend des menus, des boutons, des "ascenseurs", des cases à cocher, des listes de choix, des zones de texte. Exemple d'interface :



Elle permet de "cliquer" directement sur des images, des graphiques ou des objets pour *modifier la valeur d'une variable*, *déclencher des fonctions* ou simplement faire apparaître des informations lors d'un survol à la souris.

Matlab permet d'écrire assez simplement une interface graphique pour faire une application interactive utilisable par des utilisateurs non formés à Matlab.

Les notions principales d'une interface graphique sont :

- les divers objets graphiques, auxquels sont attribués des noms symboliques; ces "poignées de main" ou "***handles***" permettent de les repérer dans l'interface; pour envisager par exemple une modification dynamique (grisé d'un bouton provisoirement non utilisable, changement du texte d'un bouton, modification d'une liste de choix...)

- les ***propriétés*** des objets (couleur, disposition, taille, variable associée)

- les fonctions exécutées par les clic souris sur les éléments ou "***callbacks***" (décrites en ligne de commande Matlab).

Les versions actuelles de Matlab permettent de construire ces interfaces directement avec la souris grâce au GUIDE (Graphical User Interface Development Environment). Cet outil est capable de construire des applications de très haut niveau. Cependant, son approche est très délicate pour un utilisateur peu expérimenté. L'approche classique par description textuelle de l'interface, moins performante mais compréhensible, permet de se familiariser avec les principales notions.

Ce polycopié permet de faire une approche progressive du GUI; les exemples sont opérationnels et sont à tester pour se familiariser.

I - Elements de base de l'interface graphique

Pour créer une interface, il faut disposer d'une fenêtre de base dans laquelle seront insérés les éléments graphiques (objets).

A noter que tout dessin graphique ou affichage d'image (résultat de plot, mesh, imshow) peut servir de fenêtre de base.

- *Création d'une nouvelle fenêtre pour application:*

```
fig1 = figure
```

Le paramètre fig1 est le ***handle*** de la fenêtre, c'est à dire le numéro de repère de la fenêtre attribué par Matlab à sa création. Il est possible d'appliquer des fonctions sur cette fenêtre (redimensionnement, ajout de menus, boutons, ...) en précisant dans les fonctions le ***handle*** auquel elle s'applique. La fenêtre active à un instant donné a pour handle implicite gcf .

De façon générale, tout objet graphique se voit attribué un ***handle***; ce handle sert de référence à cet objet dans l'application.

- *Propriétés d'une fenêtre graphique (ou d'un objet)*

```
get(fig1)
```

Les principales propriétés sont : le titre, la position et la dimension dans l'écran, la couleur de fond, la présence et le type de menus, le redimensionnement...

Toute propriété particulière est obtenu par :

```
valeur_propriété = get( fig1, 'nom_propriété' )
```

Toute propriété (modifiable!) peut être modifiée en définissant une nouvelle valeur pour la propriété considérée (valeur numérique, chaîne, liste de valeur, tableau...)

```
set(fig1, 'nom_propriété' , valeur_propriété )
```

Ex : `set( fig1 , 'Name' , 'Demo GUI' , 'NumberTitle' , 'off' );`

La fenêtre de base est l'écran qui a pour handle "0". Par `get (0 , 'ScreenSize' )`, on obtient la taille de l'écran physique de l'écran. Ces valeurs permettent de fixer la taille d'une fenêtre en rapport avec la dimension physique de l'écran et d'éviter de créer une application qui "déborde" de l'écran!

La taille et la position de la fenêtre (ou d'un objet) se fixent par modification de sa propriété ou contrôle "position", comprenant les coordonnées (X_{or},Y_{or}) du coin inférieur gauche et ses dimensions (X_{fen},Y_{fen}):

```
set( fig1 , 'position' , [ 10 , 10 , 300 , 200 ])
```

L'ensemble des propriétés modifiables d'un objet est donné par `set(handle_objet)` . La liste s'affiche avec les valeurs possibles pour les différentes propriétés; les valeurs par défaut sont signalées par des crochets { } . Exemple :

```
set( fig1 )
```

Tout objet graphique créé pourra être supprimé par :

```
delete (handle_objet)
```

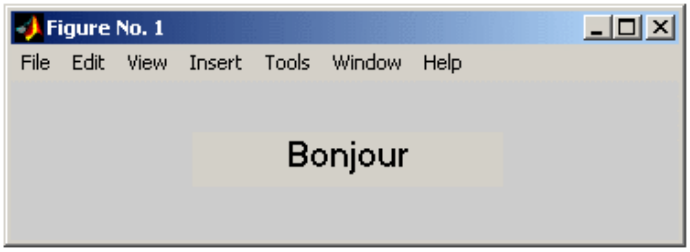
La suppression d'un objet entraîne la suppression des objets qui lui sont liés (objets fils).

▪ *Insertion d'un Objet dans la fenêtre*

L'insertion d'un objet dans une fenêtre se fait par la fonction "**uicontrol**" , dont le premier paramètre est le handle de la figure de référence. Le deuxième paramètre précise le "style" ou type d'objet à insérer.

Exemple le "texte fixe" est l'objet le plus simple; il permet de placer un texte dans la fenêtre.

```
text1 = uicontrol( fig1 , 'style' , 'text' , 'position' , [100,150,170,30] ,...
'string' , 'Bonjour' , 'fontsize' , 15 )
```

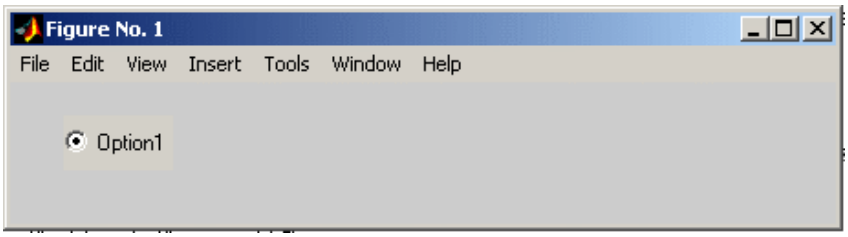


Toutes les propriétés de cet objet peuvent être modifiées par la commande "**set**" . Par exemple, le texte en lui-même ('string') étant une propriété, il peut être modifié:

```
set( text1 , 'string' , 'Au revoir' );
```

Autre exemple : insertion d'un bouton-radio :

```
radio1 = uicontrol( fig1 , 'style' , 'radio' , 'String' , 'Option1' , 'Position' , [30,30,60,30] )
```



A la différence du "texte" , on remarque que cet objet est "cliquable" à la souris, avec modification de son aspect (cet fonction est prise en charge sans aucune programmation par l'utilisateur).

II - Principe de l'interaction avec la souris

La presque totalité des objets de l'interface graphique (curseur, case à cocher...) peut interagir avec la souris.

▪ La fonctionnalité la plus courante est la modification de la valeur associée à l'objet (si elle existe): pour les objets destinés à faire une saisie (case à cocher, curseur, champ de saisie, choix de liste...), Matlab gère automatiquement la valeur associée. Cette valeur est récupérable par toute partie de l'application par la fonction "get" :

```
valeur = get (handle_objet , 'value');
```

Cette fonctionnalité permet de saisir la valeur d'une variable par l'interface graphique plutôt que par le clavier.

▪ La deuxième interaction courante est une action déclenchée par le "clic" souris sur l'objet (appuyé puis relâché): la fonction associée est décrite dans la propriété "***callback***" de l'objet. Cette fonction peut être une instruction de base Matlab ou une fonction définie par l'utilisateur (stockée en fichier .m)

```
set( radio1 , 'callback' , 'get( radio1 , ' 'value' ' ) ' );
```

Remarquer que la fonction est décrite en chaîne de caractères avec des " ' " en début et fin, ce qui oblige à doubler les " ' " pour ceux qui sont inclus dans la chaîne principale.

Cette description en chaîne permet de définir en callback une *liste d'instruction,* ce qui évite d'écrire une multitude de petites fonctions externes dédiées aux callbacks.

▪ Certains objets n'ont pas de callback (cas des figures) mais possèdent d'autres actions associées à la souris. Leur emploi est identique au callback classique (description de la fonction en liste d'instructions). Les principales sont :

```
WindowButtonUpFcn    WindowButtonDownFcn    WindowButtonMotionFcn
```

Exemple : récupération des coordonnées en pixels de la souris au clic

```
fig1 = figure ;
set( fig1 , 'WindowButtonDownFcn' , 'get( fig1 , ' 'CurrentPoint ' ' ) ' );
```

Si on désire obtenir des coordonnées dans l'espace de mesure des axes d'un graphique, plutôt qu'en pixels de la figure, il faut faire référence aux axes (gca) dans la fonction de clic :

```
plot( 20 , 20 , ' r+ ' )   % tracé d'une croix rouge au centre
set( gcf , 'WindowButtonDownFcn' , 'get( gca , ' 'CurrentPoint' ' ) ' )
```

▪ Certains objets possèdent une fonction callback et une fonction associée au clic souris (par exemple : ButtonDownFcn)

III - Principaux Objets Graphiques

▪ *Bouton poussoir*

Un bouton poussoir se crée par :

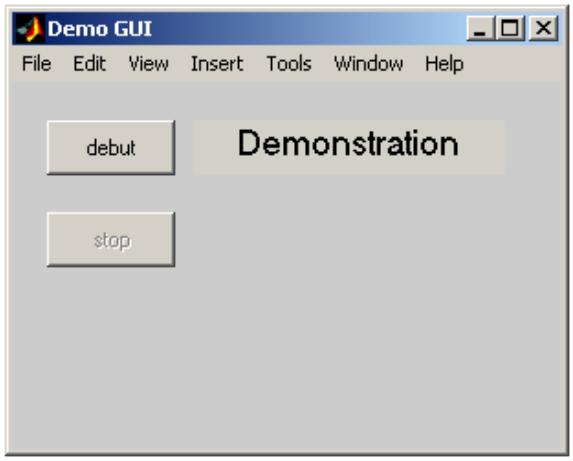
```
bp1= uicontrol ( fig1 , 'style' , 'push' , 'position' , [10 100 60 30 ] ,...
'string' , 'Début' , 'callback' , 'plot(T,X)' )
```

Lorsqu'on clique sur le bouton poussoir, il provoque l'exécution de la fonction indiquée dans le 'callback'. Cette fonction peut être une instruction de base Matlab ou une liste d'instruction, ce qui évite d'écrire une multitude de petites fonctions exécutées pat les callbacks.

Un bouton-poussoir s'inactive par la commande :

```
set(bp1 , 'enable' , 'off' )
```

Par cette commande, on peut rendre inactif certaines commandes, par exemple lorsqu'il manque des informations pour traiter un problème.



▪ *Menus*

Généralement, les menus de la fenêtre d'application ne sont pas les menus standard (voir vue ci-dessus mais des menus spécifiques. Un menu est un titre complété par une liste de sous-menu. Les actions (callbacks)sont généralement lancés à partir des sous-menus. L'ajout de menus spécifique se fait par :

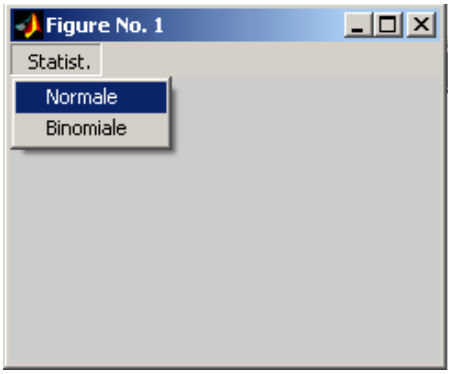
```
menu1 = uimenu( fig1 , 'label' , ' Statist.' );
```

Un sous-menu est un élément du menu principal, donc de l'entité père. Il est donc déclaré car menu du menu principal.

```
smenu1 = uimenu( menu1 , 'label' , 'Normale' , 'callback' , 'methode_normale' )
smenu2 = uimenu( menu1 , 'label' , 'Binomiale' , 'callback' , 'methode_binomiale' );
```

Pour enlever les menus standards de la fenêtre, il faut fixer la propriété "Menubar" à la valeur par défaut menubar :

```
set(fig1,'menubar',menubar);
```



▪ *Ascenseur ou slider*

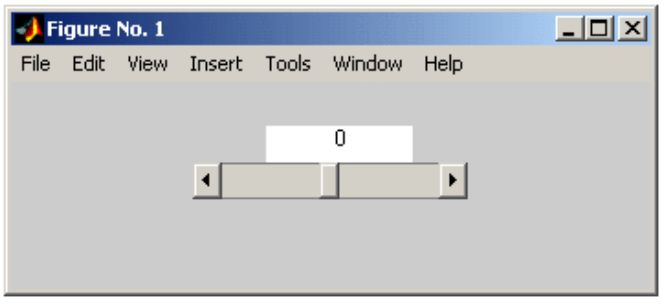
L'ascenseur a pour objectif de fixer la valeur d'un paramètre entre deux bornes fixées. La valeur prise par la variable est représentée par la position du curseur .

```
slid1=uicontrol(fig1,'style','slider','position', [100,50,150,20] , 'Min' , -1 , 'Max' , 1 , ...
 'callback' , 'val_variable = get(slid1 , "value" )' );
```

Les textes (variable affectée, valeurs..) ne sont pas définis par le slider. Il faut le compléter par des éléments textes convenablement placés et paramétrés; leur valeur est à modifier par le callback du slider.

Exemple d'ascenseur avec affichage de la valeur:

```
fig1=figure;
texte1=uicontrol(fig1,'Style','text','String',0,'Position', [140,70,80,20],'BackGroundcolor','w');
slid1=uicontrol(fig1,'style','slider','position', [100,50,150,20] , 'Min' , -1 , 'Max' , 1 , ...
 'callback' , 'set(texte1,"String", get(slid1 , "value" ))' );
```



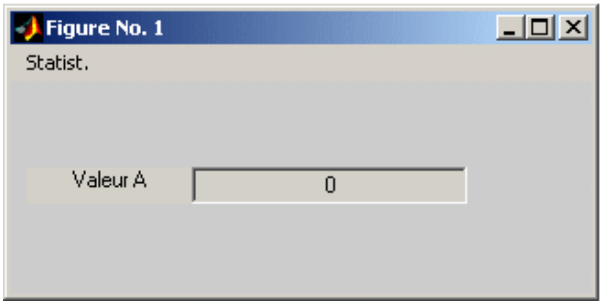
▪ *Texte Editable*

Permet à l'utilisateur de saisir une valeur. C'est une fonction importante.

```
Text1 = uicontrol ( fig1 , 'style' , ' edit' , 'position', [100,50,150,20] , 'Max' , 1 , 'string' , '0' );
```

Généralement, Il faut associer un texte fixe pour préciser le rôle de la fenêtre de saisie à l'utilisateur. Exemple : le texte est placé à gauche de la fenêtre de saisie

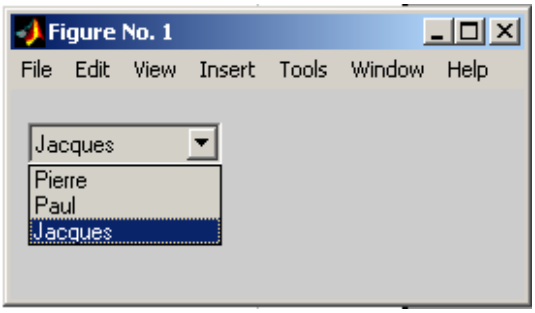
```
uicontrol ( fig1 , 'style' , ' texte' , 'position', [10,50,90,20] , 'string' , 'Valeur A' );
```



▪ *Liste de choix*

La liste de choix ou pop-up menu permet de sélectionner une valeur parmi une liste. Généralement, cette valeur est un texte . La valeur retournée lors du choix (paramètre 'Value') est le numéro de ligne du choix.

```
choix1 = uicontrol ( gcf , 'Style' , 'popup' , 'String' , 'Pierre|Paul|Jacques' , 'Position' , [10 10 100 80] );
```

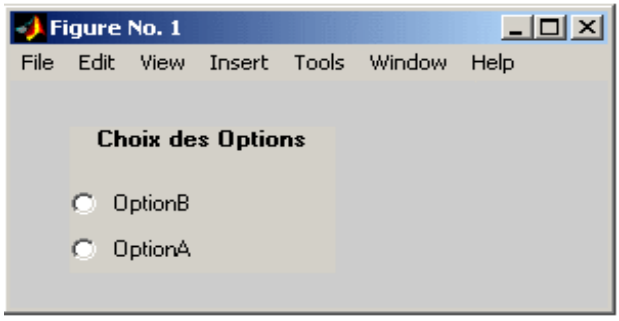


La liste des texte est modifiable après la création de la fenêtre de choix, en modifiant la propriété 'String' .

▪ *Bouton Radio*

Le bouton Radio permet de fixer un paramètre binaire (0 ou 1), représentant souvent un choix ou une option dans une application.

```
fig1 = figure ;
radio1 = uicontrol( fig1 , 'style' , 'Radio' , 'Position' , [ 30 20 130 25 ] , 'String' , ' OptionA ' );
radio2 = uicontrol( fig1 , 'style' , 'Radio' , 'Position' , [ 30 45 130 25 ] , 'String' , ' OptionB ' );
uicontrol( fig1 , 'style' , 'Text' , 'Position' , [ 30 70 130 30 ] , 'String' , ...
        ' Choix des Options ' , 'FontWeight', 'bold' );
```



Remarquer que les choix ne sont pas exclusifs (chaque choix peut être sélectionner). Pour obtenir l'exclusion mutuelle, il faut agir sur les valeurs de choix par les callbacks.

```
set( bradio1 , ' Value' , 1 );
set( radio1 , 'callback' , 'set( radio2 , ' ' Value' ' , 0 ) ' ) ;
set( radio2 , 'callback' , 'set( radio1 , ' ' Value' ' , 0 ) ' ) ;
```

- *Cadre*

Le cadre permet de dessiner un rectangle de présentation (par exemple regroupement de diverses entités graphiques dans un rectangle).

Le cadre se déclare par :

```
cadre1 = uicontrol ( fig1 , 'style' , 'frame' , 'position' , [ posX ,posY,tailleX,tailleY])
```

- *Graphiques*

Les graphiques se dessinent dans une partie de la fenêtre définie par la fonction 'subplot', dont les paramètres sont différents de l'emploi classique (division de la fenêtre en sous-fenêtres de taille égale)

subplot('Position' , [Xpos Ypos Xtaille Ytaille])

Les paramètres se définissent en % de la fenêtre (redimensionnement automatique des zones graphiques avec le redimensionnement de la fenêtre). Il est possible d'ouvrir plusieurs zones graphiques dans une même fenêtre. Les zones ne doivent pas se chevaucher, sous peine d'effacement de la zone située sous la nouvelle zone.

```
fig1 = figure ;
z1 = subplot ( 'Position' , [ .05 .1 .4 .4 ] ) ;
plot ( sin( 0: 10))
z2 = subplot ( 'Position' , [ .55 .1 .4 .4 ] ) ;
plot ( exp( 0 : 10))
```