Introduction à Scilab
Graphiques, fonctions Scilab, programmation, saisie de données

Jean-Philippe Chancelier & Michel De Lara
CERMICS, École des Ponts ParisTech

April 12, 2018

Contents

Ouvrir une fenêtre Scilab

Pour ces travaux pratiques d'introduction à Scilab, il vous faut lancer le logiciel Scilab et disposer ainsi d'une fenêtre permettant de saisir et d'exécuter des instructions.

Taper des instructions Scilab

Dans ces premiers travaux pratiques, vous trouverez une série de lignes de commandes Scilab précédées du signe `-->`. Pour commencer, il vous suffit de les recopier ou de les saisir par copier-coller (sans `-->`) pour les exécuter immédiatement dans la fenêtre Scilab.

Charger des instructions Scilab

Pour des successions de lignes de commandes, il est préférable de disposer d'un éditeur (Emacs), d'ouvrir un fichier (par exemple `nom_du_fichier.sce`), d'y écrire les lignes de commandes puis de les exécuter en tapant sous Scilab l'instruction :

```
exec("nom_du_fichier.sce")
```

On peut également utiliser la rubrique `File Operations` du menu `File`.

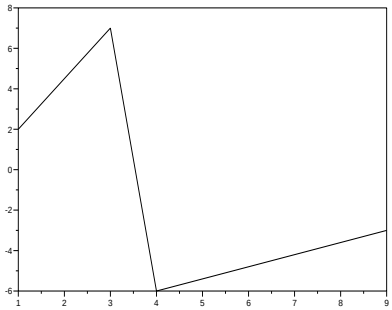
Commentaires

Toute ligne débutant par // est une ligne de commentaires.

1 Graphiques 2D

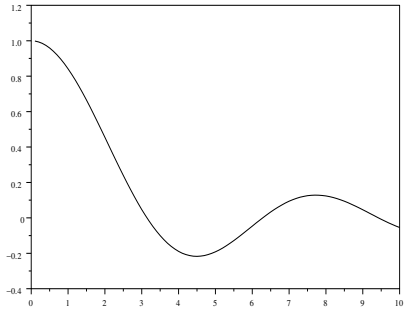
Pour effectuer des graphiques en deux dimensions, on utilise la commande `plot2d`.

```
x=[1,3,4,9];
y=[2,7,-6,-3];
plot2d(x,y)
// relie les points de coordonnées (x(1),y(1)), (x(2),y(2)), etc.
```



Tracé d'une unique courbe

```
x=0.1:0.1:10;
y=sin(x)./x;
xbasc(); // efface le contenu de la fenêtre graphique
plot2d(x,y)
```

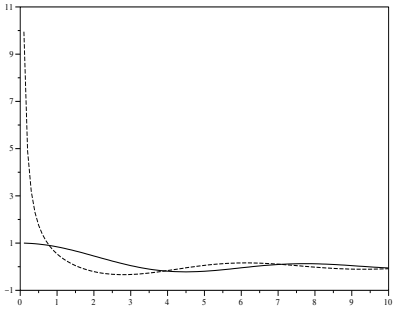


Tracé de deux courbes (par superposition)

```
x=0.1:0.1:10;
y=sin(x)./x;
z=cos(x)./x;
xset("window",1);
// ouvre une nouvelle fenêtre 1
// (la fenêtre par défaut est 0)
plot2d(x,y);plot2d(x,z) // les deux graphiques sont superposés
```

Tracé de deux courbes (simultanément)

```
x=x';y=y';z=z';
// x, y et z sont maintenant des vecteurs colonnes
// ce qui est obligatoire quand on veut dessiner plus de deux courbes.
xbasc();plot2d(x,[y z])
// idem, mais les deux graphiques sont tracés chacun avec sa couleur
```

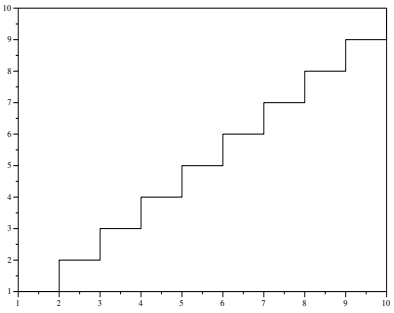


Tracé de plusieurs courbes (simultanément)

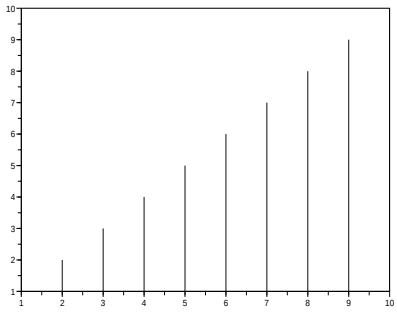
```
M=[(0:0.1:10)' (5:0.1:15)' (10:0.1:20)'];
N=[0.1*M(:,1).^2 log(M(:,2)) 0.5*M(:,3)];
xbasc();plot2d(M,N);
```

Créneaux `plot2d2`, barres `plot2d3`

```
x=1:10;
xbasc();plot2d2(x,x);
```

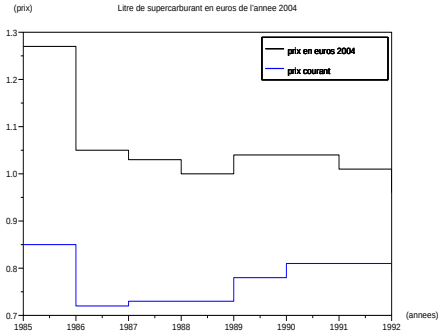


```
xbasc();plot2d3(x,x);
```



Titre `xtitle`**, légende** `legends`

```
 annees=[1985 1986 1987 1988 1989 1990 1991 1992];
 annee_ref=2004 ;
// les prix seront tous exprimés en euros de cette année
 prix_annee_ref=[1.27 1.05 1.03 1.00 1.04 1.04 1.01 0.96];
 prix_courant=[0.85 0.72 0.73 0.73 0.78 0.81 0.81 0.79];
 xbasc();plot2d2(annees',[prix_annee_ref ; prix_courant]');
 xtitle('Litre de supercarburant ...
en euros de l''annee ' +string(annee_ref),...
'(annees)','(prix)')
// Noter le '' pour signifier l'apostrophe dans l''annee.
// Dans une instruction trop longue pour tenir dans une ligne,
// mettre ... avant de passer à la ligne
 legends(['prix en euros '+string(annee_ref);'prix courant'],[1,2],'ur')
```



Exporter un graphique

Dans le menu de la fenêtre graphique, sélectionner `File`, puis `Export` pour exporter le graphique (au format Postscript par exemple, ou au format Postscript-Latex qui permet une insertion facile dans un fichier LaTeX).

2 Graphiques 2D : options

La fonction Scilab `plot2d` a de nombreux arguments optionnels permettant de spécifier des attributs d'un graphique : couleur, style et épaisseur des traits, échelle, dimensions du cadre, etc. Pour découvrir ces arguments optionnels, taper `help plot2d`.

Pour découvrir les attributs associés aux objets graphiques, utiliser le menu `Edit` de la fenêtre graphique. À travers ce menu, on peut également modifier en temps réel des attributs.

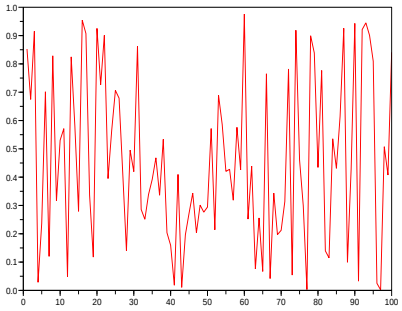
À titre indicatif, noter que les manipulations des attributs d'un graphique peuvent également être effectuées au moyen de commandes Scilab (taper `help graphics_entities`).

Nous allons illustrer des manières de changer les attributs d'un graphique 2D.

Option `style` de plot2d

L'option `style` permet notamment de changer la couleur.

```
x=(1:100)';
y=rand(100,1);
xbasc();plot2d(x,y)
// Par défaut, les points sont reliés par des droites de couleur noire
xbasc();plot2d(x,y,style=5)
// Avec l'option style=5, on obtient du rouge.
// Le code par défaut des couleurs se découvre par la commande getcolor()
```



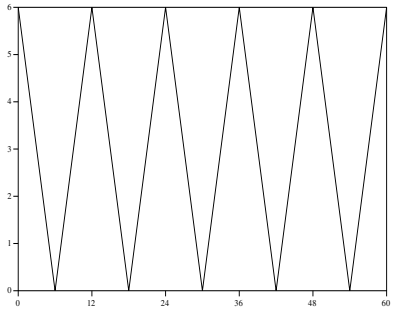
Ce changement de couleur peut également être obtenu en utilisant le menu `Edit` de la fenêtre graphique.

Question 1 *Essayez, à l'aide du menu* `Edit`*, de modifier la couleur et de rajouter des losanges aux points de la courbe. Pour cela, on cherchera Polyline dans le menu déroulant des entités graphiques.*

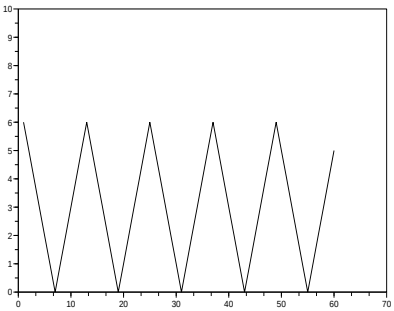
Option `rect` de plot2d

L'option `rect` permet de fixer les échelles.

```
y=[6:-1:0,1:5] ; x =[y,y,y,y,y];
xbasc();plot2d(1:60,x);
// Un graphique simple dans la fenêtre courante.
```




```
xset('window',2);
// Ouvre une deuxième fenêtre graphique.
xbasc();plot2d(1:60,x,rect=[0,0,70,10]);
// Les échelles sont fixées par rect =[xmin,ymin,xmax,ymax]
```



Question 2 *Effectuer la même manipulation à l'aide du menu* `Edit`.

3 Saisie de tableaux de chiffres

Lecture de données : le pouvoir d'achat du franc

Le tableau de chiffres suivant représente l'indicateur de pouvoir d'achat du franc de 1985 à 2001, exprimé en euros 2005 (source Insee). Il permet de traduire en euros 2005 des valeurs exprimées en francs du passé. Sur une ligne, on trouve

1. l'année ;
2. la valeur équivalente en euros, en pouvoir d'achat de 2005, d'1 FF de l'année.

Ainsi, 1 000 F de 1985 équivalent, en pouvoir d'achat, à 231,36 euros de 2005.

```
2001 0.16492
```

```
2000 0.16766
1999 0.17050
1998 0.17135
1997 0.17254
1996 0.17466
1995 0.17811
1994 0.18119
1993 0.18421
1992 0.18804
1991 0.19249
1990 0.19866
1989 0.20536
1988 0.21277
1987 0.21849
1986 0.22537
1985 0.23136
```

Le tableau de chiffres suivant représente l'indicateur de pouvoir d'achat de l'euro de 2002 à 2005, exprimé en euros 2005 (source Insee). Il permet de traduire en euros 2005 des valeurs exprimées en euros du passé.

```
2005    1.000
2004  1.018
2003  1.040
2002  1.061
```

Par copier-coller, transférer les deux séries de lignes précédentes dans deux fichiers `francs.txt` et `euros.txt`. On prendra garde à ce que le fichier finisse par un retour à la ligne, sinon la dernière ligne ne sera pas lue par `fscanfMat`. On va à présent lire les contenus de ces fichiers et stocker les résultats de la lecture dans deux matrices `Mf` et `Me` (à 2 colonnes et `n` lignes). Pour cela utiliser `help fscanfMat` :

```
Mf= fscanfMat('francs.txt');
size(Mf)
Mf(1,:)
Me= fscanfMat('euros.txt');
```

Question 3 *À partir des matrices* `Mf` *et* `Me`*, construire une matrice N dont la première colonne soit les années de 2005 à 1985, et la deuxième l'indicateur de pouvoir d'achat de l'euro exprimé en euros 2005. On rappele que 1 EUR= 6,55957 FF.*

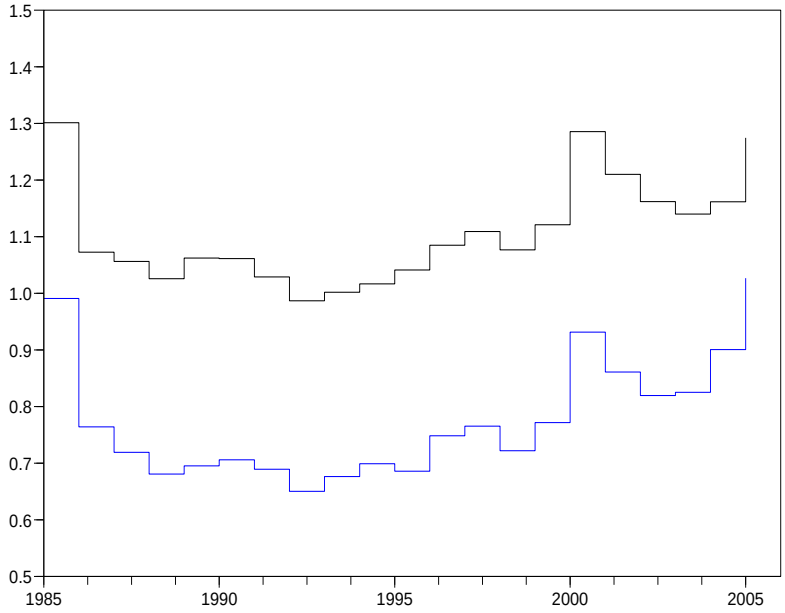
En utilisant `M=N($:(-1):1,:)`*, transformer cette matrice en une dont la première colonne soit les années de 1985 à 2005.*

Lecture de données : le prix des carburants

```
1985 85.737913076923 65.298018096154
1986 72.559868307692 51.697806788462
1987 73.706167788462 50.176247980769
1988 73.489055150943 48.769303018868
1989 78.860116961538 51.618650134615
1990 81.440023403846 54.186829923077
1991 81.495726230769 54.59726975
1992 79.991758057692 52.726837153846
1993 82.922916660377 55.966047339623
1994 85.538556557692 58.813071173077
1995 89.112313596154 58.692871
1996 94.692211192308 65.334312673077
1997 98.006277230769 67.640015865385
1998 95.804121788462 64.239142269231
1999 100.24807667711 68.998516558797
2000 116.88560606538 84.682718019723
2001 111.85613846808 79.601834232065
2002 109.5216625 77.242352692308
2003 109.60326923077 79.347115384615
2004 114.0958490566 88.470943396226
2005 127.46173076923 102.68038461538
```

Question 4 *Le tableau précédent représente, en euros courants, les prix des carburants (source direction générale de l'énergie et des matières premières, ministère de l'Industrie) en hectolitres. On touve sur une ligne : année, supercarburant, gazole.*

Par copier-coller, transférer la série de lignes précédente dans un fichiers `carburants.txt`*. Saisir le tableau dans une matrice* `Cc`*. Convertir ces prix, exprimés en euros courants,en euros de 2005 et les ramener de l'hectolitre au litre. Tracer sur un même graphique les évolutions des prix (en euros de 2005) des deux carburants de 1985 à 2005.*



4 Programmation

Scilab est un langage interprété où la transmission des arguments se fait par valeur, même pour les objets de type matrices. Il faut donc éviter les boucles qui peuvent être très inefficaces en terme de temps de calcul. Pour cela, il faut utiliser si possible des opérateurs vectoriels qui font la même chose.

Nous recommandons ici de disposer d'un éditeur (Emacs), d'ouvrir un fichier (par exemple `nom_du_fichier.sce`), d'y écrire les lignes de commandes puis de les exécuter en tapant sous Scilab `exec("nom_du_fichier.sce")`.

- condition `if`

```
p=0.4
y=rand(1)
if y< p then x=1;
elseif y>= p then x=0;
end
x
```

- boucles `while`

```
p=0.2
y=rand(1);
compteur=1;
while y >=p do compteur=compteur+1; y=rand(1); end
compteur
```

- boucles `for`

```
N=500;
y=rand(1,N)-0.5*ones(1,N);
x=sign(y);
for i=2:N,x(i)=x(i-1)+x(i);end;
xbasc();plot2d(1:N,x)
x1=cumsum(sign(y));
norm(x1-x)
```

5 Fonctions Scilab

On peut définir une fonction Scilab de façon interactive, mais il est souvent plus naturel d'écrire le code d'une fonction dans un fichier au moyen d'un éditeur de texte.

On peut mélanger des instructions Scilab et des définitions de fonctions dans un même fichier, qu'on fera terminer par l'extension `.sce`.

L'entête d'une fonction est constituée de la séquence
`function [<arguments de retour>]=<nom>(<arguments d'entrée>)`
et le corps de la fonction est terminé par le mot clef `endfunction`.

```
function [y]=cube(x), y=x.^3, endfunction;
// le nom de la fonction est cube
// elle a pour argument un vecteur x, et retourne le vecteur y=x.^3
```

On notera l'utilisation de l'opérateur `.^` (et non pas simplement `^`) car l'argument x est généralement un vecteur (et non pas un scalaire).

```
x=0.01:0.01:0.99;y=cube(x);
xbasc(); plot2d(x,y);
```

Voici un autre exemple de fonction.

```
function [H] = Heavyside(x)
  // Fonction de Heavyside
  H = bool2s(x>=0)
  // on notera que Heavyside(0)=1
endfunction

x=-1:0.2:1; xbasc(); plot2d2(x,Heavyside(x),rect=[-1,-0.1,1,1.2]);
```

Question 5 *Écrire la fonction sinus cardinal (*$\sin(x)/x$*) de telle sorte qu'elle puisse admettre un argument x vectoriel et retourne un vecteur de même taille contenant le sinus cardinal de chacun de ses éléments. Deux problèmes se posent : comment faire une division vectorielle ? comment éviter une divison par zéro ?*[1]

Une fonction dans Scilab peut avoir plusieurs arguments, scalaires ou vectoriels.

```
function [y]=Gauss(x,mu,sigma)
  // mu et sigma sont des scalaires
  // x peut être un vecteur de taille quelconque
  y=1 ./(sigma*sqrt(2*%pi))*exp(-((x-mu)/sigma).^2/2)
endfunction;

function [Y]=Normal(X,Mu,Sigma)
  // Mu est un vecteur ligne de taille [1,dim]
  // Sigma est une matrice symétrique définie positive de taille [dim,dim]
  // x peut seulement être un vecteur de taille [1,dim]
  [lhs,dim]=size(Mu)
  Y=1/ sqrt( (2*%pi)^dim * det(Sigma) ) * ...
        exp(- (X-Mu)*inv(Sigma)*(X-Mu)' / 2 )
endfunction;
```

Une fonction dans Scilab peut renvoyer plusieurs valeurs.

```
function [H,plus] = Heavyside_plus(x)
  // Fonction de Heavyside
  H = maxi(sign(x),0);
  // on notera que Heavyside(0)=0
```

[1] On fera attention au cas où $x = 0$ en évaluant la fonction $\sin(x)/x$, non pas en x, mais en `z=maxi(abs(x),%eps)`. Ainsi, $\sin(x)/x$ vaudra $\sin(|x|)/|x|$ pour $|x| \geq \varepsilon$ et $\sin(\varepsilon)/\varepsilon \approx 1$ pour $|x| \leq \varepsilon$, où ε est le "zéro-machine". Cette façon de faire est préférable à celle consistant à insérer une condition *if* $x = 0$ qui pose problème lorsque x est un vecteur et non pas un scalaire.

```
    plus=maxi(x,0)
    // Fonction +
  endfunction
```

Si on tape simplement

```
Heavyside_plus(-1:0.2:1)
```

seule la première valeur est retournée. Pour disposer de toutes les valeurs, il faut utiliser la syntaxe suivante qui attribue des noms à ces valeurs

```
[a,b] = Heavyside_plus(-1:0.2:1)
```

Les deux fonctions ci-dessous retournent une permutation de n éléments tirée au hasard. Interpréter le mécanisme de simulation de la première des deux. Il apparaît que cette méthode est plus lente que l'appel à la fonction Scilab `grand`.

```
function [x]=permutation_lent(n)
// permutation aléatoire de {1,...,n}
// version avec des boucles
  x=ones(1,n);
  for i=2:n,k=0;
    for j=1:1+int(rand(1)*(n-i+2));k=k+1;
      while x(k)>1 do
        k=k+1;
      end;
    end;
    x(k)=i;
  end;
endfunction

function [x]=permutation_rapide(n)
  // permutation aléatoire de {1,...,n}
  x=grand(1,'prm',[1:n]')
endfunction

// chargement de la fonction permutation_lent :
-->exec('nom_du_fichier.sce')
// la fonction est maintenant connue :
-->permutation_lent
// appel de la fonction permutation_lent :
-->permutation_lent(100)
```

6 Application : calculs d'emprunts

On considère un emprunt

- d'un montant K, *capital emprunté* ;
- au *taux annuel* $\theta \in [0,1]$;
- remboursé en T *périodes* ($T = 12N_a$ pour un remboursement mensuel sur N_a années) ;

et on veut connaître

- le montant R du *remboursement par période* ;
- le montant total I des *intérêts* ;
- pour chaque période $t = 1, ..., T$
  - K_t le capital restant dû ;
  - A_t le capital amorti ;
  - I_t les intérêts dus.

Le *taux actuariel* τ est (dans le cas de remboursements mensuels)

$$\text{soit } \textit{proportionnel} \quad \tau_p = \theta/12 \tag{1}$$
$$\text{soit } \textit{équivalent} \quad \tau_e = (1+\theta)^{1/12} - 1 \quad (\tau_e > \tau_p)\,. \tag{2}$$

On a les relations suivantes ($t = 1, ..., N$) :

$$I_t = \tau K_t \tag{3}$$
$$K_{t+1} = K_t - A_t \tag{4}$$
$$I_t + A_t = R \tag{5}$$

Avec $K_1 = K$, capital emprunté, on en déduit pour $t = 1, ..., T$:

$$K_t = R/\tau - (1+\tau)^{t-1}(R/\tau - K) \tag{6}$$
$$I_t = R - (1+\tau)^{t-1}(R - \tau K) \tag{7}$$
$$A_t = (1+\tau)^{t-1}(R - \tau K)\,. \tag{8}$$

Introduisons la fonction suivante (telle que $\frac{\partial \Phi}{\partial \tau} \geq 0$) :

$$\Phi(\tau, T) = \frac{\tau}{1 - (1+\tau)^{-T}}\,. \tag{9}$$

Avec $K_1 = K$ et $K_{T+1} = 0$ (plus de capital restant dû après la dernière échéance), on obtient alors facilement :

$$R = K\frac{\tau(1+\tau)^T}{(1+\tau)^T - 1} = K\Phi(\tau, T) \tag{10}$$

$$K = R\frac{(1+\tau)^T - 1}{\tau(1+\tau)^T} = \frac{R}{\Phi(\tau, T)} \tag{11}$$

$$I = \sum_{t=1}^{T} I_t = TR - K = K(T\Phi(\tau, T) - 1)\,. \tag{12}$$

```
function [mensualite,restant,amorti,interets]=...
         emprunt(capital,taux_annuel,annees)
  // On considère ici un emprunt de capital
  // à remboursements mensuels sur (12 * annees) périodes
  // au taux annuel taux_annuel
  if taux_annuel > 0.99 then
    error('Attention : mettre un taux de 4,5 pourcent sous la forme 0.045')
  end
  T=12*annees;
  tau=(1+taux_annuel).^(1/12)-1;
  // taux actuariel équivalent
  phi=tau/(1-(1+tau).^(-T));
  // fonction phi
  mensualite=capital*phi
  // mensualité
  periodes=[1:12*annees];
  // périodes de remboursement
  amorti=((1+tau).^(periodes-1)).*(mensualite-capital*tau);
  // capital amorti : vecteur de taille (1,12*annees)
  interets=mensualite-amorti;
  // intérêts : vecteur de taille (1,12*annees)
  restant=interets/tau;
  // capital restant : vecteur de taille (1,12*annees)
endfunction
```

Question 6 *Copier le code Scilab ci-dessus dans un fichier* `calculs_emprunt.sce` *et charger la fonction* `emprunt` *par la commande* `exec("calculs_emprunt.sce")`. *Choisir des paramètres d'emprunt, puis étudier l'effet d'une variation de 1 % du taux annuel. Modifier le code précédent pour qu'il fournisse également en sortie le montant total des intérêts.*

```
xbasc();
capital=100000;
taux_annuel=0.05;
annees=10;
[mensualite,restant,amorti,interets]=emprunt(capital,taux_annuel,annees);
periodes=[1:12*annees];
plot2d(periodes,restant);
[mensualite_1,restant_1,amorti_1,interets_1]=...
        emprunt(capital,taux_annuel+0.01,annees);
plot2d(periodes,restant_1);

sum(interets_1)/sum(interets)-1
// variation des intérêts versés
```

Question 7 *Programmer une fonction* `capital_emprunt` *de la forme* `[capital]=capital_emprunt(mens` *donnant le capital pouvant être emprunté en fonction de la mensualité, du taux et du nombre d'années.*

Si l'on veut calculer, pour un plan d'épargne logement (PEL), le capital que l'on peut emprunter et les mensualités, il faut procéder comme suit. Un PEL terminé se caractérise par

- des *droits à prêt* D (intérêts acquis sur le PEL) ;
- le *taux actuariel* τ_{el} *du* PEL.

Le capital empruntable K dépend du nombre N_a d'années d'emprunt : il est tel que les intérêts perçus I (par le prêteur pour un prêt au taux τ_{el}) soient égaux à $2,5$ fois les intérêts acquis (sur le PEL), soit

$$I = 2,5 \times D \tag{13}$$

(2,5 doit être remplacé par 1,5 pour un *livret* ou *compte d'épargne*). On en déduit

$$T = 12 \times Na \quad \text{et} \quad K = 2,5 \times \frac{D}{T\Phi(\tau_{el}, T) - 1}\,. \tag{14}$$

Enfin, le capital est plafonné à K_{max} (92 000 euros en 2003) et le prêt est accordé à un taux (actuariel) $\tau > \tau_{el}$:

$$K = \min(K_{max}; 2,5 \times \frac{D}{T\Phi(\tau_{el}, T) - 1}) \tag{15}$$

$$R = K\Phi(\tau, T)\,. \tag{16}$$

Question 8 *Programmer une fonction* `PEL_emprunt` *de la forme* `[capital,mensualite]=PEL_emprunt(d` *donnant le capital pouvant être emprunté et le montant des remboursements, en fonction des droits à prêt, du taux annuel du* PEL *et du nombre d'années d'emprunt.*

7 Graphiques 3D

Question 9 • *Saisir le tableau suivant sous forme de matrice*

```
M=[ 10.66   13.75   16.28   18.17   19.34   19.73   19.34   18.17   16.28   13.75   10.66
    11.31   14.59   17.28   19.28   20.52   20.93   20.52   19.28   17.28   14.59   11.31
    10.99   14.17   16.79   18.73   19.93   20.34   19.93   18.73   16.79   14.17   10.99
     9.00   11.61   13.75   15.34   16.33   16.66   16.33   15.34   13.75   11.61    9.00
     4.66    6.01    7.12    7.95    8.45    8.63    8.45    7.95    7.12    6.01    4.66
    -2.01   -2.60   -3.08   -3.43   -3.65   -3.73   -3.65   -3.43   -3.08   -2.60   -2.01
    -8.98  -11.58  -13.72  -15.31  -16.29  -16.63  -16.29  -15.31  -13.72  -11.58   -8.98
   -11.02  -14.21  -16.83  -18.78  -19.99  -20.39  -19.99  -18.78  -16.83  -14.21  -11.02
    -2.63   -3.39   -4.01   -4.48   -4.76   -4.86   -4.76   -4.48   -4.01   -3.39   -2.63
    10.14   13.08   15.49   17.29   18.40   18.77   18.40   17.29   15.49   13.08   10.14
     4.42    5.69    6.75    7.53    8.01    8.17    8.01    7.53    6.75    5.69    4.42
   -11.34  -14.63  -17.33  -19.33  -20.57  -20.99  -20.57  -19.33  -17.33  -14.63  -11.34
```

- *Faire* ***help plot3d****.*
- *On suppose que le terme M(i,j) représente la cote z du point (i,j,z). Tracer la surface correspondante.*
- *Faire* ***help contour*** *et tracer les courbes de niveau.*