Matlab : une introduction

´ François Langot
CEPREMAP et Université du Maine

Cette note a pour objectif de donner les bases nécessaire à l'utilisation du logiciel Matlab. Les conventions suivantes sont adoptées:

- le texte en caractère `typeset` décrit des codes Matlab,
- un ";" à la fin de la ligne indique à Matlab de ne pas afficher la commande qu'il exécute.
- De nombreuses fonctions et instructions ne sont pas détaillées dans cette note, mais leurs codes sont donnés en annexe.

1 Manipulations de vecteurs et de matrices

1.1 Construire une Matrice

Il y a plusieurs manières pour construire une matrice avec Matlab.

- La première, et peut être la plus simple, est de déclarer la matrice comme on l'écrit à la main:

  ```
  A=[1 2 3
     4 5 6]
  ```

  Ceci crée une matrice (2×3) matrix de la forme:

  $$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

- Une autre possibilité est d'écrire:

  ```
  A=[1 2 3;4 5 6]
  ```

  Ces exemples montrent que dans les crochets, un espace sert à séparer les colonnes, alors qu'un ";" sert à séparer les lignes de la matrice.

- Finalement, il est possible de déclarer une matrice élément par élément:

```
A(1,1)=1;
A(1,2)=2;
A(1,3)=3;
A(2,1)=4;
A(2,2)=5;
A(2,3)=6;
```

1.2 Les matrices particulières

Il y a des matrices particulières qu'il est très utile de connaître:

- La matrice zéro: on crée une matrice $(r \times c)$ de 0 en utilisant l'instruction `zeros(r,c)`. Par exemple:

```
A=zeros(2,3);
```

  crée la matrice:
$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- La matrice de un: on peut créer une matrice $(r \times c)$ de 1 en utilisant l'instruction `ones(r,c)`. Par exemple:

```
A=ones(2,3);
```

  crée la matrice:
$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- la matrice identité I_n peut être crée grâce à la commande `eye(n)`, où n est la dimension de la matrice. Ainsi,

```
A=eye(3);
```

  crée la matrice:
$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- une matrice aléatoire: on peut créer une matrice $(r \times c)$ d'éléments aléatoires en utilisant la commande `rand(r,c)`, pour des éléments uniformément distribués, ou `randn(r,c)`, pour des éléments normalement distribués. Il doit être noté que `rand` tire des nombre sur le support [0;1] alors que `randn` tire des nombres dans la une loi normale $\mathcal{N}(0,1)$.

- La matrice vide peut être utile lors de l'initialisation d'une matrice:

  ```
  A=[ ];
  ```

  définit `A` comme la matrice vide.

1.3 Manipulations de base

Une des opérations de base les plus courante consiste à extraire des éléments d'une matrice. Matlab donne un ensemble d'instruments très puissants pour effectuer ces opérations.

Considérons la matrice:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix}$$

- On veut isoler la matrice centrale:

  $$B = \begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 1 \end{pmatrix}$$

  Pour obtenir B, les commandes Matlab sont:

  ```
  B=A(1:4,2:3)
  ```

  `1:4` signifie la séquence 1 2 3 4 et `2:3` la séquence 2 3, de telle sorte que `B=A(1:4,2:3)` est l'instruction qui va permettre de définir B comme une sélection des ligne 1 2 3 4 de `A` et de colonnes 2 et 3.

- Supposons que nous voulions sélectionner les colonnes 1 et 3, mais que nous voulions garder toutes les lignes. L'instruction est alors:

  ```
  B=A(:,[1 3]);
  ```

  Le : signifie "sélectionner tout", alors que `[1 3]` est juste un vecteur contenant les nombres 1 et 3. Ainsi, tous les éléments des colonnes 1 et 3 sont sélectionné.

- Mais des opérations plus complexes peuvent être envisagées. Imaginons que l'on veuille sélectionner les éléments de la matrice A supérieur ou égal à 3, afin de les stocker dans un vecteru B:

```
B=A(A>=3);
```

Si $A(i,j) >= 3$ alors stocker dans le **vecteur** B.

- Maintenant, considérons la matrice:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

et supposons que nous voulions en obtenir la vectorialisation, alors il suffit de donner l'instruction:

```
B=A(:);
```

pour obtenir:

$$B = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 2 \\ 4 \\ 6 \end{pmatrix}$$

- Si on veut obtenir la matrice A à partir de la matrice B, il suffit d'utiliser la commande `reshape`:

```
A=reshape(B,3,2)
```

ce qui signifie que le vecteur B prendra la forme d'une matrice (3×2), comme la matrice A.

- Si l'on veut agglomérer des matrices, il suffit de suivre les instructions suivantes. Supposons que nous ayons les deux matrices suivantes:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \text{ et } B = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

et que l'on veuille la matrice suivante:

$$C = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 4 & 5 & 6 \\ 0 & 0 & 7 & 8 & 9 \end{pmatrix}$$

alors, il suffit d'écrire:

```
C=[A zeros(2,3);zeros(2,2) B];
```

- Le tableau suivant résume les autres manipulations possibles:

`rot90(A)`	rotation de A
`diag(A)`	crée ou extrait la diagonale de A
`tril(A)`	partie triangulaire inf. de A
`triu(A)`	partie triangulaire sup. de A

1.4 Opérations de base

Le tableau suivant résume les opérations matricielles disponibles sous Matlab:

Action	équivalent Math.	Matlab	Commentaire
taille	A est $(r \times c)$	`[r,c]=size(A)`	
Transposition	A'	`A'`	
Addition	$A + B$	`A+B`	$dim(A) = dim(B)$
Produit	$A\,B$	`A*B`	Compatilité
Produit élément par élément	$A_{ij} B_{ij}$	`A.*B`	$dim(A) = dim(B)$
Division 1	X solution de $A\,X = B$	`A\ B`	Compatilité
Division 2	X solution de $X\,A = B$	`A\ B`	Compatilité
Division élément par élément	A_{ij}/B_{ij}	`A./B`	$dim(A) = dim(B)$
Puissance d'une matrice	A^n	`A^ n`	A carrée
Puissance élément par élément	A_{ij}^n	`A.^ n`	
Trace	$tr(A)$	`trace(A)`	A carrée
Déterminant	$det(A)$	`det(A)`	A carrée
Produit de Kronecker	$A \otimes B$	`kron(A,B)`	
Inverse	A^{-1}	`inv(A)`	A carrée
Rang	rang(A)	`rank(A)`	
Noyau	$(A\,V = 0)$	`V=null(A)`	
Valeur propre	$A = \,D\,P^{-1}$	`[P,D]=eig(A)`	A carrée
Somme des colonnes	$\sum_{i=1}^{rows(A)} A_{i.}$	`sum(A)`	
Produit des colonnes	$\Pi_{i=1}^{rows(A)} A_{i.}$	`prod(A)`	

2 Contrôler la séquence des instructions

Comme beaucoup d'autre langage, Matlab peut contrôler la séquence des instructions d'un programme. Il y a trois façons de faire.

2.1 La boucle avec `for`

Matlab peut répéter un ensemble d'instruction un nombre donné de fois, en utilisant l'instruction `for`. La forme générale de la boucle est alors:

```
for variable=expression;
    instruction;
end;
```

En fait, `expression` est une matrice contenant des nombres.

Exemple 1 : Générer la matrice A t.q.:

$$A_{ij} = i^2 - j^3 + 1, \text{ pour } i = 1, \ldots, 20 \text{ et } j = 1, \ldots, 10$$

Les codes Matlab correspondant sont:

```
for i=1:1:20;          % i va de 1 a 20 avec un pas de 1 : for i=init:step:final
  for j=1:1:10;        % j va de 1 a 10 avec un pas de 1 : for j=init:step:final
    A(i,j)=i^2-j^3+1;
  end;
end;
```

Exemple 2 : Calculer A_{ij}^b, pour b allant de 0 à 1 avec un pas de 0.1, et afficher le résultat à chaque itération. Les codes sont:

```
A=[1 2;2 1];
for b=0:0.1:1;         % i va de 0 a 1 avec un pas de 0.1
  C=A.^b;
  disp(C)              % afficher C
end;
```

2.2 La boucle avec `while`

Matlab peut aussi utiliser des boucles pour répéter des instructions jusqu'à ce qu'une condition terminale soit satisfaite. La syntaxe générale des boucles avec `while` est la suivante:

```
while condition;
    instructions;
end;
```

Tant que `condition` est satisfait, les `instructions` seront exécutées.

Exemple 1 : trouver un point fixe de la relation dynamique suivante (pas de solution analytique):

$$x_{t+1} = 1 + x_t^{0.2}$$

Tout ce que l'on peut faire, c'est itérer sur cette relation et stopper quand la différence en valeur absolue entre deux itérations est inférieur à un seuil de tolérance. Les codes sont:

```
tol=1e-6;              % critere de tolerance
crit=1;                % initialisation du critere
x0=1;                  % valeur initiale de x
while crit>tol;
  x=1+x0^0.2;          % relation dynamique
  crit=abs(x-x0);      % valeur absolue entre 2 iterations consecutives
  x0=x;                % la nouvelle valeur de x(t)
end;
```

2.3 La boucle avec `if`

L'instruction **`if`** exécute un ensemble de commande si une condition est satisfaite. La syntaxe générale est:

```
if condition
  commandes 1
else
  commandes 2
end
```

Exemple 1 : savoir si un entier est impair ou non. Les codes sont alors les suivants:

```
x=input('entrer un entier: ');    % Demande a l'utilisateur d'entrer un entier
                                  % et le stock dans le vecteur x
if rem(x,2)==0;                   % si le reste de la division par 2 est nul, alors
  disp('x est pair');             % afficher le message: x est pair
else;                             % sinon
  disp('x est impair');           % afficher le message: x est impair
end;                              % fin du test
```

Ces commandes peuvent être enrichies de la façon suivante:

```
if condition
  commandes 1
elseif condition
  commandes 2
else
  commandes 3
end
```

Exemple 2 : Construction d'une fonction de demande de la forme:

$$D(P) = \begin{cases} 0 & \text{si } p \leq 2 \\ 1 - 0.5P & \text{si } 2 < p \leq 3 \\ 2P^{-2} & \text{sinon} \end{cases}$$

Les codes sont les suivant:

```
P=input('entrer un prix : '); if P<=2;
  D=0;
elseif (P>2)&(P<=3);
  D=1-0.5 P;
else;
  D=2*P^(-2);
end;
disp(D);
```

Ces instructions peuvent évidement être combinées les unes avec les autres. L'exemple 3 donne une illustration de la combinaison entre les instructions `for` et `if`, en utilisant l'instruction `break` qui permet de sortir d'une boucle avant la fin de celle-ci.

Exemple 3: Construire la séquence suivante:

$$x_t = x_{t-1}^{0.4} - x_{t-1}^{0.2}$$

pour $t = 1, \ldots, 100$. On doit s'assurer que x reste positif autrement Matlab trouvera une valeur complexe! Les codes suivant permettent de générer cette séquence et nous font sortir de la boucle dès que le résultat d'une itération est négatif:

```
x(1)=2;
for i=2:1:100;
  y=x(i-1)^0.4-x(i-1)^0.2;
  if y<0;
    disp('le resultat ne peut pas etre negatif');
    break;
  end;
  x(i)=y;
end;
```

Avec cette ensemble de trois instructions, il est normalement possible de résoudre tous vos problèmes numériques à l'aide de Matlab. Toutefois, une autre capacité de Matlab est de vous permettre de construire des nouvelles fonctions.

3 Définition et utilisation des vos propres fonctions

Les fonctions sont des sous-programmes annexes qu'il est possible d'appeler avec des codes usuels. La syntaxe générale pour définir une fonction est:

```
function [output]=nom_de_la_fonction(input);
     instructions;
```

où `output` désigne le vecteur des résultats, et `input` désigne le vecteur des paramètres à entrer pour la résolution de la fonction.

Cette fonction doit être **sauvée dans un fichier texte ("script file") portant le même nom que la fonction**.

Par exemple, construisons une fonction, appelée `stat`, qui nous donne la moyenne et l'écart-type d'un vecteur. On doit alors créer, à l'aide de votre éditeur favori, un fichier nommé `stat.m`, contenant le texte suivant:

```
function [m,st]=stat(x);

lx=length(x);
m=sum(x)/lx;
st=sqrt(sum(x-mx)^2)/lx;
```

Soit le vecteur:

```
x=[1;2;3;4;5;6];
```

En appelant `[mx,sx]=stat(x)` dans un autre programme, vous obtiendrez `mx=3.5000` et `sx=1.8708`.

Les fonctions sont très importantes pour le traitement de certain problème. Un des plus important en économie est le calcul de l'état stationnaire d'un modèle. Celui-ci peut être résolu en utilisant la routine `fsolve` qui résout les systèmes non-linéaires.

Par exemple, dans le cas du modèle néo-classique de croissance, on doit résoudre le suivant suivant[1]:

$$\begin{aligned} 1 &= \beta(\alpha A k^{\alpha-1} + 1 - \delta) \\ \delta k &= A k^{\alpha} - c \end{aligned}$$

Il faut alors créer une fonction à deux variables, appelée par exemple `steady`, dans un fichier appelé `steady.m`:

```
function z=steady(x);

alpha=0.35;
beta=0.99;
delta=0.025;
A=1;

k=x(1);
c=x(2);

z=zeros(2,1);                              % Initialisation de z.
                                           % Pas necessaire, mais recommande
z(1)=1-beta*(alpha*A*k^(alpha-1)+1-\delta); % Remarque: la fonction est
z(2)=delta*k-(A*k^alpha-c);                % ecrite f(x)=0
```

et, dans un autre fichier, donner les conditions initiales afin de permettre la résolution numérique de ce système:

```
k0=10;
c0=1;
x0=[k0;c0];
sol=fsolve('steady',x0);
```

Un autre avantage des fonctions est qu'elles permettent de vous créer une librairie que vous pouvez utiliser quand vous en avez besoin pour diverses problèmes.

[1] Remarque: cet exemple peut être résolu à la main, mais il a été choisi pour sa simplicité.

4 Outils Input–Output

Les instructions Input – Output sont importantes car elles permettent d'afficher, de sauver et d'utiliser vos résultats dans d'autres codes que ceux utilisés par Matlab.

4.1 Textes Input – Output

La première commande est celle permettant d'afficher une commande: `disp`. Elle vous permet d'afficher un message, une valeur ou ce que vous voulez comme texte. Ainsi

```
disp('Ceci est un beau message')
```

affichera "`Ceci est un beau message`" à l'écran.

Soit une matrice A de la forme:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

alors:

```
disp(A)
```

affichera à l'écran:

```
1 2
3 4
```

Si vous désirez stocker vos résultats dans un fichier, il faut ouvrir un fichier et faire afficher vos résultats sous la forme que vous voulez. Pour cela, utiliser la commande `diary name_of_file`. Ne pas oublier de fermer votre fichier avec la commande `diary off`. Par exemple:

```
diary result.out
disp('Salut')
diary off
```

crée un fichier appelé `result.out` contenant "`Salut`".

Attention: l'instruction `diary` n'écrase pas le(s) fichier(s) existant sous le même nom. Il est donc nécessaire, dans un premier temps, d'effacer le(s) vieu(x) fichier(s) en utilisant `delete name_of_file`, si vous ne voulez pas ajouter de nouvelles informations dans le fichier existant, mais seulement avoir un fichier donnant les nouveaux résultats.

Si on désire que l'utilisateur du programme entre une information avec le clavier, utiliser l'instruction `input`.

```
n=input('Donner un nombre');
```

affiche le message "`Donner un nombre`" à l'écran en attendant une réponse. La réponse est enregistrée dans la variable `n`.

Si on désire charger des données, vous avez simplement à créer un fichier ASCII contenant ces données. Par exemple supposons que le fichier de données `dat.txt` est de la forme:

$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{array}$$

Donner juste l'instruction:

```
load dat.txt -ascii;
```

et vos données sont stockées dans une matrice, appelée `dat`, que vous pouvez manipuler comme toutes les matrices.

4.2 Graphiques

La première instruction importante est `clg`: elle permet d'effacer le graphique à l'écran.

Dans cette note d'introduction à Matlab, seuls les graphiques en 2-d seront abordés (se sont les plus commun)[2].

Il y a essentiellement une instruction: `plot`. La syntaxe générale de `plot` est:

```
plot(X,Y,S)
```

où `X` et `Y` sont des matrices et `S` est une chaîne de caractères 1, 2 ou 3 définissant l'aspect du graphique. L'instruction ci-dessus dessine `Y` comme une fonction de `X`. La chaîne `S` est une option et peut prendre les valeurs suivantes:

y	jaune	.	point
m	magenta	o	cercle
c	cyan	x	x-marque
r	rouge	+	plus
g	vert	-	trait plein
b	bleu	*	étoile
w	blanc	:	pointillé
k	noir	-.	tirets-point
		–	traits espacés

Ainsi `plot(X,Y,'b*')` dessine une étoile bleue en chaque point de l'échantillon.

Il est possible d'ajouter des titres, des légendes pour les axes, en utilisant les instructions suivantes:

[2]Pour le graphiques en 3-d, voir le manuel "Matlab Graphics".

```
title(string)      %
xlabel(string)     % titre de l'axe des x
ylabel(string)     % titre de l'axe des y
```

Il est également possible de rajouter une grille sur votre graphique, en utilisant l'instruction `grid`.

Les graphiques, créés à partir des instructions suivantes sont reportés en annexe (A). Le graphique (1) est défini par les codes suivants:

```
x=[-3:.01:3];
y1=exp(-0.5*x.^2)/sqrt(2*pi);
y2=exp(-0.25*x.^2)/sqrt(2*pi);
clg;
plot(x,y1,'w',x,y2,'w--');
title('Un bel exemple  ?');
ylabel('Les fonctions');
xlabel('Les
valeurs');
grid;
```

Il est également possible d'avoir simultanément plusieurs graphiques à l'écran. Pour cela, utiliser la commande `subplot(rcn)` où `r`, `c`, `n` sont respectivement le numéro de la ligne, de la colonne et du graphique. Ainsi, le graphique (4) de l'annexe est défini par les codes:

```
x=[-3:.01:3];
y1=exp(-0.5*x.^2)/sqrt(2*pi);
y2=sin(x);
y3=cos(x);
y4=abs(sqrt(x));
clg;
subplot(221);plot(x,y1);ylabel('Y1');xlabel('X');title('Gauss');
subplot(222);plot(x,y2);ylabel('Y2');xlabel('X');title('Sin(X)');
subplot(223);plot(x,y3);ylabel('Y3');xlabel('X');title('Cos(X)');
subplot(224);plot(x,y4);ylabel('Y4');xlabel('X');title('Abs(Sqrt(X))');
```

Pour sauver un graphique dans un fichier, il suffit d'utiliser l'instruction print. La syntaxe générale est:

```
print -options nom_du_fichier
```

D'autres détails sont donnés en annexe (C).

5 Applications économiques: le modèle de cycles réels (RBC)

L'objectif de cette section est de montrer comment résoudre numériquement, à l'aide de Matlab, le modèle d'équilibre général à horizon de vie infini, dans un cadre stochastique. Ce modèle est le cadre de référence des modèles RBC.

Le programme est écrit dans un fichier rbc.m, et les résultats de ce programme dans un fichier rbc.res. Ainsi, au début du programme on a les codes suivants:

```
clear
delete rbc.res
diary rbc.res
```

5.1 Le modèle économique

1. $\exists$ un continuum d'agents identiques, indicés par $i \in [0,1]$.
2. Ceux-ci sont à la fois producteur et consommateur,

L'agent i maximise la fonction d'utilité suivante:

$$\max_{c_{i,s},l_{i,s},k_{i,s+1}} \sum_{s=t}^{\infty} E_t \beta^{s-t} \left[\log(c_{i,s}) - \frac{l_{i,s}^{1+\chi}}{1+\chi} \right]$$

sous la séquence de contraintes suivante:

$$c_{i,t} + k_{i,t+1} \leq (1-\delta)k_{i,t} + p_{i,t}y_{i,t} \quad \forall\ t = 1, \cdots, \infty$$

- $p_{i,t}$ prix de la production de l'agent i relativement aux prix des biens consommés,
- $k_{i,t}$, $l_{i,t}$, $c_{i,t}$ et $y_{i,t}$ représentent respectivement le stock de capital, le nombre d'heure de travail, la consommation et la production de l'agent i,
- les paramètres $\delta \in [0,1)$, $\beta \in [0,1)$ et $\chi \in [0,\infty)$ représentent respectivement le taux de dépréciation, le facteur d'escompte psychologique et l'inverse de l'élasticité de l'offre de travail

Chaque agent a accès à une technologie de production lui permettant de produire $y_{i,t}$:

$$y_{i,t} = s_t k_{i,t}^m (\gamma^t l_{i,t})^{1-m}$$

- γ taux de croissance du P.T. incorporé au travail,
- s_t choc de progrès technique, incertitude intrinsèque,
- fonction de production de l'agent i: rendements constants,
- le marché concurrentiel, $p_{i,t} = p_{j,t} = 1\ \forall\ i,j$ (le bien final est le numéraire).

En supposant que la contrainte budgétaire de l'agent est saturée à chaque période (*i.e.* l'agent utilise ces revenus), le problème de l'agent i peut se réécrire de la façon suivante:

$$\max_{k_{s+1},l_{i,s}} \sum_{s=t}^{\infty} E_t \beta^{s-t} \left[\log\left(c_{i,s+1}\right) - \frac{l_{i,s}^{1+\chi}}{1+\chi} \right]$$

$$\text{avec} \quad c_{i,s} = -k_{i,s+1} + (1-\delta)k_{i,s} + s_s k_{i,s}^m (\gamma^t l_{i,s})^{1-m}$$

Les conditions d'optimalité de ce problème sont:

$$\begin{aligned}
\frac{-1}{c_{i,t}} + \beta E_t \left[\frac{1}{c_{i,t+1}} \left(1 - \delta + \frac{\partial y_{i,t+1}}{\partial k_{i,t+1}} \right) \right] &= 0 \\
\frac{1}{c_{i,t}} \frac{\partial y_{i,t}}{\partial l_{i,t}} - l_{i,t}^{\chi} &= 0 \\
(1-\delta) k_{i,s} + s_s k_{i,s}^m (\gamma^t l_{i,s})^{1-m} - c_{i,s} &= k_{i,s+1}
\end{aligned}$$

5.2 Conditions d'équilibre

$\forall\ i$, les conditions d'optimalité sont identiques:

$$\begin{aligned}
\frac{1}{c_t} &= \beta E_t \left[\frac{1}{c_{t+1}} \left(1 - \delta + m \frac{y_{t+1}}{k_{t+1}} \right) \right] \\
l_t^{\chi} &= \frac{1}{c_t} \left((1-m) \frac{y_t}{l_t} \right)
\end{aligned}$$

Les trajectoires d'équilibre vérifient également:

$$\begin{aligned}
k_{t+1} &= (1-\delta) k_t + y_t - c_t \\
y_t &= s_t k_t^m (\gamma^t l_t)^{1-m} \\
s_t &= s_{t-1}^{\rho} v_t
\end{aligned}$$

où $\rho \in (0,1)$ représente la persistance du choc technologique et v_t est une innovation *iid*.

Remarque:

Il est possible de réduire la dimension du problème substituant l'offre de travail par l'expression suivante

$$l_t = \left((1-m) \frac{s_t k_t^m}{c_t} \right)^{\frac{1}{(\chi+m)}}$$

où l_t est une fonction de c_t, k_t et s_t.

5.3 Sentier de croissance déterministe

- hypothèses sur les fonctions de production et d'utilité $\Longrightarrow$ existence d'un sentier de croissance équilibrée,
- tendance $\Longrightarrow$ progrès technique incorporé au travail (fonction de γ),
- exprimer le modèle en variables intensives $\Longrightarrow$ déflater l'ensemble des équations définissant l'équilibre par γ^t

$$\begin{aligned}
\gamma\left(\frac{k_{t+1}}{\gamma^{t+1}}\right) &= (1-\delta)\left(\frac{k_t}{\gamma^t}\right)+\left(\frac{y_t}{\gamma^t}\right)-\left(\frac{c_t}{\gamma^t}\right)\\
\left(\frac{y_t}{\gamma^t}\right) &= s_t\left(\frac{k_t}{\gamma^t}\right)^m l_t^{1-m}\\
(1-\theta)\left(\frac{y_t}{\gamma^t}\right) &= \left(\frac{c_t}{\gamma^t}\right) l_t^{\chi+1}\\
\left(\frac{\gamma^t}{c_t}\right) &= \left(\frac{\beta}{\gamma}\right) E_t\left[\left(\frac{\gamma^{t+1}}{c_{t+1}}\right)\left\{1-\delta+m\left(\frac{y_{t+1}}{\gamma^{t+1}}\right)\left(\frac{\gamma^{t+1}}{k_{t+1}}\right)\right\}\right]
\end{aligned}$$

On définit alors les variables stationnarisées:

$$\tilde{c}_t = c_t/\gamma^t \ ; \quad \tilde{k} = k_t/\gamma^t \ ; \quad \tilde{y}_t = y_t/\gamma^t$$

5.4 Dynamique autour du sentier de croissance et état stationnaire

Les équations définissant l'équilibre se réécrivent:

$$\begin{aligned}
\frac{1}{\tilde{c}_t} &= \frac{\beta}{\gamma}E_t\left[\frac{1}{\tilde{c}_{t+1}}\left(1-\delta+m\frac{\tilde{y}_{t+1}}{\tilde{k}_{t+1}}\right)\right]\\
l_t^{\chi} &= \frac{1}{\tilde{c}_t}\left((1-m)\frac{\tilde{y}_t}{l_t}\right)\\
\phi\tilde{k}_{t+1} &= (1-\delta)\tilde{k}_t+\tilde{y}_t-\tilde{c}_t\\
\tilde{y}_t &= s_t\tilde{k}_t^m l_t^{1-m}
\end{aligned}$$

Ces équations déterminent la dynamique d'équilibre autour du sentier de croissance équilibrée.

L'état stationnaire est défini par le quadruplet $\{\overline{c},\overline{k},\overline{y},\overline{l}\}$ unique solution du système:

$$\begin{aligned}
\gamma\overline{k} &= \overline{y}+(1-\delta)\overline{k}-\overline{c}\\
\overline{y} &= s\overline{k}^m\overline{l}^{1-m}\\
(1-m)\frac{\overline{y}}{\overline{l}} &= \overline{c}\overline{l}^{\chi}\\
1 &= \frac{\beta}{\gamma}\left(1-\delta+m\frac{\overline{y}}{\overline{k}}\right)
\end{aligned}$$

5.5 Résolution du modèle: l'approximation log-linéaire

En substituant l_t par son expression en k_t, c_t et s_t, on réduit la dimension du système dynamique. Celui-ci est alors de $dim = 3 \times 3$:

$$\begin{aligned}
\gamma \tilde{k}_{t+1} &= \psi s_t^{b_5} \tilde{k}_t^{b_2} \tilde{c}_t^{b_3} + (1-\delta)\tilde{k}_t - \tilde{c}_t \\
\frac{1}{\tilde{c}_t} &= \frac{\beta}{\gamma} E_t \left[\frac{1}{\tilde{c}_{t+1}} \left(1 - \delta + \theta \psi s_{t+1}^{b_5} \tilde{k}_{t+1}^{b_4} \tilde{c}_{t+1}^{b_3} \right) \right] \\
\log(s_{t+1}) &= \rho \log(s_t) + \epsilon_{t+1}
\end{aligned}$$

où l'on note:

b_1	b_2	b_3	b_4	b_5	ψ
$1/(\chi+m)$	$m(\chi+1)b_1$	$-(1-m)b_1$	b_2-1	$(\chi+1)b_1$	$(1-m)^{-b_3}$

La résolution numérique de ce problème nécessite donc de donner une valeur aux paramètres des fonctions de comportements et du processus de variables exogène. Dans le fichier Matlab, ceci se fait de la façon suivante:

```
beta=.99;
delta=.025;
chi=0;
gamma=1.004;
rhos=.95;
epsilon=.0072;
m=.36;
mu=m;
nu=1-m;
phi=gamma^(nu/(1-mu));
```

Les coefficients b_i alors peuvent être définis dans le programme Matlab de la façon suivante:

```
b1=1/(chi- nu +1);
b2=mu*(chi+1)*b1;
b3=-nu*b1;
b4=b2-1;
b5=(chi+1)*b1;
psi=(1-m)^(-b3);
```

L'état stationnaire peut alors être calculé par Matlab, étant donné la définition de ces coefficients. Les codes sont les suivants[3]:

[3] La résolution de cet état stationnaire peut être effectué en définissant une fonction telle que celle donnée dans l'exemple "steady".

```
AA=(phi+delta-1)-(1/(beta*m))*(phi-beta*(1-delta));
BB=((phi/beta)-1+delta)*(1/(m*psi));

k=(-(BB^(1/b3))/AA)^(1/(1+(b4/b3)));

y=((phi/beta)-1+delta)*(k/m);

l=(y/(k^mu))^(1/nu);

c=y-(delta+phi-1)*k;

I=y-c;
```

Il est alors possible de calculer l'utilité, les utilité marginales, ainsi que les dérivées des utilités marginales. Les code sont les suivants:

```
util=log(c)-(1/(1+chi))*(l^(1+chi));  %utilite d'etat stationnaire
uc=1/c;                               %utilite marg de la conso
ucc=-1/(c^2);                         %derive seconde par rapport
                                      %\`a la conso
xcc=ucc*c/uc;                         %elasticite de l'utilite marg de la conso
ul=-l^chi;                            %desutilite marg du trav ;
ull=-chi*(l^(chi-1));                 %derive seconde par rapport au trav
xll=ull*l/ul;                         %elasticite de la desutilite
                                      %marg du trav
```

Toutes les équations définissant la dynamique d'équilibre ont la forme générique suivante:

$$f(\tilde{k}_t, \tilde{c}_t, s_t) = E_t[g(\tilde{k}_{t+1}, \tilde{c}_{t+1}, s_{t+1})]$$

Le développement de Taylor jusqu'au premier ordre permet d'approximer, autour de $\{\overline{k}, \overline{c}, \overline{s}\}$, toutes les conditions d'équilibre comme suit, les dérivées des fonctions f et g étant évaluées au point $(\overline{k}, \overline{c}, \overline{s})$:

$$f(\overline{k}, \overline{c}, \overline{s}) + \frac{\partial f}{\partial k}(\tilde{k}_t - \overline{k}) + \frac{\partial f}{\partial c}(\tilde{c}_t - \overline{c}) + \frac{\partial f}{\partial s}(s_t - \overline{s}) =$$

$$E_t\left[g(\overline{k}, \overline{c}, \overline{s}) + \frac{\partial g}{\partial k}(\tilde{k}_{t+1} - \overline{k}) + \frac{\partial g}{\partial c}(\tilde{c}_{t+1} - \overline{c}) + \frac{\partial g}{\partial s}(s_{t+1} - \overline{s})\right]$$

$$\iff \quad \eta_{f,k}\frac{(\tilde{k}_t - \overline{k})}{\overline{k}} + \eta_{f,c}\frac{(\tilde{c}_t - \overline{c})}{\overline{c}} + \eta_{f,s}\frac{(s_t - \overline{s})}{\overline{s}} =$$

$$E_t\left[\eta_{g,k}\frac{(\tilde{k}_{t+1} - \overline{k})}{\overline{k}} + \eta_{g,c}\frac{(\tilde{c}_{t+1} - \overline{c})}{\overline{c}} + \eta_{g,s}\frac{(s_{t+1} - \overline{s})}{\overline{s}}\right]$$

$$\Longleftrightarrow \quad \eta_{f,k}\widehat{k}_t + \eta_{f,c}\widehat{c}_t + \eta_{f,s}\widehat{s}_t =$$

$$E_t\left[\eta_{g,k}\widehat{k}_{t+1} + \eta_{g,c}\widehat{c}_{t+1} + \eta_{g,s}\widehat{s}_{t+1}\right]$$

où $\eta_{i,j}$, pour $i = f, g$ et $j = k, c, s$, sont les élasticités des fonctions f et g par rapport à k, c et s. Par exemple, $\eta_{f,k} = \frac{\partial f}{\partial k}\frac{\overline{k}}{f(\overline{k},\overline{c},\overline{s})}$.

En appliquant cette méthode d'approximation à toutes les équations, on obtient le système linéaire d'équations sous anticipations rationnelles suivant (à l'état stationnaire $\overline{s} = 1$):

$$\begin{aligned}
\widehat{k}_{t+1} &= c_1\widehat{k}_t + c_2\widehat{c}_t + c_3\widehat{s}_t \\
-\widehat{c}_t &= E_t\left[c_4\widehat{k}_{t+1} + c_5\widehat{c}_{t+1} + c_6\widehat{s}_{t+1}\right] \\
\widehat{s}_{t+1} &= \rho\widehat{s}_t + v_{t+1}
\end{aligned}$$

où les coefficients c_i sont combinaisons non-linéaires des paramètres structurels du modèle (coefficients invariants, Cf critique de Lucas):

c_1	c_2	c_3
$(\psi b_2\overline{k}^{b_2}\overline{c}^{b_3} + (1-\delta)\overline{k})/(\gamma\overline{k})$	$(\psi b_3\overline{k}^{b_2}\overline{c}^{b_3} - \overline{c})/(\gamma\overline{k})$	$(\psi b_5\overline{k}^{b_2}\overline{c}^{b_3})/(\gamma\overline{k})$
c_4	c_5	c_6
$\frac{\beta}{\gamma}m\psi b_4\overline{k}^{b_4}\overline{c}^{b_3}$	$-\frac{\beta}{\gamma}\left[1-\delta+(1-b_3)m\psi\overline{k}^{b_4}\overline{c}^{b_3}\right]$	$\frac{\beta}{\gamma}m\psi b_5\overline{k}^{b_4}\overline{c}^{b_3}$

Ces coefficients peuvent être définis dans le programme Matlab de la façon suivante:

```
c1=(psi*b2*(k^b2)*(c^b3) + (1-delta)*k)/(phi*k);

c2=(psi*b3*(k^b2)*(c^b3) - c)/(phi*k);

c3=(psi*b5*(k^b2)*(c^b3))/(phi*k);

c4=(beta/phi)*m*psi*b4*(k^b4)*(c^b3);

c5=-(beta/phi)*(1-delta +(1-b3)*m*psi*(k^b4)*(c^b3));

c6 =(beta/phi)*m*psi*b5*(k^b4)*(c^b3);

c7 =-c5;
```

5.6 Ecriture matricielle

$$\begin{bmatrix} c_1 & c_2 & c_3 \\ 0 & -1 & 0 \\ 0 & 0 & \rho \end{bmatrix} \begin{bmatrix} \widehat{k}_t \\ \widehat{c}_t \\ \widehat{s}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ c_4 & c_5 & c_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \widehat{k}_{t+1} \\ \widehat{c}_{t+1} \\ \widehat{s}_{t+1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & c_4 & c_5 & c_6 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{t+1} \\ \widehat{w}^k_{t+1} \\ \widehat{w}^c_{t+1} \\ \widehat{w}^s_{t+1} \end{bmatrix}$$

où $\widehat{w}^x_{t+1} = E_t[x_{t+1}] - x_{t+1}$, pour $x = k, c, s$.

Codes Matlab correspondant:

```
M1=[c1 c2 c3 ; 0 -1 0 ; 0 0 rho];
M2=[1 0 0 ; c4 c5 c6 ; 0 0 1];
M3=[0 0 0 0; 0 c4 c5 c6 ; -1 0 0 0];
```

En pré-multipliant le système par

$$\begin{bmatrix} c_1 & c_2 & c_3 \\ 0 & -1 & 0 \\ 0 & 0 & \rho \end{bmatrix}^{-1}$$

on obtient

$$\begin{bmatrix} \widehat{k}_t \\ \widehat{c}_t \\ \widehat{s}_t \end{bmatrix} = J \begin{bmatrix} \widehat{k}_{t+1} \\ \widehat{c}_{t+1} \\ \widehat{s}_{t+1} \end{bmatrix} + R \begin{bmatrix} v_{t+1} \\ \widehat{w}^k_{t+1} \\ \widehat{w}^c_{t+1} \\ \widehat{w}^s_{t+1} \end{bmatrix}$$

où $dim(J) = 3 \times 3$ et $dim(R) = 3 \times 4$.

Codes Matlab correspondant:

```
J=M1\M2; R=M1\M3;
```

La résolution du système ci-dessus suffit à déterminer la dynamique d'équilibre. En effet, l'approximation log-linéaire de la fonction de production et de la condition d'équilibre sur le marché du travail, donnent $\widehat{y}_t$ et $\widehat{l}_t$ comme des fonction de $\widehat{k}_t$, $\widehat{c}_t$ et $\widehat{s}_t$:

$$\begin{bmatrix} \widehat{y}_t \\ \widehat{l}_t \end{bmatrix} = M \begin{bmatrix} \widehat{k}_t \\ \widehat{c}_t \\ \widehat{s}_t \end{bmatrix} \quad \text{où } dim(M) = 2 \times 3$$

5.7 Unicité de la solution: détermination de la trajectoire selle

1. *Problème*: système d'équations récurrentes non-indépendantes $\Longrightarrow$ pour le résoudre, projection dans une base où les équations sont indépendantes.

2. *Méthode*: détermination des vecteurs propres, formant la matrice de changement de base, et des valeurs propres, donnant l'évolution dynamique dans la nouvelle base.

Etape 1: Détermination des valeurs propres
λ_i, $i \in [1,3]$ val. propres si solutions de:

$$det(J - \lambda I_3) = 0 \quad \Longrightarrow \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

Dans ce système, il existe une équation indépendante: celle du choc technologique s_t. Une des valeurs propres est donc donnée par $1/\rho > 1$, car ce processus est stationnaire.

Comme dans le modèle de croissance optimale, on a $\lambda_1 < 1$ et $\lambda_2 > 1$. L'équilibre est *déterminé*: il y a une unique trajectoire qui converge vers l'état stationnaire, la trajectoire selle.

Etape 2: vecteurs propres et changement de base
Soit la matrice Q, qui vérifie $Q^{-1}JQ = \Lambda$ où $Q = [q^1, q^2, q^3]$ avec $dim(q^i) = 3\times 1$, pour $i = 1,2,3$. On note z_t le vecteur définit par:

$$z_t = Q^{-1} \begin{bmatrix} \widehat{k}_t \\ \widehat{c}_t \\ \widehat{s}_t \end{bmatrix} \quad \text{et} \quad \eta_t = Q^{-1} R \begin{bmatrix} v_t \\ \widehat{w}_t^k \\ \widehat{w}_t^c \\ \widehat{w}_t^s \end{bmatrix}$$

Ce changement de base permet d'obtenir les trois équations indépendantes suivantes:

$$\begin{bmatrix} z_t^1 \\ z_t^2 \\ z_t^3 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} z_{t+1}^1 \\ z_{t+1}^2 \\ z_{t+1}^3 \end{bmatrix} + \begin{bmatrix} \eta_{t+1}^1 \\ \eta_{t+1}^2 \\ \eta_{t+1}^3 \end{bmatrix}$$

Codes Matlab correspondant à ces deux opérations

```
[P,MU]=eig(J);     %matrices des vect propres P et des val propres
PI=inv(P);         %inv. P

[llambda,kk]=sort(diag(MU));  %classement des val propres par ordre crois.

P1=P(:,kk);        %classement des vect propre correspondant
P1I=inv(P1);       %inv de P1
```

Détermination de la trajectoire selle

La première équation a une racine inférieure à l'unité: cette équation converge lorsque l'on itère vers le futur. Ces itérations vers le futur déterminent la restriction entre $\widehat{c}_t$, $\widehat{k}_t$ et $\widehat{s}_t$, définissant la trajectoire selle:

$$z_t^1 = \lambda_1 z_{t+1}^1 + \eta_{t+1}$$

L'anticipation rationnelle de cette équation donne:

$$z_t^1 = \lambda_1 E_t\left[z_{t+1}^1\right] \Longrightarrow z_t^1 = \lambda_1^T E_t\left[z_{t+T}^1\right]$$

Comme $z_t^1 < \infty$ et $\lim_{T\to\infty} \lambda_1^T = 0$, on a:

$$z_t^1 = 0 \quad \Longleftrightarrow \quad q_{1,k}^{-1}\widehat{k}_t + q_{1,c}^{-1}\widehat{c}_t + q_{1,s}^{-1}\widehat{s}_t = 0$$

La trajectoire selle détermine la consommation optimale, $\widehat{c}_t$, à chaque date comme une fonction linéaire de $\widehat{k}_t$ et $\widehat{s}_t$.

Après substitutions, on obtient:

$$\begin{bmatrix} \widehat{k}_{t+1} \\ \widehat{s}_{t+1} \end{bmatrix} = A \begin{bmatrix} \widehat{k}_t \\ \widehat{s}_t \end{bmatrix} + Bv_{t+1} \quad \text{with} \quad A = \begin{bmatrix} a_{11} & a_{12} \\ 0 & \rho \end{bmatrix}$$

Les codes Matlab correspondant à ces opérations sont:

```
A(1,1)=c1+c2*(-P1I(1,1)/P1I(1,2));
A(1,2)=c3+c2*(-P1I(1,3)/P1I(1,2));
A(2,2)=rho;
```

Les dynamiques des autres variables sont obtenues en utilisant:

$$\begin{bmatrix} \widehat{c}_t \\ \widehat{y}_t \\ \widehat{i}_t \\ \widehat{l}_t \\ \widehat{y/l}_t \end{bmatrix} = \Pi \begin{bmatrix} \widehat{k}_t \\ \widehat{s}_t \end{bmatrix}$$

Les codes Matlab correspondant à ces opération sont:

```
%consommation
Pi(1,1)=-P1I(1,1)/P1I(1,2);
Pi(1,2)=-P1I(1,3)/P1I(1,2);

%produit
Pi(2,1)=b2+b3*(-P1I(1,1)/P1I(1,2));
Pi(2,2)=(chi+1)*b1+b3*(-P1I(1,3)/P1I(1,2));
```

```
%investissement
Pi(3,1)=(y/I)*Pi(2,1)-(c/I)*Pi(1,1);
Pi(3,2)=(y/I)*Pi(2,2)-(c/I)*Pi(1,2);

%heures
Pi(4,1)=(mu*b1)-b1*(-P1I(1,1)/P1I(1,2));
Pi(4,2)=b1-b1*(-P1I(1,3)/P1I(1,2));

%productivite
Pi(5,1)=Pi(2,1)-Pi(4,1); Pi(5,2)=Pi(2,2)-Pi(4,2);
```

5.8 Simulation des fonctions de réponses

Il est possible de calculer les fonctions de réponses à un choc technologique. Il s'agit d'évaluer la dynamique d'ajustement des variables macroéconomiques si on perturbe l'état stationnaire par une déviation temporaire de 1% d'un choc technologique (l'exogène du modèle).

Les codes Matlab correspondant à ces opérations sont:

```
nrep=60;       %horizon de simulation
disp('nrep')
disp(nrep)

CHOC=[0 ; 1];  %vecteur des chocs sur les variables d'\'etat
               %le premier \'el\'ement est l'\'ecart du capital
               %par rapport \`a son niveau d'\'etat stationnaire
               %le second \'el\'ement est l'\'ecart de la technologie
               %par rapport \`a son niveau d'\'etat stationnaire:
               %ici, l'\'ecart est de 1%.


%calcul de la trajectoire du capital sur l'horizon de simulation
for j=1:nrep;
 DT=(A^(j-1))*CHOC;
 DTK(j)=DT(1);
end;

%calcul de la trajectoire des autres variables sur l'horizon de simulation
for j=1:nrep;
 DT=(Pi*A^(j-1))*CHOC;
 DTC(j)=DT(1);
 DTY(j)=DT(2);
```

```
 DTI(j)=DT(3);
 DTL(j)=DT(4);
 DTYL(j)=DT(5);
end;

subplot(221),plot(DTY(1:nrep))
title('Produit')
xlabel('Trimestres')
ylabel('% Dev.')

subplot(222),plot(DTC(1:nrep))
title('Consommation')
xlabel('Trimestres')
ylabel('% Dev.')

subplot(223),plot(DTI(1:nrep))
title('Investissement')
xlabel('Trimestres')
ylabel('% Dev.')

subplot(224),plot(DTK(1:nrep))
title('Capital')
xlabel('Trimestres')
ylabel('% Dev.')

print rbc1.eps
pause
clg

subplot(221),plot(DTL(1:nrep))
title('Heures')
xlabel('Trimestres')
ylabel('% Dev.')


subplot(222),plot(DTYL(1:nrep))
title('Productivit\'e')
xlabel('Trimestres')
ylabel('% Dev.')

print rbc2.eps
pause
clg
```

A Graphiques

Figure 1: Exemple 1

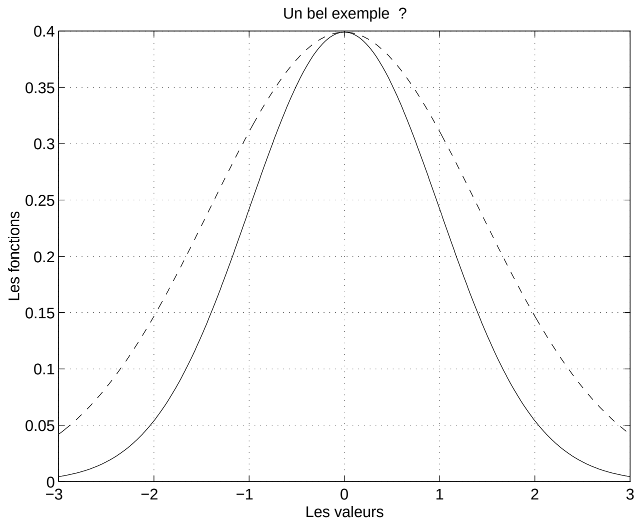


Figure 2: Exemple 2

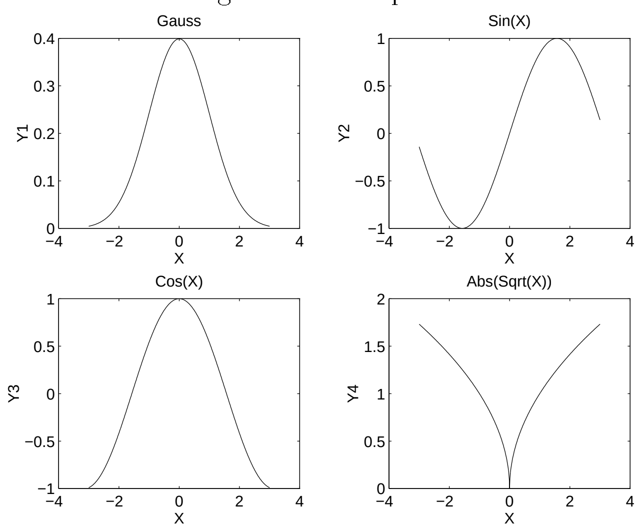


Figure 3: Rbc: fonction de réponse

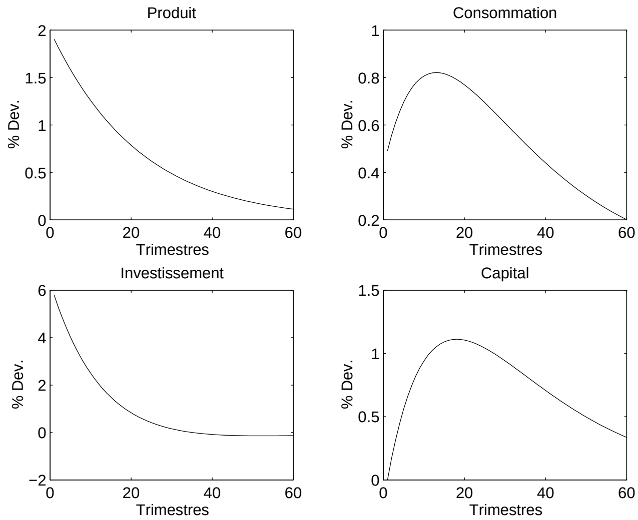
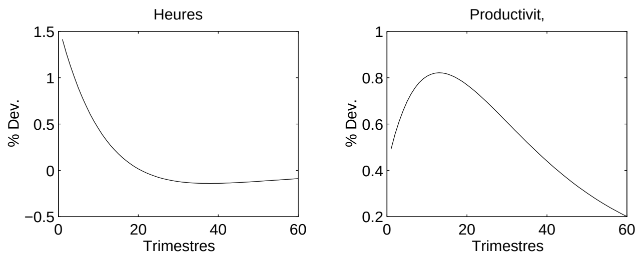


Figure 4: Rbc: fonction de réponse



B Liste des fonction élémentaires

Ces tableaux sont directement issus du Guide d'Utilisation de Matlab (MATLAB Reference Guide).

<table>
<tr><th colspan="2">Opération et caractères spéciaux</th></tr>
<tr><td>+</td><td>plus</td></tr>
<tr><td>-</td><td>moins</td></tr>
<tr><td>*</td><td>multiplication</td></tr>
<tr><td>.*</td><td>multiplication élément par élément</td></tr>
<tr><td>^</td><td>puissance</td></tr>
<tr><td>.^</td><td>puissance élément par élément</td></tr>
<tr><td>kron</td><td>produit de kronecker</td></tr>
<tr><td>\</td><td>division à gauche</td></tr>
<tr><td>/</td><td>division à droite</td></tr>
<tr><td>./</td><td>division élément par élément</td></tr>
<tr><td>:</td><td>colonne</td></tr>
<tr><td>.</td><td>point des décimales</td></tr>
<tr><td>...</td><td>continuité</td></tr>
<tr><td>;</td><td>semi - colonne</td></tr>
<tr><td>%</td><td>commentaire</td></tr>
<tr><td>'</td><td>transposé et guillemets</td></tr>
<tr><td>=</td><td>allocation</td></tr>
<tr><td>==</td><td>égalité</td></tr>
<tr><td>< > <= >=</td><td>opérateurs</td></tr>
<tr><td>&</td><td>AND logique</td></tr>
<tr><td>|</td><td>OR logique</td></tr>
<tr><td>~</td><td>NOT logique</td></tr>
<tr><td>xor</td><td>EXCLUSIVE OR logique</td></tr>
</table>

Elementary Math Functions	
`abs`	Absolute value
`acos`	Inverse cosine
`acosh`	Inverse hyperbolic cosine
`angle`	Phase angle
`asin`	Inverse sine
`asinh`	Inverse hyperbolic sine
`atan`	Inverse tangent
`atan2`	Four quadrant inverse tangent
`atanh`	Inverse hyperbolic tangent
`ceil`	Round towards plus infinity
`conj`	Complex conjugate
`cos`	Cosine
`cosh`	Hyperbolic cosine
`exp`	Exponential
`fix`	Round towards zero
`floor`	Round towards minus infinity
`imag`	Complex imaginary part
`log`	Natural logarithm
`log10`	Common logarithm
`real`	Complex real part
`rem`	Remainder after division
`round`	Round toward nearest integer
`sign`	Signum function
`sin`	Sine
`sinh`	Hyperbolic sine
`sqrt`	Square root
`tan`	Tangent
`tanh`	Hyperbolic tangent

C la commande print

Syntaxe: `print [ -ddevice] [ -options ] filename`

Available Windows Device Options	
`-dwin`	Send figure to currently installed printer in monochrome
`-dwinc`	Send figure to currently installed printer in color
`-dmeta`	Send figure to clipboard in Metafile format
`-dbitmap`	Send figure to clipboard in bitmap format
Available Postscript Devices	
`-dps`	PostScript for black and white printers
`-dpsc`	PostScript for color printers
`-dps2`	Level 2 PostScript for black and white printers
`-dpsc2`	Level 2 PostScript for color printers
`-deps`	Encapsulated PostScript (EPSF)
`-depsc`	Encapsulated Color PostScript (EPSF)
`-deps2`	Encapsulated Level 2 PostScript (EPSF)
`-depsc2`	Encapsulated Level 2 Color PostScript (EPSF)
`-dlaserjet`	HP LaserJet
`-dljetplus`	HP LaserJet+
`-dljet2p`	HP LaserJet IIP
`-dljet3`	HP LaserJet III
`-dcdeskjet`	HP DeskJet 500C with 1 bit/pixel color
`-dcdjcolor`	HP DeskJet 500C with 24 bit/pixel color
`-dcdjmono`	HP DeskJet 500C printing black only
`-ddeskjet`	HP DeskJet and DeskJet Plus
`-dpaintjet`	HP PaintJet color printer
`-dpjetxl`	HP PaintJet XL color printer
`-dbj10e`	Canon BubbleJet BJ10e
`-dln03`	DEC LN03 printer
`-dgif8`	8-bit color GIF file format
`-dpcx16`	Older color PCX file format (EGA/VGA, 16-color)
`-dpcx256`	Newer color PCX file format (256-color)
Other Options	
`-append`	Append the graph to file, rather than overwriting
`-Pprinter`	Specify the printer to use
`-fhandle`	Handle Graphics handle of figure to print

D Programme de simulation

```
clear
delete rbcs.res
diary rbcs.res

ttt=clock;

% initialisation du nombre de simulations

nsimul=100;
disp('nsimul')
disp(nsimul)

% initialisation de la longueur des series
nlong=4*40;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Parametres
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

beta=.99;
delta=.025;
chi=0;
gamma=1.004;
rho=.95;
sdepsilona=.0072;

mu=.36;
nu=.64;
m=.36;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phi=gamma^(nu/(1-mu));

b1=1/(chi- nu +1);
b2=mu*(chi+1)*b1;
b3=-nu*b1;
b4=b2-1;
b5=(chi+1)*b1;
psi=(1-m)^(-b3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Etat stationnaire
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AA=(phi+delta-1)-(1/(beta*m))*(phi-beta*(1-delta));
BB=((phi/beta)-1+delta)*(1/(m*psi));

k=(-(BB^(1/b3))/AA)^(1/(1+(b4/b3)));

y=((phi/beta)-1+delta)*(k/m);

l=(y/(k^mu))^(1/nu);

c=y-(delta+phi-1)*k;

I=y-c;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Dynamique
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c1= (psi*b2*(k^b2)*(c^b3) + (1-delta)*k)/(phi*k);
c2= (psi*b3*(k^b2)*(c^b3) - c)/(phi*k);
c3= (psi*b5*(k^b2)*(c^b3))/(phi*k);
c4= (beta/phi)*m*psi*b4*(k^b4)*(c^b3);
c5= -(beta/phi)*(1-delta + (1-b3)*m*psi*(k^b4)*(c^b3));
c6 = (beta/phi)*m*psi*b5*(k^b4)*(c^b3);

M1=[c1 c2 c3 ; 0 -1 0 ; 0 0 rho];
M2=[1 0 0 ; c4 c5 c6 ; 0 0 1];
M3=[0 ; 0 ; 1];

J=M1\M2;
R=M1\M3;

[P,MU]=eig(J);
PI=inv(P);

MU
%pause

[llambda,kk]=sort(diag(MU));

P1=P(:,kk);
```

```
P1I=inv(P1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Systeme resolu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Dynamique de k et s

A(1,1)=c1+c2*(-P1I(1,1)/P1I(1,2));
A(1,2)=c3+c2*(-P1I(1,3)/P1I(1,2));
A(2,2)=rho;

%c,y,i,l,y/l en fonction de k et s

%conso
Pi(1,1)=-P1I(1,1)/P1I(1,2);
Pi(1,2)=-P1I(1,3)/P1I(1,2);

%produit
Pi(2,1)=b2+b3*(-P1I(1,1)/P1I(1,2));
Pi(2,2)=(chi+1)*b1+b3*(-P1I(1,3)/P1I(1,2));

%investissement
Pi(3,1)=(y/I)*Pi(2,1)-(c/I)*Pi(1,1);
Pi(3,2)=(y/I)*Pi(2,2)-(c/I)*Pi(1,2);

%heures
Pi(4,1)=(mu*b1)-b1*(-P1I(1,1)/P1I(1,2));
Pi(4,2)=b1-b1*(-P1I(1,3)/P1I(1,2));

%productivite
Pi(5,1)=Pi(2,1)-Pi(4,1);
Pi(5,2)=Pi(2,2)-Pi(4,2);

A
Pi



%************************************************************
%
% generation des parties cycliques HP filtrees
%
```

```
%**********************************************************

% valeurs initiales de SS calculees

HT(1)=l;
KT(1)=k;
YT(1)=y;
CT(1)=c;
IT(1)=I;
PMT(1)=YT(1)/HT(1);

% partie tendantielle

for i=2:nlong;

KT(i)=(gamma)*KT(i-1);
CT(i)=(gamma)*CT(i-1);
HT(i)=HT(i-1);
YT(i)=(gamma)*YT(i-1);
IT(i)=(gamma)*IT(i-1);
PMT(i)=(gamma)*PMT(i-1);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Simulations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for j=1:nsimul;

disp('simulation')
disp(j)

% simulation des ji\`emes parties cycliques

rand('normal');


aleaa(1:nlong,j)=rand(nlong,1);

% tirage des innovations

  for i=1:nlong;
    epsa(j,i)= aleaa(i,j) * (sdepsilona);
```

```
  end;


% construction des chocs CHT

CHT(j,1)=0;

  for i=2:nlong;
    CHT(j,i)=rho*CHT(j,i-1) + epsa(j,i);
  end;


% initialisation de la partie cyclique du capital

KC(j,1)=0;

% parties cycliques

  for i=1:nlong;
    KC(j,i+1)=A(1,1)*KC(j,i)+A(1,2)*CHT(j,i);
  end;

  for i=1:nlong;
     CC(j,i)=Pi(1,1)*KC(j,i)+Pi(1,2)*CHT(j,i);
     YC(j,i)=Pi(2,1)*KC(j,i)+Pi(2,2)*CHT(j,i);
     IC(j,i)=Pi(3,1)*KC(j,i)+Pi(3,2)*CHT(j,i);
     HC(j,i)=Pi(4,1)*KC(j,i)+Pi(4,2)*CHT(j,i);
     PMC(j,i)=Pi(5,1)*KC(j,i)+Pi(5,2)*CHT(j,i);
  end;


% construction des series brutes

  for i=1:nlong;
    KB(j,i)=log(KT(i)*(1+KC(j,i)));
    CB(j,i)=log(CT(i)*(1+CC(j,i)));
    HB(j,i)=log(HT(i)*(1+HC(j,i)));
    YB(j,i)=log(YT(i)*(1+YC(j,i)));
    IB(j,i)=log(IT(i)*(1+IC(j,i)));
    PMB(j,i)=log(PMT(i)*(1+PMC(j,i)));
  end;


end;
```

```
  for j=1:nsimul;

disp('filtre')
disp(j)

   KTHP(j,1:nlong)=hpfilter(KB(j,1:nlong),1,nlong);
   CTHP(j,1:nlong)=hpfilter(CB(j,1:nlong),1,nlong);
   HTHP(j,1:nlong)=hpfilter(HB(j,1:nlong),1,nlong);
   YTHP(j,1:nlong)=hpfilter(YB(j,1:nlong),1,nlong);
   ITHP(j,1:nlong)=hpfilter(IB(j,1:nlong),1,nlong);
   PMTHP(j,1:nlong)=hpfilter(PMB(j,1:nlong),1,nlong);


% calcul des parties cycliques par filtrage HP


  KCHP(j,1:nlong)=KB(j,1:nlong)-KTHP(j,1:nlong);
  CCHP(j,1:nlong)=CB(j,1:nlong)-CTHP(j,1:nlong);
  HCHP(j,1:nlong)=HB(j,1:nlong)-HTHP(j,1:nlong);
  YCHP(j,1:nlong)=YB(j,1:nlong)-YTHP(j,1:nlong);
  ICHP(j,1:nlong)=IB(j,1:nlong)-ITHP(j,1:nlong);
  PMCHP(j,1:nlong)=PMB(j,1:nlong)-PMTHP(j,1:nlong);

  ETYCHP(j)=std(YCHP(j,1:nlong));
  disp(ETYCHP(j))

  end;

% calcul des moments entre parties cycliques


for j=1:nsimul;

disp('moments')
disp(j)


% ecart-types


ETKCHP(j)=std(KCHP(j,80:nlong));
ETCCHP(j)=std(CCHP(j,80:nlong));
ETHCHP(j)=std(HCHP(j,80:nlong));
```

```
ETYCHP(j)=std(YCHP(j,80:nlong));
ETICHP(j)=std(ICHP(j,80:nlong));
ETPMCHP(j)=std(PMCHP(j,80:nlong));


% ecart-types relatifs


ETRKCHP(j)=ETKCHP(j)/ETYCHP(j);
ETRCCHP(j)=ETCCHP(j)/ETYCHP(j);
ETRHCHP(j)=ETHCHP(j)/ETYCHP(j);
ETRYCHP(j)=ETYCHP(j)/ETYCHP(j);
ETRICHP(j)=ETICHP(j)/ETYCHP(j);
ETRPMCHP(j)=ETPMCHP(j)/ETYCHP(j);

ETRHPMCHP(j)=ETHCHP(j)/ETPMCHP(j);
LAMBDA(j)=((ETYCHP(j))^2)/(0.0176^2);


% generation des series retardees

for k=81:nlong;

  KCHP1(j,k)=KCHP(j,k-1);
  CCHP1(j,k)=CCHP(j,k-1);
  HCHP1(j,k)=HCHP(j,k-1);
  YCHP1(j,k)=YCHP(j,k-1);
  ICHP1(j,k)=ICHP(j,k-1);
  PMCHP1(j,k)=PMCHP(j,k-1);

end;


% autocorrelations a l'ordre 1


VCV=cov(KCHP1(j,80:nlong-1),KCHP(j,80:nlong-1));
AC1KCHP(j)=(VCV(2,1))/(ETKCHP(j)^(2));

VCV=cov(CCHP1(j,80:nlong-1),CCHP(j,80:nlong-1));
AC1CCHP(j)=(VCV(2,1))/(ETCCHP(j)^(2));

VCV=cov(HCHP1(j,80:nlong-1),HCHP(j,80:nlong-1));
AC1HCHP(j)=(VCV(2,1))/(ETHCHP(j)^(2));
```

```
VCV=cov(YCHP1(j,80:nlong-1),YCHP(j,80:nlong-1));
AC1YCHP(j)=(VCV(2,1))/(ETYCHP(j)^(2));

VCV=cov(ICHP1(j,80:nlong-1),ICHP(j,80:nlong-1));
AC1ICHP(j)=(VCV(2,1))/(ETICHP(j)^(2));

VCV=cov(PMCHP1(j,80:nlong-1),PMCHP(j,80:nlong-1));
AC1PMCHP(j)=(VCV(2,1))/(ETPMCHP(j)^(2));


% correlations instantanees avec le produit


VCV=cov(KCHP(j,80:nlong),YCHP(j,80:nlong));
CIKCHP(j)=(VCV(2,1))/(ETKCHP(j)*ETYCHP(j));

VCV=cov(CCHP(j,80:nlong),YCHP(j,80:nlong));
CICCHP(j)=(VCV(2,1))/(ETCCHP(j)*ETYCHP(j));

VCV=cov(HCHP(j,80:nlong),YCHP(j,80:nlong));
CIHCHP(j)=(VCV(2,1))/(ETHCHP(j)*ETYCHP(j));

VCV=cov(YCHP(j,80:nlong),YCHP(j,80:nlong));
CIYCHP(j)=(VCV(2,1))/(ETYCHP(j)*ETYCHP(j));

VCV=cov(ICHP(j,80:nlong),YCHP(j,80:nlong));
CIICHP(j)=(VCV(2,1))/(ETICHP(j)*ETYCHP(j));

VCV=cov(PMCHP(j,80:nlong),YCHP(j,80:nlong));
CIPMCHP(j)=(VCV(2,1))/(ETPMCHP(j)*ETYCHP(j));


% correlation emploi-productivite (moyenne)

VCV=cov(PMCHP(j,80:nlong),HCHP(j,80:nlong));
CORHPMHP(j)=(VCV(2,1))/(ETPMCHP(j)*ETHCHP(j));


end;


% Calcul des moments moyens sur l'ensemble des simulations
% et de leur ecart-type
```

```
mETKCHP=mean(ETKCHP);
mETCCHP=mean(ETCCHP);
mETHCHP=mean(ETHCHP);
mETYCHP=mean(ETYCHP);
mETICHP=mean(ETICHP);
mETPMCHP=mean(ETPMCHP);

etETKCHP=std(ETKCHP);
etETCCHP=std(ETCCHP);
etETHCHP=std(ETHCHP);
etETYCHP=std(ETYCHP);
etETICHP=std(ETICHP);
etETPMCHP=std(ETPMCHP);


mETRKCHP=mean(ETRKCHP);
mETRCCHP=mean(ETRCCHP);
mETRHCHP=mean(ETRHCHP);
mETRYCHP=mean(ETRYCHP);
mETRICHP=mean(ETRICHP);
mETRPMCHP=mean(ETRPMCHP);
mETRHPMCHP=mean(ETRHPMCHP);
mLAMBDA=mean(LAMBDA);

etETRKCHP=std(ETRKCHP);
etETRCCHP=std(ETRCCHP);
etETRHCHP=std(ETRHCHP);
etETRYCHP=std(ETRYCHP);
etETRICHP=std(ETRICHP);
etETRPMCHP=std(ETRPMCHP);
etETRHPMCHP=std(ETRHPMCHP);
etLAMBDA=std(LAMBDA);


mAC1KCHP=mean(AC1KCHP);
mAC1CCHP=mean(AC1CCHP);
mAC1HCHP=mean(AC1HCHP);
mAC1YCHP=mean(AC1YCHP);
mAC1ICHP=mean(AC1ICHP);
mAC1PMCHP=mean(AC1PMCHP);
```

```
etAC1KCHP=std(AC1KCHP);
etAC1CCHP=std(AC1CCHP);
etAC1HCHP=std(AC1HCHP);
etAC1YCHP=std(AC1YCHP);
etAC1ICHP=std(AC1ICHP);
etAC1PMCHP=std(AC1PMCHP);


mCIKCHP=mean(CIKCHP);
mCICCHP=mean(CICCHP);
mCIHCHP=mean(CIHCHP);
mCIYCHP=mean(CIYCHP);
mCIICHP=mean(CIICHP);
mCIPMCHP=mean(CIPMCHP);

etCIKCHP=std(CIKCHP);
etCICCHP=std(CICCHP);
etCIHCHP=std(CIHCHP);
etCIYCHP=std(CIYCHP);
etCIICHP=std(CIICHP);
etCIPMCHP=std(CIPMCHP);



mCORHPMHP=mean(CORHPMHP);
etCORHPMHP=std(CORHPMHP);



mET1=[mETCCHP mETHCHP mETYCHP mETICHP mETPMCHP ];

etET1=[etETCCHP etETHCHP etETYCHP etETICHP etETPMCHP ];

mETR1=[mETRCCHP mETRHCHP mETRYCHP mETRICHP mETRPMCHP ];
etETR1=[etETRCCHP etETRHCHP etETRYCHP etETRICHP  etETRPMCHP ];

mAC11=[mAC1CCHP mAC1HCHP mAC1YCHP mAC1ICHP mAC1PMCHP ];
etAC11=[etAC1CCHP etAC1HCHP etAC1YCHP etAC1ICHP etAC1PMCHP ];


mCI1=[mCICCHP mCIHCHP mCIYCHP mCIICHP mCIPMCHP ];
etCI1=[etCICCHP etCIHCHP etCIYCHP etCIICHP etCIPMCHP ];
```

```
disp('variance modele / variance observee')
disp(' ')
disp(mLAMBDA)
disp(etLAMBDA)


disp('moments des parties cycliques HP')
disp(' ')

disp('ordre: C - H - Y - I - PM')
disp(' ')

disp('ecart-types')
disp(' ')
disp(mET1)
disp(etET1)


disp('ecart-types relatifs')
disp(' ')
disp(mETR1)
disp(etETR1)

disp('autocorrelations a l ordre 1')
disp(' ')
disp(mAC11)
disp(etAC11)


disp('correlations instantanees avec Y')
disp(' ')
disp(mCI1)
disp(etCI1)


disp('correlation instantannee productivite-emploi')
disp(' ')
disp(mCORHPMHP)
disp(etCORHPMHP)


disp('ecart type relatif emploi-productivite')
disp(' ')
disp(mETRHPMCHP)
```

```
disp(etETRHPMCHP)

dur=etime(clock,ttt);
disp('temps de calcul en secondes:')
disp(dur)
diary off
```