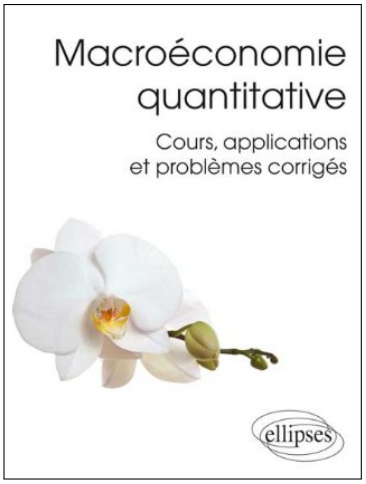


Application Chapitre 8 (corrigé)

Techniques d'estimation

Dans ce document, je propose une correction de l'application du Chapitre 8 de mon manuel de Macroéconomie quantitative (p.112). Les fonctions et les données utiles sont disponibles sur mon site [a]. Un ou plusieurs scripts Octave/Matlab accompagnent également ce document, dont une partie des éléments qui sont repris et commentés ici, apparaissent sous une police à `chasse fixe`.

a. www.christophecahn.fr/macroquant

Estimation d'un modèle néo-keynésien

Les séries contenues dans le fichier `data.txt` ont été engendrées par une économie artificielle. Les séries correspondent à l'écart de production (différence entre la production et son niveau naturel), l'inflation et le taux d'intérêt nominal, présentés en colonne. Nous allons tenter dans ce problème de déterminer le processus générateur des données (PGD), c.-à-d. le modèle qui a engendré ces données.

Estimation par maximum de vraisemblance

On envisage d'estimer le modèle suivant

$$\pi_t = \beta \mathsf{E}_t\{\pi_{t+1}\} + \kappa \tilde{y}_t + \nu_{p,t}$$

$$\tilde{y}_t = \mathsf{E}_t\{\tilde{y}_{t+1}\} - (i_t - \pi_{t+1} - \nu_{n,t})$$

$$i_t = \phi_\pi \pi_t + \phi_y \tilde{y}_t + \nu_{i,t}$$

où les chocs $\nu_{x,t}$, $x \in \{p, n, i\}$ sont des processus AR(1)

$$\nu_{x,t} = \rho_x \nu_{x,t-1} + \epsilon_{x,t},\ \epsilon_{x,t} \sim \text{i.i.d. } N(0, \sigma_x^2)$$

1a Écrire le fichier `dynare` correspondant au modèle de l'énoncé. On utilisera les valeurs suivantes : $\beta = 0.98$, $\kappa = 0.01$, $\phi_\pi = 1.2$, $\phi_y = 0.1$, $\rho_x = 0.8$ et $\sigma_x = 0.01$ pour $x \in \{p, n, i\}$.

```
// Application 8 : question 1b

var y p i nu_i nu_p nu_n ;
varexo en ei ep ;

parameters k b ap ay rho_p rho_i rho_n ;
b = .98 ;
k = .01 ;
ap = 1.2 ;
ay = .1 ;
```

```
rho_n = 0.8 ;
rho_i = 0.8 ;
rho_p = 0.8 ;

model;

    b*p(+1) + k*y + nu_p - p ;
    y = y(+1) - (i - p(+1) - nu_n) ;
    i = ap*p + ay*y + nu_i ;

    nu_n = rho_n*nu_n(-1) + en ;
    nu_i = rho_i*nu_i(-1) + ei ;
    nu_p = rho_p*nu_p(-1) + ep ;
end;

shocks;
var en; stderr .01 ;
var ei; stderr .01 ;
var ep; stderr .01 ;
end;

stoch_simul(irf=80,graph_format=pdf);
```

1b Simuler le modèle et commenter les IRF.

```
stoch_simul(irf=80);
```

1c Estimer par maximum de vraisemblance les paramètres du modèle à l'exception de β. On trouvera les valeurs estimées dans la structure `M_.params`, dans l'ordre de la déclaration des paramètres dans le fichier `mod`. Les variances des chocs sont dans la structure `M_.Sigma_e`.

```
estimated_params;
k ,    .01;
ap ,  1.2 ;
ay ,   .1 ;
rho_n, .8 ;
rho_i, .8 ;
rho_p, .8 ;
stderr en, .01 ;
stderr ei, .01 ;
stderr ep, .01 ;
end ;

varobs y p i ;

estimation(datafile='data',graph_format=pdf);

RESULTS FROM MAXIMUM LIKELIHOOD ESTIMATION
parameters
    Estimate    s.d. t-stat
```

Figure 1 – Réponses dynamiques à un choc de politique monétaire

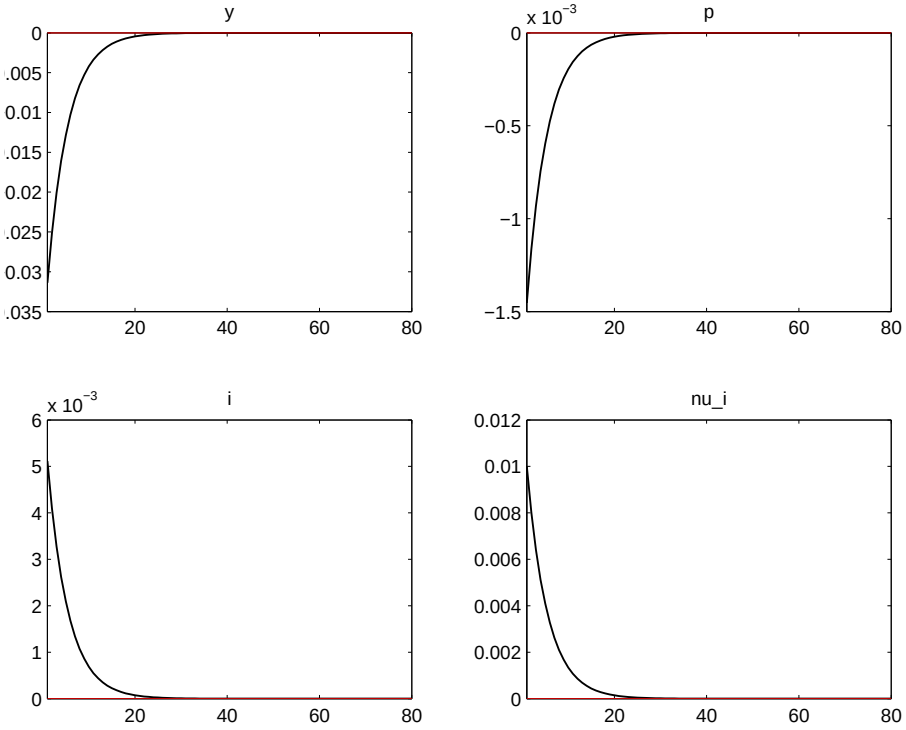


Figure 2 – Réponses dynamiques à un choc de préférence

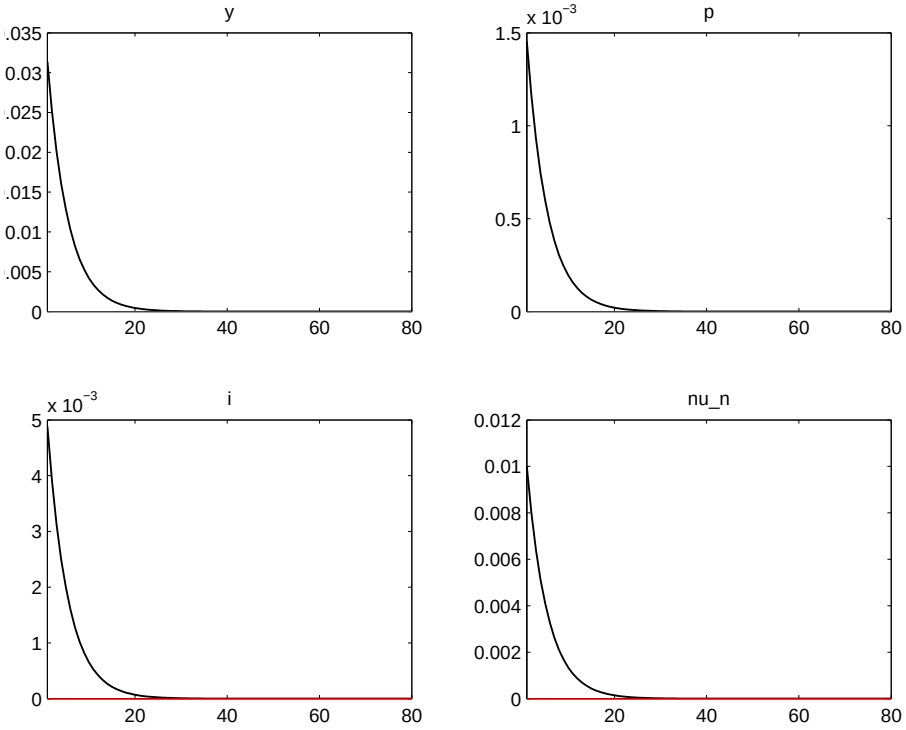
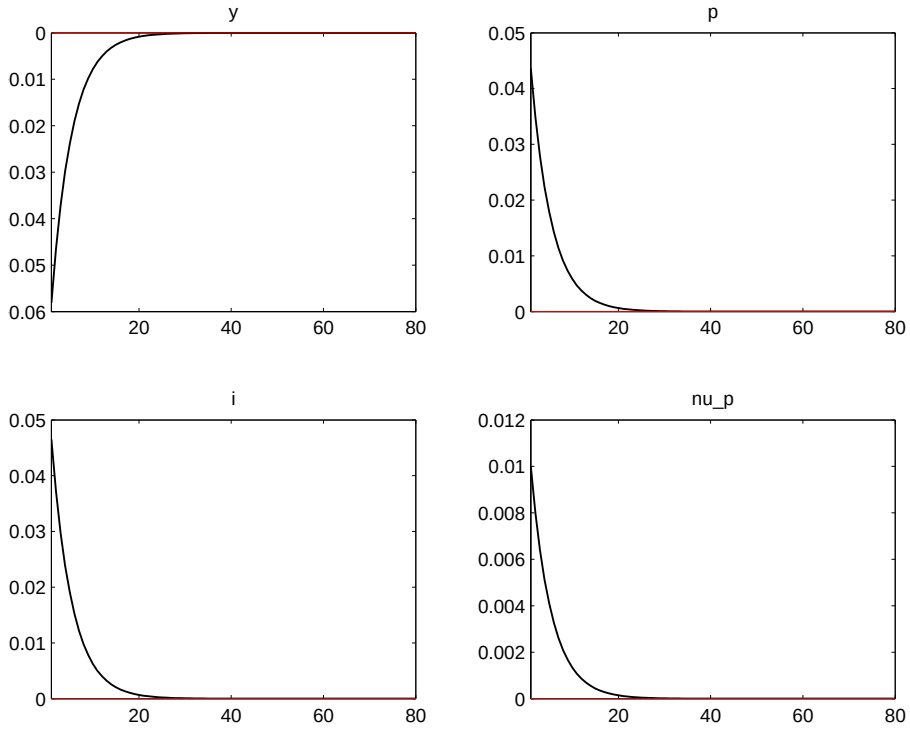


Figure 3 – Réponses dynamiques à un cost-push shock

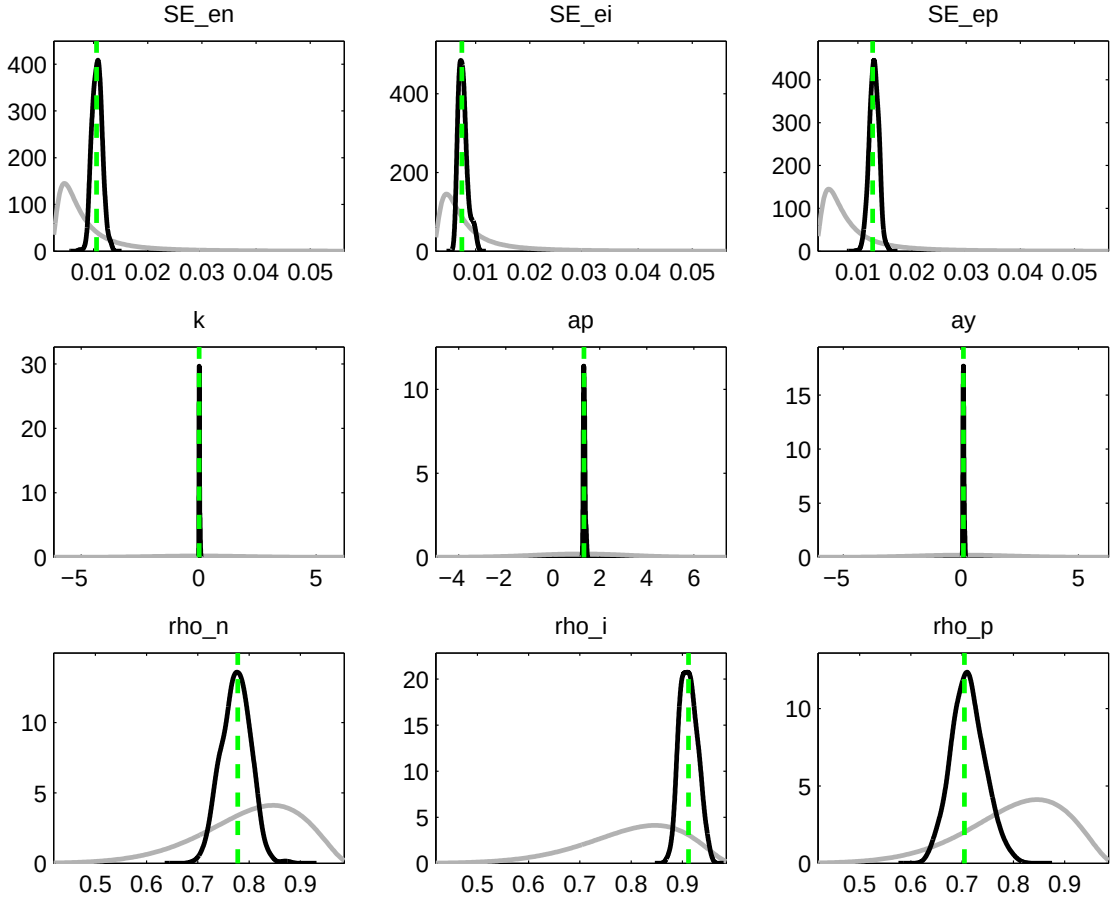


```
k    0.0204  0.0092  2.2042
ap   1.3351  0.0320 41.6907
ay   0.1136  0.0208  5.4579
rho_n   0.7678  0.0281 27.3117
rho_i   0.9152  0.0149 61.4091
rho_p   0.6934  0.0280 24.8041

standard deviation of shocks
    Estimate    s.d. t-stat

en   0.0108  0.0010 10.9669
ei   0.0076  0.0008  9.9346
ep   0.0131  0.0010 13.0694
```

1d Estimer le modèle par méthode bayésienne en utilisant dynare.

```
estimated_params;
k , normal_pdf, 0.01, 0.02;
ap , normal_pdf, 1.2, 2;
ay , normal_pdf, .1, .2;
rho_n, beta_pdf, .8, .1 ;
rho_i, beta_pdf, .8, .1 ;
rho_p, beta_pdf, .8, .1 ;
stderr en, inv_gamma_pdf, 0.01, inf;
stderr ei, inv_gamma_pdf, 0.01, inf;
stderr ep, inv_gamma_pdf, 0.01, inf;
end ;

varobs y i p ;

estimation(order=1,mh_replic=2000,mh_nblocks=2,
  mode_compute=4,datafile='data',mh_jscale = .75,graph_format=pdf);

ESTIMATION RESULTS

Log data density is 3661.043255.

parameters
     prior mean   post. mean        90% HPD interval    prior       pstdev

k         0.010       0.0268      0.0087      0.0460   norm        2.0000
ap        1.200       1.3319      1.2862      1.3873   norm        2.0000
ay        0.100       0.1102      0.0796      0.1475   norm        2.0000
rho_n        0.800       0.7760      0.7231      0.8135   beta        0.1000
rho_i        0.800       0.9098      0.8896      0.9376   beta        0.1000
rho_p        0.800       0.7077      0.6626      0.7570   beta        0.1000

standard deviation of shocks
     prior mean   post. mean        90% HPD interval    prior       pstdev

en        0.010       0.0106      0.0090      0.0121   invg           Inf
```

Figure 4 – Densités a priori et a posteriori.



ei	0.010	0.0076	0.0063	0.0087	invg	Inf
ep	0.010	0.0130	0.0116	0.0147	invg	Inf

Estimation par la méthode des moments simulés

2a Représenter le modèle sous forme matricielle.

Le modèle peut se mettre sous la forme

$$\mathrm{E}_t\{Fx_{t+1} + Gx_t + Hx_{t-1} + Lz_{t+1} + Mz_t\} = 0,\ z_{t+1} = Nz_t + u_{t+1}$$

avec $x_t = (y_t, \pi_t, i_t)'$, $z_t = (\nu_{p,t}, \nu_{n,t}, \nu_{i,t})'$, $u_t = (\epsilon_{p,t}, \epsilon_{n,t}, \epsilon_{i,t})'$ et

$$F = \begin{pmatrix} 0 & \beta & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, G = \begin{pmatrix} \kappa & -1 & 0 \\ -1 & 0 & -1 \\ \phi_y & \phi_\pi & -1 \end{pmatrix}, H = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, L = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, N = \begin{pmatrix} \rho_p & 0 & 0 \\ 0 & \rho_n & 0 \\ 0 & 0 & \rho_i \end{pmatrix}.$$

2b Proposer un algorithme permettant d'estimer par la méthode de moments simulés les paramètres du modèle. Proposer un ensemble de moments pour l'estimation.

1. engendrer une série de vecteurs de chocs $\epsilon \equiv \{(\epsilon_{p,t}, \epsilon_{n,t}, \epsilon_{i,t})'\}_{t=1}^{T}$
2. pour un jeu de paramètres $\Theta = (\kappa, \phi_y, \phi_\pi, \rho_p, \rho_n, \rho_i, \sigma_p, \sigma_n, \sigma_i)$, simuler des séries $\{x_t(\Theta) = (y_t, \pi_t, i_t)\}_{t=1}^{T}$ et calculer des moments $\mu(\Theta, \epsilon)$
3. Trouver Θ tel que $\mu(\Theta, \epsilon)$ soit très proche de μ^o, les moments calculés sur les observations, par exemple
$$\Theta^* = \arg\max_{\Theta} (\mu(\Theta, \epsilon) - \mu^o)^2.$$

On peut prendre par exemple comme moments les éléments de la matrice de variance covariance de x_t.

2c Estimer le modèle par la méthode des moments simulés.

On commence donc par calculer et stocker les moments des variables observées

```
load data ;
moments = mom([y p i]);
```

Ensuite, on crée une fonction objectif `funobj(x,ee,moments,T,burn,nsim)` dont l'objet est de donner une mesure de la distance entre les moments calculés sur données observées `moments` et des moments calculés sur des données simulées à partir d'un modèle dont les paramétres sont donnés par `x`. Ces données simulées sont engendrées à partir d'une série de chocs gaussiens iid stockée dans la matrice `ee`. La taille de l'échantillon initial est `T`, mais nous allons simuler le modèle `burn` périodes avant afin d'atteindre par apporximation la distribution ergodique. Par ailleurs, les moments obtenus à partir de la simulation du modèle dépendent du tirage de chocs gaussiens. Afin de limiter cette dépendance, nous allons itérer `nsim` fois la procédure puis calculer la moyenne sur ces `nsim` simulations. Enfin, nous prendrons la distance euclidenne pour caractériser l'écart entre les moments observés et les moments simulés.

Un telle fonction peut s'écrire ainsi

```
function [res] = funobj(x,ee,moments,T,burn,nsim)

moms = [];

for ii=1:nsim
   [y_sim, p_sim, i_sim] = modsim(x,ee(:,1 + (nsim-1)*(burn+T):nsim*(burn+T)));
y_sim = y_sim(burn+1:burn+T,:);
p_sim = p_sim(burn+1:burn+T,:);
i_sim = i_sim(burn+1:burn+T,:);
moms = [moms vec(mom([y_sim,p_sim,i_sim]))];
end
w = mean(moms,2) - moments(:) ;
res = w'*w;
```

où `modsim` est une fonction qui simule le modèle à partir de sa forme matricielle trouvée plus haut.

La suite de la procédure consiste à minimiser cette fonction objectif. Pour ce faire, on peut utiliser la commande `fminsearch`. Il se peut néanmoins qu'au cours du processus de minimisation, l'algorithme s'en aille visiter des régions dans lesquelles certains paramètres prennent des valeurs interdites par le modèle économique. Par exemple, il se peut que la valeur attribuée à un des paramètres de persistance soit négative ou plus grande que 1, ou la valeur attribuée à ϕ_π soit telle

que le principe de Taylor vu au Chapitre 7 ne soit plus respecté. Dans ce cas, on peut être amené à opérer une transformation sur les paramétres afin de modifier leur domaine de définition. Dans le cas présent, la fonction `invtrans` permet d'opérer cette transformation :

```
function res = invtrans(x)

res = zeros(size(x));
res(1) = exp(x(1));
res(2) = exp(x(2));
res(3) = 1+exp(x(3));
res(4) = .5+.5*tanh(x(4));
res(5) = .5+.5*tanh(x(5));
res(6) = .5+.5*tanh(x(6));
res(7) = exp(x(7));
res(8) = exp(x(8));
res(9) = exp(x(9));

end
```

Dans ce cas, en posant `x=invtrans(xx)`, on voit que si la première composante de `xx` peut prendre toutes les valeurs de la droite réelle, la première composante de x sera nécessairement strictement positive (ce qui doit être le cas pour le paramètre κ). De la même manière, le paramètre ρ_p qui correspond à la quatrième composante de `x` est bien compris entre 0 et 1 alors que la quatrième composante de `xx` balaye à nouveau la droite réelle.

En définitive, la procédure de minimisation peut s'écrire simplement :

```
xsol = fminsearch(@(xx) funobj(invtrans(xx),ee,moments,T,burn,nsim),xinit)
```

En appliquant cette procédure, on trouve

```
invtrans(xsol) =

    0.0315   0.0241   1.1351   0.7337   0.8123   0.7389   0.0118   0.0107   0.0071
```

en rappelant l'ordre des paramètres

```
 1:k  2:ay  3:ap  4:rp  5:rn  6:ri  7:sp  8:sn  9:si
```

Estimation par inférence indirecte

On se propose cette fois d'estimer les paramètres du modèle par inférence indirecte. À cet effet, on utilisera un modèle VAR en guise de modèle auxiliaire.

3a Estimer un modèle VAR(4) sur les données.

On peut utiliser la fonction de Sims vue au chapitre 4. On peut également utiliser la fonction `estimVar` qui calcule directement les matrices de coefficients et la matrice de variance covariance

```
function [B,S] = estimVAR(Yobs,nlag)

T = size(Yobs,1);
Y = Yobs(nlag+1:T,:)';
```

```
Z = [];
for ii=1:nlag
  Z = [Z;Yobs(nlag+1-ii:T-ii,:)'];
end
B = Y*Z'/(Z*Z');
U = Y - B*Z ;
S = 1/(T-nlag*size(Yobs,2)-1)*U*U';
```

L'estimation du VAR(4) est obtenue par la suite de commandes suivante :

```
[B,S] = estimVAR(data,4)
```

3b Proposer un algorithme permettant d'estimer les paramètres par inférence indirecte, en minimisant la distance (à définir) entre les paramètres du VAR(4) estimé sur les données simulées et observées.

L'algorithme est similaire à celui utilisé dans le cadre de la méthode des moments simulés :

1. engendrer une série de vecteurs de chocs $\epsilon \equiv \{(\epsilon_{p,t}, \epsilon_{n,t}, \epsilon_{i,t})'\}_{t=1}^{T}$
2. pour un jeu de paramètres $\Theta = (\kappa, \phi_y, \phi_\pi, \rho_p, \rho_n, \rho_i, \sigma_p, \sigma_n, \sigma_i)$, simuler des séries $\{x_t(\Theta) = (y_t, \pi_t, i_t)\}_{t=1}^{T}$ et estimer un VAR(4) dont on empilera les coefficients (matrices de retard $B = [B_1 B_2 B_3 B_4]$ et matrice de variance-covariance des innovations S) $\Xi \equiv (\text{vec}(B)', \text{vech}(S)')'$ où $\text{vec}(\cdot)$ et $\text{vech}(\cdot)$ sont des opérateurs : le premier empile verticalement les vecteurs colonnes d'une matrice, le second empile verticalement les élements sur ou sous la première diagonale.
3. Trouver Θ tel que $\Xi(\Theta, \epsilon)$ soit très proche de Ξ^o, les éléments du VAR(4) calculés sur les observations, par exemple
$$\Theta^* = \arg\max_{\Theta} (\Xi(\Theta, \epsilon) - \Xi^o)^2.$$

3c Estimer le modèle par inférence indirecte.

Il suffit d'amender la fonction objectif précédente de la manière suivante

Un telle fonction peut s'écrire ainsi

```
function res = funobjVAR(x,ee,mod_aux_target,nlag,T,burn,nsim,ponderation)

mod_aux_sim = [];

for ii=1:nsim
   [y_sim, p_sim, i_sim] = modsim(x,ee(:,1 + (nsim-1)*(burn+T):nsim*(burn+T)));
y_sim = y_sim(burn+1:burn+T,:);
p_sim = p_sim(burn+1:burn+T,:);
i_sim = i_sim(burn+1:burn+T,:);
Y = [y_sim p_sim i_sim];
    [B,S] = estimVAR(Y,nlag);
    mod_aux_sim = [mod_aux_sim [vec(B);vech(S)]];
end
w = mean(mod_aux_sim,2) - mod_aux_target ;
res = w'*ponderation*w ;
```

La procédure de minimisation de la fonction objectif est là encore

```
xsol = fminsearch(@(xx) funobjVAR(invtrans(xx),ee,VAR_coef_obs,4,T,burn,...
                                                    nsim,ponderation),xinit,opt)
```

avec `ponderation` une matrice diagonale par block contenant l'inverse de la matrice de variance-covariance de l'estimateur de `B` et l'inverse de la matrice de variance-covariance de l'estimateur de `S`.

En appliquant cette procédure (avec `nsim`=100 et `seed`=33), on trouve

```
invtrans(xsolVAR) =
  0.0439  0.1170  1.3537  0.6806  0.7909  0.9384  0.0147  0.0069  0.0053
```

en rappelant à nouveau l'ordre des paramètres

```
 1:k  2:ay  3:ap  4:rp  5:rn  6:ri  7:sp  8:sn  9:si
```