Économétrie des séries temporelles non-stationnaires

Chapitre 4: Implémentation sous MATLAB

Gilles de Truchis

Master 2 EIPMC

Les chapitres du cours

Plan

MATLAB

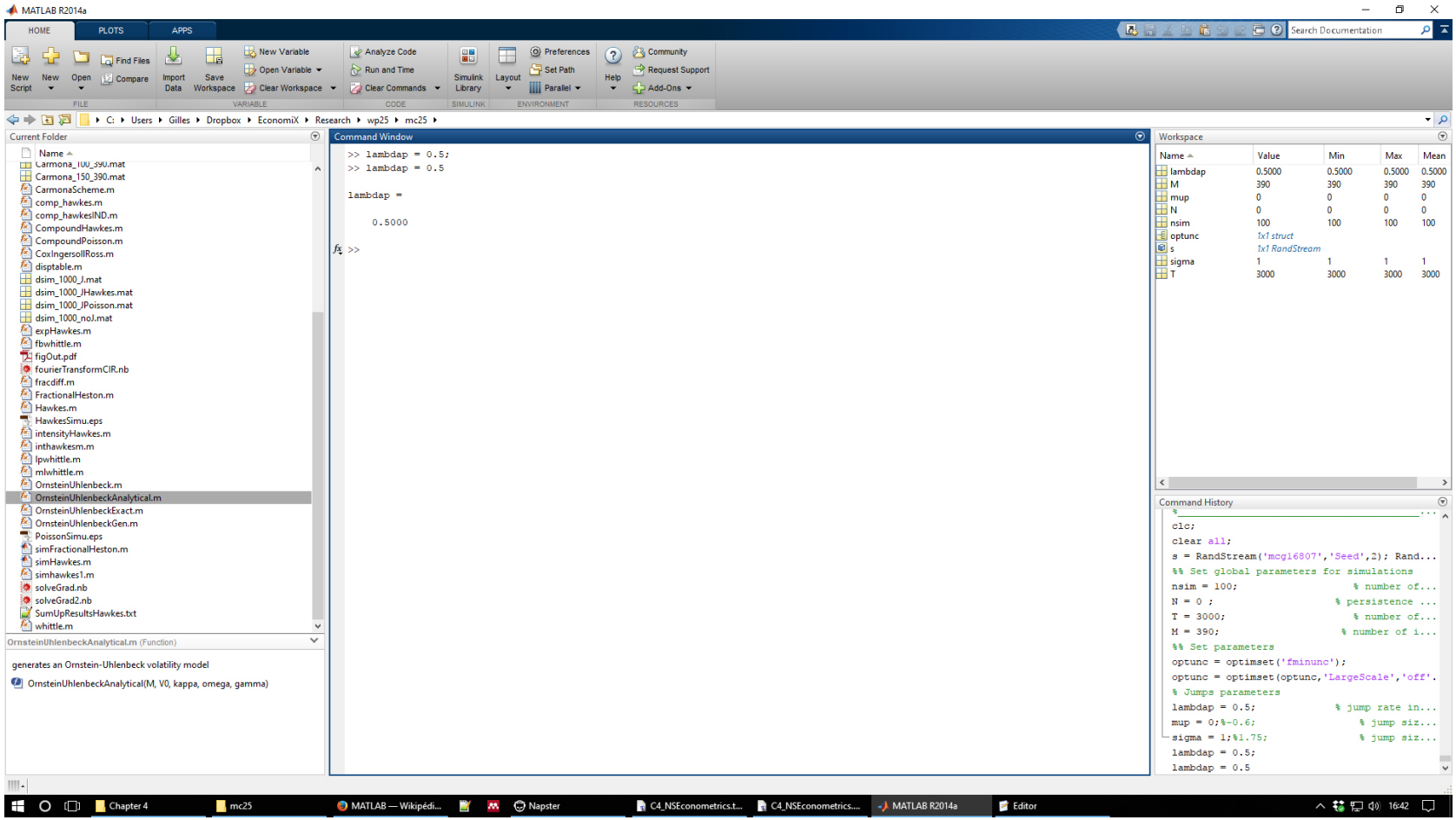
- MATLAB (Matrix Laboratory) est un langage matriciel
- L'implémentation se fait via un environnement développeur

  $\Rightarrow$ le logiciel propriétaire MATLAB

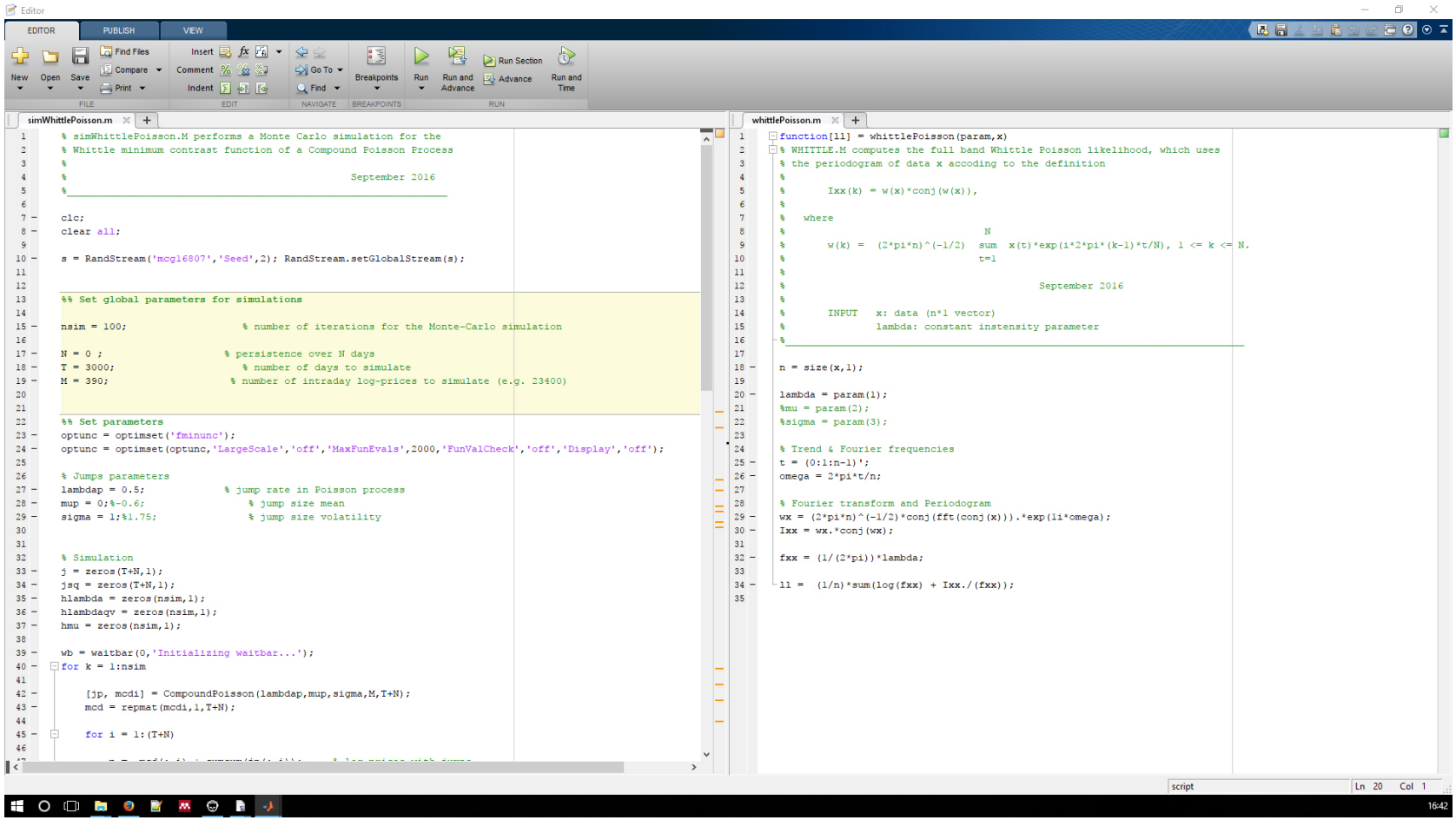
  $\Rightarrow$ le logiciel libre Octave

  $\Rightarrow$ le logiciel libre Scilab (requiert la conversion du code)
- L'interface de MATLAB
  - permet d'exécuter des commandes simples
  - d'appeler des scripts contenant des commandes plus complexes
  - de construire simplement des graphiques

Interface principale



Interface de l'éditeur de script



Plan

Trouver de l'aide

- Vous devez devenir très vite autonome
- L'aide en ligne est votre amie

  $\Rightarrow$ https://fr.mathworks.com/help/matlab/

- Il existe de nombreux codes disponibles

  $\Rightarrow$ https://fr.mathworks.com/matlabcentral/fileexchange/

  $\Rightarrow$ http://www.varennes-ecofin.com/

Affectation dans un cadre de typage faible

- Les variables sont généralement

  ⇒ un scalaire

  ⇒ une matrice

  ⇒ une chaîne de caractères

- L'affection induit le typage de l'objet

```
1 mySc = 10; %  contient un scalaire
2 myMx = [ 1 2 ; 3 4 ]; % contient une matrice 2x2
3 mySt = 'Hello World'; % contient du texte
4 myVe = 1:1:mySc % contient un vecteur 1x10
5 myVe =
6       1  2  3  4  5  6  7  8  9  10
```

Note si j'omets le " ;" le résultat s'affiche dans la fenêtre de commande

Opération simple

- Les opérations de bases

  ⇒ sur un scalaire

```
mySc = mySc + 1; % addition
mySc = mySc - 1; % soustraction
mySc = mySc * 1; % multiplication
mySc = mySc / 1; % division
mySc = mySc^2; % puissance
```

  ⇒ une matrice

```
myMx = myMx + myMx; % addition
myMx = myMx - myMx; % soustraction
myMx = [ 1 2 ]*[ 3; 4 ]; % multiplication
myMx = myMx.*myMx; % produit d'Hadamard
myMx = myMx^2; % puissance d'une matrice
myMx = myMx.^2; % puissance element par element
myMx = myMx'; % transpose la matrice
```

- **help** : https://fr.mathworks.com/help/matlab/arithmetic-operators.html

Autres opérateurs

⇒ Opérateurs de relations

- **help** : https://fr.mathworks.com/help/matlab/relational-operators.html

```
A == B; % 1 si vrai, 0 sinon
A >= B; %  1 si vrai, 0 sinon
A ~= B; % (different) 1 si vrai, 0 sinon
```

⇒ Opérateurs logiques

- **help** : https://fr.mathworks.com/help/matlab/logical-operations.html

```
A & B; % et
A ~ B; %  n'est pas
A | B; % ou
```

⇒ Opérateurs arithmétiques

```
ceil(3.5784) = 4; % entier superieur
fix(3.5784) = 3; % troncature
floor(3.5784) = 3;  % entier inferieur
round(3.5784) = 4;  % arrondi
```

Charger des bases de données

⇒ depuis **excel**

```
1 data = xlsread('data.xlsx',1);
```

- **data** contient la base de la feuille **1** sans entêtes ni dates, ni texte

```
1 [data,txt,raw] = xlsread('data.xlsx',1);
```

- **txt** contient les cellules de la base au format texte
- **raw** contient les cellules de la base quelque soit le format

⇒ depuis un fichier de type **texte** (.txt, .csv, ...)

```
1 data = textscan(fopen('data.csv'),'%f')
```

- **data** contient les données du fichier texte
- **%f** spécifie le format des données dans le fichier (ici des réels)

- **help** : https://fr.mathworks.com/help/matlab/ref/textscan.html

Générer des données

1 Supprimer les instances existantes

```
clc; % vide la fenetre de commande
clear all; % vide l'espace de travail
```

2 Initialiser le générateur de nombre aléatoire

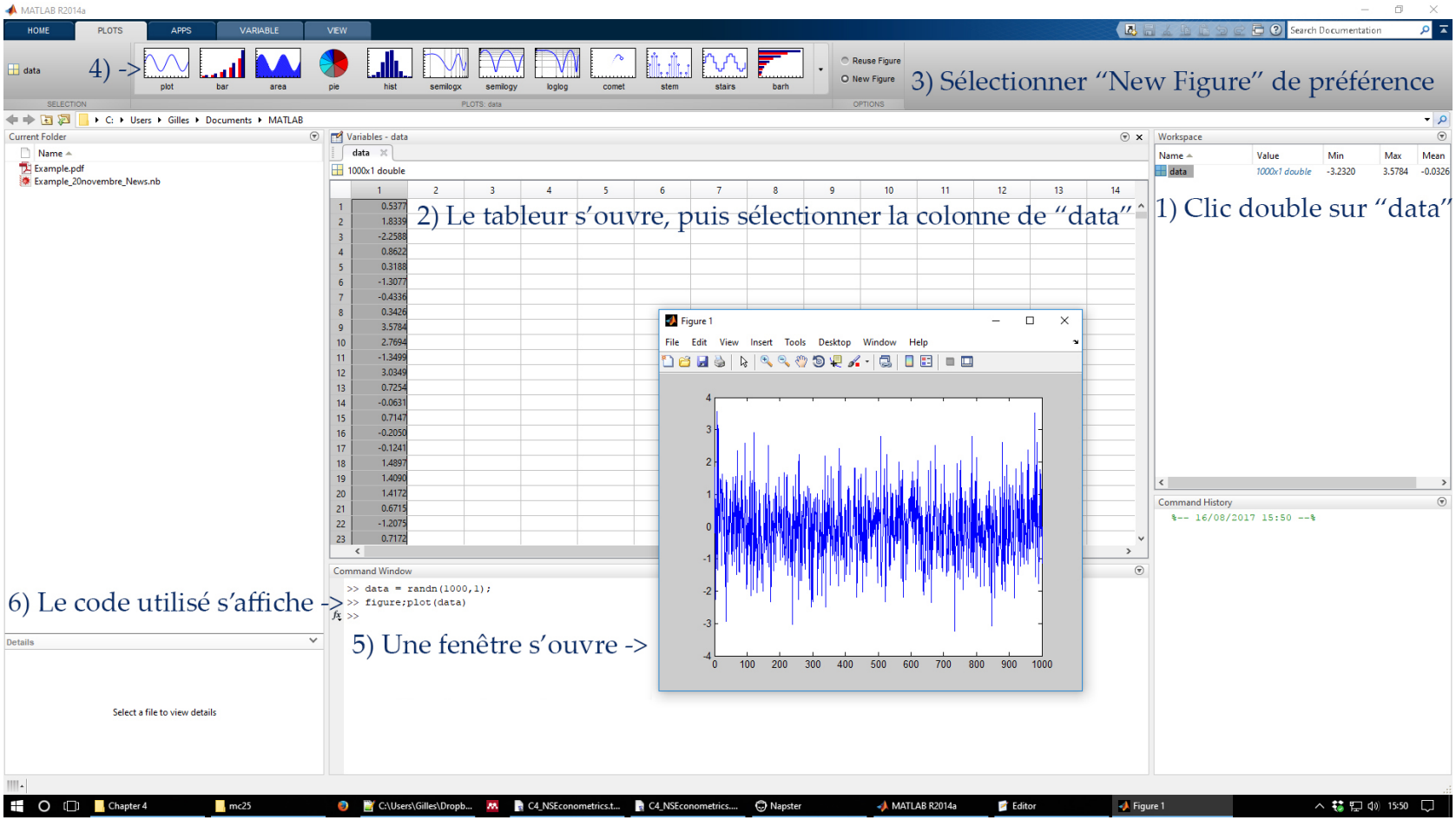
```
s = RandStream('mcg16807','Seed',6);
RandStream.setGlobalStream(s);
```

3 Générer une séquence aléatoire à partir d'une distribution

```
% genere k sequences de n observations a partir
epsilon = rand(n,k) % d'une uniform
epsilon = randn(n,k) % d'une normale (0,1)
epsilon = random('Dist',param,n,k) % toute loi
```

- **help** : https://fr.mathworks.com/help/stats/random.html

Les graphiques



Les graphiques

- Tracer une série univariée est simple

```
figure;
plot(data); % avec data un vecteur nx1 ou 1xn
```

- Tracer deux séries sur un même axe vertical est simple

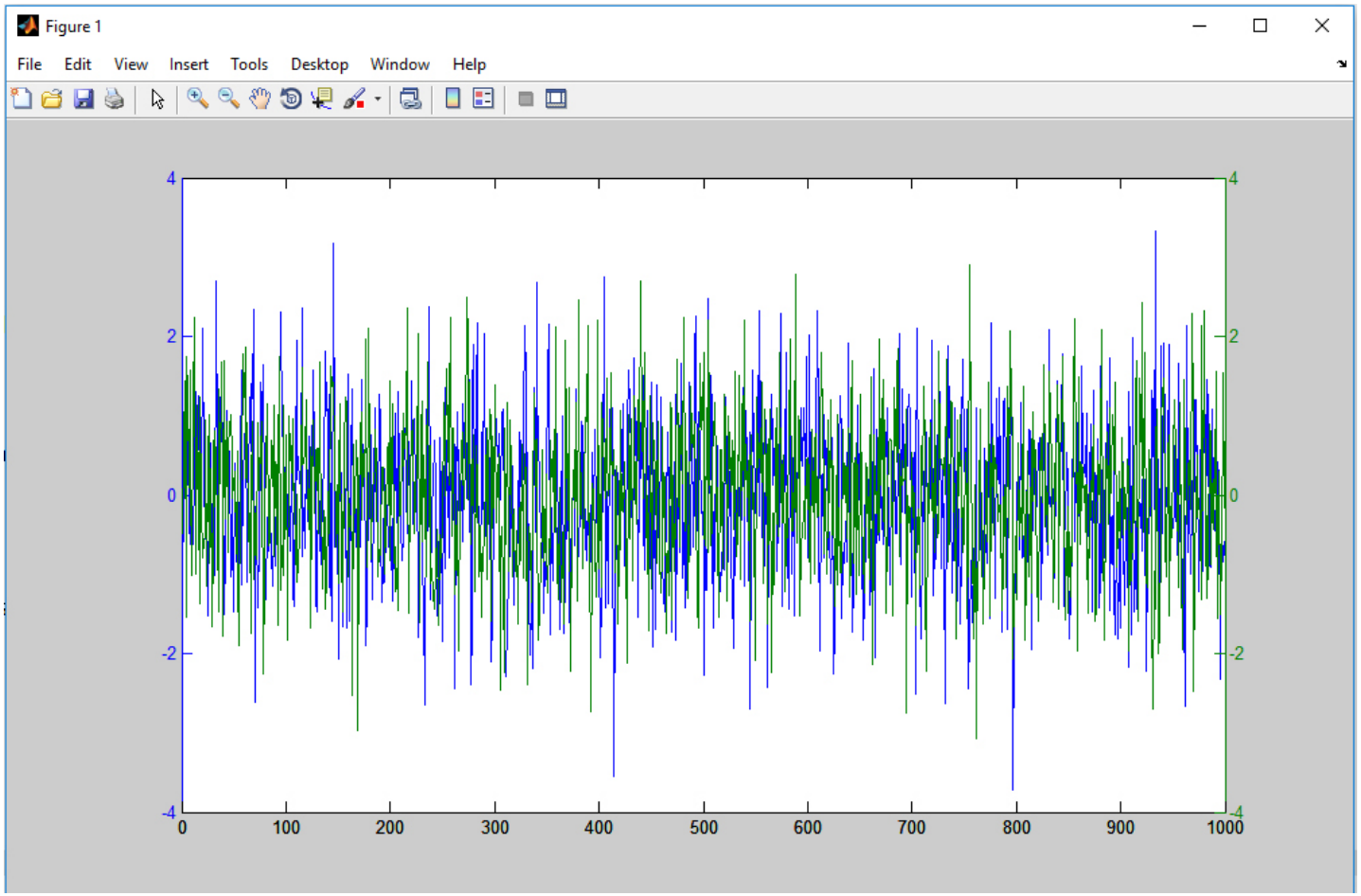
```
figure;
plot(data); % avec data un vecteur nx2 ou 2xn
```

- Mais utiliser deux échelles distinct n'est pas possible avec **plot**

```
figure;
plotyy((1:n)',data(:,1),(1:n)',data(:,2));
```

Note **data(:,1)** signifie "sélectionner toutes les lignes de la colonne 1"

Les graphiques



Boucles et expressions conditionnelles

- Création d'un boucle

```
1 for x = 1:1:10
2     disp(x)
3 end
```

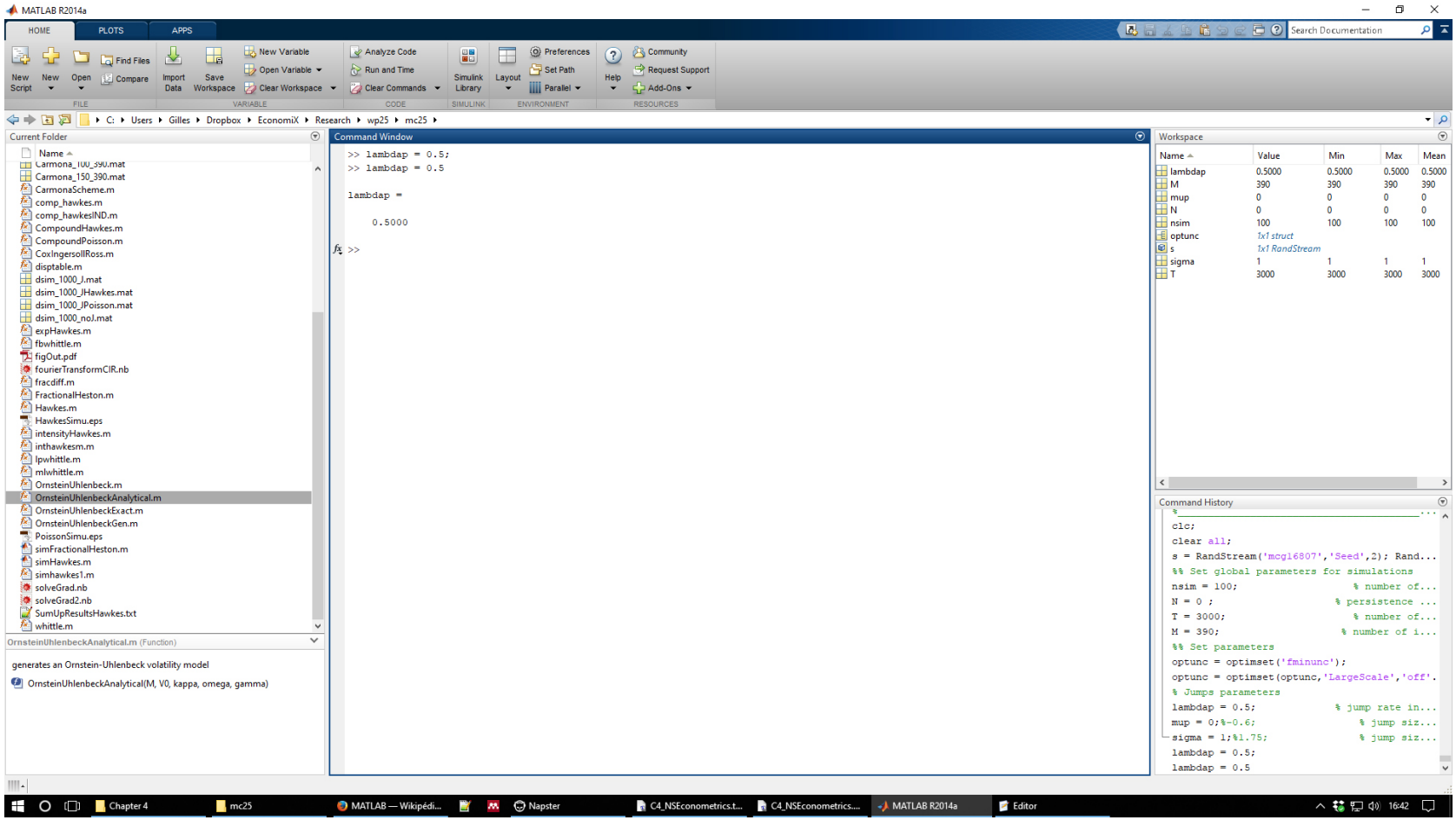
- Création d'une expression conditionnelle

```
1 if (x >= 3) && (x <= 7)
2     disp('Value within specified range.')
3 elseif (x > 7)
4     disp('Value exceeds maximum value.')
5 else
6     disp('Value is below minimum value.')
7 end
```

$\Rightarrow$ Bien d'autres instructions de contrôle existent

- **help** : https://fr.mathworks.com/help/matlab/control-flow.html

Créer un nouveau script



Main script

- Le script principal va appeler les sous-scripts et les fonctions

```
1  % myMainScript.m permet de blablabla ...
2  % Bien commenter le code est tres important
3  %                      September 2017
4  %_____________________________________
5  clc;
6  clear all;
7
8  addpath('subscripts_File'); % sous repertoire
9
10 s = RandStream('mcg16807','Seed',6);
11 RandStream.setGlobalStream(s);
12
13 % Core of the main script
14 ...
```

- les sous-scripts sont parfois nombreux

$\Rightarrow$ les placer dans un sous-répertoire est alors judicieux

Note Par la suite, les lignes 1 à 13 du **script principal** seront implicites

Les fonctions

- Simulation de l'EDS : $V_t = V_0 e^{-\kappa t} + \omega(1 - e^{-\kappa t}) + \gamma e^{-\kappa t} \int_0^t e^{\kappa t} dW$

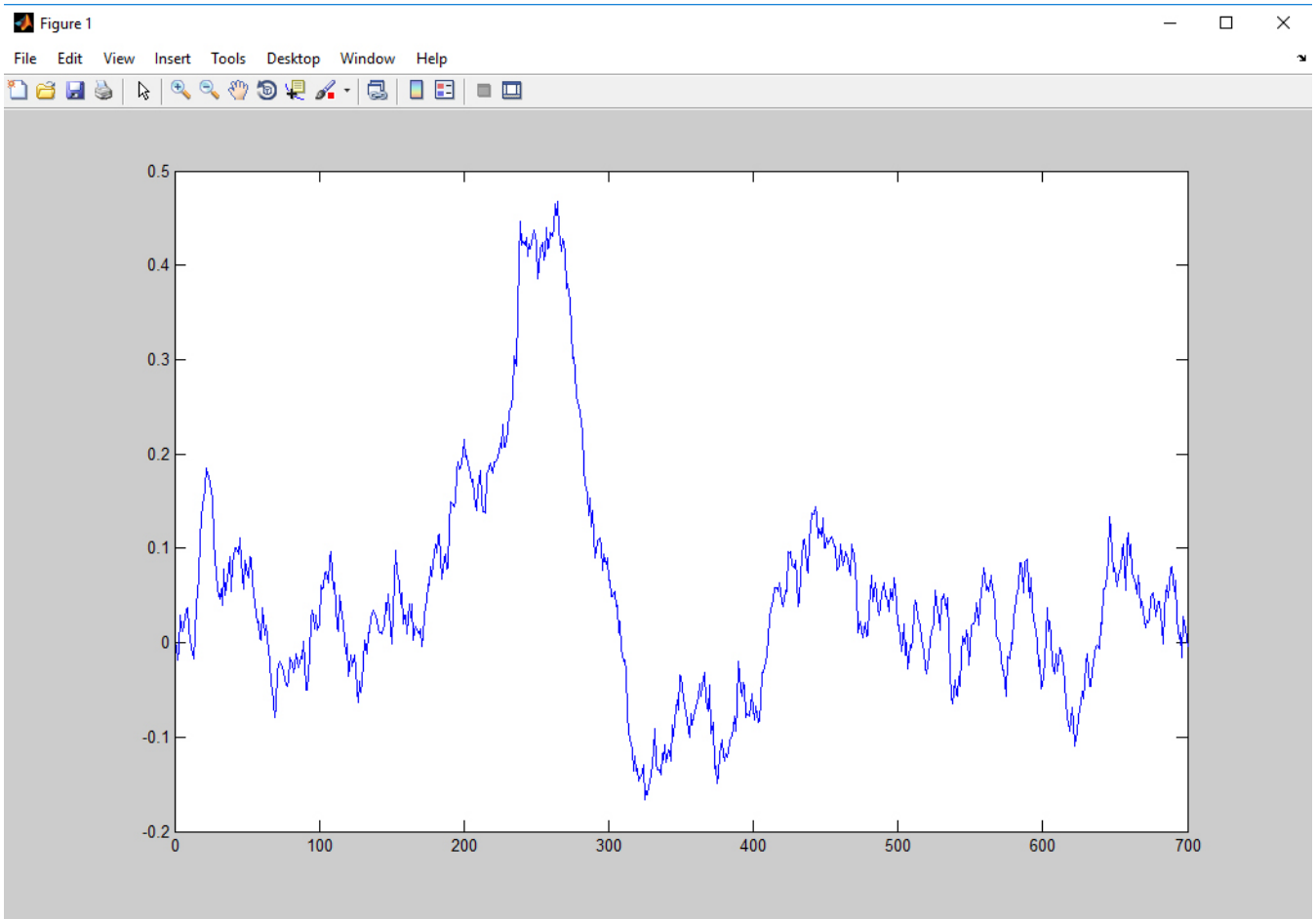
```
% OrnsteinUhlenbeckDE.m generates an OU model
%
% M: the number of observations
% V0: the process at t = 0
% kappa: the adjustement speed of the process
% omega: the asymptotic mean of the process
% gamma: the volatility of the process
%
%        Gilles de Truchis, December 2016
%_____________________________________
function [ou,dW] = OrnsteinUhlenbeckDE(M,V0,kappa,omega,gamma)

dt = 1/M;
t = (0:dt:1)';
dW = randn(M,1);

nex = exp(-kappa*t);
pex = exp(2*kappa*t);
dex = diff(pex-1);

ou = V0*nex + omega*(1-nex) + ... % permet un saut de ligne
     gamma*nex.*cumsum([0; sqrt(dex).*dW])/sqrt(2*kappa);

ou(1) = [];
```

Appel de fonctions

- Appelons cette fonction dans le script principal

```
1  ...
2
3  % Core of the main script
4  M = 390;
5  gamma = 0.5;
6  kappa = 1;
7  omega = 0.1875;
8  v0 = 0;
9
10 % [ou,dW] = OrnsteinUhlenbeckDE(M, v0, kappa, omega, gamma)
11 ouSeq = OrnsteinUhlenbeckDE(M, v0, kappa, omega, gamma);
12
13 figure;
14 plot(ouSeq);
```

Dynamique du processus Ornstein Uhlenbeck généré



Plan

Estimation par OLS

- Les OLS apparaissent comme la solution d'un système d'équation
- **mldivide** ou "\" solutionne des systèmes du type $Y = \beta X$

$\Rightarrow$ L'estimateur OLS $\hat{\beta}$ est alors obtenu très simplement

```
1  function [beta, tstat, std, yhat] = olsDE(y,x)
2
3  [n,k] = size(x);
4  beta=x\y; % calcul beta
5  yhat=x*beta;
6  epsilon=y-yhat; % les erreurs
7
8  vcv = epsilon'*epsilon/(n - k); % la variance des erreurs
9  std = sqrt(diag(vcv*inv(x'*x))); % standard error
10 tstat = beta./std; % les t-stats
```

Exemple d'estimation par OLS

```
...

% Core of the main script

%% Simulation of the linear regression model
n = 1000;
x = ones(n,2);

mu = [0,0];
sigma = [1,0.5;0.5,1];
rsim = mvnrnd(mu,sigma,n);
y = rsim(:,1);
x(:,2) = rsim(:,2);

%% OLS estimation

[beta, tstatLS, stdLS, yhat] = olsDE(y,x);
```

Estimation par MLE d'un modèle linéaire

- Soit un modèle linéaire : $Y_t = \beta X_t + \varepsilon_t$
- On suppose la normalité des erreurs, i.e. $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$
- La fonction objectif du MLE est alors

$$\ell_n(\theta; y|x) = -\frac{n}{2}\ln(\sigma_\varepsilon^2) - \frac{n}{2}\ln(2\pi) - \frac{1}{2\sigma_\varepsilon^2}\sum_{i=1}^{n}(y_i - \beta x_i)^2$$

```
1  function ll = linearMLEDE(param,y,x)
2
3  [n,k] = size(x);
4  beta = param(1:end-1);
5  sigma = param(end);
6
7  epsilon = y;
8  for i = 1:k
9      epsilon = epsilon - beta(i)*x(:,i);
10 end
11
12 ll = n/2*log(sigma^2) + n/2*log(2*pi) + ...
13      1/(2*sigma^2)*sum(epsilon.^2);
```

Note On code $-\ell_n(\theta; y|x)$ car le solver de MATLAB résous un problème de minimisation

Exemple d'estimation par MLE d'un modèle linéaire

- On invoque **fminunc** pour une optimisation numérique non-contrainte

```
1  ...
2
3  % Core of the main script
4
5  %% Simulation of the linear regression model
6  n = 1000;
7  x = ones(n,2);
8
9  mu = [0,0];
10 sigma = [1,0.5;0.5,1];
11 rsim = mvnrnd(mu,sigma,n);
12 y = rsim(:,1);
13 x(:,2) = rsim(:,2);
14
15 %% linear MLE estimation
16 optunc = optimset('fminunc');
17 optunc = optimset(optunc,'LargeScale','off','Display','off');
18
19 [betaML,fval,exitflag,output,grad,hessian] = ...
20 fminunc('linearMLEDE',[0.1,0.2,0.3]',optunc,y,x);
21
22 stdLML = sqrt(diag(inv(hessian)));
23 tstatLML = betaML./stdLML;
```

Estimation par MLE d'un modèle non-linéaire

- Soit $\varepsilon_t = \sqrt{h_t} z_t$ avec $z_t \sim \mathcal{N}(0,1)$
- On suppose $h_t = \sigma_t^2 \sim \text{GARCH}(1,1) : \sigma_t^2 = \omega + \alpha \varepsilon_{t-1}^2 + \beta \sigma_{t-1}^2$
- La fonction objectif du MLE est alors

$$\ell_n(\theta; \varepsilon) = -\frac{1}{2} \sum_{t=1}^{n} \ln(\sigma_t^2) - \frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^{n} \frac{\varepsilon_t^2}{\sigma_t^2}$$

```
1  function ll = garchMLEDE(param,x,initStd)
2
3  n = size(x,1);
4  h = ones(n,1)*initStd;
5  param(find(param <= 0)) = realmin;
6
7  omega = param(1); alpha = param(2); beta = param(3);
8  t = 2:n;
9
10 for i=2:n
11     h(i) = omega + alpha*x(i-1)^2 + beta*h(i-1);
12 end
13
14 ll = 0.5*(sum(log(h(t))) + (n-1)*log(2*pi)) + ...
15       0.5*sum((x(t).^2)./h(t));
```

Note On code $-\ell_n(.)$ car le solver de MATLAB résous un problème de minimisation

Exemple d'estimation par MLE d'un modèle non-linéaire

- On invoque **fmincon** pour une optimisation numérique contrainte

```
...

% Core of the main script

%% Simulation of the linear regression model
n = 2000;
Mdl = garch('Constant',0.01,'GARCH',0.7,'ARCH',0.2);
[v,r] = simulate(Mdl,n);

%% non-linear constrained MLE estimation
optcon = optimset('fmincon');
optcon = optimset(optcon,'Algorithm','sqp',...
'LargeScale','off','Display','off');

A = -eye(3); % A*x <= ub
ub = [0;0;0];

[thetaML,~,~,~,~,~,hessian] = fmincon('garchMLEDE',...
            [0.1,0.2,0.4]',A,ub,[],[],[],[],[],optcon,r,std(r));

stdNLML = sqrt(diag(inv(hessian)));
tstatNLML = thetaML./stdNLML;
```

Plan

OLS et marche aléatoire : estimation

- Le fichier **fxdata.xlsx** contient des cours de change et de futurs
  - données journalières entre 2000 et 2014 ($n = 3760$)
- On se demande si ces séries suivent une marche aléatoire

```
% mainAR1.m performs an OLS estimation of an AR1
%
%                       September 2017
%_____________________________________________
clc;
clear all;

addpath('subscripts');

data = xlsread('fxdata.xlsx',1);

%% Estimation of an AR1 process
n = size(data,1);
fx = data(2:end,2);
fxlag = data(1:end-1,2);

[beta, tstatLS, stdLS, yhat] = olsDE(fx,fxlag); % beta =
    0.9999919311
```

OLS et marche aléatoire : test

- On sait que si $x_t = \rho x_{t-1} + \varepsilon_t$ et $\rho = 1$

... la distribution de $\hat{\rho}$ est non-standard (voir chapitre 1)

$$s_{\widehat{\rho}} = n(\widehat{\rho} - 1) \xrightarrow{d} \frac{1/2\big(W(1)^2 - 1\big)}{\int_0^1 W(r)^2 dr} = \mathcal{L}(\widehat{\rho})$$

- La distribution $\mathcal{L}(\widehat{\rho})$ est tabulée dans le tableau qui suit
- $\mathcal{L}(\widehat{\rho})$ peut servir pour construire un test basée sur $H_0 : \rho = 1$
- Calculons la statistique $s_{\widehat{\rho}}$ pour $n = 3760$

```
%% Radom walk ?
rwStat = n*(beta - 1); % rwStat = -0.03033893
```

- Dans la table, pour $n > 500 \equiv \infty$, 95% des fois lorsque le processus est vraiment une marche aléatoire, $s_{\widehat{\rho}}$ est supérieur à -8.1

$\Rightarrow$ puisque $-0.03 > -8.1$, l'hypothèse nulle $\rho = 1$ est acceptée au seuil de 5% et $x_t \sim I(1)$

OLS et marche aléatoire : annexe

TABLE B.5
Critical Values for the Phillips-Perron Z_ρ Test and for the Dickey-Fuller Test Based on Estimated *OLS* Autoregressive Coefficient

Sample size T	*Probability that $T(\hat{\rho} - 1)$ is less than entry*							
	0.01	*0.025*	*0.05*	*0.10*	*0.90*	*0.95*	*0.975*	*0.99*
				Case 1				
25	−11.9	−9.3	−7.3	−5.3	1.01	1.40	1.79	2.28
50	−12.9	−9.9	−7.7	−5.5	0.97	1.35	1.70	2.16
100	−13.3	−10.2	−7.9	−5.6	0.95	1.31	1.65	2.09
250	−13.6	−10.3	−8.0	−5.7	0.93	1.28	1.62	2.04
500	−13.7	−10.4	−8.0	−5.7	0.93	1.28	1.61	2.04
∞	−13.8	−10.5	−8.1	−5.7	0.93	1.28	1.60	2.03
				Case 2				
25	−17.2	−14.6	−12.5	−10.2	−0.76	0.01	0.65	1.40
50	−18.9	−15.7	−13.3	−10.7	−0.81	−0.07	0.53	1.22
100	−19.8	−16.3	−13.7	−11.0	−0.83	−0.10	0.47	1.14
250	−20.3	−16.6	−14.0	−11.2	−0.84	−0.12	0.43	1.09
500	−20.5	−16.8	−14.0	−11.2	−0.84	−0.13	0.42	1.06
∞	−20.7	−16.9	−14.1	−11.3	−0.85	−0.13	0.41	1.04
				Case 4				
25	−22.5	−19.9	−17.9	−15.6	−3.66	−2.51	−1.53	−0.43
50	−25.7	−22.4	−19.8	−16.8	−3.71	−2.60	−1.66	−0.65
100	−27.4	−23.6	−20.7	−17.5	−3.74	−2.62	−1.73	−0.75
250	−28.4	−24.4	−21.3	−18.0	−3.75	−2.64	−1.78	−0.82
500	−28.9	−24.8	−21.5	−18.1	−3.76	−2.65	−1.78	−0.84
∞	−29.5	−25.1	−21.8	−18.3	−3.77	−2.66	−1.79	−0.87

The probability shown at the head of the column is the area in the left-hand tail.

Source: Wayne A. Fuller, *Introduction to Statistical Time Series*, Wiley, New York, 1976, p. 371.

Plan

Cointégration et structure par terme : théorie

- Soit $i_{m,t}$ le taux d'une obligation zéro-coupon de maturité m
- On note η_t la prime associée à la maturité m
- En l'absence d'opportunité d'arbitrage on peut montrer que

$$i_{m,t} - i_{1,t} = m^{-1} \sum_{j=1}^{m-1} (m-j)\mathbb{E}_t(\Delta i_{1,t+j}) + \eta_t = \zeta_t + \eta_t$$

  voir Giese (2008) pour une démonstration

- On vérifie assez facilement que $i_{m,t}$ est $I(1) \Rightarrow \zeta_t \sim I(0)$

$\Rightarrow$ si $\eta_t \sim I(0)$, $i_{m,t}$ et $i_{1,t}$ devraient être cointégrés de vecteur $(1, -1)'$

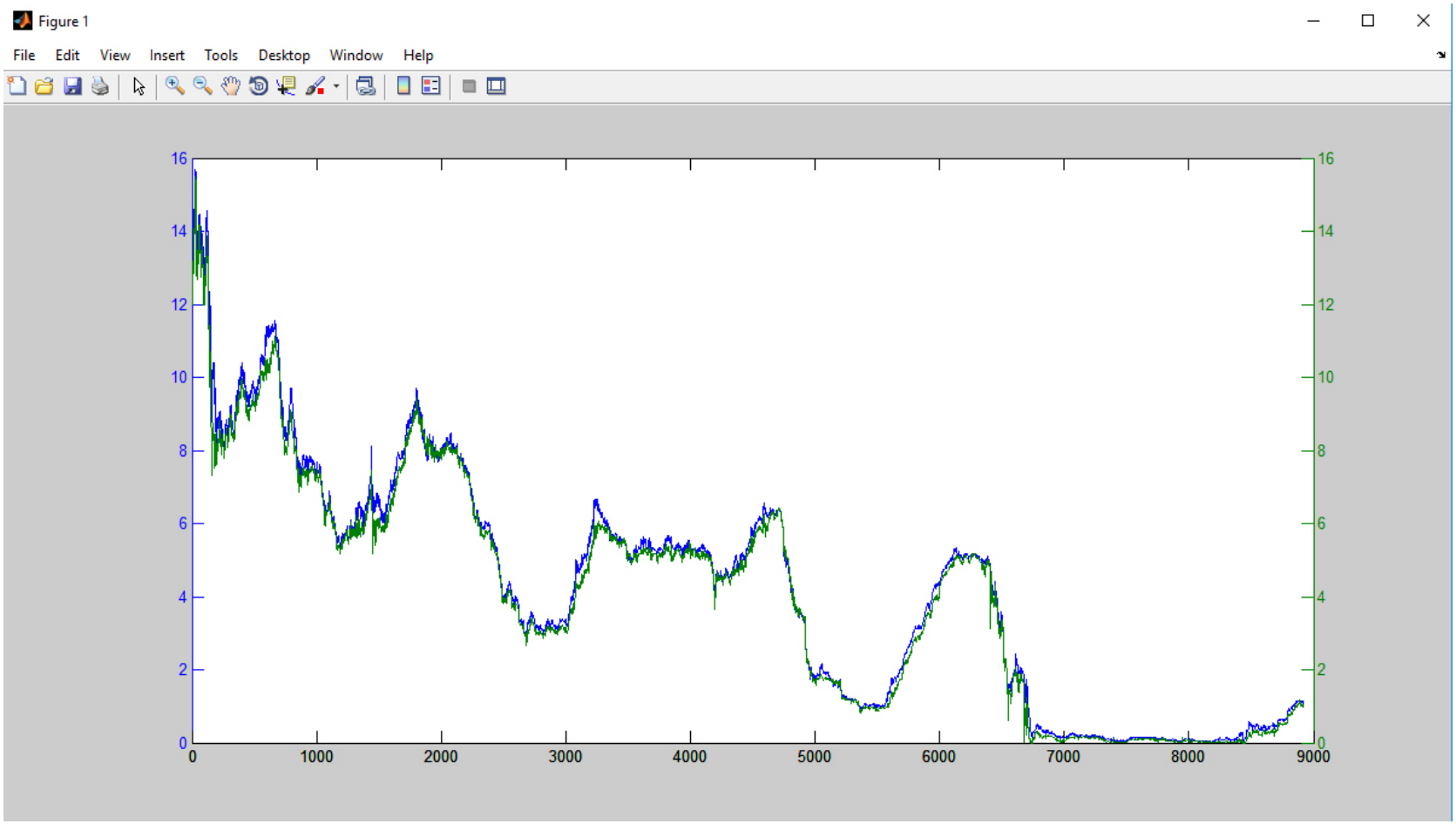
- On cherchera donc à estimer $i_{m,t} = \beta i_{1,t} + z_t$

Cointégration et structure par terme : les données

- La base **idata.xlsx** contient les taux à **3 et 6 mois**, 1 et 2 ans

```
% mainEG87.m term structure cointegration analysis
%
%                       September 2017
%_____________________________________
clc;
clear all;

addpath('subscripts');

data = xlsread('idata.xlsx',1);

%% Data analysis
n = size(data,1);

x = data(2:end,1);
xlag = data(1:end-1,1); % 3 month interest rate

y = data(2:end,2);
ylag = data(1:end-1,2); % 6 month interest rate

figure;
plotyy(1:n-1,y,1:n-1,x);
```

Cointégration et structure par terme : les données



Cointégration et structure par terme : Engle et Granger

- On cherche à savoir si les taux à **3 et 6 mois** sont cointégrées

```
%% Radom walk ?
[by, ty, stdy, ~] = olsDE(y,ylag); % beta = 0.9995942451
[bx, tx, stdx, ~] = olsDE(x,xlag); % beta = 0.9996227180

rwy = n*(by - 1); % rwStat = -3.75536802
rwx = n*(bx - 1); % rwStat = -3.59486769

%% OLS estimation of long run equilibrium
[bci, tci, stdci, yhat] = olsDE(y,[ones(n-1,1) x]);
% beta = 1.0304789539

z = y - yhat;
figure;
plot(1:n-1,z);
```

- $\hat{\beta}$ approche 1 mais seul un test sur $\hat{z}_t$ permettra de conclure
- Visuellement il est également difficile de conclure

Cointégration et structure par terme : test sur z_t

- On sait que si la relation est factice, $\hat{z}_t = \rho \hat{z}_{t-1} + \epsilon_t$ et $\rho = 1$

... la distribution de $\hat{\rho}$ est non-standard (voir chapitre 2)

$$s_{\widehat{\rho}} = (n-1)(\hat{\rho}-1) \xrightarrow{d} \left[\frac{1}{2} \left(\begin{pmatrix} 1 & -h_2' \end{pmatrix} \cdot \boldsymbol{W}(1)\boldsymbol{W}(1)' \cdot \begin{pmatrix} 1 \\ -h_2 \end{pmatrix} \right) - h_1 \boldsymbol{W}(1)' \begin{pmatrix} 1 \\ -h_2 \end{pmatrix} - \frac{1}{2}(1 + h_2' h_2) \right] H_n^{-1}$$

- La distribution $\mathcal{L}(\widehat{\rho})$ est tabulée dans le tableau qui suit
- $\mathcal{L}(\widehat{\rho})$ peut servir pour construire un test de cointégration basé sur les résidus estimés $H_0 : \rho = 1$

Cointégration et structure par terme : application du test

- Calculons la statistique $s_{\widehat{\rho}}$ pour $n = 8923$

```
%% As both series are I(1) regression might be spurious
zhat = z(2:end);
zhatlag = z(1:end-1);
[bz, tz, stdz, ~] = olsDE(zhat,zhatlag); % beta = 0.9654598939
rwzhat = (n-1)*(bz - 1); % rwStat = -308.1668258938
```

- Les cas 1, 2 et 3 indiquent que l'équation de long terme
  1. est sans constante
  2. est avec constante
  3. est avec constante et tendance
- Dans le cas 2, pour 1 régresseur et $n = 500$, 95% des fois lorsque les résidus sont vraiment $I(1)$, $s_{\widehat{\rho}}$ est supérieur à -20.5

$\Rightarrow$ puisque $-308.16 < -20.5$, l'hypothèse nulle $\rho = 1$ est rejetée au seuil de 5% et $\hat{z}_t \sim I(0)$

Cointégration et structure par terme : annexe

TABLE B.8
Critical Values for the Phillips Z_ρ Statistic When Applied to Residuals from Spurious Cointegrating Regression

Number of right-hand variables in regression, excluding trend or constant $(n-1)$	*Sample size* (T)	*Probability that* $(T-1)(\hat{\rho}-1)$ *is less than entry*						
		0.010	*0.025*	*0.050*	*0.075*	*0.100*	*0.125*	*0.150*
				Case 1				
1	500	−22.8	−18.9	−15.6	−13.8	−12.5	−11.6	−10.7
2	500	−29.3	−25.2	−21.5	−19.6	−18.2	−17.0	−16.0
3	500	−36.2	−31.5	−27.9	−25.5	−23.9	−22.6	−21.5
4	500	−42.9	−37.5	−33.5	−30.9	−28.9	−27.4	−26.2
5	500	−48.5	−42.5	−38.1	−35.5	−33.8	−32.3	−30.9
				Case 2				
1	500	−28.3	−23.8	−20.5	−18.5	−17.0	−15.9	−14.9
2	500	−34.2	−29.7	−26.1	−23.9	−22.2	−21.0	−19.9
3	500	−41.1	−35.7	−32.1	−29.5	−27.6	−26.2	−25.1
4	500	−47.5	−41.6	−37.2	−34.7	−32.7	−31.2	−29.9
5	500	−52.2	−46.5	−41.9	−39.1	−37.0	−35.5	−34.2
				Case 3				
1	500	−28.9	−24.8	−21.5	—	−18.1	—	—
2	500	−35.4	−30.8	−27.1	−24.8	−23.2	−21.8	−20.8
3	500	−40.3	−36.1	−32.2	−29.7	−27.8	−26.5	−25.3
4	500	−47.4	−42.6	−37.7	−35.0	−33.2	−31.7	−30.3
5	500	−53.6	−47.1	−42.5	−39.7	−37.7	−36.0	−34.6

The probability shown at the head of the column is the area in the left-hand tail.

Source: P. C. B. Phillips and S. Ouliaris, "Asymptotic Properties of Residual Based Tests for Cointegration," *Econometrica* 58 (1990), pp. 189–90. Also Wayne A. Fuller, *Introduction to Statistical Time Series*, Wiley, New York, 1976, p. 371.

Cointégration et structure par terme : ECM

- A partir de la relation de long terme on construit la forme ECM

$$\Delta i_{m,t} = \omega + \alpha_1 \widehat{z}_{t-1} + \gamma_{11} \Delta i_{m,t-1} + \gamma_{12} \Delta i_{1,t-1} + \zeta_t$$
$$\Delta i_{1,t} = \omega + \alpha_2 \widehat{z}_{t-1} + \gamma_{21} \Delta i_{m,t-1} + \gamma_{22} \Delta i_{1,t-1} + \xi_t$$

- On s'attend à trouver $\alpha_1 < 0$ et $\alpha_2 > 0$

```
%% ECM representation
dly = y - ylag; dy = dly(2:end); dylag = dly(1:end-1);
dlx = x - xlag; dx = dlx(2:end); dxlag = dlx(1:end-1);

[becmy,tecmy,~,~]=olsDE(dy,[ones(size(dy)) zhatlag dylag dxlag])
[becmx,tecmx,~,~]=olsDE(dx,[ones(size(dx)) zhatlag dylag dxlag])

```

	y	x
omega	−0,0013	−0,0012
alpha	−0,0070	0,0226
gamma1	−0,0717	0,0914
gamma2	0,1506	0,0238

Cointégration et structure par terme : Johansen

- L'approche d'Engle et Granger (1987) nous contraint à
  - formuler une hypothèse sur la variable faiblement exogène
  - se limiter à un système bivarié
- Johansen (1991) permet de s'affranchir de ces contraintes

$\Rightarrow$ programmons les tests $\mathcal{LR}_g^{(\tau)}$ et $\mathcal{LR}_g^{(\lambda)}$

```
1  % johansenMLEDE.m compute the Johansen trace and eigmax tests
2  %
3  % Inputs:      x = n x k matrix of time-series in levels
4  %              p = number of lags in the model
5  %              c = order of time polynomial
6  %                  c = 0, no deterministic part
7  %                  c = 1, for constant term
8  %                  c = 2, for constant plus time-trend
9  %                  c > 2, for higher order polynomial
10 %
11 %                     September 2017
12 %_____________________________________
13 function [lrt,lrm,cvt,cvm,r,nsevec] = johansenMLEDE(x,p,c)
```

Cointégration et structure par terme : étape 1

- On commence par construire les régressions auxiliaires
  - $\Delta X_t = \sum_{j=1}^{p} \Omega_j \Delta X_{t-j} + \omega_t$
  - $X_{t-1} = \sum_{j=1}^{p} \Psi_j \Delta X_{t-j} + \psi_t$

```
[n,k] = size(x);

dx = x(2:end,:) - x(1:end-1,:); % DX(t)
dx = detregDE(dx,c); % Remove deterministic components Omega0
dxlag = lagmatrix(dx,1:p); % Create DX(t-1),...,DX(t-p)

dx(1:p,:) = []; % Ajust sample size for DX(t)
dxlag(1:p,:) = []; % Ajust sample size for DX(t-1),...,DX(t-p)

r1hat = (dxlag\dx);
ome = dx - dxlag*r1hat; % First auxiliary regression residuals

x = detregDE(x,c); % Remove deterministic components Psi0
x = lagmatrix(x,p); % Create X(t-p) ...
x(1:p+1,:) = []; % ... to adjust the sample size of X(t)

r2hat = (dxlag\x);
psi = x - dxlag*r2hat; % Second auxiliary regression residuals
```

Cointégration et structure par terme : étape 2

- On mène ensuite l'analyse canonique
  - on résoud le problème des valeurs propres de

$$\left(\widehat{\Sigma}_{\psi\psi}^{-1}\widehat{\Sigma}_{\psi\omega}\widehat{\Sigma}_{\omega\omega}^{-1}\widehat{\Sigma}_{\omega\psi}\right)\widehat{v}_i = \widehat{\lambda}_i\widehat{v}_i$$

```
1  vcvoo = ome'*ome/(n-p-1); % First regression residuals VCV
2  vcvpp = psi'*psi/(n-p-1); % Second regression residuals VCV
3  vcvpo = psi'*ome/(n-p-1); % Cross-residuals VCV
4
5  vcv = inv(vcvpp)*(vcvpo)*inv(vcvoo)*(vcvpo');
6
7  [evec,eval] = eig(vcv); % Raw eigen vectors and eigen values
8  % Normalized eigenvectors (Cholesky(X) ~ sqrt(X))
9  nevec = evec*inv(chol(evec'*vcvpp*evec));
10
11 nevec = nevec'; % Eigenvec by column but sort command by rows
12 % Sort eigen values in descending order
13 [evsort, evindex] = sort(diag(eval),'descend');
14 nsevec = nevec(evindex,:); % Store associate eigen vectors
15 nsevec = nsevec'; % Put eigenvec back in column after sorting
```

Cointégration et structure par terme : étape 3

- On calcule les statistiques de tests
  - test de trace : $\mathcal{LR}_g^{(\tau)} = 2(\mathcal{L}^*_{H_1} - \mathcal{L}^*_{H_0}) = -n \sum_{i=r+1}^{k} \log(1 - \widehat{\lambda}_i)$
  - test $\lambda - \max$: $\mathcal{LR}_g^{(\lambda)} = 2(\mathcal{L}^*_{H_1} - \mathcal{L}^*_{H_0}) = -n \log(1 - \widehat{\lambda}_{r+1})$

```
% Compute the trace and max eigenvalue statistics
lrt = zeros(k,1); lrm = zeros(k,1); % init tests statistics
cvt = zeros(k,3); cvm = zeros(k,3); % init crtitical values
n2 = size(psi,1);

iota = ones(k,1); % init a k x 1 vector of ones
ilam = log(iota-evsort); % Generates the (1 - lambda(i)) vector

for i = 1:k
    ilamtmp = ilam;
    ilamtmp(1:i-1,:) = []; % Adjust (1 - lambda(i)) vector size
    lrt(i,1) = -n2*sum(ilamtmp); % Trace test statistic
    lrm(i,1) = -n2*log(1-evsort(i,1)); % Eigmax test statistic
    cvt(i,:) = c_sjt(k-i+1,c-1); % Trace test critical values
    cvm(i,:) = c_sja(k-i+1,c-1); % Eigmax test critical values

    if lrt(i,1) > cvt(i,2)
        r = i; % Gives the cointegration rank
    end
end
```

Cointégration et structure par terme : $\mathcal{LR}_g^{(\tau)}$ et $\mathcal{LR}_g^{(\lambda)}$

- On applique le programme aux taux $i_{m,t}$, pour $m = 2, 3, 4$

```
% mainJ91.m term structure cointegration analysis
%
%                        September 2017
%_____________________________________________
clc;
clear all;

addpath('subscripts');

data = xlsread('idata.xlsx',1);

%% Computation of trace and lambda-max Johansen's tests

n = size(data,1);
x = data(:,[2 3 4]);
x = log(x);
p = 2;

[lrt,lrm,cvt,cvm,r,civec] = johansenMLEDE(x,p,1);
```

- Sur la base de $\mathcal{LR}_g^{(\tau)}$ on trouve un rang de cointégration $\hat{r} = 2$

  $\Rightarrow$ présence de 2 relations de cointégration entre $i_{2,t}$, $i_{3,t}$ et $i_{4,t}$

Plan

La mémoire longue et le domaine des fréquences

- Le domaine des fréquences est un cadre d'analyse attractif pour les séries $I(\delta)$ et l'autocovariance $\gamma(h)$ de ces séries
- L'équivalent de $\gamma(h)$ en fréquence est la densité spectrale $f(\lambda)$
  - Le périodogramme $I(\lambda)$ estime non-paramétriquement $f(\lambda)$

$\Rightarrow$ simulons un bruit blanc $x_t \sim I(\delta)$ et calculons $f(\lambda)$ puis $I(\lambda)$

```
% mainLM.m long memory analysis and estimation
%
%                       September 2017
%_____________________________________________
clc;
clear all;

addpath('subscripts');
s = RandStream('mcg16807','Seed',6);
RandStream.setGlobalStream(s);

%% Generate a fractional white noise
n = 1000;
delta = 0.4;
e = randn(n,1);
% integrate (if -delta) /  differiate (if delta) fractionnally
x = fracdiff(e,-delta);
```

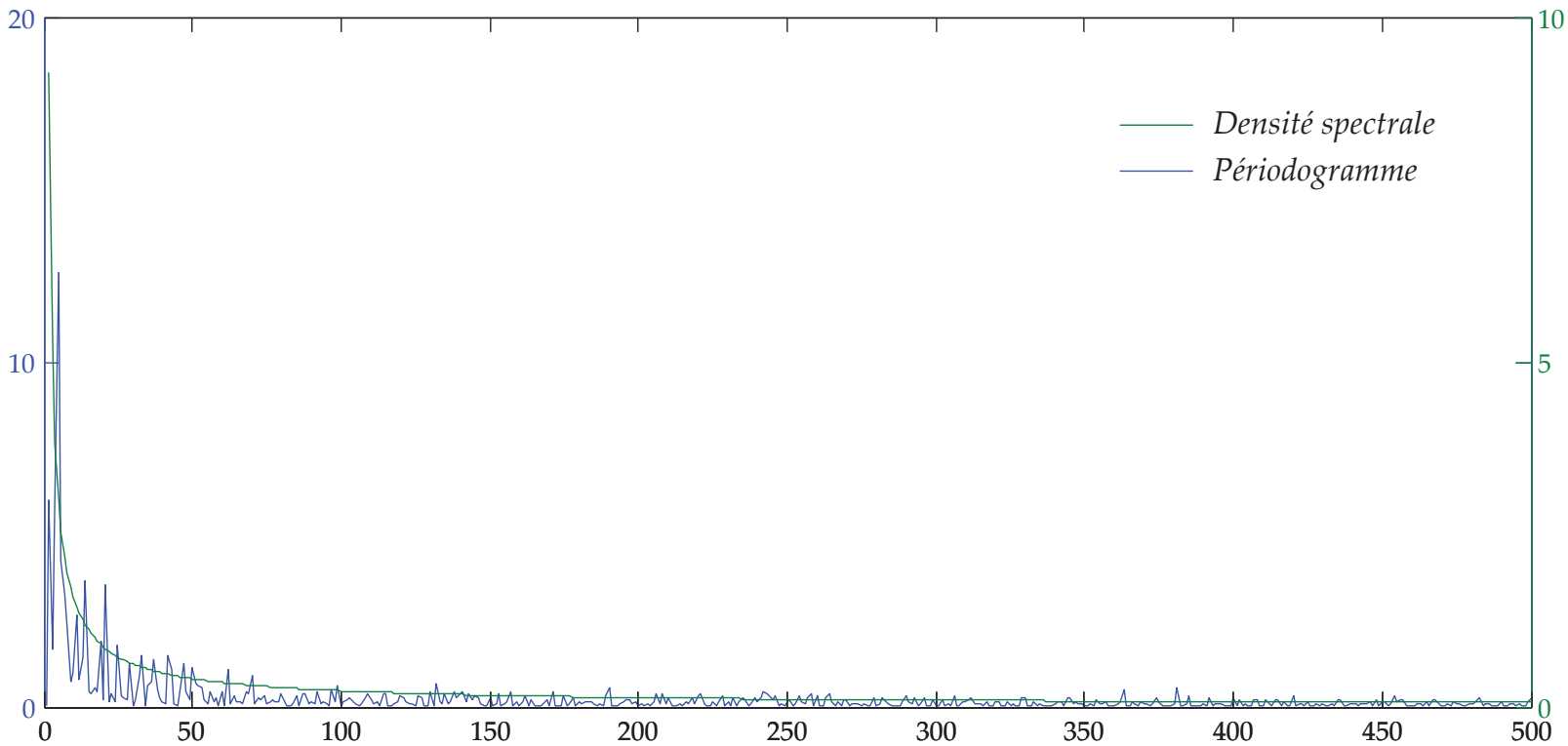
Mémoire longue, densité spectral et périodogramme

- On sait que pour un bruit blanc fractionnaire $f(\lambda) = \sigma_\varepsilon^2 (2\pi)^{-1}$
- On sait que le périodogramme est donné par

$$\widehat{I}_x(\lambda) = |\omega_x(\lambda)|^2 \text{ avec } \omega_x(\lambda) = \frac{1}{\sqrt{2\pi n}} \sum_{t=1}^{n} x_t e^{-it\lambda_j} \text{ et } \lambda_j = \frac{2\pi j}{n}$$

```
%% Periodogramm and spectral density comparison
j = fix(n/2);
t = (0:1:n-1)';
lambda = 2*pi*t/n;

wx = (2*pi*n)^(-1/2)*fft(x).*exp(1i*lambda);
Ix = wx(1:j).*conj(wx(1:j));

fx = 1/(2*pi)*abs(lambda(1:j)).^(-2*delta);

figure;
plotyy(1:j,Ix,1:j,fx);
```

Mémoire longue, densité spectral et périodogramme



Mémoire longue et estimateurs Whittle

- On a vu qu'avec l'aide de $I(\lambda)$ on pouvait construire des estimateurs de δ
- Soit paramétrique de type **Whittle**

$$\mathcal{L}_W(\vartheta; \boldsymbol{x}) \approx \frac{2}{n} \sum_{j=1}^{n} \left(\log f_x(\lambda_j; \vartheta) + \frac{I_x(\lambda_j)}{f_x(\lambda_j; \vartheta)} \right), \quad \lambda_j = 2\pi j/n$$

avec $f_x(\lambda_j)$ la densité spectrale d'un ARFIMA(p, δ, q)

- Soit semi-paramétrique de type **local Whittle**

$$\mathcal{K}_W(\delta; \boldsymbol{x}) = \frac{1}{m} \sum_{j=1}^{m} \log\left(\hat{g}\lambda_j^{-2\delta}\right) + \frac{\hat{g}}{\hat{g}} = \log g_m(\delta) - \delta \frac{2}{m} \sum_{j=1}^{m} \log \lambda_j$$

avec $\hat{g} = g_m(\delta) = m^{-1} \sum_{j=1}^{m} I_x(\lambda_j)\lambda_j^{-2\delta}$ et $m = o(n^{4/5})$

Mémoire longue et estimateur Whittle semi-paramétrique

- Commençons par l'estimateur de type **local Whittle**

```
function [r] = whittleDE(d,x,m)
% WHITTLE.M computes the local Whittle likelihood
%
%                INPUT   x: data (n*1 vector)
%                              m: truncation number
%                              d: parameter value
%
%                        October 2017
%_____________________________________________

n = size(x,1);
t = (0:1:n-1)';
lambda = 2*pi*t/n;
wx = (2*pi*n)^(-1/2)*fft(x).*exp(1i*lambda);
lambda = lambda(2:m+1);
wx = wx(2:m+1);
Ix = wx.*conj(wx);

g = mean((lambda.^(2*d)).*Ix);
r = log(g) - 2*d*mean(log(lambda));
```

Mémoire longue et estimateur Whittle paramétrique

- Poursuivons avec l'estimateur de type **Whittle**

```
function [ll,Ixx,fxxb] = fbwhittleDE(param,data,p,q)
% fbwhittle.m computes the full band Whittle likelihood estimator
%
%                                   Gilles de Truchis, February 2014
%
%                 INPUT    data: n*1
%                                   param: parameter vector
%                 param(1) = d
%                 param(2,...,p+1) = phi_1,...,phi_p
%                 param(p+2,...,q+1) = theta_1,...,theta_q
%                 p: number of lags for the AR part
%                 q: number of lags for the MA part
%
%       NOTE:     The length of the vector 'param' must be 1+p+q
%_____________________________________________________________________________

n = size(data,1);
x  = data;
delta = param(1);
if p == 0; phi = 0; else phi = param(2:1+p); end;
if q == 0; theta = 0; else theta = param(2+p:1+p+q); end;
sigma = param(end);
```

Mémoire longue et estimateur Whittle paramétrique

- Après l'initialisation des paramètres, la construction de $\mathcal{L}_W(\vartheta; \boldsymbol{x})$

```
% Trend & Fourier frequencies
t = (0:1:n-1)';
lambda = 2*pi*t/n;

% Fourier transform and Periodogram
wx = (2*pi*n)^(-1/2)*conj(fft(conj(x))).*exp(1i*lambda);
Ixx = wx.*conj(wx);

% Fractional filter
fd = abs(1-exp(-1i*lambda)).^(-2*delta);

% ARMA filter
ei = ones(n,1);

MA = ei + exp(-1i*lambda*(1:1:length(theta)))*theta;
AR = ei + exp(-1i*lambda*(1:1:length(phi)))*-phi;

% Spectral density
fxxb =  (sigma^2/(2*pi))*fd.*((abs(MA).^2)./(abs(AR).^2));

% Whittle approximation to the likelihood
fxxb=fxxb(2:end);
Ixx=Ixx(2:end);

ll =  (2/n)*sum(log(fxxb)) + (2/n)*sum(Ixx./fxxb);
```

Mémoire longue et estimation d'un bruit blanc $I(\delta)$

- Utilisons ces estimateurs pour notre bruit blanc $I(\delta = 0.4)$ simulé

```
%% Local Whittle estimation of a fractional white noise

optbnd = optimset('fminbnd');

m = fix(n^0.8);
lwhat = fminbnd('whittleDE',-1,3,optbnd,x,m);

%% Full band Whittle estimation of a fractional white noise
optunc = optimset('fminunc');
optunc = optimset(optunc,'LargeScale','off','MaxFunEvals',5000,'
    FunValCheck','off','Display','off');

% specification : ARFIMA(0,d,0)
swhat = fminunc('fbwhittleDE',[0.3,0.1],optunc,x,0,0);

% misspecified : ARFIMA(1,d,0)
mwhat = fminunc('fbwhittleDE',[0.3,0.5,0.1],optunc,x,1,0);
```

- Pour le **local Whittle** on trouve $\widehat{\delta} = 0.4109$
- Pour le **Whittle** sur un ARFIMA$(0, \delta, 0)$ on trouve $\widehat{\delta} = 0.4206$
- Pour le **Whittle** sur un ARFIMA$(1, \delta, 0)$ on trouve $\widehat{\delta} = 0.4350$

Engle, R. F., Granger, C. W. J. (1987). Co-integration and error correction : representation, estimation, and testing. Econometrica, 55(2), 251–276.

Engle, R. F., Yoo, B. S. (1987). Forecasting and testing in co-integrated systems. Journal of econometrics, 35(1), 143-159.

Giese, J. (2008). Level, Slope, Curvature : Characterising the Yield Curve in a Cointegrated VAR Model. Economics-The Open-Access, Open-Assessment E-Journal, 2, 1-20.

Johansen, S. (1991). Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. Econometrica, 1551-1580.