

Économétrie des séries temporelles non-stationnaires

Chapitre 4: Implémentation sous MATLAB

Gilles de Truchis

Master 2 EIPMC

Les chapitres du cours

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation

- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

Plan

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

MATLAB

- MATLAB (Matrix Laboratory) est un langage matriciel
- L'implémentation se fait via un environnement développeur
 - ⇒ le logiciel propriétaire MATLAB
 - ⇒ le logiciel libre Octave
 - ⇒ 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

The screenshot displays the MATLAB R2014a environment. The Command Window shows the following code and output:

```

>> lambda0 = 0.5;
>> lambda0 = 0.5;

lambda0 =

    0.5000
  
```

The Workspace panel shows the following variables:

Name	Value	Min	Max	Mean
lambda0	0.5000	0.5000	0.5000	0.5000
M	300	300	300	300
mpg	0	0	0	0
N	0	0	0	0
max	100	100	100	100
optunc	Inf struct			
sigma	1	1	1	1
T	3000	3000	3000	3000

The Command History panel shows the following code:

```

clear all;
M = RandStream('mrg32v7', 'seed', 2); Rand...
%% Set global parameters for simulation
maxN = 100; % number of...
M = 0; % persistence ...
T = 3000; % number of ...
sigma = 300; % number of ...
%% Set parameters
options = optimset('Display','off');
options = optimset(options, 'LargeScale','off');
% Dump parameters
lambda0 = 0.5; % Jump rate 10...
mpg = 0.1-0.6; % Jump siz...
sigma = 1.1-1.75; % Jump siz...
lambda0 = 0.5;
lambda0 = 0.5
  
```

Interface de l'éditeur de script

```

1 % whittlePoisson.m performs a Monte Carlo simulation for the
2 % Whittle minimum contrast function of a Compound Poisson Process
3 %
4 %
5 %
6 %
7 %
8 %
9 %
10 %
11 %
12 %
13 %
14 %
15 %
16 %
17 %
18 %
19 %
20 %
21 %
22 %
23 %
24 %
25 %
26 %
27 %
28 %
29 %
30 %
31 %
32 %
33 %
34 %
35 %
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 %
50 %
51 %
52 %
53 %
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %
65 %
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %
100 %
101 %
102 %
103 %
104 %
105 %
106 %
107 %
108 %
109 %
110 %
111 %
112 %
113 %
114 %
115 %
116 %
117 %
118 %
119 %
120 %
121 %
122 %
123 %
124 %
125 %
126 %
127 %
128 %
129 %
130 %
131 %
132 %
133 %
134 %
135 %
136 %
137 %
138 %
139 %
140 %
141 %
142 %
143 %
144 %
145 %
146 %
147 %
148 %
149 %
150 %
151 %
152 %
153 %
154 %
155 %
156 %
157 %
158 %
159 %
160 %
161 %
162 %
163 %
164 %
165 %
166 %
167 %
168 %
169 %
170 %
171 %
172 %
173 %
174 %
175 %
176 %
177 %
178 %
179 %
180 %
181 %
182 %
183 %
184 %
185 %
186 %
187 %
188 %
189 %
190 %
191 %
192 %
193 %
194 %
195 %
196 %
197 %
198 %
199 %
200 %
201 %
202 %
203 %
204 %
205 %
206 %
207 %
208 %
209 %
210 %
211 %
212 %
213 %
214 %
215 %
216 %
217 %
218 %
219 %
220 %
221 %
222 %
223 %
224 %
225 %
226 %
227 %
228 %
229 %
230 %
231 %
232 %
233 %
234 %
235 %
236 %
237 %
238 %
239 %
240 %
241 %
242 %
243 %
244 %
245 %
246 %
247 %
248 %
249 %
250 %
251 %
252 %
253 %
254 %
255 %
256 %
257 %
258 %
259 %
260 %
261 %
262 %
263 %
264 %
265 %
266 %
267 %
268 %
269 %
270 %
271 %
272 %
273 %
274 %
275 %
276 %
277 %
278 %
279 %
280 %
281 %
282 %
283 %
284 %
285 %
286 %
287 %
288 %
289 %
290 %
291 %
292 %
293 %
294 %
295 %
296 %
297 %
298 %
299 %
300 %
301 %
302 %
303 %
304 %
305 %
306 %
307 %
308 %
309 %
310 %
311 %
312 %
313 %
314 %
315 %
316 %
317 %
318 %
319 %
320 %
321 %
322 %
323 %
324 %
325 %
326 %
327 %
328 %
329 %
330 %
331 %
332 %
333 %
334 %
335 %
336 %
337 %
338 %
339 %
340 %
341 %
342 %
343 %
344 %
345 %
346 %
347 %
348 %
349 %
350 %
351 %
352 %
353 %
354 %
355 %
356 %
357 %
358 %
359 %
360 %
361 %
362 %
363 %
364 %
365 %
366 %
367 %
368 %
369 %
370 %
371 %
372 %
373 %
374 %
375 %
376 %
377 %
378 %
379 %
380 %
381 %
382 %
383 %
384 %
385 %
386 %
387 %
388 %
389 %
390 %
391 %
392 %
393 %
394 %
395 %
396 %
397 %
398 %
399 %
400 %
401 %
402 %
403 %
404 %
405 %
406 %
407 %
408 %
409 %
410 %
411 %
412 %
413 %
414 %
415 %
416 %
417 %
418 %
419 %
420 %
421 %
422 %
423 %
424 %
425 %
426 %
427 %
428 %
429 %
430 %
431 %
432 %
433 %
434 %
435 %
436 %
437 %
438 %
439 %
440 %
441 %
442 %
443 %
444 %
445 %
446 %
447 %
448 %
449 %
450 %
451 %
452 %
453 %
454 %
455 %
456 %
457 %
458 %
459 %
460 %
461 %
462 %
463 %
464 %
465 %
466 %
467 %
468 %
469 %
470 %
471 %
472 %
473 %
474 %
475 %
476 %
477 %
478 %
479 %
480 %
481 %
482 %
483 %
484 %
485 %
486 %
487 %
488 %
489 %
490 %
491 %
492 %
493 %
494 %
495 %
496 %
497 %
498 %
499 %
500 %
501 %
502 %
503 %
504 %
505 %
506 %
507 %
508 %
509 %
510 %
511 %
512 %
513 %
514 %
515 %
516 %
517 %
518 %
519 %
520 %
521 %
522 %
523 %
524 %
525 %
526 %
527 %
528 %
529 %
530 %
531 %
532 %
533 %
534 %
535 %
536 %
537 %
538 %
539 %
540 %
541 %
542 %
543 %
544 %
545 %
546 %
547 %
548 %
549 %
550 %
551 %
552 %
553 %
554 %
555 %
556 %
557 %
558 %
559 %
560 %
561 %
562 %
563 %
564 %
565 %
566 %
567 %
568 %
569 %
570 %
571 %
572 %
573 %
574 %
575 %
576 %
577 %
578 %
579 %
580 %
581 %
582 %
583 %
584 %
585 %
586 %
587 %
588 %
589 %
590 %
591 %
592 %
593 %
594 %
595 %
596 %
597 %
598 %
599 %
600 %
601 %
602 %
603 %
604 %
605 %
606 %
607 %
608 %
609 %
610 %
611 %
612 %
613 %
614 %
615 %
616 %
617 %
618 %
619 %
620 %
621 %
622 %
623 %
624 %
625 %
626 %
627 %
628 %
629 %
630 %
631 %
632 %
633 %
634 %
635 %
636 %
637 %
638 %
639 %
640 %
641 %
642 %
643 %
644 %
645 %
646 %
647 %
648 %
649 %
650 %
651 %
652 %
653 %
654 %
655 %
656 %
657 %
658 %
659 %
660 %
661 %
662 %
663 %
664 %
665 %
666 %
667 %
668 %
669 %
670 %
671 %
672 %
673 %
674 %
675 %
676 %
677 %
678 %
679 %
680 %
681 %
682 %
683 %
684 %
685 %
686 %
687 %
688 %
689 %
690 %
691 %
692 %
693 %
694 %
695 %
696 %
697 %
698 %
699 %
700 %
701 %
702 %
703 %
704 %
705 %
706 %
707 %
708 %
709 %
710 %
711 %
712 %
713 %
714 %
715 %
716 %
717 %
718 %
719 %
720 %
721 %
722 %
723 %
724 %
725 %
726 %
727 %
728 %
729 %
730 %
731 %
732 %
733 %
734 %
735 %
736 %
737 %
738 %
739 %
740 %
741 %
742 %
743 %
744 %
745 %
746 %
747 %
748 %
749 %
750 %
751 %
752 %
753 %
754 %
755 %
756 %
757 %
758 %
759 %
760 %
761 %
762 %
763 %
764 %
765 %
766 %
767 %
768 %
769 %
770 %
771 %
772 %
773 %
774 %
775 %
776 %
777 %
778 %
779 %
780 %
781 %
782 %
783 %
784 %
785 %
786 %
787 %
788 %
789 %
790 %
791 %
792 %
793 %
794 %
795 %
796 %
797 %
798 %
799 %
800 %
801 %
802 %
803 %
804 %
805 %
806 %
807 %
808 %
809 %
810 %
811 %
812 %
813 %
814 %
815 %
816 %
817 %
818 %
819 %
820 %
821 %
822 %
823 %
824 %
825 %
826 %
827 %
828 %
829 %
830 %
831 %
832 %
833 %
834 %
835 %
836 %
837 %
838 %
839 %
840 %
841 %
842 %
843 %
844 %
845 %
846 %
847 %
848 %
849 %
850 %
851 %
852 %
853 %
854 %
855 %
856 %
857 %
858 %
859 %
860 %
861 %
862 %
863 %
864 %
865 %
866 %
867 %
868 %
869 %
870 %
871 %
872 %
873 %
874 %
875 %
876 %
877 %
878 %
879 %
880 %
881 %
882 %
883 %
884 %
885 %
886 %
887 %
888 %
889 %
890 %
891 %
892 %
893 %
894 %
895 %
896 %
897 %
898 %
899 %
900 %
901 %
902 %
903 %
904 %
905 %
906 %
907 %
908 %
909 %
910 %
911 %
912 %
913 %
914 %
915 %
916 %
917 %
918 %
919 %
920 %
921 %
922 %
923 %
924 %
925 %
926 %
927 %
928 %
929 %
930 %
931 %
932 %
933 %
934 %
935 %
936 %
937 %
938 %
939 %
940 %
941 %
942 %
943 %
944 %
945 %
946 %
947 %
948 %
949 %
950 %
951 %
952 %
953 %
954 %
955 %
956 %
957 %
958 %
959 %
960 %
961 %
962 %
963 %
964 %
965 %
966 %
967 %
968 %
969 %
970 %
971 %
972 %
973 %
974 %
975 %
976 %
977 %
978 %
979 %
980 %
981 %
982 %
983 %
984 %
985 %
986 %
987 %
988 %
989 %
990 %
991 %
992 %
993 %
994 %
995 %
996 %
997 %
998 %
999 %
1000 %

```

Plan

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

Trouver de l'aide

- Vous devez devenir très vite autonome

- L'aide en ligne est votre amie

⇒ <https://fr.mathworks.com/help/matlab/>

- Il existe de nombreux codes disponibles

⇒ <https://fr.mathworks.com/matlabcentral/fileexchange/>

⇒ <http://www.varenes-ecofin.com/>

Affectation dans un cadre de typage faible

- Les variables sont généralement

⇒ un scalaire

⇒ une matrice

⇒ une chaîne de caractères

- L'affectation 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

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

⇒ une matrice

```
1 myMx = myMx + myMx; % addition
2 myMx = myMx - myMx; % soustraction
3 myMx = [ 1 2 ] * [ 3; 4 ]; % multiplication
4 myMx = myMx .* myMx; % produit d'Hadamard
5 myMx = myMx ^ 2; % puissance d'une matrice
6 myMx = myMx .^ 2; % puissance element par element
7 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>

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

⇒ Opérateurs logiques

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

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

⇒ Opérateurs arithmétiques

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

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

2 Initialiser le générateur de nombre aléatoire

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

3 Générer une séquence aléatoire à partir d'une distribution

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

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

Les graphiques

4) ->

3) Sélectionner "New Figure" de préférence

1) Clic double sur "data"

2) Le tableur s'ouvre, puis sélectionner la colonne de "data"

6) Le code utilisé s'affiche ->

5) Une fenêtre s'ouvre ->

```

>> data = randn(1000,1);
>> figure; plot(data);
>>
  
```

Les graphiques

- Tracer une série univariée est simple

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

- Tracer deux séries sur un même axe vertical est simple

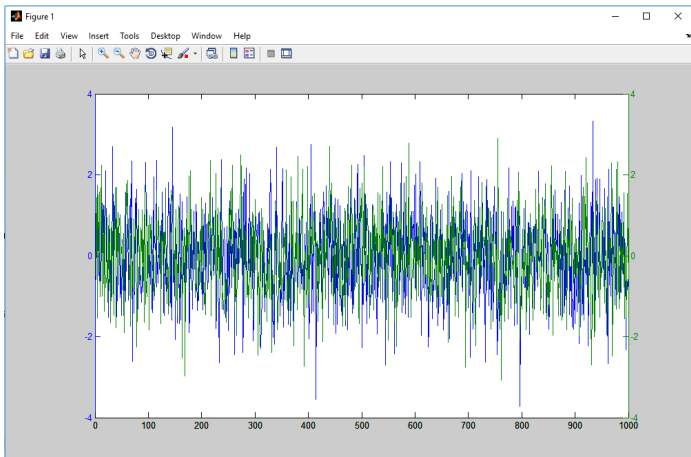
```
1 figure ;  
2 plot(data); % avec data un vecteur nx2 ou 2xn
```

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

```
1 figure ;  
2 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
```

⇒ Bien d'autres instructions de contrôle existent

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

Créer un nouveau script

The screenshot shows the MATLAB R2014a environment. The Command Window contains the following code:

```

>> lambda0 = 0.5;
>> lambda0 = 0.5;
lambda0 =
    0.5000
>>
  
```

The Workspace window displays the following variables:

Name	Value	Min	Max	Mean
lambda0	5000	0.5000	5000	5000
M	300	300	300	300
msp	0	0	0	0
N	0	0	0	0
nmax	100	100	100	100
optfunc	f_optfunc			
opt	1	1	1	1
T	3000	3000	3000	3000

The Command History window shows the following execution steps:

```

clear all;
h = RandStrenms('mvg05071', 'rand', 2); Rand...
%% Set global parameters for simulation
nmax = 100; % number of...
M = 0; % preallocation ...
T = 3000; % number of...
n = 300; % number of ...
%% Set parameters
options = optimset('Display','off');
options = optimset(options, 'LargeScale','off');
% Jump parameters
lambda0 = 0.5; % Jump rate 10...
msp = 0.1-0.6; % Jump siz...
nmax = 100; % Jump siz...
lambda0 = 0.5;
lambda0 = 0.5
  
```

Main script

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

```
1 % myMainScript.m permet de blablaba ...
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

⇒ 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 : $V_t = V_0 e^{-\kappa t} + \omega(1 - e^{-\kappa t}) + \gamma e^{-\kappa t} \int_0^t e^{\kappa t} dW$

```

1 % OrnsteinUhlenbeckDE.m generates an OU model
2 %
3 % M: the number of observations
4 % V0: the process at t = 0
5 % kappa: the ajustement speed of the process
6 % omega: the asymptotic mean of the process
7 % gamma: the volatility of the process
8 %
9 %           Gilles de Truchis , December 2016
10 %
11 function [ou,dW] = OrnsteinUhlenbeckDE (M,V0,kappa ,omega ,gamma)
12
13 dt = 1/M;
14 t = (0:dt:1)';
15 dW = randn(M,1);
16
17 nex = exp(-kappa*t);
18 pex = exp(2*kappa*t);
19 dex = diff(pex-1);
20
21 ou = V0*nex + omega*(1-nex) + ... % permet un saut de ligne
22     gamma*nex.*cumsum([0; sqrt(dex).*dW])/sqrt(2*kappa);
23
24 ou(1) = [];

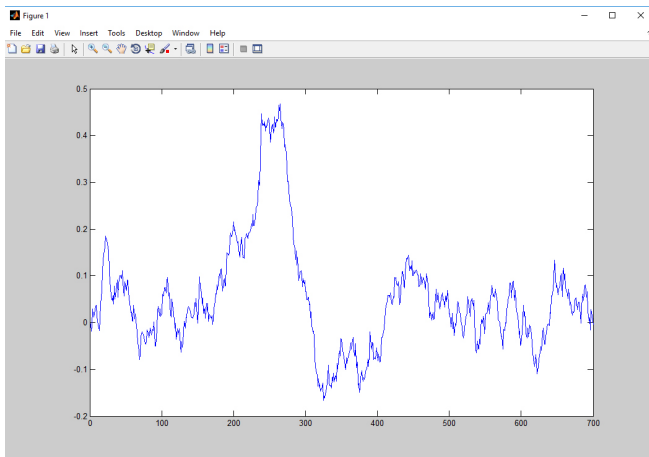
```

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

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

Estimation par OLS

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

```
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 %% OLS estimation
16
17 [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 $-\ell_n(\cdot)$ 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

```
1 ...
2
3 % Core of the main script
4
5 %% Simulation of the linear regression model
6 n = 2000;
7 Mdl = garch('Constant',0.01,'GARCH',0.7,'ARCH',0.2);
8 [v,r] = simulate(Mdl,n);
9
10 %% non-linear constrained MLE estimation
11 optcon = optimset('fmincon');
12 optcon = optimset(optcon,'Algorithm','sqp',...
13 'LargeScale','off','Display','off');
14
15 A = -eye(3); % A*x <= ub
16 ub = [0;0;0];
17
18 [thetaML,~,~,~,~,hessian] = fmincon('garchMLEDE',...
19 [0.1,0.2,0.4]',A,ub,[],[],[],[],[],optcon,r,std(r));
20
21 stdNLML = sqrt(diag(inv(hessian)));
22 tstatNLML = thetaML./stdNLML;
```

Plan

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

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

```
1 % mainAR1.m performs an OLS estimation of an AR1
2 %
3 %                               September 2017
4 %
5 clc;
6 clear all;
7
8 addpath('subscripts');
9
10 data = xlsread('fxdata.xlsx',1);
11
12 %% Estimation of an AR1 process
13 n = size(data,1);
14 fx = data(2:end,2);
15 fxlag = data(1:end-1,2);
16
17 [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)

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

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

```
1 %% Radom walk ?
2 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, $s_{\hat{\rho}}$ est supérieur à -8.1
- ⇒ 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
<i>Case 1</i>								
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
<i>Case 2</i>								
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
<i>Case 4</i>								
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

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

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 $(1, -1)'$

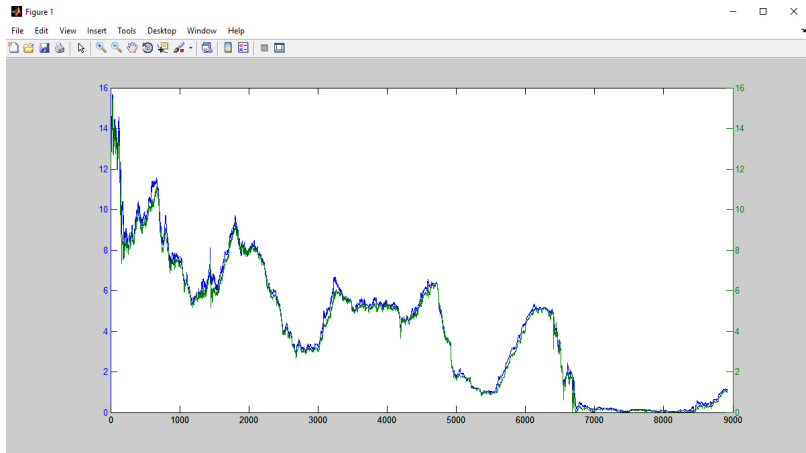
- On cherchera donc à estimer $i_{m,t} = \beta i_{1,t} + z_t$

Cointégration et structure par terme : les données

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

```
1 % mainEG87.m term structure cointegration analysis
2 %
3 %           September 2017
4 %
5 clc;
6 clear all;
7
8 addpath('subscripts');
9
10 data = xlsread('idata.xlsx',1);
11
12 %% Data analysis
13 n = size(data,1);
14
15 x = data(2:end,1);
16 xlag = data(1:end-1,1); % 3 month interest rate
17
18 y = data(2:end,2);
19 ylag = data(1:end-1,2); % 6 month interest rate
20
21 figure;
22 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

```
1 %% Radom walk ?
2 [by, ty, stdy, ~] = olsDE(y,ylag); % beta = 0.9995942451
3 [bx, tx, stdx, ~] = olsDE(x,xlag); % beta = 0.9996227180
4
5 rwy = n*(by - 1); % rwStat = -3.75536802
6 rwx = n*(bx - 1); % rwStat = -3.59486769
7
8 %% OLS estimation of long run equilibrium
9 [bci, tci, stdci, yhat] = olsDE(y,[ones(n-1,1) x]);
10 % beta = 1.0304789539
11
12 z = y - yhat;
13 figure;
14 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)

$$s_{\hat{\rho}} = (n-1)(\hat{\rho}-1) \xrightarrow{d} \left[\frac{1}{2} \left((1 \quad -h'_2) \cdot \mathbf{W}(1) \mathbf{W}(1)' \cdot \begin{pmatrix} 1 \\ -h_2 \end{pmatrix} \right) - h_1 \mathbf{W}(1)' \begin{pmatrix} 1 \\ -h_2 \end{pmatrix} - \frac{1}{2} (1 + h'_2 h_2) \right] H_n^{-1}$$

- La distribution $\mathcal{L}(\hat{\rho})$ est tabulée dans le tableau qui suit
- $\mathcal{L}(\hat{\rho})$ 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 $s_{\hat{\rho}}$ pour $n = 8923$

```

1 %% As both series are I(1) regression might be spurious
2 zhat = z(2:end);
3 zhatlag = z(1:end-1);
4 [bz, tz, stdz, ~] = olsDE(zhat, zhatlag); % beta = 0.9654598939
5 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 $I(1)$, $s_{\hat{\rho}}$ est supérieur à -20.5

⇒ 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
		<i>Case 1</i>						
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
<i>Case 2</i>								
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
<i>Case 3</i>								
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 $\alpha_2 > 0$

```

1 %% ECM representation
2 dly = y - ylag; dy = dly(2:end); dylag = dly(1:end-1);
3 dlx = x - xlag; dx = dlx(2:end); dxlag = dlx(1:end-1);
4
5 [becmy, tecmy, ~, ~] = olsDE(dy, [ones(size(dy)) zhatlag dylag dxlag]);
6 [becmx, tecmx, ~, ~] = olsDE(dx, [ones(size(dx)) zhatlag dylag dxlag]);
7
8
9      y          x
10 omega    -0,0013  -0,0012
11 alpha     -0,0070   0,0226
12 gamma1    -0,0717   0,0914
13 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
- ⇒ 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

- $\Delta X_t = \sum_{j=1}^p \Omega_j \Delta X_{t-j} + \omega_t$

- $X_{t-1} = \sum_{j=1}^p \Psi_j \Delta X_{t-j} + \psi_t$

```

1 [n,k] = size(x);
2
3 dx = x(2:end,:) - x(1:end-1,:); % DX(t)
4 dx = detregDE(dx,c); % Remove deterministic components Omega0
5 dxlag = lagmatrix(dx,1:p); % Create DX(t-1), ...,DX(t-p)
6
7 dx(1:p,:) = []; % Ajust sample size for DX(t)
8 dxlag(1:p,:) = []; % Ajust sample size for DX(t-1), ...,DX(t-p)
9
10 r1hat = (dxlag\dx);
11 ome = dx - dxlag*r1hat; % First auxiliary regression residuals
12
13 x = detregDE(x,c); % Remove deterministic components Psi0
14 x = lagmatrix(x,p); % Create X(t-p) ...
15 x(1:p+1,:) = []; % ... to adjust the sample size of X(t)
16
17 r2hat = (dxlag\x);
18 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

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

```

1 vcvo0 = 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(vcvo0)*(vcvpo');
6
7 [evec,eval] = eig(vcv); % Raw eigen vectors and eigen values
8 % Normalized eigenvectors (Cholesky(X) ~ sqrt(X))
9 nvec = evec*inv(chol(evec'*vcvpp*evec));
10
11 nvec = nvec'; % Eigenvec by column but sort command by rows
12 % Sort eigen values in descending order
13 [evsort, evindex] = sort(diag(eval),'descend');
14 nsevec = nvec(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 : $\mathcal{LR}_g^{(\tau)} = 2(\mathcal{L}_{H_1}^* - \mathcal{L}_{H_0}^*) = -n \sum_{i=r+1}^k \log(1 - \hat{\lambda}_i)$

- test $\lambda - \max$: $\mathcal{LR}_g^{(\lambda)} = 2(\mathcal{L}_{H_1}^* - \mathcal{L}_{H_0}^*) = -n \log(1 - \hat{\lambda}_{r+1})$

```

1 % Compute the trace and max eigenvalue statistics
2 lrt = zeros(k,1); lrm = zeros(k,1); % init tests statistics
3 cvt = zeros(k,3); cvm = zeros(k,3); % init critical values
4 n2 = size(psi,1);
5
6 iota = ones(k,1); % init a k x 1 vector of ones
7 ilam = log(iota-evsort); % Generates the (1 - lambda(i)) vector
8
9 for i = 1:k
10     ilamtmp = ilam;
11     ilamtmp(1:i-1,:) = []; % Adjust (1 - lambda(i)) vector size
12     lrt(i,1) = -n2*sum(ilamtmp); % Trace test statistic
13     lrm(i,1) = -n2*log(1-evsort(i,1)); % Eigmax test statistic
14     cvt(i,:) = c_sjt(k-i+1,c-1); % Trace test critical values
15     cvm(i,:) = c_sja(k-i+1,c-1); % Eigmax test critical values
16
17     if lrt(i,1) > cvt(i,2)
18         r = i; % Gives the cointegration rank
19     end
20 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 $m = 2, 3, 4$

```

1 % mainJ91.m term structure cointegration analysis
2 %
3 %                               September 2017
4 %
5 clc;
6 clear all;
7
8 addpath('subscripts');
9
10 data = xlsread('idata.xlsx',1);
11
12 %% Computation of trace and lambda-max Johansen's tests
13
14 n = size(data,1);
15 x = data(:,[2 3 4]);
16 x = log(x);
17 p = 2;
18
19 [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

- 1 Introduction à MATLAB
- 2 Manipulations de bases
- 3 Principes d'estimation
- 4 Applications du Chapitre 1
- 5 Applications du Chapitre 2
- 6 Applications du Chapitre 3

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

```

1 % mainLM.m long memory analysis and estimation
2 %
3 %                               September 2017
4 %
5 clc;
6 clear all;
7
8 addpath('subscripts');
9 s = RandStream('mcg16807','Seed',6);
10 RandStream.setGlobalStream(s);
11
12 %% Generate a fractional white noise
13 n = 1000;
14 delta = 0.4;
15 e = randn(n,1);
16 % integrate (if -delta) / differentiate (if delta) fractionnally
17 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

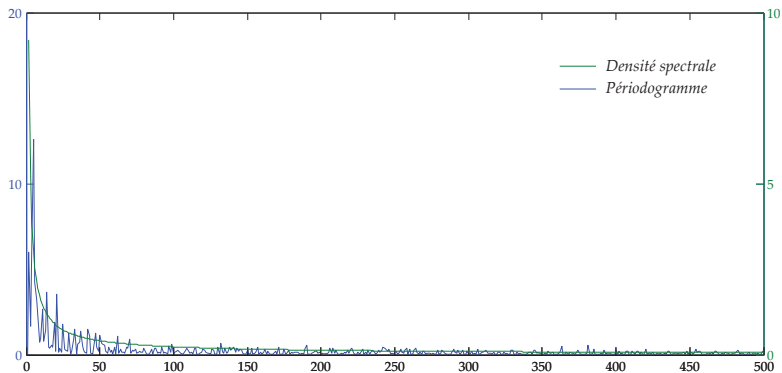
$$\hat{I}_x(\lambda) = |\omega_x(\lambda)|^2 \text{ avec } \omega_x(\lambda) = \frac{1}{\sqrt{2\pi n}} \sum_{t=1}^n x_t e^{-it\lambda_j} \text{ et } \lambda_j = \frac{2\pi j}{n}$$

```

1 %% Periodogramm and spectral density comparison
2 j = fix(n/2);
3 t = (0:1:n-1)';
4 lambda = 2*pi*t/n;
5
6 wx = (2*pi*n)^(-1/2)*fft(x).*exp(1i*lambda);
7 Ix = wx(1:j).*conj(wx(1:j));
8
9 fx = 1/(2*pi)*abs(lambda(1:j)).^(-2*delta);
10
11 figure;
12 plotyy(1:j, Ix, 1:j, fx);

```

Mémoire longue, densité spectrale 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**

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

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

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

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

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**

```
1 function [r] = whittleDE(d,x,m)
2 % WHITTLE.M computes the local Whittle likelihood
3 %
4 %           INPUT   x: data (n*1 vector)
5 %                   m: truncation number
6 %                   d: parameter value
7 %
8 %           October 2017
9 %
10
11 n = size(x,1);
12 t = (0:1:n-1)';
13 lambda = 2*pi*t/n;
14 wx = (2*pi*n)^(-1/2)*fft(x).*exp(1i*lambda);
15 lambda = lambda(2:m+1);
16 wx = wx(2:m+1);
17 Ix = wx.*conj(wx);
18
19 g = mean((lambda.^(2*d)).*Ix);
20 r = log(g) - 2*d*mean(log(lambda));
```

Mémoire longue et estimateur Whittle paramétrique

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

```

1 function [ll , Ixx , fxxb] = fbwhittleDE(param , data , p , q)
2 % fbwhittle.m computes the full band Whittle likelihood
   estimator
3 %
4 %
5 %
6 %
7 %
8 %
9 %
10 %
11 %
12 %
13 %
14 %
15 %
16
17 n = size(data , 1);
18 x = data;
19 delta = param(1);
20 if p == 0; phi = 0; else phi = param(2:1+p); end;
21 if q == 0; theta = 0; else theta = param(2+p:1+p+q); end;
22 sigma = param(end);

```

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

Mémoire longue et estimateur Whittle paramétrique

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

```

1 % Trend & Fourier frequencies
2 t = (0:1:n-1)';
3 lambda = 2*pi*t/n;
4
5 % Fourier transform and Periodogram
6 wx = (2*pi*n)^(-1/2)*conj(fft(conj(x))).*exp(1i*lambda);
7 Ixx = wx.*conj(wx);
8
9 % Fractional filter
10 fd = abs(1-exp(-1i*lambda)).^(-2*delta);
11
12 % ARMA filter
13 ei = ones(n,1);
14
15 MA = ei + exp(-1i*lambda*(1:1:length(theta)))*theta;
16 AR = ei + exp(-1i*lambda*(1:1:length(phi)))*-phi;
17
18 % Spectral density
19 fxxb = (sigma^2/(2*pi))*fd.*((abs(MA).^2)./(abs(AR).^2));
20
21 % Whittle approximation to the likelihood
22 fxxb=fxxb(2:end);
23 Ixx=Ixx(2:end);
24
25 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

```

1 %% Local Whittle estimation of a fractional white noise
2
3 optbnd = optimset('fminbnd');
4
5 m = fix(n^0.8);
6 lwhat = fminbnd('whittleDE', -1, 3, optbnd, x, m);
7
8 %% Full band Whittle estimation of a fractional white noise
9 optunc = optimset('fminunc');
10 optunc = optimset(optunc, 'LargeScale', 'off', 'MaxFunEvals', 5000, '
    FunValCheck', 'off', 'Display', 'off');
11
12 % specification : ARFIMA(0,d,0)
13 swhat = fminunc('fbwhittleDE', [0.3, 0.1], optunc, x, 0, 0);
14
15 % misspecified : ARFIMA(1,d,0)
16 mwhat = fminunc('fbwhittleDE', [0.3, 0.5, 0.1], optunc, x, 1, 0);

```

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

- 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.