Wojciech W. Charemza[*)] and Svetlana Makarova[**)]

Blini

Version 1.03

Collection of GAUSS procedures for linear and bilinear unit root analysis

[*)] University of Leicester, U.K.
wch@le.ac.uk

**) European University at St. Petersburg, Russia
makarova@eu.spb.ru

Address for correspondence:

Department of Economics
University of Leicester
University Road
Leicester LE1 7RH
U.K.

Preliminary version: not finished and not fully tested

January 2002

Financial support of the INTAS99 – 00472 Project *Nonlinear Structural VAR Modelling of East European Economies* is gratefully acknowledged. All usual disclaimers apply.

Introduction

BLINI[1] is a collection of procedures written in GAUSS, which accompany Charemza, Lifshits and Makarova (2001) paper. The procedure can be used in empirical analysis of the linear and blinear unit root processes. Currently there are two files of the procedures, which does not contain a proper GAUSS library. They should be simply #INCLUDE'd at the beginning of a file containing a GAUSS program.

The first file, PELMENI[2], contains simple and fast procedures, suitable for Monte Carlo analysis and other tasks requiring heavy computations. The procedures included in PELMENI are for simulation bilinear processes (BLIN), simple Student-t test for a linear regression model (TTEST), simple Dickey-Fuller test (DF), Leybourne (1995) $DF_{\max}$ test, *b*-test for testing bilinearity of a unit root process (BTEST), b_{max} test, which is a Leybourne-type bilinearity test (BLTEST), and the kurtosis-corrected equivalents to the *b*-test and b_{max} tests (BTESTK and BLTESTK).

The second file, BIGOS[3], is a collection of slower, but more elaborated, procedures suitable for a thorough empirical analysis. They are mainly the augmented tests where the augmentations are selected accordoing to the general to specific methodology. They are: the augmented Dickey-Fuller test, ADFH, augmented $DF_{\max}$ test (LEYBH), *KPSS* test with an automatic selection of autocorrelation lags (KPSSH) and augmented *b* and $b_{\max}$ tests (BTESTH and BTLEH). Moreover the file contains the augmented *b* and $b_{\max}$ tests in which the kurtosis correction has been imposed (BTESTHK and BTLEHK). It also includes procedure for detecting structural breaks (additive and innovative outliers) in unit root

---

1 Take 1 egg and 3 teaspoons of sugar and mix them together. Add a little milk and a tablespoon of flour and stir. Dilute with more milk, add a tablespoon of flower, mix and make a pancake. Then bake. Serve with herrings, slightly smoked fish or caviar and a glass of ice-cold *Stolichna*.

2 Make a dense pastry. Roll in a thin layer. Cut into small squares. Put a special stuffing made with. low-fat beef and pork in the middle of each square. Roll up like raviolis and freeze. Boil straight from the freezer. Serve with a proper Russian *Smirnoff* (beware of Western imitations!)

3 Take good quality sauerkraut and cabbage, add wild mushrooms and browned diced beef, pork and continental sausages. Mix well, cook very slowly for about 4 hours. Leave aside for at least two days, then reheat and serve with a glass of a Polish herbal vodka (*Soplica*, *Zubrówka* or *Jarzębiak*).

series. They are the Perron additive and innovative outliers procedures which assume that a breakpoint is known (PERR_AO and PERR_IO) and analobous procedures which sequentially detect and test a structural break in case where the breakpoint is not known (AO_TEST and IO_test).

The procedures have been tested using GAUSS 3.2.15 of June 11, 1996. It appears that they are performing well in GAUSS for Windows under Windows 2000, kernel review 3.5.14, GUV review 3.4.11. No further testing has been done.

Pelmeni

proc (1) = blin(e,a,b);

This is a procedure for simulation of a bilinear process of the type: y(t) =(a + b*e(t-1))*y(t-1) + e(t).

inputs: e: vector of random numbers

a,b: coefficients

output: y(t), simulated vector of data, n x 1

proc (1) = ttest(y,x,d);

This is a procedure for computing a single variable Student-t statistic in a static regression model.

inputs: y: a series of data, n x 1

x: a series of data, n x 1

d if 1: constant in b-test is included, otherwise it is not

output: Student-t statistic for parameter on x

proc (1) = df(y,d);

This is a procedure for computing a simple Dickey-Fuller test.

inputs: y: a series of data, n x 1

d: if 1: constant in the Dickey-Fuller equation is included, otherwise it is not

output: Student-t Dickey-Fuller statistic

proc (1) = leyb(y,d);

This is a procedure for computing the Leybourne DFmax test, see Leybourne (1995), Maddala and Kim(1998), p. 111.

inputs: y: a series of data, n x 1

d: if 1: constant in both forward and reverse Dickey-Fuller equation is included, otherwise it is not

output: DFmax statistic

proc (1) = btest(y,d);

This is a procedure for computing a b-test, that is, regressing Δy(t) on e(t-1)*y(t-1) and computing the Student-t test.

inputs: y: a series of data, n x 1
d: if 1: constant is included
if 0: constant is not included

output: Student-t statistic for b parameter

proc bltest(y,d);

This is a procedure for computing the Leybourne-type b-test. It computes the b-test statistics for the forward and backward-ordered data and taking their maximum (see Charemza, Lifshits and Makarova (2001)).

inputs: y a series of data, n x 1
d if 1: constant is included
if 0: constant is not included

output: b-test max statistic

proc (2) = btestk(y,d,g);

This is a procedure for computing the b-test adjusted for kurtosis (see Charemza, Lifshits and Makarova (2001).

inputs: y: a series of data, n x 1
d: if 1: constant is included
if 0: constant is not included
if 9: returns are demeaned
g: constant for kurtosis-adjustment (0.5 is currentlyrecommended)

outputs: Student-t statistic for the b parameter
Information about the kurtosis-adjustment:
0: no adjustment was made (kurtosis < g)
1: adjustment was made (kurtosis > g)

proc(2) = bltestk(y,d,g);

This is a procedure for computing a Leybourne-type b-test and adjusting the t-statistic for kurtosis (see Charemza, Lifshits and Makarova (2001)).

inputs: y: a series of data, n x 1

d: if 1: constant is included

if 0: constant is not included

if 9: returns are demeaned

g : constant for kurtosis-adjustment (0.5 is currently recommended)

outputs: b-test max statistic

Information about the kurtosis-adjustment:

0 : no adjustment was made (kurtosis < 0.5)

1 : adjustment was made (kurtosis > 0.5)

proc (1) = btestm(y,d);

This is a procedure for computing a b-test on partially demeaned data that is, regressing Δy_t on $\Delta z_{t-1} y_{t-1}$, where Δz_t is the demeaned series of Δy_t .

inputs: y a series of data, n x 1

d if 1: constant is included

if 0: constant is not included

output: Student-t statistic for b parameter

proc (1) = btestd(y,d);

This is a procedure for computing a b-test on partially demeaned data that is, regressing delta[y(t)] on delta[z(t-1)]*z(t-1)., where z_t is a series of cumulated demeaned series of Δy_t .

inputs: y a series of data, n x 1

d if 1: constant is included

if 0: constant is not included

output: Student-t statistic for b parameter

proc (1) = bltestm(y,d);

This is a procedure for computing a Leybourne-type partially demeaned b-test

inputs: y a series of data, n x 1
d if 1: constant is included
if 0: constant is not included

output: b-test max statistic

proc (1) = bltestd(y,d);

This is a procedure for computing a Leybourne-type fully demeaned b-test

inputs: y a series of data, n x 1
d if 1: constant is included
if 0: constant is not included

output: b-test max statistic

Bigos

proc (3) = adfh(y, d, k, crit, pr);

This is a procedure for computing Augmented Dickey-Fuller test with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations.

inputs: y: a series of data, n x 1

d: if -1: no constant and no time trend in the Dickey-Fuller equation is included
if 0: constant in the Dickey-Fuller equation is included
if 1: constant and linear time trend in the Dickey-Fuller equation is included
if 2: constant, linear and quadratic time trend in the Dickey-Fuller equation is included

k: maximum lag length

crit: critical value for selection

pr if 1: warning about the significance of the deterministic part is printed

outputs: t: Student-t Dickey-Fuller statistic

kv: vector which identifies particular significant lags in augmentation

tp: vector which identifies insignificant of constant and time trend in the Dickey-Fuller equation; returns -99 if constant and time trends are significant at the 'crit' level or if they are not included.

proc (4) = leybh(y, d, k, crit, pr);

This is a procedure for computing Augmented DFmax (Leybourne) test with the selection of lags similar to Hall (1994) general to specific method (see Leybourne (1995), Maddala and Kim (1998), p.111). The selection of lags is according to the 'significant' Student-t statistics on augmentations.

inputs: y: a series of data, n x 1

d: if -1: no constant and no time trend in the Dickey-Fuller equation is included
if 0: constant in the Dickey-Fuller equation is included
if 1: constant and linear time trend in the Dickey-Fuller equation is included
if 2: constant, linear and quadratic time trend in the Dickey-Fuller equation is included

k: maximum lag length

	crit:	critical value for selection
	pr	if 1: warning about the significance of the deterministic part is printed
outputs:	t:	Student-t Dickey-Fuller statistic
	kv:	vector which identifies particular significant lags in augmentation
	tp:	vector which identifies insignificant of constant and time trend in the Dickey-Fuller equation; returns -99 if constant and time trends are significant at the 'crit' level or if they are not included.
	ind1:	"forward", if DFmax statistic was achieved in forward regression "backward", if DFmax statistic was achieved in backward regression

proc(2)=kpssh(y,l,d);

This is a modification of the KPSS procedure by David Rapach of 27 May 1996 as posted in the GAUSS archive (see Isaac and Rapach (1996)). It computes the KPSS statistics for lags from 0 to p and returns the highest statistic. See Kwiatkowski *at al.* (1992), Maddala and Kim (1998), pp. 120-122.

inputs:	y	vector of data, n x 1
	l	maximum order of autocorrelation
	d	if 0: test with a constant if 1: test with a constant and a linear trend
outputs:	kpstat:	maximum KPSS statistic
	lag:	scalar denoting the lag length associated with the maximum KPSS statistic

proc (3) = btesth(y, d, k, crit, pr);

This is a procedure for computing an augmented b-test that is, regressing delta[y(t)] on e(t-1)*y(t-1) and constant, linear and quadratic trends (if selected), and augmentations: delta[y(t-1)], delta[y(t-2)] etc. with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations. See Charemza, Lifshits and Makarova (2001)).

inputs:	y:	vector of data, n x 1
	d:	if -1: no constant and no time trend is included if 0: constant is included if 1: constant and linear time trend is included

if 2: constant, linear and quadratic time trend are included

k: maximum lag length

crit: critical value for selection

pr: if 1: warning about the significance of the deterministic part is printed

outputs: t: Student-t statistic

kv: vector which identifies particular lags in augmentation

tp: vector which identifies insignificant of constant and time trend; returns -99 if constant and time trends are significant at 'crit' level or if they are not included.

proc (4) = btleh(y, d, k, crit, pr);

This is a procedure for computing an augmented Leybourne style b-test (b-max test) that is, it computes the augmented b-test statistics (se the procedure BTESTH above) for the forward and backward-ordered data and takes their maximum (see Charemza, Lifshits and Makarova (2001)). The selection of lags is according to the 'significant' Student-t statistics on augmentations. See Charemza, Lifshits and Makarova (2001)).

inputs: y: vector of data, n x 1

d: if -1: no constant and no time trend is included

if 0: constant is included

if 1: constant and linear time trend is included

if 2: constant, linear and quadratic time trend are included

k: maximum lag length

crit: critical value for selection

pr: if 1: warning about the significance of the deterministic part is printed

outputs: t: Student-t statistic

kv: vector which identifies particular lags in augmentation

tp: vector which identifies insignificant of constant and time trend; returns -99 if constant and time trends are significant at 'crit' level or if they are not included.

ind1: "forward", if b-max statistic. was achieved in forward regression

"backward", if b-max statistic was achieved in backward regression

proc (4) = btesthk(y, d, k, crit, pr, g);

This is a procedure for computing an augmented and kurtosis corrected b-test that is, regressing delta[y(t)] on e(t-1)*y(t-1) and constant, linear and quadratic trends (if selected), and augmentations: delta[y(t-1)], delta[y(t-2)] etc. with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations. See Charemza, Lifshits and Makarova (2001)).

inputs:	y:	vector of data, n x 1
	d:	if -1: no constant and no time trend is included
		if 0: constant is included
		if 1: constant and linear time trend is included
		if 2: constant, linear and quadratic time trend are included
	k:	maximum lag length
	crit:	critical value for selection
	pr:	if 1: warning about the significance of the deterministic part is printed
	g:	constant for kurtosis adjustment (0.5 is currently recommended)
outputs:	t:	Student-t statistic
	kv:	vector which identifies particular lags in augmentation
	tp:	vector which identifies insignificant of constant and time trend; returns -99 if constant and time trends are significant at 'crit' level or if they are not included.
	Ik:	scalar which identifies whether the kurtosis correction was binding.
		if 0: no binding correction
		if 1: the correction is binding

proc (4) = btlehk(y, d, k, crit, pr, g);

This is a procedure for computing an augmented and kurtosis corrected b-test that is, regressing delta[y(t)] on e(t-1)*y(t-1) and constant, linear and quadratic trends (if selected), and augmentations: delta[y(t-1)], delta[y(t-2)] etc. with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations. See Charemza, Lifshits and Makarova (2001)).

inputs: y: vector of data, n x 1

d: if -1: no constant and no time trend is included
if 0: constant is included
if 1: constant and linear time trend is included
if 2: constant, linear and quadratic time trend are included

k: maximum lag length

crit: critical value for selection

pr: if 1: warning about the significance of the deterministic part is printed

g: constant for kurtosis adjustment (0.5 is currently recommended)

outputs: t: Student-t statistic

kv: vector which identifies particular lags in augmentation

tp: vector which identifies insignificant of constant and time trend; returns -99 if constant and time trends are significant at 'crit' level or if they are not included.

Ik: scalar which identifies whether the kurtosis correction was binding.
if 0: no binding correction
if 1: the correction is binding

proc (3) = per_ao(y, b, k, crit);

This is a procedure for computing augmented Perron Additive Outlier (Crash) test with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations.

The break time is assumed to be known

inputs: y: a series of data, n x 1

b: time index indicating the breakpoint, b < n

k: maximum lag length

crit: critical value for selection

outputs: t: Perron Additive Outlier (AO) statistic

tt: Student-t statistic for the spike

kv: vector which identifies particular lags in augmentation

proc (3) = ao_test(y, out, kp, critp, d);

This is a procedure for computing an Additive Outlier (Crash) test in case of an unknown single break. It computes sequentially the Perron test (procedure per_ao) and chooses the best break according to the Zivot and Andrews(1992, JBEA) or Vogelsang and Perron (1998, IER) criteria.
It requires the procedure *per_ao* to be placed ABOVE this on in the program.

inputs:	y:	a series of data, n x 1
	out:	time index indicating the fraction of first and last observations which have to be discarded while testing
	kp:	maximum lag length
	critp:	critical value for selection
	d:	selector variable
		if d = 0, the Zivot-Andrews selection is performed
		if d = 1, the Vogelsang-Perron selection is performed
	(inputs kp and critp are required by the included per_ao procedure)	
outputs:	t:	final value of the test statistic
	tt:	Student-t statistic for the 'crash' dummy variable (spike)
	b:	number of the break in the series

proc (4) = per_io(y, b, k, crit);

This is a procedure for computing augmented Perron Innovative Outlier test with the selection of lags similar to Hall (1994) general to specific method. The selection of lags is according to the 'significant' Student-t statistics on augmentations. The break time is assumed to be known

inputs:	y:	a series of data, n x 1
	b:	time index indicating the breakpoint, b < n
	k:	maximum lag length
	crit:	critical value for selection
outputs:	t:	Perron Innovative Outlier (IO) statistic
	td:	Student-t statistic for the spike
	ts:	Student-t statistic on the step variable
	kv:	vector which identifies particular lags in augmentation

proc (4) = io_test(y, out, kp, critp, d);

This is a procedure for computing an Innovative Outlier test in case of an unknown single break. It computes sequentially the Perron test (procedure per_io) and chooses the best break according to the Zivot and Andrews(1992, JBEA) or Vogelsang and Perron (1998, IER) criteria. It requires the procedure per_io to be placed ABOVE this on in the program.

inputs:
- y: series of data, n x 1
- out: time index indicating the fraction of first and last observations which have to be discarded while testing
- kp: maximum lag length
- critp: critical value for selection
- d: selector variable
  - if d = 0, the Zivot-Andrews selection is performed
  - if d = 1, the Vogelsang-Perron selection is performed
  - if d = 2, sorting is according to the highest t-Student step statistic (see Harvey, Leybourne and Newbold (2001, WP) his is probably also one of the Vogelsang-Perron statistics (not checked yet).

WARNING: Initial Monte Carlo results shows that for d = 0 the estimated structural breakpoint is seriously overshot. Option d = 0 is therefore not recommended. Alternatively, we have a bug in the program.

(inputs kp and critp are required by the included per_io procedure)

outputs:
- t: final value of the test statistic
- tt: Student-t statistic for the 'crash' dummy variable (spike)
- td: Student-t statistic for the step dummy variable
- b: number of the break in the series

References

Charemza, W., M. Lifshits and S. Makarova (2001), 'Conditional testing for unit-root bilinearity in financial time series: some theoretical and empirical results', University of Leicester, mimeo.

Hall, A. (1994), 'Testing for a unit root in time series with pretest data-based model selection', *Journal of Business and Economic Statistics* **12**, pp. 461-470.

Harvey, D.I., S.J. Leybourne and P. Newbold (2001), 'Innovational outlier unit root tests with an endogenously determined break in level', University of Nottingham, mimeo.

Isaac, A.G. and D. Rapach (1996), 'Unit root testing: a collection of procedures'. Computer GAUSS code available at http://netec.mcc.ac.uk and http://adnetec/CodEc/ GaussAtAmericanU/GAUSSID/HTML.

Kwiatkowski, D., P.C.B. Phillips, P. Schmidt and Y. Shin (1992), 'Testing the null hypothesis of stationarity against the alternative of a unit root', *Journal of Econometrics* **54**, pp. 159-178.

Leybourne, S.J. (1995), 'Testing for unit root using forward and reverse Dickey-Fuller regressions', *Oxford Bulletin of Economics and Statistics* **57**, pp. 559-571.

Maddala, G.S. and I-M. Kim (1998), *Unit root, cointegration and structural change*, Cambridge University Press, Cambridge.

Perron, P. (1989), 'The Great Crash, the oil price shock and the unit root hypothesis', *Econometrica* **57**, pp. 1361-1401.

Vogelsang, T.J. and P. Perron (1998), 'Additional tests for a unit root allowing for a break in the trend function at an unknown time', *International Economic Review* **39**, pp. 1073-1100.

Zivot, E. and D.W.K. Andrews (1992), 'Further evidence on the Great Crash, the oil price shock and the unit root hypothesis', *Journal of Business and Economic Statistics* **10**, pp. 251-270.