This document details the computer codes used to conduct our empirical analyses. Downloading a file with the suffix .R or .PRG is not allowed on the Elsevier system. So, we detail below these codes.

A – Linearity test

The LRtest.R program presented below is aimed at being run under the R software environment to test for linearity.

Name: LRtest.R

```
dataframe <- read.csv(file.choose(), header=TRUE, sep=";")
dataframe

#Test 1vs2: Linear VAR versus 1 threshold TVAR
#Test 1vs3: Linear VAR versus 2 threshold2 TVAR

TVAR.LRtest(dataframe, lag=3, mTh=1,thDelay=1:2, nboot=500, plot=FALSE, trim=0.1, test="1vs")
TVAR(dataframe, model = c("TAR"),lag=3, nthresh=2, thDelay=1, trim=0.1, mTh=1, plot=TRUE)
TVAR.LRtest(dataframe, lag=3, mTh=1,thDelay=1:2, nboot=500, plot=FALSE, trim=0.1, test="2vs3")
```

B – Linearity test

The tvar_estimation.PRG program presented below is aimed at being run under the WinRATS Econometrics software. It evaluates the sup-Wald, Avg-wald and Exp-wald statistics to see whether the estimated TVAR model is statistically significant relative to a linear VAR. We thank Nathan Balke for having provided the original code.

Name: tvar_estimation.PRG

```
* Estimation of threshold
*
* Control parameters
*
* Fraction of threshold values excluded at each end
*
compute pi=.15
*
* Number of VAR lags
*
compute maxlag=3
*
* Threshold delay
*
compute d = 2
*
* Moving average terms in smoothing threshold variable
```

```
*
compute malength = 3
*
cal(m) 1992
allocate 2017:6
*
open data data1
data(unit=data,format=xls,org=col) 1992:1 2017:6 $
 us japan eu oil_price_uncertainty
*
* These are the desired start and end of the sample to be used in the
* estimation. However, the transformations use the entire range.
*
compute sstart = 1992:1
compute send   = 2017:6
*

set d1japan   =  log(japan)- log(japan{1})
set d1eu  =  log(eu)- log(eu{1})
set dlus     =  log(us)- log(us{1})

*
************************************************************
system(model=varmodel)
variables dlus d1japan d1eu oil_price_uncertainty
lags 1 to maxlag
det constant
end(system)
*
compute rstart=sstart+maxlag,rend=send
estimate(print,resids=u) rstart rend
*
compute loglr=%logl
compute nvar =%nvar
*
filter(type=lagging,span=malength) oil_price_uncertainty / oil_price_uncertaintythr
*
* Get a sorted copy of the threshold variable
*
set copy rstart rend = oil_price_uncertaintythr{d}
order copy rstart rend
*
compute piskip=fix(pi*%nobs)+(maxlag*nvar+1)
compute pistart=rstart+piskip,piend=rend-piskip
*
compute bestlogl=loglr
*
set lrstats pistart piend = 0.0
```

```
do pientry=pistart,piend
  compute thresh=copy(pientry)
  *
  * This allows for the covariance matrix to differ between regimes
  *
  sweep(var=hetero,group=oil_price_uncertaintythr{d}<thresh) rstart rend
  # %modeldepvars(varmodel)
  # %rlfromeqn(%modeleqn(varmodel,1))
  *
  * If this is best so far, save it
  *
  if %logl>bestlogl
     compute bestlogl=%logl,bestthresh=thresh
  compute lrstats(pientry)=2.0*(%logl-loglr)
end do pientry
*
disp "Best Threshold Value" bestthresh
compute suplr=2.0*(bestlogl-loglr)
*
* The degrees of freedom of the test count both the number of VAR
* parameters and the number of elements in the second estimated
* covariance matrix.
*
disp "Degrees of freedom" %nregsystem/2+%nvar*(%nvar+1)/2
*
sstats(mean) pistart piend lrstats>>avglr log(lrstats)>>avgloglr
compute explr=exp(avgloglr)
*
* Fixed regressor bootstrap
*
dec vect[series] fiddled(%nvar)
*
compute nboot=500
set avglrboot 1 nboot = 0.0
set explrboot 1 nboot = 0.0
set suplrboot 1 nboot = 0.0
*
do reps=1,nboot
  set lrstats pistart piend = 0.0
  set fiddlef = %ran(1.0)
  do i=1,%nvar
    set fiddled(i) = fiddlef*u(i)
  end do i
  *
  sweep rstart rend
  # fiddled
  # %rlfromeqn(%modeleqn(varmodel,1))
  compute loglr=%logl
```

```
  compute bestlogl=loglr
  *
  do pientry=pistart,piend
    compute thresh=copy(pientry)
    *
    * This allows for the covariance matrix to differ between regimes
    *
    sweep(var=hetero,group=oil_price_uncertaintythr{d}<thresh) rstart rend
    # fiddled
    # %rlfromeqn(%modeleqn(varmodel,1))
    *
    * If this is best so far, save it
    *
    if %logl>bestlogl
       compute bestlogl=%logl,bestthresh=thresh
    compute lrstats(pientry)=2.0*(%logl-loglr)
  end do pientry
  sstats(mean) pistart piend lrstats>>avglrboot(reps) log(lrstats)>>avgloglr
  compute explrboot(reps)=exp(avgloglr)
  compute suplrboot(reps)=2.0*(bestlogl-loglr)
end do reps
sstats(mean) 1 nboot (explrboot>explr)>>exppvalue $
                (suplrboot>suplr)>>suppvalue $
                (avglrboot>avglr)>>avgpvalue
*
report(action=define)
report(atrow=1,atcol=1) "" "Statistic" "P-value"
report(atrow=2,atcol=1) "Sup" suplr suppvalue
report(atrow=3,atcol=1) "Avg" avglr avgpvalue
report(atrow=4,atcol=1) "Exp" explr exppvalue
report(atcol=2,tocol=2,action=format,width=8)
report(atcol=3,tocol=3,action=format,picture="*.###")
report(action=show)
```

C – Computation of nonlinear impulse response functions

The tvar_irf.PRG program program presented below is aimed at being run under the WinRATS Econometrics software. That code is used to compute the nonlinear impulse response functions.

Name: tvar_irf.PRG

```
*************************************************************
*           Nonlinear impulse response functions
*************************************************************
* This program computes average IRF for output, conditional on being in
* the upper or lower regime, averaging across initial conditions and then,
* for each initial setting, across bootstrapped residuals.
*
* Control parameters
*
comp nvar    =4
comp horizon =15
*
* Number of bootstrap replications used in computing GIRF's
*
comp nkrep   =500
*
* Change to upper=0 to get the lower regime
*
comp upper   =1
*
comp [vector] shocksizes=||1.0,2.0||
comp [vector] shocksigns=||1.0,-1.0||
*
cal(m) 1992
allocate 2017:6
*
open data data1
data(unit=data,format=xls,org=cols) 1992:1 2017:6 $
  us japan eu oil_price_uncertainty
*
* These are the desired start and end of the sample to be used in the
* estimation. However, the transformations use the entire range.
*
compute sstart = 1992:2
compute send   = 2017:6

*
set d1japan   =  log(japan)- log(japan{1})
set d1eu  =  log(eu)- log(eu{1})
```

```
set dlus     =  log(us)- log(us{1})
*****************************************************************
*
dec vector[series] res(nvar) vres(nvar)
dec vector v(nvar) resmat(nvar)
dec vect[int]      depvars(nvar)
dec rect[int]      laglengths(nvar,nvar)
dec vect[equation] eqn(nvar)
dec vect[series]   resup(nvar) resdn(nvar)
dec rect[frml]     fitud(nvar,2)
dec vect[frml]     tvarf(nvar) empty(nvar) tfrml(nvar) rfrml(nvar)
dec vect[series]   bootu(nvar) data(nvar)
dec series[vect]   bootuv bootres
*
dec vect[string]   shortlabels(nvar) longlabels(nvar)
declare vector[labels] lab(nvar)
*
compute depvars    =||dlus,d1japan,d1eu,oil_price_uncertainty||
compute shortlabels=||"US","Japan","EU","oil_price_uncertainty"||
compute longlabels =||"US","Japan","Europe","oil_price_uncertainty"||
compute lab        =|| "US","Japan", "EU", "oil_price_uncertainty" ||
*
*  set thresholds, mas, and delays
*
compute d = 2
*
* 15% window
*
dec vector macoeffs
compute malength = 3  ; comp thresh = 0.038614356 ;* oil_price_uncertainty, ff

*
* This generates the actual moving average of the data
*
filter(type=lagging,span=malength) oil_price_uncertainty / uncertaintythr
*
* This generates an equation (and from it a FRML) to compute the moving
* average of the data
*
dim macoeffs(malength)
comp macoeffs = %const(1./malength)
equation(identity,coeffs=macoeffs) threqn uncertaintythr
# oil_price_uncertainty{0 to malength-1}
frml(equation=threqn,identity) thrfrml
*
* Lag lengths are allowed to differ among equations. These give the lag
* lengths with equation in a row and variable in a column.
*
```

```
input laglengths
  3 3 3 3
  3 3 3 3
  3 3 3 3
  3 3 3 3
**************************************************************************
*
* Dummy variables for the two regimes
*
set d1 = thrfrml{d}>thresh
set d2 = 1.-d1
*
compute maxlag=d+(malength-1)
*
* Create an equation for each variable with the number of lags for each
* endogenous variable given by <<laglengths>>.
*
do i=1,nvar
  compute [vect[integer]] rl=||constant||
  do j=1,nvar
    compute rl=%rladdlaglist(rl,depvars(j),%seq(1,laglengths(i,j)))
    compute maxlag=%imax(maxlag,laglengths(i,j))
  end do j
  equation eqn(i) depvars(i)
  # rl
end do i
*
compute rstart=sstart+maxlag,rend=send
do i=1,nvar
  linreg(equation=eqn(i)) * rstart rend
  frml empty(i) bootu(i) = 0.0
end do i
*
do i=1,nvar
  disp
  disp
  disp shortlabels(i)+" regression in upper regime"
  linreg(smpl=d1,equation=eqn(i),frml=fitud(i,1)) * rstart rend resup(i)
  disp
  disp
  disp shortlabels(i)+" regression in lower regime"
  linreg(smpl=d2,equation=eqn(i),frml=fitud(i,2)) * rstart rend resdn(i)
  set res(i) rstart rend = %if(d1,resup(i),resdn(i))
end do i
*
* Compute the upper and lower branch covariance matrices, their Choleski
* factors and the inverse factor. (The inverse is for standardizing the
* residuals and the factor for mapping standardized residuals back to
```

```
* the true levels.)
*
vcv(matrix=vup)
# resup
compute sup =%decomp(vup)
compute siup=inv(sup)

vcv(matrix=vdn)
# resdn
compute sdn =%decomp(vdn)
compute sidn=inv(sdn)
*

* Compute the joint covariance matrix and its factor.
*
vcv(matrix=vsigma)
# res
compute s=%decomp(vsigma)
*
* Compute (jointly) standardized residuals. Since we only use these as a
* VECTOR across variables at T (never as individual series), we define
* it as a SERIES[VECTOR].
*
dec series[vect] stdu
gset stdu rstart rend = %if(d1,siup*%xt(resup,t),sidn*%xt(resdn,t))
*
* Save the original data since we'll overwrite it as part of the
* bootstrap.
*
dec vect[series] data(nvar)
do i=1,nvar
  set data(i) = depvars(i){0}
end do i
*
* Define the switching formula. Because the bootstrap requires taking
* the standardized residuals and reflating them using regime-specific
* covariance matrices, the reflation part has to be incorporated into
* the formula. Thus TVARF(i) is (in effect) an identity given the (still
* to be created) bootres series.
*
do i=1,nvar
  frml tvarf(i) depvars(i) = %if(thrfrml{d}>thresh,$
      fitud(&i,1)+%dot(%xrow(sup,&i),bootres),$
      fitud(&i,2)+%dot(%xrow(sdn,&i),bootres))
end do i
*
* Build the threshold var model using the switching equations and the
* definitional identity for the threshold variable.
```

```
*
******************************************************************************

*
group tvar tvarf(1) tvarf(2) tvarf(3) tvarf(4) thrfrml
*
******************************************************************************
*
* Figure out which set of entries we want. This is the simplest way
* to get this. d1{0} takes two values (0 or 1) but the order isn't known
* in advance since it will depend upon which value is seen first in the
* <<rstart>> to <<rend>> range. So we have to figure out which of values(1)
* and values(2) (and the corresponding entries(1) and entries(2)) is the
* one that we need, given the choice for <<upper>>.
*
panel(group=d1{0},id=values,identries=entries) d1 rstart rend
{
if upper==1.and.values(1)==1.or.upper==0.and.values(1)==0
  compute rdates=entries(1),nrep=%size(rdates)
else
  compute rdates=entries(2),nrep=%size(rdates)
}
*
* Set up target series. There is one for each combination of test sign
* and sizes on the shock, variable shocked and target variable.
*
declare real size sign
compute nexp=%size(shocksizes)*%size(shocksigns)
dec vect[rect[series]] girfs(nexp)
do i=1,nexp
  dim girfs(i)(nvar,nvar)
  clear(zeros) girfs(i)
end do i
*
* This is the working range for calculating the GIRF's
*
compute wstart=rstart,wend=rstart+horizon-1
*
* Outer loop is over the initial conditions, which walk through the data
* points in the desired regime. The residuals are bootstrapped by taking
* the standardized residuals (across the entire range), shuffling them,
* and reflating them based upon the current threshold value in the
* generated series.
*
infobox(action=define,lower=1,upper=nrep,progress) "Bootstrapping Across Initial Values"
do jrep=1,nrep
  infobox(current=jrep)
  *
```

```
  * Copy observed data into depvar slots
  *
  compute basedate=rdates(jrep)
  do i=1,nvar
    set depvars(i) wstart-maxlag wstart-1 = data(i)(t-wstart+basedate)
  end do i
  *
  * Loop over bootstrap replications
  *
  do krep=1,nkrep
    *
    * Generate the bootstrap shuffle
    *
    boot rentries wstart wend rstart rend
    *
    * Generate the base set of shocks by premultiplying the
    * bootstrapped standardized shocks by the factor of the overall
    * covariance matrix.
    *
    gset bootuv wstart wend  = %if(%ranflip(.5),+1,-1)*stdu(rentries(t))
    *
    gset bootres wstart wend = bootuv
    *
    forecast(model=tvar,from=wstart,to=wend,results=base)
    compute ifill=0
    dofor sign = 1 -1
      dofor size = 1 2
        compute ifill=ifill+1
        do jshock=1,nvar
          *
          * Patch over the component for which are computing the
          * response with the selected size and sign.
          *
          compute bootres(wstart)=bootuv(wstart)
          compute bootres(wstart)(jshock)=sign*size
          *
          forecast(model=tvar,from=wstart,to=wend,results=withshock)
          do i=1,nvar
            set girfs(ifill)(i,jshock) wstart wend = girfs(ifill)(i,jshock)+withshock(i)-base(i)
          end do i
        end do jshock
      end do sign
    end do size
  end do krep
end do jrep
infobox(action=remove)
*
do k=1,nexp
```

```
  do i=1,nvar
    do j=1,nvar
      set girfs(k)(i,j) wstart wend = girfs(k)(i,j)/(nkrep*nrep)
    end do j
  end do i
end do k
*
* Move the data back
*
do i=1,nvar
  set depvars(i) = data(i)
end do i
*
*********************************************************************
*
dec vect[series] graphs(nexp)
compute klabels=||"+1 SD","+2 SD","-1 SD","-2 SD"||
*
spgraph(vfields=nvar,hfields=1,footer=$
  "Figure 2. Response of US to Shocks, Conditional on Regime")
do j=1,nvar
  do i=1,nexp
    set graphs(i) 1 horizon = girfs(i)(1,j)(t+wstart-1)
  end do i
  graph(series=graphs,number=0,key=upright,klabels=klabels,$)
   header="High Uncertainty Regime: Response of US",subheader="Shock to "+shortlabels(j))
end do j
spgraph(done)

*
spgraph(vfields=nvar,hfields=1,footer=$
  "Figure 3. Response of Japan to Shocks, Conditional on Regime")
do j=1,nvar
  do i=1,nexp
    set graphs(i) 1 horizon = girfs(i)(2,j)(t+wstart-1)
  end do i
  graph(series=graphs,number=0,key=upright,klabels=klabels,$
   header="High Uncertainty Regime: Response of Japan",subheader="Shock to "+shortlabels(j))
end do j
spgraph(done)

spgraph(vfields=nvar,hfields=1,footer=$
  "Figure 4. Response of Europe to Shocks, Conditional on Regime")
do j=1,nvar
  do i=1,nexp
    set graphs(i) 1 horizon = girfs(i)(3,j)(t+wstart-1)
  end do i
  graph(series=graphs,number=0,key=upright,klabels=klabels,$
```

```
    header="High Uncertainty Regime: Response of EU",subheader="Shock to "+shortlabels(j))
end do j
spgraph(done)


spgraph(vfields=nvar,hfields=1,footer=$
  "Figure 5. Response of oil price uncertainty to Shocks, Conditional on Regime")
do j=1,nvar
  do i=1,nexp
    set graphs(i) 1 horizon = girfs(i)(4,j)(t+wstart-1)
  end do i
  graph(series=graphs,number=0,key=upright,klabels=klabels,$
    header="High Uncertainty Regime: Response of OPU",subheader="Shock to "+shortlabels(j))
end do j
spgraph(done)
*********************************************************************************
************************
*                    Diagnostic checks of TVAR residuals
*********************************************************************************
************************


* Multivariate Ljung-Box test

**High regime

@mvqstat (lags=1)
#resup

** Low regime

@mvqstat (lags=1)
#resdn
*


* Ljung-Box Q-Statistics

CORRELATE(QSTATS,METHOD=YULE) RESUP(1)
@ACF(METHOD=YULE,QSTATS) RESUP(1)

CORRELATE(QSTATS,METHOD=YULE) RESUP(2)
@ACF(METHOD=YULE,QSTATS) RESUP(2)

CORRELATE(QSTATS,METHOD=YULE) RESUP(3)
@ACF(METHOD=YULE,QSTATS) RESUP(3)

CORRELATE(QSTATS,METHOD=YULE) RESUP(4)
```

```
@ACF(METHOD=YULE,QSTATS) RESUP(4)

CORRELATE(QSTATS,METHOD=YULE) RESDN(1)
@ACF(METHOD=YULE,QSTATS) REDN(1)

CORRELATE(QSTATS,METHOD=YULE) RESDN(2)
@ACF(METHOD=YULE,QSTATS) RESDN(2)

CORRELATE(QSTATS,METHOD=YULE) RESDN(3)
@ACF(METHOD=YULE,QSTATS) RESDN(3)

CORRELATE(QSTATS,METHOD=YULE) RESDN(4)
@ACF(METHOD=YULE,QSTATS) RESDN(4)


* Testing for ARCH effects

@archtest(lags=6,form=lm,span=1) RESUP(1)

@archtest(lags=6,form=lm,span=1) RESUP(2)

@archtest(lags=6,form=lm,span=1) RESUP(3)

@archtest(lags=6,form=lm,span=1) RESUP(4)

@archtest(lags=6,form=lm,span=1) RESDN(1)

@archtest(lags=6,form=lm,span=1) RESDN(2)

@archtest(lags=6,form=lm,span=1) RESDN(3)

@archtest(lags=6,form=lm,span=1) RESDN(4)




* Normality test of the residuals (Andersen-Darling test)

*** Low Regime
sample(smpl=%valid(RESDN(1))) RESDN(1) / DN1
@adtest DN1

sample(smpl=%valid(RESDN(2))) RESDN(2) / DN2
@adtest DN2

sample(smpl=%valid(RESDN(3))) RESDN(3) / DN3
@adtest DN3
```

```
sample(smpl=%valid(RESDN(1))) RESDN(4) / DN4
@adtest DN4

*** High Regime

sample(smpl=%valid(RESUP(1))) RESUP(1) / UP1
@adtest UP1

sample(smpl=%valid(RESUP(2))) RESUP(2) / UPP2
@adtest UPP2

sample(smpl=%valid(RESUP(3))) RESUP(3) / UP3
@adtest UP3

sample(smpl=%valid(RESUP(4))) RESUP(4) / UPP4
@adtest UPP4
```