‘Help’ for the Main CAPTAIN Tools used in the Book

1.1 The rivcbjid tool in CAPTAIN

```
[th,M,Mn,D,stats,e,RR,eef,var,Ps,Pn,y0,thd,statsd,AH,AHse,PH,AHn,AHnse,PHn] = ...
                            rivcbjid(z,nn,sc,flags,c,nar,gm)
                                     1  2  3   4   5  6  7

 z: I-O data, [Y,U1,...,Unu] where Y and U are column vectors (*)
 nn: Model order search range, 2 by (2+2*nu) matrix (*)
       [na,nb(1:nu),ntd(1:nu) nc nd] (row 1: search from, row 2: to)
       na: denominator, nb: numerators, ntd: time delays
       nc: noise model denominator (AR), nd: numerator (MA)
 sc: Selection criterion (1-YIC, 2-RT2, 3-BIC, 4-AIC) (1)
 flags: Additional parameters vector [Ni,dt,ddt,cf,Rc,rrts]
        Missing value or -1 implies use default value in brackets
          (1) Ni<1: convergence criterion to compare with quadratic norm
              of parameter vector change between iterations (default 1e-4)
              Ni>1: Number (maximum) of SRIV iterations
          (2) dt: Sampling interval for continuous time estimation (1)
          (3) ddt: Sampling for initial discrete time estimation (1)
          (4) cf: Constant (1) or adaptive (0) pre-filter flag (0)
          (5) Rc: Block (0-default) or recursive algorithm (1-slower)
          (6) rrts: Option to only return models with real-roots (0)
                    0: all models returned
                    1: real-roots models (rejection threshold=eps)
                    >0: set rejection threshold e.g. 1e-10
 c: Prefilter polynomial parameter (1)
      Polynomial: Polymomial with order 'na'
      Scalar <=0: Polynomial created automatically from pole value
                  e.g. zero to utilise multiple integrators, <0 for
                  stable filter (chosen so that 1/C roughly matches
                  the bandwidth of the system being estimated)
      1: Estimate discrete-time filter and convert to continous-time
         Normally C<=0 is best for fast and C=1 for coarse sampled data
         (or, if fast sampled, can use C=1 with ddt set at a more coarse
         subsampling rate for initial discrete-time estimation)
 nar: High AR model order used for ARMA modeling with IVARMA method (50)
 gm: ARMA estimation algorithm (default - 1 when SID present, else 0)
       0: IVARMA (CAPTAIN)
       1: PEM (rather quicker that IVARMA but requires SID Toolbox)

 th: Theta matrix
 M: Estimated model object (SID Toolbox required) or empty
 Mn: ARMA noise model in discrete time (SID Toolbox required) or empty
 D: Data object (SID Toolbox required) or empty
 stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,0,Ybar,RT2c,AIC]
          (RT2: I-O; RT2c: full model)
 e: System model output errors (y=fit+e)
 RR: Table of results for all models searched
 eef: Residuals from noise model (stochastic model residuals)
 var: Variance of residuals: eef if noise model present, else e (SRIV)
 Ps: Covariance matrix for I-O parameter estimates
 Pn: Covariance matrix for ARMA noise model parameter estimates
 y0: Interpolated data
 thd: Theta matrix for initial discrete time model
 statsd: Stats vector for discrete time model
 AH,AHse,PH: recursive estimates (if requested) of I-O component
   AH (Npars x Nsamp): parameter estimate vectors at each data point
   AHse (Npars x Nsamp): standard errors for the above
   PH (Npars x (Npars*Nsamp): covariance matrices concatenated horizontally
      e.g. AH(:,24) is the parameters vector estimate at sample 24
      with the standard deviations AHse(:,24) and the covariance matrix
      given by PH(:,23*size(AH,1)+(1:size(AH,1)))
 AHn,AHnse,PHn: recursive estimates of the noise (ARMA) model
    component with the contents defined as for the I-O component above

 Example: [th, M]=rivcbjid([y u], [1 1 1 1 1; 3 3 5 1 1])
   output y and input u, estimate TF models in the range [1 1 1]
   to [3 3 5], with first order ARMA(1,1) noise model and listing
   the results in order of lowest YIC first (default case)
```

1.2 The rivcbj tool in CAPTAIN

```
rivcbj  Estimation of a continuous time MISO transfer function

 [th,M,Mn,D,stats,e,eef,var,Ps,Pn,y0,thd,statsd,AH,AHse,PH,AHn,AHnse,PHn] = ...
                                    rivcbj(Z,nn,flags,c,a0,P0,nar,gm)
                                           1  2   3   4  5  6  7  8

 Or for SID object code output (SID Toolbox required)
   M = rivcbj(z,nn,flags,c,a0,P0,nar,gm)

 z: I-O data, [Y,U1,...,Unu] where Y and Ui are column vectors (*)
 nn: Model order, [na,nb(1:nu),ntd(1:nu) nc nd] (*)
       na: denominator, nb: numerators, ntd: time delays
       nc: noise model denominator (AR), nd: numerator (MA)
 flags: Additional parameters vector [Ni,dt,ddt,cf,Rc]
        Missing value or -1 implies use default value in brackets
          (1) Ni<1: convergence criterion to compare with quadratic norm
              of parameter vector change between iterations (default 1e-4)
              Ni>1: Number (maximum) of SRIV iterations
          (2) dt: Sampling interval for continuous time estimation (1)
          (3) ddt: Sampling for initial discrete time estimation (1)
          (4) cf: Constant (1) or adaptive (0) pre-filter flag (0)
          (5) Rc: Block (0-default) or recursive algorithm (1-slower)
               (Rc=1 if non-empty a0 and P0 supplied by user)
 c: Prefilter polynomial parameter (1)
      Polynomial: Polymomial with order 'na'
      Scalar <=0: Polynomial created automatically from pole value
                  e.g. zero to utilise multiple integrators, <0 for
                  stable filter (chosen so that 1/C roughly matches
                  the bandwidth of the system being estimated)
      1: Estimate discrete-time filter and convert to continous-time
         Normally C<=0 is best for fast and C=1 for coarse sampled data
         (or, if fast sampled, can use C=1 with ddt set at a more coarse
         subsampling rate for initial discrete-time estimation)
 a0: Initial values (priors) of parameters for recursive estimation (0)
 P0: Initial condition (prior) of covariance matrix diagonal (1e5)
     If supplied as a vector this will be used to define the diagonal
     elements of a diagonal P0 but a full P0 matrix can be supplied
 nar: High AR model order used for ARMA modeling with IVARMA method (50)
 gm: ARMA estimation algorithm (default - 1 when SID present, else 0)
       0: IVARMA (slower but provides recursive estimates of all
          parameters if requested)
       1: PEM (quicker than IVARMA but requires SID Toolbox). Cannot
          provide recursive estimates of noise model parameters, so
          replaced by IVARMA if recursive estimates are requested

 th: Theta matrix when called with multiple output arguments
     If only one output argument: provides model structure of SID type
     (if SID Toolbox exists) else a structure with the same fields as SID
 M: Estimated model object (SID Toolbox required) or empty
 Mn: ARMA noise model in discrete time (SID Toolbox required) or empty
 D: Data object (SID Toolbox required) or empty
 stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,0,Ybar,RT2c,AIC]
          (RT2: I-O; RT2c: full model)
 e: System model output errors (y=fit+e)
 eef: Residuals from noise model (stochastic model residuals)
 var: Variance of residuals: eef if noise model present, else e (SRIV)
 Ps: Covariance matrix for I-O parameter estimates
 Pn: Covariance matrix for AR parameter estimates
 y0: Interpolated data
 thd: Theta matrix for initial discrete time model
 statsd: Stats vector for discrete time model
 AH,AHse,PH: recursive estimates (if requested) of I-O component
   AH (Npars x Nsamp): parameter estimate vectors at each data point
   AHse (Npars x Nsamp): standard errors for the above
   PH (Npars x (Npars*Nsamp): covariance matrices concatenated horizontally
      e.g. AH(:,24) is the parameters vector estimate at sample 24
      with the standard deviations AHse(:,24) and the covariance matrix
      given by PH(:,23*size(AH,1)+(1:size(AH,1)))
 AHn,AHnse,PHn: recursive estimates of the noise (ARMA) model
    component with the contents defined as for the I-O component above

 Example: [th, M]=rivcbj([y u], [4 3 7 1 1], [-1 0.005], -gamma, [], 0);
   output y and input u, estimate model with a [4 3 7] TF structure and a
   first order ARMA(1,1) noise model using RIV with default Ni and sampling
   interval 0.005; starting MMF prefilter with user-defined parameter
   (-gamma); default AR order; and IVARMA noise model estimation
```

1.3 The srivcarma tool in CAPTAIN

```
srivc\_arma  Estimation of a continuous time MISO transfer function with separate estimation of ARMA noise Model.

  [th,M,Mn,D,stats,e,eef,var,Ps,Pn,y0,thd,statsd,AH,AHse,PH,AHn,AHnse,PHn] = ...
                                    srivcarma(Z,nn,flags,c,a0,P0,nar,gm,mod)
                                           1  2   3   4  5  6  7  8

  Or for SID object code output (SID Toolbox required)
    M = rivcbj(z,nn,flags,c,a0,P0,nar,gm)

  z: I-O data, [Y,U1,...,Unu] where Y and Ui are column vectors (*)
  nn: Model order, [na,nb(1:nu),ntd(1:nu) nc nd] (*)
        na: denominator, nb: numerators, ntd: time delays
        nc: noise model denominator (AR), nd: numerator (MA)
  flags: Additional parameters vector [Ni,dt,ddt,cf,Rc]
         Missing value or -1 implies use default value in brackets
           (1) Ni<1: convergence criterion to compare with quadratic norm
               of parameter vector change between iterations (default 1e-4)
               Ni>1: Number (maximum) of SRIV iterations
           (2) dt: Sampling interval for continuous time estimation (1)
           (3) ddt: Sampling for initial discrete time estimation (1)
           (4) cf: Constant (1) or adaptive (0) pre-filter flag (0)
           (5) Rc: Block (0-default) or recursive algorithm (1-slower)
                (Rc=1 if non-empty a0 and P0 supplied by user)
  c: Prefilter polynomial parameter (1)
       Polynomial: Polymomial with order 'na'
       Scalar <=0: Polynomial created automatically from pole value
                   e.g. zero to utilise multiple integrators, <0 for
                   stable filter (chosen so that 1/C roughly matches
                   the bandwidth of the system being estimated)
       1: Estimate discrete-time filter and convert to continous-time
          Normally C<=0 is best for fast and C=1 for coarse sampled data
          (or, if fast sampled, can use C=1 with ddt set at a more coarse
          subsampling rate for initial discrete-time estimation)
  a0: Initial values (priors) of parameters for recursive estimation (0)
  P0: Initial condition (prior) of covariance matrix diagonal (1e5)
      If supplied as a vector this will be used to define the diagonal
      elements of a diagonal P0 but a full P0 matrix can be supplied
  nar: High AR model order used for ARMA modeling with IVARMA method (50)
  gm: ARMA estimation algorithm (default - 1 when SID present, else 0)
        0: IVARMA (slower but provides recursive estimates of all
        parameters if requested)
        1: PEM (rather quicker that IVARMA but requires SID Toolbox). Cannot
            provide recursive estimates of noise model parameters, so automatically
            replaced by IVARMA if recursive estimates are requested)
  mod: specifies whether the covariance matrix Ps is computed based on the
          theretically optimal rivcbj or the srivc\_arma generated Ps (0)

  th: Theta matrix when called with multiple output arguments ('help theta')
      If only one output argument: provides model structure of SID type
     (if SID Toolbox exists) else a structure with the same fields as SID
  M: Estimated model object (SID Toolbox required) or empty
  Mn: ARMA noise model in discrete time (SID Toolbox required) or empty
  D: Data object (SID Toolbox required) or empty
  stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,0,Ybar,RT2c,AIC]
          (RT2: I-O; RT2c: full model)
  e: System model output errors (y=fit+e)
  eef: Residuals from noise model (stochastic model residuals)
  var: Variance of residuals: eef if noise model present, else e (SRIV)
  Ps: Covariance matrix for I-O parameter estimates
  Pn: Covariance matrix for AR parameter estimates
  y0: Interpolated data
  thd: Theta matrix for initial discrete time model
  statsd: Stats vector for discrete time model
  AH,AHse,PH: recursive estimates (if requested) of I-O component
    AH (Npars x Nsamp): parameter estimate vectors at each data point
    AHse (Npars x Nsamp): standard errors for the above
    PH (Npars x (Npars*Nsamp): covariance matrices concatenated horizontally
       e.g. AH(:,24) is the parameters vector estimate at sample 24
       with the standard deviations AHse(:,24) and the covariance matrix
       given by PH(:,23*size(AH,1)+(1:size(AH,1)))
  AHn,AHnse,PHn: recursive estimates of the noise (ARMA) model
     component with the contents defined as for the I-O component above

  Example: [th, M]=srivcarma([y u], [4 3 7 1 1], [0.00001 0.005], -gamma, [], 0);
    output y and input u, estimate model with a [4 3 7] TF structure and a
    first order ARMA(1,1) noise model, with conv. crit. 0.00001 and sampling
    interval 0.005; starting MMF prefilter with user-defined parameter
    (-gamma); default AR order; and IVARMA noise model estimation
```

1.4 The rivcbjfdid and rivcbjfd tools in CAPTAIN

```
 function [th,M,Mn,D,stats,e,eef,var,Ps,Pn,RR,FD,UD,model,P] = ...
           rivcbjfdid(Z,nn,flags,C,dn,intp,niv)

 RIVCBJ  Identification of the best structure of a continuous time MISO transfer function with
 fractional time delays and estimation of the model parameters.
 NOTE : See also the help instructions for RIVCBJFD

 INPUTS
 As rivcbj  +
 dn : integer number of equal increments over the
 sampling interval (defines the FTD accuracy: default 10; i.e.accuracy 0.1)
 intp : interpolation method: 'linear'; 'pchip'; cubic'; spline; etc (see Matlab interp1.m)
 range : the range of the time delays to be investigated (a matrix
 of dimesion (niv,2) with rows containing the upper and lower bounds on
 range for each input to be considered for evidence of fractionality separated
 by semicolon (e.g. [2;4] for range 2 to 4). (see niv below)
 niv : number of inputs to consider (taken consecutively so put
 the inputs that are not to be considered after these in Z)

 OUTPUTS
 As rivcbj with model M having best integer TD (not optimum model) +
 RR : each cell showing  error variance (2nd col.) changing with fractional interval index (1st col.)
 plot as plot(RR{j}(:,1),RR{j}(:,2)), where j=input index
 FD: vector of fractional delays in all input channels (if zero, no minimum found)
 UD: matrix with fractional delay-modified inputs as columns
 Zs : Data matrix adjusted for fractional delays
 Mh : FTD 'tf' model object
 opt : optimal estimate of the fractional time delays for each input
 considered (columns with optimal delay, residual variance and RT2)
 MTD : Table of all results: nx2 matrix, where n=number of intervals
 First colum is local optimum  between integer delays and second column is
 variance of residuals
```

```
RIVCBJ  Estimation of a Continuous Time MISO transfer function with
fractional time delays.

 function [th,M,Mn,D,stats,e,eef,var,Ps,Pn,RR,FD,UD,model,P] = ...
           rivcbjfd(Z,nn,flags,C,dn,intp,niv)

  RIVCBJ  Estimation of a continuous time MISO Transfer Function with
  Fractional Input Time Delays.

  NOTE 1: the search is over ONE sampling interval and this has
  to be identified. In normal usage, without any prior information,
  possible integer time delays should be identified first by RIVCBJID and
  then a local search around these, using rivcbjfd, may be required.
  FOR EACH FTD, the time delays in 'nn' should be THE LARGER of the two
  integer time delays identified by the prior RIVCBJID analysis to lie
  above and below the fractional TD (e.g. if these are 7 and 8 for one of
  the FTDs, then the 'nn' entry for this FTD should be 8.
  NOTE 2: If the inputs are steps or PRBS signals, the input-output data
  should be suitably pre-filtered prior to the analysis.

 Inputs
  As rivcbj then
  dn : integer number of equal increments over the
  sampling interval (defines the FTD accuracy: default 10; i.e.accuracy 0.1)
  intp : interpolation method
  niv : number of inputs to consider (taken consecutively so put
  the inputs that are not to be considered after these in Z)

  Ouputs
  As rivcbj but M now based on adjusted input series and shows integer TD
  RR : each cell showing  error variance (2nd col.) changing with fractional
  interval index (1st col.)
  plot as plot(RR{j}(:,1),RR{j}(:,2), where j=input index
  FD: vector of fractional delays in all input channels (if zero, no minimum found)
  UD: matrix [y ud(1:nv)] with fractional delay-modified inputs ud(1:nv)
  as columns.
  Zs : Data matrix adjusted for fractional delays
  Mh : FTD 'tf' model object
  cost: variance of residuals
  uq, yq: input and output at higher sampling rate defined by dn
  (e.g. if N is sample size and dn=100, the higher sampling rate is 0.01,
  so tht uq and yq each have N/0.01 samples)
  28/01/2019, PCY, new function that accesses RIVCBJ
  See also tdest, rivcbj,  rivcbjid
```

1.5 The dtfmct tool in CAPTAIN

```
DTFMCT  SISO Dynamic Transfer Function estimation
for Continuous-Time Models with Time Variable Parameters

[tfs,fit,fitse,par,parse,parc,e,y0,parc] = ...
       dtfmct(y,u,nn,TVP,nvr,P0,x0,smooth,ALG,nivit,dt)
                                  1 2 3   4   5  6  7  8   9  10

y: Time series (*)
u: Input (*)
nn: Model structure [na,nb(1:nu),nd(1:nu)] ([1 1 0])
TVP: Model type for each TVP (0-RW, 1-IRW) (0)
nvr: NVR hyper-parameters (0)
P0: Initial P matrix (1e5)
x0: Initial state vector (0)
sm: Smoothing on (1-default) or off (0-saves memory)
ALG: Smoothing algorithm: P (0) or Q (1-default)
nivit: Number of IV iterations (3)
dt: sampling interval (1)

tfs: Transfer function (simulation) output
fit: Model fit
fitse: Standard error of the fit
par: DT Parameter estimates
parsec: Standard errors of parameters
parc: CT Parameter estimates
e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
y0: Interpolated data

Example: dtfmct(y, [u1 u2], [1 1 1 2 3], 0, 0.001)
  DT transfer function type model y(k) = a(k)*y(k-1) + b1(k)*u1(k-2) + b2(k)*u2(k-3);
  or converted CT version of this; with RW models for both parameters (NVR=0.001)
```

Note: the associated dtfm tool is restricted to DT models and is quicker when it is only a DT model that is required.

1.6 The sdp tool in CAPTAIN

```
SDP  Non-Parametric Estimation of a Nonlinear Regression Model with
State-Dependent Parameters.

 [fit,fitse,par,parse,xs,pars,parses,rsq,nvr,y0]=...
                              sdp(y,z,x,TVP,nvrc,opts,P0,x0,nvr0,tab,nvrm)
                                  1 2 3  4   5    6   7   8   9  10   11

 y: Time series (*)
 z: Regressors (*)
 x: States on which the parameters depend, column for each regressor (z)
 TVP: Model type for each TVP (0-RW, 1-IRW) (0)
 nvrc: Constraints for each NVR (0)
         negative integer (-n) implies optimisation over first n backfit steps
 opts: Estimation options [iter con meth sm ALG plotopt], use -1 for defaults
         iter: number of backfitting iterations (10)
         con: backfitting convergence threshold (0.001)
         meth: Optimisation estimation method (0)
                 0: Maximum Likelihood
                 Integer:Sum of squares of the #-step-ahead forecasting errors
         sm: Smoothing on (1-default) or off (0-saves memory)
         ALG: Smoothing algorithm P (0) or Q (1-default)
         plotopt: Plot results during estimation on (1) or off (0-default)
 P0: Initial P matrix diagonal (1e5)
 x0: Initial state vector (0)
 nvr0: Initial NVRs specified by user (0.0001)
 tab: Display: 0=none, 1=tabulate results, 2=update window (2)
 nvrm: Maximum NVR constraint (1)

 fit: Model fit
 fitse: Standard error of the fit
 par: Parameter estimates
 parse: Standard errors of parameters
 xs: Sorted states
 pars: Sorted parameter estimates
 parses: Sorted standard errors of parameters
 rsq: R squared value
 nvr: Estimated NVRs
 y0: Interpolated data

 Example: sdp(y,[u1 u2],[x1 x2],[0 1],[-1 -2])
   regression type model y = c1(x1)*u1 + c2(x1)*u2, with an RW model for
   c1 where the dependent state is x1 and an IRW model for c2 where the
   dependent state is x2; the NVR for the first SDP (c1) is optimised at
   the first iteration and at the first two iterations for the second SDP
```

1.7 The rivbjid tool in CAPTAIN

```
rivbjid  Identification of a backward shift MISO transfer function

  [th,M,D,stats,e,RR,eef,var,Ps,Pn] = rivbjid(z,nn,sc,flags,nar,gm)
                                        1  2  3   4    5  6

  z: I-O data, [Y,U1,...,Unu] where Y and U are column vectors (*)
  nn: Model order search range, 2 by (2+2*nu) matrix (*)
        [na,nb(1:nu),ntd(1:nu) nc nd] (row 1: search from, row 2: to)
        na: denominator, nb: numerators, ntd: time delays
        nc: noise model denominator (AR), nd: numerator (MA)
  sc: Selection criterion (5)
        1: YIC symmetric; 2: RT2 (I-O only); 3: BIC; 4: EVN
        5: YIC asymmetric; 6: YIC quadratic; 7: RT2 (full model); 8-AIC
  flags: Additional parameters vector [Ni,Ft,Lr,Rc,Stb,Yini,rrts]
         Missing value or -1 implies use default in brackets
           (1) Ni<1: convergence criterion to compare with quadratic norm
               of parameter vector change between iterations (default 1e-4)
               Ni>1: Number (maximum) of SRIV iterations
           (2) Ft: Filtering in IV/SRIV (2)
                     1: Stabilised A for prefiltering only
                     2: Stabilised A for instruments and prefilter
                     0: Filtering turned off
           (3) Lr: Linear regression method (0)
                     0: Standard
                     tol>0: SVD/QR robust algorithm
           (4) Rc: Block (0-default) or recursive algorithm (1-slower)
           (5) Stb: Stabilisation of filter and model polynomial (1)
                      0: No stabilisation
                      1: Stabilise filter and instrument generation
                      2: Stabilise filter only (enables estimation
                         of marginally unstable systems)
           (6) Yini: Initial conditions (0)
                      0: Original initial y values
                      1: Mean value of initial y values
           (7) rrts: Option to only return models with real-roots (0)
                      0: all models returned
                      1: real-roots models (rejection threshold=eps)
                      >0: set rejection threshold e.g. 1e-10
  nar: High AR model order used for ARMA modeling with IVARMA method (50)
  gm: ARMA estimation algorithm (default - 1 when SID present, else 0)
        0: IVARMA (CAPTAIN)
        1: PEM (rather quicker that IVARMA but requires SID Toolbox)

  th: Theta matrix
  M: Estimated model object (SID Toolbox required) or empty
  D: Data object (SID Toolbox required) or empty
  stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,EVN,Ybar,RT2c,AIC,YICa,YICt]
         (RT2: I-O; RT2c: full model; YICa: asymmetric P; YICt: quadratic)
  e: System model output errors (y=fit+e)
  RR: Table of results for all models searched
  eef: Residuals from noise model (stochastic model residuals)
  var: Variance of residuals: eef if noise model present, else e (SRIV)
  Ps: Covariance matrix for I-O parameter estimates
  Pn: Covariance matrix for ARMA noise model parameter estimates

  Example: [th, M]=rivbjid([y u], [1 1 1 1 1; 3 3 5 1 1])
    output y and input u, estimate TF models in the range [1 1 1]
    to [3 3 5], with first order ARMA(1,1) noise model and listing
    the results in order of lowest YIC first (default case)
```

1.8 The rivbj tool in CAPTAIN

```
RIVBJ  Estimation of a Discrete-Time MISO Transfer Function

 [th,M,D,stats,e,eef,var,Ps,Pn,y0,AH,AHse,PH,Pr,AHn,AHnse,PHn] = ...
                                        rivbj(z,nn,flags,a0,P0,nar,gm)
                                              1  2   3   4   5  6  7

 Or for SID object code output (SID Toolbox required)
   M = rivbj(z,nn,flags,a0,P0,nar,gm)

 z: I-O data, [Y,U1,...,Unu] where Y and U are column vectors (*)
 nn: Model order, [na,nb(1:nu),ntd(1:nu) nc nd] (*)
       na: denominator, nb: numerators, ntd: time delays
       nc: noise model denominator (AR), nd: numerator (MA)
 flags: Additional parameters vector [Ni,Ft,Lr,Rc,Stb,Yini]
        Missing value or -1 implies use default in brackets
          (1) Ni<1: convergence criterion to compare with quadratic norm
              of parameter vector change between iterations (default 1e-4)
              Ni>=1: Number (maximum) of SRIV iterations
          (2) Ft: Filtering in IV/SRIV (2)
                    1: Stabilised A for prefiltering only
                    2: Stabilised A for instruments and prefilter
                    0: Filtering turned off
          (3) Lr: Linear regression method (0)
                    0: Standard
                    tol>0: SVD/QR robust algorithm
          (4) Rc: Block (0-faster) or recursive algorithm (1-default,
                slower but more robust convergence)
               (Rc=1 if non-zero a0, or missing input values)
          (5) Stb: Stabilisation of filter and model polynomial (1)
                     0: No stabilisation
                     1: Stabilise filter and instrument generation
                     2: Stabilise filter only (enables estimation
                        of marginally unstable systems)
          (6) Yini: Initial conditions (0)
                     0: Original initial y values
                     1: Mean value of initial y values
 a0: Initial values (priors) of parameters for recursive estimation (0)
 P0: Initial condition (prior) of covariance matrix diagonal (1e5)
     If supplied as a vector this will be used to define the diagonal
     elements of a diagonal P0 but a full P0 matrix can be supplied
 nar: High AR model order used for ARMA modeling with IVARMA method (50)
 gm: ARMA estimation algorithm (default - 1 when SID present, else 0)
       0: IVARMA (slower but provides recursive estimates of all
       parameters if requested)
       1: PEM (quicker than IVARMA but requires SID Toolbox). Cannot
          provide recursive estimates of noise model parameters, so
          replaced by IVARMA if recursive estimates are requested

 th: Theta matrix when called with multiple output arguments
     If only one output argument: provides model structure of SID type
     (if SID Toolbox exists) else a structure with the same fields as SID
 M: Estimated model object (SID Toolbox required) or empty
 D: Data object (SID Toolbox required) or empty
 stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,EVN,Ybar,RT2c,AIC,YICa,YICt]
        (RT2: I-O; RT2c: full model; YICa: asymmetric P; YICt: quadratic)
 e: System model output errors (y=fit+e)
 eef: Residuals from noise model (stochastic model residuals)
 var: Variance of residuals: eef if noise model present, else e (SRIV)
 Ps: Covariance matrix for I-O parameter estimates
 Pn: Covariance matrix for ARMA noise model parameter estimates
 y0: Interpolated data
 AH,AHse,PH: recursive estimates (if requested) of I-O component
   AH (Npars x Nsamp): parameter estimate vectors at each data point
   AHse (Npars x Nsamp): standard errors for the above
   PH (Npars x (Npars*Nsamp): covariance matrices concatenated horizontally
      e.g. AH(:,24) is the parameters vector estimate at sample 24
      with the standard deviations AHse(:,24) and the covariance matrix
      given by PH(:,23*size(AH,1)+(1:size(AH,1)))
 Pr: Asymmetric RIV covariance matrix estimate
 AHn,AHnse,PHn: recursive estimates of the noise (ARMA) model
    component with the contents defined as for the I-O component above

 Example: [th, M]=rivbj([y u], [2 1 3 1 1], 1e-5, [], [], [], 1)
   output y and input u, estimate model with a [2 1 3] TF structure,
   using convergence criterion norm 1e-5, with ARMA(1,1) model estimated
   by PEM gradient algorithm if available
```

1.9 The dtfm and dtfmopt tools in CAPTAIN

```
  dtfm  Multivariable Dynamic Transfer Function estimation
                                        using instrumental variables

  [tfs,fit,fitse,par,parse,e,y0]=dtfm(y,u,nn,TVP,nvr,P0,x0,sm,ALG,niv)
                                     1 2 3   4   5  6  7  8   9  10

  y: Time series (*)
  u: Input (*)
  nn: Model structure [na,nb(1:nu),nd(1:nu)] ([1 1 0])
  TVP: Model type for each TVP (0-RW, 1-IRW) (0)
  nvr: NVR hyper-parameters (0)
  P0: Initial P matrix (1e5)
  x0: Initial state vector (0)
  sm: Smoothing on (1-default) or off (0-saves memory)
  ALG: Smoothing algorithm: P (0) or Q (1-default)
  niv: Number of IV iterations (3)

  tfs: Transfer function (simulation) output
  fit: Model fit
  fitse: Standard error of the fit
  par: Parameter estimates
  parse: Standard errors of parameters
  e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
  y0: Interpolated data

  Example: dtfm(y, [u1 u2], [1 1 1 2 3], 0, 0.001)
    transfer function type model y(k) = a(k)*y(k-1) + b1(k)*u1(k-2) + b2(k)*u2(k-3)
    with RW models for both parameters (NVR=0.001)
```

```
dtfmopt  Hyper-parameter estimation for DTFM
[nvr,opts,parse]=dtfmopt(y,u,nn,TVP,meth,nvrc,nvr0,opts,ALG,tab,P0)
                         1 2 3   4   5    6    7    8    9  10  11

  y: Time series (*)
  u: Input (*)
  nn: Model structure [na,nb(1:nu),nd(1:nu)] ([1 1 0])
  TVP: Model type for each TVP (0-RW, 1-IRW) (0)
  meth: Estimation method ('ml')
       'ml': Maximum Likelihood
       'f#': Sum of squares of the #-step-ahead forecasting errors
  nvrc: Constraints for each NVR (-2)
        -2: Free estimation
        -1: Constrained estimation (all parameters with nvrc=-1 are equal)
       >=0: NVR constrained to this value (it is not estimated)
  nvr0: Initial NVR hyper-parameters (0.0001)
  opts: Optimisation options, see help for OPTIMSET (or FOPTIONS for ALG=3)
  ALG: Optimisation algorithm: 0=fminsearch, 1=fminunc, 2=lsqnonlin (not ML) (0)
         ALG=3 uses older FMINS based optimisation with interrupt button
  tab: Display: 0=none, 1=tabulate results, 2=update window (2)
  P0: Initial P matrix (1e5)

  nvr: Estimated NVR hyper-parameters
  opts: Returned optimisation options. Type 'help foptions' for details
  parse: Standard Error of hyper-parameters (omit to reduce computation time)

  Example: dtfmopt(y, [u1 u2], [1 1 1 2 3], 0, [], [0 -2])
    transfer function type model y(k) = a1(k)*y(k-1) + b1(k)*u1(k-2) + b2(k)*u2(k-3)
    with a RW model for all parameters and a1 assumed constant (NVR fixed at zero)
```

1.10 The dhr and dhropt tools in CAPTAIN

```
dhr  Dynamic Harmonic Regression analysis

  [fit,fitse,tr,trse,comp,e,amp,phs,ts,tsse,y0,dhrse,Xhat,Phat,ers,ykk1]...
      = dhr(y,P,IRWharm,NVR,alpha,P0,x0,smooth,ALG,Int,IntD,iout,pinv)
            1 2    3     4    5   6  7    8     9  10   11   12   13
  y: Time series (*)
  P: Periodic components; set P(1)=0 to include a trend (*)
  TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW, 2-Trigonommetric) (0)
         (for LLT use RW and IRW trends simultaneously)
  nvr: NVR hyper-parameters (0)
  alpha: alpha parameters; set alpha=1 for RW or IRW model (1)
  P0: Initial P matrix (1e6)
  x0: Initial state vector (0)
  sm: Smoothing on (1-default) or off (0-saves memory)
  ALG: Smoothing algorithm: P (0) or Q (1-default)
  Int: Vector of variance intervention points (zeros(length(y),1))
  IntD: Variance intervention matrix diagonal (1e2 for trend level)
  iout: (integer) intermediate results on (1) or off (0-default)
  pinv: Pseudoinverse on (1) or off (0-default)
  fit: Model fit
  fitse: Standard errors of the fit
  tr: Trend (when trend is specified as IRW, second column is trend slope)
  trse: Standard errors of the trend (as above: slope standard errors)
  comp: Harmonic components
  e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
  amp: Amplitude of harmonic components
  phs: Phase of harmonic components (-pi,pi)
  ts: Total seasonal component
  tsse: Standard errors of the total seasonal component
  y0: Interpolated data
  dhrse: Standard errors of all the components (not grouped)
  Xhat: Parameters X(k/k) when just KF is run, X(k/N) when both KF/FIS
  Phat: P-matrix P(k/k) when just KF is run, P(k/N) when both KF/FIS
  ers:  Normalised error as returned by kalmsmo: 1+H(k)'*P(k/N)*H(k)
  ykk1: KF one step ahead predictions
  Example: dhr(y, [0 12./(1:6)], [1 0], [0.001 0.01], [0.95 1])
    SRW trend model (NVR=0.001, alpha=0.95) together with 6 periodic
    components (12 and harmonics) each modelled with a RW (NVR=0.01)
 %%%%%%%%%%%%%%%%%%%%%%%%
  dhropt  Hyper-parameter estimation for DHR
  [nvr,alpha,opts,amp,parse]=...
           dhropt(y,P,TVP,meth,nvrc,alphac,nvr0,alpha0,opts,ALG,tab,tf,Int)
                  1 2  3   4    5     6     7     8     9   10  11  12 13
  y: Time series (*)
  P: Periodic components; set P(1)=0 to include a trend (*)
  TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW, 2-Trigonommetric) (0)
         (for LLT use RW and IRW trends simultaneously)
  meth: Estimation method (24)
          meth>0: Frequency domain optimisation based on the AR(meth) spectrum
          meth<0: ML in time domain with abs(meth) used as the order of the AR
                  spectrum to estimate initial conditions for hyper-parameters
          meth='f#': Sum of squares of the #-steps-ahead forecasting errors with
                     initial conditions from the frequency domain using AR(24)
          meth='f# n': As above but specifying AR(n) to estimate initial conditions
  nvrc: Constraints for each NVR (-2)
      -2: Free estimation
      -1: Constrained estimation (all parameters with nvrc=-1 are equal)
     >=0: NVR constrained to this value (it is not estimated)
  alphac: Constraints for each alpha (-2, -1, or >=0 as for nvrc) (1)
  nvr0: Initial NVR hyper-parameters (0)
  alpha0: Initial alpha hyper-parameters (1)
  opts: Optimisation options, see help for OPTIMSET (or FOPTIONS for ALG=3)
  ALG: Optimisation algorithm: 0=fminsearch, 1=fminunc, 2=lsqnonlin (not ML) (2)
         ALG=3 uses older FMINS based optimisation with interrupt button
  tab: Display: 0=none, 1=tabulate results, 2=update window and frequency plot (2)
  tf: User supplied frequency information (1032)
        scalar: Number of subdivisions of frequency axes
        vector: Frequency axis (between 0-0.5)
        matrix (nx2): Frequency axis (1st column) and AR spectrum amplitude
  Int: Vector of variance intervention points (0)
  nvr: Estimated NVR hyper-parameters
  alpha: Estimated alpha hyper-parameters
  opts: Returned optimisation options. Type 'help foptions' for details
  amp: Spectra [t, amp, ampm], e.g. use semilogy(t, amp, t, ampm)
         t: Frequency axis at which the spectra are evaluated
         amp: Empirical spectrum
         ampm: Fitted model spectrum
  parse: Standard Error of hyper-parameters (omit to reduce computation time)
  Example: dhr(y, [0 12./(1:6)], [1 0], 24, -2, [-2 1])
    optimise for a SRW trend, together with 6 periodic components
    (12 and harmonics) each modelled with a RW (alpha fixed at 1)
```

1.11 The darx and darxopt tools in CAPTAIN

```
darx  Dynamic AutoRegressive multi-eXogenous variables analysis

  [tfs,fit,fitse,par,parse,comp,e,y0]=darx(y,u,nn,TVP,nvr,alpha,P0,x0,sm,ALG)
                                          1 2 3   4   5    6   7  8  9  10

  y: Time series (*)
  u: Input (*)
  nn: Model structure [na,nb(1:nu),nd(1:nu)] ([1 1 0])
  TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW) (0)
  nvr: NVR hyper-parameters (0)
  alpha: alpha hyper-parameters for SRW model (1)
  P0: Initial P matrix (1e5)
  x0: Initial state vector (0)
  sm: Smoothing on (1-default) or off (0-saves memory)
  ALG: Smoothing algorithm: P (0) or Q (1-default)

  tfs: Transfer function (simulation) output
  fit: Model fit
  fitse: Standard error of the fit
  par: Parameter estimates
  parse: Standard errors of parameters
  comp: Model components for y(k-1), ..., u(k-1), ...
  e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
  y0: Interpolated data

  Example: darx(y, [u1 u2], [1 1 1 2 3], 0, 0.001)
    transfer function type model y(k) = a(k)*y(k-1) + b1(k)*u1(k-2) + b2(k)*u2(k-3)
    with RW models for both parameters (NVR=0.001)
```

```
darxopt  Hyper-parameter estimation for DARX

  [tfs,fit,fitse,par,parse,comp,e,y0]=darx(y,u,nn,TVP,nvr,alpha,P0,x0,sm,ALG)
                                          1 2 3   4   5    6   7  8  9  10

  y: Time series (*)
  u: Input (*)
  nn: Model structure [na,nb(1:nu),nd(1:nu)] ([1 1 0])
  TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW) (0)
  nvr: NVR hyper-parameters (0)
  alpha: alpha hyper-parameters for SRW model (1)
  P0: Initial P matrix (1e5)
  x0: Initial state vector (0)
  sm: Smoothing on (1-default) or off (0-saves memory)
  ALG: Smoothing algorithm: P (0) or Q (1-default)

  tfs: Transfer function (simulation) output
  fit: Model fit
  fitse: Standard error of the fit
  par: Parameter estimates
  parse: Standard errors of parameters
  comp: Model components for y(k-1), ..., u(k-1), ...
  e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
  y0: Interpolated data

  Example: darx(y, [u1 u2], [1 1 1 2 3], 0, 0.001)
    transfer function type model y(k) = a(k)*y(k-1) + b1(k)*u1(k-2) + b2(k)*u2(k-3)
    with RW models for both parameters (NVR=0.001)
```

1.12 The dlr and dlropt tools in CAPTAIN

```
dlr  Dynamic Linear Regression analysis

[fit,fitse,par,parse,comp,e,y0]=dlr(y,z,TVP,nvr,alpha,P0,x0,sm,ALG)
                                   1 2  3   4    5   6  7  8   9

y: Time series (*)
z: Regressors (*)
TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW) (0)
nvr: NVR hyper-parameters (0)
alpha: alpha hyper-parameters for SRW model (1)
P0: Initial P matrix (1e5)
x0: Initial state vector (0)
sm: Smoothing on (1-default) or off (0-saves memory)
ALG: Smoothing algorithm: P (0) or Q (1-default)

fit: Model fit
fitse: Standard error of the fit
par: Parameter esfSEtimates
parse: Standard errors of parameters
comp: Linear components
e: Normalised innovations; use e=e(~isnan(e)) to remove NaNs
y0: Interpolated data
Ph: Complete stacked P matrices

Example: dlr(y, [ones(size(u)) u], [0 1], 0.001, [0.95 1])
  regression type model y = c1 + c2(t)*u, with an AR(1) model for
  c1 (alpha=0.95) and an IRW model for c2 (NVR=0.001 in both cases)

See also dlropt, fcast, stand, dhr, dhropt, sdp
```

```
dlropt  Hyper-parameter estimation for DLR

[nvr,alpha,opts,parse]=dlropt(y,z,TVP,meth,nvrc,alphac,nvr0,alpha0,opts,ALG,tab,P0)
                              1 2  3   4    5     6     7     8     9   10  11  12

y: Time series (*)
z: Regressors (*)
TVP: Model type for each TVP (0-RW/AR(1), 1-IRW/SRW) (0)
meth: Estimation method ('ml')
    'ml': Maximum Likelihood
    'f#': Sum of squares of the #-step-ahead forecasting errors
nvrc: Constraints for each NVR (-2)
      -2: Free estimation
      -1: Constrained estimation (all parameters with nvrc=-1 are equal)
     >=0: NVR constrained to this value (it is not estimated)
alphac: Constraints for each alpha (-2, -1, or >=0 as for nvrc) (1)
nvr0: Initial NVR hyper-parameters (0.0001)
alpha0: Initial alpha hyper-parameters (1)
opts: Optimisation options, see help for OPTIMSET (or FOPTIONS for ALG=3)
ALG: Optimisation algorithm: 0=fminsearch, 1=fminunc, 2=lsqnonlin (not ML) (0)
       ALG=3 uses older FMINS based optimisation with interrupt button
tab: Display: 0=none, 1=tabulate results, 2=update window (2)
P0: Initial P matrix (1e5)

nvr: Estimated NVR hyper-parameters
alpha: Estimated alpha hyper-parameters
opts: Returned optimisation options. Type 'help foptions' for details
parse: Standard Error of hyper-parameters (omit to reduce computation time)

Example: dlropt(y, [ones(size(u)) u], 0, [], [0 -2])
  regression type model y = c1 + c2(t)*u, with an RW model for
  both parameters and c1 assumed constant (NVR fixed at zero)
```

1.13 The rivcdd tool in CAPTAIN

This is the least satisfactory tool in CAPTAIN because it utilizes a sub-optimal iterative routine derived from a similar sub-optimal routine rivdd for discrete-time models developed many years ago There is no general proof that these iterations will converge and, indeed, an example of very slow convergence is encountered in chapter 2 of the book. An alternative is to apply the rivcbj tool and use its common denominator results as the starting estimate of the model in the Matlab tfest routine.. However, a new rivcbjdd tool that will replace rivcdd is now undergoing β testing and is available in the current version of CAPTAIN.

```
 rivcdd  Estimate continuous-time MISO TF with different denominators
        using a backfitting algorithm that calls RIVCBJ at each iteration

[MM,MN,DD,stats,e,eef,var,Ps,Pn,y0,AH,AHse,A,B,C,D,Acd,Bcd]=...
                                              rivcdd(z,nn,flags,nnr,CC)

z: I-O data, [Y,U1,...,Um] where Y and Ui are column vectors (*)
nn: Component model structures where each of m rows represents a
    SISO model associated with each input: [na, nb, ntd, nc, nd] (*)
      na: denominator, nb: numerator, ntd: time delay
      nc: noise model denominator (AR), nd: numerator (MA)
flags: Additional parameters vector [Nit, Ni, dt, ddt, cd, Rc]
       Missing value or -1 implies use default value in brackets
         (1) Nit<1: convergence criterion to compare with quadratic norm
             of parameter vector change between backfitting iterations
             Nit>=1: Number of backfitting iterations (default 1e-4)
         (2) Ni<1: convergence criterion to compare with quadratic norm
             of parameter vector change between RIV iterations (1e-4)
             Ni>1: Number (maximum) of SRIV iterations
         (3) dt: Sampling interval for continuous-time estimation (1)
         (4) ddt: Sampling for initial discrete time estimation (1)
         (5) cf: Constant (1) or adaptive (0-default) pre-filter flag
         (6) Rc: Block (0-default) or recursive algorithm (1-slower)
nnr: model structure for backfitting starting estimate (1)
       nnr: common denominator model [na,nb(1:nu),ntd(1:nu),nc,nd]
            (e.g. use RIVCBJID to find suitable values for nnr)
              na: denominator, nb: numerators, ntd: time delays
              nc: noise model denominator (AR), nd: numerator (MA)
       1: automatically generate common denominator model nnr from nn
            (where na is the sum of all the SISO models)
       0: use SISO models based on nn above
CC: Prefilter polynomial parameter (1)
     Polynomial: Polymomial with order 'na'
     Scalar <=0: Polynomial created automatically from pole value
                 e.g. zero to utilise multiple integrators, <0 for
                 stable filter (chosen so that 1/C roughly matches
                 the bandwidth of the system being estimated)
     1: Estimate discrete-time filter and convert to continous-time
        Normally C<=0 is best for fast and C=1 for coarse sampled data
        (or, if fast sampled, can use C=1 with ddt set at a more coarse
        subsampling rate for initial discrete-time estimation)

MM: Estimated model object (SID Toolbox required) or empty
MN: ARMA noise model (SID Toolbox required) or empty
DD: Data object (SID Toolbox required) or empty
stats: Statistics [cond(P),YIC,RT2,BIC,S2,o2,EVN,Ybar,RT2c,AIC]
e: System model output errors (y=fit+e)
eef: Residuals from noise model (stochastic model residuals)
var: Variance of residuals: eef if noise model present, else e (SRIV)
Ps: Covariance matrix for I-O parameter estimates
Pn: Covariance matrix for ARMA noise model parameter estimates
y0: Interpolated data
AH,AHse,PH: recursive estimates (if requested) of I-O component
  AH (Npars x Nsamp): parameter estimate vectors at each data point
  AHse (Npars x Nsamp): standard errors for the above
A: denominator polynomials for each input as row vector
B: numerator polynomials for each input as row vector
C: ARMA noise denominator polynomial
D: ARMA noise numerator polynomial
Acd: common denominator for backfitting starting estimate (when nnr~=0)
Bcd: numerator polynomials associated with Acd

Example: M=rivcd([y u1 u2 u3], [2 2 0 1 1; 1 2 0 1 1 ; 2 2 1 1 1])
  output y and inputs u1, u2 and u3, estimate the following MISO model
  TF1*u1+TF2*u2+TF3*u3+TF4*e where TF1, TF2 and TF3 are defined by each
  row of the 2nd input argument, and TF4 is a 1st order ARMA noise model
```

1.14 The irwsm and irwsmopt tools in CAPTAIN

The irwsm tool is a very useful practical device for quickly smoothing time series using the FIS algorithm. It is easy to apply with the *Noise-Variance Ratio* (NVR) hyper-parameter either optimized using the associated irwsmopt tool or specified by trial and error. The latter is required because the optimized NVR can sometimes not provide the kind of smoothing required in a particular situation where optimization is not justified, either too much or too little smoothing. It is then for the user to decide.

```
[ys,deriv,fitse,trse,derivse,y0,PkN,ers,ykk1,er]=irwsm(y,TVP,nvr,Int,dt,delt)

 y: Time series (*)
 TVP: Model type (RW=0, IRW=1, DIRW=2) (1)
 nvr: NVR hyper-parameter (1605*(1/(2*dt))^4)
 Int: Vector of variance intervention points (0)
 dt: decimation rate (1)
       with dt>1 only the following returned arguments
       are decimated: ys, deriv, fitse, trse, derivse, y0
 delt: sampling rate (1)

 ys: Decimated (or simply smoothed if dt=1) series
 deriv: Derivatives
 fitse: Standard error of model fit (including observation noise)
 trse: Trend (state) standard error
 derivse: standard error of the remaining states (trend derivs.)
 y0: Interpolated data
 PkN: Full smoothed covariance matrix info
 ers: Smoothed error norm. factor
 ykk1: One-step-ahead predictions of the trend
 er: KF error normalisation factor
```

```
[nvr,opts,parse] = irwsmopt(y,TVP,meth,Int,opts,ALG)

 y: Time series (*)
 TVP: Model type (RW=0, IRW=1, DIRW=2) (1)
 meth: Estimation method ('ml')
     'ml': Maximum Likelihood
     'f#': Sum of squares of the #-step-ahead forecasting errors
 Int: Vector of variance intervention points (0)
 opts: Optimisation options, see help for OPTIMSET (or FOPTIONS for ALG=3)
 ALG: Optimisation algorithm: 0=fminsearch, 1=fminunc, 2=lsqnonlin (not ML) (0)
        ALG=3 uses older FMINS based optimisation with interrupt button

 nvr: Estimated NVR hyper-parameters
 opts: Returned optimisation options. Type 'help foptions' for details
 parse: Standard Error of hyper-parameters (omit to reduce computation time)
```

1.15 The ivarma and ivarmaid tools in CAPTAIN

These tools are used quite lot in this book, the first to estimate a specified order ARMA noise model and the latter to identify the order of a noise model from a range of specified ARMA models (using ivarma). Note that this is an older tool and the SIC here is referred to as BIC (Bayesian Inf. Criterion).

```
[C,D,Pn,resvar,eef,R2,SIC,BIC,FPE,AR,AH,AHse,PH]=ivarma(n,nn,nar,nit,rec)

Or for SID object code output (SID Toolbox required)
  M = ivarma(n,nn,nar,nit,rec)

n: Noise time series (*)
nn: ARMA model order: e.g. [2 1] (*)
nar: Order of AR model used to start algorithm (default AIC or 50)
nit: Number of iterations (10)
rec: Recursive (1) or en bloc (default - 0)

C: ARMA denominator polynomial when called with multiple output arguments
   If only one output argument: provides model structure of SID type
   (if SID Toolbox exists) else a structure with the same fields as SID
D: ARMA numerator polynomial
Pn: Covariance matrix
resvar: Residiual variance
eef: Residual series
R2: Coefficient of Determination
AIC: Akaike Inf. Criterion
SIC: Swartz Inf. Criterion
FPE: Final Prediction Error
AR: High oder AR model
AH: Recursive parameter estimates
AHse: Standard errors on AH
PH: Covariance Matrices for AH
```

```
[p,q,C,D,Pcc,resvar,eef,RR]=ivarmaid(n,nnr,nar,sortcrit,gm)

n: Noise time series (*)
nnr: Range of ARMA model orders ([1 1 5 5])
nar: Order of AR model used to start algorithm (50)
sortcrit: Criterion for sorting: 1=SIC; 2=AIC; 3=FPE
gm: ARMA estimation algorithm (0)
       0: IVARMA (CAPTAIN)
       1: PEM (rather quicker that IVARMA but requires SID Toolbox)

p: AR order
q: MA order
C: AR poly
D: MA poly
Pcc: Covariance matrix
resvar: Residual variance
eef: Residuals
RR=[i,j,AIC,SIC,FPE,cov(eef),R2]
```

Note in the above that `IVARMA` and `PEM` can sometimes yield different results.