```
(* ----- read in packages ----- *)

<< Statistics`LinearRegression`
<< Statistics`NormalDistribution`
<< Statistics`HypothesisTests`

<< "timeseri\\timeseri.m"
<< "timeseri\\datasmoo.m"
<< "timeseri\\userfunc.m"

SetDirectory ["e:\\Geodesy & Math\\Analyza CR"];

(* ----- settings ----- *)

SetOptions[ListPlot, PlotJoined → True, PlotRange → All, DisplayFunction → Identity];

Off[General::spell1];

(* ----- user functions ----- *)

fShow[plot__, o___] := Show[plot, DisplayFunction :→ $DisplayFunction];

(* ----- read in the data ----- *)

data = ReadList["bor1.dat", Table[Number, {3}]];

mat = Transpose[data];

x1 = mat[[1]];
y1 = mat[[2]];
z1 = mat[[3]];

Show[GraphicsArray[{gd`x1 = ListPlot[x1],
                    gd`y1 = ListPlot[y1]}]];
```



```
n = Length[x1]

730

μ = Mean /@ {x1, y1}
        σ2 = Variance /@ {x1, y1};
σ = StandardDeviation /@ {x1, y1}

{-0.0143014, 0.0849315}

{8.68534, 13.1996}


(* --- removin' linear trend --- *)
```

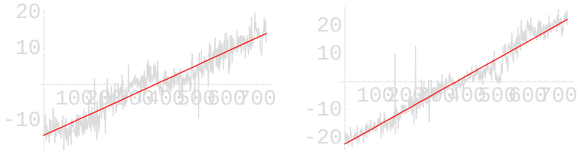
```
Regress @@@ {  {x1, {t}, t},
               {y1, {t}, t}}
```

```
                          Estimate      SE             TStat         PValue
{{ParameterTable → 1      -14.227       0.212248       -67.0302      1.135516932586×10^-313 ,
                   t      0.0388857     0.000503078    77.2956       3.62020123304×10^-353
  RSquared → 0.891386, AdjustedRSquared → 0.891236, EstimatedVariance → 8.2046,
                          DF        SumOfSq      MeanSq       FRatio       PValue
  ANOVATable →   Model    1         49019.3      49019.3      5974.61      3.62020123304×10^-353 },
                 Error    728       5972.95      8.2046
                 Total    729       54992.2
                          Estimate      SE             TStat         PValue
 {ParameterTable → 1      -22.2415      0.213629       -104.113      1.91582035557×10^-439,
                   t      0.0610846     0.000506352    120.637       1.83549805546×10^-483
  RSquared → 0.95236, AdjustedRSquared → 0.952294, EstimatedVariance → 8.31173,
                          DF        SumOfSq      MeanSq       FRatio       PValue
  ANOVATable →   Model    1         120962.      120962.      14553.2      1.83549805546×10^-483 }}
                 Error    728       6050.94      8.31173
                 Total    729       127013.
```

```
(* x1 *)

regx = Regress[x1, {t}, t, RegressionReport → {BestFit, FitResiduals}];
regy = Regress[y1, {t}, t, RegressionReport → {BestFit, FitResiduals}];

trendx = BestFit /. regx;
trendy = BestFit /. regy;

resx = FitResiduals /. regx;
resy = FitResiduals /. regy;
```

```
General::spell : Possible spelling error: new symbol name "resy" is similar to existing symbols {regy, resx}.
```

```
gtx = Plot[trendx, {t, 0, n}, PlotStyle → {RGBColor[1, 0, 0]}, DisplayFunction -> Identity];
gty = Plot[trendy, {t, 0, n}, PlotStyle → {RGBColor[1, 0, 0]}, DisplayFunction -> Identity];

Show[GraphicsArray[{
            Show[ListPlot[x1], gtx],
            Show[ListPlot[y1], gty]}  ]];
```




```
(* now residuals become new timeseries x,y *)
x1 = resx; y1 = resy;
μ = Mean /@ {x1, y1}
        σ2 = Variance /@ {x1, y1};
σ = StandardDeviation /@ {x1, y1}
```

```
{1.43736×10^-15, -3.76997×10^-15}

{2.8624, 2.88103}
```

```
(* --- choosing the order of AR models --- *)

x1y1 = Table[ {x1[[i]], y1[[i]]} , {i, n}];
```

```
MatrixForm[CorrelationFunction[x1y1, 0]]
```

$$\left(\begin{pmatrix}1.\\0.185757\end{pmatrix}\ \begin{pmatrix}0.185757\\1.\end{pmatrix}\right)$$

```
MatrixForm[km = Flatten[CovarianceFunction[x1y1, 0], 1]]
```

$$\begin{pmatrix}8.18213 & 1.52978\\1.52978 & 8.28896\end{pmatrix}$$

$$\left\{\text{Det}[\text{km}],\ \sum_{i=1}^{2}\text{km}[\![i, i]\!]\right\}$$

```
{65.4811, 16.4711}
```

```
(ar1 = LevinsonDurbinEstimate[x1y1, pmax = 15]);

(* - from determinants of covariance matrices - *)

(km = Map[#[[-1]] &, ar1]);

dkm = Table[Det[km[[i]]], {i, pmax}]
```

```
{42.9252, 37.3619, 36.1237, 34.8593, 34.8126, 34.5717, 34.1008,
 33.5321, 33.1636, 32.5471, 32.4007, 31.9337, 31.7717, 31.365, 31.2916}
```

```
fShow[ListPlot[dkm]];
```



```
(* Determinants of covariance matrices from Durbin-
  Levinson modeling of univariate series (x1,y1) *)
Show[GraphicsArray[{
    ListPlot[Map[#[[-1]] &, LevinsonDurbinEstimate[x1, pmax]]],
    ListPlot[Map[#[[-1]] &, LevinsonDurbinEstimate[y1, pmax]]]} ]];
```



```
(* - from information criteria - *)
fVAIC[m_, n_, p_] := Log[dkm[[p]]] + 2 * (m + m^2 * p) / n
fVBIC[m_, n_, p_] := Log[dkm[[p]]] + Log[n] * (m + m^2 * p) / n
fVHQIC[m_, n_, p_] := Log[dkm[[p]]] + 2 * Log[Log[n]] * (m + m^2 * p) / n

(* Do[Print["p = ", p,"   AIC= ",fVAIC[2,n,p],
    "   BIC= ",fVBIC[2,n,p],"   HQI= ",fVHQIC[2,n,p]],{p,pmax}]; *)

{aic, bic, hqic} =
  Transpose[Table[{fVAIC[m, n, p], fVBIC[m, n, p], fVHQIC[m, n, p] }, {p, pmax}]] /. m → 2;
```

```
fShow[ListPlot /@ {aic, bic, hqic}];
```



```
fArgMin[vec_, numb_] := Ordering[vec]⟦Range[numb]⟧;
Map[fArgMin[#, 2] &, {aic, bic, hqic}] (* first 2 argmins of IC`s *)

{{10, 4}, {2, 3}, {4, 3}}


p = 4; (* most appropriate value chosen from the graphs above *)

(* u                variables *)
data = dat =
  regx = regy = trendx = trendy = gtx = gty = resx = resy = μ = σ = σ2 = ar1 = km = dkm = aic = bic = hqic =.

(* --- testing for nonlinearity --- *)

(* set new assignment /as both series make a multivariate timeseries y=[y1,y2]/ *)
y = x1y1;
y2 = y1;
y1 = x1;
 (x1 = x1y1 =.;)

d = 5;          (* /d - threshold lag (delay) *)
q = 0;          (* /q - order of AR for exogenous variables *)
h = Max[p, q, d]

5

k = Length[y[[1]]]     (* /k - dimension of time series "y" /number of modeled variables *)
v = 0;                  (* /v - dimension of exogenous variable time series "x" *)

2

m0 = 3 √n // Round   (* /m0 - starting point of the recursive least square estimation *)

81

(* rearrange data according to increasing values of threshold variable "z" *)

z = y1;
z`ar = z[[ (tz = Ordering[ z[[ Range[h + 1 - d, n - d] ]], n - h] + h - d ) ]];

y1`ar = y1[[ tz]];
y2`ar = y2[[ tz]];

Length /@ {y, z, tz, y1`ar}

{730, 730, 725, 725}
```

```
fShow[GraphicsArray[ {{ListPlot[y1`ar], ListPlot[y1]},
                       {ListPlot[y2`ar], ListPlot[y2]}}]];
```



```
(* Pure
   function: create i-th row of (1+pk+qv+...) dimensional information matrix <regressor X>
         /p - order of AR for n×k-dimensional time series y
    /q - order of AR for n×v-dimensional time series x (exogenous variables)
    /Example: fRIM[i, {y,x}, {p,q}] or [i, {y,x1,x2,...}, p] if p=q=... *)

fRIM[i_, tseries_, p_] := Module[{},
      pom[ii_, rad_, pp_] := Reverse[Take[rad, {ii - pp, ii - 1}]];
      Flatten[{1, Thread[f[i, tseries, p]] /. (f :> pom)}]
       ]

X_{tz+d}[i_] := fRIM[tz[[i]] + d, {y}, p];
γ_{tz+d}[i_] := y[[tz[[i]] + d]];

(* improving performance by saving data
  into temporary lists /unused elements are set to be zero *)
X_{tz+d} = Table[X_{tz+d}[j], {j, 1, n - h}];
γ_{tz+d} = Table[γ_{tz+d}[j], {j, 1, n - h}];
```

```
(* least square estimate Φ of equation y_tz(i)+d = X_tz(i)+d Φ + ϵ_tz(i)+d
    "m" must lie in { (1+kp+qv),... n-h } *)
Φ̂_m[m_] := Module[{ypom, XpomT, Xpom},
            ypom = Table[ϒ_tz+d[[i]], {i, 1, m} ];
            Xpom = Table[X_tz+d[[i]], {i, 1, m}];
            XpomT = Transpose[Xpom];
      Inverse[XpomT.Xpom].XpomT.ypom
            ]
```

General::spell : Possible spelling error: new symbol name "Xpom" is similar to existing symbols {pom, XpomT}.

```
Φ̂_m = PadLeft[Table[Φ̂_m[j], {j, m0, n - h}], n - h, {0}];

V_m[m_] := Inverse[Sum_{i=1}^{m} Outer[Times, X_tz+d[[i]], X_tz+d[[i]]]];
    V_m = PadLeft[Table[ V_m[j] , {j, m0, n - h}], n - h, {0}];
    (* computing time (n=435,p=4,m0=63,k=2,v=0) = 23 s *)

ê_tz+d[i_] := ϒ_tz+d[[i]] - X_tz+d[[i]].Φ̂_m[[i - 1]];          (* predictive residuals *)
    ê_tz+d = PadLeft[Table[ ê_tz+d[j] , {j, m0 + 1, n - h}], n - h, {0}];

η̂_tz+d[i_] := ê_tz+d[[i]] / (1 + X_tz+d[[i]].V_m[[i - 1]].X_tz+d[[i]])^(1/2);
(* standardized predictive residuals *)
    η̂_tz+d = PadLeft[Table[ η̂_tz+d[j] , {j, m0 + 1, n - h}], n - h, {0}];

(* least square estimate Ψ of equation η_tz(l)+d = X_tz(l)+d Ψ + w_tz(l)+d
    "m" must lie in { (1+kp+qv),... n-h } *)
Ψ = Module[{ypom, XpomT, Xpom},
            ypom = Table[η̂_tz+d[[i]], {i, m0 + 1, n - h} ];
            Xpom = Table[X_tz+d[[i]], {i, m0 + 1, n - h}];
            XpomT = Transpose[Xpom];
      Inverse[XpomT.Xpom].XpomT.ypom
            ];

% // MatrixForm
```

0.0321423	0.00527471
−0.0256836	0.018771
0.00465837	0.0384256
0.082	−0.0185242
0.0213739	0.0190878
−0.050683	−0.0183188
0.00346258	−0.0887042
0.0878427	−0.00358282
−0.00900207	0.0184206

```
ŵ_tz+d[ℓ_] := η̂_tz+d[[ℓ]] - X_tz+d[[ℓ]].Ψ;
    ŵ_tz+d = PadLeft[Table[ ŵ_tz+d[j] , {j, m0 + 1, n - h}], n - h, {0}];
```

$$S0 = \frac{1}{n - h - m0} \sum_{\ell=m0+1}^{n-h} \text{Outer}[\text{Times}, \hat{\eta}_{tz+d}[[\ell]], \hat{\eta}_{tz+d}[[\ell]]]$$

{{5.78925, −0.111831}, {−0.111831, 6.33406}}

$$S1 = \frac{1}{n - h - m0} \sum_{\ell=m0+1}^{n-h} \text{Outer}[\text{Times}, \hat{w}_{tz+d}[[\ell]], \hat{w}_{tz+d}[[\ell]]]$$

{{5.68213, −0.0950337}, {−0.0950337, 6.27852}}

```
(* test statistic *)
Cd = (n - h - m0 - (k p + v q + 1)) (Log[Det[S0]] - Log[Det[S1]])

17.3966

(* degrees of freedom *)
df = k (k p + v q + 1)

18

(* confidence level *)
cl = {0.95, 0.99};

(* χ2df for confidence level "cl" *)
chiq = Quantile[ChiSquareDistribution[df], cl]

{28.8693, 34.8053}

fLinearityQ[teststatistic_, quantile_] := Print[teststatistic < quantile]

Thread[f[Cd, chiq]] /. f → fLinearityQ;
(* gives answer, whether time series are linear or not*)

True

True

ChiSquarePValue[Cd, df]

OneSidedPValue → 0.496017


(* --- estimation --- *)

(* u                variables *)
m0 =
 y1`ar = y2`ar = z`ar = Xtz+d = ϒtz+d = Φ̂m = Vm = êtz+d = η̂tz+d = Ψ = ŵtz+d = S0 = S1 = Cd = df = cl = chiq = d =.

(* entry parameters for estimation *)
p = 4;
 q = 0;
 (* s=2 ; *) (* number of regimes *)
 h = Max[p, q];
 h

4 Null^3

dimXt = 1 + k p + v q

9

(* parameter grid *)

{rmin, rmax} = Map[Sort[z][[Round[#]]] &, {n * 10 / 100, n * 90 / 100}]
 (* min,max values of threshold "r" as 10 % percentiles of "z" *)

{-3.41914, 3.59666}
```

```
Show[ListPlot[z], Plot[{rmin, rmax}, {i, 1, n}, DisplayFunction → Identity],
 DisplayFunction → $DisplayFunction]
```

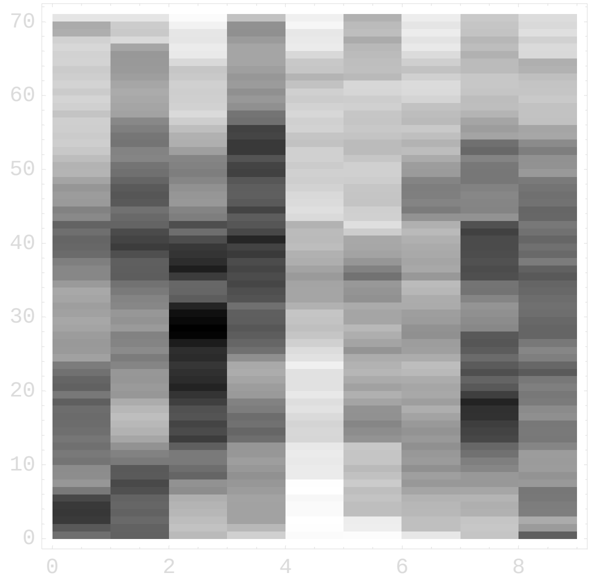


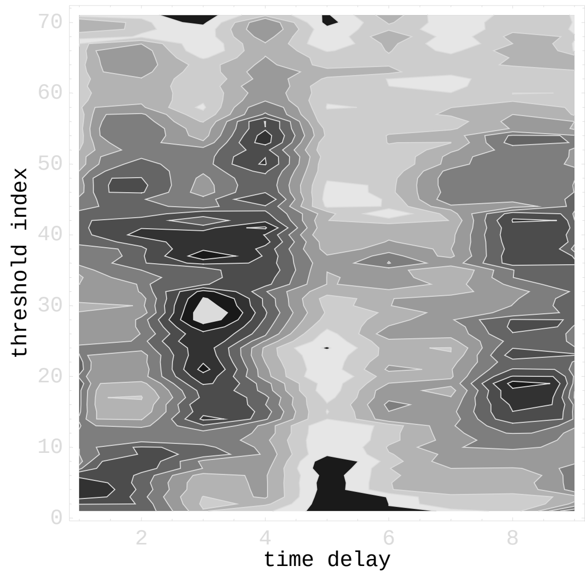
```
- Graphics -

(* group time-indices according to threshold intervals with respect to time lag
             /r - vector or threshold value
             /d - time lag
             /h - starting time index of time series estimation h=max (p,q,d)
             /z - threshold variable (time series)
                example: tr[0.3, 1, 4, y1]; or tr[{-0.5,2,5.1}, d, h, z] *)

tr[r_, d_, h_, z_] := Module[{vec, rsort, nint, tz, j, hh},
                          If[h < d, hh = d, hh = h];
                          tz = Ordering[ z[[Range[hh + 1 - d, n - d]]] ] + hh - d;
                          rsort = If[VectorQ[r], Sort[r], {r}];
                          nint = Length[rsort] + 1;
                          rsort = PadRight[rsort, nint, z[[Last[tz]]]];
                          vec = Table[{}, {nint}];
                          For[j = 1; i = 1, i ≤ nint, i++,
                           While[ If[j < n - (hh - 1), z[[ tz[[j]] ]] ≤ rsort[[i]], False],
                                vec[[i]] = Append[vec[[i]], tz[[j]] + d];
                                j ++ ]];
                          vec]

(* (pk+vq+1)-dimensional regressor Xt=(1,yt-1,...,yt-p,xt-1,...,xt-q) *)
X = PadLeft[Table[fRIM[ i, {y}, p], {i, h + 1, n}], n, { {} }];
```

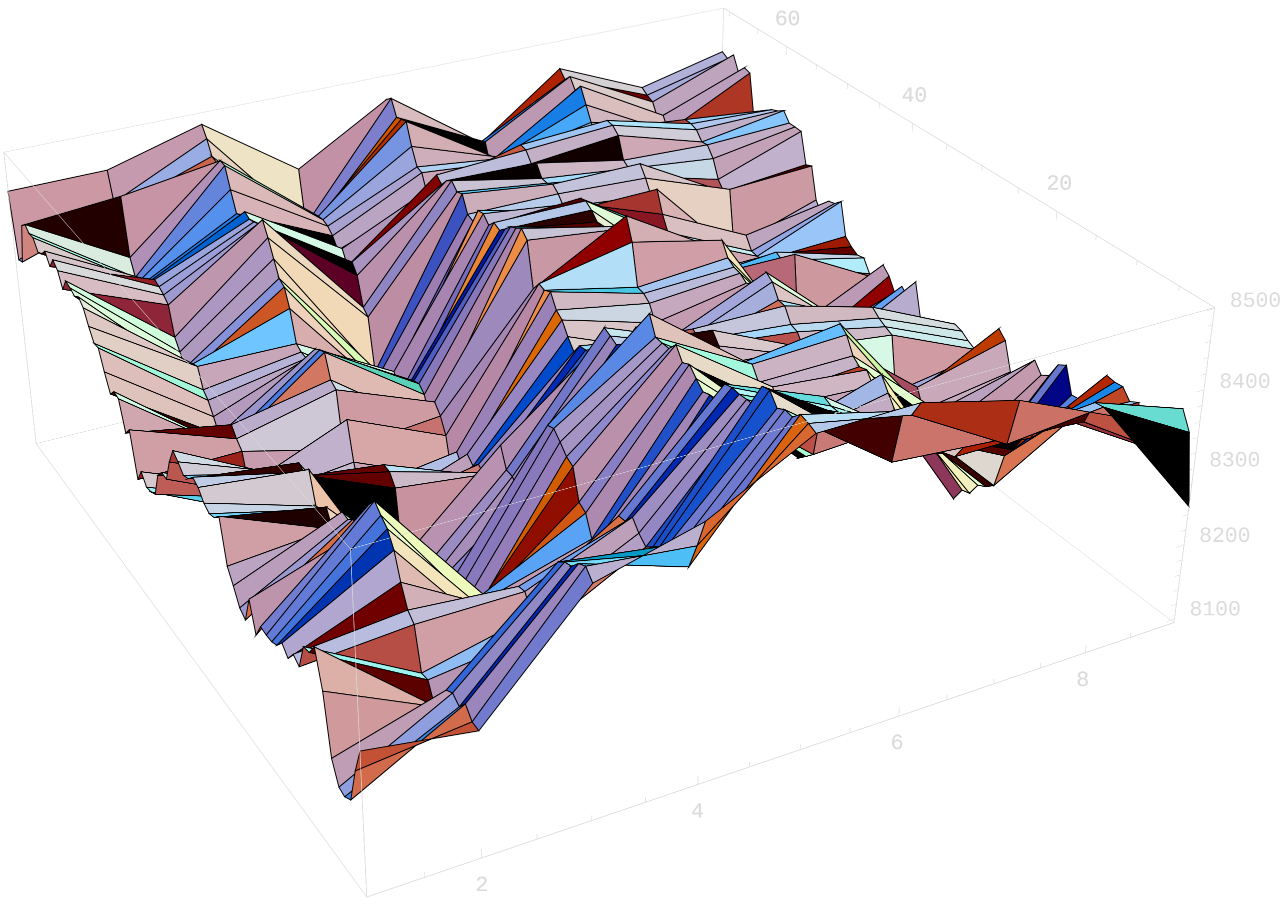
```
(* Sum of allregimes traces as function of threshold "r" and delay "d" *)
fS[r_, d_] := Module[{trpom, nint, Φpom, Σpom, Spom},
        trpom = tr[ r, d, h, z ];
        nint = Map[Length, trpom];
        Φpom = Σpom = Spom = Table[, {Length[nint]}];
        For[j = 1, j ≤ Length[nint], j ++,
                  Φpom[[j]] = Inverse[ Sum[Outer[Times, X[[ trpom[[j, i]] ]], X[[ trpom[[j, i]] ]]], {i, 1, nint[[j]]}] ].
        Sum[Outer[Times, X[[ trpom[[j, i]] ]], y[[ trpom[[j, i]] ]]], {i, 1, nint[[j]]}];
                  Σpom[[j]] = (Sum[Outer[Times, y[[ trpom[[j, i]] ]] - X[[ trpom[[j, i]] ]].Φpom[[j]],
          y[[ trpom[[j, i]] ]] - X[[ trpom[[j, i]] ]].Φpom[[j]]], {i, 1, nint[[j]]}]) / (nint[[j]] - dimXt);
                  Spom[[j]] = Tr[(nint[[j]] - dimXt) * Σpom[[j]]]];
        Plus @@ Spom];
```

```
General::spell : Possible spelling error: new symbol name "Spom" is similar to existing symbols {pom, Xpom}.
```

```
(* show row and column profile of a matrix *)
fRCshow[row_, col_, matrix_] :=
  fShow[GraphicsArray[{ListPlot[matrix[[row, All]]], ListPlot[matrix[[All, col]]]}]];

(* show 2 column profiles of a matrix *)
fCshow[col1_, col2_, matrix_] :=
  fShow[GraphicsArray[{ListPlot[matrix[[All, col1]]], ListPlot[matrix[[All, col2]]]}]];

(* find minima of matrix (!) and print first
  "numb" of them in the following form: {{row,col},value} *)

fMinMatrix[matrix_, numb_] := Module[{pom, pom1, pom2, n, m, fp, ip},
                                 {n, m} = Dimensions[matrix];
                                 pom = Ordering[Flatten[matrix]];
                                 pom1 = pom / m;
                                 fp[i_] := FractionalPart[pom1[[i]]];
                                 ip[i_] := IntegerPart[pom1[[i]]];
                                 pom2 =
    Table[If[fp[i] == 0, {ip[i], m}, {ip[i] + 1, fp[i] * m}], {i, n m}];
                                 Table[{pom2[[i]], matrix[[ pom2[[i, 1]], pom2[[i, 2]] ]]} , {i, numb}]
                                 ];


(* Rough grid r×d *)
r1 = Range[rmin1 = rmin, rmax1 = rmax, rstep1 = 0.1];
d1 = Range[dmin1 = 1, dmax1 = 9];
S`rd1 = Table[fS[r, d], {r, rmin1, rmax1, rstep1}, {d, dmin1, dmax1}];

GraphicsArray[{ ListDensityPlot[S`rd1, Mesh → False, DisplayFunction -> Identity],
                  ListContourPlot[S`rd1,
      FrameLabel → {"time delay", "threshold index"}, DisplayFunction -> Identity]}] // fShow;
                  ListPlot3D[S`rd1, ViewPoint → {-1.3, -2.4, 1.5},
   DisplayFunction → Identity] // fShow;
```







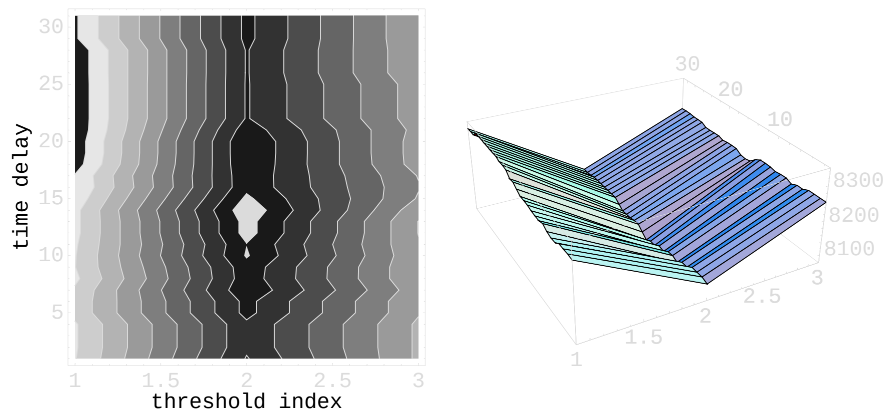

```
fRCshow[40, 3, S`rd1];
```

```
fMinMatrix[S`rd1, 10] // MatrixForm
Print["r = ", r1⟦%⟦1, 1, 1⟧⟧, ",     ", "d = ", d1⟦%⟦1, 1, 2⟧⟧]
```

```
{29, 3}  8104.4
{28, 3}  8111.84
{30, 3}  8120.55
{31, 3}  8130.28
{27, 3}  8148.4
{37, 3}  8152.29
{19, 8}  8157.92
{21, 3}  8157.98
{32, 3}  8159.27
{41, 4}  8164.78
```

```
r = -0.61914,    d = 3
```

```
(* Fine grid r×d *)
r2 = Range[rmin2 = -0.8, rmax2 = -0.5, rstep2 = 0.01];
d2 = Range[dmin2 = 2, dmax2 = 4];
S`rd2 = Table[fS[r, d], {r, rmin2, rmax2, rstep2}, {d, dmin2, dmax2}];

GraphicsArray[{ ListContourPlot[S`rd2,
     FrameLabel → {"threshold index", "time delay"}, DisplayFunction -> Identity],
                ListPlot3D[S`rd2, ViewPoint → {-1.3, -2.4, 1.5}, DisplayFunction → Identity]}] /.
  fShow;
```




```
fMinMatrix[S`rd2, 5] // MatrixForm
Print["r = ", r2⟦%⟦1, 1, 1⟧⟧, ",     ", "d = ", d2⟦%⟦1, 1, 2⟧⟧]
```

```
{14, 2}  8083.93
{12, 2}  8092.43
{13, 2}  8092.43
{15, 2}  8096.82
{10, 2}  8099.76
```

```
r = -0.67,    d = 3
```

```
r3 = Range[rmin3 = -0.8, rmax3 = -0.5, rstep3 = 0.01];
d3 = Range[dmin3 = 3, dmax3 = 3];
S`rd3 = Table[fS[r, d], {r, rmin3, rmax3, rstep3}, {d, dmin3, dmax3}];
```

```
ListPlot[Flatten[S`rd3]] // fShow;
```



```
fMinMatrix[S`rd3, 5] // MatrixForm
Print["r = ", r3[[ %[[1, 1, 1]] ]], ",      ", "d = ", d3[[ %[[1, 1, 2]] ]]]
Print["r = ", r3[[ %%[[2, 1, 1]] ]], ",      ", "d = ", d3[[ %%[[2, 1, 2]] ]]]
```

{14, 1}	8083.93
{12, 1}	8092.43
{13, 1}	8092.43
{15, 1}	8096.82
{10, 1}	8099.76

r = -0.67,   d = 3

r = -0.69,   d = 3

```
fΦΣy[p_, q_, r_, d_] := Module[{P, R, s, ni, H, TR, X, ϕ, σ, ym},
                          R = If[VectorQ[r], r, {r}];
                          s = Length[R] + 1;
                          P = If[VectorQ[p], p, Table[p, {s}]];
                          TR = tr[r, d, H = Max[p, q, d], z];
                          ni = Map[Length, TR];
                          ϕ = σ = Table[, {s}];
                          ym = Table[ Table[0, {s}],  {n}];
                          For[j = 1, j ≤ s, j++,
                           X = PadLeft[Table[fRIM[i, {y}, P[[j]]], {i, H + 1, n}], n, { {} }];
                           dimX = Length[X[[n]]];
                           ϕ[[j]] = Inverse[ Sum[Outer[Times, X[[ TR[[j, i]] ]], X[[ TR[[j, i]] ]]], {i, 1, ni[[j]]}] ].
     Sum[Outer[Times, X[[ TR[[j, i]] ]], y[[ TR[[j, i]] ]]], {i, 1, ni[[j]]}];

                           σ[[j]] = ( Sum[Outer[Times, y[[ TR[[j, i]] ]] - X[[ TR[[j, i]] ]].ϕ[[j]],
       y[[ TR[[j, i]] ]] - X[[ TR[[j, i]] ]].ϕ[[j]]], {i, 1, ni[[j]]}] ) / (ni[[j]] - dimX);
                           For[i = 1, i ≤ ni[[j]], i++,
                            ym[[ TR[[j, i]] - 1 ]] = (X[[ TR[[j, i]] ]].ϕ[[j]])];
                           ];
                          {ϕ, σ, ym}];

{Φ, Σ, y_m} = fΦΣy[4, 0, -0.67, 3];
```

```
MatrixForm /@ Φ
```

$$\left\{\begin{pmatrix} -0.878266 & 0.181902 \\ 0.33602 & 0.048456 \\ -0.0514867 & 0.454105 \\ -0.0325374 & 0.0637941 \\ 0.225029 & 0.0240707 \\ -0.0868232 & -0.0076824 \\ -0.062344 & 0.135737 \\ 0.0408783 & 0.0738319 \\ 0.0852954 & 0.136432 \end{pmatrix}, \begin{pmatrix} 0.389975 & 0.061375 \\ 0.313233 & 0.0854483 \\ -0.00532233 & 0.0712893 \\ 0.198924 & 0.0462444 \\ 0.0733865 & 0.309571 \\ -0.104012 & -0.00930218 \\ 0.193612 & 0.0381747 \\ 0.147542 & -0.0121817 \\ -0.0927181 & 0.0775122 \end{pmatrix}\right\}$$

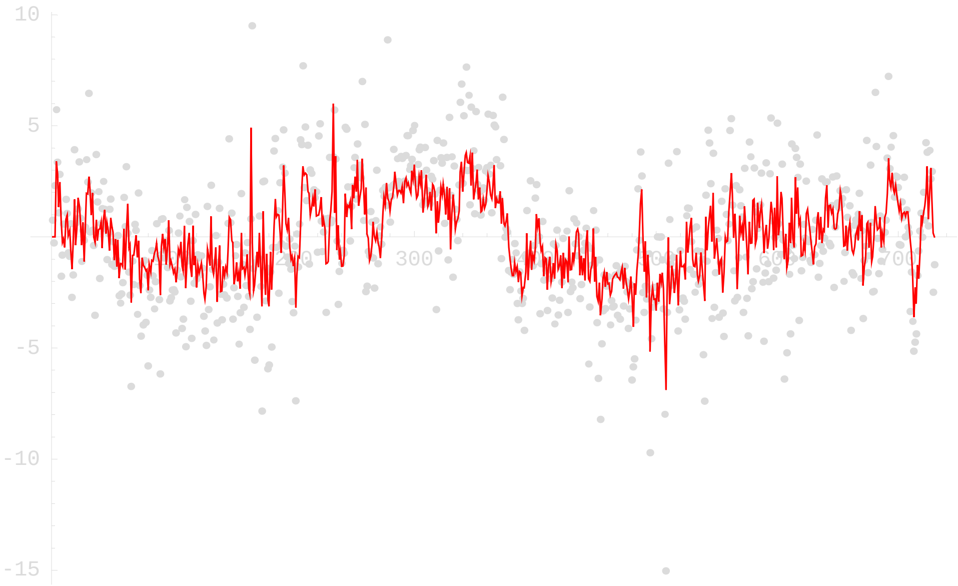
```
MatrixForm /@ Σ
```

$$\left\{\begin{pmatrix} 5.89322 & 0.0711409 \\ 0.0711409 & 5.7838 \end{pmatrix}, \begin{pmatrix} 5.31043 & 0.107268 \\ 0.107268 & 5.91262 \end{pmatrix}\right\}$$

$$\left\{\begin{pmatrix} 4.73640373323072\text{`} & -0.2866012438390284\text{`} \\ -0.2866012438390284\text{`} & 3.1942166324628998\text{`} \end{pmatrix}, \begin{pmatrix} 4.398913578842942\text{`} & -0.8975591691086565\text{`} \\ -0.8975591691086565\text{`} & 4.691562132201497\text{`} \end{pmatrix}\right\}$$

```
{{{4.7364, -0.286601}, {-0.286601, 3.19422}}, {{4.39891, -0.897559}, {-0.897559, 4.69156}}}
```

```
Show[ListPlot[y1, PlotJoined → False], ListPlot[ym⟦All, 1⟧,
   PlotStyle → {RGBColor[1, 0, 0], Thickness[0.002]}], DisplayFunction → $DisplayFunction];
```



```
Show[ListPlot[y2, PlotJoined → False], ListPlot[yₘ⟦All, 2⟧,
  PlotStyle → {RGBColor[1, 0, 0], Thickness[0.002]}], DisplayFunction → $DisplayFunction];
```