```
// T33stPre.sce
// STATE PREDICTION (with KALMAN FILTER)
// Experiments
// - change length of prediction np
// - change model parameters M,N,A,B
// - set different system and model covariances rwS,rvS and rwE,rvE
// - try lower stat-estimate covariance Rx
// ----------------------------------------------------------------------
exec("ScIntro.sce",-1), mode(0), getd()

nd=200;                                  // number of data                    // 1
np=5;                                                                         // 2
// SIMULATION                                                                 // 3
M=[.8 .1                                 // parameters of simulation          // 4
   .3 .6];                                                                    // 5
N=[.5 -.5]';                                                                  // 6
A=[.9 -.2];                                                                   // 7
B=0;                                                                          // 8
rwS=.1*eye(2,2);                          // noise covariances                // 9
rvS=.1;                                                                       // 10
x(:,1)=[-2 1]';                           // initial state                    // 11
ut=signal(nd,1);                         // input                             // 12
// time loop of simulation                                                    // 13
```

```
for t=2:(nd-np)
  x(:,t)=M*x(:,t-1)+N*ut(t)+rwS*rand(2,1,'n');
  y(t)  =A*x(:,t)+B*ut(t)+rvS*rand(1,1,'n');
end

// ESTIMATION
// initialization of estimation
rwE=.1*eye(2,2);                // state noise covariance
rvE=.1;                         // output noise vovariance
Rx=10*eye(2,2);                 // estimated state covariance
xE(:,1)=zeros(2,1);             // initial state
// loop for state estimation
for t=2:(nd-np)
  [xE(:,t),Rx,yp(t)]=..              // full Kalman filter
  Kalman(xE(:,t-1),y(t),ut(t),M,N,[],A,B,[],rwE,rvE,Rx);
  xp=xE(:,t);
  for i=1:np
    [nill1,nill2,nill3,nill4,xp,nill5]=.. // prediction only
    Kalman(xp,0,ut(t+i),M,N,[],A,B,[],rwE,rvE,Rx);
  end
  xP(:,t+np)=xp;                // remember prediction
end

```

```
// RESULTS                                                         // 37
s=(np+1):(nd-np);                                                  // 38
subplot(311),plot(s,x(1,s),s,xE(1,s))                              // 39
set(gcf(),"position",[700 100 600 500])                            // 40
set(gca(),'data_bounds',[1, nd -5 5])                              // 41
title('First state entry')                                         // 42
legend('state','estimate');                                        // 43
subplot(312),plot(s,x(2,s),s,xE(2,s))                              // 44
set(gca(),'data_bounds',[1, nd -5 5])                              // 45
title('Second state entry')                                        // 46
legend('state','estimate');                                        // 47
subplot(313),plot(s,y(s),s,yp(s)')                                 // 48
set(gca(),'data_bounds',[1, nd -5 5])                              // 49
title('Output')                                                    // 50
legend('output','estimate');                                       // 51
```

Description of the program

The program is practically identical with `T31stEst.sce` (state estimation), the only difference is the predictive loop in the rows 29–32. It calls the Kalman subroutine but as a result it takes only the state prediction x_p (not the full estimate which is corrected in the filtration part - see subroutine `Kalman`).

Thus, in each step, it takes filtered state (from the previous step - rows 27–28 and then it continues until the time $t+np$ but only with state prediction. In the row 33 the predicted state is remembered.