Time Series Analysis 2023
Lund University

Computer Exercise 3
Recursive Estimation and
Models with Time-Varying Parameters

This computer exercise treats recursive parameter estimation using Kalman filtering and recursive least squares. We attempt to model dynamic systems of both the SARIMA-type, having time-varying A and C polynomials, as well as to allow for ARMAX processes which have a synthetic input signal and time-varying B polynomial.

1 Preparations before the lab

Read Chapter 8 in the course textbook as well as this guide to the computer exercise. Answers to some of the computer exercise will be graded using the course's *Mozquizto* page, available at `https://quizms.maths.lth.se`. Ensure that you can access the system before the exercise and answer the preparatory questions as well as (at least) three of numbered exercise questions below *before the exercise*. These questions aim at allowing to check your implementation.

Before the computer exercise:

1. Express an AR(2) process on state space form and estimate the parameters of the process using a Kalman filter as specified in Sections 2.2 and 2.3.
2. Write a Matlab script that simulates the process u_t in Section 2.4 below. Let u_t be a Markov chain that switches slowly between two states, using $p_{11} = p_{22} = 7/8$ and $p_{12} = p_{21} = 1/8$. Hint: This is easy to do using a loop where you at each time instance change state according to the specified probabilities.

Note that you are expected to be able to answer detailed questions on your implementation. It should be stressed that a thorough understanding of the material in this exercise is important to be able to complete the course project, and we encourage you to discuss any questions you might have on the exercises with the teaching staff. This will save you a lot of time when you start working with the project! You are allowed to solve the exercise in groups of two, but not more. Please respect this.

2 Lab tasks

The computer program Matlab and the functions that belong to its System Identification Toolbox (SIT) will be used. In addition, some extra functions will also be used in this exercise. Make sure to download these functions and the required data files from the course homepage. You are free to use other programs, such as R or Python, but then need to find the appropriate functions to use on your own.

2.1 Recursive least squares estimation

You will begin by examining the recursive least squares (RLS) estimate of a time-varying AR process.

1. Load the data material `tar2.dat`, the data is an AR(2)-process with one time dependent parameter and the other one constant. The correct parameter trajectories are stored in the file `thx.dat`. Use `subplot` to plot the data and the parameter in the same figure.

2. Use the Matlab object `recursiveAR` to estimate the $A(z)$ polynomial recursively with the Matlab code

```
X = recursiveAR(2);
X.ForgettingFactor = 1;
X.InitialA = [1 0 0];
for kk=1:length(tar2)
    [Aest(kk,:),yhat(kk)] = step(X, tar2(kk) );
end
```

   Here, `Aest` is the estimated parameters, `yhat` is the estimate of y_t based on the estimated $A(z)$ polynomial and past values of y_t. Try different forgetting factors, using $\lambda = 1, 0.95, 0.9$. Plot the parameter estimates together with the true parameter. What effect does the value of λ have?

3. To choose λ, one option is to use the least squares estimate.

```
n=100;
lambda_line = linspace(0.85,1,n);
ls2   = zeros(n,1);
yhat = zeros(n,1);
for i=1:length(lambda_line)
     reset(X);
     X.ForgettingFactor = lambda_line(i);
     X.InitialA = [1 0 0];
     for kk=1:length(tar2)
          [~,yhat(kk)] = step(X, tar2(kk) );
     end
     ls2(i)=sum((tar2-yhat).^2);
end
plot(lambda_line,ls2)
```

QUESTION 1 *In Mozquizto, answer question 1.*

2.2 Kalman filtering of time series

A quite important drawback of the RLS estimate is that it should not be used to estimate MA parameters, and one should therefore not use it for MA or ARMA processes. We continue to again estimate the AR parameters from the previous section, but by using the Kalman filter that you have implemented in the preparatory exercises. Note that the Kalman implementation can be extended to also allow for MA coefficients.

We here make use of the example code given in Section 3. At first, ignore the part of the example code for the 2-step prediction. The code use data up to time $t-1$ to predict the state value $\hat{x}_{t|t-1}$, stored in the variable `xtt_1`, and then use this value to predict $\hat{y}_{t|t-1}$. The prediction error (also often termed the prediction residual) between $\hat{y}_{t|t-1}$ and y_t, i.e.,

$$\epsilon_{t|t} = y_t - \hat{y}_{t|t-1}$$

is then used to update the Kalman filter. Here, the prediction residual is stored in the variable `ehat`.

Proceed to complete the missing part of the code (ignore the part for the 2-step prediction). Then, to be able to check your implementation, set

```
Re    = [ .004  0; 0  0];
Rw    = 1.25;
Rxx_1 = 10*eye(2);
x_1   = [0  0]';
```

This will set the covariance matrix of the observation (`Re`) and measurement (`Rw`) noises, as well as the initial state vector and its covariance matrix. The large initial value for `Rxx_1` indicates that we have little confidence in the initial states `x_1`. Notice in particular that the way `Re` is selected reflects the assumption that the first parameter varies, whereas the second does not. Plot the resulting parameter estimates and notice the difference in convergence between the two parameters.

QUESTION 2 *In Mozquizto, answer question 2.*

This question strives to check that your implementation is correct. As you will use this as the basis for the following steps, as well as in the project, it is important that you get this to work properly. Ask the teaching staff to help you if you do not get the correct answer!

QUESTION 3 *In Mozquizto, answer question 3.*

What effect has the choice of `Rw` and `Re` for the parameter estimates? Did you manage to improve the estimation by using Kalman filtering instead of RLS? Can you reduce the sum of the squared residual (`ehat`) by tuning `Re` and `Rw`?

2.3 Using the Kalman filter for prediction

We now proceed to form a 2-step prediction using the Kalman filter. To do this, you need to complete the latter part of the example code. As it can be difficult to verify that the implementation is correct, we will use simulated data for this. To be able to check your implementation, use

```
rng(0);
N  = 10000;
ee = 0.1*randn(N,1);
A0 = [1 -.8 .2];
y  = filter(1, A0, ee);
Re = [1e-6 0; 0 1e-6];
Rw = 0.1;
```

We here select `Re` small to ensure that we get states that converge close to the true values (which is also why we select N so large).

In order to form $\hat{y}_{t+2|t}$, you first need to form $\hat{C}_{t+2|t}$, as

$$\hat{y}_{t+2|t} = \hat{C}_{t+2|t}\hat{x}_{t+2|t} = \hat{C}_{t+2|t}\hat{x}_{t|t}$$

Note that, in general, $\hat{x}_{t+2|t} \neq \hat{x}_{t|t}$. Why does this equality hold here?

Proceed to write down an expression for $\hat{C}_{t+2|t}$ and note that this will depend on $\hat{y}_{t+1|t}$. As a result, you will first need to estimate $\hat{y}_{t+1|t}$, then use this value to form $\hat{C}_{t+2|t}$, and then finally use this to form $\hat{y}_{t+2|t}$. Update the example code with the missing lines.

To examine the predictions when the filter has converged, use the following code to plot the last 100 samples of y_t, $\hat{y}_{t+1|t}$, and $\hat{y}_{t+2|t}$.

```
plot( y(end-100-2:end-2) )
hold on
plot( yt1(end-100-1:end-1),'g' )
plot( yt2(end-100:end),'r')
hold off
legend('y','k=1','k=2')
```

Notice the shifting to align the predictions with the corresponding data points. This is done as $\hat{y}_{t|t-2}$ is a prediction of y_t, using data up to $t-2$, and should thus be compared with this value (as a result, the last element of `yt1` should be zero; you are using the data up to $N-2$ to predict $\hat{y}_{N-1|N-2}$, but you do not form $\hat{y}_{N|N-1}$ as the loop only runs to $N-2$). Plot the estimated states and check that they converge properly.

QUESTION 4 *In Mozquizto, answer question 4.*

Again, it is important as you get your code to work properly as you will use this code in the project. Ask the teaching staff if you have problems!

Compute the sum of the squared prediction residual for the last 200 samples (why not all?). Can you improve the estimate, i.e., lower the sum of the squared prediction residual by tuning the choice of `Rw` and `Re`?

Show the plot of y_t, $\hat{y}_{t+1|t}$, and $\hat{y}_{t+2|t}$, as well of the predicted states, to the teaching staff. Why is it that, in general, $\hat{x}_{t+2|t} \neq \hat{x}_{t|t}$?

Be prepared to answer these questions when discussing with the examiner at the computer exercise!

It is worth noting that in case you are using the Kalman filter to predict an MA or an ARMA process, your `C` vector will contain earlier noise values, i.e., e_t, e_{t-1}, etc. These noise values are obviously not known, so one then use the corresponding one-step prediction errors in place of these, i.e., $\epsilon_{t|t}$, $\epsilon_{t-1|t-1}$, etc. These values are here stored in the variable `ehat`.

Important: Note that when using the Kalman filter for prediction, you *should not* use the polynomial division techniques discussed in the earlier computer exercise, as presented in Chapter 6 in the course textbook. This will not yield the correct estimates! Instead, predictions needs to be made by updating the states as indicated in Chapter 8.

2.4 Quality control of a process

In the quality control division at a factory, one has found that the process which is to be followed shows a drift like

$$x_t = x_{t-1} + e_t.$$

However, it is not possible to measure the quality variable x_t exactly, and one instead is limited to the observations

$$y_t = x_t + bu_t + v_t,$$

where the processes e_t and v_t are two mutually uncorrelated sequences of white noise, with the variances σ_e^2 and σ_v^2. Furthermore, b is a parameter. Assume that the external signal u_t is known.

Use the m-file written in the preparatory exercise for the computer exercise to simulate the process with the input signal u_t. Select $b = 20$, $\sigma_e^2 = 1$ and $\sigma_v^2 = 4$, but feel free to change these at will. Now consider x_t and b to be unknown, and use the Kalman filter you prepared and implement a filter that estimates b. Plot your estimates of the hidden states together with the true values.

How should the state space vector be chosen? How would you chose an initial value of `Re` and `Rw`? How can then proceed to fine-tune the filter?

Be prepared to answer these questions when discussing with the examiner at the computer exercise!

2.5 Recursive temperature modeling

The file `svedala94.mat` contains temperature measurements from the Swedish city Svedala, taken every four hours throughout 1994. The temperature can be modeled using a SARIMA$(2,0,2) \times (0,1,0)_6$ process, i.e., $A(z)\nabla_6 y_t = C(z)e_t$, where the $A(z)$ and $C(z)$ polynomials are of order 2.

1. Plot the temperature, differentiate the process to form $\nabla_6 y_t$ (use `filter` and remember those initial samples) and plot the differentiated temperature. To obtain months on the x-axis, use

```
T = linspace(datenum(1994,1,1),datenum(1994,12,31),...
  length(svedala94));
plot(T,svedala94);
datetick('x');
```

2. Use `armax` to estimate an ARMA(2,2)-process for the differentiated data. Do this (i) for the entire year, (ii) for January-March, and (iii) for June-August. Compare the different estimated parameters.

```
th = armax(ydiff,[2 2]);
th_winter = armax(ydiff(1:540),[2 2]);
th_summer = armax(ydiff(907:1458),[2 2]);
```

3. Use the `recursiveARMA` object to recursively estimate the $A(z)$ and $C(z)$ polynomials for the differentiated process.

```
X = recursiveARMA([2 2]);
X.InitialA = [1 th_winter.A(2:end)];
X.InitialC = [1 th_winter.C(2:end)];
X.ForgettingFactor = 0.99;
for k=1:length(ydiff)
    [Aest(k,:),Cest(k,:),yhat(k)] = step( X, ydiff(k) );
end
```

   It is possible to use the parameters estimated for January-March as initial values for the recursive estimation. Try some different forgetting factors. Compare with the non-recursive estimates obtained above. Use the following code to plot the results (also include a similar plot for `Cest` in subplot 313). Explain why and when the parameters vary.

```
subplot 311
plot(T,svedala94);
datetick('x')
subplot 312
plot(Aest(:,2:3))
hold on
plot(repmat(th_winter.A(2:end),[length(ydiff) 1]),'g:');
plot(repmat(th_summer.A(2:end),[length(ydiff) 1]),'r:');
axis tight
hold off
```

Given your results, do you find it likely that the `th_winter` and `th_summer` are different processes? Does the recursive process coincide with the non-recursive winter and summer processes?

2.6 Recursive temperature modeling, again

An alternative to the seasonal operator and recursive ARMA-models is to use a reasonable input signal that models both the mean value and the seasonal variation of the process. To illustrate this, we will once more examine the temperature in Svedala, now using an ARMAX-model.

1. Load the Svedala data, `load svedala94`, extract a suitable period of data, e.g. `y = svedala94(850:1100)`, and subtract the mean value.

2. We want to use a sinusoidal signal as input in an ARMAX model to avoid seasonal filtering. However, since the phase of the oscillation is unknown, we will use a combination of a sine and a cosine signals. Construct the external signal, and estimate the parameters by first forming the model

```
t = (1:length(y))';
U = [sin(2*pi*t/6) cos(2*pi*t/6)];
Z = iddata(y,U);
model = [3 [1 1] 4 [0 0]];
      %[na [nb_1 nb_2] nc [nk_1 nk_2]];
```

   Then, use `thx = armax(Z,model)` to estimate the coefficients for the sines and cosines. The coefficients are stored in `thx.b`. Plot `y`, together with the seasonal function `U*cell2mat(thx.b)`.

   What happens if you change the season? Why? Is the season well modeled using this method? How can it be improved?

3. A varying mean can be estimated by inclusion of a constant external signal, $u_t = 1$, if the parameters are estimated recursively. Using Kalman filter estimation allows the estimated coefficients to be expressed in state from

$$x_t = Ax_{t-1} + e$$

where $e \sim N(0, R_e)$ and x_t is the estimated coefficients at time t. Assuming that the parameter estimates are independent, the state covariance matrix R_e is diagonal, with diagonal elements corresponding to the noise variance for each of the states. Thus, if the ℓth diagonal index of R_e is non-zero, it is assumed to vary over time, otherwise not.

Hint: Order the states such that it contains the A coefficients, the B coefficients, and the C coefficients, in that order. Implement the code below and add the diagonal components of `Re` so that the only time dependent coefficient is the mean (choose this value to be 1).

```
U = [sin(2*pi*t/6) cos(2*pi*t/6) ones(size(t))];
Z = iddata(y,U);
m0 = [thx.A(2:end) thx.B' 0 thx.C(2:end)];
Re = diag([? ? ? ? ? ? ? ? ? ?] );
model=[3 [1 1 1] 4 0 [0 0 0] [1 1 1]];
[thr,yhat] = rpem(Z,model,'kf',Re,m0);
```

The `model` vector in `rpem` indicates the various model orders in the form

$$\Big[\text{na } [\text{nb}_1 \cdots] \text{ nc nd } [\text{nf}_1 \cdots] \ [\text{nk}_1 \cdots]\Big]$$

Here, the time delay for the input signals is necessary due to limitations in the command `rpem`.

4. The parts of the temperature that is due to the varying mean and the sine/cosine signals can now be reconstructed as

```
m = thr(:,?);
a = thr(end,4);
b = thr(end,5);
y_mean = m + a*U(:,1)+b*U(:,2);
y_mean = [0;y_mean(1:end-1)];
```

In the same plot, display `y` and `y_mean`. The values `a` and `b` are taken as the last values in `thr` as these are the final estimate of the constant coefficients.

5. We proceed to study the data from the entire year, recalling your earlier results from section 2.5. Due to stability issues the first values of the process should be close to zero.

```
y = svedala94;
y = y-y(1);
```

As before, estimate the process, letting only the mean value vary. Try finding a reasonable value for R_e.

Compare `y_mean` and `y`. Are these similar? If not, how and why do they differ? Is it a good estimate? If not, explain why and give a solution to make it better.

It is worth remarking that one difficulty with using functions such as `rpem` is that one cannot easily omit coefficients; this means that in cases that one has a model with only some active coefficients, say a_1 and a_7 for an AR(7), it is often preferable to use your own Kalman implementation to have full control of the estimation.

Hint: Do *not* use `rpem` to form your time-varying predicts in the project. It will almost surely result in incorrect predictions, or tempt you into using a simplistic model, or one that does not omit any coefficients, which is likely not a good idea. Instead, use a simulated signal mimicking your desired signal to ensure that your Kalman filter works...

3 Kalman Filter Outline

```
% Example of Kalman filter

% Simulate N samples of a process to test your code.
y = ?                 % Simulated data

% Define the state space equations.
A   = [? ?; ? ?];
Re  = [? ?; ? ?];          % State covariance matrix
Rw  = ?                    % Observation variance

% Set some initial values
Rxx_1 = ? * eye(2);        % Initial state variance
xtt_1 = [? ?]';            % Initial state values

% Vectors to store values in
Xsave = zeros(2,N);        % Stored states
ehat  = zeros(1,N);        % Prediction residual
yt1   = zeros(1,N);        % One step prediction
yt2   = zeros(1,N);        % Two step prediction

% The filter use data up to time t-1 to predict value at t,
% then update using the prediction error. Why do we start
% from t=3? Why stop at N-2?
for t=3:N-2
    Ct = [ ? ? ];          % C_{t|t-1}
    yhat(t) = ?            % y_{t|t-1}
    ehat(t) = ?            % e_t = y_t - y_{t|t-1}

    % Update
    Ryy  = ?               % R^{yy}_{t|t-1}
    Kt   = ?               % K_t
    xt_t = ?               % x_{t|t}
    Rxx  = ?               % R{xx}_{t|t}

    % Predict the next state
    xt_t1 = ?              % x_{t+1|t}
    Rxx_1 = ?              % R^{xx}_{t+1|t}

    % Form 2-step prediction. Ignore this part at first.
    Ct1       = [ ? ? ];   % C_{t+1|t}
    yt1(t+1)  = ?          % y_{t+1|t} = C_{t+1|t} x_{t|t}

    Ct2       = [ ? ? ];   % C_{t+2|t}
    yt2(t+2)  = ?          % y_{t+2|t} = C_{t+2|t} x_{t|t}

    % Store the state vector
    Xsave(:,t) = xt_t;
end
```