Chapter 1

Computer Laboratories

1.1 Computer Lab 1: Least Squares

1.2 Computer Lab 2: Time Series

1.3 Computer Lab 3: System Identification Toolbox

1.4 Computer Lab 4: Recursive Identification

1.4.1 Goals

In this computer laboratory we will study some features of recursive identification, as well as some practical aspects of system identification, including:

- System identification as a way of model approximation, when the model structure is not rich enough to describe the true dynamics.
- Estimation of physical parameters.

1.4.2 Recursive Identification

In recursive identification methods, measured input-output data are processed recursively (sequentially) as they become available, *i.e.*, the model is based on observations up to the current time. That is, when θ_t is the estimate based on the samples $(u_1, y_1), \dots, (u_t, y_t), (u_{t+1}, y_{t+1}), \dots$ the recursive scheme gives a sequence

$$\theta_1, \theta_2, \dots, \theta_t, \theta_{t+1}, \dots, \tag{1.1}$$

where each new estimate θ_t is computed whenever a new sample (u_t, y_t) becomes available. The understanding is that an estimate θ_n can make efficient (computational) use of the previous estimate θ_{n-1}. Recursive identification, also referred to as on-line or adaptive identification, is used in various areas, such as: adaptive systems, fault detection and parameter-tracking. Most off-line (batch) algorithms can be converted (exactly or approximately) into a recursive counterpart in a straightforward manner. This insight led to the various algorithms

RARX: Estimates recursively the parameters θ_t of an ARX model.

RPEM: Recursive version of a Prediction Error Method.

RPLR: Recursive method of a Prediction Error Method, using an approximate least squares formulation.

RIV: Recursive estimate of the parameters in the context of colored noise using an instrumental variable approach.

Those functions use a number of techniques which boil down to

- Kalman filter. Here the estimates are treated as the states of a linear system, and the Kalman filter is used to estimate those states given the observations.
- Gradient descent steps. This technique uses the gradient of the parameters with respect to the information in the new sample to update the old estimate.
- Newton-Raphson steps. This technique uses the gradient and the Hessian of the parameters with respect to the information in the new sample to update the old estimate.

The exact computations of those techniques are implemented in the MATLAB SI functions `RARX`, `RPEM` and `RPLS`.

1.4.3 A Simple Recursive Scheme

The following system is to be simulated:

$$(1-0.7q^{-1})y_t = 0.8q^{-1}u_t + (1-0.7q^{-1})e_t,$$

where u_t and e_t are uncorrelated white noise sequences with zero mean and unit variance. Identify the system using the following estimation methods: recursive least squares (RLS), recursive instrumental variable (RIV), recursive pseudo linear regression (RPLR) and recursive prediction error methods (RPEM). For RLS and RIV the model structure is

$$y_t + ay_{t-1} = bu_{t-1} + \varepsilon_t, \tag{1.2}$$

where $\theta = (a\ \ b)^T$. For RPLR and RPEM the model structure is

$$y_t + ay_{t-1} = bu_{t-1} + e_t + ce_{t-1} \tag{1.3}$$

where $\theta = (a\ \ b\ \ c)^T$. What can you say about the performance of the methods? Especially, give comments about consistency and convergence for the different methods. This example is implemented in the m-file `lab4a.m`.

1.4.4 Tracking Case

In many cases there is a need to modify the algorithms so that time-varying dynamics can be tracked. This may occur in cases where the dynamics underlying the signals vary over time, but also in cases where the plant changes operating point, requiring a modified approximate model. Two approaches for such modifications are:

- Change the loss function to be minimized. For instance, for the least squares method we can modify the loss function according to

$$V_t(\theta) = \sum_{s=1}^{t} \lambda^{t-s} \varepsilon_s^2 \tag{1.4}$$

  where λ is known as a forgetting factor. This means, as an example, that measurements that are older than $T_0 = 1/(1-\lambda)$ are included in the criterion with a weight that is $\approx 36\%$ of that of the most recent measurement (T_0 is called the memory time constant).

- Model the parameter variations as a state space model (*e.g.* a random walk), and apply Kalman filtering techniques.

The first approach will be illustrated further in this lab. The second approach is mentioned in the lectures.

1.4.5 Effect of the initial values

We will here study how the choice of the initial P matrix influences the estimate. The following system is to be simulated

$$y_t - 0.9y_{t-1} = 0.5u_{t-1} + e_t, \tag{1.5}$$

where u_t is a white binary sequence uncorrelated with the white noise sequence e_t, which has zero mean and variance 1. Identify the system using RLS and a first-order model

$$y_t + ay_{t-1} = bu_{t-1} + \varepsilon_t, \tag{1.6}$$

The P matrix is initialized by

$$P = \rho \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.7}$$

This example is implemented in the m-file `lab4a.m`. How does ρ influence the result?

1.4.6 Effect of the forgetting factor

Next, we will study how the forgetting factor affects the estimate. In particular we will study the problem of tracking a time varying system. Consider a first order ARX system which makes an abrupt change at time $t = 100$.

$$y_t - 0.8y_{t-1} = b_0 u_t + e_t \qquad b_0 = \begin{cases} 1.5 & t \leq 100 \\ 0.5 & t > 100 \end{cases} \tag{1.8}$$

Run the MATLAB function `lab4c` to identify the system using a RARX method with different forgetting factors. Describe the trade off which has to be made when choosing the forgetting factor. Study also if the estimated a parameter is negatively affected by a low forgetting factor.

1.5 Model Approximation

In this task we will examine how system identification can be viewed as a form of model approximation, when the system dynamics is too complex to belong the model structure considered. To simplify the study we consider a noise–free situation. Consider the following system, which has two distinct resonances.

$$G_0(q^{-1}) = \frac{1.0q^{-2} - 1.3q^{-3} + 0.8q^{-4}}{(1 - 1.5q^{-1} + 0.9q^{-2})(1 + 0.9q^{-2})} \tag{1.9}$$

Simulate the system using the input u_t as white binary noise of zero mean and unit amplitude. Use the function `gendata2`, listed in the Appendix, to generate 100 data points. Next, we will estimate the system above as a second order ARMAX model (denoted by $G(q^{-1}, \theta)$), by means of a prediction error method using prefiltered data

$$u^F(t) = F(q^{-1})u(t) \tag{1.10}$$

$$y^F(t) = F(q^{-1})y(t) \tag{1.11}$$

The filtering has the effect that we give emphasis to certain frequency ranges, depending on the choice of the filter. In fact, one can show that the parameter vector θ is determined as the minimizing element of

$$V(\theta) = \int |F(e^{i\omega})|^2 |G_0(e^{i\omega}) - G(e^{i\omega}, \theta)|^2 \Phi_u(\omega)\, d\omega \tag{1.12}$$

where $\Phi_u(\omega)$ is the spectral density of the input signal. This means that $|F(e^{i\omega})|^2\Phi_u(\omega)$ weights in what frequency region the deviation $|G_0(e^{i\omega}) - G(e^{i\omega,\theta})|$ will be penalized. Hence, by adjusting the prefilter, the user can directly influence the model fit in different frequency regions. The following tasks is to be considered:

- Identify the system for $F(q^{-1}) = 1$. This task can be carried out by running the MATLAB function `lab4d`. The result is evaluated in the frequency domain by drawing Bode plots of the model, the filter and the true dynamics.
- Let $F(q^{-1})$ be a sharp bandpass filter around one of the resonance frequencies of the system. Use a 5'th order Butterworth filter. The MATLAB function `filtdes` can be useful when designing the filter. Once the filter is designed (numerator and denominator stored in nn and dd, respectively. This is automatically done by `filtdes`) run `lab4d` to perform the estimation procedure.
- Repeat the previous task but for a filter with emphasize on the other resonance frequency.
- Repeat the previous task but let F be a low pass filter.

Summarize your findings.

1.6 Extra: Estimation of Physical parameters

Note: Do this exercise if you have the time. It might be useful in your future career to know how to estimate physical parameters.

In this section, we shall now see how system identification can be used to estimate physical parameters. As an example we will consider a DC motor. Its (continuous-time) transfer function from voltage to angular position, is given by

$$G(s) = \frac{K}{s(1+sT)} \tag{1.13}$$

where K and T are the parameters to be determined. By choosing the angular position and velocity as state variables, we can represent the motor in state space form as

$$\begin{cases} \dot{x} &= \begin{pmatrix} 0 & 1 \\ 0 & -1/T \end{pmatrix} x + \begin{pmatrix} 0 \\ K/T \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x \end{cases} \tag{1.14}$$

We are interested in estimating the parameters K and T from discrete-time measurements of the velocity and the position. This is done is the MATLAB demo number 1-6. Run the demo, and see how this quite complex problem can be solved by using MATLAB and theory covered in the SI course. To run the demo, just type `iddemo` at the MATLAB prompt and choose number 1-6, *'Building structured and user-defined models'*. Focus on the first part of the demo (1 Free Parameters). The second part (2 Coupled parameters) is, however, also of interest.