TNG022 Lab 3 preparation

Daniel Fridström (danfr755), Hans Fredö (hanfr829)

Date: 2024-12-12

Version (1)

1. We used ARX, ARMAX and BJ.

ARX is a simple model with fast estimation, ARMAX is a more complex version of ARX that takes a longer time to estimate. BJ is the most complex and it's the most flexible and works in most cases, but it takes a longer time to estimate.

2. (a) The ARX model is the fastest one to estimate.

   (b) The BJ model is the most general

   (c) The ARX is best suited for jobs where you need simplicity and speed where there is not too much noise. ARMAX is best when the noise is bigger but not too complex. The BJ when you need to most detailed modeling and you have a lot of noise in the system.

3. By sampling faster, we can make sure to be within the Nyqvist frequency so that the system works correctly.

4. $\tau = 0.1$, $T_s = 20\omega_B = \frac{20}{\tau} = \frac{20}{0.1} = 200$, (2*pi)/Ts . Två viktiga former! Ger : 0.03 för lab3 och 0.11=0.1 = Ts för lab3.

5. We want to have as many frequencies as possible, so we use white noise.

6.

a) **<u>When is u(t) white noise with variance 1?: when</u> $b_1 = 1$**

White noise with variance 1 gives

$$(R_u(k) = 1, when\ k = 0\ and$$
$$R_u(k) = 0, when\ k \neq 0),$$

$$y(t) = u(t-1) + 3u(t-2) + e(t) = b_1 u(t-) + e(t)$$

$$\text{This gives: } b_1 = \frac{R_u(0) + 3R_u(1)}{R_u(0)} = \frac{1 + (3 * 0)}{1} = 1$$

b) u(t) has the covariance function $R_u(0) = 2$,

$R_u(1) = 1$, $R_u(2) = 0.5$,..., **when** $\boldsymbol{b_1 = \frac{5}{2} = 2.5.}$

$$b_1 = \frac{R_u(0) + 3R_u(1)}{R_u(0)} = \frac{2 + (3 * 1)}{2} = 2.5$$

7.

```
function theta = mkskatt(y, u)
    % getting the length of the signals
    N = length(y);

   % initializing the matrix and output vector
    x = zeros(N, 2);
    y_vec = zeros(N, 1);

    % Constructing the x matrix and y vector
    for t = 2:N
        x(t, 1) = -y(t-1);  % -y(t-T)
        x(t, 2) = u(t-1);   % u(t-T)
        y_vec(t) = y(t);      % y(t)
    end

    % solving the theta using the normal equation
    theta = (x' * x) \ (x' * y_vec);
end
```

OBS!!! Not totally correct code, but it works for this task.