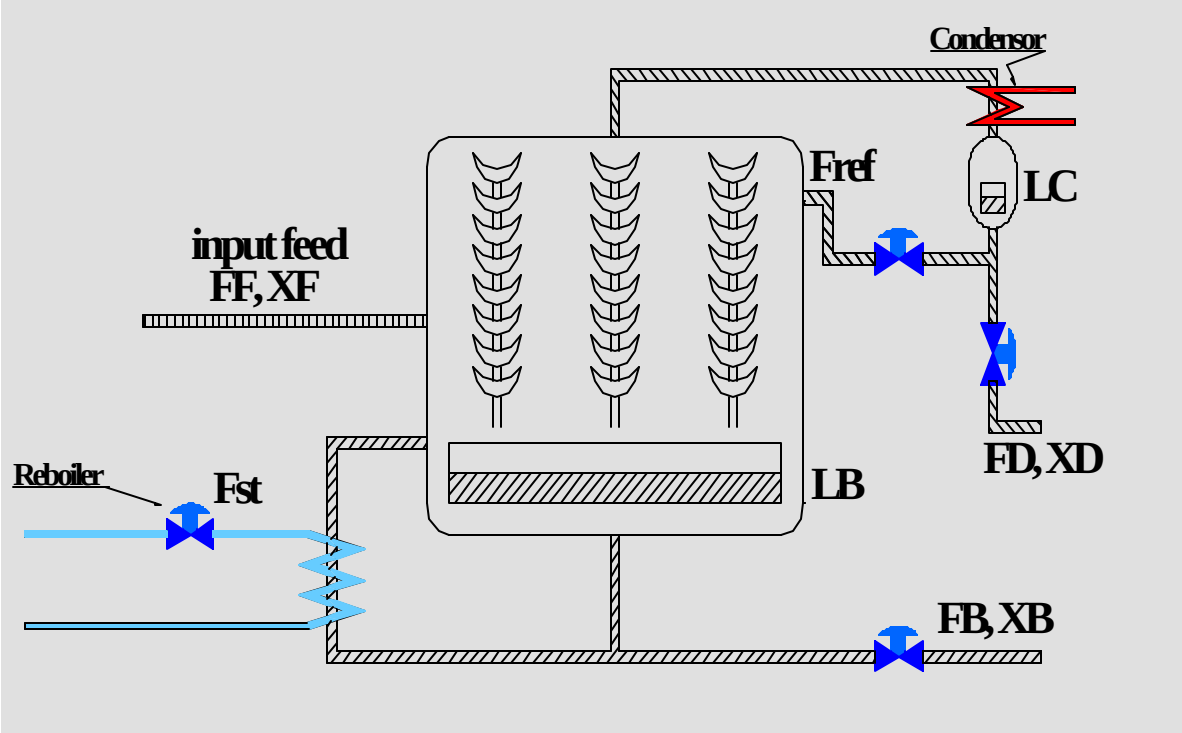
# Model Predictive Control




Begeleider/docent: Ron J.J. van den Boom

| | | | |
|---|---|---|---|
| Studenten: | M. Patrascu | en | J. Hidajat |
| Studenrs.: | 9643118 | en | 9338019 |

Technische Universiteit Delft,
Faculty of Information, Technology and Systems
Delft, november 2000

## Exercise 1 – Models & Prediction

### 1.1. Impulse response

To compute the impulse response for the true process and the three models, first we transform each model and the process in a SYSTEM format.
For the transformation of the state space, we use `ss2sys.m`, for the impulse response model, `imp2sys.m` and for the polynomial model, `tf2sys.m`.
Then we can calculate the response by using `impulse.m`.
We calculated the impulse responses for `k` = 0,...,20 with `nimp` $\in$ {10,15,20,25}, `npol` $\in$ {1,2,3,4} and `nss` $\in$ {1,2,3,4} respectively.

### 1.2. Parameters

From the plots that we get from exercise 1.1, we can determine the number of parameters that are sufficient for each model. We can conclude this, if the plot from a model is (almost) the same as the plot from the true process.
For the impulse response model, we can't choose `nimp` < 20. Because `k` goes from 0 to 20, if `nimp` < 20 then for `nimp` < `k` < 20 the response is equal to zero. In this case, 20 is sufficient (`nimp = 20`).
Then we have the other two models. For the polynomial model, 3 is sufficient (`npol = 3`) and for the state space model, 2 is sufficient (`nss = 2`). For these last two models, if we take bigger order we don't get significantly more qualitative values.
From this exercise on, we will use the sufficient parameters. In figure 1.1 we make a comparison between the "worst" case (`nimp = 10`, `npol = 1` and `nss = 1`) and the "sufficient" case (`nimp = 20`, `npol = 3` and `nss = 2`).

### 1.3. IO models

The three models from exercise 1.1 are also IO models. We give for all three the vector `dim` and explain it. We can get that vector by transforming all the models in SYSTEM format. Because we've already done that, we can ask the vector `dim` directly.

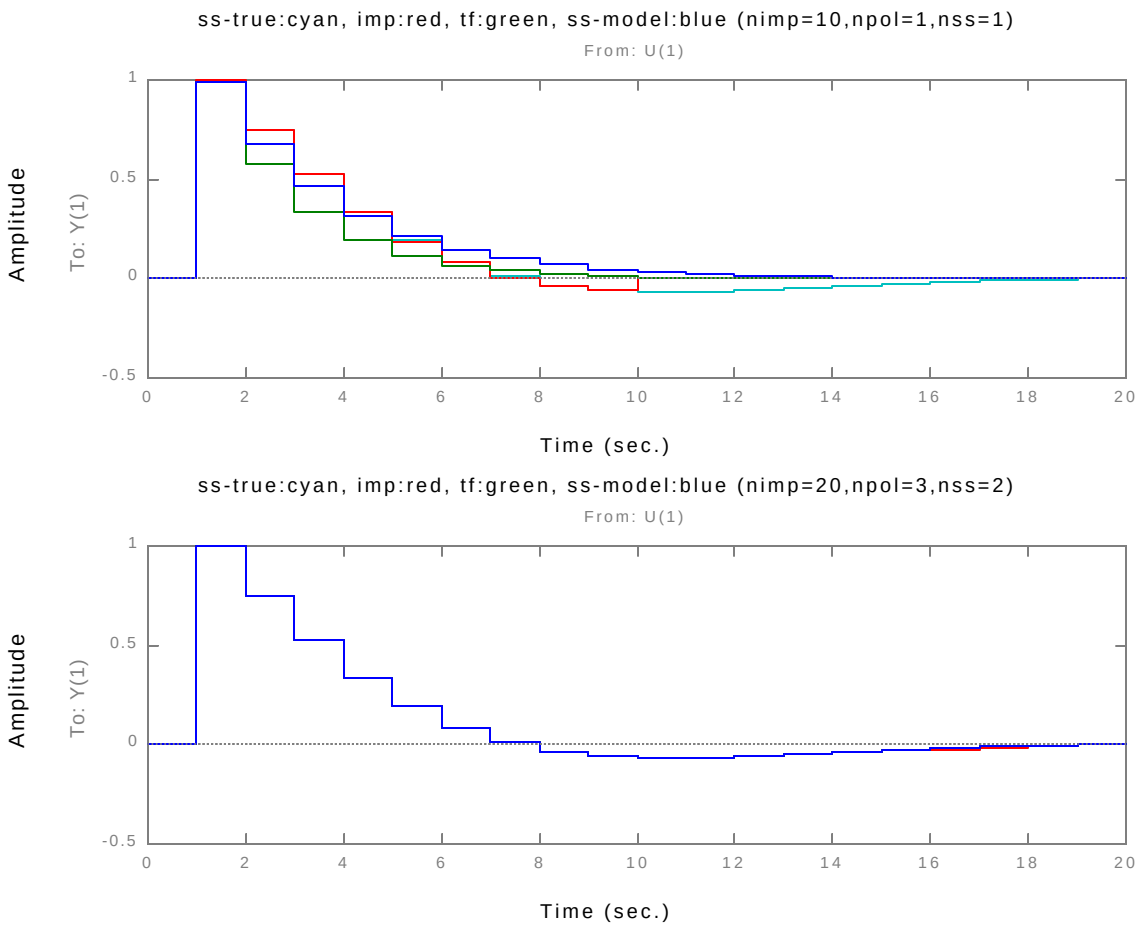



Figure 1.1. A comparison between the “worst”- and the “sufficient” case.

We were using the sufficient parameters that we found in exercise 1.2, `nimp` = 20, `npol` = 3 and `nss` = 2 and we have the three vectors as follows:

The impulse response model:

`dim_imp` = 20 1 1 1 1 0 0 0

The polynomial model:

`dim_pol` = 3 1 1 1 1 0 0 0

The state space model:

`dim_ss` = 2 1 1 1 1 0 0 0

Generally:

`dim` = [ nA $nB_1$ $nB_2$ $nB_3$ $nC_1$ $nC_2$ $nC_3$ $nC_4$]

We can derive for example for the state space model:

- nA = Ao is a 2x2 matrix and it means that this system has 2 states,
- $nB_1$ = Ko is a 2x1 matrix (= a column vector with 2 elements),
- $nB_2$ = Lo is a 2x1 matrix,
- $nB_3$ = Bo is a 2x1 matrix,
- $nC_1$ = Co is a 1x2 matrix (= a row vector with 2 elements),
- $nC_2 = nC_3 = nC_4 = 0$ because the last 3 values in the dim-vectors are 0.

Similarly, we can derive for the other two models the same thing.

## 1.4. IIO models

After we have observed the IO models, we transform them into IIO models. We use `io2iio.m` for this transformation. Then by giving the vector `dim`, we can see that the dimensions of `nA` of each model increases with one and this means that the size of the A matrix increases with one. The cause is an increase in the number of states. This extra state is the extra integrator that is required to calculate the actual output from the input increment.

The impulse response model gives:

```
dim_imp_iio =     21     1     1     1     1     0     0     0
```

The polynomial model:

```
dim_pol_iio =      4     1     1     1     1     0     0     0
```

The state space model:

```
dim_ss_iio  =      3     1     1     1     1     0     0     0
```

## 1.5. Prediction models

After we transformed it into IIO models, we make the prediction model for the output signals. Therefore we use 15 steps.

We calculate the three predictions with `pred.m` and find the sizes of the system matrices from these models. We got the three vectors as follows:

The impulse response model gives:

```
dimp_imp =  21     1     16    16    16    0     0     0
```

The polynomial model:

```
dimp_pol =   4     1     16    16    16    0     0     0
```

The state space model:

```
dimp_ss  =   3     1     16    16    16    0     0     0
```

Just like in exercise 1.3, We can derive for example for state space model:

nA = Ao is a 3x3 matrix and it means that this system has 3 states,

$nB_1$ = Ko is a 3x1 matrix (= a column vector with 3 elements),

$nB_2$ = Lo is a 3x16 matrix,

$nB_3$ = Bo is a 3x16 matrix,

$nC_1$ = Co is a 16x3 matrix.

With `size.m`, we can explain the sizes of the system matrices of the prediction model:

```
State-space model with 33 input(s), 16 output(s), and 2 state(s).
```

We have here 33 inputs, because it's the sum of the columns of the B matrices ($B_1$, $B_2$ and $B_3$). Each matrix (except the A matrix) has 15 elements more. These extra elements are the prediction values from each k with the prediction horizon N = 15. So we're not getting only the value from k, but also 15 values later. That's why the matrices increase from 1 to 16.
Similarly, we can derive for the other two models the same thing.

## 1.6. Simulation of prediction model

In this exercise we try to make a simulation of the prediction model for a special case from the previous models. Here we want to make a prediction of the output at time $k = 0$. There's also given that the disturbance signal $d_o(k) = 0$ (= $w_o(k)$), noise signal $e_o(k) = 0$ and initial state $x_o(0) = 0$.
First we have the equation of the prediction of the output:

$$\tilde{y}(k) = \tilde{C}_1 x_o(k) + \tilde{D}_{11} e_o(k) + \tilde{D}_{12} \tilde{w}_o(k)$$

and from the given signals (disturbance, noise and initial state), the equation simplifies and depends only to the state.

$$\tilde{y}(k) = \tilde{C}_1 x_o(k)$$

From this equation we can conclude that the prediction output at time k = 0, is equal to zero ($\tilde{y}(0) = 0$, because of the initial state). We go on with the state equation:

$$x_o(k+1) = A x_o(k) + \tilde{B}_1 e_o(k) + \tilde{B}_2 \tilde{w}_o(k) + \tilde{B}_3 \tilde{v}(k)$$

Also from the given signals, we can simplify this state equation:

$$x_o(k+1) = A x_o(k) + \tilde{B}_3 \tilde{v}(k) \qquad (1)$$

Then we substitute the state equation in the output equation and we get:

$$\tilde{y}(k) = \tilde{C}_1 (A x_o(k-1) + \tilde{B}_3 \tilde{v}(k-1))$$

in other formulation:

$$\tilde{y}(k+1) = \tilde{C}_1 (Ax_o(k) + \tilde{B}_3 \tilde{v}(k)) \qquad (2)$$

We have some extra information about the input: $u(k)$ is a step at time k = 0. It means that the $Du(k) = v(k) = 1$ for $k = 0$. And also the predicted values of $v(k)$ (= $\tilde{v}(k)$) is equal to 1 under the same condition.

Now we can plot the predicted response at time $k = 0$.

From exercise 1.5, we've already made a prediction from each model. Then we transform those three predicted models in state space format. By choosing the right matrices we can make the calculation with the equations (1) and (2).

We get from each model an output, which is a (N+1)x(N+1) matrix. Because of the receding horizon principle we get (at each k) N extra outputs, but only the first element is of interest. That is the reason why we only take the first element of each output.

We must differentiate these outputs to get the outputs that we desire (in IO form).

In figure 1.2 we can see the outputs of the three predicted models.

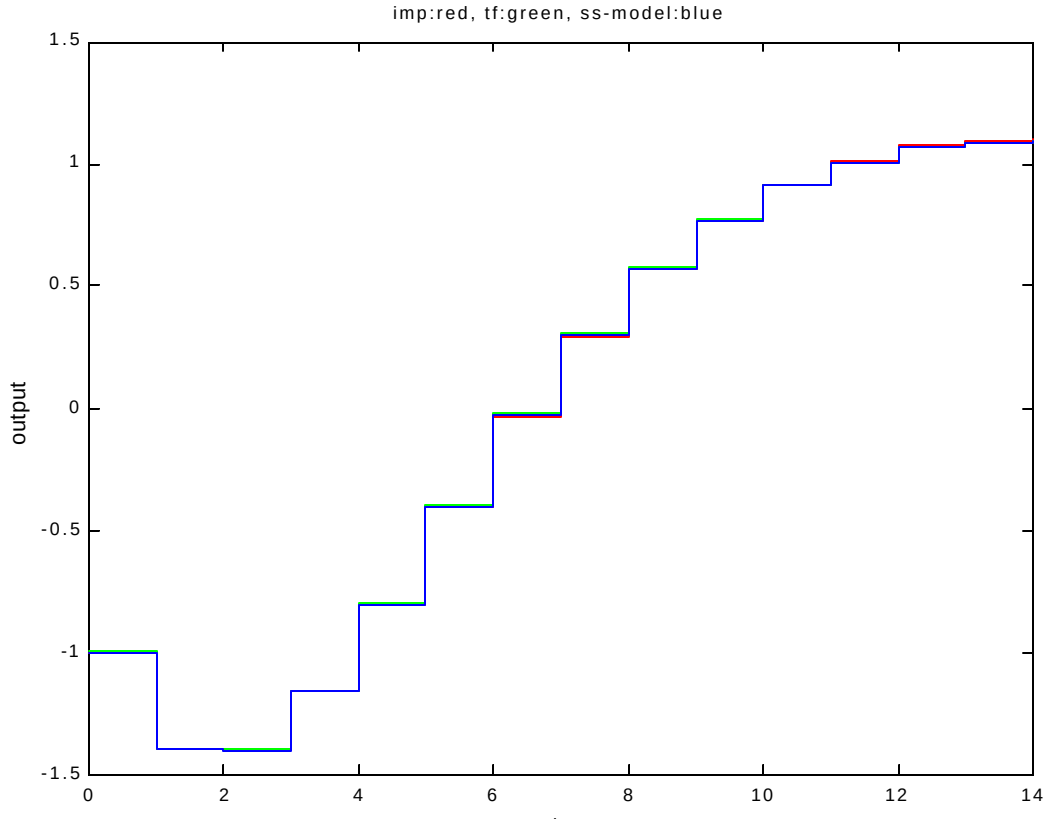



Figure 1.2. The outputs from the three predicted models in special case

The m-file of this exercise (`exer1.m`) is given in *Appendix A*.

## Exercise 2 – Standard Formulation

### 2.1. Proof of the weighting function

We have a weighting function $W_u(q)$. This function is defined as:

$$W_u(q) = \frac{1}{1-\delta q^{-1}} I \qquad 0 < \delta < 1,$$

and has the following system matrices in state space notation:

$$A_u = \delta,\ B_u = \delta,\ C_u = 1,\ D_u = 1.$$

These two notations are equivalent, because:

$$W_u(q) = C_u((qI - A_u)^{-1})B_u + D_u$$

If we substitute the system matrices in this equation, we get the weighting function $W_u(q)$ in transfer function notation.

$$W_u(q) = 1 \cdot ((q-\delta)^{-1}) \cdot \delta + 1 \qquad \Leftrightarrow$$

$$W_u(q) = \frac{\delta}{q-\delta} + 1 \qquad \Leftrightarrow$$

$$W_u(q) = \frac{\delta}{q-\delta} + \frac{q-\delta}{q-\delta} \qquad \Leftrightarrow$$

$$W_u(q) = \frac{\delta + q - \delta}{q-\delta} \qquad \Leftrightarrow$$

$$W_u(q) = \frac{q}{q-\delta}$$

Then we divide the numerator and the denominator of this weighting function by $q$.

$$W_u(q) = \frac{1}{1-\delta q^{-1}}$$ End Proof

### 2.2. Standard predictive control problem

We have a performance index $J(k)$, that is given by:

$$J(u,k) = \sum_{j=N_1}^{N_2} \left\| \hat{e}(k+j \mid k) \right\|^2 + \sum_{j=1}^{N_2} \hat{x}_o^T(k+j \mid k) Q \hat{x}_o(k+j \mid k) + \sum_{j=1}^{N_2} \left\| W_u(q) u(k+j-1) \right\|^2$$

The problem is finding the control law that minimizes this performance index under the constraints as a standard predictive control problem:

There are various types of performance indices in predictive control. The first type is LQPC performance index. In this case, the input signal is measured together with the state. The second one is GPC performance index. This type measures the input increment signal together with the tracking error.

The problem is that the performance index that has been given contains all three elements: tracking error (= $e$), the state (= $x_o$) and the input (= $u$). So it's difficult to find the control law that could minimize this performance index.

Though, we still can compute the corresponding system matrices from this performance index and the selection matrices.

The corresponding system matrices are:

$$z(k) = \begin{bmatrix} y(k+1) - r(k+1) \\ Q^{1/2} x_o(k+1) \\ W_u u(k) \end{bmatrix},$$

$$x(k) = x_o(k), \quad v(k) = u(k), \quad w(k) = \begin{bmatrix} r(k) \\ d_o(k) \end{bmatrix}, \quad e(k) = e_o(k),$$

$$A = A_o, \quad B_1 = K_o, \quad B_2 = \begin{bmatrix} 0 & L_o \end{bmatrix}, \quad B_3 = B_o,$$

$$C_1 = C_o, \quad D_{11} = 1, \quad D_{12} = 0,$$

$$C_2 = \begin{bmatrix} C_o A_o \\ Q^{1/2} A_o \\ 0 \end{bmatrix}, \; D_{21} = \begin{bmatrix} C_o K_o \\ Q^{1/2} K_o \\ 0 \end{bmatrix}, \; D_{22} = \begin{bmatrix} -I & C_o L_o \\ 0 & Q^{1/2} L_o \\ 0 & 0 \end{bmatrix}, \; D_{23} = \begin{bmatrix} C_o B_o \\ Q^{1/2} B_o \\ W_u \end{bmatrix}$$

The selection matrices are:

$$N = N_2 - 1$$

$$\Gamma(j) = \begin{bmatrix} \Gamma_1(j) & 0 \\ 0 & I \end{bmatrix}$$

$$\Gamma_1(j) = \begin{cases} 0 & for \quad 0 \le j \le N_1 - 1 \\ I & for \quad N_1 \le j \le N \end{cases}$$

then:

$$x(k+1) = Ax(k) + B_1 e(k) + B_2 w(k) + B_3 v(k)$$
$$y(k) = C_1 x(k) + D_{11} e(k) + D_{12} w(k)$$
$$z(k) = C_2 x(k) + D_{21} e(k) + D_{22} w(k) + D_{23} v(k)$$

## Exercise 3 – Solution of the SPCP

### 3.1. The changing of the performance index of the SPCP

In this exercise, we want to see the changing of the performance index when we change some parameters.

First if the prediction horizon (= $N_2$) is increased, the performance index will also increase. Assuming there is a control horizon, there is only the chance to choose the parameters until time $N_c$; the values for the times after the control horizon will be constant. Thus, if we will increase $N_2$, the performance index can only become bigger, as there will be more times which will build up the error between $N_c$ and $N_2$ (and so the performance index will increase).
Decreasing $N_2$ leads to the opposite, e.g. decreasing performance index.

Secondly, when we increase the control horizon (= $N_c$), the performance index will decrease.
This is because there will be more degrees of freedom to choose our parameters and we can choose our parameters for more time steps, so we can decrease the error. There will be less parameter with a fixed value (between $N_c$ and $N_2$) and so the performance index will be smaller. This means that the performance index will decrease with increasing $N_c$.
Decreasing NC will increase the performance index.

Then if an inequality constraint is introduced, the performance index will increase. Because by adding an inequality constraint, it will be more difficult for the system to maintain the desired values, as it cannot for example jump above a certain speed (the constraint is here speed) for redressing (minimizing) a certain distance error. It must remain under the constraint, and so it will take longer to get to the desired value (or the position in this case). By this way, the performing index will also increase.
Removing an equality constraint will decrease the performance index.

### 3.2. Find a ranking in magnitude

We have here six performance indices that we like to compare:

- $J_1$: unconstrained SPCP with finite prediction horizon $N_2$,
- $J_2$: SPCP with finite prediction horizon $N_2$ and finite control horizon ($N_c < N_2$),
- $J_3$: SPCP with infinite prediction horizon and infinite control horizon,

- $J_4$: SPCP with infinite prediction horizon and finite control horizon,
- $J_5$: SPCP with finite prediction horizon, subject to inequality constraints $\boldsymbol{y}(k) \leq \boldsymbol{Y}(k)$,
- $J_6$: SPCP with infinite prediction horizon and finite control horizon, subject to inequality constraints.

We have also defined:

$$J_i^* = \min_{\tilde{v}} J_i(k)$$

The six possibilities $J_1 \ldots J_6$ are schematically sketched in figure 3.1.

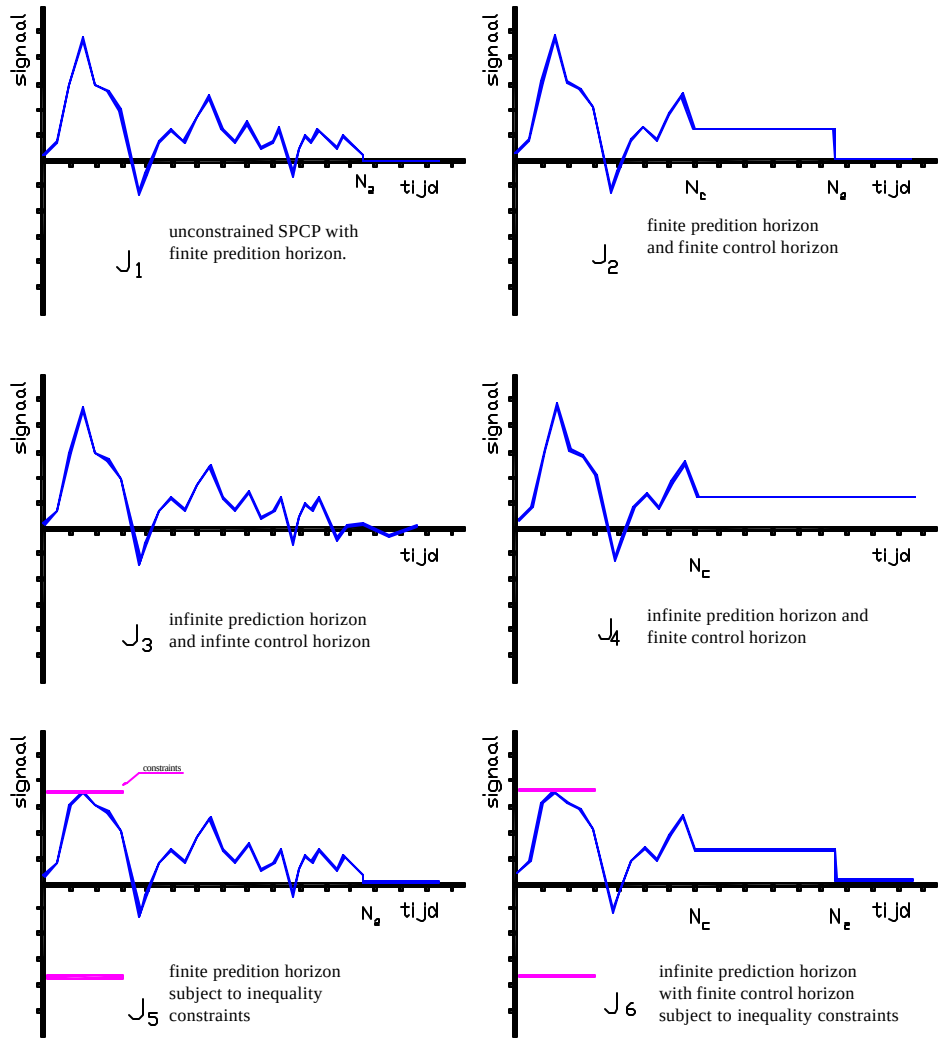



Figure 3.1. The six performance indices, $J_1 \ldots J_6$

There can be made the next general inequality systems:

(1) $J_3^* \leq J_1^* \leq J_2^*$

(2) $J_3^* \leq J_4^*$

(3) $J_3^* \leq J_1^* \leq J_5^*$

(4) $J_3^* \leq J_2^* \leq J_6^*$

(5) $J_5^* \leq J_6^*$

(1) Performance index $J_3^*$ is considered to be the best one, because here both the prediction horizon and the control horizon are infinite and so one has the chance to influence the system in infinite many ways. After $J_3^*$ follows $J_1^*$, having only finite prediction horizon. Until step $N_2$, one can choose any value for each time step, so that there are many degrees of freedom, though not more than at $J_3^*$, where there are infinitely many time steps. This is why $J_3^* \leq J_1^*$. Further, $J_1^*$ is better than $J_2^*$ regarding the fact that $J_2^*$ has a finite control horizon, which includes that a constant value must be chosen between $N_c$ and $N_2$. This will decrease the degrees of freedom from which one can choose for minimizing the performance index. That's why $J_1^* \leq J_2^*$.

(2) Because at $J_3^*$ one can choose parameters for a longer time than at $J_4^*$ (only $N_2$ steps), there are more degrees of freedom in $J_3^*$ and so $J_3^* \leq J_4^*$.

(3) $J_5^*$ looks like performance index $J_1^*$, the only difference is that $J_5^*$ has an inequality constraint. If the system might need to severely increase the dynamics to minimize the errors, it will not be able to pass the constraints and so the errors will only become bigger compared to the unconstrained case $J_1^*$. In conclusion, $J_i^* \leq J_5^*$. Combining this result with (1) gives $J_3^* \leq J_5^*$. Together, we have $J_3^* \leq J_1^* \leq J_5^*$.

(4) At $J_6^*$ we have the same problem like at $J_5^*$ regarding to the constraints which diminish the degrees of freedom and thus increase the performance index. The system can only react between the constraint margins; if it should be necessary to go above the constraint, it will not be possible and so the error will increase compared to the unconstrained case $J_2^*$. Combining this with (1) gives: $J_3^* \leq J_2^* \leq J_6^*$.

(5) Finally, we can conclude that $J_5^* \leq J_6^*$ from the same reason as was given for $J_1^* \leq J_2^*$, see (1).

## 3.3. Controller A versus B

We can't say that controller A is better than controller B if $J_A < J_B$. If the performance indices from both controllers are calculated under the same conditions (for example: $N_{c,A} = N_{c,B}$ etc) then we can say that controller A is better than controller B. But in this case we can't conclude that, because there is lack of information (about $N_2$, constraints etc).

## Exercise 4 – GPC & LQPC

We have the following system:

$$a_o(q)\,y(k) = b_o(q)u(k) + f_o(q)d_o(k) + c_o(q)e_o(k)$$

where:

$$a_o(q) = (1-1.5q^{-1})(1-0.7q^{-1})(1-0.2q^{-1})$$
$$b_o(q) = 1.5q^{-1}(1-1.2q-1)(1-0.5q^{-1})$$
$$c_o(q) = (1-0.8q^{-1})(1-0.4q^{-1}+0.13q^{-2})$$
$$f_o(q) = 0.1q^{-1}(1-0.7q^{-1})(1-0.2q^{-1})$$

### 4.1. GPC controller design

The system that is given is a polynomial IO model. If we want to find a GPC controller, first we must transform the model into IIO model. With MATLAB, we calculated the transfer functions.
Actually we transform only $a_o$ into $a_i$, because for the IIO model we have: $b_i(q) = b_o(q)$, $c_i(q) = c_o(q)$ and $f_i(q) = f_o(q)$. For $a_i$ we use `conv.m`. We convolute $a_o$ with $Dq = (1 - q^{-1})$ (= `[1 -1]`).
Then we can find the GPC controller by using `gpc.m`; also, we must define some parameters first:

$N_1 = 1$, $N_2 = 25$, $N_c = 2$, $P(q) = 1$ and $l = 1$.

For the simulation (`lensim = 50`), choose $r(k)$ as a step at time $k = 10$, $d_i(k)$ as a pulse at time $k = 20$ and $e_i(k)$ as ZMWN with standard deviation $5.10^{-3}$.

From this controller, we must first study the effect of changing `dumax` $\in$ `{0.25,0.5,1}` (with `apr = 4`). This changing effects the duration of the system to go to its reference signal (= $r(k)$) and also the compensation time to a disturbance.
If we double the value of `dumax` to 0.5, the system will go to its reference signal about three times faster (also if a disturbance is introduced).

Then we study the effect of changing the apriori knowledge `apr` $\in$ `{1,2,3,4}`, without constraint on $Du$ (so `dumax = 0`)
First we make a summary over the apriori:

`apr = 1`: $r(k+j)$ is apriori known for $j > 0$,
$d(k+j)$ is apriori known for $j > 0$.

`apr = 2`: $r(k+j)$ is not apriori known for $j > 0$,
$d(k+j)$ is apriori known for $j > 0$.

`apr = 3`: $r(k+j)$ is apriori known for $j > 0$,
$d(k+j)$ is not apriori known for $j > 0$.

`apr = 4`: $r(k+j)$ is not apriori known for $j > 0$,
$d(k+j)$ is not apriori known for $j > 0$.

From the plot, we can conclude that with an apriori knowledge of the reference signal the system can go to its reference much and much faster. Also the system can anticipate a disturbance when it has an apriori knowledge of the disturbance.

In figure 4.1 we can see for `apr = 1` and `apr = 3` that the output of the system react before a reference has been given. For disturbance, in case `apr = 1` and `apr = 2`, we can see that the disturbance has no effect to the system.

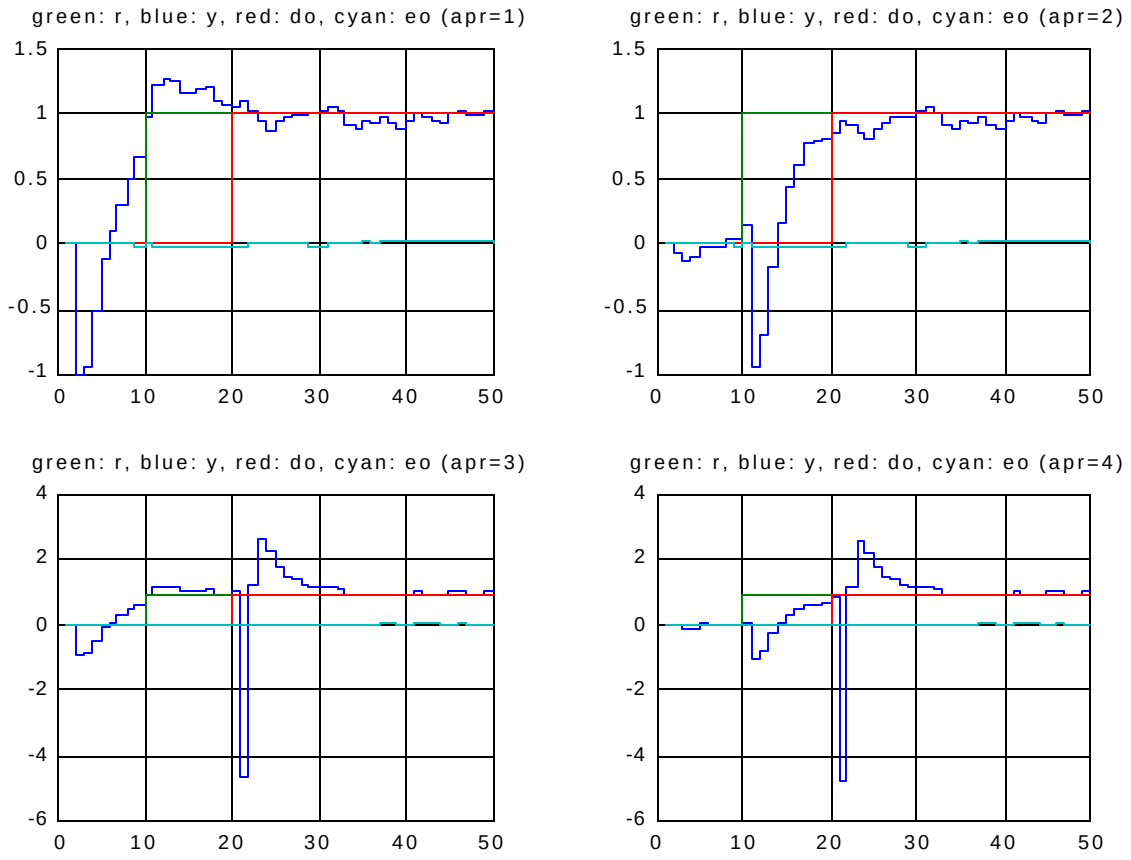



Figure 4.1. The output signal of GPC controller for `apr` $\in \{1,2,3,4\}$

## 4.2. LQPC controller design

Now we make a LQPC controller for the system. The first step is to transform the polynomial IO model that we have into a state space IO model. But there isn't a direct function for a transformation from polynomial- into state space model. So we use `tf2sys.m` first to transform the polynomial into SYSYEM format and then we use `syst2ss.m` to get the state space model of the system.

Then we can find the LQPC controller by using `lqpc.m`, also we must first define some parameters:

$N_1 = 1, N_2 = 10, N_c = 10$ and $Q = C_o{}^T C_o$.

For the simulation (`lensim = 25`), choose $d_o(k) = 0$ and $e_o(k)$ as ZMWN with standard deviation 0.25. There are no inequality constraints.

With `lqpc.m`, `apr = 1` or `apr = 2` mean the reference signal d(k+j) is apriori known for $j > 0$.

For `apr = 3` or `apr = 4` mean the reference signal d(k+j) is not apriori known for $j > 0$.

For this controller, we must first study the effect of changing $R \in \{0.1,1,10\}$. And for the initial state we use $x_o{}^T$ = `[1 1 1]`.

The plot is given in figure 4.2. We can see that with $R = 0.1$, the system becomes unstable. But with other values, $R = 1$ and $10$, the system is stable ($y \rightarrow 0$ for $k \rightarrow \infty$).

We can compare these two values from the plot. The system with $R = 10$ has a bad response in the beginning, but it goes faster (to zero) later on, than $R = 1$.

We then study the effect of different initial states for $R = 1$. Here, we define four initial states. For the first state we use the state from the previous study ($x_{o1}{}^T$ = `[1 1 1]`). Then we define the other three: $x_{o2}{}^T$ = `[0 0 0]`, $x_{o3}{}^T$ = `[-1 1 -1]` and $x_{o4}{}^T$ = `[-1 -1 -1]`.

From the plot we can directly see that for the first ($x_{o1}$) and fourth ($x_{o4}$) case the systems are stable, but not for the other two systems. We observed, actually, that for the second initial state ($x_{o2}$) the system is stable, but it is sensitive to the noise. And for the third state ($x_{o3}$) we can conclude that the system is unstable.

Note: For the study of different initial states, we have plot the systems in figure 4.3.

We use the same ZMWN sequence in all simulations to compare the results.

The m-file of this exercise (`exer4.m`) is given in *Appendix B*.

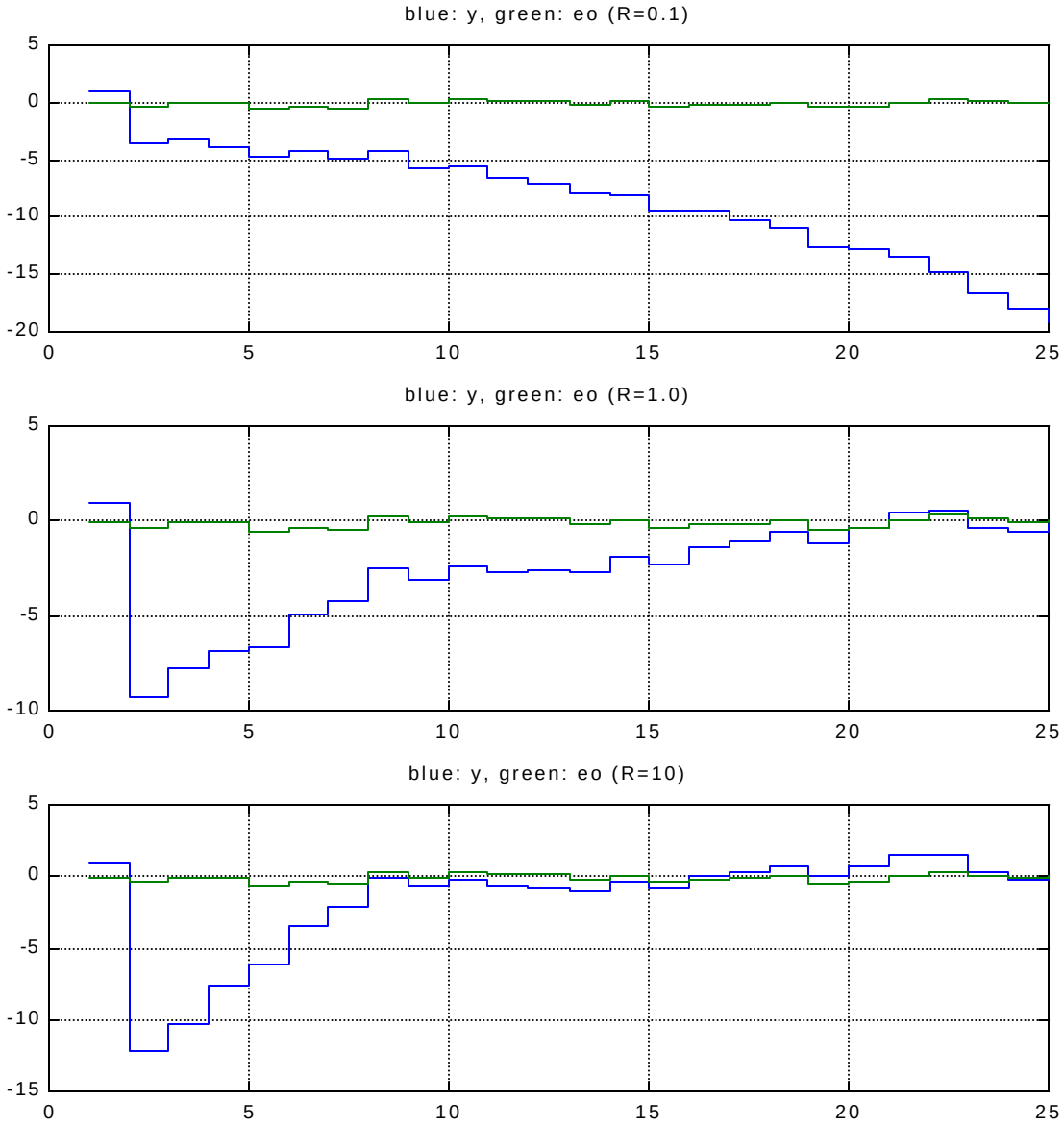



Figure 4.2. The study of LQPC controller by changing $R \in \{0.1,1,10\}$

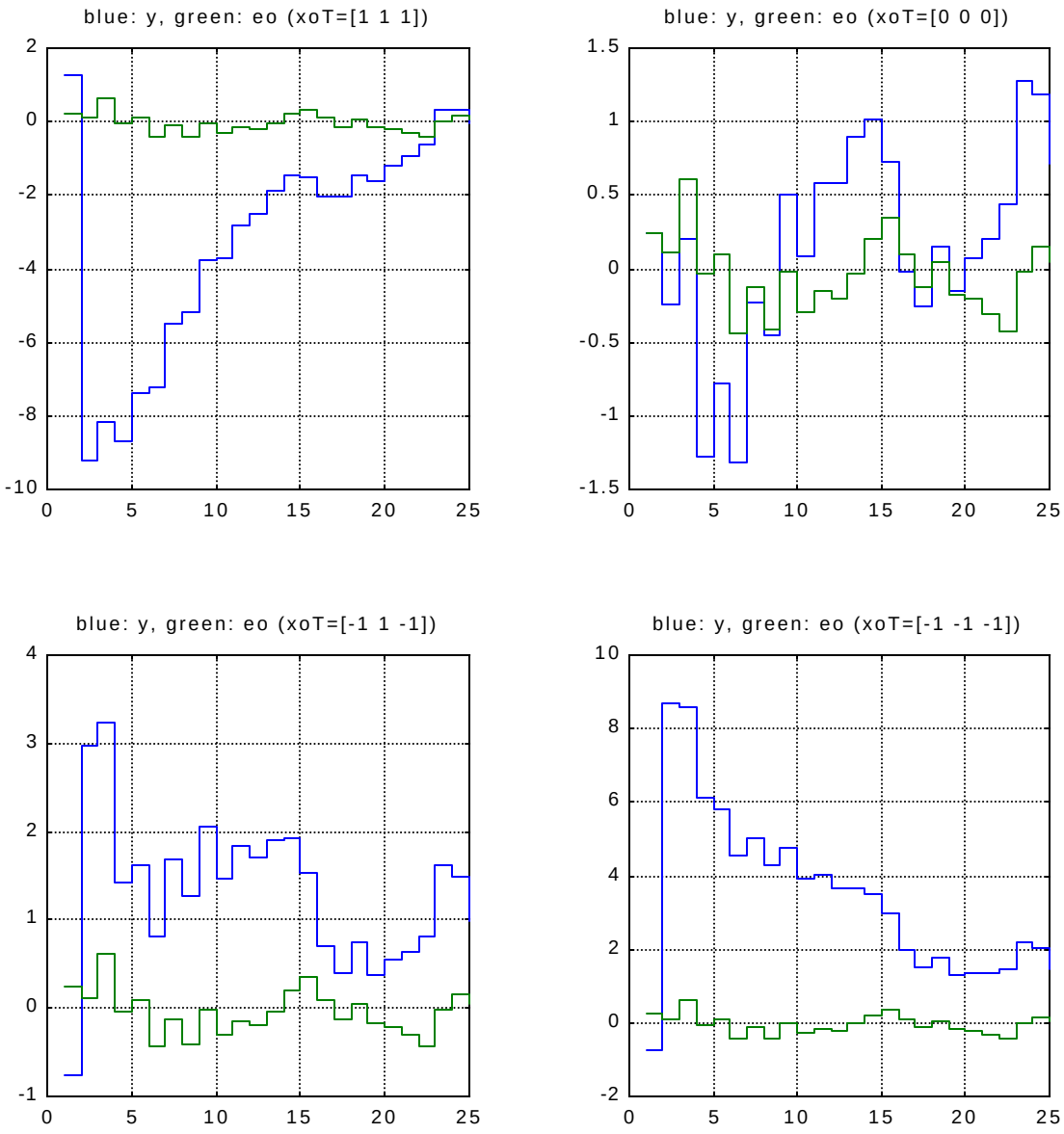



Figure 4.3. The study of LQPC controller with different initial states

## Exercise 5 – Tuning GPC & LQPC

### 5.1. Tuning of a GPC controller

In this exercise we consider the problem of exercise 4.1 for `dumax = 0` (so $\Delta u_{max} = \infty$). Then with the tuning rules we find the initial values of the tuning parameters.
For the following initial setting are recommended:

| | | |
|---|---|---|
| $N_1 = 1 + d$ | ($d$ = the dead time of the process) |
| $N_c = n + 1$ | ($n$ = the dimension of the matrix $A$) |
| $N_2 \geq \text{int}(t_S/T_S)$ | ($t_S$ = the settling time of the process; $T_S$ = the sampling time) for well-damped systems |
| $N_2 \geq \text{int}(2\omega_s/\omega_b)$ | ($\omega_s$ = the sampling frequency; $\omega_b$ = the bandwidth of the process) for badly-damped and unstable systems |
| $\lambda = 0$ | ($\lambda$ = the weighting parameters, for minimum phase process) |
| $\lambda > 0$ | (but as small as possible, for non-minimum phase process) |

We can get some of these parameters from the system itself, but there are also some parameters that require the original continuous time model. To get this model, we can use `d2cm.m` to the (discrete) system, with a sampling time. For the sampling time we use $T = 0.01$. Parameters like dead time and the dimension of matrix $A$ can be found in exercise 4.1.

This system has a dead time that is equal to 1, so we have 2 for $N_1$. From the $A$ matrix we get 5 for $N_c$. Then for $N_2$ we need the continuos time model. First we need to know the stability of the system. We can find it out by looking at the eigenvalues of $A$ matrix (from the continuos model). We have a stable system if all the eigenvalues of $A$ are in the left half plane. But this is not the case, that's why we conclude that the system is unstable. We need to find the bandwidth of the process $\omega_b$ to get $N_2$.
This bandwidth of the system with transfer function $G_{c,o}(s)$ is the frequency at which the magnitude ratio drops to 1/√2 of its zero-frequency level. So:

$$\omega_b = \max_{\omega \, in \, \Re} \left\{ \omega \,\middle|\, |G_{c,o}(j\omega)|^2 / |G_{c,o}(0)|^2 \geq 0.5 \right\}$$

After we have found this bandwidth, we can calculate $N_2$. It is only a lower bound and often by increasing $N_2$, it may improve the stability and the performance of the closed loop.

The last parameter to tune is the signal weighting parameter $\lambda$. To tune this parameter, we must first know if the system is minimum phase or non-minimum phase. From the exercise 4.1 we can see that it is a non-minimum phase system and it means that we must take $\lambda$ as small as possible ($\lambda > 0$). For this exercise we take $\lambda = 0.01$.

Then we obtain again the optimal GPC controller by using these parameters. The result of this optimization can be seen in figure 5.1.

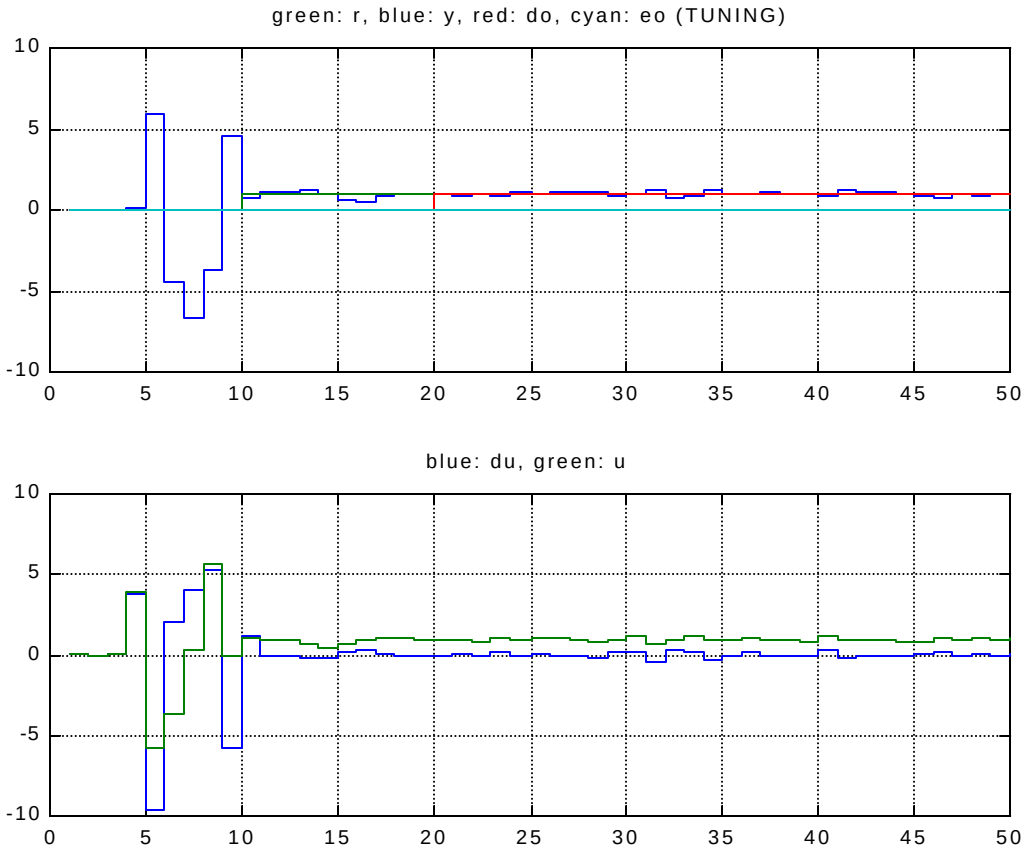



Figure 5.1. The optimal GPC controller ($N_1 = 2, N_2 = 7, N_c = 5, \lambda = 0.01$)

## 5.2. Tuning of an LQPC controller

In this exercise we do almost the same thing as in the previous exercise (for the LQPC controller). For the initial settings for the summation parameters, we can use the values that we found from 5.1. ($N_1 = 2, N_2 = 7, N_c = 5$).

For the weighting parameters from this LQPC controller, we consider that the system has single-input-single-output (SISO). Then we choose:

$$Q = C_1^T C_1 \qquad \text{and} \qquad R = I^2$$

With the same procedure from exercise 4.2 we get the result that is given in figure 5.2.

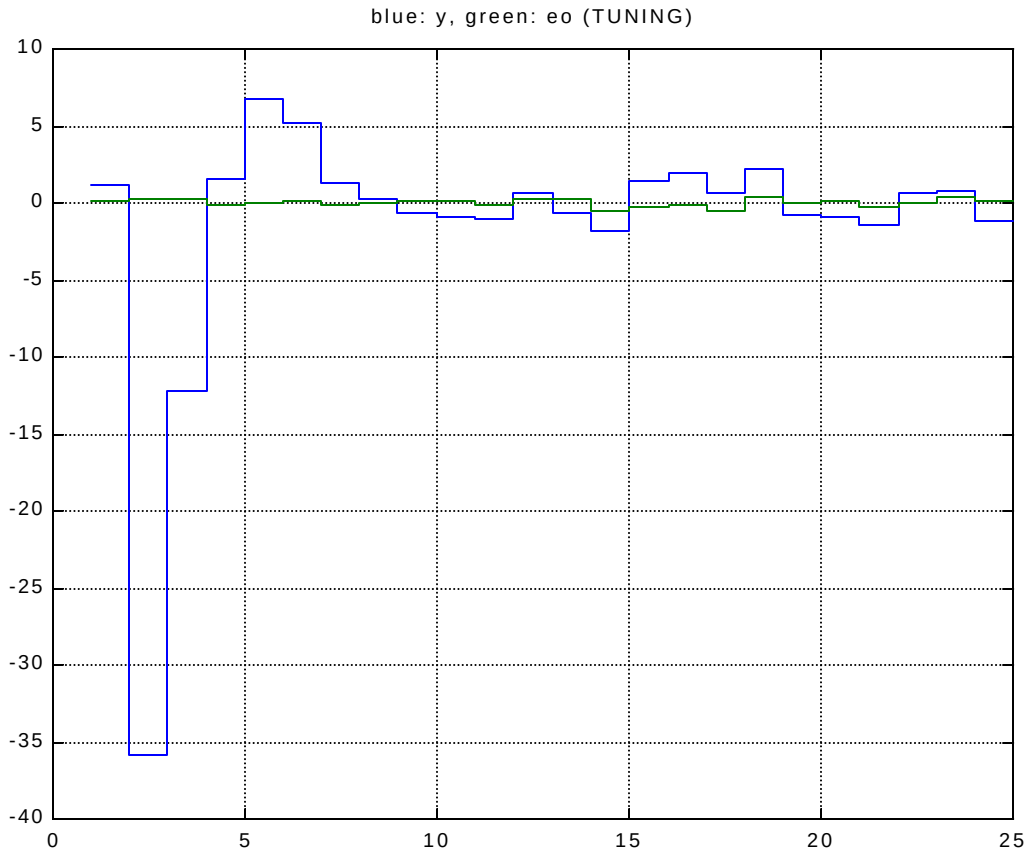



Figure 5.2. The optimal LQPC controller ($N_1 = 2, N_2 = 7, N_c = 5, Q = C_1{}^T C_1, R = I^2$)

## 5.3. Elevator example

In this exercise we reproduce the elevator example (chapter 5, example 21 from the lecture notes). But first, we will summarize the system of the elevator.

Taking a zero-order hold transformation with sampling time $T$ gives a discrete-time IO model:

$$x_o(k+1) = A_o x_o(k) + K_o e(k) + B_o u(k)$$

$$y(k) = C_o x_o(k)$$

where

$$A_o = \begin{bmatrix} 1 & 0 & 0 \\ T & 1 & 0 \\ T^2/2 & T & 1 \end{bmatrix} \quad B_o = \begin{bmatrix} T \\ T^2/2 \\ T^3/6 \end{bmatrix} \quad K_o = \begin{bmatrix} 0.1T \\ T + T^2/20 \\ T/2 + T^2/6 + T^3/60 \end{bmatrix} \quad C_o = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

For the state we define:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} \dddot{y}(t) \\ \ddot{y}(t) \\ y(t) \end{bmatrix}$$

We use here $u(k)$ rather than $Du(k)$. This is because the system already consists of a triple integrator and another integrator in the controller is not desired. The aim of this example is to direct the elevator from position $y = 0$ at time $t = 0$ to position $y = 1$ as fast as possible. For a comfortable and safe operation, control is done subject to constraints on the jerk, acceleration, speed and overshoot. We can translate these constraints into linear constraints on the control signal $u(k)$ and prediction of the state $\hat{x}(k)$:

$$\begin{aligned}
u(k+j-1) &\leq 0.4 && \text{(positive jerk)} \\
-u(k+j-1) &\leq 0.4 && \text{(negative jerk)} \\
\hat{x}_1(k+j) &\leq 0.4 && \text{(positive acceleration)} \\
-\hat{x}_1(k+j) &\leq 0.4 && \text{(negative acceleration)} \\
\hat{x}_2(k+j) &\leq 0.4 && \text{(positive speed)} \\
-\hat{x}_2(k+j) &\leq 0.4 && \text{(negative speed)} \\
\hat{x}_3(k+j) &\leq 1.01 && \text{(overshoot)}
\end{aligned}$$

for all $j = 1, \ldots, N_2$.

For the sampling-time, we choose $T = 0.1$ and prediction and control horizon $N_2 = N_c = 30$ and minimum cost-horizon $N_1 = 1$. Further we consider a constant reference signal $r(k) = 1$ for all $k$ and the weightings parameters $P(q) = 1$, $l = 0.1$. The predictive control problem is solved by minimizing the GPC performance index (for an IO model) subject to the above linear constraints.

For this exercise we consider the algorithm that has been used in the `gpc.m`.
In this algorithm, we must transform the system into a SPC problem. With `gps2spc.m`, we can realize that transformation. Because the system is in the state space format, we must first transform it in the SYSTEM format.

Also we need to define some parameters and constraints that we need for the simulation. These steps follow from the manual and the previous exercises except for the $E$ matrix, where we need to add the state constraint to the standard problem. First we must notice that the function `add_x.m` can not be applied to the prediction model. So we must add the state constraints before we make a prediction of the system. In this function the constraints is given by (for `sgn = 0`):

$$-1 \le Ex(k+j) \le 1 \qquad \text{for } j = 1, \ldots, N$$

In the predictive control, the constraints are usually given by $y(k) \le Y(k)$. With some matrix manipulation, we obtain for this elevator example:

$$E = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 1.01 \end{bmatrix}^{-1}$$

The system model works properly (no warnings) for noiseless case, but when the noise is introduced, we will get:

```
Warning: The constraints are overly stringent;
there is no feasible solution.
```

The result can be accepted, if the (amplitude of the) noise is not too big. The result of this model is given in figure 5.3. For noise we used ZMWN with standard deviation $5 10^{-3}$.

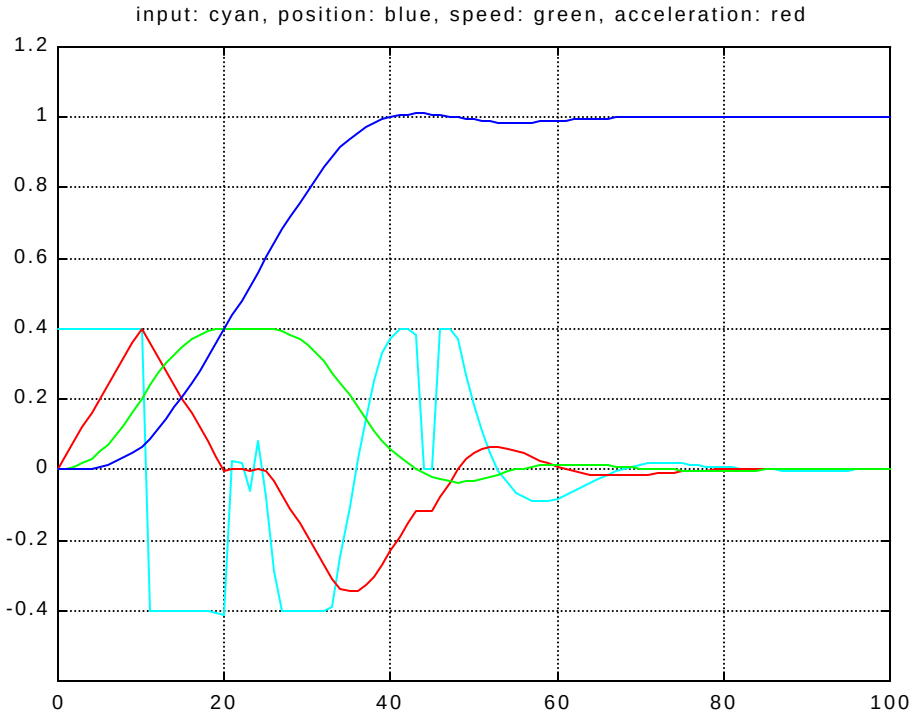



Figure 5.3. Predictive control of an elevator

## 5.4. Stability analysis

Again we consider exercise 4.2 and compute the LQPC controller by using `lticll.m`. We use different values of $N_c$ to analyze the stability by observing the closed-loop poles.

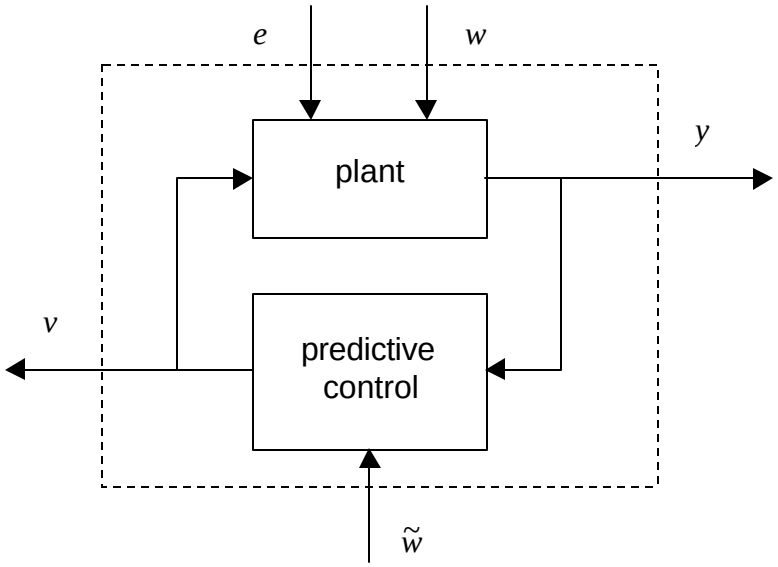



Figure 5.4. Closed loop configuration

The LQPC controller that we get this time contains the state space matrices of the closed loop in the LTI case. For the stability of the total system (plant + predictive control). We can observe that by finding the eigenvalues of the $A$ matrix. These eigenvalues are the closed-loop poles of the total system.

For $N_c$ = 2, we get for the closed-loop poles:

Pole1 = – 0.1216
Pole2 = 0.9898
Pole3 = 0.4077
Pole4 = 0.8000
Pole5 = 0.2000 + 0.3000i
Pole6 = 0.2000 – 0.3000i

Because all of these poles are strictly inside the unit circle, we conclude that the total system is stable.

For $N_c$ = 5, we get for the closed-loop poles:

Pole1 = – 0.1722
Pole2 = 0.9759
Pole3 = 0.4421
Pole4 = 0.8000
Pole5 = 0.2000 + 0.3000i
Pole6 = 0.2000 - 0.3000i

Also in this case, the total system is stable.

Then if we take $N_c$ = 10, the system becomes unstable. (one of the poles goes outside the unit circle). The poles are:

Pole1 = – 0.1722
Pole2 = 1.0415 ← unstable pole
Pole3 = 0.4414
Pole4 = 0.8000
Pole5 = 0.2000 + 0.3000i
Pole6 = 0.2000 - 0.3000i

In the next exercise, we will get a stable system by adding an end-point constraint.

## 5.5. End-point constraint

In case $N_c$ = 10, we have an unstable system. In this exercise we try to stabilize this system by adding an end-point constraint. By introducing an end-point constraint, the closed-loop system becomes asymptotically stable. Since the end-point constraint is an equality constraint, it still holds that the resulting controller is linear and time-invariant.

After we introduced the end-point constraint into the system, we get for the closed-loop poles:

Pole1 = – 0.1722
Pole2 = 0.8304 ← the pole becomes stable
Pole3 = 0.4415
Pole4 = 0.8000
Pole5 = 0.2000 + 0.3000i
Pole6 = 0.2000 - 0.3000i

Conclusion: the system is now stable.

## Exercise 6 – MPC and LMIs

### 6.1. Performance index and the predicted state

In this problem, we have the next set of equations:

$$\begin{aligned} x(k+1) &= Ax(k) + B_2 w(k) + B_3 v(k) \\ y(k) &= C_1 x(k) + D_{12} w(k) \\ z(k) &= C_2 x(k) + D_{22} w(k) + D_{23} v(k) \end{aligned}$$

Because $w(k) = 0$ and $e(k) = 0$, there follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B_3 v(k) \\ y(k) &= C_1 x(k) \\ z(k) &= C_2 x(k) + D_{23} v(k) \end{aligned}$$

Generally, a performance index can be written as follows:

$$J(k) = \sum_{j=0}^{N_s-1} \hat{z}(k+j \mid k)^T \Gamma \hat{z}(k+j \mid k)$$

In our case, we have:

$$J(k) = \sum_{j=0}^{N_s-1} \left[C_2 \hat{x}(k+j \mid k) + D_{23} \tilde{v}(k+j)\right]^T \Gamma \left[C_2 \hat{x}(k+j \mid k) + D_{23} \tilde{v}(k+j)\right]$$

Further,

$$\left(v_{ss}, x_{ss}, w_{ss}, z_{ss}\right) = (D_{ssv} w_{ss}, D_{ssx} w_{ss}, w_{ss}, 0) = (0,0,0,0)$$

and

$$\begin{aligned} v_\Delta(k) &= v(k) - v_{ss} = v(k) \\ x_\Delta(k) &= x(k) - x_{ss} = \hat{x}(k) = x_\Delta(k) \end{aligned}$$

Thus,

$$J(k) = \sum_{j=0}^{N_S-1} \left[C_2\, x(k+j \mid k) + D_{23}\, v(k+j)\right]^T \Gamma \left[C_2\, x(k+j \mid k) + D_{23}\, v(k+j)\right]$$

$$\Leftrightarrow$$

$$J(k) = \sum_{j=0}^{N_S-1} \left[x(k+j \mid k)^T C_2^T + v(k+j)^T D_{23}{}^T\right] \Gamma \left[C_2\, x(k+j \mid k) + D_{23}\, v(k+j)\right]$$

$$\Leftrightarrow$$

$$\begin{aligned} J(k) = \sum_{j=1}^{N_S-1} [ & x(k+j \mid k)^T C_2^T \Gamma C_2 x(k+j \mid k) + v(k+j)^T D_{23}^T \Gamma C_2 x(k+j \mid k) + \\ & x(k+j \mid k)^T C_2^T \Gamma D_{23} v(k+j) + v(k+j)^T D_{23}^T \Gamma D_{23} v(k+j)] \end{aligned}$$

We can use

$$z^T \Gamma D v + v^T D^T \Gamma z = 2 v^T D^T \Gamma z$$

and

$$\tilde{v}(k) = \begin{bmatrix} v(k) \\ \vdots \\ v(k+N_S-1) \end{bmatrix}$$

to obtain the performance index written in terms of state x(k) and v(k):

$$J(k) = \sum_{j=0}^{N_S-1} \left( \left[ C_2\, x(k+j \mid k) \right]^2 + \left[ D_{23} \tilde{v}(k) \right]^2 + 2\, x(k+j \mid k)^T C_2^T \Gamma D_{23} \tilde{v}(k) \right)$$

From the system equations we can also derive:

$$\hat{x}(k+N_S \mid k) = A \hat{x}(k+N_S-1 \mid k) + B_3\, v(k+N_S-1)$$

$$\Leftrightarrow$$

$$\hat{x}(k+N_S \mid k) = A \hat{x}(k+N_S-1 \mid k) + B_2 [0\,0 \ldots 1] \tilde{v}(k)$$

$$\Leftrightarrow$$

$$\hat{x}\left(k+N_S \mid k\right) = A\, x\left(k+N_S-1 \mid k\right) + B_2 [0\,0 \ldots 1] \tilde{v}(k)$$

## 6.2. LMI

First we will write out the performance index found in the previous exercise:

$$J(k) = \sum_{j=0}^{N_S-1} [\hat{x}(k+j \mid k)^T C^T \Gamma C \hat{x}(k+j \mid k) + \tilde{v}(k+j)^T D^T \Gamma D \tilde{v}(k+j) +$$
$$\hat{x}(k+j \mid k)^T C^T \Gamma D \tilde{v}(k+j) + \tilde{v}(k+j)^T D^T \Gamma C \hat{x}(k+j \mid k)]$$

Next we define a symmetric, positive definite matrix $G$, such that $x = G v$; then, $v = G^{-1} x$. After this substitution, $J(k)$ becomes:

$$J(k) = \sum_{j=0}^{N_S-1} ([G\tilde{v}(k+j)]^T C^T \Gamma C [G\tilde{v}(k+j)] + \tilde{v}(k+j)^T D^T \Gamma D \tilde{v}(k+j) +$$
$$[G\tilde{v}(k+j)]^T C^T \Gamma D \tilde{v}(k+j) + \tilde{v}(k+j)^T D^T \Gamma C [G\tilde{v}(k+j)])$$

$$\Leftrightarrow$$

$$J(k) = \sum_{j=0}^{N_S-1} (\tilde{v}(k+j)^T [G^T C^T \Gamma C G] \tilde{v}(k+j) + \tilde{v}(k+j)^T D^T \Gamma D \tilde{v}(k+j) +$$
$$\tilde{v}(k+j)^T [G^T C^T \Gamma D] \tilde{v}(k+j) + \tilde{v}(k+j)^T [D^T \Gamma C G] \tilde{v}(k+j))$$

$$\Leftrightarrow$$

$$J(k) = \sum_{j=0}^{N_S-1} \tilde{v}(k+j)^T \left[G^T C^T \Gamma C G + D^T \Gamma D + G^T C^T \Gamma D + D^T \Gamma C G\right] \tilde{v}(k+j)$$

$$\Leftrightarrow$$

$$J(k) = \sum_{j=0}^{N_S-1} \tilde{v}_\Delta(k+j)^T \left\{\left[D^T + G^T C^T\right] \Gamma \left[D + C G\right]\right\} \tilde{v}_\Delta(k+j)$$

We now translate this problem to a minimization problem with LMI constraints. If we take the quadratic function $V(k) = \tilde{v}_\Delta(k)^T P \tilde{v}_\Delta(k)$ and $P > 0$, then

$$\Delta V(k) = V(k+1) - V(k) < -\tilde{v}_\Delta{}^T(k) \, (D^T + G^T C^T) \Gamma (D + C G) \, \tilde{v}_\Delta(k)$$

Because $\Delta V(k) < 0$ and $V(k) > 0$, $V(\infty)$ must be equal to zero. This leads to $V(k) = J(k)$. $\tilde{v}_\Delta{}^T(k) P \tilde{v}_\Delta(k)$ is a Lyapunov function and also an upper bound for $J(k)$.

Next, we write $V(k)$ as:

$$\Delta V(k) = V(k+1) - V(k)$$

$$\Leftrightarrow$$

$$\Delta V(k) = \tilde{v}_\Delta{}^T(k+1) P \tilde{v}_\Delta(k+1) - \tilde{v}_\Delta{}^T(k) P \tilde{v}_\Delta(k) \quad \text{with } P = (D^T + G^T C^T) \Gamma (D + C G)$$

We know that

$$\hat{x}(k+1) = A \hat{x}(k) + B \tilde{v}(k) \quad \text{and} \quad x = G v$$
$$G \tilde{v}(k+1) = A G \tilde{v}(k) + B \tilde{v}(k)$$
$$\tilde{v}(k+1) = G^{-1} A G \tilde{v}(k) + G^{-1} B \tilde{v}(k)$$
$$\tilde{v}(k+1) = G^{-1} (A G + B) \tilde{v}(k)$$

Now we can substitute $\tilde{v}(k+1) = \tilde{v}_\Delta(k+1)$ in $\Delta V(k)$:

$$\Delta V(k) = \tilde{v}_\Delta{}^T(k) \left\{\left[G^{-1} A G + G^{-1} B\right]^T P \left[G^{-1} A G + G^{-1} B\right] - P\right\} \tilde{v}_\Delta(k)$$

This leads to the condition:

$$\tilde{v}_\Delta{}^T(k) \left\{\left[G^{-1} A G + G^{-1} B\right]^T P \left[G^{-1} A G + G^{-1} B\right] - P\right\} v_\Delta(k) +$$
$$v_\Delta{}^T(k) \, (D^T + G^T C^T) \Gamma (D + C G) \, \tilde{v}_\Delta(k) < 0$$

$$\Leftrightarrow$$

$$\left\{\left[G^{-1}AG+G^{-1}B\right]^T P\left[G^{-1}AG+G^{-1}B\right]-P+\left[D^T+G^TC^T\right]\Gamma\left[D+CG\right]\right\}<0$$

$$\Leftrightarrow$$

$$\left(\left[G^TA^T\left(G^{-1}\right)^T+B^T\left(G^{-1}\right)^T\right]P\left[G^{-1}AG+G^{-1}B\right]-P+\left[D^T+G^TC^T\right]\Gamma\left[D+CG\right]\right)<0$$

Now, we must introduce $P=\eta S^{-1}$ and also $G=Y S^{-1}$. $S^{-1}$ is symmetric, so $S^{-1}=\left(S^{-1}\right)^T$. Further, because G is symmetric and Y is a number, $G^{-1}=\left(Y S^{-1}\right)^{-1}=\frac{1}{Y}S$. Doing this, we obtain:

$$\boldsymbol{h}\left(S^{-1}YA^T\left[YS^{-1}\right]^{-1}+B^T\left[YS^{-1}\right]^{-1}\right)S^{-1}\left(\left[YS^{-1}\right]^{-1}A\left[YS^{-1}\right]+\left[YS^{-1}\right]^{-1}B\right)-$$
$$\boldsymbol{h}S^{-1}+\left[D^T+S^{-1}YC^T\right]\Gamma\left[D+CYS^{-1}\right]<0$$

We divide by $\boldsymbol{h}$:

$$\left(S^{-1}YA^T\left[YS^{-1}\right]^{-1}+B^T\left[YS^{-1}\right]^{-1}\right)S^{-1}\left(\left[YS^{-1}\right]^{-1}A\left[YS^{-1}\right]+\left[YS^{-1}\right]^{-1}B\right)-S^{-1}+$$
$$\boldsymbol{h}^{-1}\left[D^T+S^{-1}YC^T\right]\Gamma\left[D+CYS^{-1}\right]<0$$

Then, we apply pre- and post-multiplying with S:

$$\left(YA^T\left[YS^{-1}\right]^{-1}+SB^T\left[YS^{-1}\right]^{-1}\right)S^{-1}\left(\left[YS^{-1}\right]^{-1}AY+\left[YS^{-1}\right]^{-1}BS\right)-S+$$
$$\boldsymbol{h}^{-1}\left[SD^T+YC^T\right]\Gamma\left[DS+CY\right]<0$$

Next, we take into account the fact that Y is a real number:

$$\left(A^TS+SB^TY^{-1}S\right)S^{-1}\left(SA+Y^{-1}SBS\right)-S+$$
$$\boldsymbol{h}^{-1}\left[SD^T+YC^T\right]\Gamma\left[DS+CY\right]<0$$

We can rewrite bringing all the terms to the right side:

$$S-\left(A^TS+SB^TY^{-1}S\right)S^{-1}\left(SA+Y^{-1}SBS\right)-$$
$$\boldsymbol{h}^{-1}\left[SD^T+YC^T\right]\Gamma\left[DS+CY\right]>0$$

This can also be written in matrix form:

$$S-\left[A^TS+SB^TY^{-1}S\left[SD^T+YC^T\right]\Gamma^{1/2}\right]\cdot\begin{bmatrix}S^{-1} & 0\\ 0 & \eta^{-1}I\end{bmatrix}\cdot\begin{bmatrix}SA+Y^{-1}SBS\\ \Gamma^{1/2}\left[DS+CY\right]\end{bmatrix}>0$$

If we assume $S > 0$ and we use the Schur Complement Transformation, we get the next matrix:

$$\begin{bmatrix} S & A^T S + S B^T Y^{-1} S & \left[S D^T + Y C^T\right]\Gamma^{1/2} \\ S A + Y^{-1} S B S & S & 0 \\ \Gamma^{1/2}\left[D S + C Y\right] & 0 & \eta I \end{bmatrix} > 0$$

We can conclude that any , $S$ and $Y$, which satisfy this LMI, will give an upper bound:

$$V(k) = \eta \tilde{v}_{\Delta}{}^{T}(k)\, S^{-1}\, \tilde{v}_{\Delta}{}^{T}(k) \geq J(k)$$

Introducing an additional constraint $\tilde{v}_{\Delta}{}^{T}(k)\, S^{-1}\, \tilde{v}_{\Delta}{}^{T}(k) \leq 1$, finally gives:

$$\begin{bmatrix} 1 & \hat{v}_{\Delta}{}^{T}(k) \\ v_{\Delta}(k) & S \end{bmatrix} \geq 0$$

Every $S$ that fulfills this inequality will assure the additional constraint $\tilde{v}_{\Delta}{}^{T}(k)\, S^{-1}\, \tilde{v}_{\Delta}{}^{T}(k) \leq 1$. Thus, $> J(k)$.

## Exercise 7 – LMI-based MPC

### 7.1. The translation

From the given system, we must translate it in terms of the systems description of a structured model uncertainty:

$$x(k+1) = Ax(k) + B_3v(k) \quad (1)$$

$$z(k) = C_2x(k) + D_{23}v(k) \quad (2)$$

$$y(k) = C_4x(k) + D_{43}v(k) \quad (3)$$

where

$$P = \begin{bmatrix} A & B_3 \\ C_2 & D_{23} \\ C_4 & D_{43} \end{bmatrix} \in Co\left\{ \begin{bmatrix} \overline{A}_1 & \overline{B}_{3,1} \\ \overline{C}_{2,1} & \overline{D}_{23,1} \\ \overline{C}_{4,1} & \overline{D}_{43,1} \end{bmatrix}, \ldots, \begin{bmatrix} \overline{A}_L & \overline{B}_{3,L} \\ \overline{C}_{2,L} & \overline{D}_{23,L} \\ \overline{C}_{4,L} & \overline{D}_{43,L} \end{bmatrix} \right\}$$

For the state equation (1), we take the state equation from the system itself, with $B$ instead of $B_3$. Then we have the performance signal (2). To get this equation we need the performance index and that has been given as:

$$J(k) = \sum_{j=0}^{\infty} y^2(k+j) \quad (4)$$

From this performance index, we conclude that that the performance signal is equal to the output equation ($z(k) = y(k+1)$). With this, we can now substitute the system equation into the performance signal equation.

For the constraint equation (3), we first define:

$$y(k) = \begin{bmatrix} v(k) \\ y(k) \end{bmatrix},$$

Then again, we substitute the system equation in this constraint equation.

We have now the systems description and uncertainty:

$$x(k+1) = Ax(k) + B_3v(k) \quad (5)$$

$$z(k) = C_2x(k) + D_{23}v(k) \quad (6)$$

$$y(k) = \begin{bmatrix} 0 & C \end{bmatrix} x(k) + \begin{bmatrix} I & 0 \end{bmatrix} v(k) \quad (7)$$

where

$$P = \begin{bmatrix} A & B_3 \\ CB & DB \\ \begin{bmatrix} 0 & C \end{bmatrix} & \begin{bmatrix} I & 0 \end{bmatrix} \end{bmatrix} \in Co\left\{ \begin{bmatrix} \overline{A}_1 & \overline{B}_{3,1} \\ \overline{C}_{2,1} & \overline{D}_{23,1} \\ \overline{C}_{4,1} & \overline{D}_{43,1} \end{bmatrix}, \ldots, \begin{bmatrix} \overline{A}_L & \overline{B}_{3,L} \\ \overline{C}_{2,L} & \overline{D}_{23,L} \\ \overline{C}_{4,L} & \overline{D}_{43,L} \end{bmatrix} \right\}$$

## 7.2. LMI optimization problem

After we translated the system in the previous exercise, we give the optimization problem as an LMI optimization problem.

We have the system with uncertainty description (5)-(7) and a control problem of minimizing:

$$\min_{F} \max_{G \in g} J(k)$$

where $J(k)$ is defined in equation (4), subject to constraint:

$$\max_{G \in g} \left| y_{\ell}(k+j) \right| \le y_{\ell,\max} , \qquad \text{for } \ell = 1,\ldots,m, \quad \forall j \ge 1$$

where

$$y_{\ell,\max} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

The state feedback $F = Y S^{-1}$ minimizing the worst case $J(k)$ can be found by solving:

$$\min_{g,S,Y} g$$

subject to

(LMI for Lyapunov stability)

$$S > 0 \tag{8}$$

(LMI for ellipsoid boundary)

$$\begin{bmatrix} 1 & x^T(k \mid k) \\ x(k \mid k) & S \end{bmatrix} \ge 0 \tag{9}$$

(LMIs for worst case MPC performance index)

$$\begin{bmatrix} S & S\overline{A}_i^T + Y^T \overline{B}_{3,i}^T & (S\overline{C}_{2,i}^T + Y^T \overline{D}_{23,i}^T)\Gamma^{1/2} \\ \overline{A}_i S + \overline{B}_{3,i} Y & S & 0 \\ \Gamma^{1/2}(\overline{C}_{2,i} S + \overline{D}_{23,i} Y) & 0 & g\,I \end{bmatrix} \ge 0, \quad i = 1,\ldots,L \tag{10}$$

(LMIs for constraint on $y$)

$$\begin{bmatrix} S & (\overline{C}_{4,i} S + \overline{D}_{43,i} Y)^T E_{\ell}^T \\ E_{\ell}(\overline{C}_{4,i} S + \overline{D}_{43,i} Y) & y_{\ell,\max}^2 \end{bmatrix} \ge 0, \quad \ell = 1,\ldots,m, \quad i = 1,\ldots,L \tag{11}$$

where $E_{\ell} = \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}$, with the 1 on the $\ell$ - the position.

LMI (8) guarantees Lyapunov stability, LMI (9) describes an invariant ellipsoid for the state, (10) gives the LMIs for the worst case MPC criterion and (11) correspond to the state constraint on $y$.

## Exercise 8 – MPC of a distillation column

First, the factors $N_1$, $N_2$ and $N_C$ are approximated with the aid of the rules in chapter 7 of the book Model Predictive Control.

In this manner, $N_1$ should be equal to $1+d$, where $d$ is the dead time. We looked in the figure and concluded there was no dead time, so we took as initial value $N_1$=1.
$N_2$ should cover the most of the dynamics of the system. We took the value of 20, because after 20 steps, there was no more substantial change in the dynamics.
$N_c$ was chosen as follows: first, the size of the $A_O$ matrix was found to be 5. $A_O$ is for the IIO, so $n$=6. We were tempted to choose for $N_c$ a value just above 6. We noticed that the system might need more degrees of freedom because of the hard constraints concerning impurity and purity (max 1% near the hard limits). This is why we have chosen a value bigger than $N_2/2$. For instance, we choose $N_C$=11 and later we will find a minimum around this value. Of course, the bigger $N_C$ is, the better the system will approximate but the more worthless our computer would become because of the calculation time[1].

### 8.1. Tuning the controller for the linear model

We have run the program seen in *Appendix D* to tune the parameters $N_1$, $N_2$ and $N_C$ around the approximated values. We have done this for the linear model.

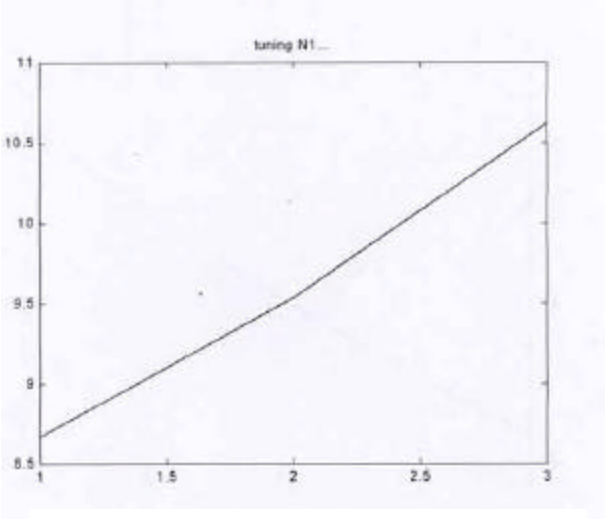



Figure 8.1. The minimum of $N_1$ is 1

[1] The calculations of this exercise were made with two computers, a Pentium II and a Pentium III. Still, the calculations took several hours for exercises 8.2 and 8.3.

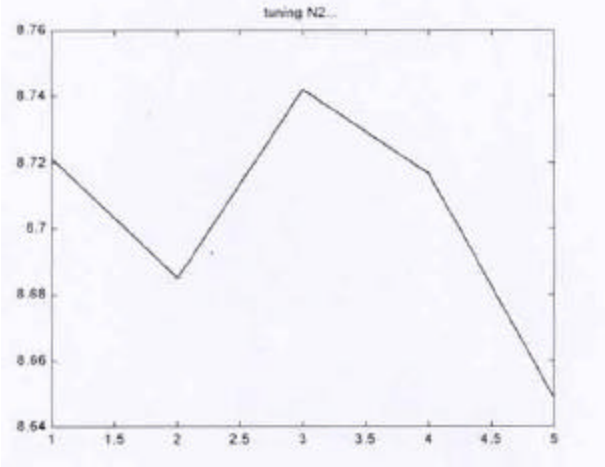



Figure 8.2 The minimum of $N_2$ is (18+1=) 19.

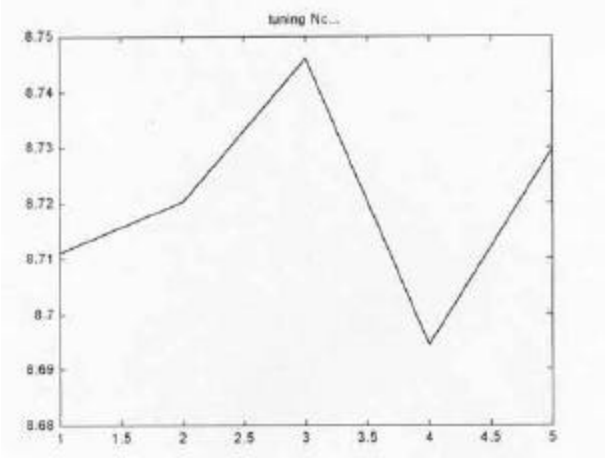



Figure 8.3. The minimum of $N_C$ is (9+3=) 12.

As can be seen in the following three pictures, the best values are:

$N_1$=1

$N_2$=19

$N_C$=12

We can see in the next figure the deviations of the two outputs for $N_1$=1, $N_2$=19 and $N_C$=12.

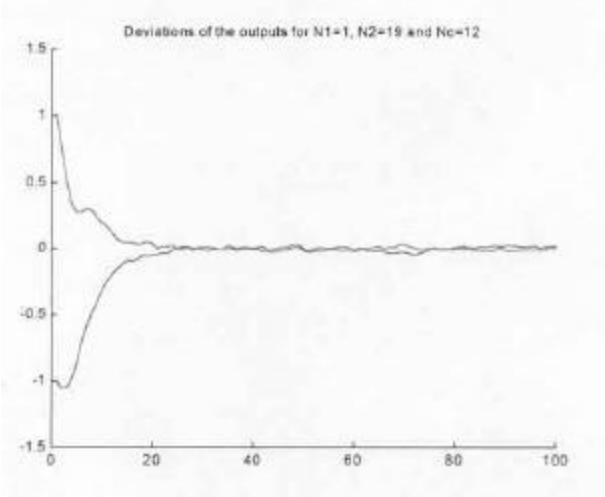



Figure 8.4. Deviation of the outputs for the linear system (**lin**).

## 8.2 Tuning the controller for the nonlinear model in the correct working point

Now we do the same as in 8.1, with slight some differences:

- we will take some smaller ranges for searching the best value
- the optimal values found in 8.1 will be used as reference

The results can be seen in the next three figures:

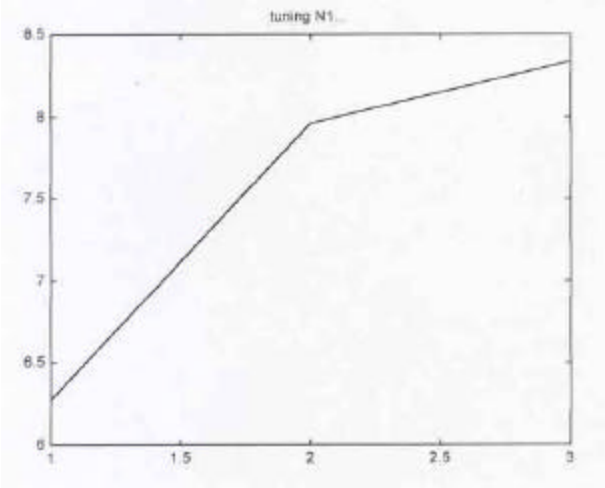



Figure 8.5. The minimum value of $N_1$ is 1.

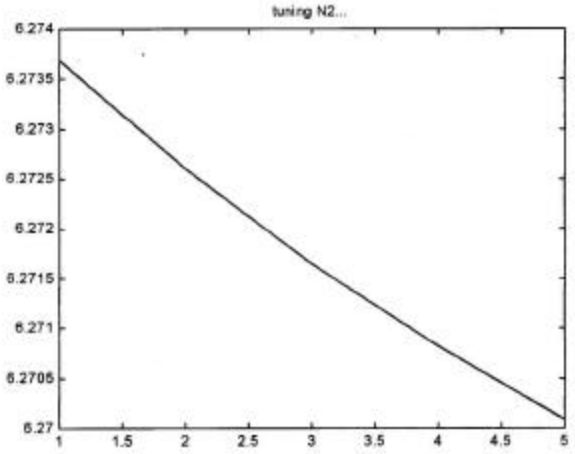



Figure 8.6. Values of $N_2$ (from 18 to 22).

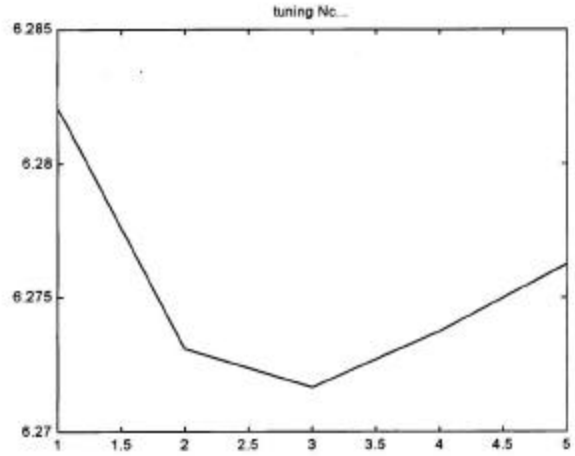



Figure 8.7. The minimum value of $N_C$ is (10+2=) 12.

We have the minima:

$N_1$=1

$N_2$=19 (here we chose for 19 again because the graphic is almost horizontal)

$N_C$=12

Using these parameters, we plot again the deviation of the outputs:

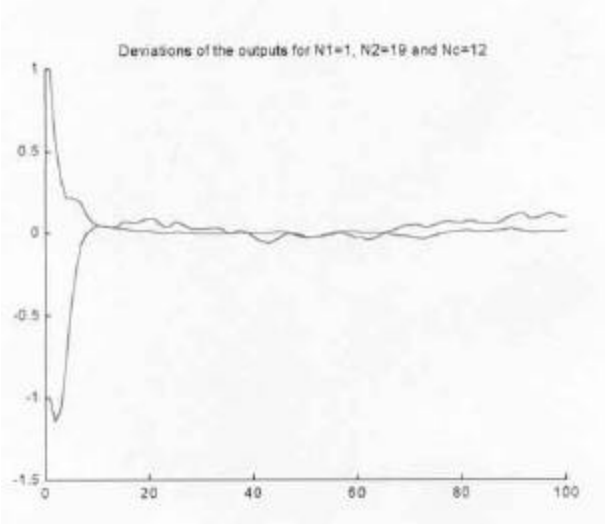



Figure 8.8. Deviation of the outputs for the first non-linear system (**nl1**).

We notice that one output remains in well stabilized, but that the other one slightly drifts around the desired value.

## 8.3 Tuning the controller for the nonlinear model not in the correct working point

In the last part, we need to better tune the system. Therefore, we will use the same method as above. Further, we will change $W_u$ and $W_y$. We cannot change them too much, otherwise the model will not take into account the noise. In the reality this could give problems and the model might not represent the real system with the desired accuracy. We will use the optimal values found in 8.2 for $N_1$, $N_2$ and $N_C$.

Once again, here are the three figures we obtained:

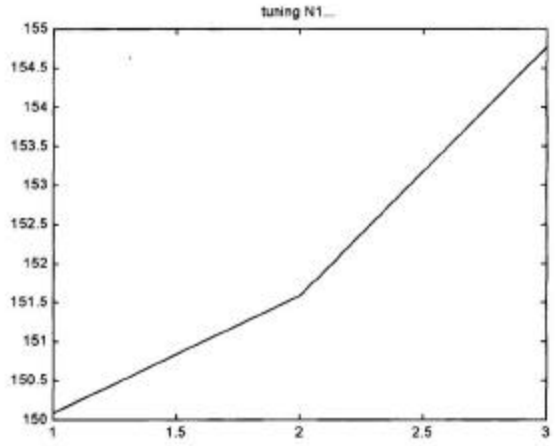



Figure 8.9. The minimum value of $N_1$ is 1.

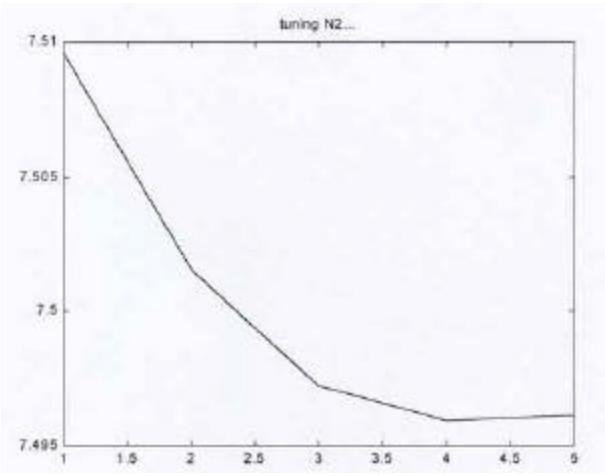



Figure 8.10. The minimum value of $N_2$ is 21.

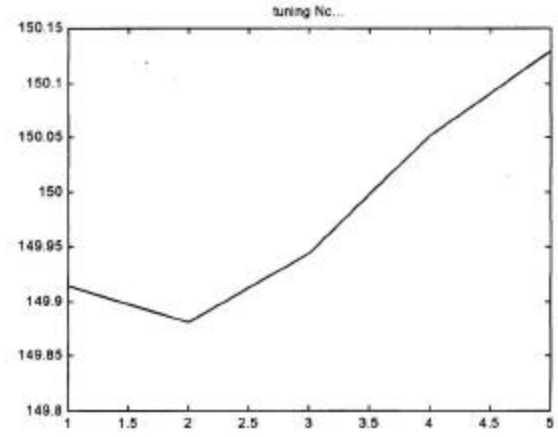



Figure 8.11. The minimum value of $N_C$ is 11.

We get the following values:

$N_1$=1

$N_2$=21

$N_C$=11

Above that, we will slightly change the weighting matrices $W_u$ and $W_y$.

We chose the matrices $W_u$ and $W_y$ with nonzero components between 0.5 and 1.0 in steps of 0.1.

We get the next two figures, from which we can derive that the best values of $W_u$ and $W_y$ are

$W_u = W_y = 0.9 * \boldsymbol{I}$ (where $\boldsymbol{I}$ s the identity matrix).

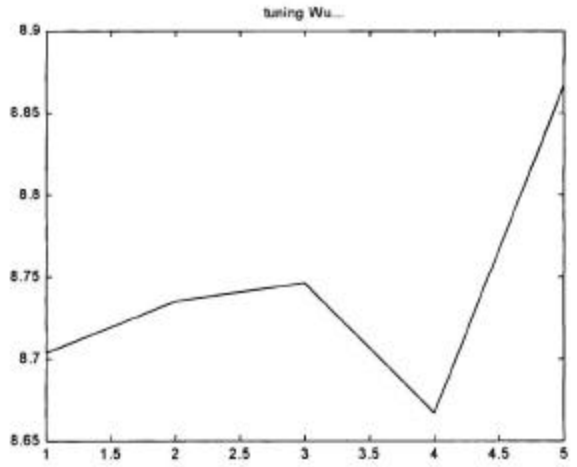



Figure 8.12. Minimum of $W_u$.

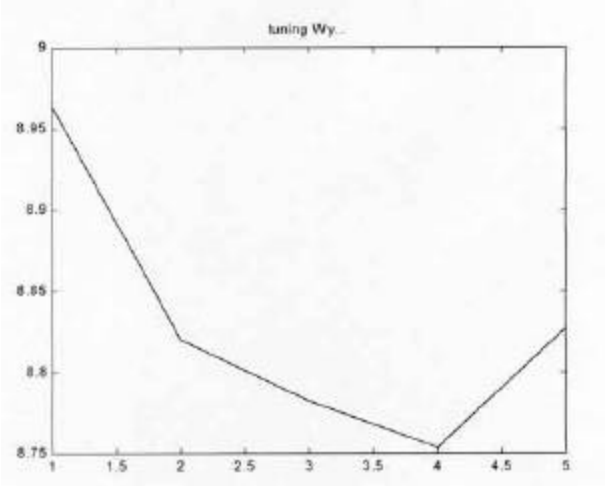



Figure 8.13. Mimimum of $W_y$.

From this figures, we can derive that the best values of $W_u$ and $W_y$ are $W_u = W_y = 0.9*\boldsymbol{I}$.

Conclusion:

We deduced that the following values should be used:

$N_1$=1

$N_2$=22

$N_C$=11

$W_u = 0.9*\boldsymbol{I}$

$W_y = 0.9*\boldsymbol{I}$

Finally, we give the deviation of the outputs. We can observe that the deviations are bigger than if the good working point was chosen (**nl1**):

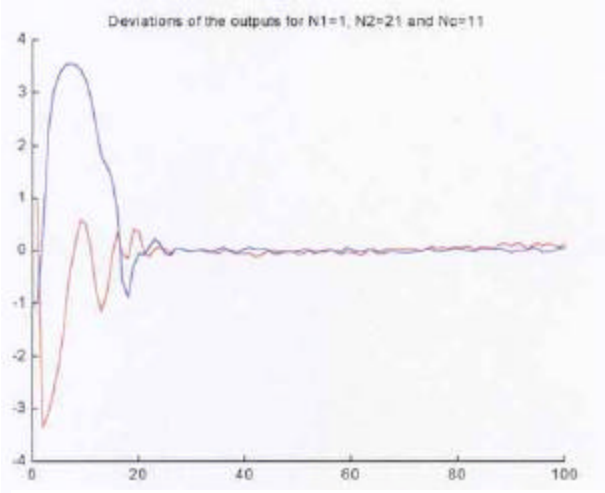



Figure 8.14. Deviation of the outputs for the first non-linear system (**nl2**).

## Appendix A – exer1.m (m-file of exercise 1)

```
%     M.Patrascu  &  J. Hidajat
%     9643118     &  9338019
%     Model Predictive Control (ET4-096)
%     EXERCISE 1 models & prediction

%     ------------------------------------------------------------------
%     1.1 Impulse response
%     ------------------------------------------------------------------

%     we can change these values to study the response,
%     nimp=(10,15,20,25)and npol=nss=(1,2,3,4) respectively
clear;
nimp = 40;
npol = 4;
nss  = 4;
lab1;

figure(11);
%     state space representation of true proces:
%     [G,dim]=ss2sys(A,B1,B2,B3,C1,D11,D12,D13)
[G_true,dim_true]=ss2sys(At,Kt,Lt,Bt,Ct,0,0,0);
%     k=0,...,20
impulse(G_true(1,2),20,'c');
hold on;
%     impulse-response model:
%     [G,dim]=imp2sys(h,f,nimp)
[G_imp,dim_imp]=imp2sys(hgo,hfo,nimp);
impulse(G_imp(1,2),20,'r');
hold on;
%     polynominal model:
%     [G,dim]=tf2sys(a,b,c,f)
[G_pol,dim_pol]=tf2sys(ao,bo,co,fo);
impulse(G_pol(1,2),20,'g');
hold on;
%     state space model:
%     [G,dim]=ss2sys(A,B1,B2,B3,C1,D11,D12,D13)
[G_ss,dim_ss]=ss2sys(Ao,Ko,Lo,Bo,Co,0,0,0);
impulse(G_ss(1,2),20,'b');
title('ss-true:cyan, imp:red, tf:green, ss-model:blue');
hold on;

%     to compare the "worst"- and the "sufficient" case (figure 1.1)
clear;
nimp = 10;
npol = 1;
nss  = 1;
lab1;
figure(12);
subplot(2,1,1);
[G_true,dim_true]=ss2sys(At,Kt,Lt,Bt,Ct,  0,  0,  0);
impulse(G_true(1,2),20,'c');
hold on;
[G_imp,dim_imp]=imp2sys(hgo,hfo,nimp);
impulse(G_imp(1,2),20,'r');
```

```
hold on;
[G_pol,dim_pol]=tf2sys(ao,bo,co,fo);
impulse(G_pol(1,2),20,'g');
hold on;
[G_ss,dim_ss]=ss2sys(Ao,Ko,Lo,Bo,Co,  0,  0,  0);
impulse(G_ss(1,2),20,'b');
title('ss-true:cyan, imp:red, tf:green, ss-model:blue
(nimp=10,npol=1,nss=1)');
hold on;
clear;
nimp = 20;
npol = 3;
nss  = 2;
lab1;
figure(12);
subplot(2,1,2);
[G_true,dim_true]=ss2sys(At,Kt,Lt,Bt,Ct,  0,  0,  0);
impulse(G_true(1,2),20,'c');
hold on;
[G_imp,dim_imp]=imp2sys(hgo,hfo,nimp);
impulse(G_imp(1,2),20,'r');
hold on;
[G_pol,dim_pol]=tf2sys(ao,bo,co,fo);
impulse(G_pol(1,2),20,'g');
hold on;
[G_ss,dim_ss]=ss2sys(Ao,Ko,Lo,Bo,Co,  0,  0,  0);
impulse(G_ss(1,2),20,'b');
title('ss-true:cyan, imp:red, tf:green, ss-model:blue
(nimp=20,npol=3,nss=2)');
hold on;

%     ------------------------------------------------------------------
%     1.3 IO models
%     ------------------------------------------------------------------

dim_imp
dim_pol
dim_ss
disp(' ')

%     ------------------------------------------------------------------
%     1.4 IIO models
%     ------------------------------------------------------------------

%     we transform the IO-models into IIO-models
%     [Gi,dimi]=io2iio(Go,dimo)
[G_imp_iio,dim_imp_iio]=io2iio(G_imp,dim_imp);
[G_pol_iio,dim_pol_iio]=io2iio(G_pol,dim_pol);
[G_ss_iio,dim_ss_iio]=io2iio(G_ss,dim_ss);
dim_imp_iio
dim_pol_iio
dim_ss_iio
disp(' ')

%     ------------------------------------------------------------------
%     1.5 Prediction models
%     ------------------------------------------------------------------
```

```
%     N=15, number of steps to be calculated.
N=15;

%     we calculate the four predictions as follows:
%     [sysp,dimp]=pred(sys,dim,N)
[sysp_imp,dimp_imp]=pred(G_imp_iio,dim_imp_iio,N);
[sysp_pol,dimp_pol]=pred(G_pol_iio,dim_pol_iio,N);
[sysp_ss,dimp_ss]=pred(G_ss_iio,dim_ss_iio,N);
dimp_imp
dimp_pol
dimp_ss
disp(' ')

%     with size(sys), we get the explanation of the model.
disp('The impulse response model =')
size(sysp_imp)
disp(' ')
disp('The polynomial model =')
size(sysp_pol)
disp(' ')
disp('The state space model =')
size(sysp_ss)
disp(' ')

%     ------------------------------------------------------------------
%     1.6 Simulation of prediction model
%     ------------------------------------------------------------------

%     for the calculation, we need to transform all the three models in
%     state space format.
%     [A,B1,B2,B3,C1,C2,C3,C4]=syst2ss(sys,dim)
[A_i,B1_i,B2_i,B3_i,C1_i,C2_i,C3_i,C4_i]=syst2ss(sysp_imp,dimp_imp);
[A_p,B1_p,B2_p,B3_p,C1_p,C2_p,C3_p,C4_p]=syst2ss(sysp_pol,dimp_pol);
[A_s,B1_s,B2_s,B3_s,C1_s,C2_s,C3_s,C4_s]=syst2ss(sysp_ss,dimp_ss);

%     we initiate some variables that we need for the calculation.
xo_i=zeros(length(A_i),1);
xo_p=zeros(length(A_p),1);
xo_s=zeros(length(A_s),1);
y_i=zeros(N+1,1);
y_p=zeros(N+1,1);
y_s=zeros(N+1,1);
%     v=delta_u
v=[ones(1,1) zeros(1,N)]';

%     after the initiation, we can calculate the predicted output by
using
%     these equations:
%     xo(k+1)=A.xo(k)+B3_tilda.v_tilda(k)
%     y_tilda(k+1)=C1_tilda.(A.xo(k)+B3_tilda.ones)

for k=1:N,
   xo_i=A_i*xo_i+B3_i*v;
   y1=C1_i*xo_i;
   y_i=[y_i,y1];
end
```

```
for k=1:N,
   xo_p=A_p*xo_p+B3_p*v;
   y1=C1_p*xo_p;
   y_p=[y_p,y1];
end

for k=1:N,
   xo_s=A_s*xo_s+B3_s*v;
   y1=C1_s*xo_s;
   y_s=[y_s,y1];
end

%     we take only the first element of each output.
Y_imp=y_i(1,:);
Y_i=diff(Y_imp);
Y_pol=y_p(1,:);
Y_p=diff(Y_pol);
Y_ss=y_s(1,:);
Y_s=diff(Y_ss);

%     it's plotting time!
k=0:N-1;
figure(13);
[kk_i,YY_i]=stairs(k,Y_i);
[kk_p,YY_p]=stairs(k,Y_p);
[kk_s,YY_s]=stairs(k,Y_s);
plot(kk_i,YY_i,'r',kk_p,YY_p,'g',kk_s,YY_s,'b');
title('imp:red, tf:green, ss-model:blue');
xlabel('k');
ylabel('output');
hold on;
```

## Appendix B – exer4.m (m-file of exercise 4)

```
%     M.Patrascu  &  J. Hidajat
%     9643118     &  9338019
%     Model Predictive Control (ET4-096)
%     EXERCISE 4 GPC & LQPC
function exer4(num);

if nargin==0;
   disp(' ');
   disp('Make choice (1-2)');
   disp(' ');
else
   if num==1;
      exer41;
   else
      if num==2;
         exer42;
      else
         disp(' ');
         disp('Make choice (1-5)');
         disp(' ');
      end;
   end;
end;


%     ------------------------------------------------------------------
%     4.1 GPC controller design
%     ------------------------------------------------------------------
function exer41;
clear;

%     Polynomial IO model
%     The calculation of the polynomials.
a1=tf([-1.5 1],1);
a2=tf([-0.7 1],1);
a3=tf([-0.2 1],1);
ao_tf=a1*a2*a3;
ao = [1 -2.4 1.49 -0.21];
%     ao_tf = -0.21 s^3 + 1.49 s^2 - 2.4 s + 1
b1=tf([1.5 0],1);
b2=tf([-1.2 1],1);
b3=tf([-0.5 1],1);
bo_tf=b1*b2*b3;
bo = [0 1.5 -2.55 0.9];
%     bo_tf = 0.9 s^3 - 2.55 s^2 + 1.5 s
c1=tf([-0.8 1],1);
c2=tf([0.13 -0.4 1],1);
co_tf=c1*c2;
co = [1 -1.2 0.45 -0.104];
%     co_tf = -0.104 s^3 + 0.45 s^2 - 1.2 s + 1
f1=tf([0.1 0],1);
f2=tf([-0.7 1],1);
f3=tf([-0.2 1],1);
fo_tf=f1*f2*f3;
```

```
fo = [0 0.1 -0.09 0.014];
%     fo_tf = 0.014 s^3 - 0.09 s^2 + 0.1 s

%     we transform the above polynomial IO model into a polynomial
%     IIO model.
ai = conv([1 -2.4 1.49 -0.21],[1 -1]);
bi = [0 1.5 -2.55 0.9 0];
ci = [1 -1.2 0.45 -0.104 0];
fi = [0 0.1 -0.09 0.014 0];

%     then we initiate some variables to obtain the optimal GPC
controller.
%  [y,du]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr)
P=1;
lambda=1;
N1=1;
N2=25;
Nc=2;
lensim=50;
r=[zeros(1,9) ones(1,100)];
di=[zeros(1,19) ones(1,1) zeros(1,89)];
ei_1=randn(1,50);
ei=0.005*(ei_1/std(ei_1));
%     we study the effect by changing dumax = {0.25,0.5,1}, with apr=4
dumax025=0.25;
dumax050=0.50;
dumax100=1.00;
apr=4;

[y025,du025]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax025,
apr);
[y050,du050]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax050,
apr);
[y100,du100]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax100,
apr);

%     after calculation, we transform it back to IO model.
u025 = cumsum(du025);
u050 = cumsum(du050);
u100 = cumsum(du100);
do   = cumsum(di(1:50));
eo   = cumsum(ei(1:50));

%     plotting time!
t=1:50;

%     for dumax = 0.25
figure(41);
subplot(2,1,1);
stairs(t,[y025' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (dumax=0.25)');
grid;
subplot(2,1,2);
stairs(t,[du025' u025']);
title('blue: du, green: u');
grid;
```

```
%     for dumax = 0.50
figure(42);
subplot(2,1,1);
stairs(t,[y050' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (dumax=0.50)');
grid;
subplot(2,1,2);
stairs(t,[du050' u050']);
title('blue: du, green: u');
grid;

%     for dumax = 1.00
figure(43);
subplot(2,1,1);
stairs(t,[y100' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (dumax=1.00)');
grid;
subplot(2,1,2);
stairs(t,[du100' u100']);
title('blue: du, green: u');
grid;

%     then we study the effect by changing apr = {1,2,3,4}, without
constraint
dumax = 0;
apr1  = 1;
apr2  = 2;
apr3  = 3;
apr4  = 4;

[y1,du1]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr1);
[y2,du2]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr2);
[y3,du3]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr3);
[y4,du4]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr4);
u1 = cumsum(du1);
u2 = cumsum(du2);
u3 = cumsum(du3);
u4 = cumsum(du4);

figure(44);
subplot(2,2,1);
stairs(t,[y1' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (apr=1)');
grid;
subplot(2,2,2);
stairs(t,[y2' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (apr=2)');
grid;
subplot(2,2,3);
stairs(t,[y3' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (apr=3)');
grid;
subplot(2,2,4);
stairs(t,[y4' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (apr=4)');
grid;
figure(45);
```

```
subplot(2,2,1);
stairs(t,[du1' u1']);
title('blue: du, green: u (apr=1)');
grid;
subplot(2,2,2);
stairs(t,[du2' u2']);
title('blue: du, green: u (apr=2)');
grid;
subplot(2,2,3);
stairs(t,[du3' u3']);
title('blue: du, green: u (apr=3)');
grid;
subplot(2,2,4);
stairs(t,[du4' u4']);
title('blue: du, green: u (apr=4)');
grid;

%     ------------------------------------------------------------------
%     4.2 LQPC controller design
%     ------------------------------------------------------------------
function exer42;
clear;

%     Polynomial IO model
ao = [1 -2.4 1.49 -0.21];
bo = [0 1.5 -2.55 0.9];
co = [1 -1.2 0.45 -0.104];
fo = [0 0.1 -0.09 0.014];

%     we transform the polynomial IO model into state space IO model.
[G,dim]=tf2sys(ao,bo,co,fo);
[Ao,Ko,Lo,Bo,Co,D11,D12,D13]=syst2ss(G,dim);

%     then we initiate some variables to obtain the optimal LQPC
controller.
%
   [y,u,x]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R,N1,N2,Nc,lensim,xo,do,eo,umax,apr);
N1=1;
N2=10;
Nc=10;
lensim=25;
do=zeros(1,25);
e_o=randn(1,25);
eo=(e_o/std(e_o))*0.25;
xo=[[1;1;1] zeros(3,100)];
umax=0;
apr=1;

Q=[Co]'*Co;
R001=0.1;
R010=1.0;
R100=10;

[y001,u001,x001]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R001,N1,N2,Nc,lensim,xo,do,eo,um
ax,apr);
[y010,u010,x010]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R010,N1,N2,Nc,lensim,xo,do,eo,um
ax,apr);
```

```
[y100,u100,x100]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R100,N1,N2,Nc,lensim,xo,do,eo,um
ax,apr);

t=1:25;
figure(46);
subplot(3,1,1);
stairs(t,[y001' eo']);
title('blue: y, green: eo (R=0.1)');
grid;
subplot(3,1,2);
stairs(t,[y010' eo']);
title('blue: y, green: eo (R=1.0)');
grid;
subplot(3,1,3);
stairs(t,[y100' eo']);
title('blue: y, green: eo (R=10)');
grid;

%     now we study the effect of different initial states for R=1
R=1;
xo1=[[1;1;1] zeros(3,100)];
xo2=[[0;0;0] zeros(3,100)];
xo3=[[-1;1;-1] zeros(3,100)];
xo4=[[-1;-1;-1] zeros(3,100)];
[y1,u1,x1]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R,N1,N2,Nc,lensim,xo1,do,eo,umax,apr);
[y2,u2,x2]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R,N1,N2,Nc,lensim,xo2,do,eo,umax,apr);
[y3,u3,x3]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R,N1,N2,Nc,lensim,xo3,do,eo,umax,apr);
[y4,u4,x4]=lqpc(Ao,Ko,Lo,Bo,Co,Q,R,N1,N2,Nc,lensim,xo4,do,eo,umax,apr);
figure(47);
subplot(2,2,1);
stairs(t,[y1' eo']);
title('blue: y, green: eo (xoT=[1 1 1])');
grid;
subplot(2,2,2);
stairs(t,[y2' eo']);
title('blue: y, green: eo (xoT=[0 0 0])');
grid;
subplot(2,2,3);
stairs(t,[y3' eo']);
title('blue: y, green: eo (xoT=[-1 1 -1])');
grid;
subplot(2,2,4);
stairs(t,[y4' eo']);
title('blue: y, green: eo (xoT=[-1 -1 -1])');
grid;
```

## Appendix C – exe5.m (m-file of exercise 5)

```
%     M.Patrascu  &  J. Hidajat
%     9643118     &  9338019
%     Model Predictive Control (ET4-096)
%     EXERCISE 5 Tuning GPC & LQPC
function exer5(num);

if nargin==0;
   disp(' ');
   disp('Make choice (1-5)');
   disp(' ');
else
   if num==1;
      exer51;
   else
      if num==2;
         exer52;
      else
         if num==3;
            exer53;
         else
            if num==4;
               exer54;
            else
               disp(' ');
               disp('Make choice (1-4)');
               disp(' ');
            end;
         end;
      end;
   end;
end;


%     ------------------------------------------------------------------
%     5.1 Tuning of a GPC controller
%     ------------------------------------------------------------------

function exer51;
clear;

%     Polynomial IO model
ao = [1 -2.4 1.49 -0.21];
bo = [0 1.5 -2.55 0.9];
co = [1 -1.2 0.45 -0.104];
fo = [0 0.1 -0.09 0.014];

%     we transform the above polynomial IO model into a polynomial IIO
model.
ai = conv([1 -2.4 1.49 -0.21],[1 -1]);
bi = [0 1.5 -2.55 0.9 0];
ci = [1 -1.2 0.45 -0.104 0];
fi = [0 0.1 -0.09 0.014 0];

Ts=0.01;
```

```
[G,dim]=tf2sys(ao,bo,co,fo);
[A,B1,B2,B3,C1,D11,D12,D13]=syst2ss(G,dim);
B=[B1 B2 B3];
C=C1;
D=[D11 D12 D13];

%     The transformation in a continous time model.
%     [Ac,Bc,Cc,Dc]=d2cm(A,B,C,D,Ts,'method')
%     we use this command with Ts = 0.01 and method -> 'zoh'.
%     'zoh' = zero-order transformation.
[Ac,Bc,Cc,Dc]=d2cm(A,B,C,D,Ts,'zoh');

%     system is stable if all of the eigenvalues of A matrix in LHP.
disp(' ');
disp('The eigenvalue of A:'); eig(Ac)
disp(' ');
%     conclusion: system is unstable.

Gc=ss(Ac,Bc,Cc,Dc);

%     Now we find the bandwidth of the system.
G = Cc*inv((1*eye(length(Ac))-Ac))*Bc+Dc;
G = G';
for w=2:1000,
   Ga = Cc*inv((w*eye(length(Ac))-Ac))*Bc+Dc;
   G  = [G Ga'];
end;
G  = G(3,:);
G  = ((abs(G)).^2)./((abs(G(1))).^2);
%plot(G); grid; axis([0 1000 0 1]);
%     we can plot G and from the plot we can conclude that wb = 180.
wb = 180;
%     the sampling frequency is:
ws = 2*pi/Ts;

%     from exercise 4.1 we know that the deadtime of the process is
equal to 1
%     and the system is a non-minimum system.
d=1;

%     then we initiate some variables to obtain the optimal GPC
controller.
%
   [y,du]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr)
P=1;
lambda=0.01;      %  lambda > 0, but as small as possible.
N1=1+d;
N2=round(2*ws/wb);
Nc=length(A)+1;
lensim=50;
r=[zeros(1,9) ones(1,100)];
di=[zeros(1,19) ones(1,1) zeros(1,89)];
ei_1=randn(1,50);
ei=0.005*(ei_1/std(ei_1));
dumax=0;
apr=1;
```

```
[y,du]=gpc(ai,bi,ci,fi,P,lambda,N1,N2,Nc,lensim,r,di,ei,dumax,apr);
disp(' ');

u  = cumsum(du);
do = cumsum(di(1:50));
eo = cumsum(ei(1:50));

t=1:lensim;
figure(51);
subplot(2,1,1);
stairs(t,[y' r(:,1:50)' do' eo']);
title('green: r, blue: y, red: do, cyan: eo (TUNING)');
grid;
subplot(2,1,2);
stairs(t,[du' u']);
title('blue: du, green: u');
grid;


%     ------------------------------------------------------------------
%     5.2 Tuning of an LQPC controller
%     ------------------------------------------------------------------
function exer52;
clear;

%     Polynomial IO model
ao = [1 -2.4 1.49 -0.21];
bo = [0 1.5 -2.55 0.9];
co = [1 -1.2 0.45 -0.104];
fo = [0 0.1 -0.09 0.014];

%     we transform the polynomial IO model into state space IO model.
[G,dim]=tf2sys(ao,bo,co,fo);
[A,B1,B2,B3,C1,D11,D12,D13]=syst2ss(G,dim);
B=[B1 B2 B3];
C=C1;
D=[D11 D12 D13];

%     from the previous exercise we have also: wb = 180.
wb = 180;
%     the sampling frequency is:
Ts = 0.01;
ws = 2*pi/Ts;

%     from exercise 4.2 we know that the deadtime of the process is
equal to 1
%     and the system is a non-minimum system.
d=1;

%     then we initiate some variables to obtain the optimal LQPC
controller.
%     [y,u,x]=lqpc(A,K,L,B,C,Q,R,N1,N2,Nc,lensim,xo,do,eo,umax,apr);
N1=1+d;
N2=round(2*ws/wb);
Nc=length(A);
lensim=25;
do=zeros(1,lensim);
```

```
e_o=randn(1,lensim);
eo=(e_o/std(e_o))*0.25;
xo=[[1;1;1] zeros(3,100)];
umax=0;
apr=4;
rhc=0;

%     we consider it's a SISO system, so: Q=C'C and R=lambda^2.
Q=C'*C;
R=0.01^2;

[y,u,x]=lqpc(A,B1,B2,B3,C,Q,R,N1,N2,Nc,lensim,xo,do,eo,umax,apr,rhc);
disp(' ');

t=1:lensim;
figure(52);
stairs(t,[y' eo']);
title('blue: y, green: eo (TUNING)');
grid;


%     ------------------------------------------------------------------
%     5.3 Elevator example
%     ------------------------------------------------------------------
function exer53;
clear;

%     P(q) must be given in state space representation!
P=1;
[Ap,Bp,Cp,Dp]=tf2ss(P,[1 zeros(1,length(P)-1)]);
%     A discrete-time SS model with sample time 1.
sysP = ss(Ap,Bp,Cp,Dp,1);

%     Initialization of other parameters.
lambda=0.1;
N1=1;
N2=30;
Nc=30;
N=N2-1;
lensim=100;
r=ones(1,lensim);
umax=0.4;
xmax=0.4;
ymax=1.01;
Einv=[xmax 0 0;0 xmax 0;0 0 ymax];
E=inv(Einv);

%     The system in SYSTEM format.
T=0.1;
Ao=[1 0 0;T 1 0;(T^2)/2 T 1];
Bo=[T;(T^2)/2;(T^3)/6];
Ko=[0.1*T;T+(T^2)/20;T/2+(T^2)/6+(T^3)/60];
Co=[0 0 1];
B2=zeros(length(Ao),1);

na = length(Ao);
u  = zeros(1,lensim);
```

```
y  = zeros(1,lensim);
e_1= randn(1,lensim);
e  = 0.001*(e_1/std(e_1));
x  = zeros(na,lensim+1);
xc = zeros(na,lensim+1);

disp(' ');
disp(' ss2sys   <<');     [G,dim]     = ss2sys(Ao,Ko,B2,Bo,Co,1,0,0);
disp(' gpc2spc  <<');     [sys,dim]   = gpc2spc(G,dim,sysP,lambda);
disp(' dgamma   <<');     [dGam]      = dgamma(N1,N2,Nc,dim);
disp(' external <<');     tw          = external(0,r,'gpc',4,lensim,N);
disp(' add_x    <<');     [sys,dim]   = add_x(sys,dim,E,0);
disp(' pred     <<');     [sysp,dimp] = pred(sys,dim,N);
disp(' add_nc   <<');     [sysp,dimp] = add_nc(sysp,dimp,dim,Nc,'iio');
disp(' add_u    <<');     [sysp,dimp] = add_u(sysp,dimp,umax,0);
disp(' add_y    <<');     [sysp,dimp] = add_y(sysp,dimp,ymax,1);
                    [vv,HH,ff,AA,bb] = contr(sysp,dimp,dim,dGam);
disp(' simul    <<');     [x,xc,y,u]  =
simul(sys,sys,dim,N,x,xc,y,u,tw,e,1,lensim,vv,HH,ff,AA,bb);
disp(' ');

figure(53);
t=0:lensim;
plot(t,[u(1) u],'c');
hold on;
plot(t,xc(1,:),'r');
plot(t,xc(2,:),'g');
plot(t,xc(3,:),'b');
title('input: cyan, position: blue, speed: green, acceleration: red');
grid;
axis([0 100 -0.6 1.2]);

%     ------------------------------------------------------------------
%     5.4 Stability analysis
%     ------------------------------------------------------------------
function exer54;
clear;

ao = [1 -2.4 1.49 -0.21];
bo = [0 1.5 -2.55 0.9];
co = [1 -1.2 0.45 -0.104];
fo = [0 0.1 -0.09 0.014];

N1=1;
N2=15;
Nc02=2;
Nc05=5;
Nc10=10;
N=N2-1;
lensim=25;
do=zeros(1,25);
e_o=randn(1,25);
eo=(e_o/std(e_o))*0.25;
xo=[[1;1;1] zeros(3,100)];
umax=0;
apr=1;
```

```
[G,dim]=tf2sys(ao,bo,co,fo);
[A,B1,B2,B3,C1,D11,D12,D13]=syst2ss(G,dim);
Q=[C1]'*C1;
R=0.01;

[na,nK]=size(B1);
[nL,nB]=size(B2'*B3);
u  = zeros(1,lensim);
y  = zeros(1,lensim);
x  = [xo,zeros(na,lensim)];
xc = [xo,zeros(na,lensim)];

disp('  ');
disp(' lqpc2spc <<');     [sys,dim]   = lqpc2spc(G,dim,Q,R);
disp(' external <<');     [tw]        =
external(do,[],'lqpc',apr,lensim,N);
disp(' dgamma02 <<');     [dGam02]    = dgamma(N1,N2,Nc02,dim);
disp(' dgamma05 <<');     [dGam05]    = dgamma(N1,N2,Nc05,dim);
disp(' dgamma10 <<');     [dGam10]    = dgamma(N1,N2,Nc10,dim);
disp(' pred     <<');     [sysp,dimp] = pred(sys,dim,N);
disp(' add_Nc02 <<');     [sy02,dim02]=
add_nc(sysp,dimp,dim,Nc02,'IO');
disp(' add_Nc05 <<');     [sy05,dim05]=
add_nc(sysp,dimp,dim,Nc05,'IO');
disp(' add_Nc10 <<');     [sy10,dim10]=
add_nc(sysp,dimp,dim,Nc10,'IO');
disp(' contr02  <<');[vv02,HH02,ff02,AA02,bb02] =
contr(sy02,dim02,dim,dGam02);
disp(' contr05  <<');[vv05,HH05,ff05,AA05,bb05] =
contr(sy05,dim05,dim,dGam05);
disp(' contr10  <<');[vv10,HH10,ff10,AA10,bb10] =
contr(sy10,dim10,dim,dGam10);
disp(' lticon02
<<');[Ac02,B1c02,B2c02,Cc02,D1c02,D2c02]=lticll(sys,dim,N,vv02);
disp(' lticon05
<<');[Ac05,B1c05,B2c05,Cc05,D1c05,D2c05]=lticll(sys,dim,N,vv05);
disp(' lticon10
<<');[Ac10,B1c10,B2c10,Cc10,D1c10,D2c10]=lticll(sys,dim,N,vv10);
disp('  ');

poles02 = eig(Ac02)
poles05 = eig(Ac05)
poles10 = eig(Ac10)


%     ------------------------------------------------------------------
%     5.5 End-point constraint
%     ------------------------------------------------------------------

disp(' add_end  <<');   [sy_end,dim_end]=add_end(sys,dim,sy10,dim10,N);
disp(' contr    <<');[vv,HH,ff,AA,bb]=
contr(sy_end,dim_end,dim,dGam10);
disp(' lticon   <<');[Ac,B1c,B2c,Cc,D1c,D2c]=lticll(sys,dim,N,vv);
disp('  ');

poles  = eig(Ac)
```

## Appendix D – exe8.m (m-file of exercise 8)

```
%  Exercise 8

function exer8(num);
clc;
if nargin==0;
   disp(' ');
   disp('Make choice (1-3)');
   disp(' ');
else
   if num==1;
      exer81;
   else
      if num==2;
         exer82;
      else
         if num==3;
            exer83;
               else
                  disp(' ');
                  disp('Make choice (1-3)');
                  disp(' ');
               end;
       end;
    end;
 end;

%  8.1 Tuning for the linear model (lin)

function exer81;
clear;

disp('TUNING N1? PRESS SPACE...') %TUNING N1......
pause;

%  First we initialize the process
f2ss;% Now we have the matrices A,B,K,L and C
size(Ai) %Size of the state matrix (is 5)
disp('PRESS SPACE....')
pause;
simlength=100; % Length of simulation is 100 steps
N1=1;  % we assumed there is no deadtime
N2=20; % because the greatest deal of dynamics is within 20 steps after
start
Nc=11; % the size of Ai is 5 so we need for an IIO model Nc equal to 5
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
y0=[1 -1]';

N1=(1:1:3)% We toggle the value of N1 from 1 to 3 to find a minimum

sz=size(N1)
k=zeros(i,sz(2));
for i=(1:20)
   vl=0;
```

```
   for N1=(1:1:3)
      clc;
      i
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'lin');
      y11=y1(1,:);
      y12=y1(2,:);
      vl=[vl sum(y11.^2)+sum(y12.^2)];
      a=size(vl);
      vl1=[vl(2:1:a(2))];
   end
   k(i,:)=vl1;
end
figure(1)
plot(y11, 'r')
hold on;
plot(y12, 'b')

m=size(k)
p=m(2) %number of colons
q=m(1) %number of rows

l=sum(k,1)/q;
disp('l is: ')
l
disp('Once more k')
k
figure(2)
plot(l, 'k')
title('tuning N1...')

%------------------------------------------------------------
disp('TUNING N2? PRESS SPACE....') %   TUNING N2.....
pause;
N1=1;
N2=20;
Nc=11;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];

N2=(18:1:22) %We toggle N2 to find a minimum
szz=size(N2)
kk=zeros(i,szz(2));
for ii=(1:20)
   vll=0;
   for N2=(18:1:22)
      clc;
      ii
      N1
      N2
      Nc
```

```
   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'lin');
      y11=y1(1,:);
      y12=y1(2,:);
      vll=[vll sum(y11.^2)+sum(y12.^2)];
      aa=size(vll);
      vll1=[vll(2:1:aa(2))];
   end
   kk(ii,:)=vll1;
end

figure(3)
plot(y11, 'r')
hold on;
plot(y12, 'b')

mm=size(kk)
pp=mm(2) %number of colons
qq=mm(1) %number of rows

ll=sum(kk,1)/qq;
disp('ll is: ')
ll
disp('Once more kk')
kk
figure(4)
plot(ll, 'k')
title('tuning N2...')

%-------------------------------------------------------

disp('TUNING Nc? PRESS SPACE....') %TUNING Nc......
pause;

N1=1;
N2=20;
Nc=11;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];

Nc=(9:1:13) %We toggle Nc to find a minimum
szzz=size(Nc)
kkk=zeros(i,szzz(2));
for iii=(1:40)
   vlll=0;
   for Nc=(9:1:13)
      clc;
      iii
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'lin');
      y11=y1(1,:);
```

```
      y12=y1(2,:);
      vlll=[vlll sum(y11.^2)+sum(y12.^2)];
      aaa=size(vlll);
      vlll1=[vlll(2:1:aaa(2))];
   end
   kkk(iii,:)=vlll1;
end

figure(5)
plot(y11, 'r')
hold on;
plot(y12, 'b')

mmm=size(kkk)
ppp=mmm(2) %number of colons
qqq=mmm(1) %number of rows

lll=sum(kkk,1)/qqq;
disp('lll is: ')
lll
disp('Once more kkk')
kkk
figure(6)
plot(lll, 'k')
title('tuning Nc...')

%-----------------------------------------------------
%  8.2 Tuning for the first non-linear model (nl1)

function exer82;
clear;

disp('TUNING N1? PRESS SPACE...') %TUNING N1......
pause;

f2ss;
simlength=100;
N1=1;
N2=19;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
y0=[1 -1]';

N1=(1:1:3)% We toggle the value of N1 from 1 to 3 to find a minimum

sz=size(N1)
k=zeros(i,sz(2));
for i=(1:20)
   vl=0;
   for N1=(1:1:3)
      clc;
      i
      N1
      N2
      Nc
```

```
   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl1');
      y11=y1(1,:);
      y12=y1(2,:);
      vl=[vl sum(y11.^2)+sum(y12.^2)];
      a=size(vl);
      vl1=[vl(2:1:a(2))];
   end
   k(i,:)=vl1;
end

m=size(k)
p=m(2) %number of colons
q=m(1) %number of rows

l=sum(k,1)/q;
disp('l is: ')
l
disp('Once more k')
k
figure(1)
plot(l, 'k')
title('tuning N1...')

%-------------------------------------------------------------
disp('TUNING N2? PRESS SPACE....') %   TUNING N2.....
pause;
N1=1;
N2=19;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];

N2=(18:1:22) %We toggle N2 to find a minimum
szz=size(N2)
kk=zeros(i,szz(2));
for ii=(1:20)
   vll=0;
   for N2=(18:1:22)
      clc;
      ii
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl1');
      y11=y1(1,:);
      y12=y1(2,:);
      vll=[vll sum(y11.^2)+sum(y12.^2)];
      aa=size(vll);
      vll1=[vll(2:1:aa(2))];
   end
   kk(ii,:)=vll1;
end
```

```
mm=size(kk)
pp=mm(2) %number of colons
qq=mm(1) %number of rows

ll=sum(kk,1)/qq;
disp('ll is: ')
ll
disp('Once more kk')
kk
figure(2)
plot(ll, 'k')
title('tuning N2...')

%-------------------------------------------------------
disp('TUNING Nc? PRESS SPACE....') %TUNING Nc......
pause;
N1=1;
N2=20;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];

Nc=(10:1:14) %We toggle Nc to find a minimum
szzz=size(Nc)
kkk=zeros(i,szzz(2));
for iii=(1:20)
   vlll=0;
   for Nc=(10:1:14)
      clc;
      iii
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl1');
      y11=y1(1,:);
      y12=y1(2,:);
      vlll=[vlll sum(y11.^2)+sum(y12.^2)];
      aaa=size(vlll);
      vlll1=[vlll(2:1:aaa(2))];
   end
   kkk(iii,:)=vlll1;
end
mmm=size(kkk)
ppp=mmm(2) %number of colons
qqq=mmm(1) %number of rows

lll=sum(kkk,1)/qqq;
disp('lll is: ')
lll
disp('Once more kkk')
kkk
figure(3)
plot(lll, 'k')
title('tuning Nc...')
```

```
%-----------------------------------------------------
%  8.3 Tuning for the second non-linear model (nl2)

function exer83;
clear;
disp('TUNING N1? PRESS SPACE...') %TUNING N1......
pause;

f2ss;
simlength=100;
N1=1;
N2=20;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
y0=[1 -1]';

N1=(1:1:3)% We toggle the value of N1 from 1 to 3 to find a minimum

sz=size(N1)
k=zeros(i,sz(2));
for i=(1:20)
   vl=0;
   for N1=(1:1:3)
      clc;
      i
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl2');
      y11=y1(1,:);
      y12=y1(2,:);
      vl=[vl sum(y11.^2)+sum(y12.^2)];
      a=size(vl);
      vl1=[vl(2:1:a(2))];
   end
   k(i,:)=vl1;
end

m=size(k)
p=m(2) %number of colons
q=m(1) %number of rows

l=sum(k,1)/q;
disp('l is: ')
l
disp('Once more k')
k
figure(1)
plot(l, 'k')
title('tuning N1...')

%-----------------------------------------------------------
```

```
disp('TUNING N2? PRESS SPACE....') %   TUNING N2.....
pause;
N1=1;
N2=20;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];

N2=(18:1:22) %We toggle N2 to find a minimum
szz=size(N2)
kk=zeros(i,szz(2));
for ii=(1:20)
   vll=0;
   for N2=(18:1:22)
      clc;
      ii
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl2');
      y11=y1(1,:);
      y12=y1(2,:);
      vll=[vll sum(y11.^2)+sum(y12.^2)];
      aa=size(vll);
      vll1=[vll(2:1:aa(2))];
   end
   kk(ii,:)=vll1;
end
mm=size(kk)
pp=mm(2) %number of colons
qq=mm(1) %number of rows

ll=sum(kk,1)/qq;
disp('ll is: ')
ll
disp('Once more kk')
kk
figure(2)
plot(ll, 'k')
title('tuning N2...')

%-------------------------------------------------------

disp('TUNING Nc? PRESS SPACE....') %TUNING Nc......
pause;

N1=1;
N2=20;
Nc=12;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
```

```
Nc=(10:1:14) %We toggle Nc to find a minimum
szzz=size(Nc)
kkk=zeros(i,szzz(2));
for iii=(1:20)
   vlll=0;
   for Nc=(10:1:14)
      clc;
      iii
      N1
      N2
      Nc

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'nl2');
      y11=y1(1,:);
      y12=y1(2,:);
      vlll=[vlll sum(y11.^2)+sum(y12.^2)];
      aaa=size(vlll);
      vlll1=[vlll(2:1:aaa(2))];
   end
   kkk(iii,:)=vlll1;
end
mmm=size(kkk)
ppp=mmm(2) %number of colons
qqq=mmm(1) %number of rows

lll=sum(kkk,1)/qqq;
disp('lll is: ')
lll
disp('Once more kkk')
kkk
figure(3)
plot(lll, 'k')
title('tuning Nc...')

%-----------------------------------------------------
% CHOOSING PARAMETERS FOR Wu AND Wy
%-----------------------------------------------------

f2ss;
size(Ai)
simlength=100;
N1=1;
N2=20;
Nc=11;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
y0=[1 -1]';
n=(1:1:5);

sz=size(n)
k=zeros(i,sz(2));

for i=(1:10)
   vl=0;
   for n=(1:1:5)
```

```
      Wy=[0.5+0.1*n 0; 0 0.5+0.1*n];
      clc;

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'lin');
      y11=y1(1,:);
      y12=y1(2,:);
      vl=[vl sum(y11.^2)+sum(y12.^2)];
      a=size(vl);
      vl1=[vl(2:1:a(2))];
   end
   k(i,:)=vl1;
end

   size k(a,:)
   size vl1



m=size(k)
p=m(2) %number of colons
q=m(1) %number of rows

l=sum(k,1)/i;
disp('l is: ')
l
disp('Once more k')
k
figure(4)
plot(l, 'b')
title('tuning Wy...')

%-----------
clear;
f2ss;
size(Ai)
simlength=100;
N1=1;
N2=20;
Nc=11;
Wy=eye(2);
Wu=eye(2);
dumax=[0.4;0.4];
y0=[1 -1]';
n=(1:1:5);

sz=size(n)
k=zeros(i,sz(2));

for i=(1:10)
   vl=0;
   for n=(1:1:5)
      Wu=[0.5+0.1*n 0; 0 0.5+0.1*n];
      clc;

   [y1,du1,xc1]=distcol(Ai,Ki,Li,Bi,Ci,Wy,Wu,N1,N2,Nc,simlength,y0,duma
x,'lin');
```

```
      y11=y1(1,:);
      y12=y1(2,:);
      vl=[vl sum(y11.^2)+sum(y12.^2)];
      a=size(vl);
      vl1=[vl(2:1:a(2))];
   end
   k(i,:)=vl1;
end

   size k(a,:)
   size vl1



m=size(k)
p=m(2) %number of colons
q=m(1) %number of rows

l=sum(k,1)/i;
disp('l is: ')
l
disp('Once more k')
k
figure(5)
plot(l, 'b')
title('tuning Wu...')

%%%%%  %%  %%  %%%%
%%     %%% %%  %% %%
%%%%   %%%%%%  %% %%
%%     %% %%%  %% %%
%%%%%  %%  %%  %%%%   %%
```