**NTNU**
**Norges teknisk-naturvitenskapelige universitet**

**Fakultet for kjemi og biologi**
**Institutt for kjemisk prosessteknologi**

# HOVUDOPPGÅVE 2000

| | |
|---|---|
| **Title: Adaptive model predictive control of chemical processes** | **Key words: Generalized Predictive Control, adaptive control, model development, RLS** |
| **Author: Elvira Marie Bergheim** | **Time period: Sept 1, 2000 – March 1, 2001** |
| **Supervisor: Prof. David E. Clough** | **Number of pages 128**<br><br>**Main report: 69**<br>**Appendix: 59** |

**ABSTRACT**

**One popular control strategy of model predictive control is General Predictive Control (GPC). Background and theory was studied for the GPC algorithm. The GPC algorithm was tested in both an experimental SISO case and a simulated MIMO case.**

**The experimental case was carried out on a shell-and-tube heat exchanger while the simulated case consisted two coupled distillations columns and was carried out in HYSYS.Plant.**

**ARX-model were used to describe the processes and they were developed from PRBS-data. The models were updated by an adapter. The adapter was based on recursive least square method.**

**The SISO GPC showed excellent control with relatively fast rise-time, almost no overshoot and smooth actions on the control valve. The GPC controller was compared to a PID controller and the GPC had much better performance than the PID.**

**The MIMO GPC showed poorer control and this could be caused by implementation error. The GPC was compared to a built-in MPC controller in HYSYS.Plant. The built-in MPC showed good control , much better than the MIMO GPC.**

**I declare that the work was carried out independent and in accordance to NTNUs regulations.**

**Date and signature:** .....

# Acknowledgement

I would like to thank prof. David E. Clough for all help with organizing my stay here in Boulder CO and supervision during the work with the thesis. In the lab Scott Whitehead and Dennis Burcham have been a great help. Bradley Dunkin has been a good help with Labview questions.

Elvira Marie

# Abstract

In recent years, model predictive control (MPC) has become more popular in industry. One control strategy of MPC is General Predictive Control (GPC). GPC computes control signals based on predicted outputs that are calculated from a model of the process. Background and theory for the GPC controller was studied and then a GPC controller was implemented and used in a single-input single-output (SISO) experimental case and in a multi-input multi-output (MIMO) simulation case.

The SISO experimental case was carried out using a heat exchanger interfaced to a computer. The main goal was to control the temperature in the cold outlet stream by the hot water flow rate. An ARX-model for the process was developed from pseudo-random-binary-signal (PRBS) test. The model was updated by an adapter based on the recursive least squares (RLS) method while running tests. The control structure was cascade where the GPC controller was the master controller that measured the temperature and calculated a set point for the slave controller. The slave controller controls directly the hot water flow rate.

The GPC controller was tested with several different cost and control horizons, weighting parameters and forgetting factors. The most important parameter to the performance was cost horizon. If the cost horizon was too long, the prediction of the output became poor because of uncertainty in the system. If the cost horizon is too short, the prediction did not include the dynamics of the process. A longer control horizon gave more active control but it might cause more fluctuations in the manipulated variable compared to a shorter control horizon. The weighting of the control signal needed to be quite conservative or else it produced oscillations in the process. If the weighting value was too conservative, the rise-time to the system became longer than necessary. The forgetting factor had no influence on the control performance and the control horizon had minor influence on the performance.

The GPC controller was then compared with a PID controller for the same process in a cascade. The PID controller was not as effective in smoothing out oscillations as the GPC controller. In a step test, the rise-time was shorter for the PID controller. However, the set point to the slave controller changed between fully open and fully closed valve which makes the output oscillate and caused unnecessary wear and tear on the valve. The control signal changes smoothly when using the GPC controller, and there were minimal oscillations.

The MIMO simulation case was arranged in HYSYS.Plant. The process included two distillation columns coupled through recycle streams, which separated natural gas liquid (NGL) into ethane and propane and a bottom stream that went on to further separation. The simulated process has local PID controllers and a multivariable GPC controller connected to the second column.

An ARX-model based on data from PRBS-tests is used in GPC. The GPC included also an adaptive part, which updated the model. The adapter is based on the RLS method. Tests were carried out by decreasing the ethane molar flow rate in the feed stream. Performance of the

GPC controller was evaluated for several different cost horizons, control horizons and control signal weighting matrices. The performance is particularly sensitive to cost horizon and control signal weighting.

The GPC controller was compared with the MPC controller that is built into HYSYS.Plant. The built-in MPC used a first order plus dead time (FOPDT) model to predict outputs. The controller was tested on the exactly same process with the same changes in ethane molar flow rate in the feed stream. Built-in MPC showed smooth control for several parameters.

The built-in MPC was preferred to the GPC because it showed good performance and was very simple to use. The built-in MPC was not as sensitive to parameter choices as the GPC controller. However, the GPC controller would probably be better if the process had more complex dynamics, which would be poorly approximated by FOPDT.

In both the SISO experimental case and the MIMO simulation case the cost horizon and control signal weight was found to be important to the performance of the GPC controller. The control horizon was of lesser importance, and the forgetting factor in the RLS made no difference in the GPC performance.

# Contents

# Contents

# 1. INTRODUCTION

Model predictive control (MPC) was introduced in the late seventies and has developed significant since then. The term MPC does not entitle a specific control strategy but including a range of control methods. Commons for the control methods are that a process model is used to predict output in future time to obtain control signals by minimizing an objective function. The different control methods only differ in the model used to represent the process and the noises and the cost function to be minimized.

One of the MPC control strategies is the Generalized Predictive Control (GPC) and was developed by Clarke, Mohtadi and Tuffs. Their main work on the method was presented in 1987. The method has been implemented in many industrial applications, showing good performance and a certain degree of robustness for a wide range of plants.

This thesis has studied GPC in both a SISO experimental case and a MIMO simulation case. They were treated separately in this thesis. The experimental part included most of the theory the thesis was based on, but the simulation part treats supplemental parts of the theory when extending the algorithm from a single variable case to a multivariable case.

The thesis was carried out at University of Colorado at Boulder in fall 2000 and spring 2001.

# 2. THEORY EXPERIMENTAL CASE

## 2.1 Pseudo-random-binary-signal tests

To develop a model for a process, pseudo-random-binary-signal (PRBS) can be used for developing data. The PRBS binary inputs, either $u_1$ or $u_2$, are selected randomly in chosen time interval $\Delta t$. The responses are observed and recorded. A typical PRBS input sequence can be as depicted in figure 2.1, from [9].

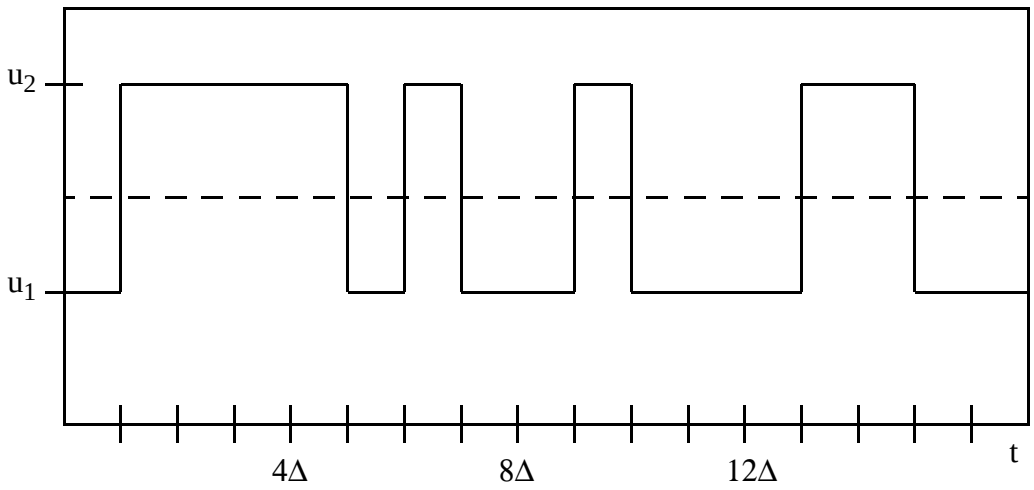



**Figure 2.1 Illustration of a typical PRBS input**

It is important to avoid missing dynamics information from the system, so the sampling interval has to be chosen properly. The sampling interval has to be long enough so the system has a chance to show responses, and not too short so missing dynamics is avoided. To detect deadtime and responstime, a steptest can be executed.
The difference between the binary input signals needs to be big enough to produce a significant difference in the output. The binary inputs should also be selected in the operating range and the inputs should not reach saturation.

## 2.2 Model development

All MPC methods need a model for the process to predict the output. It is important that the model gives a good description of the process to avoid poor prediction. An autoregressive model with input, also named ARX can be used. This model can be identified from PRBS-test data as described by Clough [6].
An ARX-model is on the form

$$A(q^{-1})y(t) = B(q^{-1})u(t-d) + e(t) \tag{2.1}$$

where *A* and *B* are the model parameters and they are functions of the backward shift operator $q^{-1}$. *y(t)* is the output, *u(t)* is the input, *d* is the delay in the model and *e(t)* is the noise. The *A* and *B* have the form:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na} \tag{2.2}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \ldots + b_{nb} q^{-nb} \tag{2.3}$$

where *na* and *nb* are the order of *A* and *B* respectively.

Aikake's Final Prediction Error (FPE) can be used as indicator to select the best model between the different model structures. FPE penalizes over-parameterization of the model and prefer less complex models. Fitting PRBS-data to an ARX-model can be done by for instance Matlab with the function *arx*.

## 2.3 The General Predictive Controller

One popular predictive control algorithm is the Generalized Predictive Control (GPC). The GPC method was proposed by Clarke *et al*. [3]. The basic idea of GPC is to calculate a sequence of future control signals in such way that it minimize a multistage cost function defined over a prediction horizon.

The theoretical deduce of the GPC controller is derive from Camacho and Bordons [1] and Clarke et. al. [3]. Most single-output single-input (SISO) plants can be described by a Controlled AutoRegressive and Integrated Moving Average (CARIMA) model when considering operation around a particular set-point and after linearization. The CARIMA model is given by

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t-1) + C(q^{-1})\left(\frac{e(t)}{\Delta}\right) \tag{2.4}$$

with $\Delta = 1 - q^{-1}$ and $C(q^{-1})$ is the noise polynomial and has the same form as $A(q^{-1})$ given in equation (2.2), and $q^{-1}$ is the backward shift operator. *y(t)* is the output, *u(t)* is the input, *d* is the dead-time and *e(t)* displays the noise. The noise is assumed as zero mean white noise. For simplicity, the *C* polynomial is chosen to be 1.
To derive a *j*-step predictor of *y(t+j)*, consider the Diophantine equation

$$1 = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F_j(q^{-1}) \tag{2.5}$$

where $E_j$ and $F_j$ are polynomials uniquely defined given $A(q^{-1})$ and the prediction interval *j*. The degrees of $E_j$ and $F_j$ polynomials is *j - 1* and *na* respectively.

If equation (2.4) is multiplied with $E_j \Delta q^j$ we get

$$E_j(q^{-1})A\Delta y(t+j) = E_j(q^{-1})B(q^{-1})\Delta u(t+j-d-1) + E_j(q^{-1})e(t+j) \quad (2.6)$$

Considering Diophantine equation in (2.5), equation (2.6) can be written as

$$(1 - q^{-j}F_j(q^{-1}))y(t+j) = E_j(q^{-1})B(q^{-1})\Delta u(t+j-d-1) + E_j(q^{-1})e(t+j) \quad (2.7)$$

which can be rewritten as

$$y(t+d) = F_j(q^{-1})y(t) + E_j(q^{-1})B(q^{-1})\Delta u(t+j-d-1) + E_j(q^{-1})e(t+j) \quad (2.8)$$

As the degree of polynomial $E_j(q^{-1})$ is $j-1$ the noise terms in equation (2.8) are all in the future. The optimal prediction of $y(t+j)$ is therefore,

$$\hat{y}(t+j \mid t) = G_j(q^{-1})\Delta u(t+j-d-1) + F_j(q^{-1})y(t) \quad (2.9)$$

where $G_j(q^{-1}) = E_jB$.

In GPC a whole set of predictions is considered, for which $j$ runs from a minimum up to a large value, termed as the minimum and maximum prediction horizons. A system with a dead time $d$ will the prediction process $\hat{y}(t+j \mid t)$ for $j< d$ only depends on the available data, but for $j \geq d$ assumptions need to be made about future control actions.

### *2.3.1 Recursion of the Diophantine equation*

To solve the prediction in equation (2.9), the polynomials $E_j$ and $F_j$ are needed in addition to the model parameters $A$ and $B$. The polynomials $E_j$ and $F_j$ can be obtained recursively from the Diophantine equation, given in equation (2.5).

Suppose for clarity of notation $E = E_j$, $R = E_{j+1}$, $F = F_j$, $S = F_{j+1}$ and $\tilde{A}$ defined as $A\Delta$. The Diophantine equation in (2.10) corresponds to the prediction for $\hat{y}(t+j \mid t)$ and equation (2.11) corresponds to the prediction for $\hat{y}(t+j-1 \mid t)$.

$$1 = E\tilde{A} + q^{-j}F \quad (2.10)$$

$$1 = R\tilde{A} + q^{-(j+1)}S \quad (2.11)$$

Subtracting equation (2.10) from equation (2.11) gives

$$0 = \tilde{A}(R-E) + q^{-j}(q^{-1}S - F) \quad (2.12)$$

The polynomial $R$ - $E$ is of degree $j$ and may be split into to parts

$$R - E = \tilde{R} + r_j q^{-j} \tag{2.13}$$

where $\tilde{R}$ is a polynomial of degree smaller and equal to *j-1* and $r_j$ is a scalar.
The equation (2.12) then becomes

$$\tilde{A}\tilde{R} + q^{-j}(q^{-1}S - F + \tilde{A}r_j) = 0 \tag{2.14}$$

Since $\tilde{A}$ is monic, that is all coefficients are unequal to zero and first coefficient equals 1, $\tilde{R}$ needs to be equal to zero. This indicates that $S = q(F - \tilde{A}r_j)$. As $\tilde{A}$ as ha unit leading element, we have

$$r_j = f_0 \tag{2.15}$$

$$s_i = f_{i+1} - \tilde{a}_{i+1} r_j \tag{2.16}$$

for *i = 0* to the degree of $S(q^{-1})$ and

$$R(q^{-1}) = E(q^{-1}) + q^{-j} r_j \tag{2.17}$$

$$G_{j+1} = B(q^{-1})R(q^{-1}) \tag{2.18}$$

Given the plant polynomials $A(q^{-1})$ and $B(q^{-1})$ and one solution of $E_j(q^{-1})$ and $F_j(q^{-1})$, then equation (2.15) and equation (2.16) can be used to obtain $F_{j+1}(q^{-1})$. Vector $E_{j+1}(q^{-1})$ can be obtained by equation (2.17).

To initialize the iterations, the Diophantine equation can easily be solved for *j = 1*, that is $1 = E_1\tilde{A} + q^{-1}F_1$. Since the leading element of $\tilde{A}$ is 1, then is $E_1 = 1$ and $F_1 = q(1-\tilde{A})$.

### *2.3.2 The Predictive Control Law*

The GPC algorithm consists of applying a control sequence that minimizes a multistage cost function of the form

$$J(N_1, N_2, N_u) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \tag{2.19}$$

where $\hat{y}(t+j|t)$ is an optimum *j*-step ahead prediction of the system output on data up to time *t*, $N_1$ and $N_2$ are the minimum and maximum costing horizons, $N_u$ is the control horizon, $\delta(j)$ and $\lambda(j)$ are weighting sequences and $w(t+j)$ is the future reference trajectory. The cost horizon *N* can be expressed as

$$N = N_2 - N_1 + 1. \tag{2.20}$$

and represents how far into the future the predictions are made when calculation the controller output. The control horizon *NU* represents the number of controller moves into the future that will be made to achieve the final set point.

The set of *j* ahead optimal predictions is given by equation (2.9) and can be expressed as

$$\begin{aligned} \hat{y}(t+d+1\,|\,t) &= G_{d+1}\Delta u(t) + F_{d+1}y(t) \\ \hat{y}(t+d+2\,|\,t) &= G_{d+2}\Delta u(t+1) + F_{d+2}y(t) \\ &\cdots \\ \hat{y}(t+d+N\,|\,t) &= G_{d+N}\Delta u(t+N-1) + F_{d+N}y(t) \end{aligned} \tag{2.21}$$

which can be written in matrix form as

$$y = Gu + F(q^{-1})y(t) + G'(q^{-1})\Delta u(t-1) \tag{2.22}$$

where

$$y = \begin{bmatrix} \hat{y}(t+d+1\,|\,t) \\ \hat{y}(t+d+2\,|\,t) \\ \cdots \\ \hat{y}(t+d+N\,|\,t) \end{bmatrix} \qquad u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \cdots \\ \Delta u(t+N-1) \end{bmatrix}$$

$$G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ g_{N-1} & g_{N-2} & \cdots & g_0 \end{bmatrix} \qquad F(q^{-1}) = \begin{bmatrix} F_{d+1}(q^{-1}) \\ F_{d+2}(q^{-1}) \\ \cdots \\ F_{d+N}(q^{-1}) \end{bmatrix}$$

$$G'(q^{-1}) = \begin{bmatrix} (G_{d+1}(q^{-1}) - g_0)q \\ (G_{d+2}(q^{-1}) - g_0 - g_1 q^{-1})q^2 \\ \cdots \\ (G_{d+N}(q^{-1}) - g_0 - g_1 q^{-1} - \cdots - g_{N-1}q^{-(N-1)})q^N \end{bmatrix}$$

The last two terms in equation (2.22) depend only on the past outputs and past inputs and corresponds to the free response, *f*, leading to

$$y = Gu + f \tag{2.23}$$

where the vectors *y, u* and *f* have dimension *N* x 1.

If equation (2.23) is inserted in the cost function in equation (2.19), the cost function can be written as

$$J = (Gu + f - w)^T(Gu + f - w) + \lambda u^T u \tag{2.24}$$

where $w = \begin{bmatrix} w(t+d+1) & w(t+d+2) & \dots & w(t+d+N) \end{bmatrix}^T$, $\lambda(j)$ is assumed constant and $\delta(j)$ is assumed as one.
Multiplying out equation (2.24) gives:

$$J = \frac{1}{2}u^T H u + b^T u + f_0 \tag{2.25}$$

where

$$\begin{aligned} H &= 2(G^T G + \lambda I) \\ b^T &= 2(f - w)^T G \\ f_0 &= (f - w)^T(f - w) \end{aligned} \tag{2.26}$$

By making the gradient of $J$ with respect to $u$ equal to zero, the minimum of $J$ can be found, assuming there are no constraints on the control signal. This gives

$$u = -H^{-1}b = (G^T G + \lambda I)^{-1} G^T (w - f) \tag{2.27}$$

It is only the first element of vector $u$ that is actually sent to the process, so the control law is given by

$$u(t) = u(t-1) + \dot{g}^T(w - f) \tag{2.28}$$

where $\dot{g}^T$ is the first row of $(G^T G + \lambda I)^{-1} G^T$. This control law can be implemented in Matlab.

### *2.3.3 Cost function with constrains on control signal*

If there are constraints on the control signals, the control law given in equation (2.28) needs to be written as a function of $u$ instead of $\Delta u$. The process output can be written as in equation . The vector of future control increments is given by

$$\begin{bmatrix} u(k) - u(k-1) \\ u(k+1) - u(k) \\ \dots \\ u(k+N) - u(k+N-1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} u(k) \\ u(k-1) \\ \dots \\ u(k+N) \end{bmatrix} - \begin{bmatrix} u(k-1) \\ 0 \\ \dots \\ 0 \end{bmatrix} \tag{2.29}$$

which can be written as

$$u = DU - f_1 \tag{2.30}$$

The equations for the future control signal is substituted in the equation of future predictions, equation , and we get

$$y = G(DU - f_1) + f = G'U + f_2 \tag{2.31}$$

where $G'$ is a lower triangular matrix and $f_2$ can be expressed as

$$(f_2)_i = (f)_i + g_i u(k-1) \tag{2.32}$$

The cost function can now be expressed as a function of the future control actions $U$,

$$\begin{aligned} J(U) &= (G'U + f_2 - w)^T(G'U + f_2 - w) + \lambda(DU - f_1)^T(DU - f_1) \\ &= U^T(G'^T G' + \lambda D^T D)U + 2[(f_2 - w)^T G' - \lambda f_1 D]U + (f_2 - w)^T(f_2 - w) + \lambda f_1{}^T f_1 \end{aligned} \tag{2.33}$$

Written $J(U)$ in a quadratic form,

$$J(U) = \frac{1}{2}U^T H'U + b'U + f' \tag{2.34}$$

where

$$\begin{aligned} H' &= 2(G'^T G' + \lambda D^T D) \\ b' &= 2[(f_2 - w)^T G' - \lambda f_1 D] \\ f' &= (f_2 - w)^T(f_2 - w) + \lambda f_1{}^T f_1 \end{aligned} \tag{2.35}$$

The cost function is now a function of future control actions and constrains can easily be added on the control signals. The control signal can be found by minimizing the cost function in equation (2.34) with the constrains added to the minimizing problem. The minimizing problem can be solved in for instance Matlab with the function *fmincon* or solved as a QP-problem with function *quadprog*.

## 2.4 Cascade control

Cascade control can be useful when disturbances are associated with the manipulated variable or the final control element exhibit nonlinear behavior. Nonlinear behavior can for instance be significant hysteresis in a valve. The output signal of the master controller serves as the set point for the slave controller. The secondary loop is a faster loop that is located inside the primary control loop.

Dependent on the control problem, the slave controller can be a PI controller in velocity

mode. The theoretical deduce of PI controller in velocity mode is derived from Seborg *et al*. [12]. The velocity form of the PI controller does not compute summations which is needed in position mode and therefore windup is avoided. The velocity mode does not require specification of the bias term.

The design equation for a PI controller in velocity mode can be found from the general position mode for a PI controller, given in equation (2.36),

$$u(t) = \pm K_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(t')dt'\right) + b \tag{2.36}$$

where $u$ is the control signal, $K_c$ is the gain, $e$ represent the error between the measurement and the set point, $\tau_I$ is the integral time and $b$ is the bias.
The velocity mode is found by taking the time derivative of equation (2.36).

$$\frac{du(t)}{dt} = \pm K_c\left(\frac{de(t)}{dt} + \frac{e}{\tau_I}\right) \tag{2.37}$$

With approximations for the derivative, this becomes

$$\frac{\Delta u}{\Delta t} = \pm K_c\left(\frac{\Delta e}{\Delta t} + \frac{e}{\tau_I}\right) \tag{2.38}$$

In the discrete case this can be expressed as

$$u_i = u_{n-1} \pm K_c\left(\left[1 + \frac{\Delta t}{\tau_I}\right]e_n - e_{n-1}\right) \tag{2.39}$$

Equation (2.39) can be implemented directly in virtual instrument, for instance Labview.

## 2.5 The Recursive Least Squares method

The deduce of the recursive least square method is based on a handout from Clough [5]. An adapter tasks is to adjust the controller parameters automatically to compensate for changing process condition. This can be very useful when the process conditions changes in time. A illustration of single-input single-output process is displayed in figure 2.2.

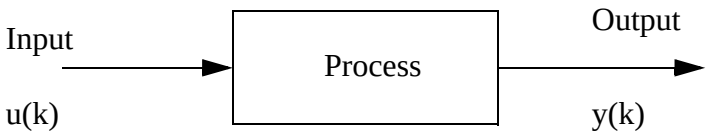



**Figure 2.2 Illustration of a single-input single-output process.**

The adaptive controller use recursive least squares (RLS) method to calculate the *A* and *B*

polynomial in the model. The *A* and the *B* polynomial is as given in equation (2.2) and equation (2.3). The degree of the *A* polynomial is *na* and the degree of the *B* polynomial is *nb*. Define the parameter column vector as

$$\beta = \begin{bmatrix} a_1 & \ldots\ldots & a_{na} & b_0 & \ldots & b_{nb} \end{bmatrix}^T \tag{2.40}$$

The output vector is defined as

$$y = \begin{bmatrix} y(1 + na + d) \\ \ldots \\ y(N + na + d) \end{bmatrix} \tag{2.41}$$

where *N* is the cost horizon and *d* is the dead-time. The matrix $\varphi$ is defined as

$$\varphi = \begin{bmatrix} -y(na + d) & \ldots & -y(1 + d) & u(nb) & \ldots & u(1) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ -y(na + d + N - 1) & \ldots & -y(N + d) & u(nb + N - 1) & \ldots & u(N) \end{bmatrix} \tag{2.42}$$

The error vector is defined as

$$e = \begin{bmatrix} e(1 + na + d) \\ \ldots \\ e(N + na + d) \end{bmatrix} \tag{2.43}$$

The evaluation of the model for the data set can be represented by

$$e = y - \varphi\beta \tag{2.44}$$

The loss function can be written as the sum of the squares of the residuals

$$V = e^T e \tag{2.45}$$

To minimize the loss function can be done by setting

$$\frac{\partial V}{\partial \beta_i} = 0 \text{ for i = 1 to } na + nb \tag{2.46}$$

The result from minimizing the loss function is

$$\hat{\beta} = [\varphi^T \varphi]^{-1} \varphi^T y \tag{2.47}$$

The derivation of the minimization is given in [5]. The RLS algorithm can be described as in figure 2.3. The same algorithm can be used in Matlab implementation.

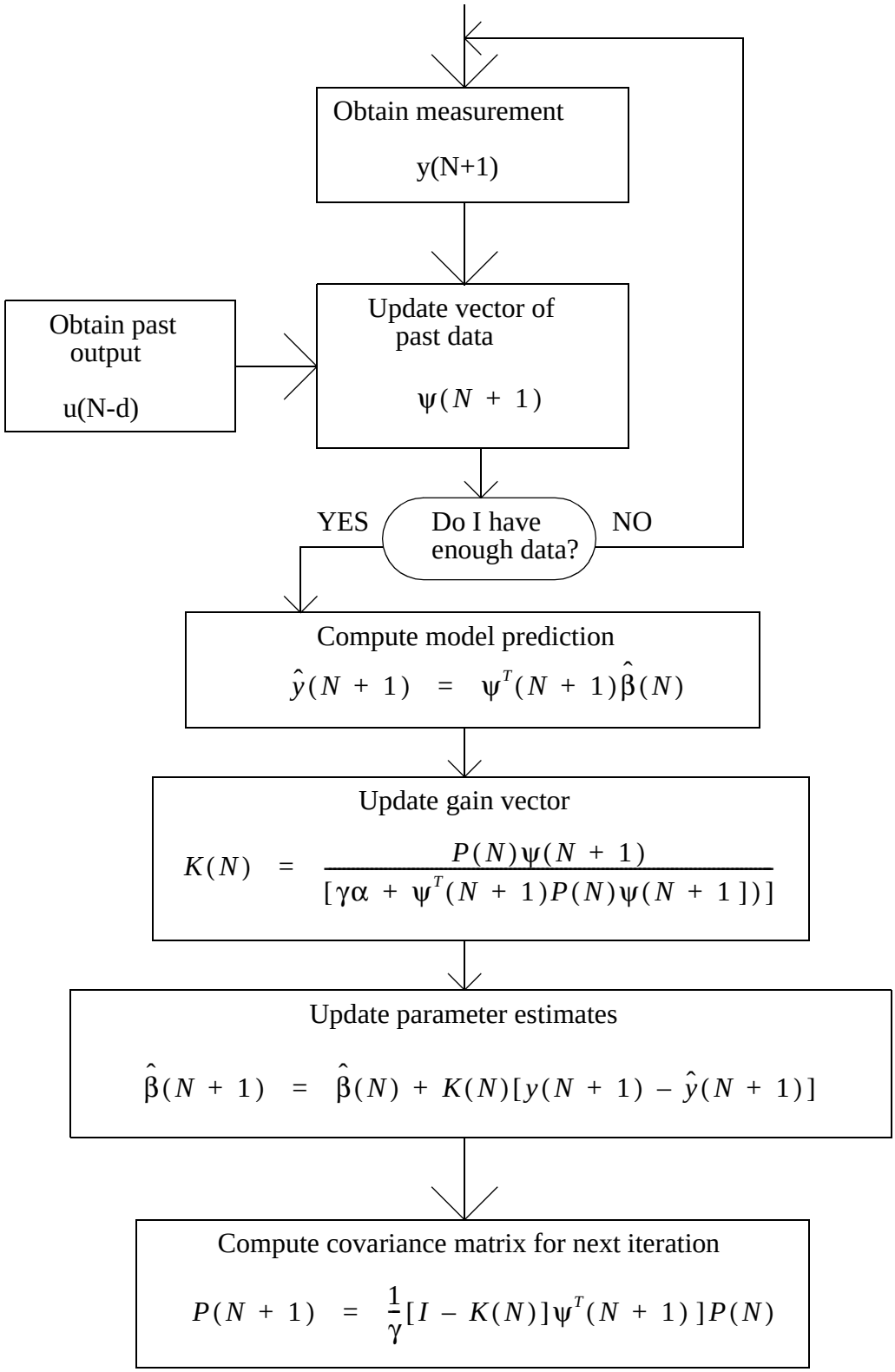



**Figure 2.3 The RLS Parameter Estimation Algorithm**

In figure 2.3 $\psi$ is a vector that contains the past values of measurement and control signals with the size (m+n) x 1 and has the form

$$\psi(N+1) = \begin{bmatrix} -y(N) \\ \cdots \\ -y(N-na+1) \\ u(N-d) \\ \cdots \\ u(N-d-nb+1) \end{bmatrix} \tag{2.48}$$

The scalar $\hat{y}$ is the prediction of the measurement based on the model. *K* is the gain vector and $\gamma$ is the forgetting factor. The forgetting factor weight out or "forget" gradually the past values. The forgetting factor has a value between 0 and 1, and the closer $\gamma$ is to 1, the slower the algorithm "forgets" the past values. In other words, it becomes more conservative. $\beta$ is a parameter vector as described in equation (2.40). *P* is the covariance matrix and is defined by

$$P(N) = [\varphi^T(N)\varphi(N)]^{-1}\alpha \tag{2.49}$$

where

$$\alpha = var\{e(k)\} \tag{2.50}$$

The parameter $\alpha$ indicates the error the model has not manage to include compare to the process. A value of $\alpha$ can be found in the result from fitting ARX-model based on PRBS-data. Matrix *I* is the identity matrix which has the same dimension as *K*. The question "Do I have enough data?" is a question if $\psi$ is filled up with data from the process or not.

To initialize the algorithm, values for $\hat{\beta}$, *P* and $\alpha$ is needed. The initialization can done by setting

$$\begin{aligned} \hat{\beta}(0) &= 0 \\ P(0) &= cI \end{aligned} \tag{2.51}$$

where *c* is a large number. The vector $\hat{\beta}$ can also be initialized by using the *A* and *B* parameters in the ARX-model.

To update the model parameters, RLS needs changing inputs to the algorithm. This is usually not a problem in real world but with small, separated systems with no turbulent flow or simulations with no disturbances, this needs to be considered. With too little noise in the system the matrixes in the RLS can become singular and weird values for the *A* and the *B* parameters can occur which again gives poorly prediction of the outputs.

## 2.6 PID Controller in discrete time

The theoretical deduce of PID digital controller is derive from Seborg *et al*. [12].
Design equation for digital PID controller is based on the ideal, continuous (analog) PID controller, which is described as

$$u(t) = \pm K_c \left[ e(t) + \frac{1}{\tau_I} \int_0^t e(t')dt' + \tau_D \frac{de(t)}{dt} \right] + b \tag{2.52}$$

where *u(t)* is the control signal, $K_c$ is the gain, *e(t)* represent the error between the measurement and the set point, $\tau_I$ is the integral time, $\tau_D$ is the derivative time and *b* is the bias. To convert this expression to its digital equivalent, the following finite difference approximations are used

$$\int_0^t e(t')dt' \approx \sum_{k=1}^{n} e_k \Delta t \tag{2.53}$$

$$\frac{de}{dt} \approx \frac{e_n - e_{n-1}}{\Delta t}$$

When the approximations in equation (2.53) are introduced in the ideal analog PID, the position form of the digital PID controller can be written as

$$u(t) = \pm K_c \left[ e_n + \frac{\Delta t}{\tau_I} \sum_{k=1}^{n} e_k + \frac{\tau_D}{\Delta t}(e_n - e_{n-1}) \right] \tag{2.54}$$

The position form of the PID controller can now be implemented as the control low because it yields the valve of the controller output directly. The controller can be implemented in for instance Labview.

# 3. EXPERIMENTAL PHASE - SINGLEVARIABLE

The experimental part of this paper tested the general predictive control (GPC) algorithm with a recursively least square (RLS) adapter on a single-input single-output (SISO) process.

## 3.1 Process description of heat exchanger and additional equipment

The heat exchanger was an already existing unit interfaced to a converter and a computer in the Chemical Engineering Laboratory at University of Colorado. The heat exchanger is illustrated in figure 3.1.

The heat exchanger was a shell-and tube type. The cold water flowed inside the tubes and the hot water flowed outside the tubes. It had four termocouples that were placed at each inlets and outlets. The heat exchanger had also a valve on each inlet stream that was controlled by air pressure. The valve on the hot water inlet had also a valve positioner. In addition there were safety valves on each stream.

The computer used Labview software from National Instruments Corp., a graphical programming language to create block diagram structures. Program modules in Labview are called virtual instruments (VI's).

The heat exchanger was connected to a computer by analog-to-digital converter and digital-to-analog converter. The interface between the heat exchanger and the computer is present in figure 3.2.

The termocouples gave signals in mV and needed to be converted to $^{o}$C. The voltage signals were sent to the computer and displayed in Labview's virtual instrument (VI).

The flow was measured in differential pressure ($\Delta$P). The differential pressure was converted to current in the pressure-to-current (P/I) converter. This signal was again converted to voltage in the current-to-voltage (I/V) converter before it reached the computer. The flow signals from the computer followed the opposite direction and were converted to pressure, which made the valves move. This is displayed in figure 3.3 including the range at each conversion.

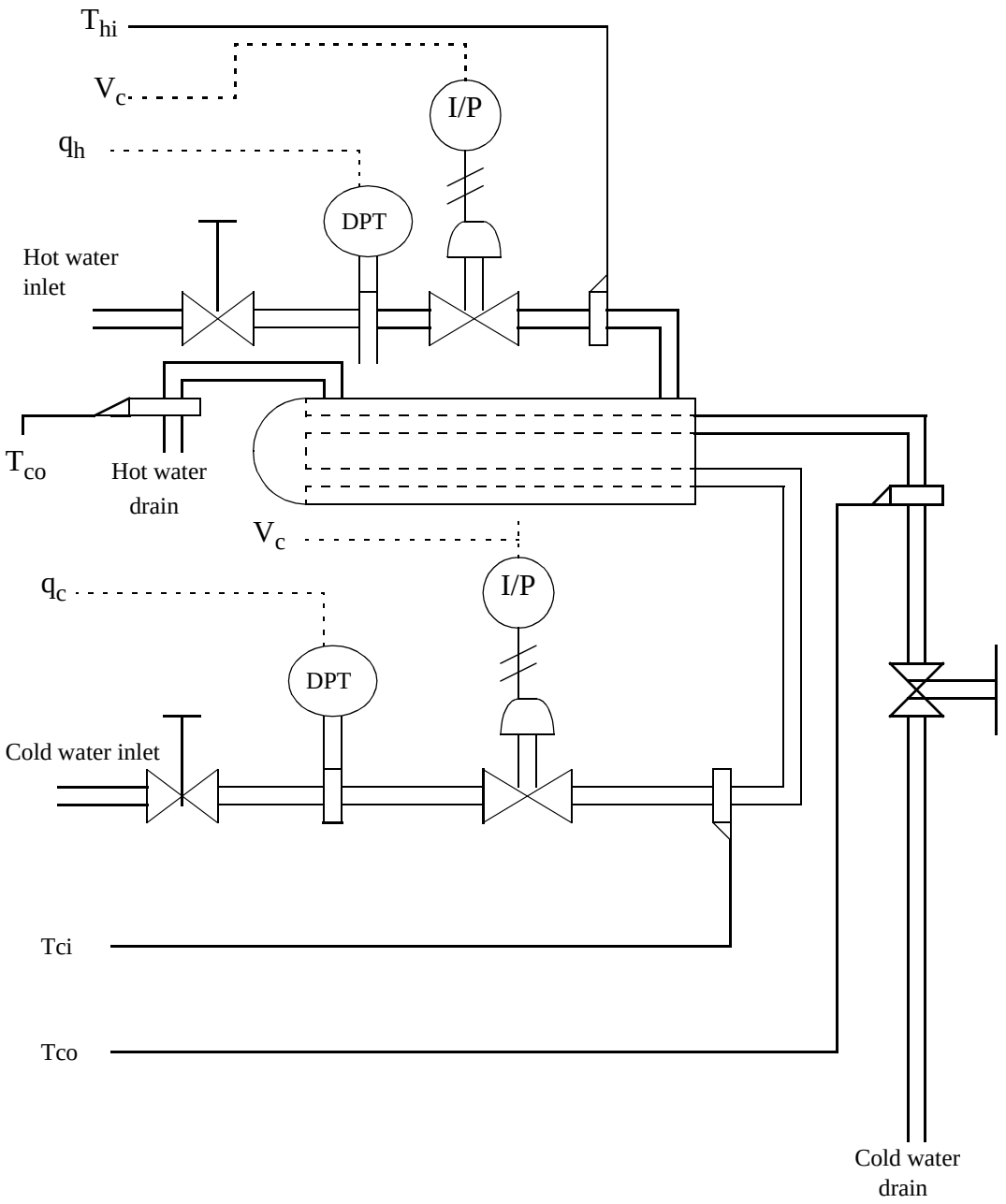



**Figure 3.1 Diagram of the heat exchanger**

.

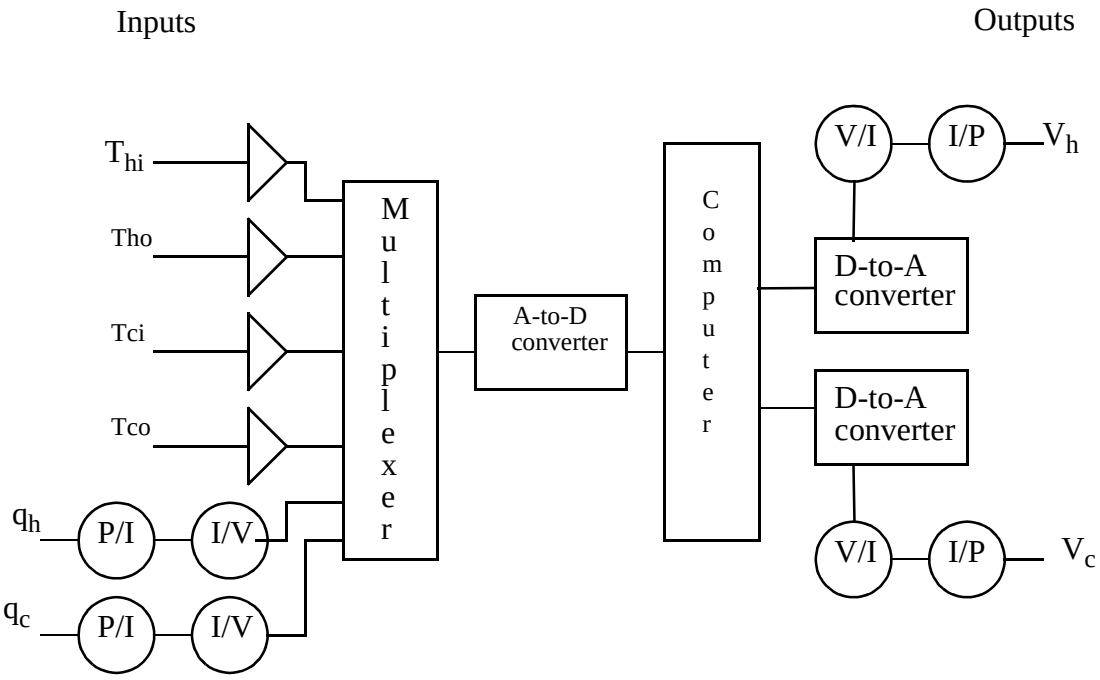



**Figure 3.2 Diagram of the interface between the computer and the heat exchanger**

.

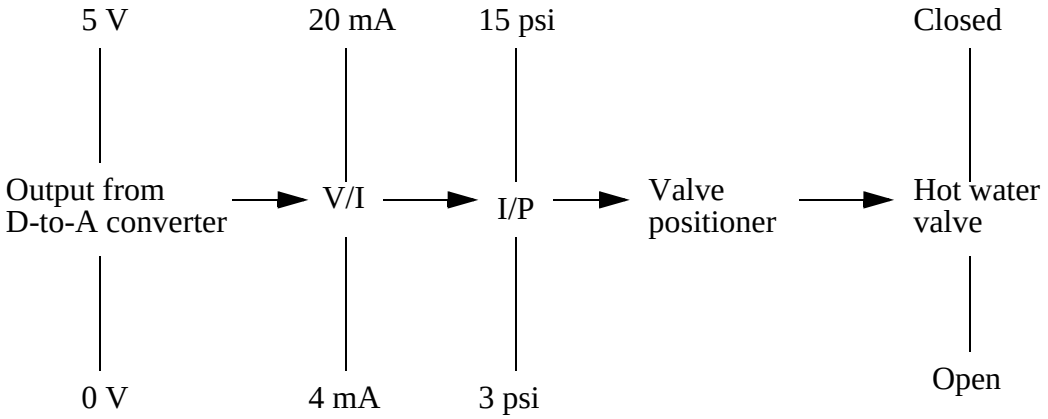



**Figure 3.3 Signal processing between the computer and the hot water valve at the heat exchanger**

## 3.2 Calibration

The temperature- and flow measurements needed to be calibrated before running tests. First the process was implemented in Labview and the signals were connected to the right channel in the D-to-A converter. The Labview diagram was based on an existing program

that is used in a lab course at University of Colorado. Prof. D.E. Clough at University of Colorado made this program. The Labview diagram is displayed in appendix A.

The flow rate was measured by collecting the flow in a bucket over a certain time interval, while the voltage displayed in Labview's VI was noted. The collected water were weighted, and with the assumption that 1 kg equals 1 liter; the flow rate was expressed in l/min. This were done to both the hot water and the cold water flow.

When the flow was measured, the current was measured in the current-to-voltage (I-to-V) converter with a multimeter. The zero and the span in the I-to-V converter were adjusted to zero flow and difference between zero and maximum flow respectively. This were done both with the hot water and the cold water flow.

The flow measurements were plotted against the voltage in an Excel worksheet and the relation between the flow and the voltage were found. The theory background for flow and voltage relation is displayed in appendix B. When the voltage dropped below a certain threshold value it became necessary to set the flow equal to zero to avoid numerical problems, due to the square root in the relation between flow and voltage. Zero flow was observed for the hot water when the voltage drops below the threshold value. The cold water flow was about 2 l/min when the voltage dropped below the threshold value. The cold water valve never managed to close the valve totally. The relation between flow rate and volt was implemented in Labview.

During the calibration tests, hysteresis was discovered on both the hot water valve and the cold water valve. The hysteresis was less on the hot water valve than the cold water valve. This was expected due to the valve positioner placed on the hot water valve. The relationship between the flow and the square root of the voltage became poorly at high voltage, so this area was avoided in the further tests.

For the termocouples there existed already a relation between temperature in $^{o}$C and voltage in mV from earlier experiments on the heat exchanger. The termocouples were checked against standard thermometers in ice and boiling water respectively. The temperatures from the termocouples were plotted against the temperatures from the standard thermometers and relations were found. The relation between mV and $^{o}$C was implemented in Labview.

## 3.3 Model development

The GPC algorithm needed a mathematical model that describes the process, as explained in chapter 2.3. PRBS-tests was chosen to generate process data and used to develop an ARX-model.

A VI in Labview was made to generate the PRBS-tests, based on the Labview program made for calibration. The diagram of the VI is displayed in appendix C. PRBS-test was carried out

with binary inputs from four different ranges. Both low and high values for both hot and cold water were used as binary input. The binary input-values for the hot water valve settings were found by try and fail. Input to PRBS was a gauge-value from Labview's VI and was in the range 0 to 5. To find a suitable sampling interval, a step test was executed to find the dead time to the system. The dead time was 3-6 seconds, depending on the flow rate. The sampling interval in the PRBS-test was chosen to be 3 seconds. Each PRBS-tests ran for about 10 minutes. The test data was recorded in a text file and used for model development.

Four models were developed, one for each range the PRBS-test were done. ARX-models were developed by Matlab using data from the PRBS-tests. The Matlab algorithm used for developing models is displayed in appendix F.1. An example for how the PRBS-test where carried out, is shown in appendix D.

## 3.4 Implementation of the GPC Controller

A VI in Labview was implemented to serve GPC. The VI was based on the VI from PRBS-tests, and the diagram is displayed in appendix E.

The implementation of the GPC controller could be divided into two parts. The Labview part got the signals from the heat exchanger in a certain time interval and sent signals to the hot water valve to control the cold water outlet temperature. The Matlab part received information about the past outputs and inputs from Labview. Based on the information from Labview, the GPC algorithm calculated the future control actions. The first control action was received in Labview, which sent the signal to the hot water valve.

First, the control law given in equation (2.28) was implemented in Matlab. In the control law the control signal was not limited in the algorithm. The control signal became limited when it was sent to the heat exchanger. Test with the GPC without limited control signal showed poorly control. The control gave offset and oscillations with periods about 30 seconds. This may be caused from the unlimited control signal sent to the process.

Therefore a new cost function with constrains, as presented in chapter 2.3.3, was implemented in Matlab. The GPC algorithm implemented in Matlab is presented in appendix F.2, with additional functions in appendix F.3 and appendix F.4.

Tests showed oscillations in the system even though the control signals were limited. The oscillations had two periods, one faster with periods about 12 seconds and the other slower with periods about 120 seconds. The amplitudes were approximately 0.7$^{o}$C and 2$^{o}$C respectively. This pattern did not occur when the GPC with constraints was connected to the model instead of the process. When the GPC "controlled" the model, the control was perfect. From that observation, the equipment was the cause to the oscillations. The oscillations occurred probably because of the hysteresis in the hot water valve.

To reduce hysteresis, a cascade was implemented with a flow controller as slave and the GPC as the master controller. The cascade was set up as illustrated in figure 3.4.

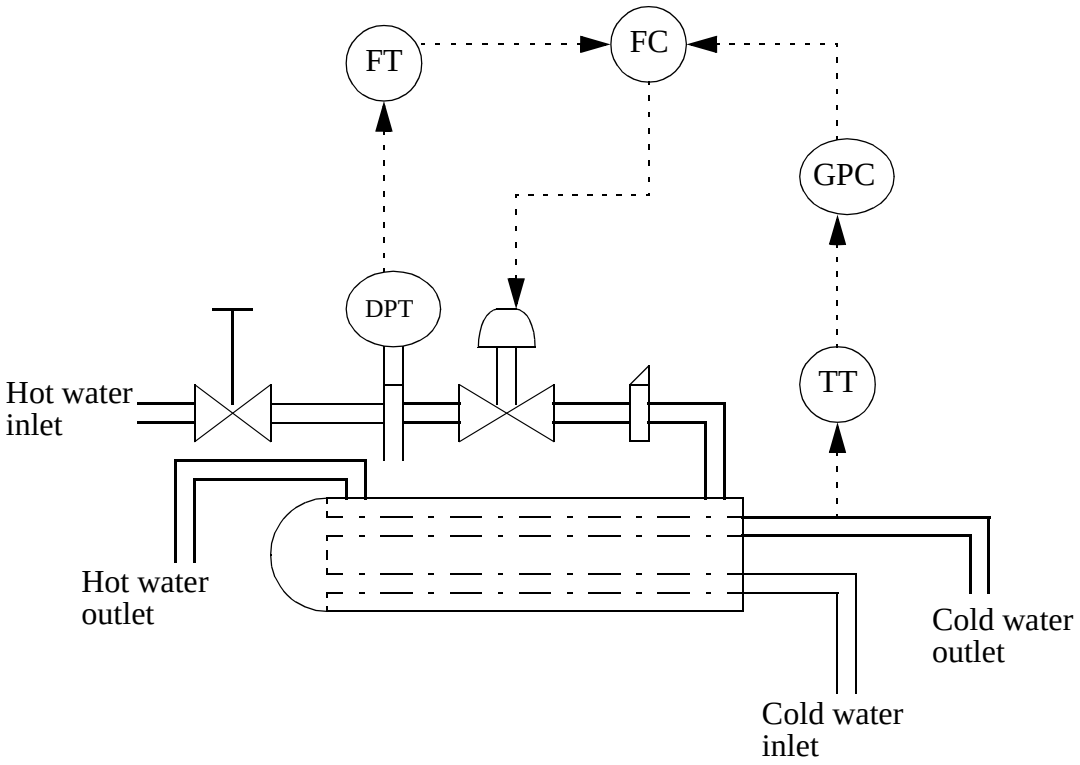



**Figure 3.4 Illustration of the heat exchanger with cascade control**

For the slave controller, a PI controller in velocity mode was used as described in chapter 2.4. The slave controller was implemented in Labview, by using equation (2.39). Step tests in different operating areas were performed in manual mode to find values for the parameters $K_c$ and $\tau_I$. The tuning parameters were found by Skogestad's tuningsrules [13]. Mean values of the tuning parameters are found by average the different parameters from the different step tests. The sampling time to the slave controller was set to 0.25 seconds. With a gain $K_c = 0.06$ and $\tau_I = 1.0$, the slave controller worked properly.

When the slave controller was implemented in Labview, the process from the GPC point of view changed. GPC algorithm gave now a set point to the slave controller instead of a gauge-value to the hot water valve. In other words, the process now contained a slave controller, and a new PRBS-test was performed to obtain a new model for the GPC. Only one PRBS-test was performed in the middle flow area, because the model developed from the PRBS-test supposed later to be updated by an adapter. More detailed description of the PRBS-test is present in appendix D.

One problem with the Labview program implemented for the cascade control was the timing. The GPC controller and the slave controller did not operate independent in time. So when

the Labview program ran, it let the GPC algorithm work first and find the optimal control signal. This signal was sent as a set point to the slave controller. After GPC finished its calculations, then the slave controller started to correct the valve opening due to set point. So the slave controller actually waited for the GPC controller to finish its job.

## 3.5 Implementation of the adaptive controller

The adaptive controller job was to update the model, which the GPC controller used for prediction. The model might be changed with different flows and temperatures. The adaptive part was based on recursive least squares and implemented as a function in Matlab. The Matlab algorithm is shown in appendix F.4.

To check that the adapter worked properly, the adapter was tested before connected to the GPC. Tests were done by sending PRBS-signals with different average set points to the adapter. The model parameters were observed and model changes could be detected.

When the adapter was connected to GPC, it received the values for the predicted inputs and the past outputs through the GPC algorithm. The adapter was actually not directly linked to Labview but only to the GPC algorithm. With recursive least squares method, the adapter calculate the coefficients to the *A* and *B* polynomial in the model and gave the new values for *A* and *B* back to the GPC. The data flow is displayed in figure .

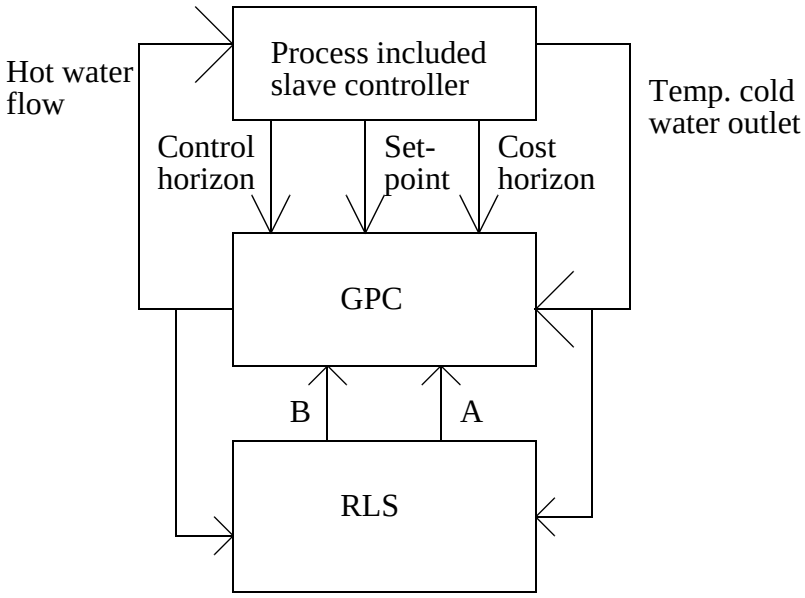



**Figure 3.5 The interface between the process, GPC and RLS including the data flow**

## 3.6 Implementation of the PID controller

The GPC controller was replaced with a PID controller for comparison. A PID controller in position mode was chosen. The Labview program for the PID controller was based on an existing program from Prof. D.E. Clough at University of Colorado at Boulder. The Labview diagram is displayed in appendix H.

A step test was carried out to collect data from the process. The sampling time were set to 1.5 second, which was the fastest sampling time possible due to the CPU time. The step test data were treated in control software called Control Station to obtain values for the process gain ($K_c$), integral time ($\tau_I$) and derivative time ($\tau_D$). The parameters were tested and adjusted. Skogestad's tuning rules [13] was also applied on the same step test data to develop tuning parameters for the PID controller. The tuning parameters developed by Control Station and the parameters developed with Skogestad's tuning rules were very similar. This was expected when both methods are based on Internal Model Control (IMC).

Since the sampling time was shorter when using the PID controller compare when using the GPC, the conditions for the slave controller changes. The slave controller was implemented in the same way as when the GPC was the master controller. Because of the implementation where the master controller and slave controller were in the same loop, the number of times the for-loop for the slave controller was executed was decided of computation time of master controller and sampling time. The sampling time was shorter for the PID controller, but the computation time was significantly shorter compared to the GPC controller. The number of times the for-loop for the slave controller was executed was increased from 5 to 11.

## 3.7 Testing the controllers

When testing both GPC controller and PID controller, the test were all tried to be as equal as possible. The same changes in set point and disturbances were therefore done in all tests. First, during start up, the set point for the cold water outlet temperature was set to 28 $^oC$. When the output settled down as much as it seemed possible, the set point was increased to 33 $^oC$. After the output settled down after the set point change, the set point was decreased to 27 $^oC$. During this set point changes the cold water flow was around 5 l/min. While the set point was 27 $^oC$, the cold water flow, considered as a disturbance, was increased to 15 l/min. Then the cold water flow was decreased to 6 l/min before the test finished.

# 4. RESULTS FROM EXPERIMENTAL PHASE

## 4.1 Results using GPC controller

To test the GPC controller several tests were carried out with changing parameters in the GPC algorithm. In the GPC algorithm the parameters cost horizon (*N*), control horizon (*NU*) and the control-weighting λ needed to be decided. In addition the forgetting factor in the adapter needed to be set. The sampling interval was 4 seconds, because this was the fastest time the computer could work through the virtual instrument and the functions in Matlab.

The parameters in the GPC algorithm were initially chosen based on suggestions from Clarke *et al*. [3] and Clarke *et al*. [5] and then other parameters where tried out, based on the performance in the earlier test that where carried out.

The minimum costing horizon value was based on that $N_1$ could be chosen as the dead time in the process or more to minimize computations, from Clarke *et al*. [3]. Since the dead time was observed to be longer than one sampling interval at lower flows, $N_1$ was chosen to be 2 in all tests.

The maximum costing horizon value should be chosen so it encompassed the response, which was significantly affected by the current control. $N_2$ should be at least greater than the degree of $B(q^{-1})$, or more typically $N_2$ is set to approximate the rise-time of the plant, from Clarke *et al*. [3]. The degree of the *B* polynomial was seven, from the ARX-model developed. The rise time for the process was about 30 seconds, which is about 8 samples with a sampling interval of 4 seconds. Several values for $N_2$ was tested among the recommendations.

The control horizon *NU* equals one is generally satisfactory for a process plant, while a "difficult" plant requires that the control horizon is about the same as number of unstable or underdamped poles, from Clarke *et al*. [5]. Several cost horizons values were tested from the recommendations.

The control weighting factor, λ, can be selected as zero or λ can be selected as a small number because it helps numerical robustness, from Clarke *et al*. [5]. The weighting factor was changed between four different values to detect how it affected the performance of the controller. The forgetting factor in the RLS adapter was changed between two values. Typical choices of forgetting factor are in the range between 0.98 and 0.995, from Ljung *et al*. [9].

Several tests were executed and the parameters were changed in the tests to find the optimal parameters values. The goal was to control the process recently well and to avoid the RLS to fell asleep. The test parameters and a short observation note are shown in table 4.1.

**Table 4.1 Tests of the GPC including the RLS, with various values for *N2*, λ, *NU* and forgetting factor**

| Case | Min. cost horizon, $N_1$ | Max. cost horizon, $N_2$ | Cost horizon, *N* | Control horizon, *NU* | Control weight, λ | Forgetting factor, ff | Observation |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 4 | 4 | 0.3 | 0.98 | Not controlling the process |
| 2 | 2 | 6 | 5 | 2 | 0.3 | 0.98 | Controls the process well |
| 3 | 2 | 6 | 5 | 5 | 0.3 | 0.98 | Controls the process well |
| 4 | 2 | 7 | 6 | 5 | 0.1 | 0.98 | Oscillates |
| 5 | 2 | 7 | 6 | 5 | 0.3 | 0.98 | Controls the process well |
| 6 | 2 | 7 | 6 | 5 | 0.3 | 0.9 | Controls the process well |
| 7 | 2 | 7 | 6 | 5 | 0.5 | 0.98 | Controls the process well |
| 8 | 2 | 7 | 6 | 6 | 0.3 | 0.98 | Controls the process well |
| 9 | 2 | 7 | 6 | 6 | 0.3 | 0.9 | Controls the process well |
| 10 | 2 | 8 | 7 | 7 | 0.3 | 0.98 | Oscillates at low hot water flow rate |
| 11 | 2 | 10 | 9 | 5 | 0.3 | 0.98 | Oscillates at low hot water flow rate |
| 12 | 2 | 10 | 9 | 5 | 0.5 | 0.98 | Oscillates at low hot water flow rate |
| 13 | 2 | 10 | 9 | 5 | 0.8 | 0.98 | Oscillates at low hot water flow rate |
| 14 | 2 | 10 | 9 | 8 | 0.3 | 0.98 | Oscillates at low hot water flow rate |

From the experiments, the cost horizon was the most important parameters to the controller performance. If the cost horizon was too short, it seemed like the controller did not react to set point changes. This happened in case 1 table 4.1. If the cost horizon was too long, the output from the controller started to oscillate, particularly at low flows. This pattern was observed in cases 10-14 in table 4.1. A significant difference was detected between a cost horizon = 4 and a cost horizon = 5, and a graph comparison between case 1 and 3 of a set point change is displayed in appendix G.2.

The control horizon had not the same power to decide if the GPC managed to control the process or not. Changing the control horizon did not affect the control performance significantly. A comparison between case 2 and 3 from table 4.1 with a control horizon 2 and 5 respectively is displayed in appendix G.1. From the graph it seemed like the rise-time was a little shorter for the case with longer control horizon, but the response settled down faster for the case with shorter control horizon. The overshoot was also less for the case with shorter

control horizon. Therefore a shorter control horizon would be preferable.

The control weight, $\lambda$, was assumed constant for *j* - ahead prediction to simplify the algorithm. If the $\lambda$ was too small, it produced oscillations because it cost too little to change the control signal. The oscillations were unacceptable large with amplitude of about 2 $^{o}$C with a period of about 40 seconds for a control weight equal 0.1 in case 4 in table 4.1. When $\lambda$ increased, the oscillations decreased or disappeared but the responses became slower. A set point response with parameters from case 4 and 5 in table 4.1, with control weight of 0.1 and 0.3 respectively is displayed in appendix G.4.

The forgetting factor affected the RLS and not the GPC directly but through the *A* and *B* polynomials, which GPC got from RLS. There was not detected any significant difference in the GPC controller performance with different forgetting factors. A step response with parameters chosen as in case 5 and 6 from table 4.1 are compared in appendix G.3. The graph displays that there was only a random difference between the outputs in these to cases.

Between all the cases tested for the GPC controller, case 2 in table 4.1 with cost horizon equals 5, control horizon equals 2, control weighting factor equals 0.3 and forgetting factor equals 0.98, gave the best performance. In that case there were no oscillations, very little overshoot by set point changes and the output value was corrected quickly when disturbances were introduced.

## 4.2 Compare GPC and PID controller

A PID controller was implemented to benchmark the GPC. The same tests with increasing and decreasing set point, increasing and decreasing cold water flow rate, were carried out for the PID controller.

The PID controller made the process oscillate with tuning parameters found by Control Station. The period of the oscillations were about 40 seconds, and the amplitude is somewhat between 1.5 to 2 $^{o}$C. The sampling time was increased from 1.5 to 4 seconds and the number of times the for-loop for the slave controllers executed was increased from 5 to 11 times. The system still oscillated, but now with a slower period. The amplitudes were about the same as for a sampling interval of 1.5 seconds. In appendix I.1 a set point change with the two different sampling intervals is displayed.

The oscillations that occurred using the PID controller with tuning parameters found by Control Station were tried tuned down by changing the tuning parameters. To avoid oscillations the integral time was increased and the gain was reduced. The process gain was reduced from 2 to 0.5 and the integral time was increased from 60 seconds to 150 seconds. From earlier test the influence of the derivative time equals to 2 seconds was already so small that it remained unchanged. With increased integral time and reduced gain, the oscillations seemed to settle down to an acceptable level, but it took a long time, about 500 seconds from

start up. The GPC controller used not more then 1 minute to place the output at set point during startup. The time was of course dependent of the start conditions. A startup response with the two different sets of tuning parameters for the PID controller is displayed in appendix I.2.

The PID showed poorer control than GPC, mainly because of the oscillations. A comparison of the PID and GPC with both input and output is displayed in appendix I.3. The selected GPC controller was the same as in case 4 in table 4.1. The output temperature in the cold water outlet oscillated by using PID controller. By using the GPC controller, the temperature raised smoothly by set point changes. The rise-time was about 45 seconds shorter by using PID controller compare to GPC. The output from the PID controller was already increasing because of the oscillations when the set point increased. When the output from the GPC controller reached the new set point value, there was no overshoot so the control was very smooth. The hot water flow increased smoothly and had almost no overshoot. When using the PID to control the hot water flow, it oscillated between minimum and maximum flow almost all the time. This is of course not preferable, because it causes oscillations in the output and it caused unnecessary wear and tear on the valve.

# 5. DISCUSSION EXPERIMENTAL PHASE

## 5.1 Equipment

During the calibration, the relation between flow rate and voltage became poorly at higher flow rates. This area was then avoided for further tests. This result usage of only 1/4 of the maximum valve opening of both the hot water valve and the cold water valve. All the flows used in the test were actually low flows compare to the maximum flow that was possible with the present equipment. The poor relation between flow rate and voltage could be caused by the transformation of the signal between the D-to-A converter and the valve, as illustrated in figure 3.3. Since the capacity to the hot water valve was not used, the valve got easier saturated at bigger disturbances and set point changes when the constraints were included.

Hysteresis in the valves was a problem before cascade control was implemented. The slave controller seemed to handle this due to the hot water valve. Nothing were done to try to reduce was consider as a disturbance and not a manipulated variable, which was the case for the hot water.

The hot water inlet temperature changed during tests. If the temperature dropped so much that the valve got saturated during set point changes and disturbances, the test was ended. Most of the time the inlet temperature was stable, but it was a disturbance in the process.

The valves were changed by air pressure. The air pressure in the pipes might have been changed during tests. The pressure was not measured so how much it could have changed is unknown. Changes in air pressure was considered as a disturbance to the process, but from performance it seemed stable.

## 5.2 Implementation

The PRBS-test for the cascade case was executed with a sampling interval of 3 seconds. The ARX-model was based on data from the PRBS-test and used in the initial phase to RLS. The GPC performed with a sampling interval of 4 seconds. The sampling interval needed to be increased from 3 seconds to 4 seconds due to computation time for GPC and RLS. The computer could not compute one loop faster than 4 seconds. The PRBS-data would have changed if the sampling interval was 4 seconds instead of 3 seconds and the ARX-model might have been changed in values and degrees. The degrees of the model parameters $A$ and $B$ used in RLS were based on the optimal model from ARX. The PRBS-test should be carried out with the same sampling interval as GPC performs, so the PRBS capture the same dynamics and transferred info about the dynamics in the ARX model that was used in GPC. Since a sampling time of 3 seconds could not be executed and a PRBS-test with 4 seconds

sampling interval was not carried out, it is difficult to tell how much the influence of increasing the sampling time had on the control performance.

In the Matlab program that produced the ARX-model was limited with regard to the degrees of *A, B* and *d*. The degrees could only be a value from 1 to 5, to limit computations. With higher degrees of the model parameters, it got more complex. Aikake's FPE weighted between how correct model was and how complex the model was, so the gain of increasing the model degree could be lost because of higher complexity. A test with maximum degree of 10 instead of 5 for *A, B* and *d* gave a more complex optimal model. The degree of *A* and *B* increased but *d* remained the same. Since the system had a dead time about one sampling interval, this will not be limited of a maximum value of 5, and it was expected that *d* was equal in both cases. The FPE decreases 1.6% for the optimal model when the maximum degrees of *A* and *B* were increased from 5 to 10. The limitation of degrees of *A* and *B* should therefore not be significant for the control performance. A more complex model would may be needed a longer sampling interval because of bigger computations in Matlab. A simpler model was preferred, since this decreased the computation time.

Sampling time was an important parameter, so it must be chosen properly to avoid missing process dynamics. The GPC used a relatively long time to compute one loop, mainly because of the dynamic data exchange (DDE) to Matlab. This limited the choice of sampling time, because the sampling time needed to be a little bit longer than the compute time. The sampling time for the GPC was selected to be 4 seconds. Compare to the response-time, a sampling time of 4 seconds should be adequate. It is difficult to tell if there was any difference at a smaller sampling time, because it could not be tested with existing equipment. A shorter sampling interval could probably be selected if a faster computer had carried out the computations. Since it was particularly DDE that took time, the GPC controller could be implemented in Labview instead of Matlab, but the saved computation time by avoiding DDE could be lost since a Matlab is much better tool for matrix calculations than Labview.

The problem with the timing in the cascade control where the slave controller actually waited for the GPC to finish was not adequate. If a disturbance entered the process while the GPC was running, the slave controller could not correct the disturbance before the GPC sent the newly calculated control signal into the process. This was an implementation problem that can be improved. If the slave controller and master controller were implemented in two independent loops, the controllers could also run independently. Labview can deal with two running loops at the same time, but this was not tried to implement.

The future set point was implemented in such way that when it changed in Labview, Matlab observed this change in the next loop, actually next time when Labview poked values to Matlab. This made reference trajectory change immediately. Instead, the implementation could be done so the system could react before the change had effectively been made. By changing the reference trajectory gradually, effects of delay in the process response could be avoided. The effect with a gradually change in the reference trajectory would let the process made the response faster, but this was not critical for the heat exchanger, not even in the initial phase. However, this could be an improvement of the controller, but how big the

earnings will be was not certain.

The slave controller was a PI controller in velocity mode. Velocity mode was chosen because windup was avoided and bias term did not require specification. The slave controller was faster than the master controller, as it supposed to be, and a sampling interval of 0.25 second was therefor selected. The sampling interval was actually just a number used in calculation in the controller. The slave controller was implemented in a for-loop that executed a selected number of times. The for-loop executes as fast as it can, that is, the sampling interval in not the execution interval for the slave controller. The slave controller executed actually faster than 0.25 seconds. This could be improved if the slave controller and the GPC controller were implemented in two independent loops.

## 5.3 Observations

Clarke *et al*. [3] suggest that maximum cost horizon should be at least greater than the degree of the *B* polynomial, named as *nb*. The degree of $B(q^{-1})$ used in GPC in the tests was 6. The smallest cost horizon the GPC actually managed to control the process with was five.The maximum cost horizon was here smaller than the degree of the *B* polynomial. If the maximum cost horizon was increases to 7, thus greater than *nb,* the GPC also managed to control the process. From the test, it seemed like suggestions from Clarke *et al*. about the maximum cost horizon could be used as guidelines and not as absolute values for the control. The test shows that the cost horizon did not needed to be greater than *nb,* but at least equal to *nb* to control the process.

The significant difference in GPC performance between a cost horizon of 5 compare to a cost horizon of 4 was striking. It seemed like there existed a threshold value equals to 5 where GPC manage to control the heat exchanger. At a shorter cost horizon the GPC did not manage to control the process, it seemed like the GPC did not react to neither set point changes nor disturbances. The reason for this behavior could be that the cost horizon was so short that the algorithm was not capable to include the dynamics in the process. This threshold value was likely to change with the degree of the *B* polynomial. A more complex model could for instance be tested to see if threshold value changed with a *B* polynomial of higher degree. The threshold value would probably increased with an increasing degree of the *B* polynomial.

Clarke *et al*. [3] suggest also that the minimum cost horizon $N_1$ could be set equal to the dead time if the dead time to the process is known. In all tests the minimum cost horizon was chosen to be equal to two. From a step test that was executed, the dead time was between 3 and 6 seconds, dependent on the hot water flow rate. Even though the dead time varies with the flow rate, it was about one sampling interval. Since the dead time was about one sampling interval, the GPC algorithm did not loose any information by set $N_1$ equals to two. Choosing the minimum cost horizon bigger than 1 if the dead time was variable or unknown was a risk

to loose information. When choosing $N_1$ equals or bigger than dead time reduce computation in the algorithm. But there is no point setting $N_1$ less than the dead time because the output is only affected by the past outputs and inputs.

The control horizon showed better control if it had a smaller value like 2 instead of a larger value like 5. This as can be seen in the comparison graph in appendix G.1. From Clarke *et al*. [3] the control becomes smooth and sluggish if the control horizon is chosen as small values. Larger values of the control horizon provide more active controls. From the comparison graph you can see that the rise-time was shorter for the case with longer control horizon. It was more active and wanted to change the set point fast. In the case where the control horizon is shorter, the rise time is longer but when the manipulated variable reach se point, the value laid closer to the set point, the control was smoother. This observation agreed with the literature.

Clarke *et al*. [3] describes that a large class of plant models can be stabilized by GPC with default values of 1 and 10 for $N_1$ and $N_2$ and $\lambda$ can be set to zero. These parameters values were not tested, but from the tests this values will probably give large oscillations. From the tests the oscillations increased when value of $\lambda$ parameters was decreased. The lowest $\lambda$ value tested was 0.1 and this value gave oscillations particular at low flow rate when the cost and control horizon indicated good control from other tests with a larger $\lambda$ value. The largest maximum cost horizon tested was 10, that equals a maximum cost horizon of 9. This gave oscillations at particular low flow rate. The default values suggested by Clarke *et al*. [3] gave not sufficient control of the heat exchanger.

Stability was affected by the cost horizon. If the cost horizon was too long, the output from the heat exchanger started to oscillate at particularly low flows. When the hot water flow was around 5 l/min the set point to the slave controller oscillated between 0-10 l/min. The set point to the slave controller changes between 10-13 l/min when the hot water flow is around 11 l/min. Both these observations were done when cost horizon was seven or bigger. The oscillations in the input produced oscillations in the outputs, especially at low flows. The oscillations occurred because of the prediction from the GPC controller was poorer. The predictions was based on the ARX-model that updated by RLS. The ARX-model had constant values for the degrees of the *A* and *B* polynomials from fitting the ARX-model to PRBS-data. The PRBS-test was executed with the binary outputs 10 l/min and 14 l/min, which was actually in the higher hot water flow rate range. The binary inputs in the PRBS-test should be a compromise between high and low hot water flow rates, but since the inputs were more in the high flow rate area, it was not surprising that prediction is better in the high flow rate area. The dynamics in the process changed with different flow rates, for instance the dead time increased at lower flows. The *B* polynomial in the model might have too low degree to detect all the dynamics at low flow. This could be improved by making an adapter where the model order is changeable.

The PID had problems with controlling the process due to the oscillations that occurred. The oscillations may be caused from the interactions between the PID and the slave controller. The PID controller could be tested without the slave controller to find out if it was the

interactions that caused the oscillations. A problem with testing PID controller without slave controller was that hysteresis in the hot water valve could gives oscillations, therefore it would be difficult to detect whether or not there were interactions between the PID and slave controller. The GPC needs the slave controller to deal with the non-linearity, because the GPC is a model based controller. The PID is not model-based so the controller may not actually need a slave controller to deal with the non-linear part.

A lower control weight value could be considered instead of a higher value that gave better control if there were little noise in the system. It seemed like the process itself produced enough noise to keep the RLS active, because of the resolution to the temperature measurements were low enough to avoid the RLS to fall asleep.

# 6. THEORY MULTIVARIABLE CASE

In many cases a change in one manipulated variable affects more than one of the controlled variables. These interactions between process variables may result poor control performance. If the interactions are not negligible, the control structure needs to be changed from a set of independent loops to controllers with multi-inputs multi-outputs. One of the advantages of GPC algorithm is that the method in multivariable case is similar to the single variable case.

## 6.1 ARX-model in multivariable case

The derivation is extracted from Camacho and Bordons [1].
The CARIMA model in equation (2.4) remains the same for the multivariable case, but it is now a matrix equation. In the multivariable case with *ny* outputs and *nu* inputs, $A(q^{-1})$ and $C(q^{-1})$ are *ny* x *ny* monic polynomial matrices and $B(q^{-1})$ is *nu* x *nu* polynomial matrix. They are defined as

$$\begin{aligned} A(q^{-1}) &= I_{ny \times ny} + A_1 q^{-1} + A_2 q^{-2} + \ldots + A_{na} q^{-na} \\ B(q^{-1}) &= B_0 + B_1 q^{-1} + B_2 q^{-2} + \ldots + B_{nb} q^{-nb} \\ C(q^{-1}) &= I_{ny \times ny} + C_1 q^{-1} + C_2 q^{-2} + \ldots + C_{nc} q^{-nc} \end{aligned} \tag{6.1}$$

The variables *y(t), u(t)* and *e(t)* are now *ny* x 1 output vector, a *nu* x 1 input vector and a *ny* x 1 noise vector respectively in the CARIMA model.

To describe the system an auto regressive model with input, ARX, can be used. The ARX-model is based on PRBS data given from simulation. To develop values for *A* and *B* the PRBS data are fitted to an ARX-model. The fitting is done by Matlab, and the algorithm is displayed in appendix J.1.

## 6.2 Diophantine equation in multivariable case

In the GPC algorithm the Diophantine equation needs to be solved. The Diophantine equation for the multivariable case can be written as

$$I_{ny \times ny} = E_j(q^{-1})\tilde{A}(q^{-1}) + q^{-j}F_j(q^{-1}) \tag{6.2}$$

where *ny* is number of outputs, $\tilde{A}(q^{-1}) = A(q^{-1})\Delta$ where $\Delta$ equals $1-q^{-1}$, $\boldsymbol{E}_j$ and $\boldsymbol{F}_j$ are unique polynomial matrices of order $j-1$ and $n_a$ respectively. The polynomial matrices $\boldsymbol{E}$ and $\boldsymbol{F}$ can be written as

$$E_j(q^{-1}) = E_{j,0} + E_{j,1}q^{-1} + E_{j,2}q^{-1} + \ldots + E_{j,j-1}q^{j-1}$$

$$F_j(q^{-1}) = F_{j,0} + F_{j,1}q^{-1} + F_{j,2}q^{-1} + \ldots + F_{j,na}q^{-na}$$

Solving Diophantine equation can be done by recursion, the same method as in the single variable case, as described in chapter 2.3.1. The only difference is the equations are now matrix equations.

Consider the Diophantine equation corresponding to the prediction of $(t + j + 1 \mid t$

$$I_{ny \times ny} = E_{j+1}(q^{-1})\tilde{A}(q^{-1}) + q^{-(j+1)}F_{j+1}(q^{-1}) \tag{6.3}$$

Subtracting equation (6.2) from equation (6.3) gives

$$0_{ny \times ny} = (E_{j+1}(q^{-1}) - E_j(q^{-1}))\tilde{A}(q^{-1}) + q^{-j}(q^{-1}F_{j+1} - F_j(q^{-1})) \tag{6.4}$$

The subtraction between the two different ***E*** polynomials in equation (6.4) is of degree *j* and can be written as

$$E_{j+1}(q^{-1}) - E_j(q^{-1}) = \tilde{R}(q^{-1}) + R_j q^{-j} \tag{6.5}$$

where $\tilde{R}(q^{-1}) = R(q^{-1})\Delta$ and $\boldsymbol{R}(q^{-1})$ is a *ny* x *ny* polynomial matrix of degree smaller or equal to *j*-1 and $R_j$ is an *ny* x *ny* real matrix. By substituting equation (6.5) into equation (6.4), it gives

$$0_{ny \times ny} = \tilde{R}(q^{-1})\tilde{A}(q^{-1}) + q^{-j}R_j\tilde{A}(q^{-1}) + q^{-j}(q^{-1}F_{j+1} - F_j(q^{-1})) \tag{6.6}$$

Since $\tilde{A}(q^{-1})$ is monic, that is all coefficient are non-zero and the first coefficient are equal to one, $R(q^{-1})$ needs to be equal to $0_{ny \times ny}$, according to equation (6.6). This means ***E*** matrix can be computed recursively by

$$E_{j+1}(q^{-1}) = E_j(q^{-1}) + R_j q^{-j} \tag{6.7}$$

From equation (6.6) and the relation in equation (6.7), following expressions for ***F*** matrix can be obtained

$$R_j = F_{j,0}$$

$$F_{j+1,i} = F_{j,i+1} - R_j\tilde{A}_{i+1} \text{ for } i = 0 \text{ to the degree of } \boldsymbol{F}_{j+1}$$

Initial conditions for the recursion equation can easily be seen from equation (6.2) and are given by

$$E_1 = I_{ny \times ny}$$
$$F_1 = q(I - \tilde{A})$$

This recursion method can be implemented and solved in for instance Matlab.

## 6.3 GPC in multivariable case

After solving the Diophantine equation, the matrix $\boldsymbol{G}$ can be calculated from

$$E_j(q^{-1})B(q^{-1}) \;=\; G_j(q^{-1}) \;+\; q^{-j}G_{jp}(q^{-1}) \tag{6.8}$$

with the degree of $\boldsymbol{G}_j$ is less than $j$. The prediction equation can now be written as

$$\hat{y}(t+j|t) \;=\; G_j(q^{-1})\Delta u(t+j-1) + G_{jp}(q^{-1})\Delta u(t-1) + F_j(q^{-1})y(t) \tag{6.9}$$

The last two terms on the right hand of equation (6.9) only depends on past values of the process outputs and inputs and correspond to the free response of the process.

Equation (6.9) can be rewritten as

$$\hat{y}(t \;+\; j \,|\, t) \;\;=\;\; G_j(q^{-1})\Delta u(t \;+\; j \;-\; 1) \;+\; f_j \tag{6.10}$$

where $f_j$ is the free response and is equal to $G_{jp}(q^{-1})\Delta u(t \;-\; 1) \;+\; F_j(q^{-1})y(t)$. Matrix $\boldsymbol{G}_j$ is of dimension *ny* x *nu* and $\boldsymbol{f}_j$ is of dimension *ny* x *1*. Now, consider a set of *N j*-ahead predictions:

$$\begin{aligned} +\;1\,|\,t) \;\;&=\;\; G_1\Delta u(t) \;+\; f_1 \\ +\;2\,|\,t) \;\;&=\;\; G_2\Delta u(t \;+\; 1) \;+\; f_2 \\ &\cdots \\ +\;N\,|\,t) \;\;&=\;\; G_N\Delta u(t \;+\; N \;-\; 1) \;+ \end{aligned} \tag{6.11}$$

Due to the recursive properties of the $\boldsymbol{E}_j$ polynomial matrix, expressions in equation (6.11) can be rewritten as

$$\begin{bmatrix} \hat{y}(t+1|t) \\ \hat{y}(t+2|t) \\ \cdots \\ \hat{y}(t+j|t) \\ \cdots \\ \hat{y}(t+N|t) \end{bmatrix} = \begin{bmatrix} G_0 & 0 & \ldots & 0 & \ldots & 0 \\ G_1 & G_0 & \ldots & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ G_{j-1} & G_{j-2} & \ldots & G_0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ G_{N-1} & G_{N-2} & \ldots\ldots & \ldots & \ldots & G_0 \end{bmatrix} \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \cdots \\ \Delta u(t+j-1) \\ \cdots \\ \Delta u(t+N-1) \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_j \\ \cdots \\ f_N \end{bmatrix} \tag{6.12}$$

The matrix equation in (6.12) can be written in a more condense form as

$$y \;\;=\;\; Gu \;+\; f \tag{6.13}$$

where the $\boldsymbol{G}$ is a matrix which contains several smaller matrices $\boldsymbol{G}_j$ with $j = 0...N\text{-}1$. Matrix $\boldsymbol{G}$ has the dimension $N{\cdot}ny$ x $NU{\cdot}nu$ and $y$, $u$ and $f$ have dimensions $N{\cdot}ny$ x 1, $NU{\cdot}ny$ x 1 and

*N·ny* x 1 respectively.

The cost function in the multivariable case is quite similar to the cost function in the single variable case given in equation (2.19) and can be written as

$$J(N_1, N_2, N_u) = \sum_{j=N_1}^{N_2} ||\hat{y}(t+j|t) - w(t+j)||^2_R + \sum_{j=1}^{N_u} ||\Delta u(t+j-1)||^2_Q \quad (6.14)$$

where $\boldsymbol{R}$ and $\boldsymbol{Q}$ are positive definite weighting matrices, $N_1$ is the minimum cost horizon, $N_2$ is the maximum cost horizon and *NU* is the control horizon. If equation (6.13) is introduced in the cost function given in equation (6.14), the cost function can be written as a quadratic objective function

$$J(u) = \frac{1}{2}u^T H u + bu + f_0 \quad (6.15)$$

as deduced in chapter 2.3.2. The coefficients in the quadratic function are

$$\begin{aligned} H &= 2G^T R G + Q \\ b &= 2(f - w)^T R G \\ f_0 &= (f - w)^T R (f - w) \end{aligned}$$

Matrix $\boldsymbol{G}$ is the matrix calculated from equation (6.8), $f$ is the free response and $w$ is the future set point. The matrix $H$ has dimension (*NU·nu* x *NU·nu*), $b$ has dimension (1 x *NU·nu*) and $f_0$ is a scalar. The weighting matrices $\boldsymbol{R}$ and $\boldsymbol{Q}$ have dimension (*N·ny* x *N·ny*) and (*NU·nu* x *NU·nu)* respectively.

## 6.4 Including constraints in the GPC algorithm

When signals in the process are limited, we have a process subject to constraints. When constrains actions like limits on the control signal or limits on the output signals, this should be included in the GPC algorithm. Constraints are relatively easily to incorporate in the GPC algorithm.

Constrains on the input variables are usually due to physically constraints, for instance a valve opening is limited that puts a limit on the flow. A calculated control signal from GPC which is out of range, becomes limited by either the control program or by the actuator. Anyway, the reason for incorporate input constrains in the GPC algorithm is to make sure that the optimum will be obtained and this may not happen when constraints are violated in the GPC algorithm.

Constrains on the outputs are mainly incorporated because of safety reasons or product specifications. Constraints on the outputs give the possibilities to have set point ranges

instead of a single set point values. This can be attractive in from an optimization point of view. Composition product specifications are typically given in minimum or maximum values. When a mole fraction specification is given as minimum 0.95, a set point value of 0.95 means that 0.94 and 0.96 is equally poor, which in many cases does not make sense. By introducing set point range this is avoided and the optimization gets more freedom to change the manipulated variables, and a better combination of control variables due to the cost may be obtained.

For a *nu*-input *ny*-output process with constrains acting over a cost horizon *N* and control horizon *NU*, constraints on control signals and output variables can be expressed respectively as

$$\begin{aligned} lu_{min} \le u \le lu_{max} \\ ly_{min} \le Gu + f \le ly_{max} \end{aligned} \tag{6.16}$$

where equation (6.13) is inserted for the variable *y* and ***l*** is a (*N·ny x nu*) matrix formed by *N* identity matrices with dimension (*ny x nu*). Subscript *min* and *max* indicates the limited region for input and output variables.

The constraints can be expressed in a single matrix equation as

$$Ru \le c \tag{6.17}$$

where

$$R = \begin{bmatrix} I_{(N \cdot ny) \times (NU \cdot nu)} \\ -I_{(N \cdot ny) \times (NU \cdot nu)} \\ G \\ -G \end{bmatrix}$$

where ***I*** is the identity matrix and ***G*** is the matrix in equation (6.13). The ***c*** matrix in the constraints equation is

$$c = \begin{bmatrix} lu_{max} \\ -lu_{min} \\ ly_{max} - f \\ -ly_{min} + f \end{bmatrix}$$

The QP-problem has the standard form

$$min\ 0.5 \cdot u^{T} \cdot H \cdot u + b \cdot u + f_0 \tag{6.18}$$

$$subject\ to\ Ru \le c$$

The optimization of control actions calculated by GPC can then be solved as a quadratic problem (QP) since the cost function is a quadratic function as shown in equation (6.15) and its constraints are linear inequality or equality as shown in equation (6.17).
When the optimization problem is formulated like in equation (6.18), it can easily be solved in for instance Matlab with the *quadprog*-function.

## 6.5 Recursive least squares parameters estimation in multivariable case

The recursive least squares (RLS) parameters estimation method is used to update the model parameters ***A*** and ***B***, which are on the form given in equation (6.1). The RLS algorithm is similar to the algorithm in the single variable case described in chapter 2.5. Again the difference between the single variable and the multivariable case is dimension on the equations. In the multivariable case the parameter vector in equation (2.40) becomes a parameter matrix on the form

$$\hat{\beta} = \begin{bmatrix} A_1 & A_2 & \dots & A_{na} & B_1 & \dots & B_{nb} \end{bmatrix}^T \tag{6.19}$$

where *na* is the degree of the ***A*** polynomial and *nb* is the degree of the ***B*** polynomial. Parameter matrix $\beta$ has dimension (*na*·*ny*+*nb*·*nu* x *ny*). The matrices $\boldsymbol{A}_1$ to $\boldsymbol{A}_{na}$ indicates each a matrix of the form

$$A_j = \begin{bmatrix} a_{11,\,j} & \dots & a_{1ny,\,j} \\ \dots & \dots & \dots \\ a_{ny1,\,j} & \dots & a_{nyny,\,j} \end{bmatrix} \tag{6.20}$$

The ***B*** matrices have the same form as the ***A*** matrices.
The output matrix is defined as

$$y = \begin{bmatrix} y_1(1+na+d) & \dots & y_{ny}(na+d+1) \\ \dots & \dots & \dots \\ y_1(N+na+d) & \dots & y_{ny}(N+na+d) \end{bmatrix} \tag{6.21}$$

where *N* is the cost horizon and *d* is the dead time in the system. Output matrix ***y*** has dimension *N* x *ny*. The error matrix has the same structure as the output matrix. The matrix $\phi$ is defined as

$$\phi = \begin{bmatrix} y_{1,1}(na+d) & y_{1,1}(na+d-1) & \dots & y_{1,1}(1+d) & \dots & y_{ny,1}(na+d) & \dots & y_{ny,1}(1+d) & u_{1,1}(nb) & \dots & u_{1,1}(1) & \dots & u_{nu,1}(nb) & \dots & u_{nu,1}(1) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ y_{1,N}(na+d) & y_{1,N}(na+d-1) & \dots & y_{1,N}(1+d) & \dots & y_{ny,N}(na+d) & \dots & y_{ny,N}(1+d) & u_{1,N}(nb) & \dots & u_{1,N}(1) & \dots & u_{nu,N}(nb) & \dots & u_{nu,N}(1) \end{bmatrix}$$

and has the dimension *N* x *na*·*ny*+*nb*·*nu*. The evaluation for the model is now a matrix equation on the same form as equation (2.44).

The RLS algorithm can be described as in figure 2.3. The vector $\psi$ containing past data has now the form

$$\psi = \begin{bmatrix} -y_1(N) \\ \cdots \\ -y_{ny}(N) \\ \cdots \\ -y_1(N-na+1) \\ \cdots \\ -y_{ny}(N-na+1) \\ u_1(N-d) \\ \cdots \\ u_{nu}(N-d) \\ \cdots \\ u_1(N-d-nb+1) \\ \cdots \\ u_{nu}(N-d-nb+1) \end{bmatrix}$$

The covariance matrix $P$ has the dimension $na \cdot ny + nb \cdot nu$ x $na \cdot ny + nb \cdot nu$, but the parameter $\alpha$, representing the variance in the error, is still a scalar. $\alpha$ now represent the average error of the error in each variable.

The adapter can be implemented as a function and solved in for instance Matlab.

# 7. EXPERIMENTAL SIMULATION PHASE

## 7.1 Description of simulation process

The simulation case was based on information from Bill R. Minton in Black & Veatch Pritchard, Inc [7], who gave the problem to a senior design project at University of Colorado at Boulder. Parts of simulation result from the senior design project by Halevi *et al*. [7] was also used as a starting point in the simulations, like selecting stage for side draws and feed streams, operating pressure and flow rate in recycle streams. Control strategy and control was the main focus in the simulation. The simulations took place in HYSYS.Plant.

The main task to the process was separation of natural gas liquid (NGL) into ethane, propane, iso-butane, n-butane and gasoline. The whole separation process contained 5 distillation towers, T-1 to T-5, where the first distillation column had neither condenser nor reboiler, and auxiliary equipment like pumps, heat exchangers, coolers, heaters and compressors.

The two first columns, T-1 and T-2, were coupled through two recycle streams. The recycle streams were side draws from column T-2, one vapor stream and one liquid stream. The liquid stream returned to the top of column T-1 and served as reflux, the vapor stream returned in the bottom of T-1 and served as boil up. Product streams from T-1 were feed streams at different stages in T-2. The T-2 column was a dual distributor where ethane was a product stream from condenser and propane was a product from side draw. The bottom stream from reboiler in T-2 went to further separation.

It is only the two coupled columns that were studied here. The simulation was also simplified by omitting preheating of the feed, cooling and pressure rising of the product streams. When the process was simplified by disregarding pressure and temperature specifications in the product streams, only three auxiliary components were needed; one pump, one cooler and one compressor.

The process was simulated in steady state and dynamic mode where control performance was studied. The coupled columns were interesting from a control point of view, because if poor control was implemented, disturbances were sent back to the process by the recycle streams. The product specifications could therefore be hard to maintain with too big disturbances in the process. The second column T-2 was also very important because it produced two products and a bottom product that went on to further separation.

The main goals of the coupled distillation column were to maintain specification on product streams ethane and propane. The product specifications are shown in table 7.1.

**Table 7.1 Product spesifications in the simulation**

| Component | Ethane | Propane |
|---|---|---|
| Pressure [psig] | 1000 | 300 |
| Temperature [F] | max. 100 | max. 100 |
| Composition | C2: min 96.0 mol% | C3: min 95.0 LV% |
| " | C3: max 2.0 mol% | C4+: max 2.5 LV% |
| " | C4: max 0.25 mol% | |

Only the key composition specifications were considered for the products, i.e. ethane in ethane product and propane in propane product. Pressure and temperature specifications were not considered for the product streams since the process was simplified as mentioned above. The composition specifications on the bottom stream from T-2 were also considered, since the bottom stream went to further separation where other specifications needed to be maintained. All the propane in bottom stream from T-2 was assumed to consign in the iso-butane product, since this was the lightest product produced from the bottom stream in T-2. Since the iso-butane product should not contain more than 0.5 LV% propane, this value was used as a specification value for the bottom stream from T-2 column.

Both distillation towers were trayed towers, simulated as sieve-plate dispersers. Number of stages was given in the process description but column diameter needed to be calculated. Calculations were based on methods described in Løvland [10] and Perry and Chilton [11].

The calculated tray diameter in column T-1 would HYSYS.Plant not accept, it gave an error message that tray section dry hole pressure drop is too large. HYSYS.Plant suggested a bigger diameter, and the suggested diameter for T-1 was used in simulations. The diameter for column T-2 was increased in simulations because the tray section dry hole pressure drop became too large when the reboil ratio was bigger than 0.96 for a shorter period. Key sizing parameters for the columns are given in table 7.2.

**Table 7.2 Key sizing parameters for the simulated distillation columns**

| Columns | T-1 | T-2 |
|---|---|---|
| # of trays | 20 | 60 |
| Tray diameter [m] | 2.1 | 9.2 |
| Tray diameter used in simulation [m] | 5.9 | 10.5 |
| Average pressure in column [kPa] | 2066 | 2055 |

The composition profiles for ethane and propane in steady state were studied to check which trays the side draws should be placed, particularly the propane side draw. In steady state there was a problem to maintain the composition specifications for the propane stream. In the steady state simulations, propane side draw was placed at stage 27, after suggestions from [7]. From the composition profile, displayed in appendix K, a propane side draw from stage 20 was more preferable. In dynamic simulation the propane side draw was therefore placed at stage 20. When the propane side draw was moved to stage 20, the composition profile remains almost unchanged.

Column T-1 had a feed stream entering at stage 10 and the vapor and liquid recycle streams from column T-2 acted as boil up and reflux streams in column respectively. Column T-2 had two feed streams and both streams were from column T-1. The overhead from T-1 entered column T-2 at stage 14, bottom stream from T-1 entered at stage 44. Column T-2 had three side draws, i.e. propane product stream, liquid recycle stream and vapor recycle stream. These side draws were placed at stage 20, 14 and 45 respectively. The bottom stream from column T-2 was outlet from reboiler and ethane product stream was overhead from condenser.

The liquid recycle stream from T-1 got cooled about 57 $^{o}$C and then it went through a pump with pressure rise of approximately 82 kPa. The vapor recycle stream went through a compressor with a pressure rise of 18 kPa. It was not considered what kind of energy sources that were used in the different energy streams.

The economy for the process was not considered. There were only three energy streams in the process in the addition to the reboiler and condenser duty in column T-2. Since pressure and temperature specifications for the product stream were not considered, several energy streams were actually missing compare to a real process.

The process flow diagram of the simulated process is displayed in figure 7.1.

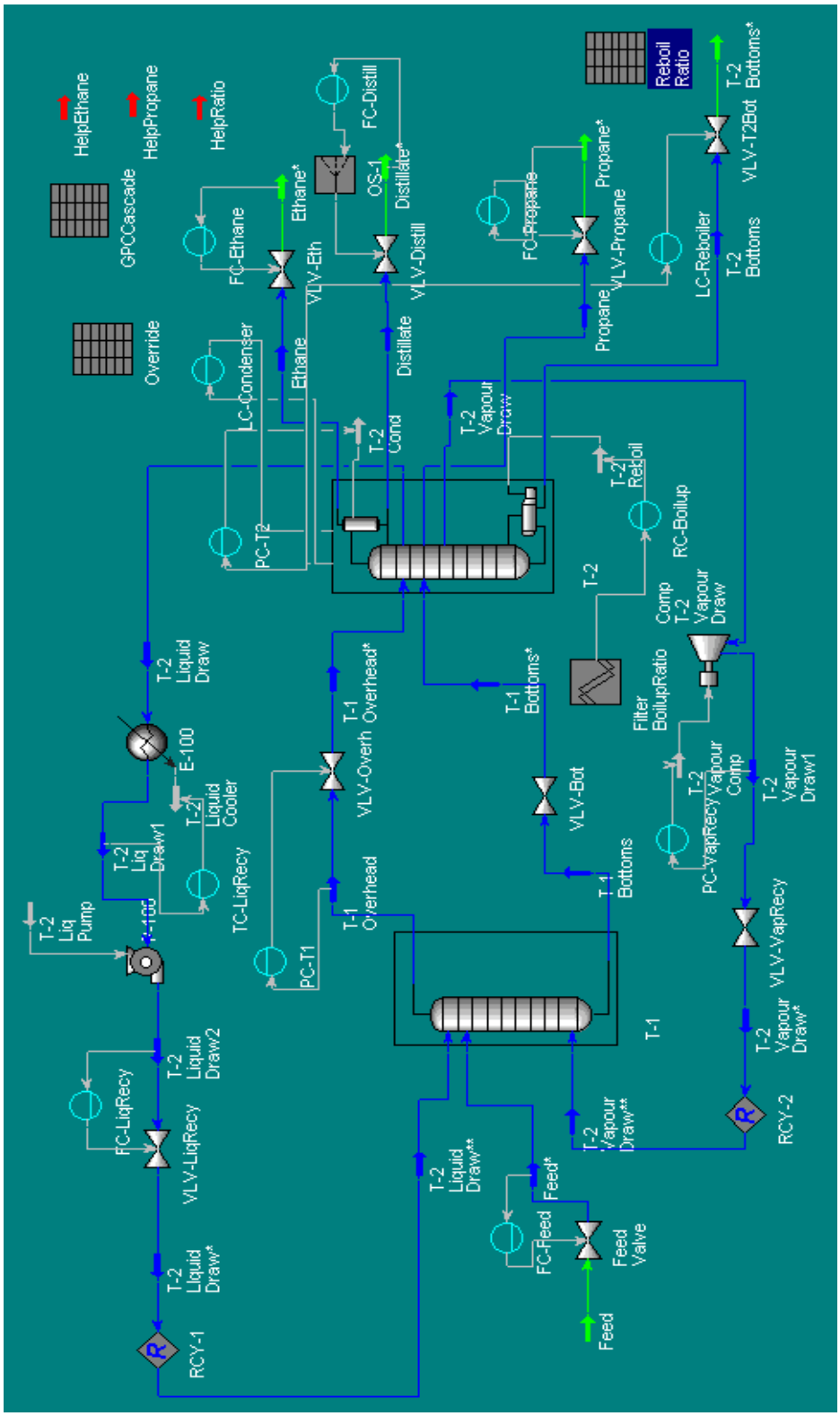



**Figure 7.1 Process flow diagram of the simulated process including control structure.**

## 7.2 Control structure

The main task for the controllers was to maintain specifications for the product streams ethane and propane, and the bottom stream from the column which went on to further separation. An another important case was to try to keep the recycle streams constant to avoid disturbances were sent back to the process. The pressures in the columns were also important to keep constant, since they had influence on almost all the streams in the simulated process.

The feed stream entering the process has a flow controller to avoid flow rate disturbances further down in the process. Preheating and pressure adjustments of the feed were not included in the simulation and therefore no control actions were considered on the pressure and the temperature in the feed stream.

The first distillation column that contains neither reboiler nor condenser had only one controller. The pressure in the column was controlled by the flow rate of the overhead stream from the column.

The vapor recycle stream, which was a side draw from T-2 and went back to the bottom of column T-1, was pressure controlled. A compressor was needed to increase the pressure in the vapor recycle stream, due to pressure drop downstream in the process. Compressor duty was used as manipulated variable to control the pressure in the stream.

The flow rate to the liquid recycle stream, which entered the top stage at column T-1 was controlled by a flow controller connected to a valve at the side draw. The liquid recycle stream got cooled before entering the T-1 column. A temperature controller controlled the the cooler duty which was the manipulated variable.

The second column, T-2, had much more advanced control structure than T-1 column because it contained both reboiler and condenser, but it had also several side draws. The liquid percent level in reboiler was controlled via a valve attached on the outlet stream from the reboiler. A level controller attached to a valve on the reflux stream controlled the liquid percent level in the condenser.

The pressure in column T-2 was tried to controlled by the product stream from the condenser, that is, the ethane stream. This produced large fluctuations in ethane flow rate and also in the composition. The composition specifications on the ethane stream could not be maintain with this large fluctuations, so the pressure in the column was therefore controlled by manipulating the condenser duty instead.

The condenser had partial reflux but the bottom stream from the condenser was only used in emergencies at very high liquid levels in the condenser. There was a flow controller on the bottom stream from the condenser, which was connected to a valve through a selector. The selector worked as a sort of alarm that opened the valve when the liquid percent level was higher than 90%.

Compositions to the product streams and bottom stream from the column, were sensitive to the temperature profile in the column. To control the outlet compositions, the temperature profile in the column where tried to be maintained via the reboiler duty. This seemed to be difficult because the temperature profile between the reboiler and the last stage where a side draw stream was removed from the column was quite flat. When the temperature transmitter was placed at higher stage in the column, the side draws leaving the column and feed stream entering the column affected the temperature too much. Also the distance between the reboiler and the stage where temperature transmitter was placed increased and the effect between the control signal and manipulated variable decreased.

Instead, the boil up ratio was chosen to control the temperature profile. This worked much better due to maintain the composition specifications. The reboil ratio was the controlled variable that was calculated in a spreadsheet in HYSYS.Plant. Reboiler duty was the manipulated variable in the reboil ratio controller.

The product streams, ethane and propane, have each a flow controller which was connected to a valve on the streams. Set points to these two flow controllers and the reboil ratio controller needed to change due to the feed composition and interactions between the controllers.

All the controllers were PI controllers. After a control structure was established, the controllers needed to be tuned. Each controller was set in manual mode and step test on the output variable for the controller was performed. The responses were registered and the response data analyzed with Skogestad's tuning rules [13]. Since all the PI controllers were treated as SISO controllers, no interactions were considered between the controllers.

The tuning parameters calculated from Skogestad's tuning rules were implemented in the PI controllers and thereafter the performance for each controller where tested by step tests. Some PI controllers had too aggressive parameters, so these parameters were adjusted. Particularly gain parameters in flow controllers calculated from Skogestad's tuning rules seemed to be too aggressive.

To deal with interactions between controllers, the reboil ratio controller and flow controller on ethane and propane streams were put into a cascade where the master controller was a multivariable GPC. The GPC algorithm gave set points to the slave controllers based on the past outputs, past inputs and model for the process.

The GPC algorithm considered three inputs and three outputs, also a multivariable case. The inputs in the GPC were ethane flow rate, propane flow rate and reboil ratio. The outputs were mole fraction ethane in ethane stream, liquid volume fraction propane in propane stream and liquid volume fraction propane in bottom stream from the T-2 column.

The PI controllers in the simulation could be divided into two groups; local controllers and secondary controllers in cascade. The local controllers were not involved in the cascade, and they were also unknown to the GPC algorithm. Their purpose was to keep process variables

at their set point and smooth out disturbances produced by the cascade. The slave controllers got set point values from the GPC algorithm, and were in that manner much more active controllers, since the set point values for these controller changed during the simulation.

## 7.3 Implementation

A multivariable GPC controller and a built-in MPC controller in HYSYS.Plant were tested in the simulations.

The first task was to establish the process in HYSYS.Plant. The process was first simulated in steady state to obtain steady state values, something that was very useful when switching to dynamic mode. The control structure as described in chapter 7.2 was implemented and the simulation took then place in dynamic mode.

The pressure and temperature for the products including the bottom stream from column T-2 were not satisfied the specifications quoted in [7]. Pressure- and temperature adjustments after the streams had left the column were not important for the MPC algorithms and therefore not included in the simulations.

In dynamic mode the reboil ratio signal had numerical problems because the value shifted randomly between two values. The difference between the values was relatively large, so the signal was too noisy to use in control. The reboil ratio value was therefore smoothed by a transfer function block in HYSYS.Plant with gain 1 and time constant 1 minutes to avoid noisy signals in the control.

### *7.3.1 Implementation of the GPC controller*

The GPC algorithm including the RLS adapter was implemented in Matlab, based on the theory described in chapter 6. The Matlab algorithm including the recursion of the Diophantine equation, RLS parameter estimation and initializing the algorithm is given in appendix J.

Calculation of the GPC algorithm was executed by Matlab. The simulation of the process took place in HYSYS.Plant, and a link between Matlab and HYSYS.Plant was therefore needed to be established to exchange information. This could be done through Visual Basic for Applications (VBA). VBA acted as a server where HYSYS.Plant and Matlab were clients. VBA was programmed to get necessary information from HYSYS.Plant and sent the information to Matlab, then asked Matlab to execute the GPC algorithm. VBA got then the control signals calculated from the GPC algorithm and sent the data to HYSYS.Plant. VBA then commanded HYSYS.Plant to run the simulation for a certain time. The control interval for the GPC was chosen to be 2 minutes, so the HYSYS.Plant simulation ran for two minutes before a new loop was started. The VBA algorithm is displayed in appendix L.

The GPC algorithm needed a starting model for the calculation of the predicted outputs. To develop a starting model, a PRBS-test had been performed, as described in chapter 2.1. The PRBS-test was accomplished with three binary output signals. The binary inputs selected low and high values independently. The binary signals were chosen in the operating range from the steady state values and responses in ethane in ethane stream, propane in propane stream and propane in T-2 bottom stream. The data were analyzed in Matlab and an ARX-model is fitted to the data. Matlab had numerical problems when using the *arx* function when searching for the model parameters. Matrices were close to singular and results may then be inaccurate. Therefore the PRBS-data were normalized to avoid this problem. The model parameters then became normalized, so all data in GPC algorithm needed also to be normalized.

To make the GPC controller to work in a cascade, the values from Matlab could not be put directly to the set point in the slave controllers. This would not work for some reason from VBA. Therefore the new control signals calculated from GPC were placed as molar flow in material streams in HYSYS.Plant. The control signals were then treated in a spreadsheet in HYSYS.Plant to make the reboil ratio unit less, before cells in spreadsheet were set as a remote set point source to the slave controllers. When the calculated signals finally were put to the set points in slave controllers, the values were changed, probably due to units. The control signals were therefore scaled in Matlab before they were sent to HYSYS.Plant. Set points to the flow controllers on the ethane and propane streams were scaled by a constant ratio. The set point to the reboil ratio controller had a linear relation and was scaled by a linear equation.

In the GPC algorithm the free response was calculated. The free response needed set points to the outputs in its calculations. Since the outputs in GPC had set point ranges instead of exact set points, a pretended set points were needed in the algorithm. The pretended set points were the same as the starting values in the simulations.

The GPC algorithm solved a QP-problem with linear constraints, as describes in chapter 6.4. The constraints on the control signals and manipulated variables were mostly based on specifications and physical limits. The control signals were ethane molar flow, propane molar flow and reboil ratio. The molar flows of ethane and propane had a lower physical limit of 0 kmol/h and the upper limit was set to 3000 kmol/h and 2500 kmol/h respectively. The lower limit to the reboil ratio was set to 0.5 because vapor flow was needed in the column to prevent weeping and to maintain pressure and separation capacity. The manipulated variables were all composition fractions. The upper physical limits to the ethane and propane purity were 100 mol% and LV% respectively. The lower limit due to specifications for the ethane purity was 96 mol% and the propane purity needed to be at least 95 LV%. Propane in T-2 bottom stream had a physical lower limit of 0 LV% and a specification limit that put the upper limit to 0.5 LV%.

In case the QP-solving function in Matlab did not converge, that is the problem could be unbounded, infeasible or the algorithm could not find a feasible starting point. The steady state values were sent to HYSYS.Plant as fallback values.

For the GPC algorithm, parameters like cost and control horizons, forgetting factor and weighting matrices ***R*** and ***Q*** needed to be decided. The matrix ***R*** weighted the difference between outputs and set points, and was chosen to be equal the identity matrix. Matrix ***Q*** weighted the inputs, i.e. how much it actually cost to use the inputs. Matrix ***Q*** was equal to $\lambda$ times identity matrix where different values for $\lambda$ was tested. Cost and control horizons were changed in simulations to detect performance changes. The forgetting factor in the adapter was constant during all tests and equal to 0.9.

The simulation including the GPC algorithm was very time-consuming. The simulation itself with two distillation columns required lot of resources, but the link between Matlab and HYSYS.Plant was also time-consuming. The simulation in HYSYS.Plant stopped between each calculation of the control signal in Matlab, and the start-stop in HYSYS.Plant took time.

### *7.3.2 Implementation of the built-in MPC*

A built-in MPC in HYSYS.Plant was tested to benchmark the GPC controller. The process was exactly the same as when using the GPC controller. The only difference was that the slave controllers got its set point values from the built-in MPC instead of the help material streams. Information about the built-in MPC in HYSYS.Plant was received from Hyprotech Ltd. [8]. The built-in MPC in HYSYS.Plant was based on dynamic matrix control (DMC) algorithm. Several parameters needed to be specified in the built-in MPC. First, inputs and outputs needed to be determined. The inputs and outputs were the same as in the GPC. The built-in MPC was a master controller with flow controller on the ethane stream and propane stream and reboil ratio controller as slave controllers. Second, first order plus dead time (FOPDT) models described correlation between inputs and outputs. Step response data could also be used instead of FOPDT. Third, parameters needed to be set. These parameters were control interval, step response length, prediction horizon, control horizon, weighting matrices called Gamma_U and Gamma_Y and reference trajectory.

FOPDT models were developed by changing one of the set points to the slave controllers. The response in the controlled variables were observed when the others controllers were placed in manual. From the response data the process gain, time constant and the delay were detected for each model. Since the system is 3 x 3 (three inputs and three outputs), nine different models needed to be determined.

FOPDT models were implemented in the MPC controller. The calculated values for the process gain was small, overall values was $10^{-5}$. The gain was so small that when changing the set points in the MPC controller, it did not change the manipulated variables at all. It seemed like a round off problem in the built-in algorithm, where the changes in set points multiplied by process gain produced so small number that they were actually rounded off. The controlled variables where first changed from mole fraction and LV fraction to mole% and LV% to increase the gain, but the problem remained the same. Instead, the responses from the step response models where normalized to avoid these small process gain values.

It is preferable to have a constrained built-in MPC but this option could for some reason not be chosen. So the built-in MPC is unconstrained. If the control signals were outside their range, they will be constrained when they were sent to the slave controller. The MPC algorithm will not be aware of the constrains and therefore may use other past values than were actually implemented in the process. The built-in MPC would try to keep compositions specifications at their set points instead of a set point range.

The default parameters in the built-in MPC where used as starting points for the parameter selection. Two different control intervals was chosen and they were selected to be 30 seconds and 2 minutes. For each of these two cases FOPDT model were established. The FOPDT models changed slightly in the two cases. The built-in MPC was tested for different control interval and prediction horizons to detect differences in performance.

The simulation including the built-in MPC was less time-consuming than simulation with GPC. This is mainly because there existed link to Matlab, and the simulation did not need to start and stop between every control interval.

# 8. SIMULATION RESULTS

Two MPC controllers were tested in the same simulated process. The performances for the two controllers are described for different parameters.

## 8.1 Results using GPC including RLS adapter

The maximum feed disturbances in NGL were expected to be ±5 % of the ethane flow rate and ±1 % of the methane flow rate from, Halevi *et al.* [7]. Only a decrease in the ethane molar flow rate was tested due to lack of time. The feed rate and pressure remains constant. So when disturbances were put into the process, the GPC controller should change set points to the slave controllers to ensure that product spesifications are met.

The controllers have been tested by introducing decrease in the ethane flow rate corresponding to the largest expected variation. Simulations ran for 6 minutes before set point change was put into the process and then simulations ran for totally 120 minutes. The GPC controller was tested with parameters suggested from the literature among others. Some of the responses with decreasing ethane molar flow in the feed stream is displayed in appendix M. All the tested parameter sets is displayed in table 8.1.

**Table 8.1 Test parameters in the GPC algorithm for the simulation case**

| Case | Min.cost horizon, $N_1$ | Max.cost horizon, $N_2$ | Control horizon, *NU* | Control weight, λ | Execution interval [min] |
|---|---|---|---|---|---|
| 1 | 1 | 5 | 3 | 3 | 2 |
| 2 | 1 | 6 | 3 | 4 | 2 |
| 3 | 1 | 7 | 2 | 4 | 2 |
| 4 | 1 | 4 | 2 | 3 | 2 |
| 5 | 1 | 5 | 2 | 3 | 2 |
| 6 | 1 | 4 | 3 | 3 | 2 |
| 7 | 4 | 7 | 4 | 3 | 2 |
| 8 | 1 | 4 | 3 | 2 | 2 |
| 9 | 1 | 4 | 3 | 2 | 1 |

Common for all the responses were that more significant fluctuation in the set points sent to the slave controllers occurred when the cost horizon increased. The manipulated variables could go outside their range because of these fluctuations. When the manipulated variables were outside their range, it makes it difficult for the QP-algorithm to find a feasible starting point. When a feasible starting point was not found, the steady state values for the control signals were sent to the process instead. In a test the parameters were chosen as case 3 in table 8.1. The cost horizon was large and bigger than the degree of the *B* polynomial that was 4 in this case. The fluctuations in the set point sent to the slave controllers were big and the QP-problem could not find a feasible starting point several times during the simulation.

Two test were carried out with equal horizons but with different control signal weighting values. The parameters sets were equal to the cases 6 and 8 in table 8.1.The test with the smaller control signal weighting value had more fluctuations in the set points. The fluctuations were particularly visible in the set point to the reboil ratio.

To check how the control horizon affected the performance, two tests were carried out with parameters chosen as in case 1 and 5 in table 8.1. The parameters were equal in the two tests except for the control horizon which were unlike. The changes in set points during simulation have the same pattern, but the fluctuations is typically bigger for the case with longer control horizon.

There were also carried out a test where the parameters were selected after suggestions from Clarke and Mohtadi [2], equal case 7 in table 8.1. The control signal weighting was set to 3 instead of no penalty which they recommended for most cases. With these parameters the QP-algorithm had troubles to find a feasible starting point after the ethane molar flow rate was changed. Since the steady state values were used almost the all time, the ethane molar fraction decreased due to the reduction of the ethane molar flow rate.

There were only a couple of parameters set that actually managed to control the process in the simulated time. This happened with parameters selected like those in case 6 and 8 in table 8.1. In only these two test the QP-algorithm was able to solve the optimization problem during the whole simulated time. All others parameters set used the fallback of the steady state values one or several times.

All the data used in the GPC algorithm were normalized before used in calculations. To normalize data, a mean value is subtracted from the real value. Start mean values for the normalizing were chosen as start values from the simulations. If the process drifts from its starting steady states values, the mean values needs to be updated. The updating has a weighting factor which indicates how conservative the changing is suppose to be. The mean values show how the process drifts during the simulation, and an example is shown in appendix P. In all cases there was drifting in the process but in different degree for the different inputs and outputs.

## 8.2 Results using built-in MPC in HYSYS.Plant

The built-in MPC was tested with the same control interval as the GPC, that is 2 minutes, but also one test where carried out with 30 seconds control interval to display the importance of the control interval. The controller was tested with two different cost horizons. The other parameters were not changed during the tests. The parameters in the built-in MPC were selected as displayed in table 8.2 and each parameter set were tested in the process.

**Table 8.2 Parameters used in the built-in MPC in different tests.**

| Case | Control interval [minutes] | Step response length | Predictio n horizon | Control horizon | Gamma_ U | Gamma_ Y | Reference trajectory |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 50 | 25 | 2 | 1 | 1 | 1 |
| 2 | 2 | 50 | 5 | 2 | 1 | 1 | 1 |
| 3 | 0.5 | 50 | 5 | 2 | 1 | 1 | 1 |

The built-in MPC showed smooth and acceptable control for all three parameters set. The manipulated variables and the control signals are displayed in appendix N for all three cases.

The responses to the manipulated variables fluctuates clearly most in case 2 where the control interval is 2 minutes and the prediction horizon is 5. When the prediction horizon increases to 25, the responses to the manipulated variables smoothed down The use of control signal smoothed too, compare to the case where the prediction horizon is only 5. In the case where the control interval was decreased to 30 seconds, it seemed like the controller adjust the set point change in feed composition faster and the effect on the manipulated variables from the set point change decreases.

# 9. DISCUSSION MULTIVARIABLE CASE

## 9.1 The separation process

During the simulation the reboil ratio needed to be limited to maximum 0.95. If the reboil ratio was higher than 0.95 for a shorter period, the tray section dry hole pressure drop became too large. Also a reboil ratio larger than 0.95 is not preferable since very little bottom product will be produced and the liquid velocity in the column could be too low.

When developing the ARX-model from the PRBS-data, the degrees of *A* and *B* polynomials were limited to a value of 1 to 5. If the limitations were not done, the computations would be too big. The model fitting was also tried with a maximum value 10 for the degree of the model parameters, but this case seemed to be running in a forever loop. Therefore it is difficult know how big the effect of the limitations was. Particularly in a multivariable case the implementations were important to keep the programs as small and with as few computations as possible. The model fitting program could possibly be implemented more effectively so a bigger model degree could be selected for the *A* and the *B*.

The simulated process was simplified from original process description given from Halevi *et al*.[7]. Only two of the total five columns in the separations were simulated. The simplifications were also preheating of the feed, pressure and temperature adjustments of the product. The preheating of the feed was replaced by a higher temperature in the inlet feed and had the same temperature as the output temperature from the preheater. This should have no effect on the GPC controller performance. If the preheater had been included, local PID controller would have been needed to control the temperate. The omission of temperature and pressure adjustments on the products should have very little effect on the GPC controller since the products left the process and were not used in further simulations.

In the calculation of the tray diameters several assumptions were made. Among the assumptions were that the system was low or non-foaming and the weir-height was less than 15 percent of the plate spacing. Both of these assumptions were satisfied. The tray diameters in both columns were big. In column T-1 the tray diameter needed to be increased almost three times as much as the calculated value. It could be possible to run with a smaller diameter. The dynamic assistant in HYSYS.Plant suggests diameters that may be far too large from earlier experience. The diameter in the T-1 column has not been tried to be reduced. The diameter in T-2 needed to be increased because of tray section dry hole pressure became too large. This happened when the reboil ratio was very large during the simulations, like bigger than 0.96. The diameter was increased, but it may have not been necessary to increase the diameter after limitations of the maximum reboil ratio to 0.95. After the upper limit of the reboil ratio was reduced to 0.95, there was attempts to reduce the diameter because of lack of time. If the diameter had been reduced the model of the process would need to be updated with a new PRBS-test because of a change in the liquid and vapor flow

rates.

The noisy signal from the reboil ratio needed to be smoothed by a lag function before it could be used in control. This lead to a difference between the actual value of the reboil ratio and the smoothed signal when the set point to the reboil ratio changes. The smoothing generated sluggishness in the control of the reboil ratio since the input to the controller was the smoothed signal. The control managed to put the manipulated variable at its set point and the slowness did not appear to affect the performance significantly.

The way the calculated control signal went when transferring the value from Matlab to HYSYS.Plant was not refined. It was a very cumbersome method to transfer the value by go trough both a material stream and a spreadsheet in HYSYS.Plant. The control signals also changed in value on their way and this needed to be adjusted in Matlab. It might be possible to transfer the control signals with other commands in VBA than used to avoid this cumbersome path.

The simulation ran for 114 minutes after the change in ethane molar flow rate in the feed stream was introduced. The changes should have gone through the system throughout that time. Since the GPC contained a RLS that needed changes in the process to give reasonably results, the simulation could not be carried trough for a too long time without adding disturbances or set point changes. The process did not reached steady state during the simulation to avoid problems for the adapter.

The local PID controllers were not given that much attention after they were tuned. But that did not mean that they were not important. They needed to keep the manipulated variables that was not included in the cascade at theirs set point. Particularly set point changes for the reboil ratio controller affected the local PID controllers. The reboil ratio controller had most influence on the pressure in column T-2. The pressure again needed to be fairly stable to avoid disturbances in the recycle streams, i.e. temperature, flow and pressure in the streams. If the recycle streams changed too much, this affected the performance of the first column, which again gave feed streams to the T-2 column.

The interactions between the pressure controller and reboil ratio controller in column T-2 showed that they were not independent. Neither were the flow and temperature controllers on the liquid recycle stream independent. The controllers attached to one distillation column were very seldom independent of each other. These were still implemented as single-loops controllers with fairly good result. Multiple-loops are more complex and there exist a balance of the complexity and the improvement in performance by introducing multivariable control system. Others multiple-loops than the one in the cascade was not considered since the main focus were on the GPC controller. The local PID controllers worked well after tuning and it did not appear necessary to introduce multiple-loops other the one that already exist.

## 9.2 The GPC controller

The GPC algorithm optimize the three outputs ethane composition in ethane outlet, propane composition in propane outlet and propane composition in bottom outlet by changing the reboil ratio, flow rate ethane outlet and flow rate propane outlet. The optimization was only based on how to use the manipulated variables to obtain controlled variables within the constraints. No economy factors were included in the optimizing algorithm. Economy factors could be included by maximize market value of production subject to minimize energy usage. So the optimized set of inputs might not be used in a real world because the same set of inputs might not represent the optimized case from an economy point of view.

The GPC algorithm was tested with the parameters suggested by Clarke and Mohtadi [2]. This gave very poorly control. The fluctuations were large and after a while the QP-algorithm could not find a feasible starting point. Which parameters that mainly causes the fluctuations in this case was difficult to tell. First, the control signal weighting matrix was equal to a zero matrix, and this had not worked in other tests with other cost and control horizons because it causes too big fluctuations. Second, the minimum cost horizon was equal to 4 in this case, where the other test the minimum cost horizon is equal to 1. The maximum cost horizon is set to 7 in this case which may be too long and poorly prediction could occur. The fluctuation causes were probably both from a too long cost horizon and the lack of penalty on the control signal.

Default parameters value suggestions from Clarke *et al*. [3] were also tested. This gave a poor result too. The composition specifications were not maintained and the QP-algorithm could not find feasible point most of the simulated time. The control signals fluctuate a lot, and this could be caused of both a too long maximum cost horizon and too low control signal weighting matrix.

From Clarke *et al*. [4] the selection of maximum cost horizon was generally recommended to be chosen relatively large. This was not the case for the implemented GPC controller here. When the maximum cost horizon was selected bigger than the degree of the *B* polynomial, that is 4 in this case, the controller did not showed particularly good control performance.

There seemed to be an implementation problem in the QP-algorithm where the algorithm could not find a feasible starting point. The starting point was always the steady state values, and these points are feasible in the beginning of the simulation. The steady state values were normalized before they were used in the QP-algorithm and this might have changed the steady state values into an infeasible region because of drifting in the process and changes of the cost function due to the RLS.

When the cost horizon was chosen to be too long from the performance point of view, set points sent from GPC algorithm varied a lot and this of course affected the manipulated variables. The same pattern was discovered in the experimental phase when the GPC algorithm just included a single-input single-output case. This underlines the importance of

choosing the right cost horizon. With a cost horizon too long, the GPC algorithm had to predict too long in the future where the uncertainty is too big and this gave poorly predictions.

In the simulation the reference trajectory was identical with the set point. By smoothing approximation from the current value of the output towards the known reference, the system can react before the change has effectively been made and with that avoiding the effects of delay in process response. There were not executed set point changes in the GPC controller. The GPC controller did actually not had set points for the controlled variables, but set point ranges, where the only task for the GPC controller is to keep the controlled variables inside the constraints. For set point changes in the process an active use of reference trajectory could probably improve the controller.

The surface of the QP-problem could be developed to see whether there were other, more preferable, starting points in the algorithm. The surface change with time, because of the adaptive part that update the process model. Anyway, the searching area could be explored to detect if there were flat areas where the searching algorithm could have trouble or there are other sets of manipulated variable which were more preferable. Some model parameters developed during the simulation were used to explore the eigenvalues to the $H$ matrix. In all test cases the $H$ matrix had only positive eigenvalues, that is the optimization problem was positive definite at every test. In other words the QP-problem was strictly convex and had only one global solution.

The forgetting factor in the GPC algorithm was not changed during the test, but had a constant value of 0.9. From the experimentally part where a SISO GPC controller was tested, forgetting factor had very little effect on the controller. The same pattern was expected in the MIMO case and it should not produce a significant difference in the control performance.

Set points values were needed in the free response calculations in the GPC. Since outputs from the GPC algorithm had set point ranges instead of exact set point values, a pretended set point values had been chosen. The free response affects the $b$ and $f_0$ in the quadratic cost function displayed in equation (6.15). The scalar $f_0$ did not affect the selection of the manipulated variables, just the function value. The free response was included in the calculation of the $b$ vector, and $b$ affected the manipulated variables. So the pretended set points actually affected the optimization of the cost function. The performance of the GPC algorithm should not be significantly changed by the selection of the pretended set point variables, as long as the selections were done wisely. The selections of the set points should be of course in the set point range and not on the limits of set point range.

In some cases the compositions specifications were maintained and the QP-algorithm did not need any fallback during the whole simulation. In this cases the reboil ratio was decreasing slowly and this had biggest effect on the propane purity in propane stream that also decreased slowly. In one of the cases the simulation was carried out for another 80 minutes to discover what happened. The reboil ratio decreased and the propane purity also decreased until the specification was not maintained any more. Some time after the specification for the propane

stream was violated, the QP-algorithm could not find a feasible starting point and the steady state values were sent to the process instead.

The GPC algorithm was executed every second minute. The sampling interval was chosen so the slave controller managed to change the set point and the manipulated variables will be affected. This may have been a too long interval. Two different tests with the same parameters but with an execution interval of one and two minutes respectively, were carried out. The algorithm started with the same ARX-model but the RLS updates the model in both cases. The purity of the product streams, ethane and propane, were lower in the test with sampling interval 1 minute compare to 2 minutes sampling interval while the bottom product had a better quality, that is less propane contents. The inputs react earlier to the set point change in the case with the shortest sampling interval but did not end at the same place because the updated model differs in the two cases. One disadvantage for the test with the shortest sampling interval is that the PRBS test is carried out with a sampling interval of 2 minutes instead of 1 minute. The RLS should be able to update the model, but it was difficult to know when the ARX-model actually fitted the process with a shorter sampling interval. From the two test with different sampling interval, the performance had the mostly the same pattern but the end points of the simulation were not equal because of different updating models. Apart from the disadvantages of the model, the performance did not seem to improve with shorter sampling interval. The test indicates that a sampling interval of two minutes was not too long.

## 9.3 The built-in MPC Controller

For the built-in MPC in HYSYS.Plant, FOPDT models were determined by step responses. For the step responses it was difficult to get a perfect step input. But the ramp constant was small compared to the process time constant, so a good approximation to a step should have been obtained. Since the smallest time constant in the FOPDT models were 2 minutes or bigger, should this not have affected the model.

When the step responses were carried out, the controller where its set point changed was in auto mode, while the other controllers in were in manual mode. This were done due to controllers were not suppose to interact with each other. When executing the set point changes, the flows in the cascade where constant when these controllers were in manual mode. The reboil ratio changed during the step response tests when the controller where in manual mode. Since the reboil ratio was most important to the compositions in products and bottom stream this affected the step response. This may have affected the first order models used in the built-in MPC.

The process was approximated to a first order model. True processes are very seldom ideal first order. This gave an error in the predicted outputs. When the process drifted, new step response models should be implemented. The built-in MPC did not contain an adaptive part, so the model could not be updated during simulation. Even though the built-in MPC did not

have an adapter and the prediction was based on FOPDT models, the controller showed good prediction.

How the performance of the built-in MPC was affected by the choice of parameters was not studied in detail. Only the control interval and prediction horizon were tested with more than one value. The difference in prediction horizon was large between the two tests; still the performance did not become unacceptable. It seemed from the test that the controller was not very sensitive of the choice of parameters. There is difference in performance of course, but non of the test with built-in MPC gave unacceptable control.

## 9.4 Comparison of the GPC Controller and the built-in MPC Controller

The GPC based its predictions on an ARX-model and the built-in MPC used FOPDT models to predict the outputs. It was expected that the GPC was able to predict better for processes that have a more complex dynamics since the ARX-model is able to include more complex dynamics than a FOPDT model. It is limited how many processes the dynamics can be simplified with a FOPDT model. Anyway, the built-in MPC showed better performance and was able to control the process for all parameters set that where tested.

The built-in MPC was very simple to use. It needed only number of inputs times number of outputs FOPDT models and a control interval. FOPDT models were easy to develop and did not require any implementation compared to an ARX-model, which needed some implementation in Matlab to fit PRBS-responses to the model. In the built-in MPC, parameters like cost and control horizons, weighting matrices and reference trajectory could be specified but it was not necessary since the controller had default values. This made the built-in MPC very attractive to use because it was so simple to make it work. The parameters selections were not that critical that it appeared to be in the GPC controller.

The GPC controller needed much more effort to work. The implementation was time consuming and quite difficult with a lot for matrix calculations. The parameters needed much more attention than in the built-in MPC. The parameters selections were critical for the performance of the controller. To obtain the optimal set of parameters in the GPC algorithm, several tests was usually needed since there was no standard recipe that appear to work for this case to obtain these parameters. Suggestions from the literature had not been successful in all cases, so the parameters must be found by try and fail.

## 9.5 Comparison using GPC in experimental phase and simulation phase

The cost horizon seemed to be a very important parameter for the performance of the GPC

controller, both in the SISO experimental phase and the MIMO simulation phase. In both cases the best performance occurred when the maximum cost horizon had the same value as the degree of the *B* polynomial.

One advantage of the GPC algorithm was that when the SISO case was established, it just an extension in dimension to make the GPC work in a MIMO case. The algorithm was quite similar and based on the same equations. Also adding constrains to the optimization problem was quite easy when the cost function was written as a quadratic function.

Both the SISO GPC and the MIMO GPC handled constraints in their algorithms. When constraints were added to the optimization problem, the solution was not analytical as it was without constraints. The two algorithms used two different optimization functions in Matlab, *fmincon* in the SISO case and *quadprog* in the MIMO case. The difference in the two functions were that *fmincon* found the minimum of a constrained nonlinear multivariable function where the function returns a scalar, while the *quadprog* solved a defined QP-problem with a quadratic function and linear inequalities and linear equalities. The problem in the SISO case could also be solved as a QP-problem instead of using *fmincon*. It was expected that both algorithms finds the minima of the function subject to the constraints, but the *fmincon* search method might be more extensive and therefore more time-consuming since it supposed to search in nonlinear multivariable functions compare to quadratic functions. The time-consumption in data transfer and calculation is an issue in the experimental phase since it limits the sampling interval. The reduction of time-consumption by using *quadprog* instead of *fmincon* is little compare to the time-consume of the DDE link between Labview and Matlab.

The SISO GPC worked well and it was expected that the GPC controller had better performance than the PID controller. Some of the suggestions from literature could be used as guidelines for selections of the parameters in the GPC algorithm. An exception was the control signal weighting which could not be zero to perform good control.

The MIMO GPC did not work as expected compare to the built-in MPC. Since the GPC was based on ARX model for its predictions while the built-in MPC made its prediction based on FOPDT model. The ARX model should manage to include the dynamics in the process better than the FOPDT model, therefore was a poorer performance of the GPC compare to the built-in MPC a surprise. Mostly of the parameter suggestions from literature resulted in poor control by the MIMO GPC. The parameters suggested in literature were widely tried in several different types of processes, among other things unstable and non-minimum phase plants. So a question is of course why the suggested parameters could not be used in control coupled distillation columns.

In both the SISO and the MIMO case the GPC was a master controller in a cascade. The calculated outputs from the SISO GPC controller changed smoothly in set point changes, and this gave smooth change in the manipulated variable. In the MIMO case the set point change might suddenly jumped to another value. The same smoothness did not exist in the controller for the multivariable case. Why the performance characterization changes that much when

the number of variables was extended was not fully understood. The processes were different in the two cases, but this should not change the performance characterization so drastically.

The GPC performance in the simulation case was surprising based on several observations as mentioned above. First, the GPC controller did not showed good control compare to the built-in MPC, and second, the difference between the performance of the SISO GPC and the MIMO GPC. This could indicate that the GPC controller did not work as it should be and there could be something wrong with the implementation. The GPC controller was implemented in a quite large Matlab-file and this program called two large functions in addition. This code could contain errors; among other things variables might not be assigned the right values. There was also a link between Matlab and HYSYS.Plant trough VBA that could contain errors. The errors were difficult to find and could not be detected in the bounded time. The algorithms needs throughout debugging to detect possible errors.

# 10. CONCLUSION

The Generalized Predictive Control (GPC) algorithm was tested in both an experimental SISO case and a simulated MIMO case.

The experimental case was carried out on a heat exchanger connected to a computer. The heat exchanger was a shell-and-tube with one hot water and one cold water flow. The control was a single-input single output (SISO) where the goal was to control temperature of the cold water outlet with the hot water flow as manipulated variable. The control structure contained a master SISO GPC controller and a PI slave controller. Labview was used to run the experiment. Matlab was used to calculate the prediction of the control signal from GPC and the model parameters from RLS. Matlab was connected to Labview through dynamic data exchange (DDE).

The MIMO case was conducted using a HYSYS.Plant-simulated distillation train. The simulated process was a part of separation of natural gas liquid (NGL). The process contained two coupled distillations columns where the first column had neither reboiler nor condensator. The products were ethane and propane. There was also and bottom stream form the second column that went on to further separation. The control strategy contained local PID controllers and a multivariable GPC as a master controller in a cascade. The GPC controller algorithm was implemented in Matlab, and HYSYS.Plant was linked together with Matlab via Visual Basic for Applications (VBA).

In both cases pseudo-random-binary-signal (PRBS) tests were carried out and fitted to autoregressive with input (ARX) models. To update the models an adapter based on the recursive least squares (RLS) method was used.

The constrained SISO GPC including RLS showed good control with relatively fast rise-time, almost no overshoot and smooth actions on the control valve for several parameters set. The SISO GPC was also compared with a PID controller. The PID controller was not effective in smoothing out oscillations. The oscillations could be tuned down a little but the the settling time was undesirable long. The oscillations in the PID controller may be caused from interactions with the slave controller. The PID showed poorer control compare to the GPC mainly because of the oscillations.

The constrained multivariable GPC tried to control ethane purity in ethane stream, propane purity in propane stream and limit propane in bottom stream from the second column in the separation. The control signal were ethane flow rate, propane flow rate and reboil ratio. The GPC performance was evaluated with various parameters by decreasing ethane molar flow rate in the feed.

The MIMO GPC showed poorer control than expected and it was sensitive to the selection of parameters. The GPC controller was compared with the MPC controller which is built into HYSYS.Plant. The built-in MPC was based on first order plus dead time (FOPDT) model

and showed very smooth control and manage to maintain the specifications during the whole simulation.

The built-in MPC needed a limited amount of effort to work. The FOPDT models were easy to develop and the controller was not that sensitive to the selected parameters compare to the GPC controller. Therefore the built-in MPC was preferred to the GPC controller.

In both the SISO case and the MIMO case the cost horizons and control signal weight were important for the GPC performance. The control horizon had minor effect on the performance. The forgetting factor in the RLS made no difference in the GPC performance. In both cases the suggestions from the literature could partly be used as guidelines for parameters selections in the GPC algorithm.

# REFERENCES


[1] Camacho, E.F and Bordons C., *Model Predictive Control*, Springer, 1999.

[2] Clarke, D.W and Mohtadi, C., *Properties of Generalized Predictive Control*, Automatica, 25(6), 859-875, 1989.

[3] Clarke, D.W, Mohtadi, C. and Tuffs, P.S, *Generalized Predictive Control - Part I. The Basic Algorithm*, Automatica, 23(2), 137-148, 1987.

[4] Clarke, D.W, Mohtadi, C. and Tuffs, P.S, *Generalized Predictive Control - Part II. Extensions and Interpretations*, Automatica, 23(2), 149-160, 1987.

[5] Clough, D.E., *A derivation of Recursive Least Squares Parameter Estimation*, Handout.

[6] Clough, D.E., *Time Series Analysis*, Handout for MCEN 5126 Applied Statistics for the Manufacturing and Process Industries, 1999.

[7] Halevi, B., Quarles, J. and Wilson, M., *NGL Separation by Distributed Distillation*, Senior Design Project, University of Colorado, 2000.

[8] Hyprotech Ltd.

[9] Ljung, L., *System Identification: Theory for the user*, Prentience-Hall Inc., 1987.

[10] Løvland, J., *Kompendium i separasjonsteknikk*, Institutt for kjemiteknikk NTNU, 1997.

[11] Perry, R.H and Chilton C.H, *Chemical Engineers' Handbook*, Fifth edition, McGraw-Hill Inc., 1973.

[12] Seborg, D.E., Edgar, T.F. and Mellichamp, D.A, *Process Dynamics and Control*, John Wiley & Sons, Inc., 1989.

[13] Skogestad, S., *Tillegg til fag 52041 Prosessregulering*, NTNU, 1999.

[14] Åstrøm, K.J. and Wittenmark, B., *Computer Controlled System. Theory and Design*, Prenticene-Hall, 1984.

# GLOSSARY

### Notation

| | |
|---|---|
| ***A*** | Bold, italic and upper case letters denote matrices |
| *A* | Italic letters denotes vectors or scalars |

### Symbols

| | |
|---|---|
| $q^{-1}$ | backward shift operator |
| $q$ | forward shift operator |
| $A(q^{-1})$ | left polynomial in model |
| $B(q^{-1})$ | right polynomial in model |
| $C(q^{-1})$ | noise polynomial in model |
| $d$ | dead-time |
| $y(t)$ | output variables at instant $t$ |
| $u(t)$ | input variables at instant $t$ |
| $e(t)$ | discrete white noise with zero mean |
| $na$ | model order of $A$ polynomial |
| $nb$ | model order of $B$ polynomial |
| $\Delta$ | $1 - q^{-1}$ |
| $E_j$ | polynomial in Diophantine equation |
| $F_j$ | polynomial in Diophantine equation |
| $G_j$ | $E_jB$ |
| $f$ | free response |
| $R$ | $E_{j+1}$ |
| $S$ | $F_{j+1}$ |
| $J$ | cost function |

| | |
|---|---|
| $w$ | future setpoint |
| $\delta$ | weighting factor for difference between prediction and future setpoint |
| $\lambda$ | weighting factor for control signal |
| $\hat{y}(t+j\mid t)$ | expected value of *y(t+j)* with available information at instant *t* |
| $N_1$ | minimum cost horizon |
| $N_2$ | maximum cost horizon |
| $N_u$ | cost horizon |
| $N$ | number of pints of prediction horizon |
| $K_c$ | process gain |
| $\tau_I$ | integral time |
| $b$ | bias |
| $\beta$ | vector/matrix with model coefficient |
| $\varphi$ | matrix with past inputs and outputs |
| $V$ | loss function |
| $\psi$ | vector/matrix of past values of inputs and outputs |
| $\alpha$ | variance of errors |
| $c$ | scalar that contains a large number |
| $I$ | identity matrix, subscript indicates the dimension |
| $\tilde{A}$ | polynomial *A* multiplied by $\Delta$ |
| $\|\|v\|\|_Q^2$ | $v^TQv$ |

# APPENDIX

# A LABVIEW DIAGRAM FROM CALIBRATION

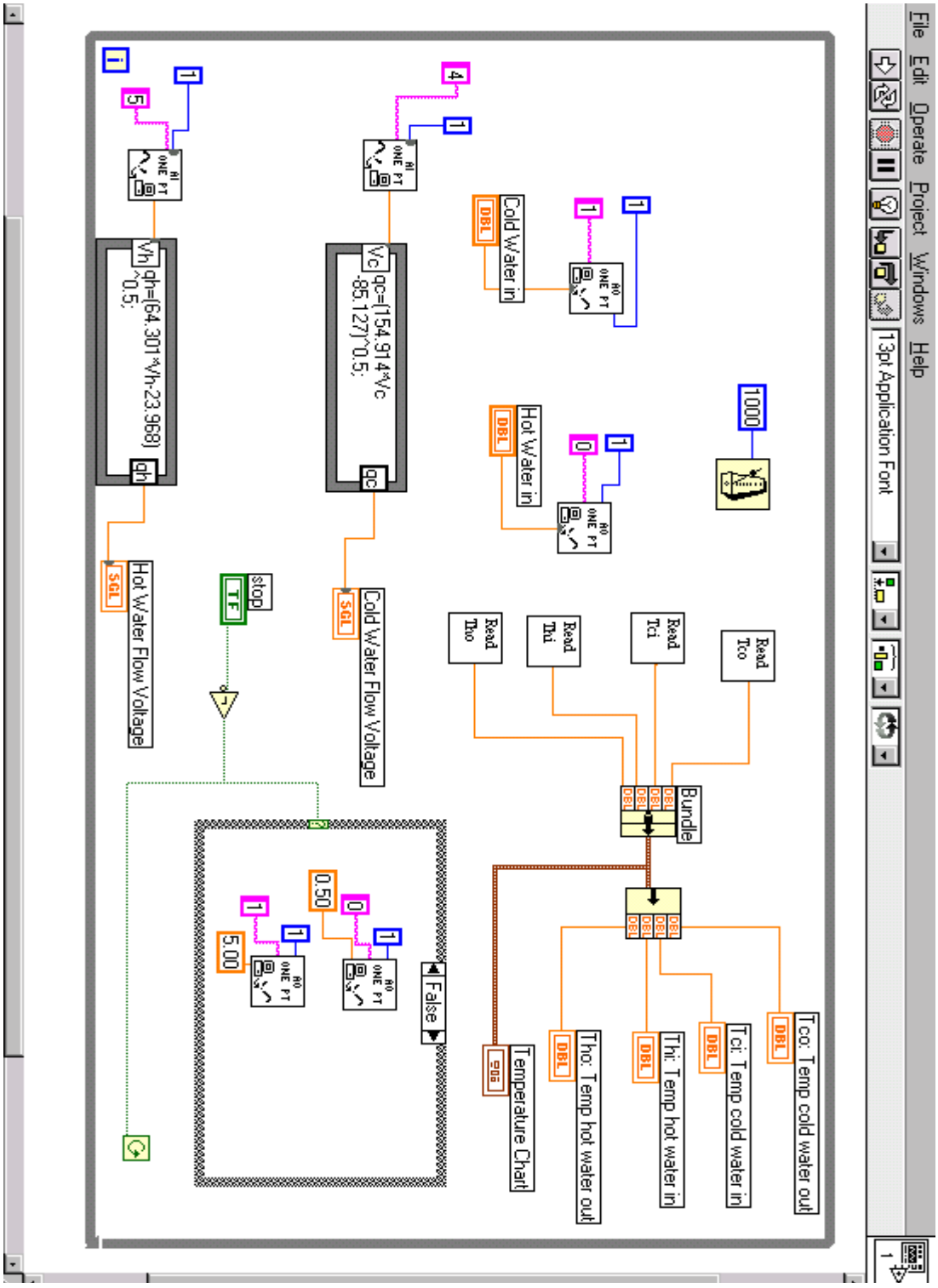



**Figure A.1 Labview diagram of the calibration file**

**Figure A.2 SubVI "Read Tci" in the calibration file which read the temperature from termocouple at cold water inlet.**

All four subVI that read temperature are almost equal. The only difference are the scaling from voltage to $^{o}$C.

# B THEORY BASIS FOR CALIBRATION

The flow measurement signals from heat-exchanger are given in voltage, and this has to be converted to a more practical unit. The flow through a valve is proportional to the square root of the pressure difference over the valve,

$$q = k\sqrt{\Delta P} \tag{B-1}$$

where $q$ is the flow, $\Delta P$ is the differential pressure and $k$ is a proportional constant. The output signals from heat exchanger are in voltage, and the voltage can be expressed as a linear function of the pressure difference,

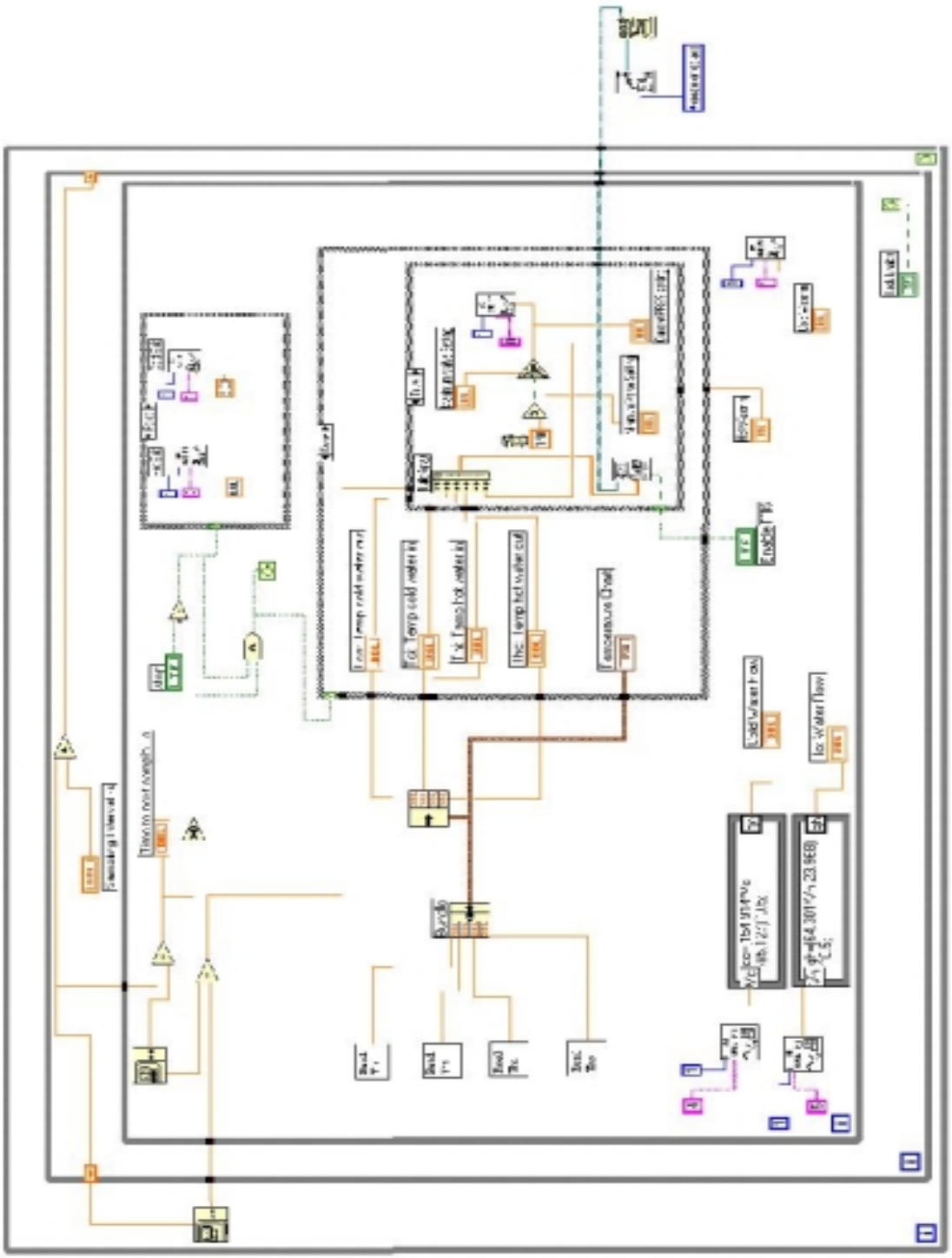
$$V = a + b\Delta P \tag{B-2}$$

where $V$ is the voltage $a$ and $b$ are constants. If equation (B-2) is substituted into equation (B-1), the flow can be expressed as

$$q = k'\sqrt{V - a} \tag{B-3}$$

where $k'$ is a constant. With a series of measurement of the flow and the voltage, the relation between flow and voltage can be found based on equation (B-3).

# C LABVIEW DIAGRAM FOR THE PRBS-TESTS



**Figur C.1 Labview diagram of PRBS**

# D PRBS-EXAMPLE. MODEL DEVELOPMENT FOR THE HEAT EXCHANGER

PRBS-test is executed to develop a model which the GPC controller is based on. The binary input to the PRBS-test is hot water flow rate. Cold water flow rate is considered as disturbances. Parameters for the PRBS-test is described in table D.1.

**Table D.1 PRBS-test parameters. Hot water flow is setpoint to slave controller, cold water flow is gauge-value**

| | |
|---|---|
| Hot water flow rate | 10 - 14 l/min |
| Cold water flow rate | 3.88 (gauge value) |
| Sampling interval | 3 seconds |

The PRBS-test can be displayed in figure D.1. Only 150 seconds of the 600 seconds the test was executed is displayed.

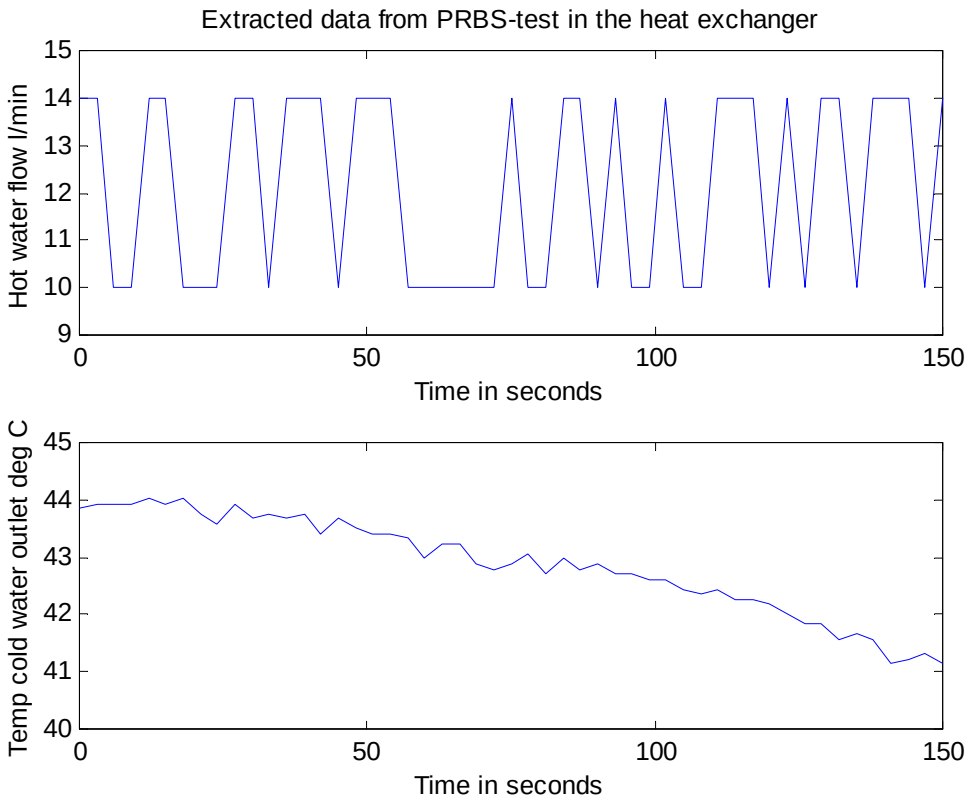



**Figure D.1 Extracted PRBS-data from heat exchanger with control signal in the upper graph and manipulated variable lower graph**

After PRBS-test is executed, the data is fitted to an ARX-model by Matlab. Results from the fitting by Matlab are given in table D.1.

**Table D.1 Data from Matlab when fitting ARX-model from PRBS-data**

| | |
|---|---|
| A | [1.0000 -0.5186 -0.2520 -0.2040] |
| Standard deviation of A | [0 0.0679 0.0750 0.0653] |
| B | [0 0 0.0170 0.0190 0.0244 0.0012 0.0061] |
| Standard deviation of B | [0 0 0.0053 0.0054 0.0055 0.0056 0.0055] |
| na | 3 |
| nb | 5 |
| nk | 2 |
| FPE | 0.0248 |

The ARX-model polynomials will then have the form

$$A(q^{-1}) = 1 - 0.517q^{-1} - 0.252q^{-2} - 0.204q^{-3}$$
$$B(q^{-1}) = 0.017q^{-2} + 0.019q^{-3} + 0.024q^{-4} + 0.001q^{-5} + 0.006q^{-6}$$

where the degree of *A* polynomial is 3 and the degree of the *B* polynomial is 6.

# E LABVIEW DIAGRAM - PROCESS INCLUDING GPC

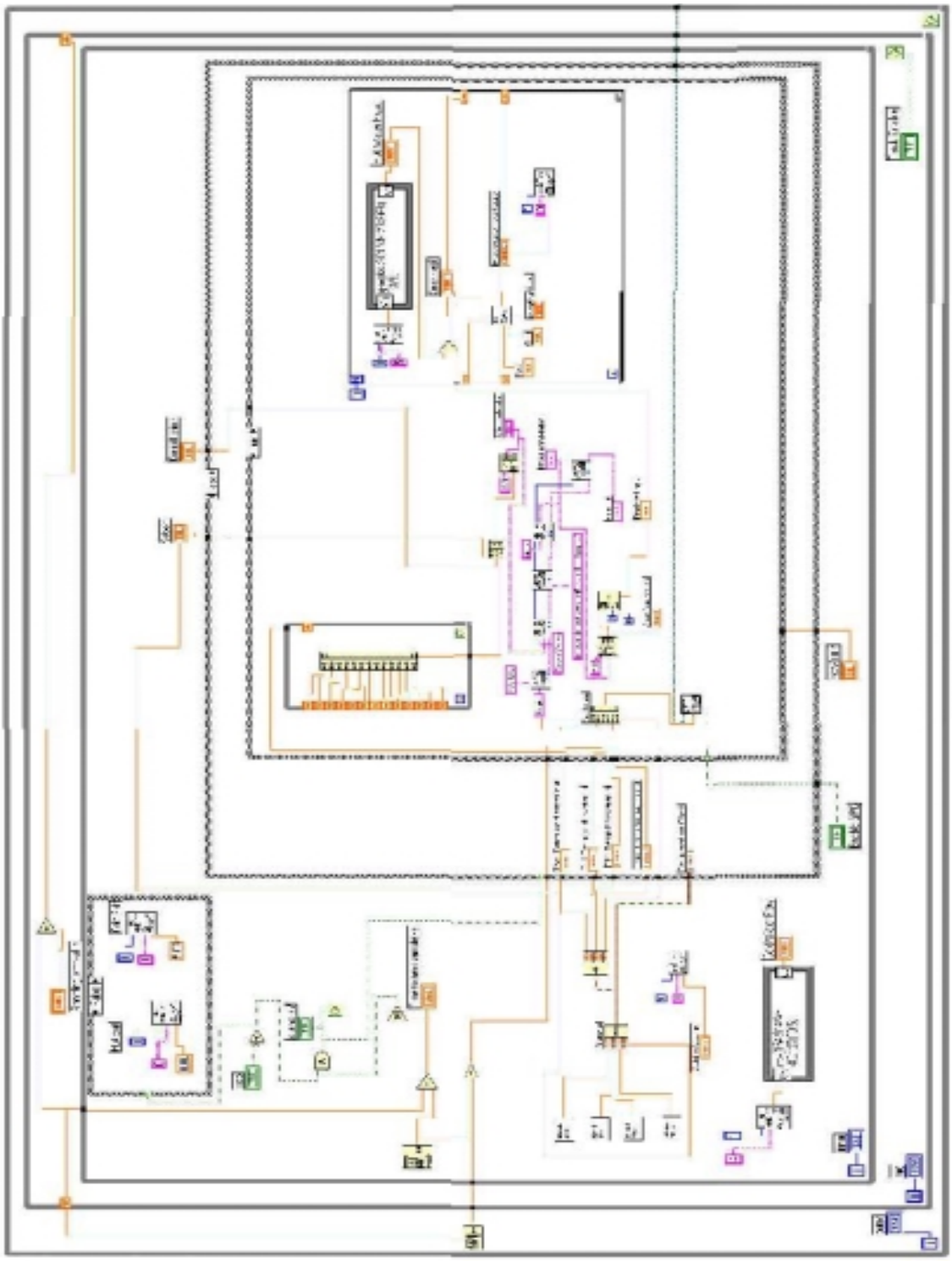


**Figure E.1 Labview diagram othe GPC.**

# F MATLAB ALGORITHMS IN SINGLEVARIABLE CASE

## F.1 Matlab code for developing ARX-models

```
%Present an ARX-model from experimental data collected by
%Author: E.M Bergheim   Fall-2000

load c:\MATLABR11\work\Diplom\prbsinclslave.txt
%reading output and input from a loaded file,
%the name is the same as the name to the loaded file
y=prbsinclslave(:,2); %select 2nd coloumn which contains temp cold water
outlet
u=prbsinclslave(:,6); %select 2nd coloumn which contains the control sig-
nal
z=[y u]; %set the output and input in a vector

%constants
namin=1; %minimum degree of A polynomial
namax=5; %maximum degree of A polynomial
nbmin=1; %minimum degree of B polynomial
nbmax=5; %maximum degree of B polynomial
nkmin=1; %%minimum degree of delay
nkmax=5; %maximum degree of delay

%Initialized model
na=1;  %initialize the degree of A polynomial
nb=1;  %initialize the degree of A polynomial
nk=1;  %initialize the degree of delay
nn=[na nb nk]; %put the degrees in a vector
th=arx(z,nn); %develop the arx-model
FPEint=th(2,1);%initialize Aikake's Final Prediction Error.
%See help th for structure of the th matrix

%Check all the combinations of model structure that gives the best
describtion of the process
for na=namin:1:namax
   for nb=nbmin:1:nbmax
      for nk=nkmin:1:nkmax
         nn=[na nb nk];
         th=arx(z,nn);%develop the arx-model
         FPE=th(2,1); %Displays the FPE in the th matrix
          if FPEint>FPE %checks the FPE, less value for FPE gives better
model
            FPEint=FPE; %update the FPE value if its less than the past
value
```

```
            naopt=na; %save the optimal degree of A
            nbopt=nb; %save the optimal degree of B
            nkopt=nk; %save the optimal degree of delay
            optparameters=[naopt nbopt nkopt FPE];
           end
       end
    end
end
%Displays the optimal modelparameters na,nb,nk and FPE and the modelcoeff.
A and B
display('The optimal model paramters')
th=arx(z,[naopt nbopt nkopt]); %Find the arx-model with the optimal
paramters
present(th) %displays the paramters
display(optparameters)
loss_funciton=th(1,1)  %See help theta for exact structure of th
e=resid(z,th); %compute the residuals and display correlation graphs
stdev_e=std(e) %finds the standard devation of the error
var_e=var(e) %finds the variance of the error

%Transform the arx model to a transfer function
[num_tf,den_tf]=th2tf(th)
```

## F.2 Matlab code GPC including RLS

```
function [upast,A,B]=costandcontrol(DatatoMatlab,upast)
%Function called by Labview. Needs past input and output from the process
%from Labview, in addition to control horizon, maximum cost horizon and
ref.trajectory
%Initialize before startup upast=ones(N,1) in the command window
%where N is the cost horizon
%Function that minimize the cost function subject to constrains
%on the control signal. The control signal is the setpoint to the slave-
controller
% Author: E.M Bergheim   Fall-2000
global H b fmark w A B ypast uopt yrls urls beta P K

%Startvalues for the model. Updates by RLS
A=[1  -0.4548 0.0218 -0.5309];
B=[0 0 -0.0235 0.0552 0.0255 0.0414 -0.0058];

%Constants
lamda=0.3; %control-weighting sequense
N1=2;   %minimum costing horizon
```

```
%Data from Labview
NU=DatatoMatlab(3); %control horizon
N2=DatatoMatlab(2); %maximum costing horizon
N=N2-N1+1;  %the cost horizon
ypast=DatatoMatlab(4:16)'; %past output
w=DatatoMatlab(1); %referense trajectory(=setpoint)

%Calculate delta u = u(t)-u(t-1). Used in free response for cost function
with
%delta u as variable. This free response is used for calculated the free
response
%for cost function with u as variable.
for t=1:size(upast)-1
   delta_u(t)=upast(t)-upast(t+1);
end
delta_u=delta_u';  %transpose the vector to get a column vector

%Calculate the Diophantine eq recursivly. Calling the function Diophantine
[E_rec,F_rec]=diophantine(A,B,N);

%Generate the G1,G2...GN2 polynominal. Used in the free response calc.
for k=1:N
   g(k,:)=(conv(E_rec(k,:),B)); %Multiplies each row of E matrix from
Diophantine with B
end

%Generate the G matrix
Gprime=diag(g); %Pick out the diagonal of the g matrix
G=zeros(N);
for j=1:N
   G(:,j)=[zeros(j-1,1);Gprime(1:N-j+1)];
end

G=G(1:N,1:NU); %Set the size of G matrix

%Compute the G1,G2,...GN2 for the free response
gfree=triu(g,1);  %pick out the lower triangual matrix of g
m=size(gfree,2);
for p=1:N
   gtemp(p,:)=[gfree(p,p:m) zeros(1,p)];
end
gdiff=gtemp(:,1:size(gtemp,2)-1);
gdiff=gdiff(:,1:N);

%The free response
f=[gdiff(:,2:size(gdiff,2))*delta_u(1:size(gdiff,2)-
1)+F_rec*ypast(1:size(F_rec,2))];
```

```
%Making f and w of the same dimention. Labview poke often more data than
%necessary, depending on the cost and control horizons.
w=w*ones(size(f,1),1);

%Compute variables for the cost function for u, not deltau
D1=diag(ones(NU,1));
D2=diag(-ones(NU-1,1),-1);
D=D1+D2; %D is a matrix with element 1 on diag and -1 on subdiagonal
Gmark=G*D;
f1=[upast(1);zeros(NU-1,1)];
f2=f-G(:,1)*f1(1);
H=2*(Gmark'*Gmark+lamda*D'*D); %Matrix in the cost function
b=2*((f2-w)'*Gmark-lamda*f1'*D); %Vector in the cost function
fmark=(f2-w)'*(f2-w)+lamda*f1'*f1; %the new free response when the cost
fun is based on u

%Minimizing the cost function
ustart=7*ones(NU,1);  %Startvalue for the optimalization
umin=0.5*ones(NU,1);  %Min value for the output for the optimalization
umax=18*ones(NU,1); %Max calue for the output for the optimalization
[u,fval]=fmincon('costfuntotal',ustart,[],[],[],[],umin,umax);
%fmincon finds the minimumsvalue from the cost function. Cost function is
in the function costfuntotal
uopt=u(1); %not a necessary statement when noise is not added. If adding
noise, add here.
upast=[uopt; upast(1:size(upast,1)-1)]; %updates the past values for the
control signal
%upast is only a value Matlab work with and is not send between Matlab and
Labview.

%The RLS part
rlsdata=[ypast(1),uopt]; %call the adapterfunction with the latest values
of in-and output
[A,B]=adaptinGPC(rlsdata);%Calles the RLS function adaptinGPC that
updates A and B polynomial

%function called by costandcontrol.m.
%Stored in a own file called costfun
function J=costfun(u)
%Gives the cost function to fmincon function in costandcontrol
global H b fmark w A B ypast uopt yrls urls beta P K

J=0.5*u'*H*u+b*u+fmark; %the cost function based on u (not delta u)
```

## F.3 Matlab code for Recursion of the Diophantine Equation

```
function [E_rec,F_rec]= diophantine(A,B,N)
%Solves the Diophantine equation recursively, A and B is modelparameters,
N2 is the cost horizon
% Author: E.M Bergheim fall-2000

%Initializing E
E(1)=1;

%Initializing F
Atilda=[A 0]-[0 A]; %Atilda=A(1-q^-1)
v=zeros(size(Atilda));
v(1)=1;
Aprime=v-Atilda; %Aprime= 1-Atilda
F=Aprime(2:size(Aprime,2));   %F=q(1-Atilda)
m=size(Aprime,2)-size(F,2);
F=[F zeros(1,m)];  %Size F needs to be equal Atilda= size Aprime

%Saving the E & F for each loop, including the initial values
E_rec=zeros(N); %Initialize the E that is recorded
E_rec(1,1)=E;
F_rec(1,:)=F; %Initilize F that is recorded

%Recursion loop for Diophantine
for j=2:N
   r(j)=F(1);
   R=[E,r(j)]; %updates the new R value, which takes the first element in
F and add in
   for i=1:size(F,2)-1 %updates the S polynomial
      S(i)=F(i+1)-Atilda(i+1)*r(j);
   end
   r=[R zeros(1,N-j)]; % Set R which is E(j+1) so R represent E(j+2) in
next loop
   E_rec(j,:)=r;
   E=R;
   F=[S zeros(1,size(Atilda,2)-size(S,2))]; %Set S which is F(j+1) so S
represent F(j+2) in next loop

   F_rec(j,:)=F;
end

F_rec=F_rec(:,1:size(F_rec,2)-1);   %Erase the last column.In the calcula-
tion
%the size of F needs to equal the size of Atilda. This adds a column.
```

```
%Check that 1=E*Atilda-q^-1F
%Sum=conv(E,Atilda);
%F=F(1:size(F,2)-1);
%F=[zeros(1,(size(Sum,2)-size(F,2))) F];
%eq=Sum+F
```

## F.4 Matlab code for RLS

```
function [A,B]=adaptinGPC(rlsdata)
%Adaptiver based on RLS method
%rlsdata is data from the GPC function
% Author: E.M Bergheim   Fall-2000

global H b fmark w A B ypast uopt yrls urls beta P K
%H,b and fmark is variables in the costfuntion
%w is the setpoint trajectory
%A and B is modelparamters
%ypast is the past mesured values
%uopt is the optimal control signal found by the GPC algorithm
%yrls and urls is vectors the past vector is made from
%beta is the parameter estimate in a column vector, P is the covar. matrix
%and K is the gain vector

%Save the past data
yrls=[ypast(1);yrls]; %update the past measurement vector
urls=[uopt;urls]; %update the past control signals vector
i=size(yrls,1); %count how many data the rls algorithm has

%Choose values for the degree of polynomnials and the deadtime
n=3;  %Degree of the A polynominal
m=5;  %Degree of the B polynominal
d=2;  %Deadtime

%Test if there is enough data to run RLS
if i>=m+d+1
   %Constants
   c=1000; %for initializing of the P matrix
   gama=0.9;%forgetting factor
   alfa=0.0224; %variance to the error, found from PRBS-data

   %Initializing of the system
   beta_init=zeros(n+m,1); %initialize the beta-vector
   I=eye(m+n);%identity matrix
   Pinit=c*I; %initialize the P matrix
   ksi=[yrls(2:n+1);urls(d+2:m+d+1)]; %vector with past data
```

```
   %Compute model prediction
   if i==m+d+1
      ypred=ksi'*beta_init; %if the loop runs the first time, uses the
initlized beta
   else
      ypred=ksi'*beta; %if the loop runs 2nd time or more, uses the beta
from last run
   end

   %Update gain vector
   if i==d+m+1
      K=(Pinit*ksi)./(alfa*gama+ksi'*Pinit*ksi);
   else
      K=(P*ksi)./(alfa*gama+ksi'*P*ksi);
   end

   %Update parameter estimates
   if i==d+m+1
      beta=beta_init+K*(yrls(1)-ypred);
      A=[1;-beta(1:n)]'; %update modelparameter A. The first coeff is
always 1
     B=[zeros(d,1);-beta(n+1:m+n)]';%updata B. -beta comes from different
definition of A and B in RLS and GPC
   else
      beta=beta+K*(yrls(1)-ypred);
      A=[1;-beta(1:n)]';
      B=[zeros(d,1);-beta(n+1:m+n)]';
   end

   %Compute covariance matrix for next iteration
   if i==d+m+1
      P=(1/gama)*(I-K*ksi')*Pinit;
   else
      P=(1/gama)*(I-K*ksi')*P;
   end
else %else if i>m+d+1
   %if not enough data, the PRBS model is used
   A=[1  -0.4548 0.0218 -0.5309]; %A given from the PRBS data
   B=[0 0 -0.0235 0.0552 0.0255 0.0414 -0.0058]; %B given from the PRBS
data
end %end if i>m+d+1
```

# G GPC TEST INCLUDING RLS FOR SEVERAL SETTINGS

## G.1 Comparison of setpoint responses with different control horizons

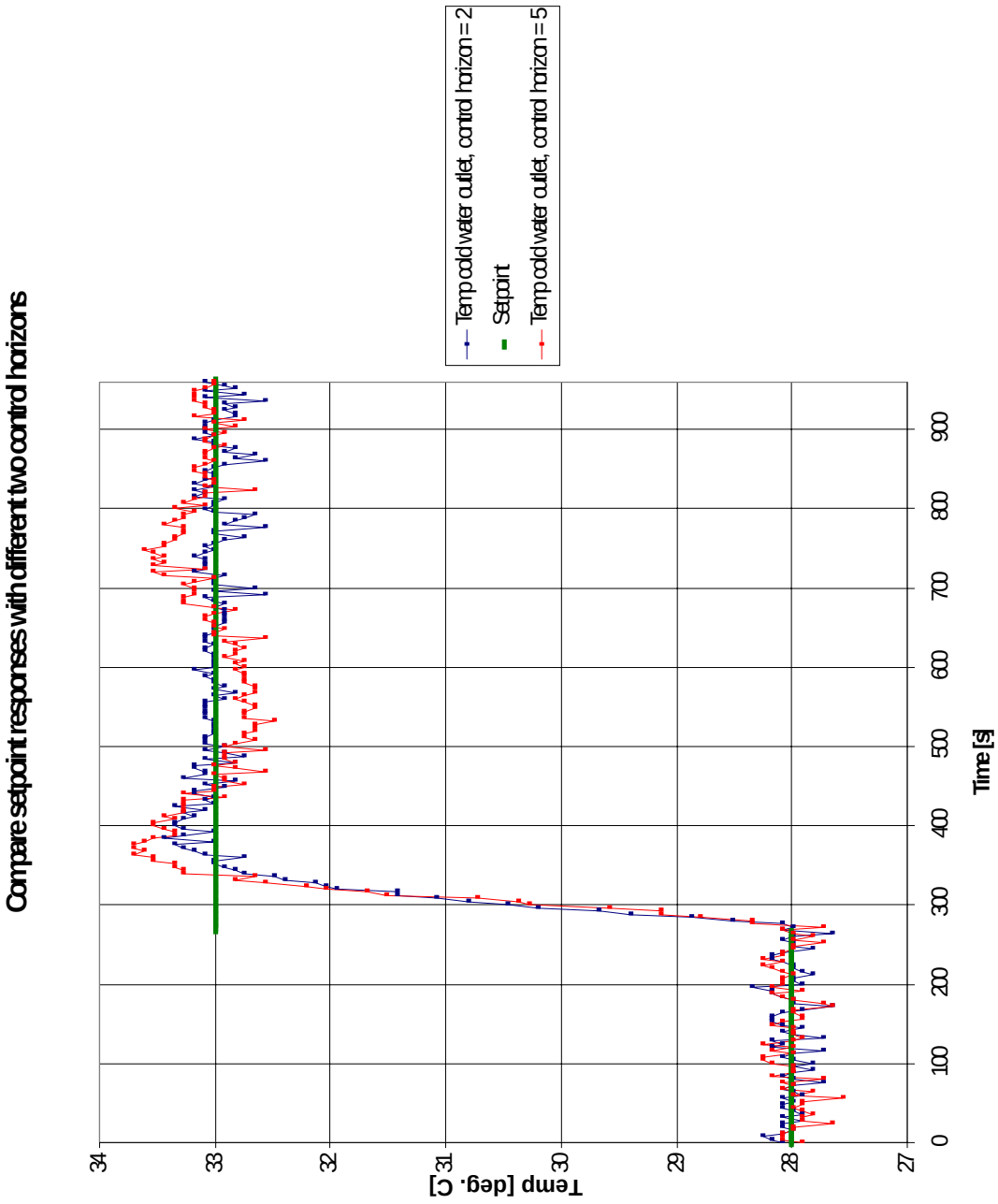



**Figure G.1 Compare test of GPC including RLS with cost horizon =6, ff = 0.98, λ = 0.3 and control horizon = 2 or 6.**

## G.2 Comparison of setpoint responses with different cost horizons

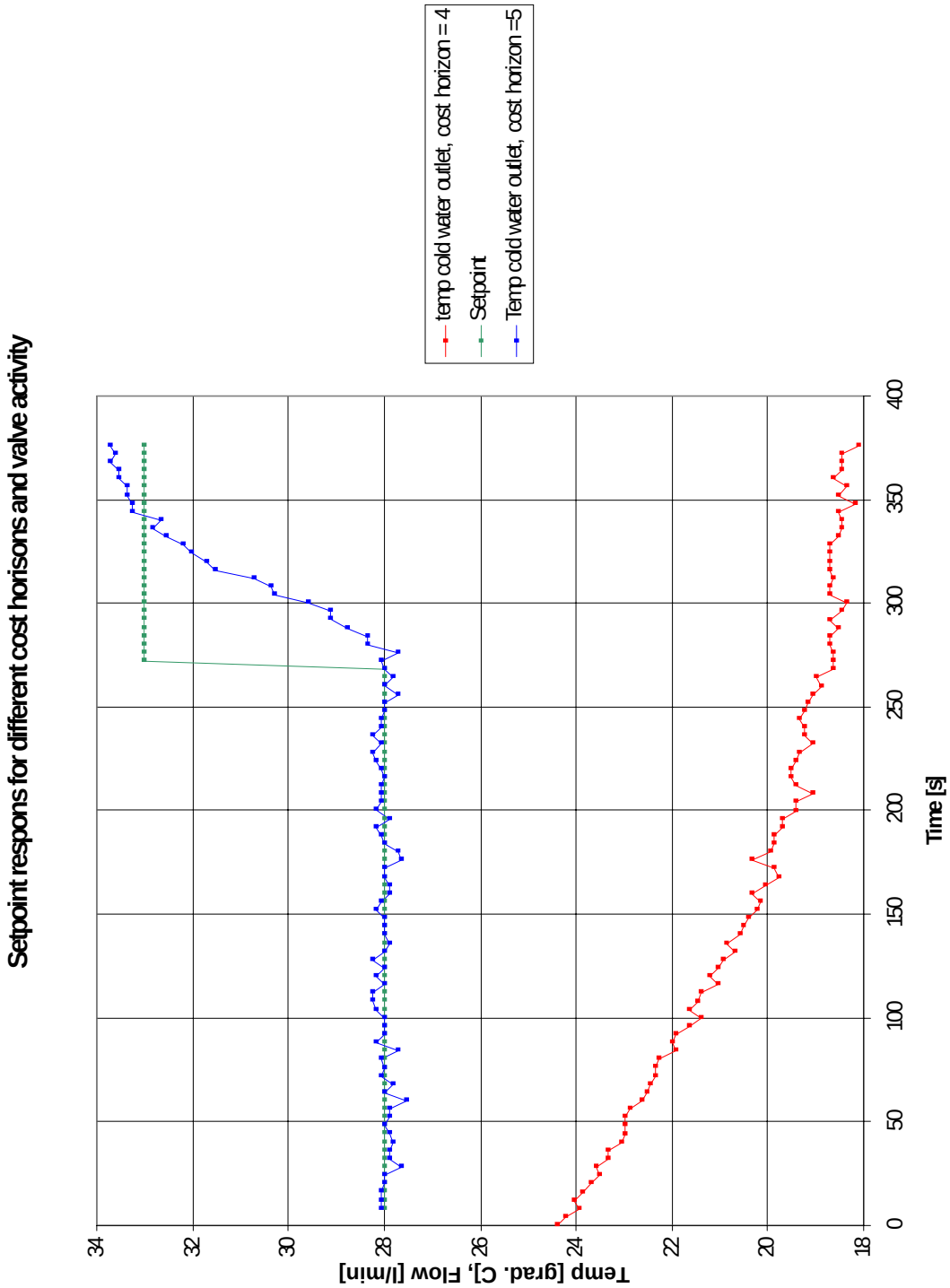



**Figure G.2 Compare test of GPC including RLS with cost horizon = 4, control horizon =4, λ = 0,3, ff =0.98 and cost horizon =5, control horizon =5, λ = 0,3, ff =0.98.**

## G.3 Comparison of setpoint responses with different forgetting factors

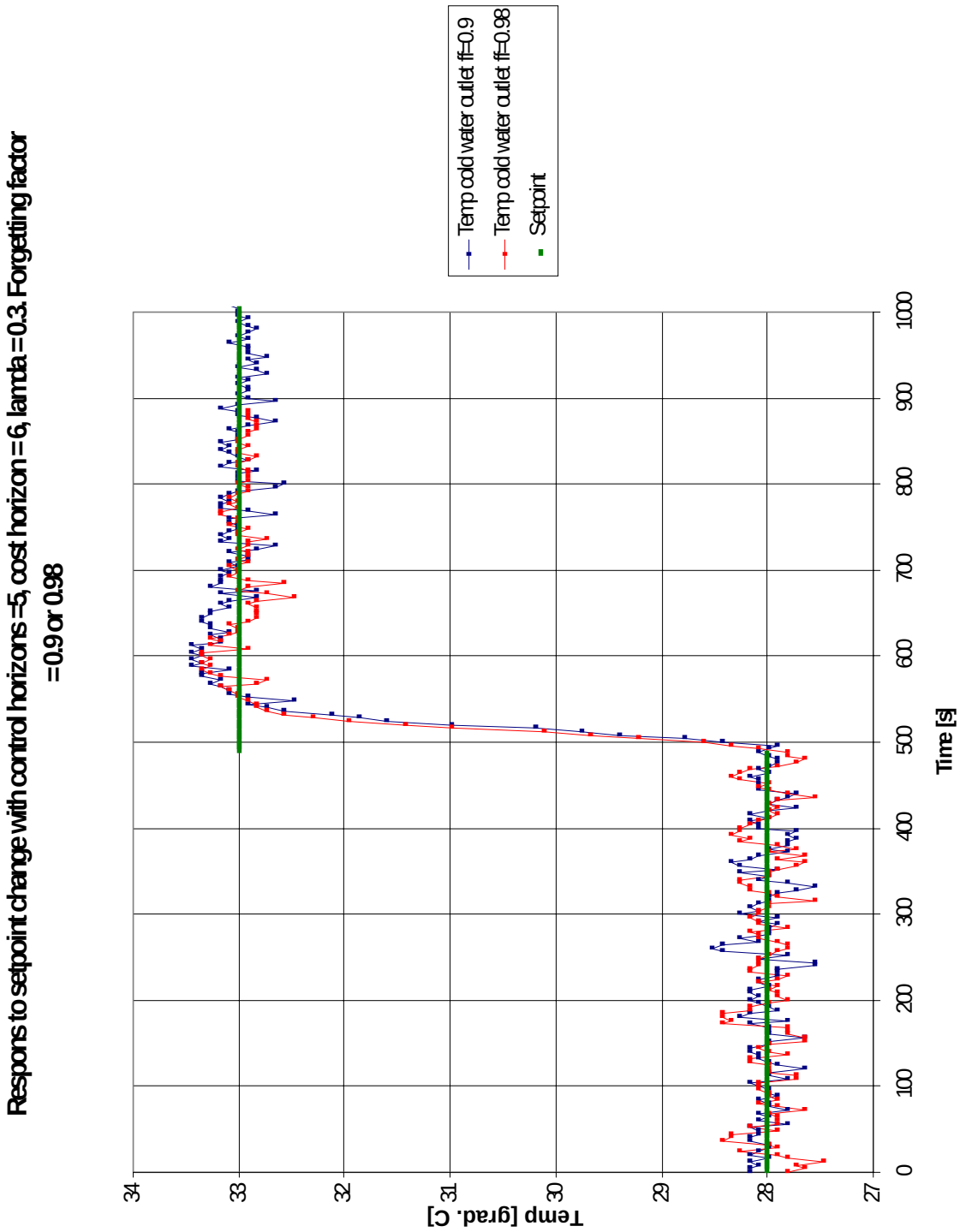



**Figure G.3 Compare test of GPC including RLS with different forgetting factor. Cost horizon = 6, control horizon = 5, $\lambda$ = 0.3 and with ff = 0.9 and 0.98.**

## G.4 Comparison of setpoint responses with different control weighting

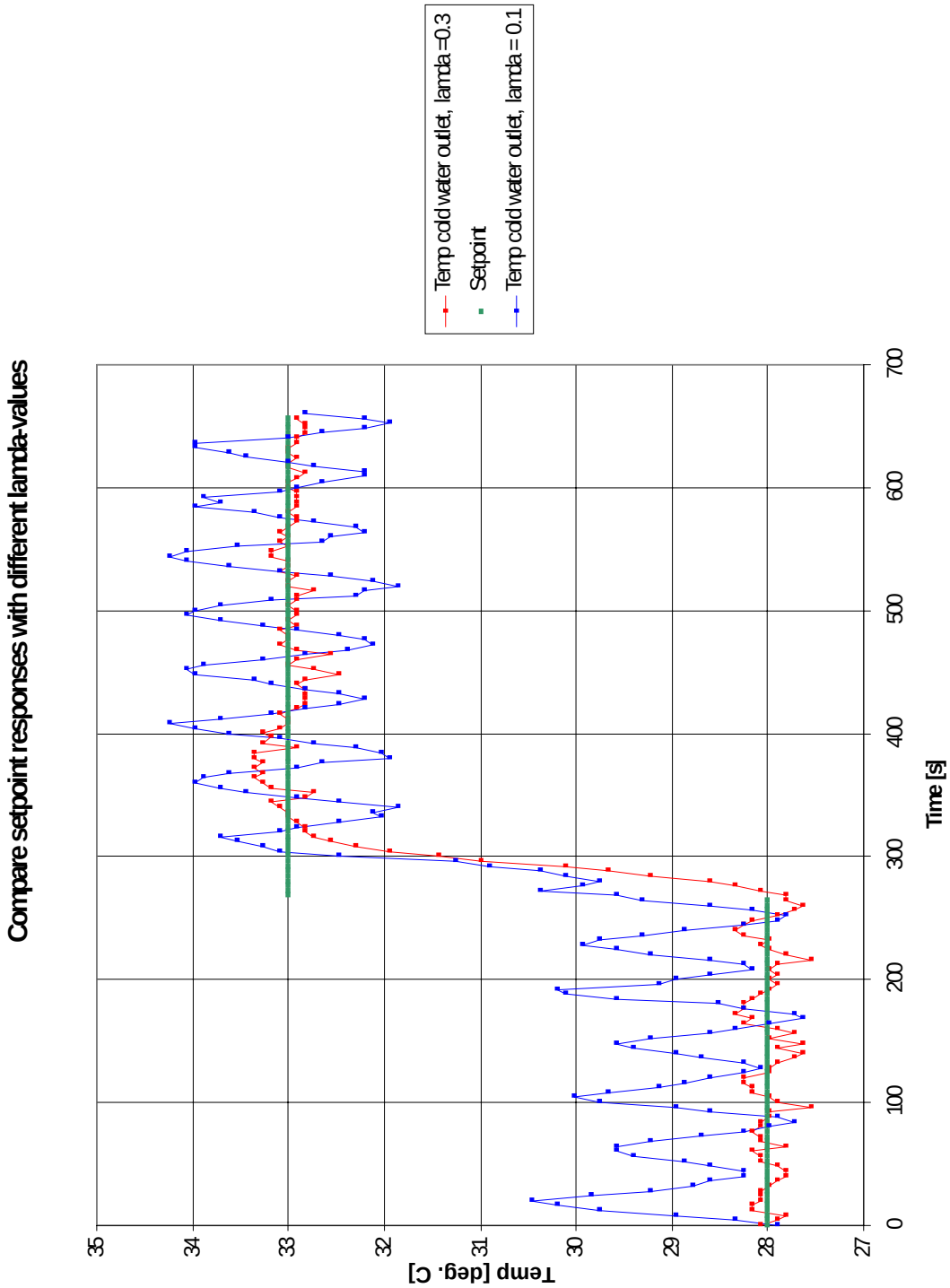



**Figure G.4 Compare tests of GPC including RLS with cost horizon = 6, control horizon = 5, ff = 0.98 and $\lambda$ = 0.1 or $\lambda$ =0.3.**

# H LABVIEW DIAGRAM - PROCESS INCLUDING PID

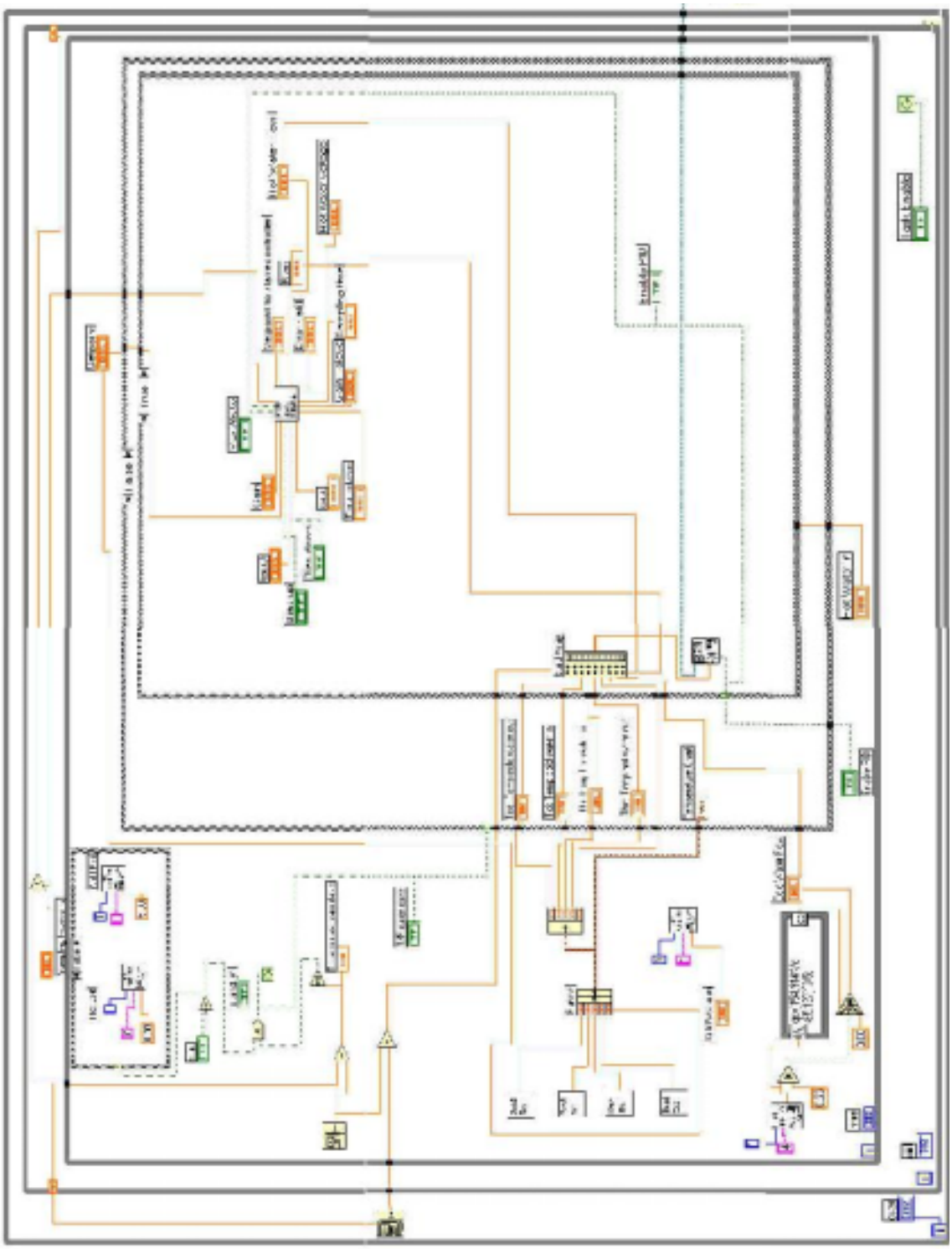


**Figure H.1 Labview diagram of PID controller**

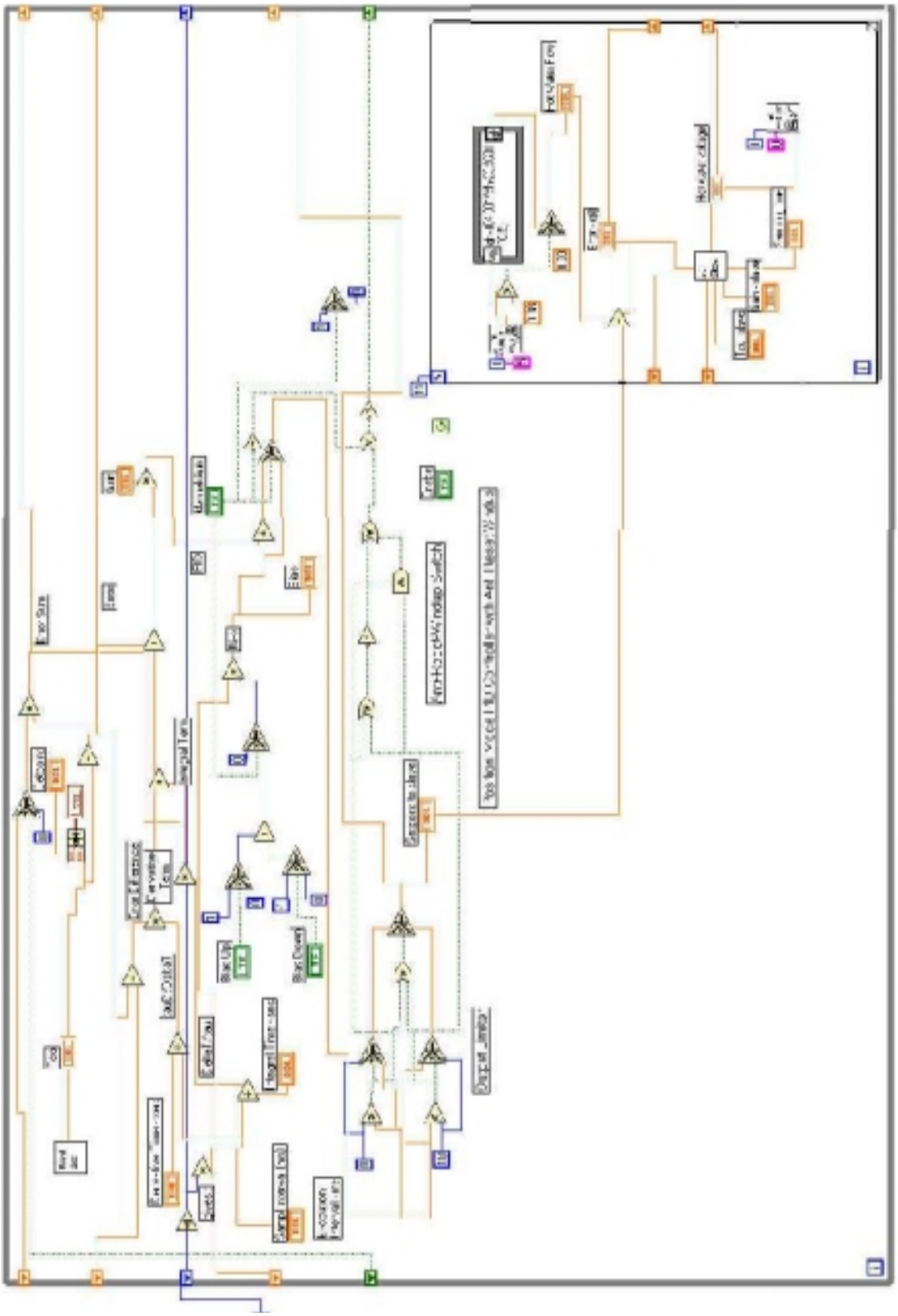


**Figure H.2 Labview diagram of slavecontroller and PID used as in subVI**

# I TESTS OF THE PID CONTROLLER

## I.1 Setpoint response with PID controller with different sampling time

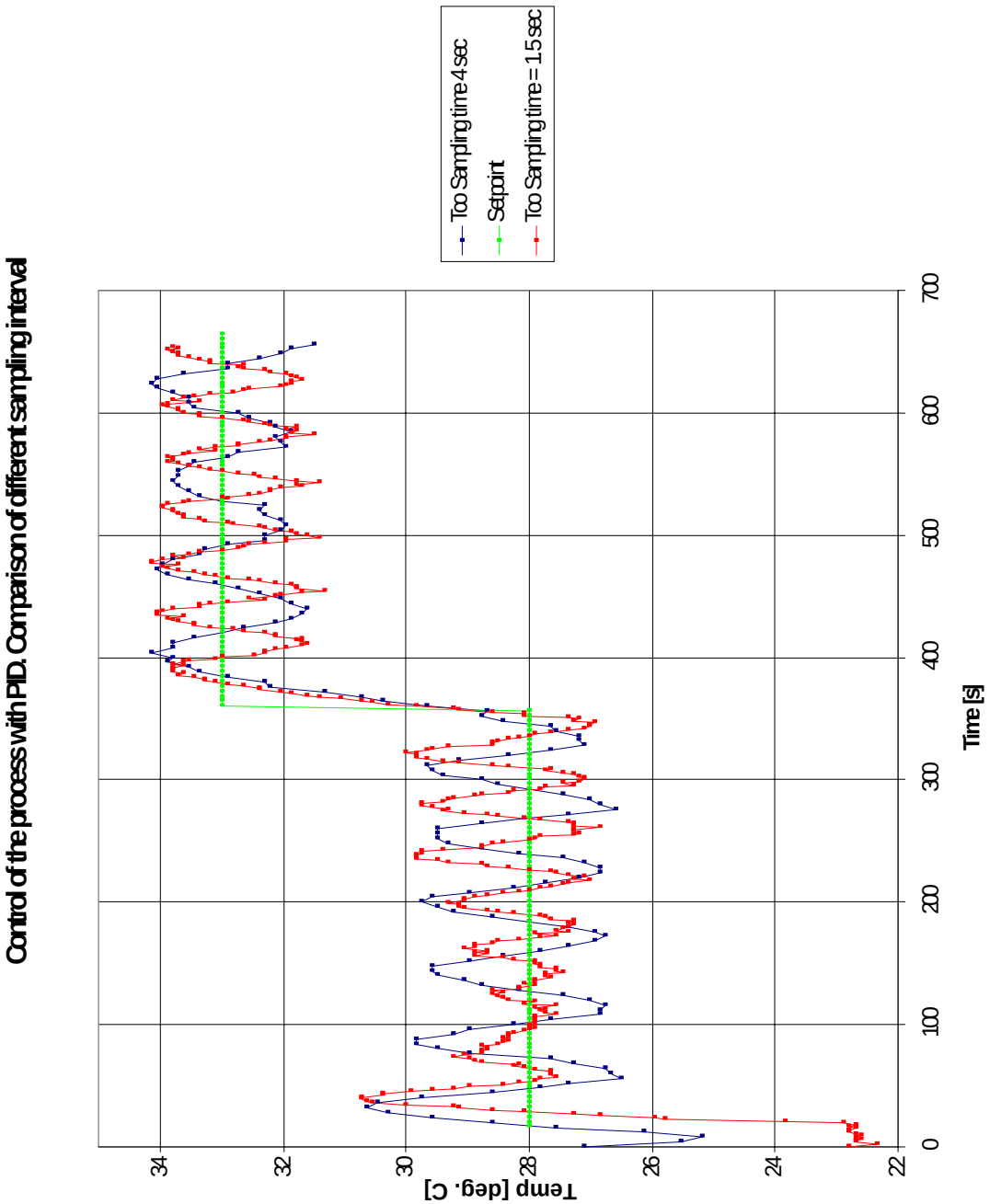



**Figure I.1 Test of PID with different sampling time. Sampling time = 1.5 sec. and 4 sec. Kc = 2, $\tau_I$ = 60 and $\tau_D$ = 2.**

## I.2 Startup response with different tuning parameters for the PID

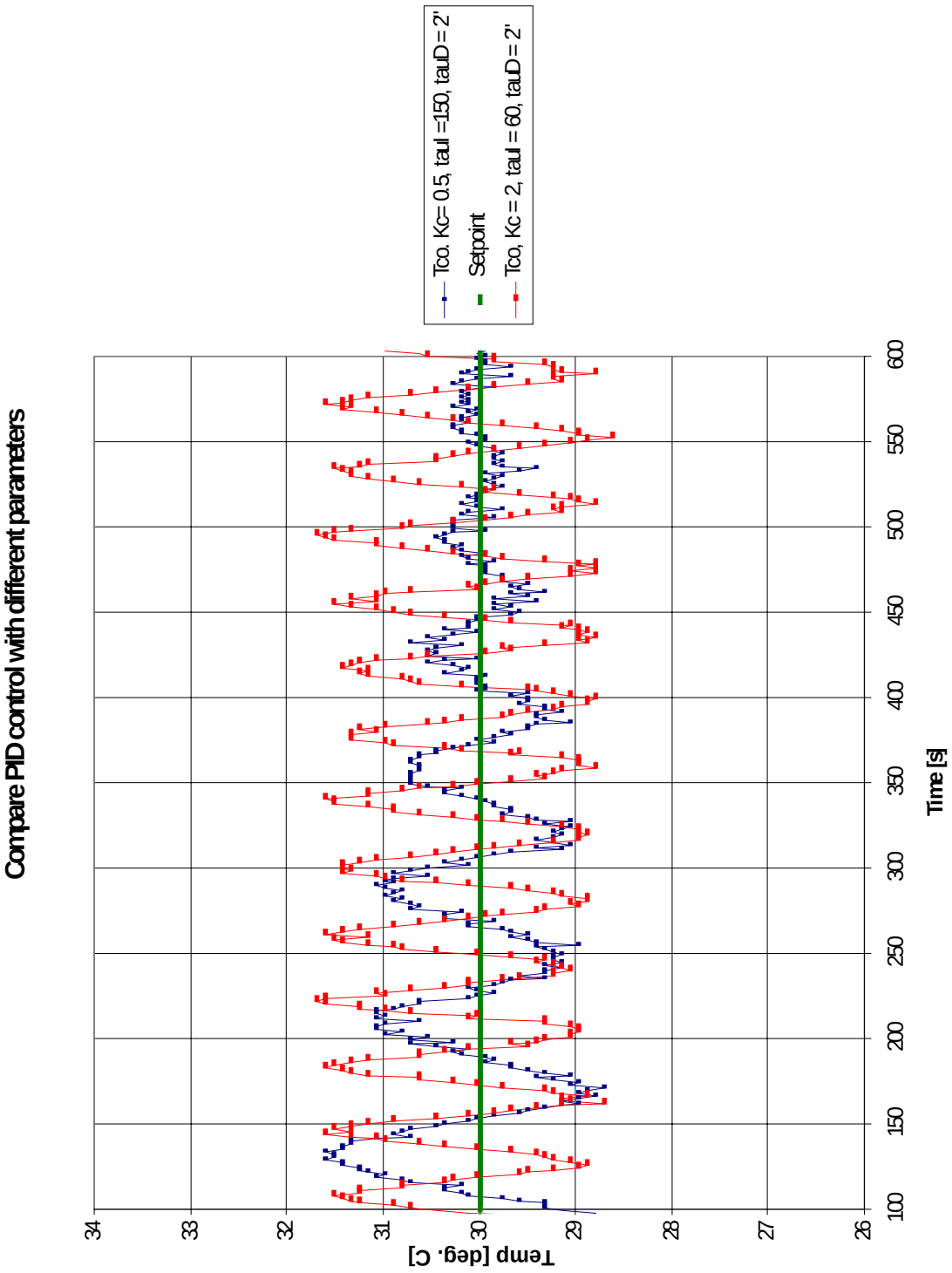



**Figure I.2 Comparison of PID with tuning parameters $K_c$ = 2, $\tau_I$ = 60, $\tau_D$ = 2 and $K_c$ = 0.5, $\tau_I$ = 150, $\tau_D$ = 2.**

## I.3 Compare performance of GPC and PID controller

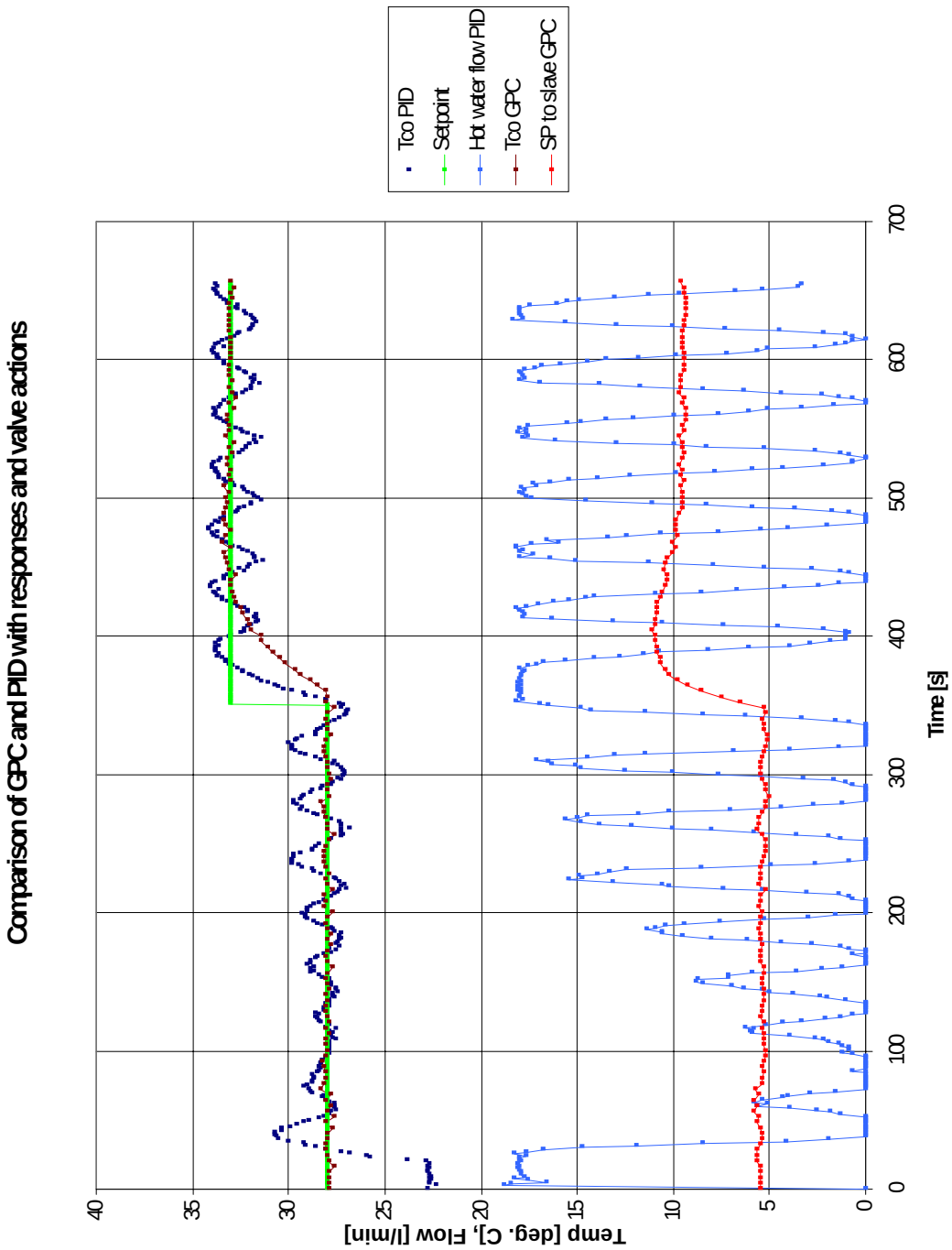



**Figure I.3 Comparison of GPC and PID with setpoint responses and valve actions with cost horizon = 5, control horizon = 2, ff = 0.98, $\lambda$ = 0.3 and sampling time = 4 sec. for the GPC and $K_c$ = 2, $\tau_I$ = 60, $\tau_D$ = 2 and sampling time = 1. 5 sec. for PID**

# J MATLAB PROGRAM CODES FOR THE MULTIVARIABLE CASE

## J.1 Program Code Fitting ARX-Model Multivariable Case

```
%Present an ARX-model from simulation data for a multivariable case

load c:\MATLABR11\work\Diplom\MatlabHysys\NewTests\prbs3x3new.txt
%reading output and input from a loaded file,
%the name is the same as the name to the loaded file
u1=prbs3x3new(:,8);
ua=normaliz(u1);
u2=prbs3x3new(:,9);
ub=normaliz(u2);
u3=prbs3x3new(:,10);
uc=normaliz(u3);
y1=prbs3x3new(:,5);
ya=normaliz(y1);
y2=prbs3x3new(:,6);
yb=normaliz(y2);
y3=prbs3x3new(:,7);
yc=normaliz(y3);

z=[ya yb yc ua ub uc];%set the output and input in a vector
ny=3; %number of outputs
nu=3; %number of inputs

%constants
namin=1*ones(ny); %minimum degree of A polynomial
namax=5*ones(ny); %maximum degree of A polynomial
nbmin=1*ones(ny,nu); %minimum degree of B polynomial
nbmax=5*ones(ny,nu); %maximum degree of B polynomial
nkmin=1*ones(ny,nu); %%minimum degree of delay
nkmax=5*ones(ny,nu); %maximum degree of delay

%Initialized model
na=namin;  %initialize the degree of A polynomial
nb=nbmin;  %initialize the degree of A polynomial
nk=nkmin;  %initialize the degree of delay
nn=[na nb nk]; %put the degrees in a vector
th=arx(z,nn); %develop the arx-model
FPEint=th(2,1);%initialize Aikake's Final Prediction Error.
%See "help theta" for structure of the th matrix
naopt=na; %save the optimal degree of A
```

```
nbopt=nb; %save the optimal degree of B
nkopt=nk; %save the optimal degree of delay
optparameters=[naopt nbopt nkopt];
optFPE=FPEint;
optloss_function=th(1,1);
opt_th=th;
%Check all the combinations of model structure that gives the best
%describtion of the process
for i=namin(1,1):1:namax(1,1)
   for j=nbmin(1,1):1:nbmax(1,1)
      for k= nkmin(1,1):1:nkmax(1,1)
         na=i*ones(ny,ny);
         nb=j*ones(ny,nu);
         nk=k*ones(ny,nu);
         nn=[na nb nk];
         th=arx(z,nn); %develop the arx-model
         FPE=th(2,1); %Displays the FPE in the th matrix
         loss_funciton=th(1,1);
         if FPE<FPEint %checks the FPE, less value for FPE gives better
model
            FPEint=FPE; %update the FPE value if its less than the past
value
            naopt=na; %save the optimal degree of A
          nbopt=nb; %save the optimal degree of B
            nkopt=nk; %save the optimal degree of delay
            optparameters=[naopt nbopt nkopt];
            optFPE=FPE;
            optloss_function=th(1,1);
            opt_th=th;
         end %if
      end %for
   end %for
end %for

%optimal parameters
e=resid(z,opt_th); %compute the residuals and display correlation graphs
stdev_e=std(e) %finds the standard devation of the error
var_e=var(e) %finds the variance of the error

%Display modelparameters and other keyparamters
[Aopt,Bopt]=th2arx(opt_th)
optFPE
optloss_function
optparameters
```

## J.2 Program Code Recursion of Diophantine in Multivariable Case

```
function [E_rec,F_rec]= diophantinemulti(A,B,N,ny)
%Solves the Diophantine equation, A and B is modelparameters, N is the
cost horizon
%Uses the recursively method described by Clarcke et. al. in
%"Generalized Predictive Control - Part I. The Basic Algorithm"
%Automatica. Vol. 23, No.2, pp 137-148, 1987.
%Rembember A= I(nyxny)+A1*z(-1)+A2*z(-2)+.....+Ana*z(-na)
%where I, A1, A2 ..... Ana are matrixes all of size (ny)x(ny) (ny=number
of outputs)

%Initializing E
E=eye(ny);

%Initializing F
Atilda=[A zeros(ny)]-[zeros(ny) A]; %Atilda=A(1-q^-1)
d=size(Atilda);
v=[eye(ny) zeros(d(1),d(2)-ny)];
Aprime=v-Atilda; %Aprime= 1-Atilda
F=Aprime(:,ny+1:size(Aprime,2));   %F=q(1-Atilda)
m=size(Aprime,2)-size(F,2);
F=[F zeros(ny,m)];  %Size F needs to be equal Atilda= size Aprime

%Saving the E & F for each loop, including the initial values
E_rec=zeros(N*ny); %Initialize the E that is recorded
E_rec(1:ny,1:ny)=E;
F_rec=zeros(N*ny,size(F,2));
F_rec(1:ny,1:size(F,2))=F; %Initilize F that is recorded

%Recursion loop for Diophantine
for j=2:N
   r=F(:,1:ny);
   R=[E r]; %updates the new R value, which takes the first element in F
and add in
   S=F(:,ny+1:end)-r*Atilda(:,ny+1:end);
   r=[R zeros(ny,N*ny-j*ny)]; % Set R which is E(j+1) so R represent
E(j+2) in next loop
   E_rec(j*ny-(ny-1):j*ny,:)=r;
   E=R;
   %Set S which is F(j+1) so S represent F(j+2) in next loop
   F=[S zeros(ny,size(Atilda,2)-size(S,2))];
   F_rec(j*ny-(ny-1):j*ny,:)=F;
end

%Erase the last column.
```

```
%In the calculation the size of F needs to equal the size of Atilda. This
adds ny columns.
F_rec=F_rec(:,1:size(F_rec,2)-ny);
```

## J.3 Program Code for Initializing

```
global ny nu yrls urls w A B ypast uopt  beta P K nu ny ymean umean ymstore
umstore EthInEth PropInProp PropInBot ReboilRatio EthaneFlow PropaneFlow

%Initialize data
upast=zeros(50,1);
ypast=zeros(50,1);
delta_u=zeros(50,1);
yrls=[];
urls=[];
ymean=[0.9705;0.9623;9.0e-5];
umean=[0.4722;0.3611;0.934];
ymstore=[];
umstore=[];
EthInEth=[];
PropInProp=[];
PropInBot=[];
ReboilRatio=[];
EthaneFlow=[];
PropaneFlow=[];
ExitFlag=[];

%Startvalues for the model. Updates by RLS

A=[ 1.0000         0         0   -1.6900    0.2388    0.0595    0.6971
         0    1.0000         0    0.0251   -1.9607    0.0451   -0.0256
         0         0    1.0000    0.1511    0.0849   -1.6938   -0.1500

   -0.2364   -0.0571
    0.9633   -0.0429
   -0.0797    0.6990];


B=[0         0         0    0.0179    0.0055   -0.0967    0.0032  -0.0023
   0         0         0    0.0008    0.0001    0.0037    0.0004  -0.0005
   0         0        0     0.0009    0.0009   -0.0235    0.0049   0.0002

   0.l1440   -0.0029   -0.0018   -0.0362    0.0011   -0.0004  -0.0148
  -0.0005   -0.0005    0.0001   -0.0054    0.0007    0.0002   0.0010
   0.0353    0.0003   -0.0007   -0.0132    0.0040    0.0000   0.0005];
```

## J.4 Program Code GPC Including QP- Algorithm

```
global ny nu yrls urls w A B ypast uopt beta P K ny nu ymean umean ymstore
umstore EthInEth PropInProp PropInBot ReboilRatio EthaneFlow PropaneFlow

%Algorithm that minimize the cost function subject to constrains
%on the control signals and the outputs. Multi-input multi-output case
%The file initialize.m needs to be executed before mpcqp.m runs for the
first time
%The file is executed from VBA

%Declare variables
alfanorm=0.05; %smoothing grade mean values for inputs and outputs
%Declare counters
j=1;
p=1;
s=1;
i=1;
l=1;
q=1;
r=1;

%Constants, from VBA
N1=const(1); %minimum costing horizon
NU=const(2); %control horizon
N2=const(3); %maximum costing horizon
lamda=const(4); %control-weighting sequense
ny=const(5); %number of outputs
nu=const(6); %number of inputs
w1=const(7);%referense trajectory(=setpoint) Mole Fraction Ethane in
Ethane
w2=const(8); %ref.traj. LV fraction propane in propane
w3=const(9); %ref.traj. LV fraction propane in T-2 Bottoms

N=N2-N1+1;  %cost horizon

%Declare variables
y=zeros(ny,1);  %outputs
u=zeros(nu,1);  %inputs

%Process data from Hysys
%Past output
u(1)=DataFromHysys(1);  %Flow ethane. Unit in kmol/s
u(2)=DataFromHysys(2); %Flow propane. Unit in kmol/s
u(3)=DataFromHysys(3); %Reboil ratio
y(1)=DataFromHysys(4); % Mole Fraction ethane in ethane* stream
```

```
y(2)=DataFromHysys(5); %LV Fraction propane in propane* stream
y(3)=DataFromHysys(6); %LV Fraction propane in T-2 Bottoms* stream

%Normalize data from Hysys because time-series models are developed with
%normalized data
ynorm=y-ymean;
ymean_new=alfanorm*y+(1-alfanorm)*ymean;
unorm=u-umean;
umean_new=alfanorm*u+(1-alfanorm)*umean;

%Create setpoint vector
%length (w) needs to be N*ny
w=[];
for l=1:N
   w=[w;w1;w2;w3];
end

%Storage of process data
upast=[unorm; upast(1:size(upast,1)-nu)];
ypast=[ynorm; ypast(1:size(ypast,1)-ny)];

%Calculate delta u = u(t)-u(t-1). Used in free response for cost function
with
%delta u as variable. Not generalized for different numbers of nu
for s=1:(size(upast,1)-nu)/nu
   delta_u(nu*s-2)=upast(nu*s-2)-upast(nu*s+nu-2);
   delta_u(2*s-1)=upast(2*s-1)-upast(2*s+nu-1);
   delta_u(2*s)=upast(2*s)-upast(2*s+nu);
end

%Calculate the Diophantine eq recursivly
[E_rec,F_rec]=diophantinemulti(A,B,N,ny);

%Create gconv, used to calculate the G matrix
%Not generalized for different numbers of ny!
for r=1:N
   gconv(ny*r-(ny-1),:)=conv(E_rec(ny*r-(ny-1),:),B(ny-2,:));
   gconv(ny*r-(ny-2),:)=conv(E_rec(ny*r-(ny-2),:),B(ny-1,:));
   gconv(ny*r,:)=conv(E_rec(ny*r,:),B(ny,:));
end

%Select diagonals from gconv matrix
for i=1:nu*NU
   gdiag(:,i)=diag(gconv,i-1);
end
```

```
%Create G matrix used in the model prediction
G=zeros(N*ny,NU*nu);
for j=1:NU*nu
   G(:,j)=gdiag(:,j);
end

%Compute the G1,G2,...GN2 for the free response
gfree=triu(gconv,nu);
for p=1:N*ny
   gtemp(p,:)=[gfree(p,nu+p:end) zeros(1,p+nu)];
end
gdiff=gtemp(:,1:size(gtemp,2)-ny);

%The free response
free2=F_rec*ypast(1:size(F_rec,2));
free1=gdiff(:,2:end)*delta_u(1:size(gdiff,2)-1);
f=free1+free2;

%Compute cost function as a qp-problem
I=eye(size(G,2));
H=2*(G'*G+lamda*I); %matrix H in the qp-function
b=2*((f-w)'*G); %row vector in the qp-function
Ir=eye(N*ny,NU*nu);
R=[Ir;-Ir;G;-G]; %constrain-matrix. Ru <= c
Iy=eye(ny,nu);
Iqpy=zeros(N*ny,nu); %help matrix for constrains calc.
for i=1:N
   Iqpy(i*ny-(ny-1):i*ny,:)=Iy;
end

%Constraints values
umin=[0;0;0.6];   %minimum control signal values
umax=[0.8333;0.6944;0.95];  %maximum control signal values
u0=[0.4722;0.3611;0.934];   %start values for optimalization
ymin=[0.96;0.95;0];  %minimum output values
ymax=[1;1;0.005];  %maximum output values

%Normalize constraints values
uminnorm=umin-umean;
umaxnorm=umax-umean;
yminnorm=ymin-ymean;
ymaxnorm=ymax-ymean;
u0norm=u0-umean;

%Start vector for the optimalization. Needs to be of dim. NU*nu
u0norm1=[u0norm;u0norm];
```

```
%compute c matrix in constrains eq Ru <= c
c=[Iqpy*umaxnorm;-Iqpy*uminnorm;Iqpy*ymaxnorm-f;-Iqpy*yminnorm+f];

%Solving the QP-problem
[uoptnorm,fval,exitflag,output]=quadprog(H,b,R,c,[],[],[],[],u0norm1);
ExitFlag=[ExitFlag exitflag];

%when the maximum number of iterations was exceeded or the problem is
unbounded,
%infeasible, or QUADPROG failed to converge with a solution X,
%the optimal solution = startpoint to avoid unstability in Hysys
if exitflag <=0
   uoptnorm=u0norm1;
end

%Change optimalized inputs from normalized to real
uopt=uoptnorm(1:nu)+umean;

setpoints=[uopt(1:nu)]; %only the first control actions are sended to
Hysys

%Scaling setpointsdata.
%There is a convertion in Hysys which needs to be
%adjusted in Matlab
setpoints(1)=setpoints(1)*1300/10.83;
setpoints(2)=setpoints(2)*1190/8.264;
setpoints(3)=(setpoints(3)*142.89-71.458)/3600;
%reboil ratio need to be to be divided with 3600
%because of convertion due to units in Hysys. Reboil ratio is sended to
Hysys as a
%molar flow in kmol/s



%The RLS part
%select inputs and outputs data sended to RLS algorithm
rlsdata=[ypast(ny+1:2*ny);unorm];
%call the RLS algorithm
[A,B]=adaptmulti(rlsdata);

%Update meanvalues for normalizing
ymean=ymean_new;
umean=umean_new;

%Store ymean and umean to see drifting in process
ymstore=[ymstore ymean];
umstore=[umstore umean];
```

```
%Store data from process
EthInEth=[EthInEth DataFromHysys(4)];
PropInProp=[PropInProp DataFromHysys(5)];
PropInBot=[PropInBot DataFromHysys(6)];
ReboilRatio=[ReboilRatio DataFromHysys(3)];
EthaneFlow=[EthaneFlow DataFromHysys(1)];
PropaneFlow=[PropaneFlow DataFromHysys(2)];

done=1; %value that tells VBA that Matlab finished the calculations
```

## J.5 Program Code RLS Multivariable Case

```
function [A,B]=adaptmulti(rlsdata)
%Adaptiver based on RLS method in the multivariable case
%rlsdata is data from the GPC function

global ny nu yrls urls w A B ypast beta P K

%declare variables
ypred=zeros(1,3);

%w is the setpoint trajectory
%A and B is modelparamters
%ypast is the past mesured values
%yrls and urls is vectors the past vector is made from
%beta is the parameter estimate in a column vector, P is the covar. matrix
%and K is the gain vector

%Save the past data
yrls=[rlsdata(1:ny);yrls]; %update the past measurement vector
urls=[rlsdata(ny+1:ny+nu);urls]; %update the past control signals vector
i=size(yrls,1); %count how many data the rls algorithm has

%Choose values for the degree of polynomnials and the deadtime
n=2;  %Degree of the A polynominal
m=4;  %Degree of the B polynominal
d=1;  %Deadtime

%Test if there is enough data to run RLS
if i>=(m+d+ny)*ny
   %Constants
   c=1000; %for initializing of the P matrix
   gama=0.9;%forgetting factor
```

```
   alfa=0.0045; %variance to the error, found from PRBS-data

  %Initializing of the system
  beta_init=zeros(n*ny+m*nu,ny); %initialize the beta-vector
  I=eye(m*nu+n*ny);%identity matrix
  Pinit=c*I; %initialize the P matrix
   %vector with past data from measurement and ctrl signal
   ksi=[yrls(1:ny*n);urls(1+d:nu*m+d)];
   %Compute model prediction
   if i==(m+d+ny)*ny
      ypred=ksi'*beta_init; %if the loop runs the first time, uses the
initlized beta
   else
      ypred=ksi'*beta; %if the loop runs 2nd time or more, uses the beta
from last run
   end

   %Update gain vector
   if i==(m+d+ny)*ny
      K=(Pinit*ksi)./(alfa*gama+ksi'*Pinit*ksi);
   else
      K=(P*ksi)./(alfa*gama+ksi'*P*ksi);
   end

   %Update parameter estimates
   if i==(m+d+ny)*ny
      beta=beta_init+K*((yrls(1:ny))'-ypred);
      %update modelparameter A. The first coeff is always the identity
matrix
      A=[eye(ny);-beta(1:ny*n,1:ny)]';
      %updata B. -beta comes from different definition of A and B in RLS
and GPC
      B=[zeros(d*nu,ny);-beta(n*ny+1:n*ny+m*nu,1:ny)]';
   else
      beta=beta+K*((yrls(1:ny))'-ypred);
      A=[eye(ny);-beta(1:ny*n,1:ny)]';
      B=[zeros(d*nu,ny);-beta(n*ny+1:n*ny+m*nu,1:ny)]';
   end

   %Compute covariance matrix for next iteration
   if i==(m+d+ny)*ny
      P=(1/gama)*(I-K*ksi')*Pinit;
   else
      P=(1/gama)*(I-K*ksi')*P;
   end
else %else if i>m+d+1
   %if not enough data, the ARX-model found from PRBS-data is used
   %A and B given from the PRBS data
```

```
A=[ 1.0000         0         0   -1.6900    0.2388    0.0595    0.6971
         0    1.0000         0    0.0251   -1.9607    0.0451   -0.0256
         0         0    1.0000    0.1511    0.0849   -1.6938   -0.1500

   -0.2364   -0.0571
    0.9633   -0.0429
   -0.0797    0.6990];


B=[0         0         0    0.0179    0.0055   -0.0967    0.0032  -0.0023
   0         0         0    0.0008    0.0001    0.0037    0.0004  -0.0005
   0         0        0     0.0009    0.0009   -0.0235    0.0049   0.0002

   0.l1440   -0.0029   -0.0018   -0.0362    0.0011   -0.0004  -0.0148
  -0.0005   -0.0005    0.0001   -0.0054    0.0007    0.0002   0.0010
   0.0353    0.0003   -0.0007   -0.0132    0.0040    0.0000   0.0005];


end %end if i>m+d+1
```

# K COMPOSITION PROFILE IN T-2 COLUMN

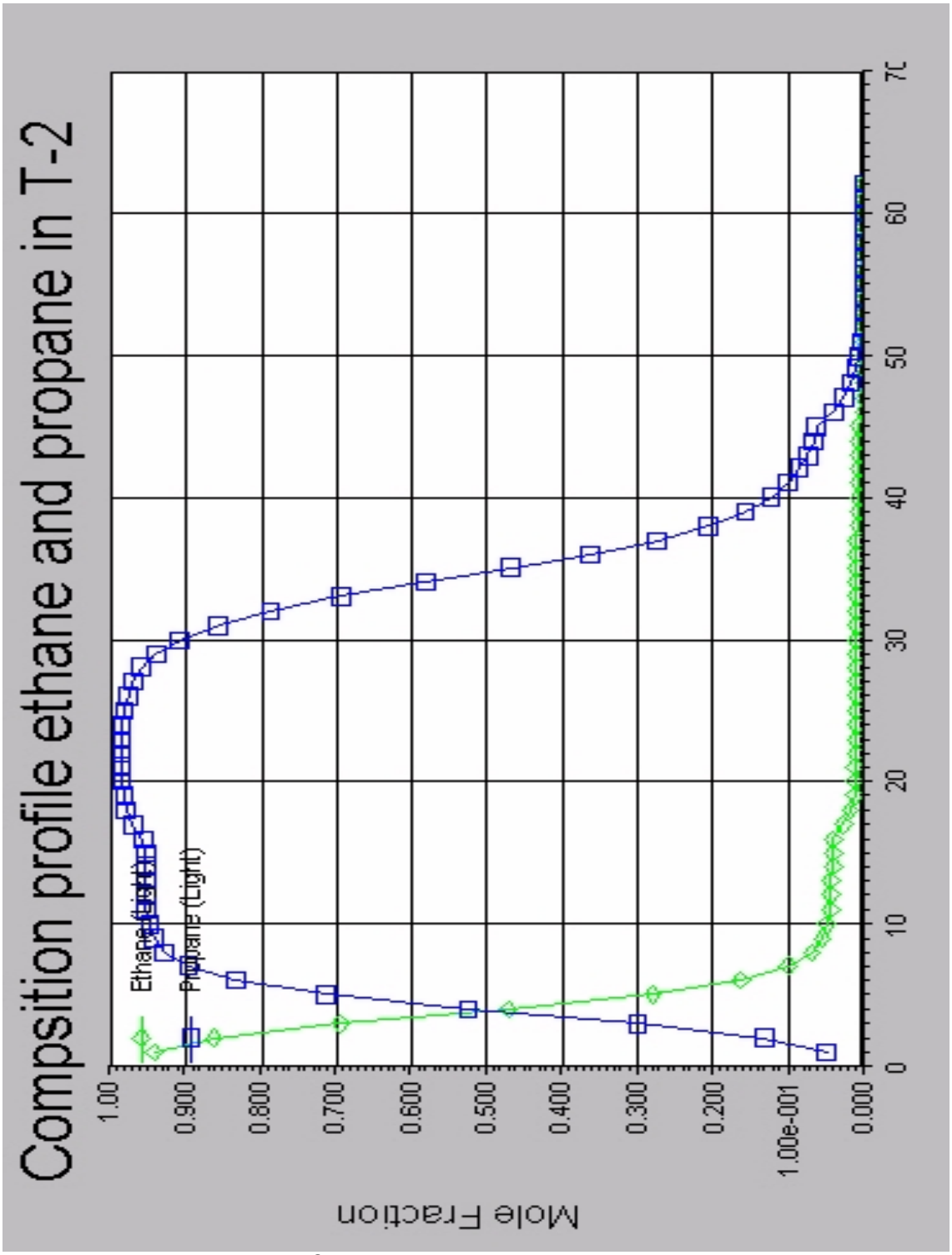



**Figure K.1 Composition profile ethane and propane in column T-2**

# L PROGRAM CODE IN VBA

```
'*********************************************************
'Description: Program that execute HYSYS.Plant simulation
'and Matlab algorithm that gives control inputs from
'GPC with model-updating based on RLS, and transfer
'information between the two programs.
'Excel File: MPCControl.xls
'Matlab file: mpcqp.m
'HYSYS file:B&VSimulationDiam.hsc
'Author: Elvira Marie Bergheim
'Date: 30.Jan.2001
'*********************************************************

'Require all variables to be explicit declared
Option Explicit

'Decleare HYSYS objects
'Note: chech HYSYS Type Library and Matlab Aut. Server type
'under Tools --> References
Public HyAppl As Object
Public HyCase As SimulationCase
Public HyFlowsheet As Flowsheet
Public HyStreams As Streams
Public HyQStreams As Streams
Public Matlab As Object
Public ColumnFlowsheet As ColumnFlowsheet

'Declare variables
Dim Res As String  ' Help variable for Matlab commands
Dim DataToMatlab(5, 0) As Double  'array sending data to Matlab
Dim DataToMatlabIm() As Double  'Imaginary part of DataToMatlab
Dim Constants(8, 0) As Double  'Array collection constants values
Dim ConstantsIm() As Double  'Imaginary part of Constants
Dim FirstRun As Boolean  'Value True when algorithm runs for the first time
Dim MatlabFinish(0, 0) As Double 'tells if Matlab finished the calculations
Dim MatlabFinishIm() As Double  'Imaginary part of MatlabFinish
Dim ToHysysR(2, 0) As Double  'Array containing information to Hysys
Dim ToHysysIm() As Double  'Imaginary part of ToHysysR
Dim EthController As Controller
Dim PropController As Controller
Dim BoilupController As Controller
Dim T2 As ColumnOp
Dim CompEth As Variant
```

```
Dim CompProp As Variant
Dim CompT2Bot As Variant
---------------------------------------------------------------------
Sub MPCControl()

FirstRun = True 'First time the sub is runned

'Connect to Matlab if not already connected
If Matlab Is Nothing Then
   Set Matlab = CreateObject("Matlab.Application")
End If

'Connect to HYSYS objects
Set HyCase = GetObject("c:\Mine Dokument\Diplom\Hysys\Dynamics\B&VSimulationDiam105.hsc")
'Connect to applications object
Set HyAppl = HyCase.Application
'Set an object reference to currently active Hysys simulation
Set HyCase = HyAppl.ActiveDocument

'If there is no currently active case then display error message
If HyCase Is Nothing Then
   MsgBox "A HYSYS Simulation Case must be open"
   Exit Sub
End If

'Connect to flowsheet in active simulation case
Set HyFlowsheet = HyCase.Flowsheet
'Connect to a collection of all material streams
Set HyStreams = HyFlowsheet.MaterialStreams
'Connect to a collection of all energy streams
Set HyQStreams = HyFlowsheet.EnergyStreams
'Connect to column
Set T2 = HyFlowsheet.Operations.Item("T-2")

'Connect to operations
Set EthController = HyFlowsheet.Operations.Item("FC-Ethane")
Set PropController = HyFlowsheet.Operations.Item("FC-Propane")
Set BoilupController = HyFlowsheet.Operations.Item("RC-Boilup")


'Change directory
Res = Matlab.Execute("cd c:\MATLABR11\work\Diplom\MatlabHysys")

'Initialize data
```

```
If FirstRun Then
   Res = Matlab.Execute("Initialize")
   FirstRun = False   'not first time sub is executed anymore
End If


'Let the communication between Matlab and Hysys go for a certain time
While True
'    'Check if Hysys solver is ready
   If HyAppl.ActiveDocument.Solver.Integrator.IsRunning Then
      DoEvents
   Else

      'Send GPC constants to Matlab
      Constants(0, 0) = 1 'Minimimum cost horizon
      Constants(1, 0) = 2 'Control horizon
      Constants(2, 0) = 5 'Maximum cost horizon
      Constants(3, 0) = 1.5 'Lamda (control-weigthing sequence)
      Constants(4, 0) = 3 'Number of outputs
      Constants(5, 0) = 3 'Number of inputs
      Constants(6, 0) = 0.9708  'Setpoint mol fraction ethane in ethane
      Constants(7, 0) = 0.9622 'Setpint Liquid Volume Fraction propane in propane
      Constants(8, 0) = 0.000086 ' Setpoint Liquid Volume Fraction propane in T-2 Bottoms


      'Send Constant-values to Matlab
      Call Matlab.PutFullMatrix("const", "base", Constants, ConstantsIm)

      'Get data from Hysys that is necessary for control calculation.
      'past inputs (u)
      DataToMatlab(0, 0) = HyStreams.Item("Ethane*").MolarFlow
      DataToMatlab(1, 0) = HyStreams.Item("Propane*").MolarFlow
      DataToMatlab(2, 0) = BoilupController.PVValue  ' send the smoothed value

      'get component fractions
      CompEth = HyStreams.Item("Ethane*").ComponentMolarFraction.Values
      CompProp = HyStreams.Item("Propane*").ComponentVolumeFraction.Values
      CompT2Bot = HyStreams.Item("T-2 Bottoms*").ComponentVolumeFraction.Values

      'Past outputs
      DataToMatlab(3, 0) = CompEth(1) 'Molar Fraction Ethane in Ethane
      DataToMatlab(4, 0) = CompProp(2) 'LV Fraction Propane in Propane
      DataToMatlab(5, 0) = CompT2Bot(2) 'LV Fraction Propane in T-2 Bottoms

      'Send data to Matlab
```

```
                Call Matlab.PutFullMatrix("DataFromHysys", "base", DataToMatlab,
DataToMatlabIm)

    'Matlab execute file
    Res = Matlab.Execute("mpcqp")
    MatlabFinish(0, 0) = 0
    'chech if Matlab fishish algorithm execution
    While MatlabFinish(0, 0) = 0
      Call Matlab.GetFullMatrix("done", "base", MatlabFinish, MatlabFinishIm)
      DoEvents
    Wend

    'Get setpoints to slavecontrollers for control in Hysys
    'Variable "setpoints" from Matlab need to be columnvector
    Call Matlab.GetFullMatrix("setpoints", "base", ToHysysR, ToHysysIm)

    'Give Hysys the new setpoints values from Matlab
    HyStreams.Item("HelpEthane").MolarFlow.SetValue ToHysysR(0, 0)
    HyStreams.Item("HelpPropane").MolarFlow.SetValue ToHysysR(1, 0)
    HyStreams.Item("HelpRatio").MolarFlow.SetValue ToHysysR(2, 0)

    'Run model one timestep and stop
    HyAppl.ActiveDocument.Solver.Integrator.RunFor 2, "minutes"
    DoEvents
  End If
Wend 'Return to new loop

End Sub
```

# M PERFORMANCE OF THE GPC CONTROLLER

## M.1 Responses in manipulated variables

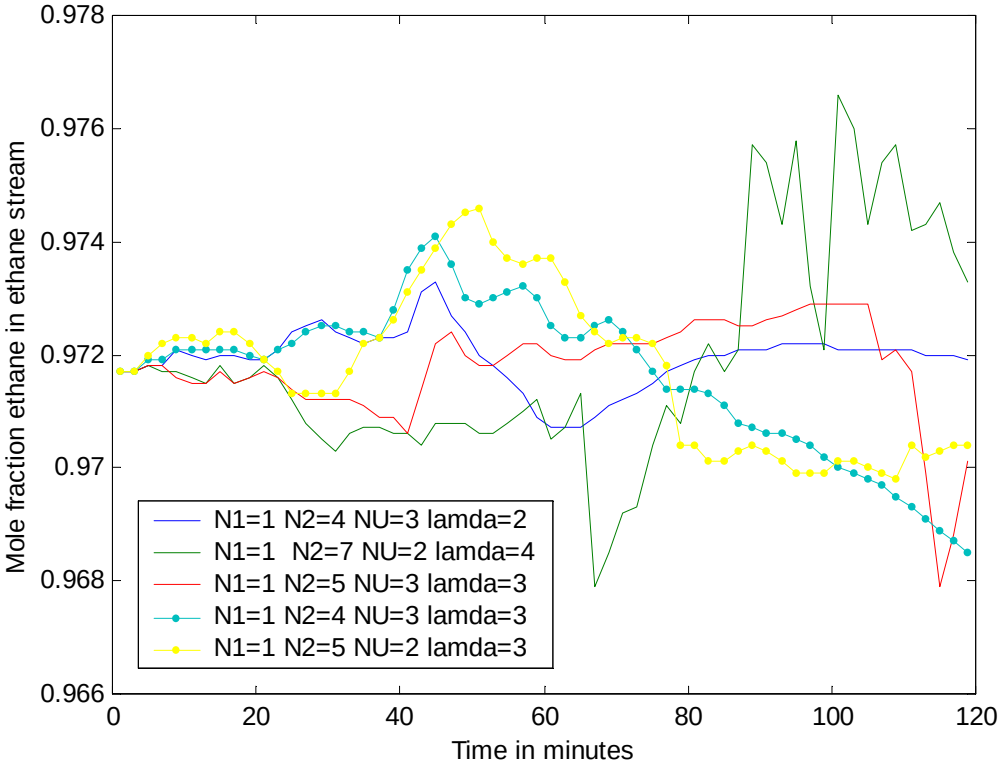



**Figure M.1 Mole fraction ethane in ethane outlet for different cost and control horizons when decreasing ethane molar flow in feed stream**

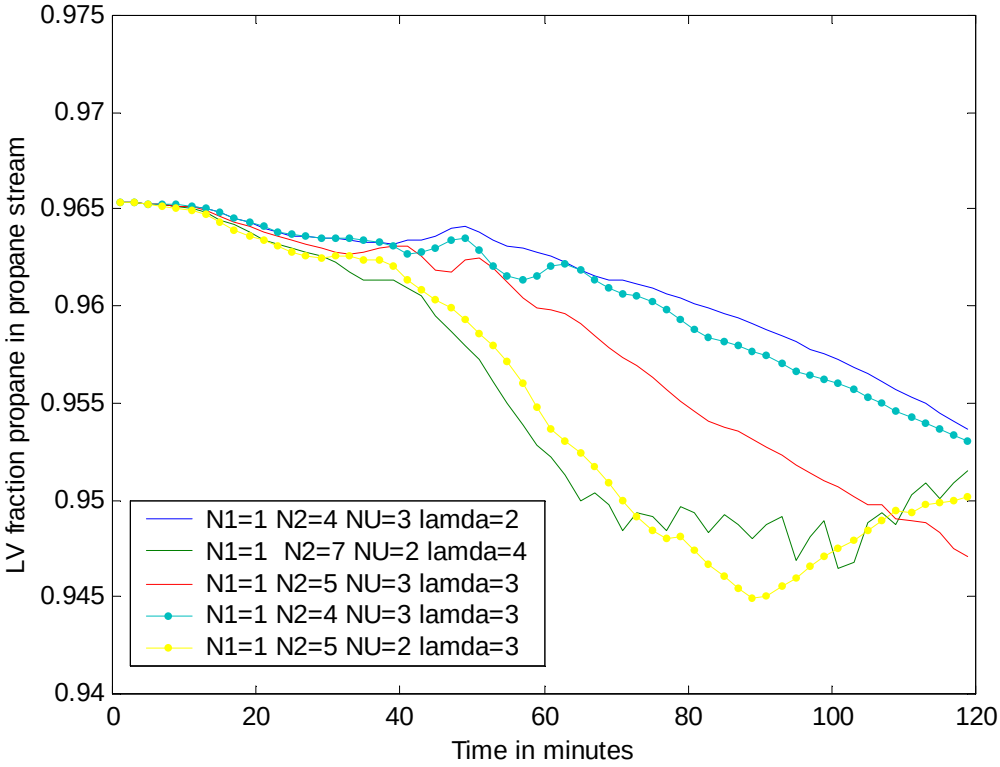



**Figure M.2 LV fraction propane in propane outlet for different cost and control horizons when decreasing ethane molar flow in feed stream**

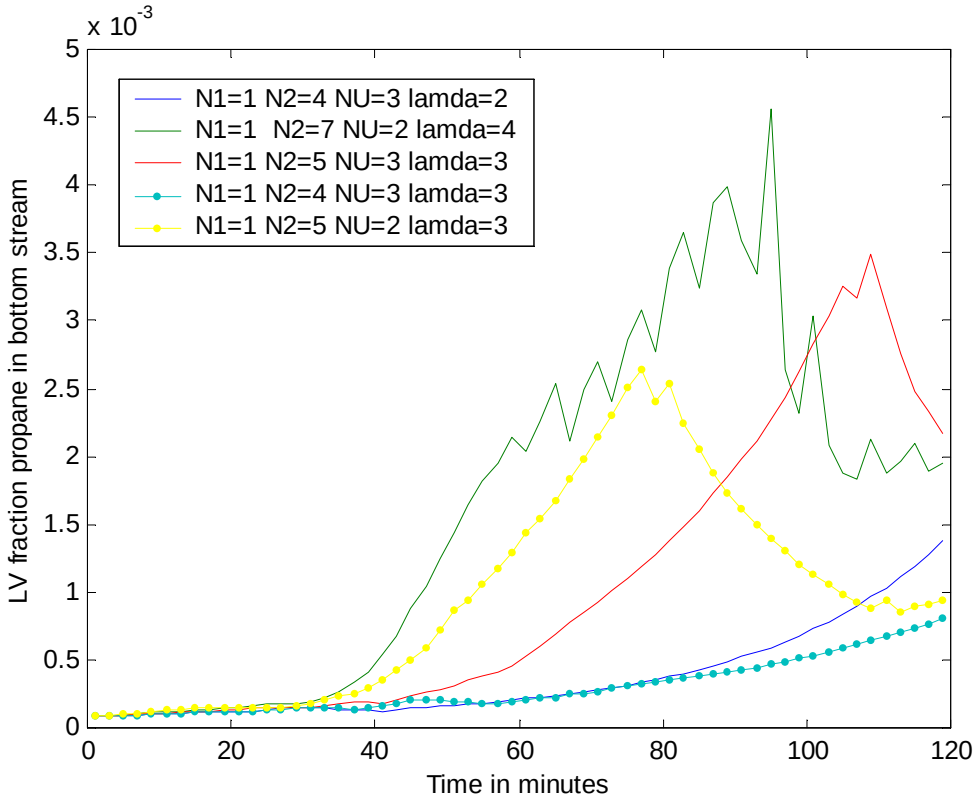



**Figure M.3 LV fraction propane in bottom stream for different cost and control horizons when decreasing ethane molar flow in feed stream**

## M.2 Responses in control signals

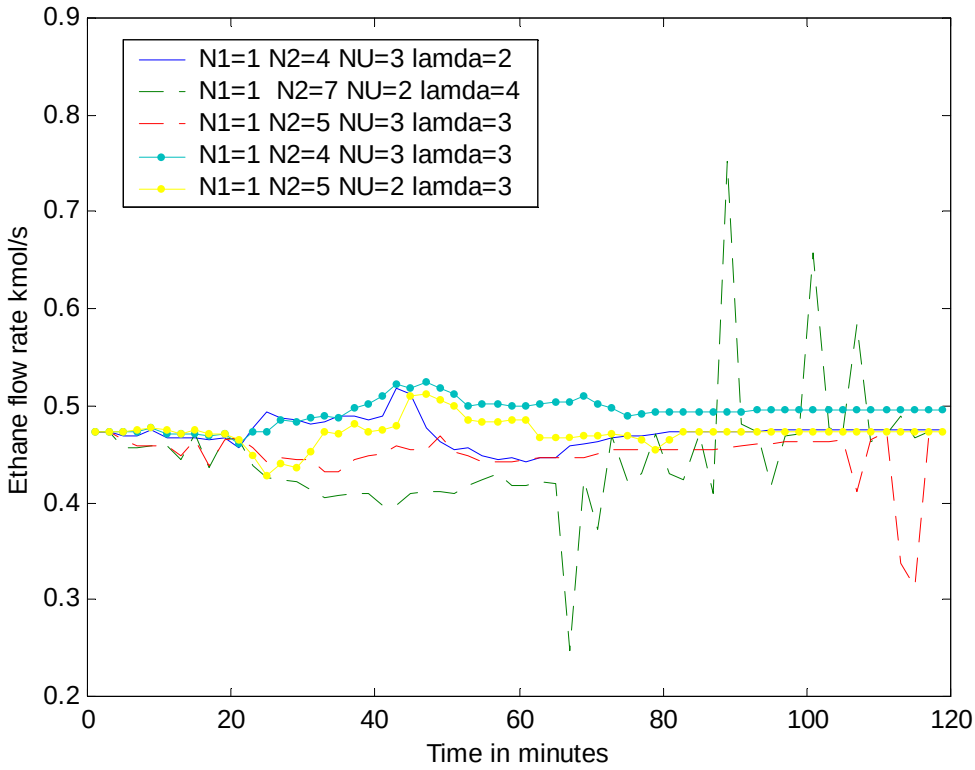



**Figure M.4 Flowrate ethane outlet for different cost and control horizons when decreasing ethane molar flow in feed stream.**

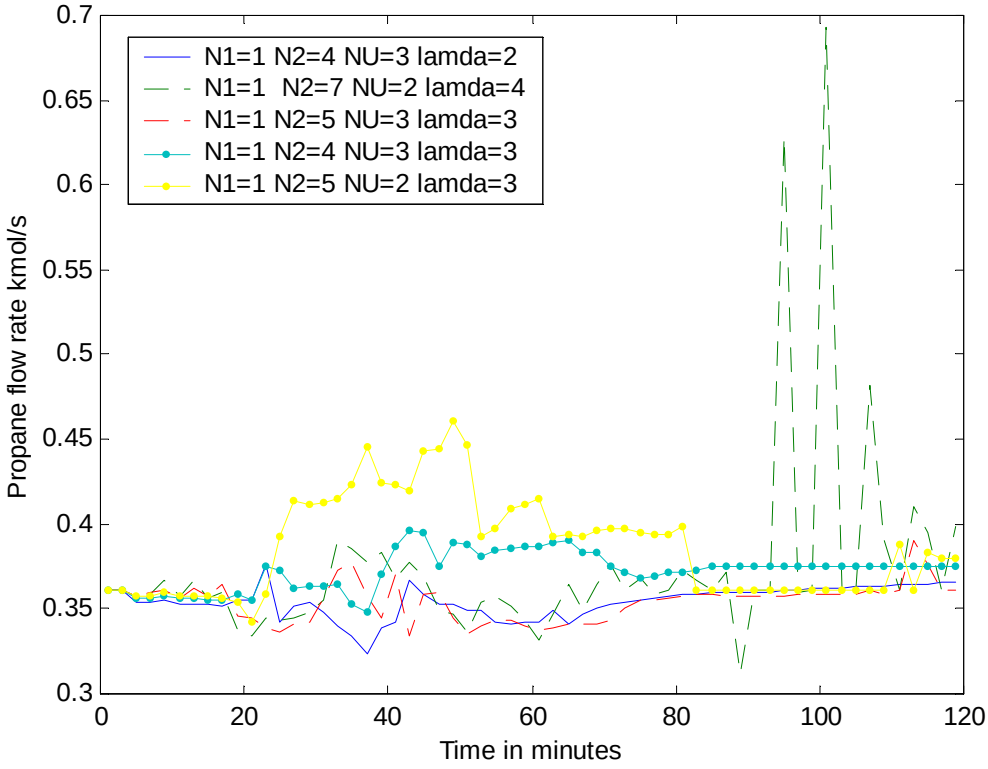



**Figure M.5 Flowrate propane outlet for different cost and control horizons when decreasing ethane molar flow in feed stream**

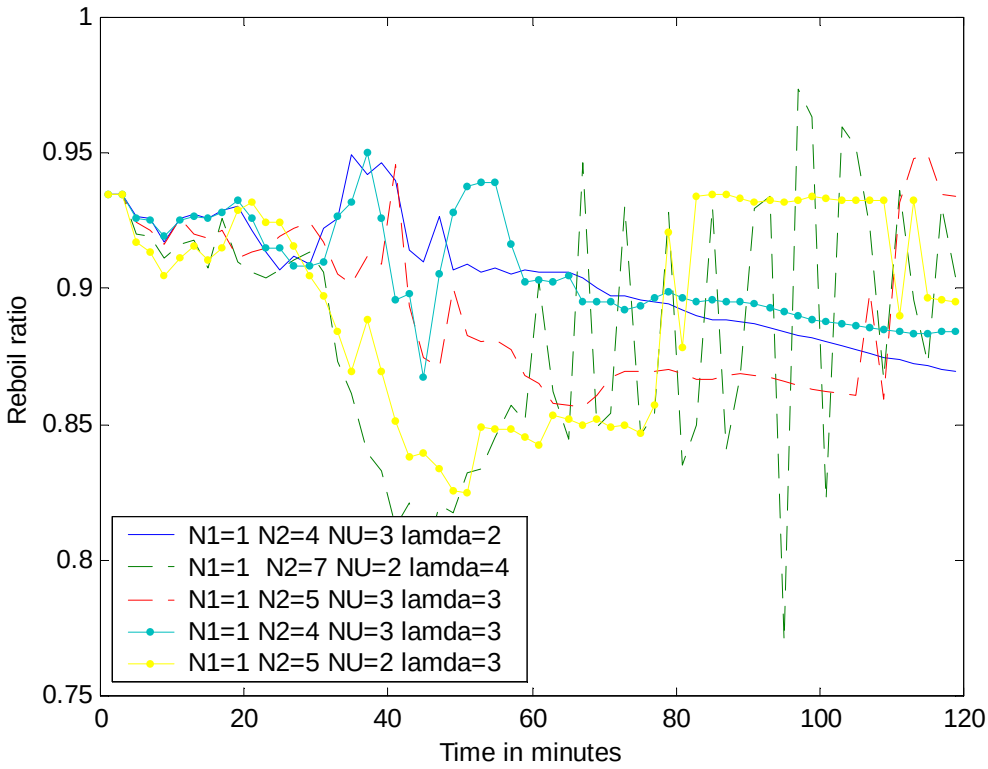



**Figure M.6 Reboil ratio for different cost and control horizons when decreasing ethane molar flowrate in feed stream.**

# N PERFORMANCE OF THE BUILT-IN MPC

## N.1 Responses manipulated variables

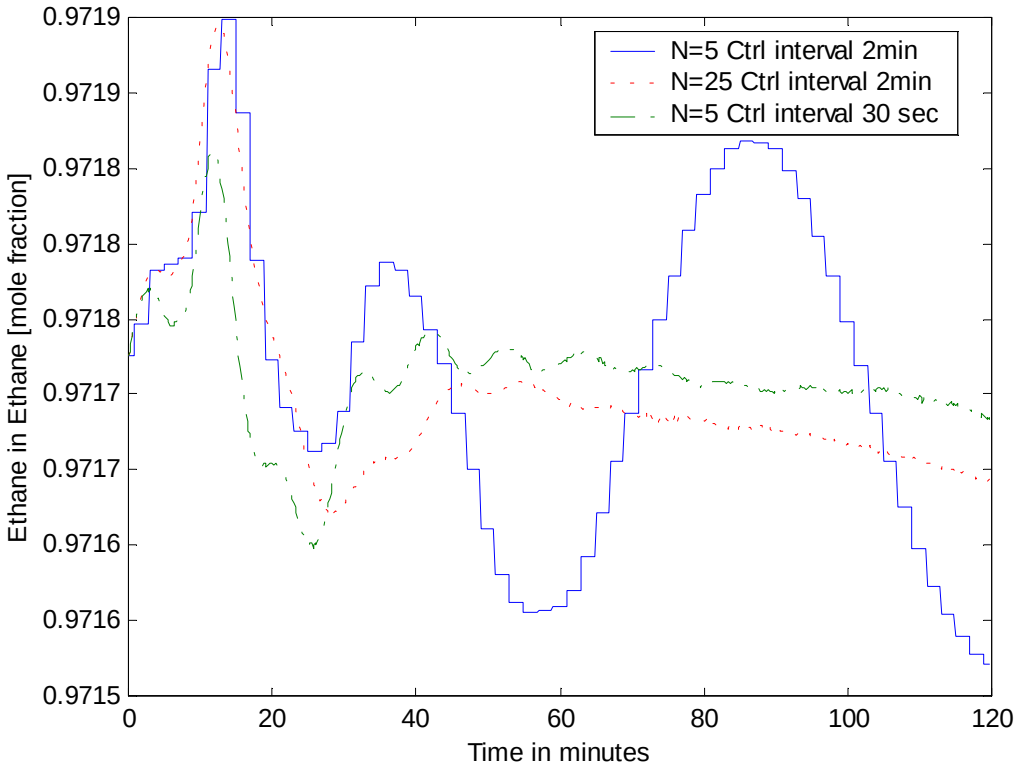



**Figure N.1 Mole fraction ethane in ethane outlet for different cost horizon and control interval when decreasing ethane molar flow in feed stream**

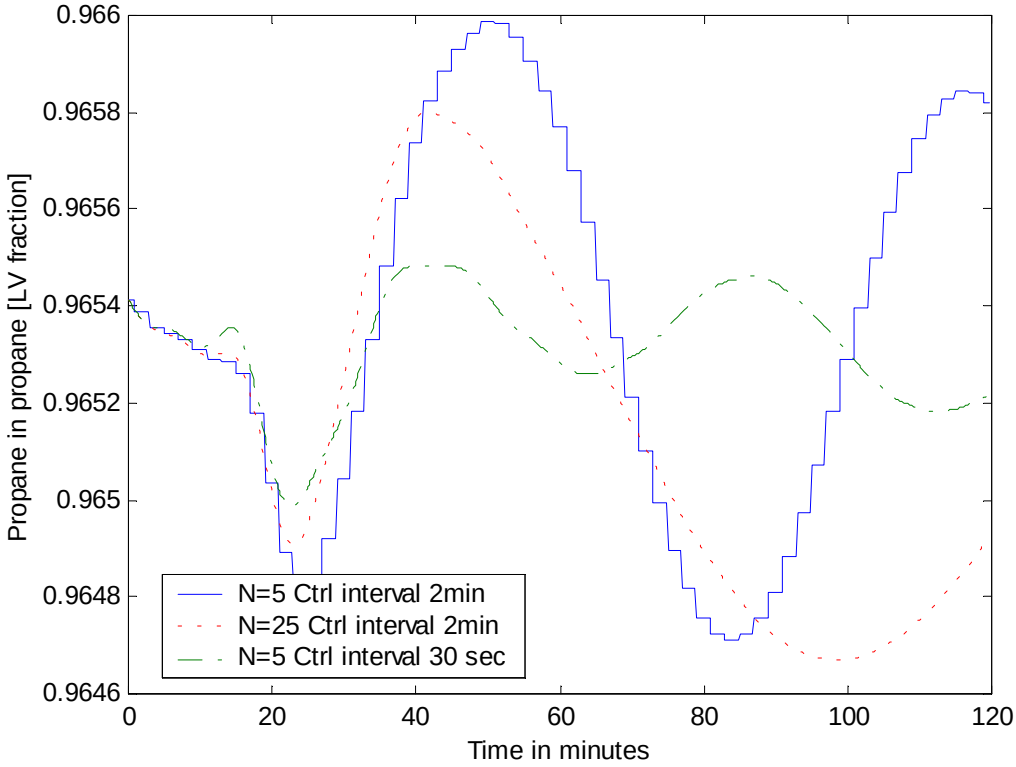



**Figure N.2 LV fraction propane in propane outlet for different cost horizons and control intervals when decreasing ethane molar flow in feed stream**

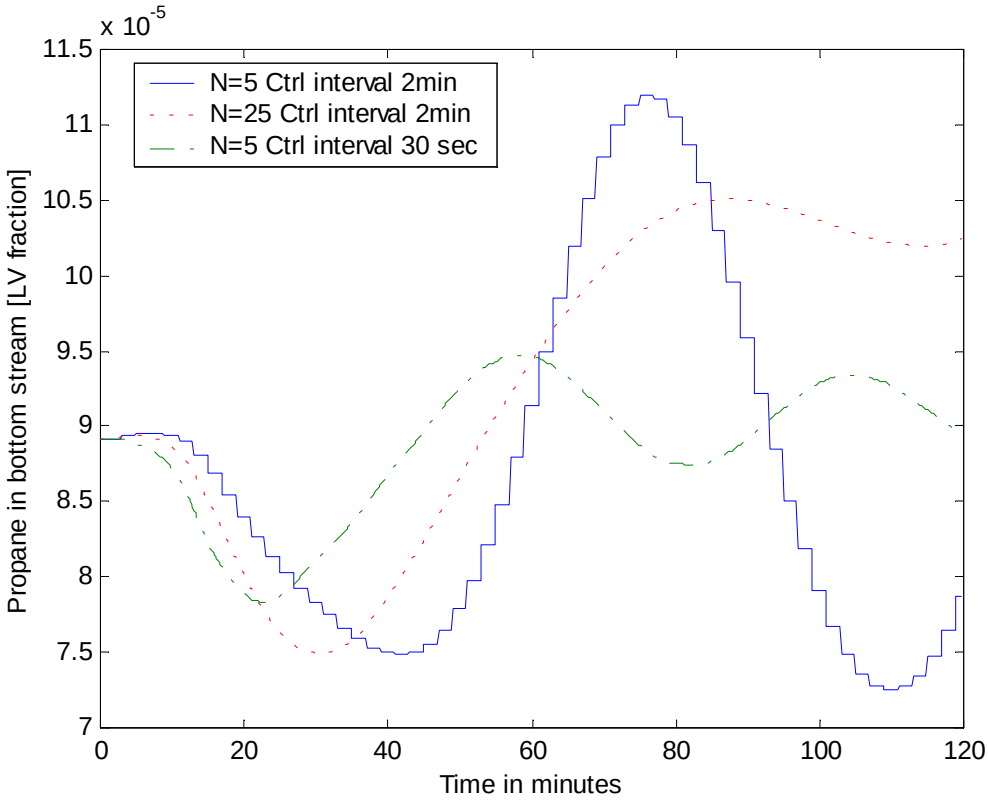



**Figure N.3 LV fraction propane in bottom stream for different cost horizons and control intervals when decreasing ethane molar flow in feed stream**

## N.2 Responses in control signals

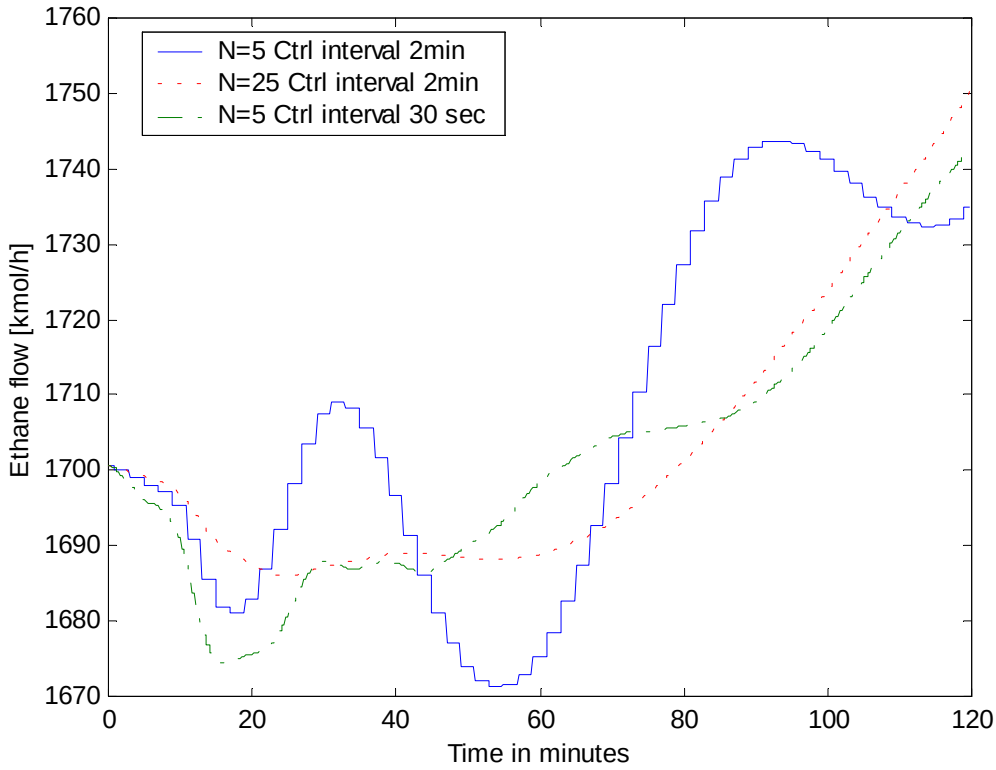



**Figure N.4 Flowrate ethane outlet for different cost horizons and control intervals when decreasing ethane molar flow in feed stream.**

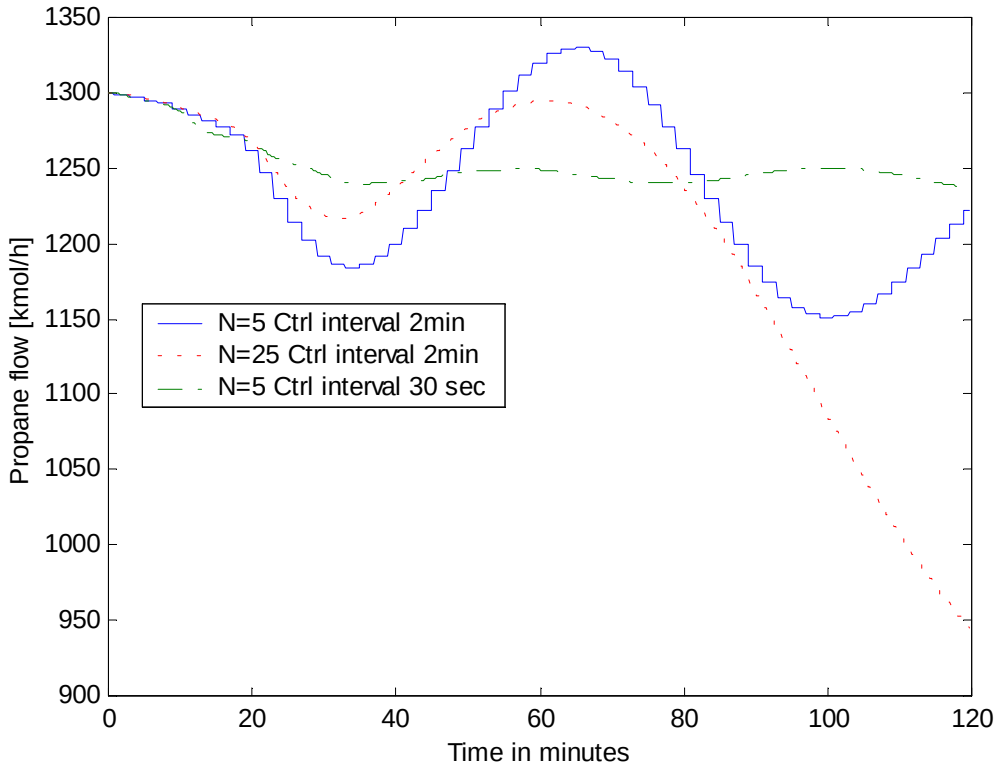



**Figure N.5 Flowrate propane outlet for different cost horizons and control intervals when decreasing ethane molar flow in feed stream**

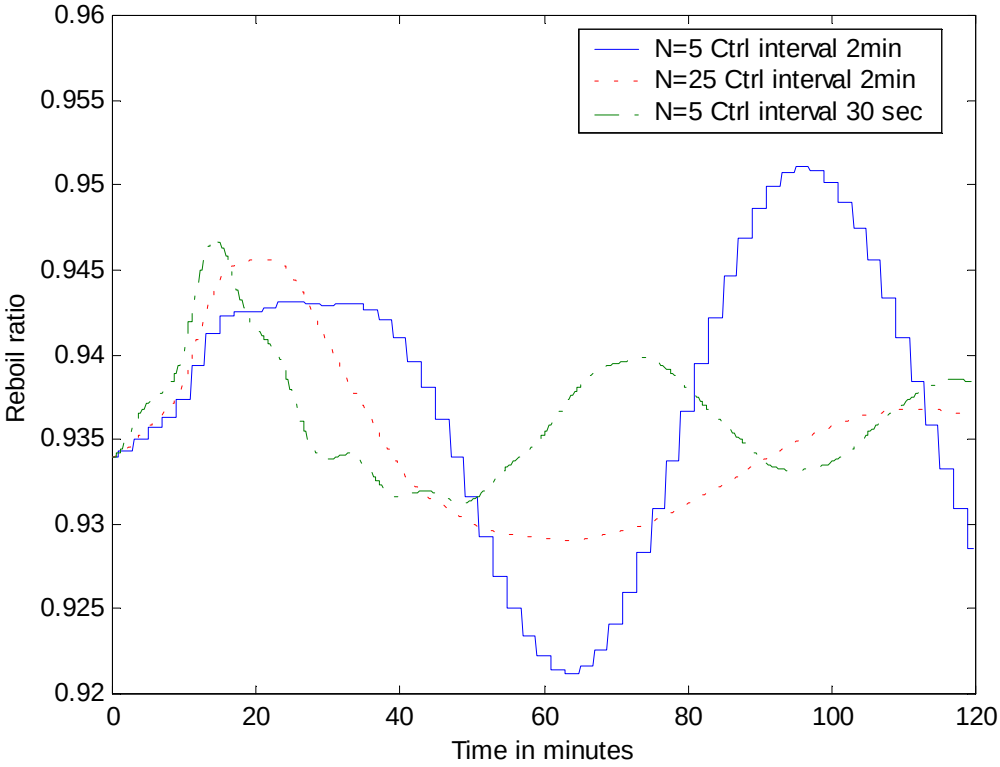



**Figure N.6 Reboil ratio for different cost horizons and control intervals when decreasing ethane molar flowrate in feed stream.**

# O COMPARISON OF GPC AND BUILT-IN MPC

## O.1 Responses manipulated variables

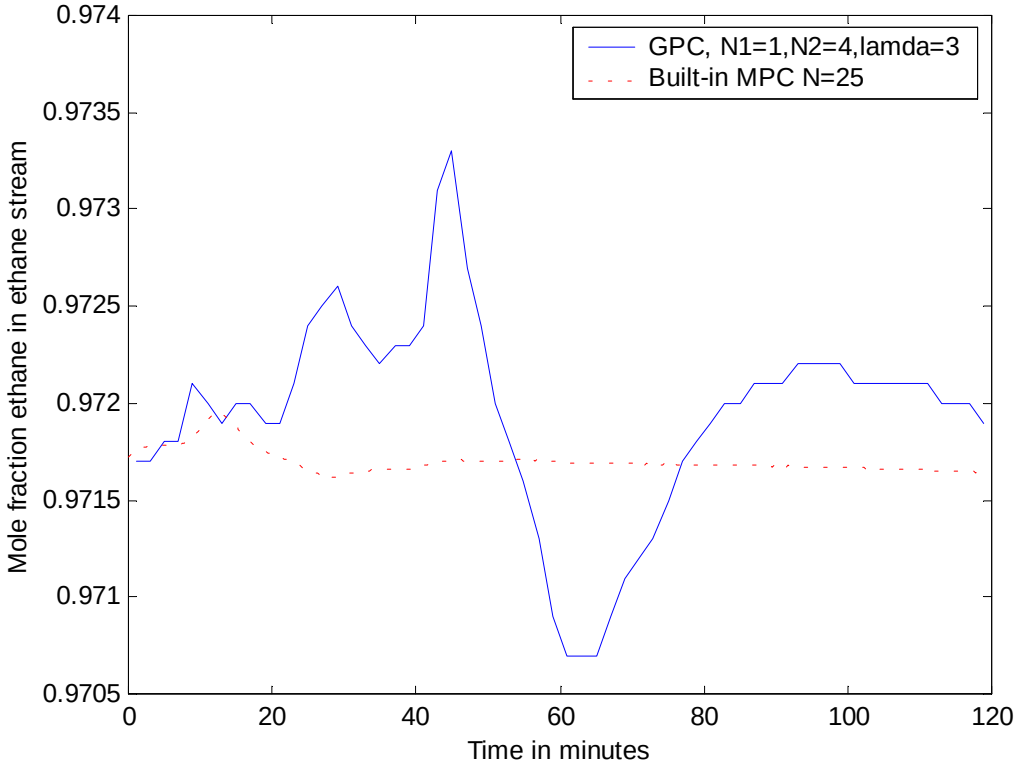



**Figure O.1 Mole fraction ethane in ethane outlet responses for the GPC and the built-in MPC when decreasing ethane molar flow in feed stream**

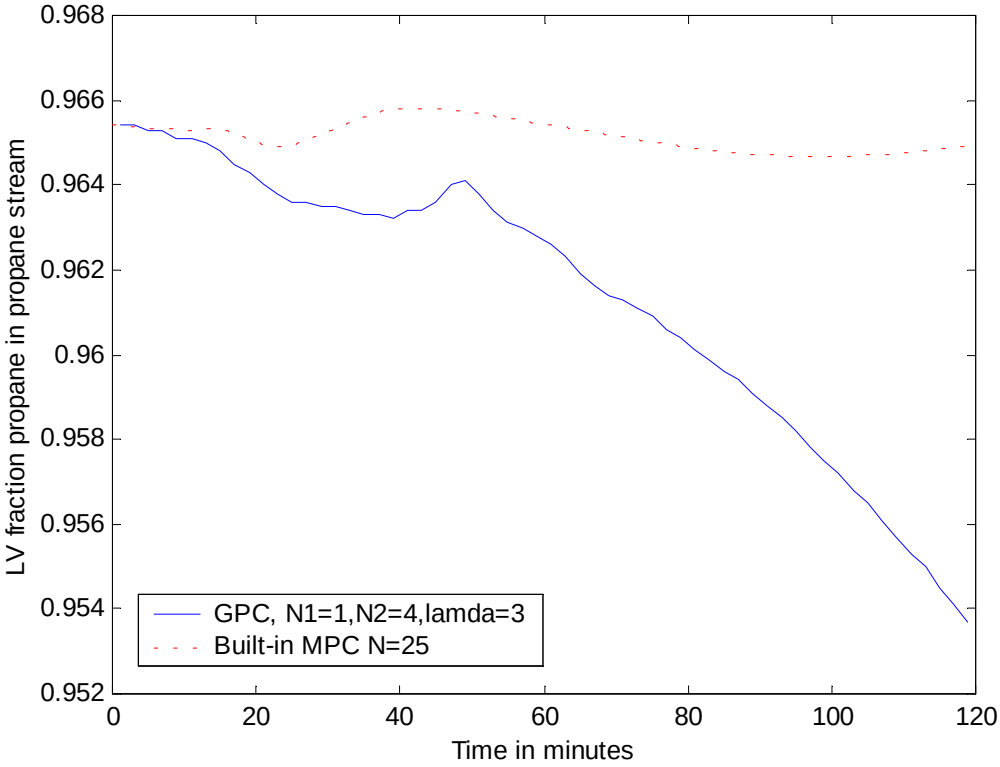



**Figure O.2 LV fraction propane in propane outlet responses for the GPC and the built-in MPC when decreasing ethane molar flow in feed stream**

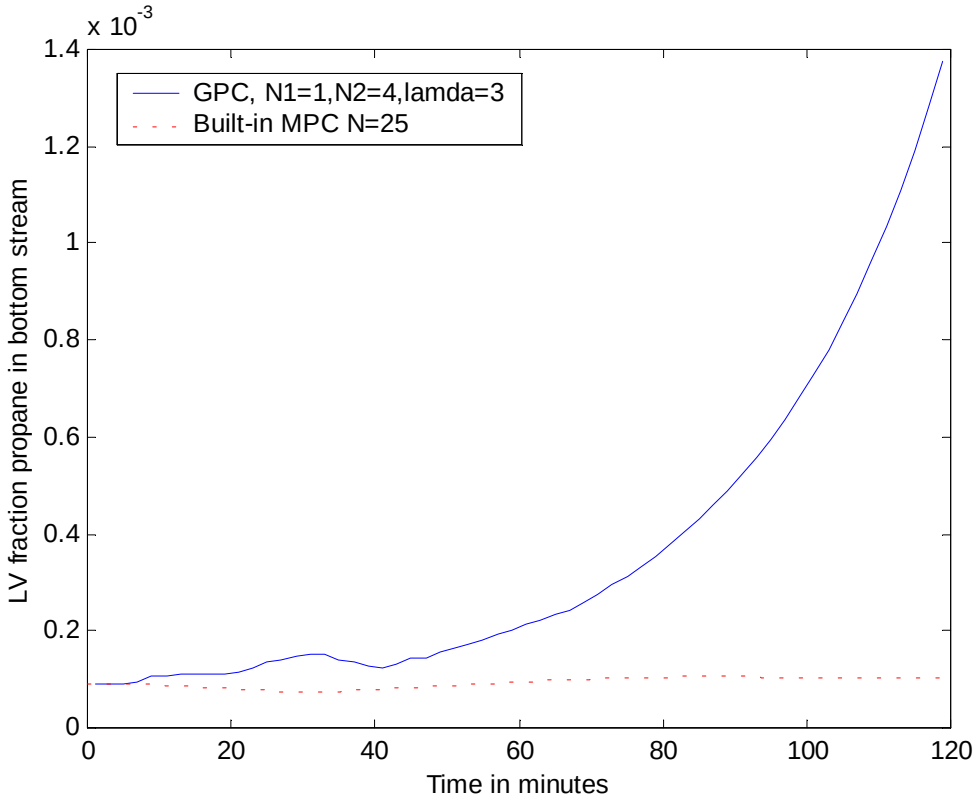



**Figure O.3 LV fraction propane in bottom stream responses for the GPC and the built-in MPC when decreasing ethane molar flow in feed stream**

## O.2 Responses in control signals

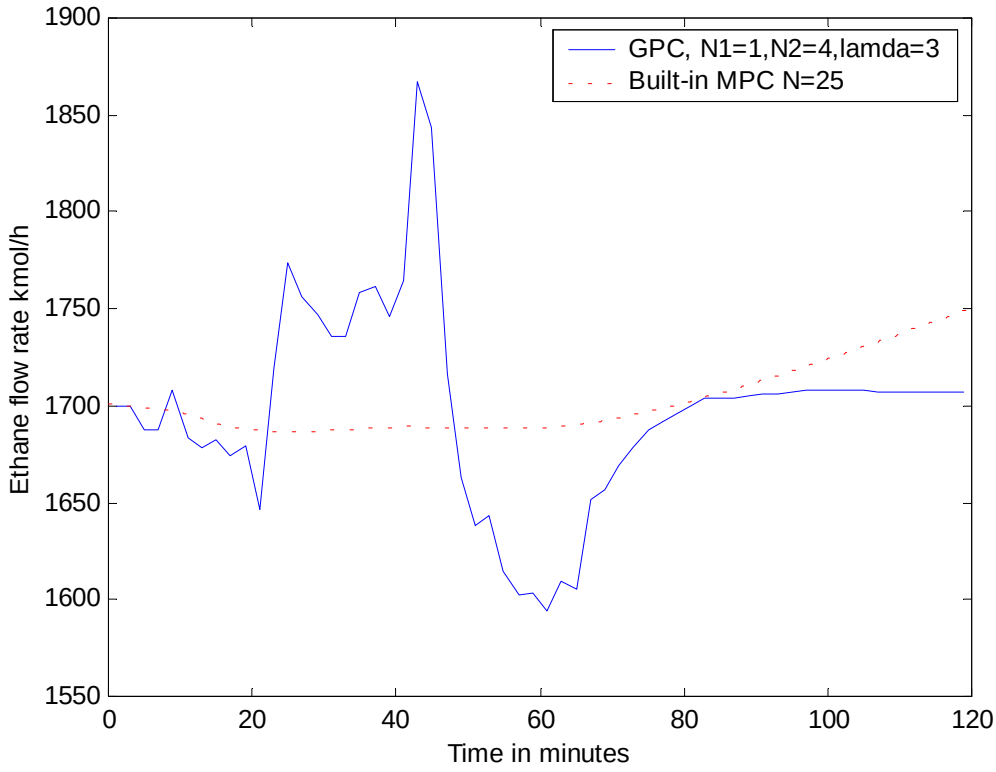



**Figure O.4 Flowrate ethane outlet responses for the GPC and the built-in MPC when decreasing ethane molar flow in feed stream.**

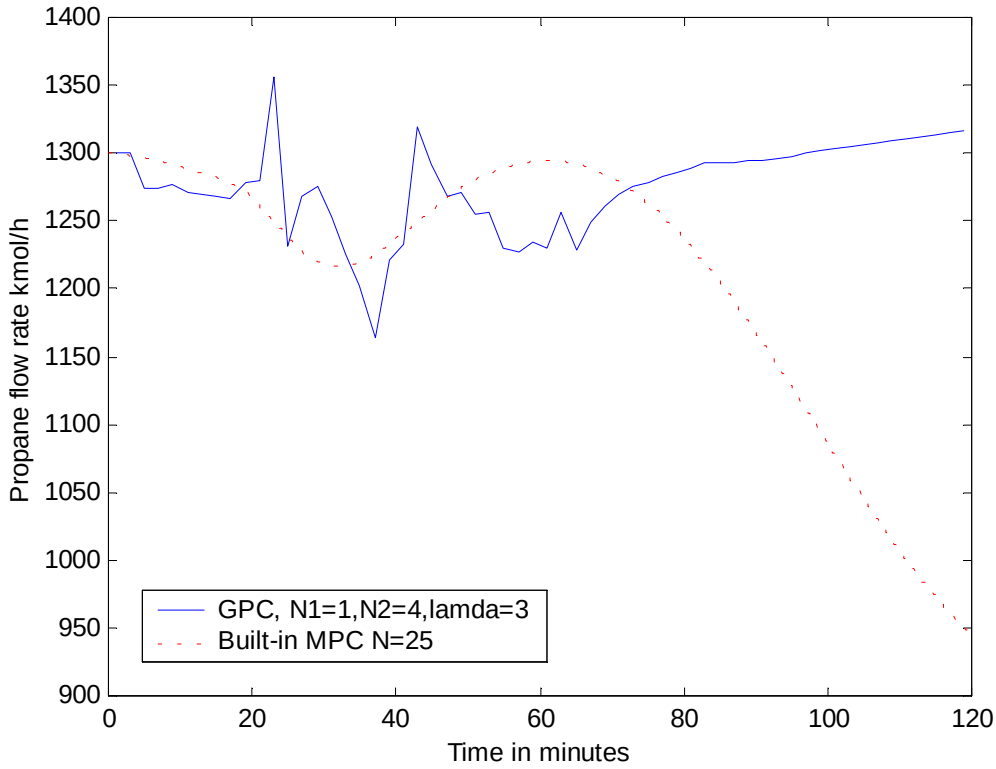



**Figure O.5 Flowrate propane outlet responses for the GPC and the built-in MPC when decreasing ethane molar flow in feed stream**

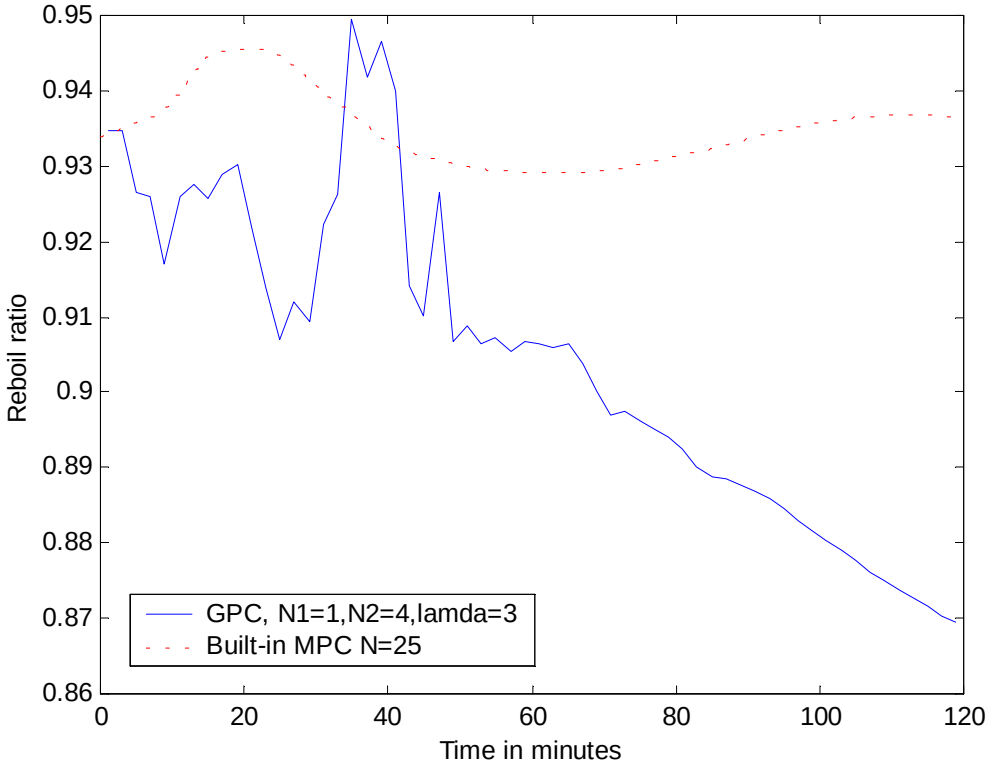



**Figure O.6 Reboil ratio responses for the GPC and the built-in MPC when decreasing ethane molar flowrate in feed stream.**

# P EXAMPLE DRIFTING IN PROCESS

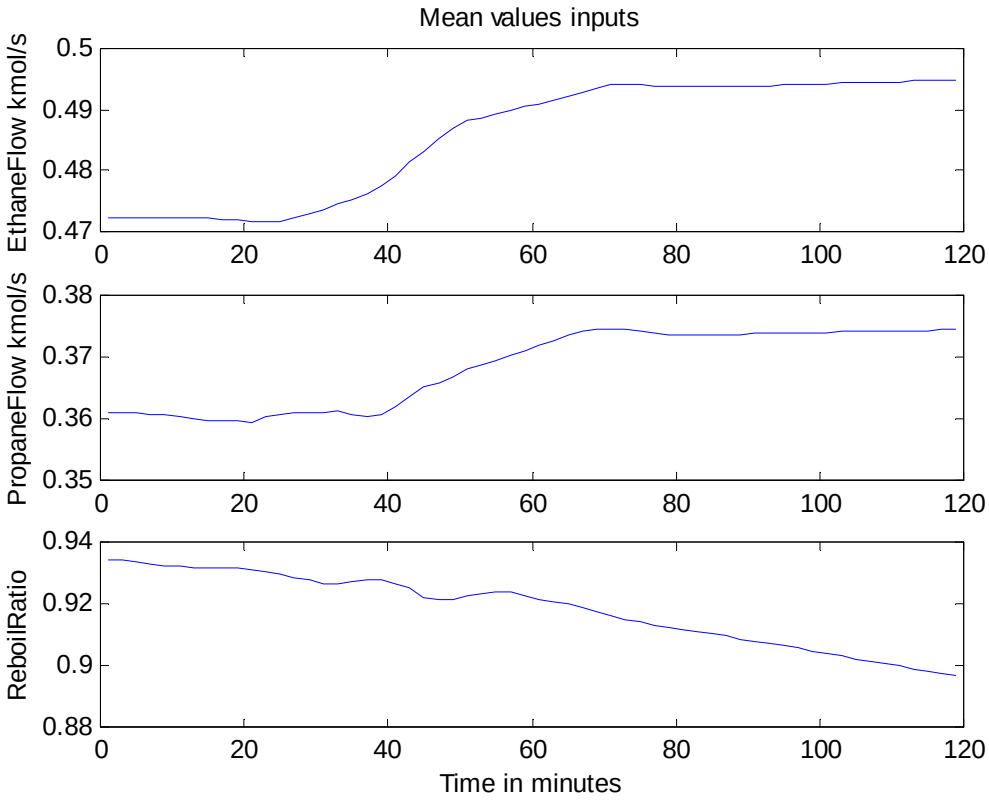



**Figure P.1 Changing in input mean values for cost horizon = 4, control horizon = 3 and control signal weighting = 3 with decreasing ethane molar flow rate in feed stream.**

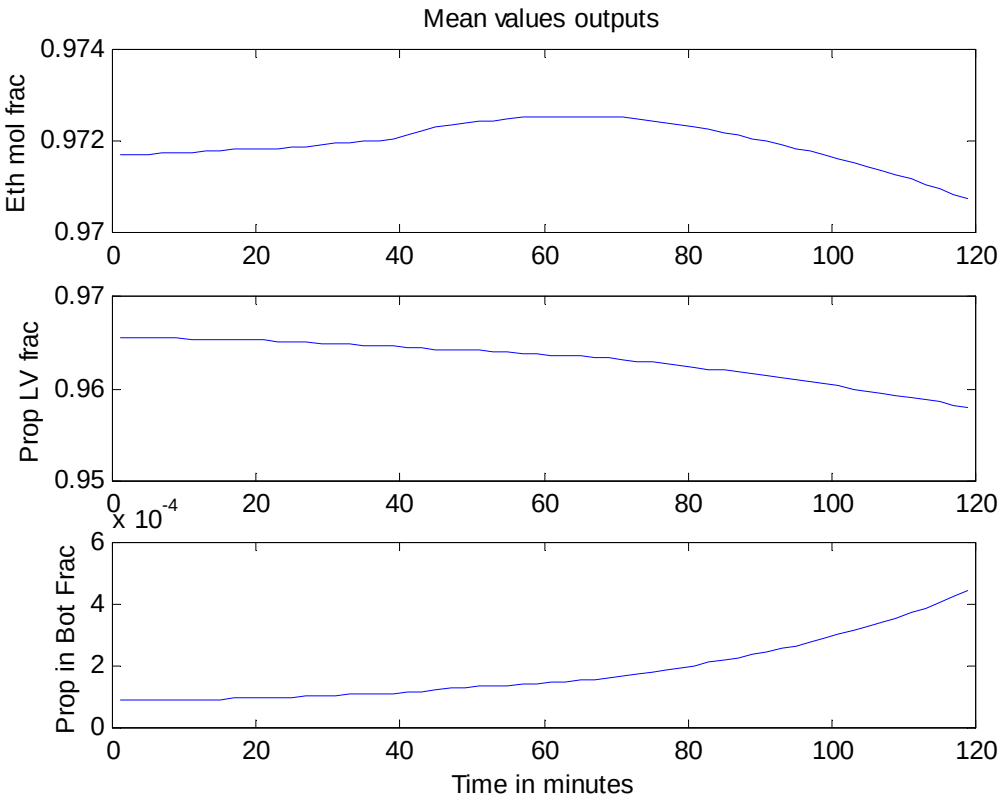



**Figure P.2 Changing in output mean values for cost horizon = 4, control horizon = 3 and control signal weighting = 3 with decreasing ethane molar flow rate in feed stream**