ISSN 0280–5316
ISRN LUTFD2/TFRT--5649--SE

A Matlab Tool for Rapid Process Identification and PID Design

Martin Andersson

Department of Automatic Control
Lund Institute of Technology
September 2000

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
September 2000

Document Number
ISRN LUTFD2/TFRT--5649--SE

Author(s)
Martin Andersson

Supervisor
Krister Forsman, ABB
Anders Wallén, Tore Hägglund

Sponsoring organisation

Title and subtitle
A Matlab Tool for Rapid Process Identification and PID Design
(Ett Matlabverktyg för snabb processidentifiering och PID-design)

Abstract

In process industry many control loops are far from optimally tuned. In this master's thesis a tool for rapid process identification and control design has been developed. The tool, which is implemented in Matlab 5.3, identifies a process model out of the step response of the real process. A PI or PID controller can then be designed and evaluated. The tool contains three different automatic design methods: Lambda tuning, which is a well known PI design method in process industry, PI design based on non-convex optimization and PID design based on constrained optimization. The two later methods are both recently developed at the Department of Automatic Control at Lund Institute of Technology.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
37

Recipient's notes

Security classification

1 Introduction

1.1 Brief background.

In today's process industry many controllers are far from optimally tuned. The reason for this is partly lack of knowledge but also lack of time. It is time demanding to make identifications of the processes and calculate new controller parameters. Tests have to be done on the processes and that means time is lost and raw material is wasted from the production. Short tests are desired to make the production disturbances as small as possible, preferably at a time when the production should change anyway. The control designs are often limited to rules of thumb. In this master thesis a tool for rapid process identification and control design has been developed. The tool is implemented in Matlab 5.3 and is an extension of a similar tool presented in Wallén (2000) and uses the same name, Transient Response Analyser, TRA.

1.2 The idea

The tool is intended to help the user to get new controller parameters out of some simple tests on the process. The figure below shows what part of the work TRA helps with.

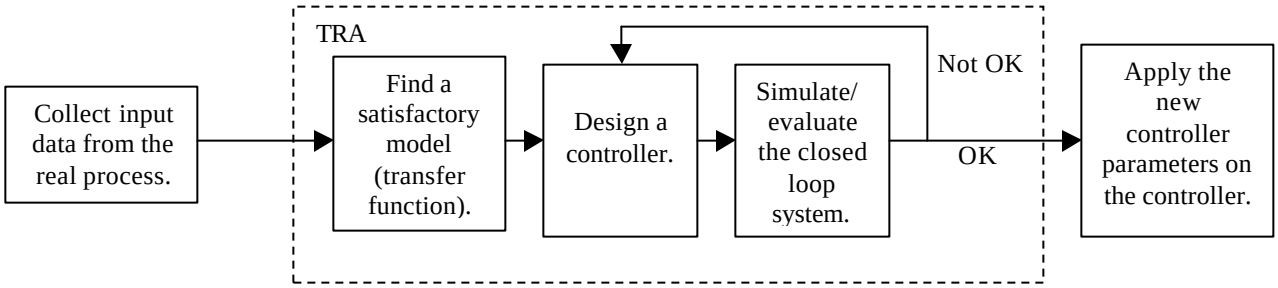

Figure 1: Overview of how to use TRA.

First data has to be collected during tests on the real process and converted into Matlab vector format. The tool identifies, with help from the user, a linear transfer function. A controller of PI/PID-type can then be designed with one of the included design methods. A simulation and an evaluation of the closed loop system can be done to determine the qualities of the controller. All calculations that otherwise would have required skills and experience in control theory are done by the tool under the user's supervision. When a satisfactory controller has been designed the user can return to the real process and enter the new controller parameters. For specific guidance regarding the tool, see User's guide.

1.3 Input data

The tests that are needed for the tool are one or several step responses, or bumps. The controller is disconnected and the control signal is adjusted manually. Under manual control one or a few steps are made on the control signal. A data logger samples both (the manual) control signal u and the process output y. The tool demands at least one step in the control signal, otherwise it won't have anything to analyse. Several step responses make the result more reliable. Generally, it is a good idea to make several steps in different regions of the measurement range. Non-linear behaviour like varying gain is then easily discovered. If the process has a strongly non-linear behaviour perhaps a more complex controller than a standard PI/PID should be used. A PID controller with gain scheduling is enough, in many cases. The tool does however not support any non-linear modelling or non-linear control design in its present status. Needed input data is the sampled control signal (u), the sampled process output (y) and a time vector (t), if the sample time is constant the vector t may be replaced by a scalar defining the sample time (Ts).

2 Process identification

Identifying a completely accurate model of a process can be, if not impossible, very time demanding. In this thesis a simple way of identifying a process model is used. It is based on step response analysis and it catches some of the most important properties of the process behaviour. The process is identified by comparing the step responses of the real process and the responses of the suggested model. The user selects an appropriate model structure. Model parameters are then adjusted until a high similarity is obtained.

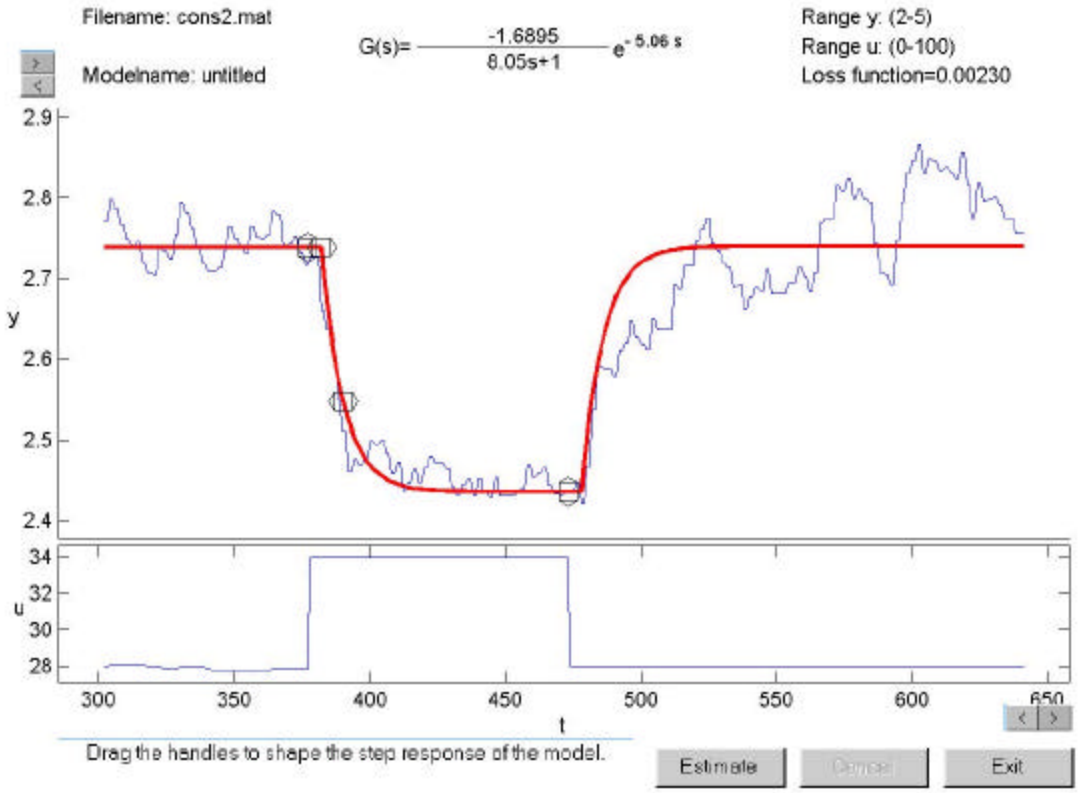

Figure 2: The main window.

The Loss function is calculated to measure how well the models step response agrees with real data. The loss function tells the average square deviation between model response and data. It is calculated as:

$$Loss\_function = \frac{\sum_{k=1}^{n} (y[k] - y_m[k])^2}{N}$$

Where y[k] is sample number k of the process output and $y_m[k]$ is the calculated model output at the same position, N is the total number of samples.
The parameter estimation can be made both manually and automatically. The automatic estimation optimises the parameters to minimise the loss function.

2.1.1 Open loop step response analysis

Analysing a single step response is in most cases not enough to identify all dynamics of a process. But since the purpose of this rapid identification is to get sufficient information to design a PI or PID controller it covers our needs. Properties that can be determined with this method are for example time delay, static gain and dominating time constant.

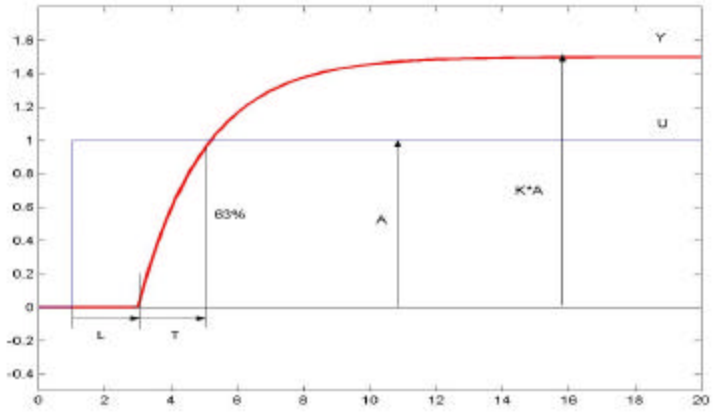

Figure 3: The step response for a model structure of type 1 with n=1 (described below). A is the amplitude of the input step, L is the process time delay, K is the static gain and T is the time constant of the process. Current parameters are L=2, T=2 and K=1.5.

2.1.2 Model Structures

The five model structures are common in process industry and cover many kinds of processes.

1. $G(s) = \frac{K}{(sT+1)^n} \cdot e^{-sL}$

2. $G(s) = \frac{K(sT_z+1)}{(sT+1)^n} \cdot e^{-sL}$

3. $G(s) = \frac{K}{s} \cdot e^{-sL}$

4. $G(s) = \frac{K}{s(sT+1)} \cdot e^{-sL}$

5. $G(s) = \frac{K(sT_z+1)}{s(sT+1)} \cdot e^{-sL}$

For model structures 1 and 2 the static gain is calculated as

$$K = \frac{y_{final} - y_0}{A}.$$

A is the amplitude of the input step.
Model structures 3 to 5 are integrating. K is then called the velocity gain instead of static gain since an integrating process doesn't reach a steady state.
The velocity gain in integrating processes is calculated as

$$K = \frac{(\frac{dy}{dt})_{final} - (\frac{dy}{dt})_0}{A}$$

i.e. the difference between initial slope and final slope normalised by the amplitude of the input step.

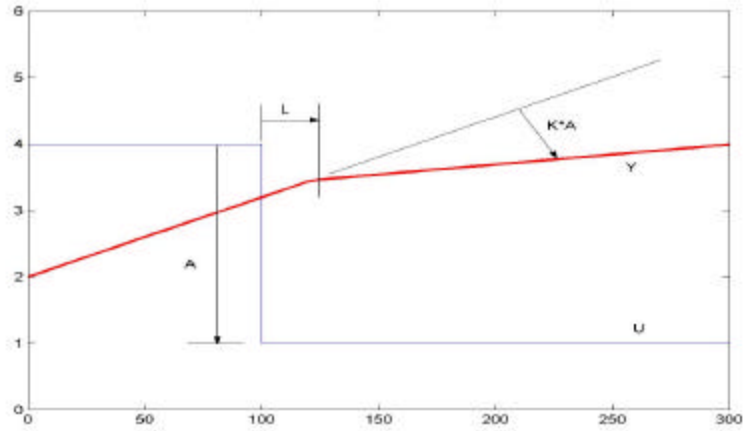

Figure 4: The step response of an integrating process of type 3, current parameters are L=20 s and K=0.03.

2.1.3 Step responses

If the input is a step with amplitude A, the output becomes

$$Y(s) = G(s) \cdot \frac{A}{s}$$

The step responses in time domain are then:

$$y_1(t) = K \cdot \left(1 - e^{-\frac{t-L}{T}} \left(\sum_{k=0}^{n-1} \frac{1}{k!} \left(\frac{t-L}{T} \right)^k \right) \right) \cdot A, \; t \geq L, \; n \geq 1$$

$$y_2(t) = K \cdot \left(1 + e^{-\frac{t-L}{T}} \left(\frac{T_z}{T \cdot (n-1)!} \cdot \left(\frac{t-L}{T} \right)^{n-1} - \sum_{k=0}^{n-1} \frac{1}{k!} \cdot \left(\frac{t-L}{T} \right)^k \right) \right) \cdot A, \; t \geq L$$

$$y_3(t) = K \cdot (t-L) \cdot A, \; t \geq L$$

$$y_4(t) = K \cdot \left(t - L - T \cdot \left(1 - e^{-\frac{t-L}{T}} \right) \right) \cdot A, \; t \geq L$$

$$y_5(t) = K \cdot \left(t - L - (T - T_z) \cdot \left(1 - e^{-\frac{t-L}{T}} \right) \right) \cdot A, \; t \geq L$$

In the equations above it is assumed that the initial value is zero and that the system is at rest before the step. This is too much to expect when dealing with real processes. The step response is added to the initial value to get the models true output signal. For a model of type 1 the complete output $y_m(t)$ after an input step will be:

$$y_m(t) = y_0 + y_1(t).$$

For an integrating model structure, such as structure 3, the expression is slightly more complicated. Now both the initial value and the initial slope have to be considered,

$$y_m(t) = y_0 + k_0 \cdot t + y_3(t).$$

If the control signal (u) contains more than one step, the responses are simply added. The response to each single step in the control signal is calculated as if the initial values (initial output value and initial slope) were zero. The separate responses are then added with the initial conditions to become the complete model output.

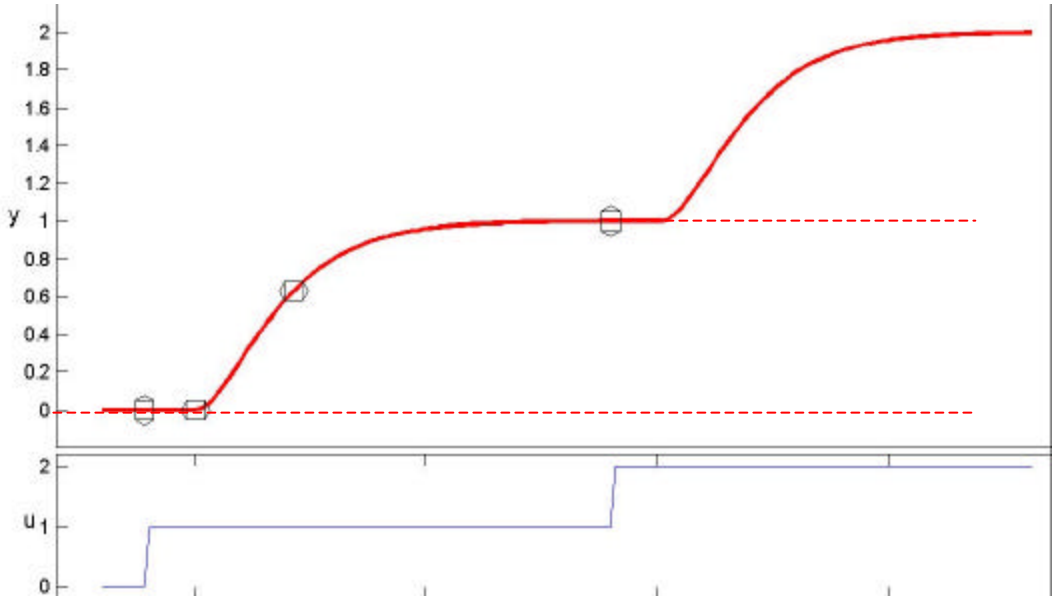

Figure 5: The lower dashed line shows the initial value y_0 and the upper dashed line shows the final level after the first step.

In the figure it is easy to see that multiple input steps only result in adding the step responses.

$$y_m(t) = y_0 + y_1(t)^{step1} + y_1(t)^{step2} + \ldots$$

Integrating models such as type 3 will consequently get the output

$$y_m(t) = y_0 + k_0 \cdot t + y_3(t)^{step1} + y_3(t)^{step2} + \mathrm{K}\ .$$

3 Control design

A basic description of a control system is shown below.

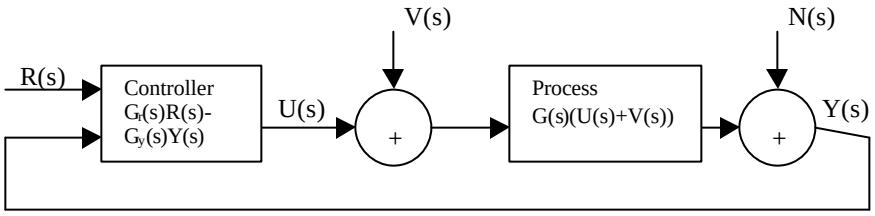

Figure 6: A standard control system.

The two extra inputs V(s) and N(s) represent load disturbances and measurement noise respectively.

3.1 Only PID controllers

Only PI and PID controllers can be designed with the tool. This choice was made since the majority of all controllers in process industry are of PI or PID-type and one of the major employment areas for the tool is to calculate new parameters to already existing controllers.

Three different design methods are used; two different PI-designs and one PID design. The PID design and one of the PI designs share design problem formulation. These two strongly connected design methods are based on constrained optimization. The last method, a PI-design called lambda tuning is because of its simplicity very common in process industry. Each method has a design parameter. The design parameter somehow lets the user decide the trade off between stability and performance. A PI controller is described by

$$u(t) = K_c\left(b \cdot r(t) - y(t)\right) + \frac{K_c}{T_i}\int\left(r(t) - y(t)\right)dt$$

in time domain and have the Laplace transform

$$U(s) = K_c(b \cdot R(s) - Y(s)) + \frac{K_c}{T_i \cdot s} \cdot (R(s) - Y(s))$$

The controller parameters to design are K_c, T_i, b.

A PID controller is described by

$$u(t) = K_c\left(b \cdot r(t) - y(t)\right) + \frac{K_c}{T_i}\int\left(r(t) - y(t)\right)dt + K_c \cdot T_d \cdot \left(-\frac{dy(t)}{dt}\right),$$

with Laplace transform

$$U(s) = K_c(b \cdot R(s) - Y(s)) + \frac{K_c}{T_i \cdot s} \cdot (R(s) - Y(s)) + K_c \cdot T_d \cdot s \cdot Y(s)$$

and have an additional design parameter T_d.

b is the set point weight. Unfortunately many control systems don't have the possibility to adjust b, then b=1 or b=0. Normal values are $0 \le b \le 1$. A low b-value gives a slow response to set point changes. The advantage is decreased amplitude on the control signal. If a fast set point response is required b is set to one or more.

3.2 Constrained/Non-Convex optimization

These are two recently developed control design methods based on constrained/non-convex optimization, Panagopoulos (2000). The two methods share design problem formulation. In the tool these methods are referred to as PI_opt and PID_opt.

The goal of the design is a well-damped, robust system with good rejection of load disturbances. Robust means that the system has low sensitivity to modelling errors and changes in process behaviour.
One way to measure a systems ability to reject disturbances is to calculate the integrated error, IE after a step load disturbance.

$$IE = \int \big(r(t) - y(t)\big)dt$$

A small IE means no steady state error (if the control error doesn't change sign) and fast rejection of load disturbances. It has been shown that IE is minimised by maximising k_i ($= K_c/T_i$). The disadvantage with IE is that a harmonic oscillation around the set point is invisible. An oscillating output seems perfect if IE is the only measure since the positive error compensates the negative error.

An additional constraint is needed to make sure that the closed loop system stays within desired stability margins. The stability of a closed loop system can be viewed in the Nyquist curve of the system, from now on called Nc. Nc is the curve $G_0(i\omega)$ for all real w. $G_0(i\omega)$ is called the loop transfer function and is explained thorough further down. If Nc encircles the point –1 on the real axis the system is unstable. The further away from –1 Nc passes the more stable the system is.

3.2.1 The design variable Ms

The added constraint is a measure of how close to –1 Nc may pass. 1/Ms is the shortest distance between Nc and the point –1. It can be viewed as the radius of a circle with centre in –1, which Nc must avoid.

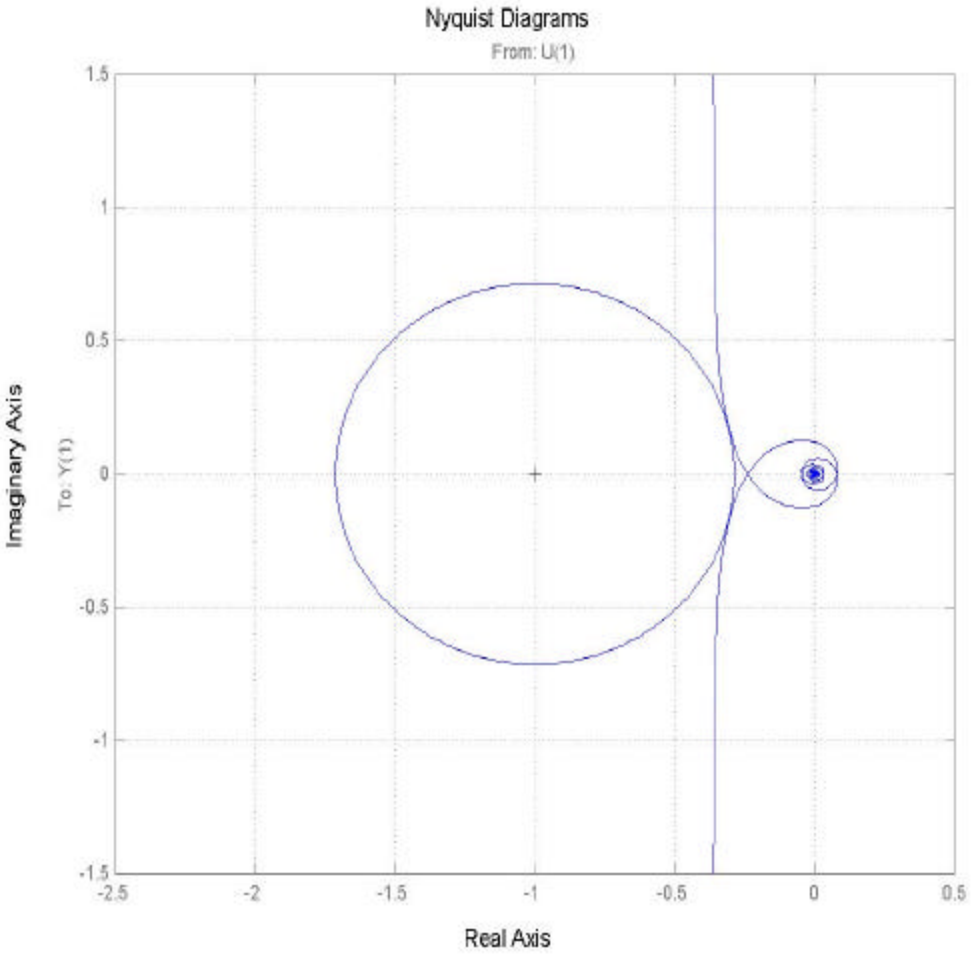

Figure 7: Nyquist curve and circle with centre in –1 and radius 1/Ms. Ms is the maximum gain of the sensitivity function and is used as design variable.

Typical values for Ms are 1.2-2.0. Low Ms values gives robust systems, i.e. systems with low sensitivity to modelling errors.

3.2.2 Optimization

To put it short, the optimization problem is to maximise ki (= K/Ti), while keeping the Nyquist curve outside the circle with centre in –1 and radius 1/Ms. In mathematical terms:

$$\max(k_i)$$

$$\left\| -1 + G_0(iw) \right\| \geq \frac{1}{M_s}$$

3.3 Lambda tuning

This method can only be used to design PI-controllers for two different process structures. One is called a self-regulating process and is the same as model structure type 1 with n=1. The other is an integrating process, the same as model structure type 3.
The transfer functions are:

$$G(s) = \frac{K_p}{sT+1} e^{-sL}$$ for the self-regulating process and

$$G(s) = \frac{K_v}{s} e^{-sL}$$ for the integrating process.

3.3.1 PI controller for self-regulating processes

For a self-regulating process lambda (λ) is the time constant of the desired closed loop system. In other words λ indicates how fast the process value will reach 63% of a set point change.

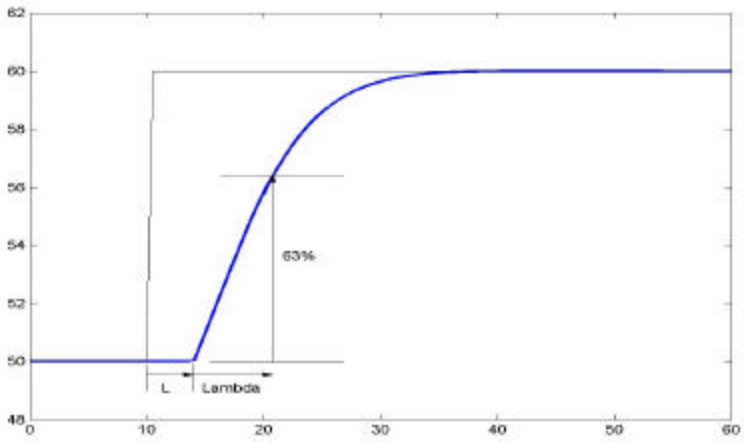

Figure 8: l for self-regulating processes.

In this example the process has a time delay L=4 s and is controlled by a controller designed with λ = 8.

λ is selected with respect to L and T. This is where the indirect design-parameter τ enters. λ is selected as:

$$l = t * \max(L, T) \qquad (1)$$

τ decides the trade off between stability and speed for the closed loop system. Normal values are $2 \leq \tau \leq 3$.

$\tau > 3$.	Very stable but very slow design.
$2 \leq \tau \leq 3$.	Normal design, stable if process model is known with some accuracy.
$1 \leq \tau < 2$.	Fast design, stable if the process model is known with high accuracy.
$\tau < 1$.	Faster, but even a small deviation between process model and real process may give an unstable system.

Once λ is determined, the controller parameters Kc and Ti are easily calculated through

$$T_i = T \qquad (2)$$

$$K_c = \frac{T_i}{K_p(l+L)} \qquad (3)$$

3.3.2 PI controller for integrating processes

For an integrating process λ indicates another measure than above. Here λ is defined as the time elapsed, after the entrance of a step load disturbance, until the output error has reached its maximum and starts to decrease.

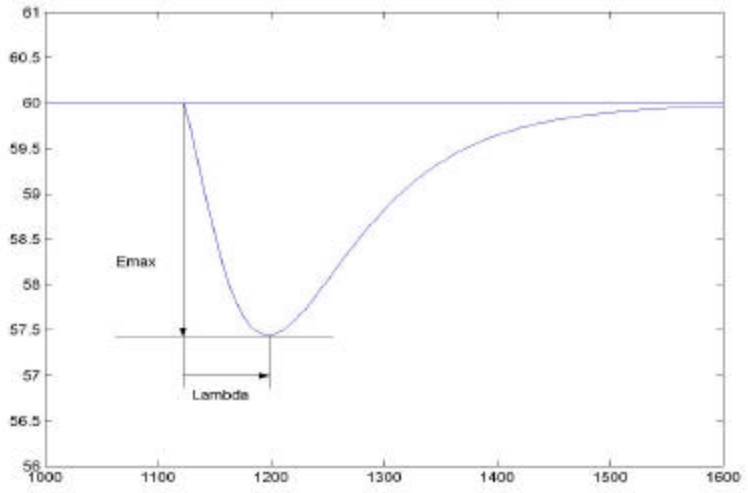

Figure 9: l for integrating processes.

In this type of processes λ is determined by the control demands on the process, λ may be chosen as:

$$l = \frac{e_{\max}}{100 \cdot K_v} \qquad (4)$$

Where e_{max} is the most extreme output error that may occur during normal use, 100 * K_v says how much the process output at the most can change in one second (with a control signal at 100%).
If a tank level has the reference value 75% and the tank is full at 100%, e_{max} = 25%. Then $\lambda = 25/(100*K_v)$.

When λ is determined, control parameters Ti and Kc are calculated through

$$T_i = (2 \cdot l) + L \qquad (5)$$

$$K_c = \frac{T_i}{K_v (l + L)^2} \qquad (6)$$

4 Controller performance assessment

If a controller has been designed with one of the methods above it is most likely stable, but to get a clearer insight in the closed loop behaviour and to compare the qualities of different controllers it is useful to make some kind of evaluation. The tool has three different functions.

1. Simulate closed loop system
2. Plot sensitivity functions
3. Evaluate closed loop system

Number 3 makes calculations on the simulation result, the sensitivity functions and the loop transfer function.
A comparison between different design methods and design parameters is done at the end of this chapter using the Evaluate part of the tool.

4.1 Simulation of closed loop system

A simple simulation can be done in the tool to see how well the closed loop system behaves. During the simulation the system is exposed to a set point change and a step load disturbance.

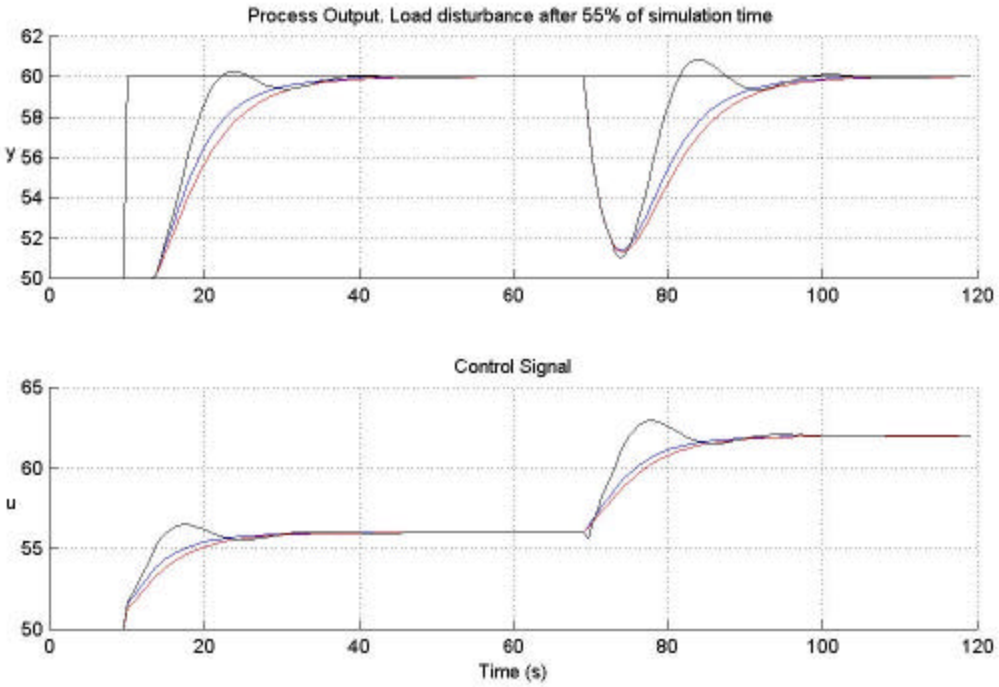

Figure 10: Simulation with set point change at t=10 s and a step load disturbance at t = 70 s.

The simulation gives a clue of how well the controller handles disturbances and set point changes, but the results are only true if the estimated process model is a good approximation of the real process. In other words the simulation should mainly be used to compare different controllers, not to predict exact performance of the real closed loop system. If the process model is badly chosen the simulation and following evaluation are meaningless.

4.2 Sensitivity functions

A number of sensitivity functions can be calculated to describe the properties of a control system. A sensitivity function tells how an input signal affects the process output. That description also fits the closed loop transfer function, but then we specifically mean how the *reference* signal affects the output. The other input signals are not real inputs but theoretical inputs that represents disturbances acting on the process V(s) and measurement noise in the sensor at the process output N(s). By investigating the sensitivity function from V(s) to Y(s) it is possible to tell how sensitive the system is to load disturbances. The sensitivity function from N(s) to Y(s) tells how sensitive the system is to noise and to modelling errors.

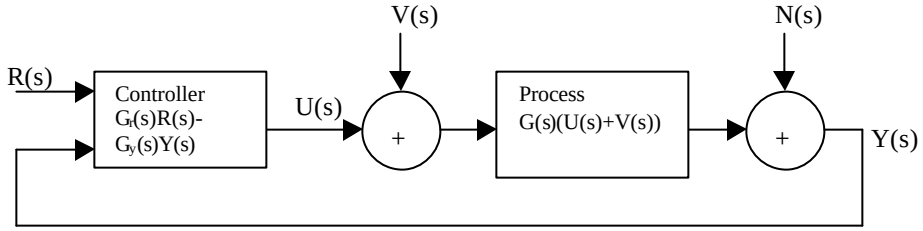

Figure 11: Standard control system.

$G_r(s)$ is the controllers transfer function from R to U and $G_y(s)$ is its transfer function from Y to U. In the most basic versions of PID-controllers such as PI_lambda $G_r(s) = G_y(s)$.

The closed loop transfer function is calculated by putting V and N to zero.

$$Y(s) = G(s) \cdot U(s) = G(s) \cdot \left(G_r(s)R(s) - G_y(s)Y(s)\right) \Leftrightarrow Y = \frac{G(s)G_r(s)}{1 + G(s)G_y(s)} \cdot R(s)$$

The sensitivity function from V to Y is calculated by putting R and N to zero.

$$Y(s) = G(s) \cdot \left(U(s) + V(s)\right) = G(s) \cdot \left(-G_y(s)Y(s) + V(s)\right) \Leftrightarrow Y(s) = \frac{G(s)}{1 + G(s)G_y(s)} \cdot V(s)$$

The sensitivity function from N to Y is calculated by putting R and V to zero.

$$Y(s) = G(s) \cdot \left(U(s)\right) + N(s) = G(s) \cdot \left(-G_y(s)Y(s)\right) + N(s) \Leftrightarrow Y(s) = \frac{1}{1 + G(s)G_y(s)} \cdot N(s)$$

Apparently all functions have the same denominator.

$$1 + G(s) \cdot G_y(s) = 1 + G_0(s)$$

$G_0(s)$ is called the loop transfer function and can be viewed in a Bode plot or a Nyquist diagram where it is easy to decide if the system is stable. More discussions about stability and loop transfer function further down.

4.3 Evaluate

To evaluate the closed loop system in a more scientific way than just looking at the simulation results the tool contains the Evaluate function. A few important measures concerning load disturbance rejection and stability margins are here calculated.

4.3.1 Evaluation of load disturbance rejection

For a servo system the responses to set point changes would be most interesting, but the tool is developed with regulator systems in mind. For those the load disturbance rejection is of primary concern. All calculations on load disturbance rejection are done on the simulation results after the entrance of the step load disturbance. Four measures are calculated and displayed.

1. IAE
2. E_{max}
3. $u_{overshoot}$
4. Var(u)

The Integrated Absolute Error, IAE is calculated as

$$IAE = \int \left| r(t) - y(t) \right| dt \, .$$

It indicates how well the controller corrects the process output after a step load disturbance, but it doesn't tell the actual size of the error or how long it takes to correct it.

E_{max} is the maximum error,

$$\max\left(e(t)\right) = \max\left(\left| r(t) - y(t) \right|\right).$$

It is of great interest to get small E_{max} since it is a natural quality measurement. A fast compensation for a load disturbance could mean the difference between acceptable and unacceptable quality.

The two remaining measures concern the behaviour of the control signal. It is desirable to have a smooth control signal without swinging or large overshoots. A nervous control signal will increase the wear on the actuators and consumes more energy than a smooth signal. If the control signal becomes too large it may saturate. That gives a non-linear behaviour, which is not expected and not very well handled by an ordinary PID controller.
The normalised overshoot of the control signal is calculated as

$$u_{overshoot} = \frac{|u_{\max} - u_0|}{|u_{final} - u_0|} - 1.$$

Figure 12: Control signal with overshoot.

The variance of u is calculated as

$$\mathrm{var}(u) = \frac{1}{T}\int_0^T \left(u(t) - \bar{u}\right)^2 dt$$

A large variance off course tells that the control signal is varying a lot. Oscillations are easily revealed when inspecting the variance.

4.3.2 Stability margins

The stability margins are obtained from the bode plot of the loop gain.

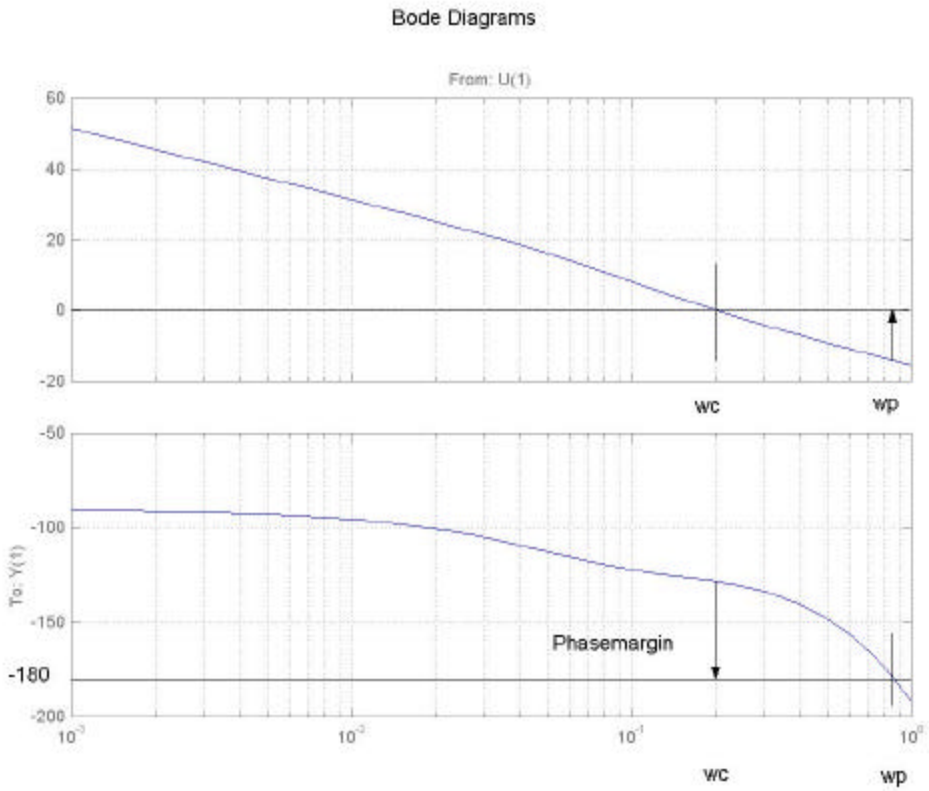

Figure 13: Definition of phase margin and amplitude margin.

ω_c, the (gain) crossover frequency, is the frequency where $|G_0| = 1$. ω_p, the phase-crossover frequency, is the frequency where $\varphi = (-180°)$. The phase margin is calculated as

$$\varphi_m = \varphi(\omega_c) - (-180°)$$

and the amplitude margin is

$$A_m = \frac{1}{\left|G_0\left(iw_p\right)\right|}.$$

When the phase margin is known it is possible to calculate a delay margin. It tells how much the process delay may increase before the system becomes unstable. This is useful since the delay can vary a lot in some processes. If the phase margin is expressed in radians the delay margin is simply $L_m = \frac{j_{m,r}}{w_c}$.

Phase margins are however commonly given in degrees. The delay margin then becomes

$$L_m = \frac{j_m \cdot p}{w_c \cdot 180}.$$

4.4 Comparison between Lambda tuning, PI_opt and PID_opt

4.4.1 Why?

PI_opt and PID_opt are as mentioned earlier two new design methods for PI and PID-controllers. As the most common control design method in the paper industry is Lambda tuning it would be interesting to see how large the improvements are when using the new methods.

4.4.2 How?

Lambda tuning is only defined for two model structures, the first order pole and the integrating structure. This is a great disadvantage since more complex models better describe many processes. Not only Lambda tuning has limitations regarding models, PI_opt and PID_opt requires for example that the process model have a monotonously decreasing phase.
The comparison is made on data from four real processes. The best possible model is the one with least loss function. This model is considered to be the true one and is used during the simulation and evaluation regardless which model was used in the control design. If the model used in the simulation isn't supported by a design method the best of the supported models is used instead.
During simulations the system is exposed to a set point change and a step load disturbance with amplitude 10.

4.4.3 Process one

Data from file: flow2 (water flow 0-100 litres/second).
The process model used in the simulation could also be used in the "opt-designs"

$$G(s) = \frac{1.6592}{(1.07s+1)^3} \cdot e^{-2.40s}.$$

Loss function = 0.0935.
The model used in lambda tuning was

$$G(s) = \frac{1.6637}{2.13s+1} \cdot e^{-3.60s}$$

Loss function = 0.1110.
Simulation time = 200 s.

Measure /controller	Controller parameters	IAE	E_{max}	$u_{overshoot}$	Var(u)	φ_m (°)	L_m (s)	L_{max} (s)	A_m	Ms
Lambda λ=4 τ =1	Kc=0.1682, Ti=2.13	78.11	8.63	0.02	2.41	64	8.7	11.1	3.23	1.57
Lambda λ=8 τ =2	Kc=0.1103, Ti=2.13	116.6	8.91	0	2.74	73	15.0	17.4	4.93	1.33
Lambda λ=12 τ =3	Kc=0.0820, Ti=2.13	156.7	9.05	0	3.01	77	20.9	23.3	6.62	1.23
PI_opt Ms=1.6	Kc=0.1879, Ti=2.283	74.67	8.61	0.02	2.34	63	7.7	10.1	2.99	1.60
PI_opt Ms=1.4	Kc=0.1464, Ti=2.394	98.52	8.76	0	2.54	71	12.1	14.5	4.12	1.40
PI_opt Ms=1.2	Kc=0.0886, Ti=2.569	174.5	9.05	0	3.00	80	23.7	26.1	7.04	1.20
PID_opt Ms=1.6	Kc=0.349, Ti=2.62 Td=1.23	54.84	7.73	0.08	1.82	55	4.5	6.9	2.80	1.62

PID_opt Ms=1.4	Kc=0.266, Ti=2.74 Td=1.22	65.16	8.02	0.02	1.96	65	7.3	9.7	3.77	1.41
PID_opt Ms=1.2	Kc=0.156, Ti=2.95 Td=1.19	113.8	8.41	0	2.47	77	15.0	17.4	6.4	1.20

The results for the two PI methods are very similar. The extra complexity in the model used to design PI_opt didn't make any large improvements compared to Lambda tuning. PID_opt has better simulation results but less stability margins.

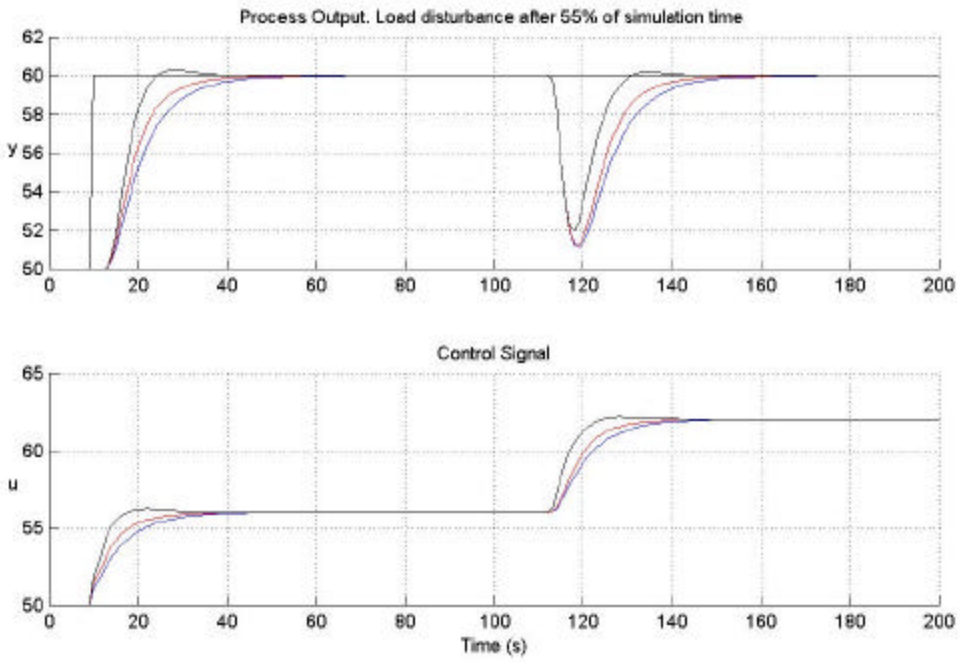

Figure 14: Simulations with three different controllers, PID_opt and PI_opt with Ms=1.4 and Lambda with t=2. Fastest response is received with PID_opt, slowest with Lambda and PI_opt in the middle.

4.4.4 Process two

Data from file: lev6. Level in a condensate vessel.
The process model used in the simulation is the same as the one used in PI_opt design:

$$G(s) = \frac{-0.00652}{s(49.33s+1)} \cdot e^{-2.00s}.$$

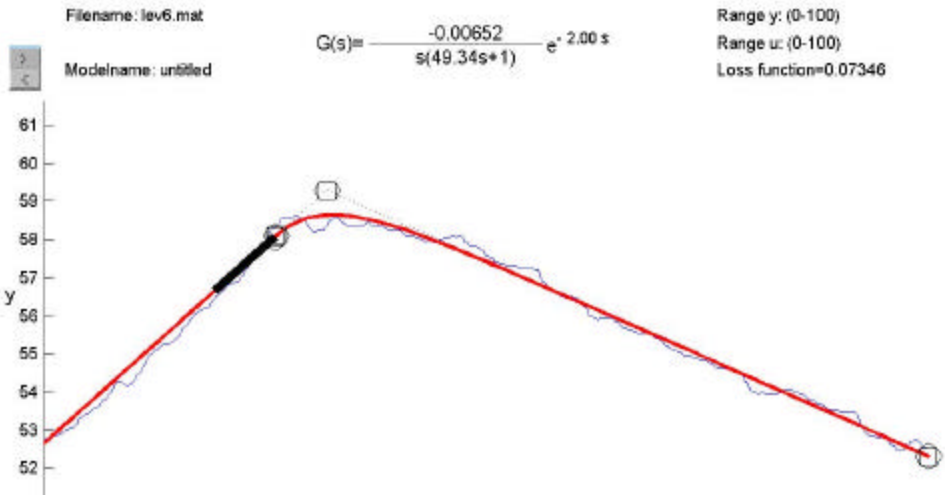

Figure 15: Integrating model with pole.

This model can for some reason not be used in the PID_opt design for Ms values between 1.2-2.0. The model used in lambda tuning and PID_opt is

$$G(s) = \frac{-0.00641}{s} \cdot e^{-44.37s}$$

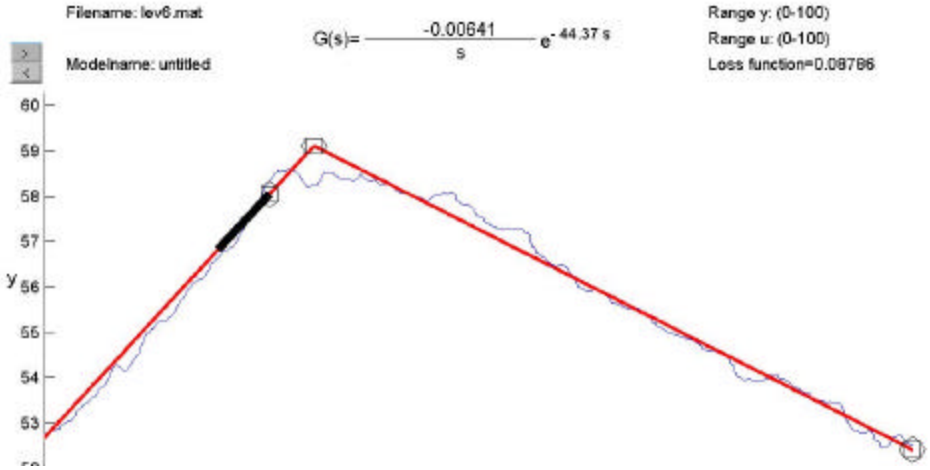

Figure 16: Pure integrating model.

Simulation time = 5000 s.

Measure /controller	Controller parameters	IAE	E_{max}	$u_{overshoot}$	Var(u)	φ_m (°)	L_m (s)	L_{max} (s)	A_m	Ms
Lambda λ=100	Kc=-1.83, Ti=244	1335	4.86	0.393	3.98	40	63	65	33.6	1.64
Lambda λ=150	Kc=-1.423, Ti=344	2420	6.08	0.304	4.32	47	94	96	47.3	1.44
Lambda λ=300	Kc=-0.848, Ti=644	7543	9.60	0.194	5.75	58	184	186	86.9	1.24
PI_opt Ms=1.6	Kc=-2.009, Ti=288	1432	4.64	0.358	3.63	42	63.4	65.4	30.6	1.60

PI_opt Ms=1.4	Kc=-1.298, Ti=373	2870	6.56	0.286	4.53	49	101	103	51.9	1.40
PI_opt Ms=1.2	Kc=-0.6246, Ti=628	10190	12.0	0.228	7.33	57	237	239	118	1.20
PID_opt Ms=1.6	Kc=-2.1, Ti=110 Td=23.6	960	3.40	0.450	3.40	34	46.6	48.6	73.8	1.71
PID_opt Ms=1.4	Kc=-1.59, Ti=152 Td=22.9	1464	4.49	0.397	3.82	40	68.4	70.4	99.1	1.49
PID_opt Ms=1.2	Kc=-0.925, Ti=282 Td=21.4	3829	7.58	0.310	5.24	49	134	136	188.8	1.25

The calculated Ms-values for systems with PID_opt controllers are far from the desired ones, 1.25, 1.49 and 1.71 instead of 1.2, 1.4 and 1.6. This happens when a model not equal to the "real one" is used during the design. In other respects the results are not surprising, what is won in performance is lost in stability.

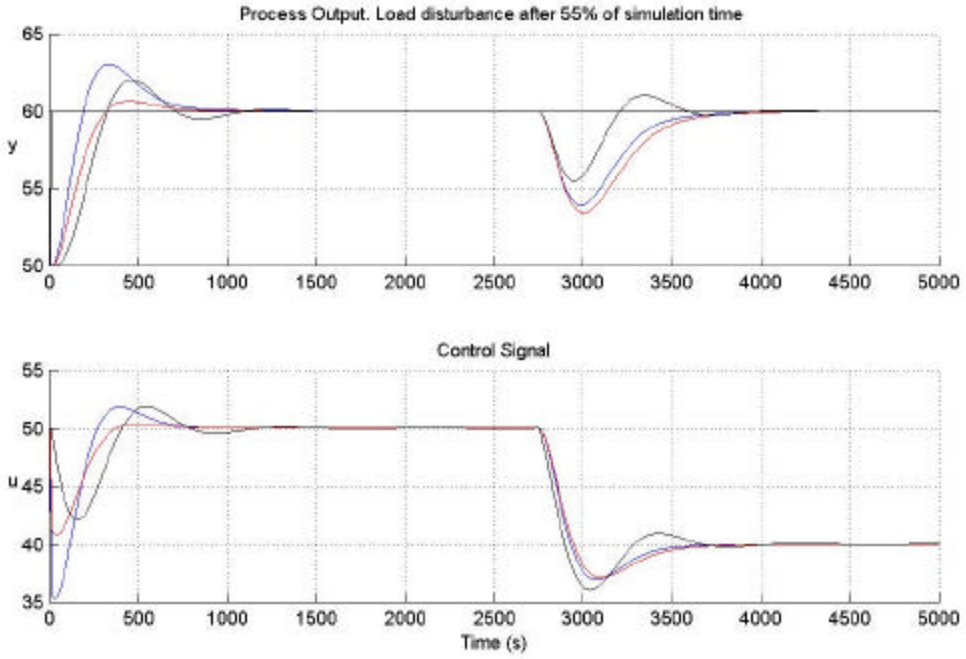

Figure 17: Simulations made with PI/PID_opt, Ms = 1.4 and Lambda with l=150. PID is fast but not as stable as the other two. PI_opt is the slowest but has largest stability margins. Lambda is in the middle.

4.4.5 Process three

Data from file: dp2, differential pressure across a steam cylinder.
The process model used in the simulation is

$$G(s) = \frac{0.3292(28.978s+1)}{(8.58s+1)^3} \cdot e^{-2.02s}.$$

Loss function = 0.3499.
This model cannot be used in PI_opt and PID_opt since the phase is not monotonously decreasing ($T_z > T$). The second order model worked although the transfer function at a quick glance looks even worse, but the larger time delay compensates and presses the phase down.

$$G(s)=\frac{0.3104(39.251s+1)}{(14.28s+1)^2}\cdot e^{-4.44s}.$$

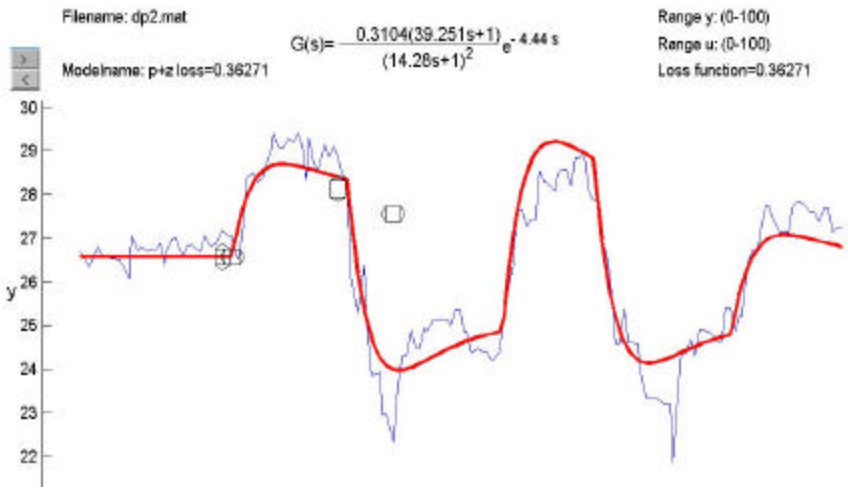

Figure 18: 2:nd order pole with zero.

The model used in lambda tuning is

$$G(s)=\frac{0.4483}{5.74s+1}\cdot e^{-3.46s}.$$

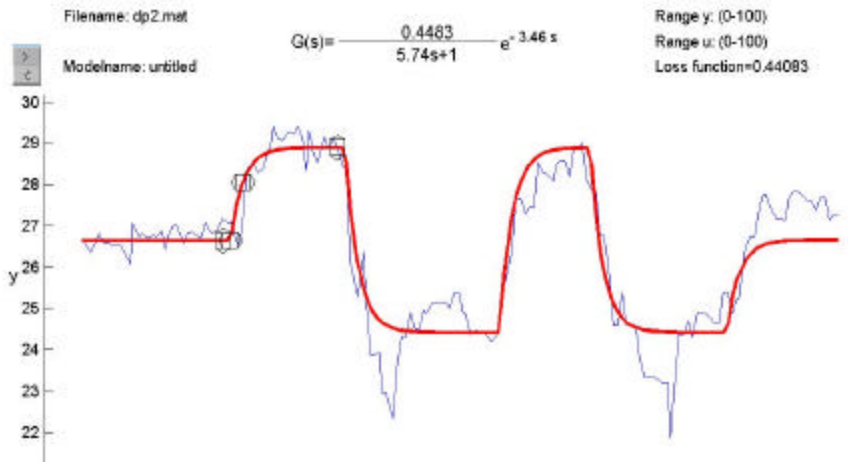

Figure 19: 1:st order pole for Lambda tuning.

Simulation time = 400 s.

Measure /controller	Controller parameters	IAE	E_{max}	$u_{overshoot}$	Var(u)	φ_m (°)	L_m (s)	L_{max} (s)	A_m	Ms
Lambda λ=8 τ =1	Kc=1.117, Ti=5.74	187.1	8.84	0.06	47.4	59	10.5	12.5	4.67	1.65
Lambda λ=16 τ =2	Kc=0.6578, Ti=5.74	264,5	9.96	0	55.0	83	25.7	27.8	7.93	1.35
Lambda λ=24 τ =3	Kc=0.4662, Ti=5.74	371.6	10.6	0	61.2	93	48.2	50.2	11.18	1.24

PI_opt Ms=1.6	Kc=1.476, Ti=6.194	165.5	8.24	0.137	44.8	50	7.5	9.5	3.61	1.82
PI_opt Ms=1.4	Kc=1.132, Ti=6.635	190.4	8.92	0	46.9	67	13.1	15.1	5.2	1.53
PI_opt Ms=1.2	Kc=0.6662, Ti=7.317	331.8	10.2	0	56.0	94	40.4	42.4	9.79	1.26
PID_opt Ms=1.6	Kc=2.42, Ti=6.61 Td=1.23	117.5	6.71	0.21	36.4	42	4.8	6.8	5.36	1.84
PID_opt Ms=1.4	Kc=1.88, Ti=7.18 Td=1.28	135.4	7.38	0.08	37.3	54	7.3	9.3	7.89	1.55
PID_opt Ms=1.2	Kc=1.12, Ti=8.05 Td=1.37	217.6	8.74	0	44.9	82	19.2	21.3	13.82	1.28

The calculated Ms-values for PI/PID_opt differ from the desired values since the model used in the design isn't the same as the one used in the simulation.
Lambda tuning is for the first time noticeably worse than PI_opt. This is probably due to the large modeling errors in the simple model used in Lambda tuning.

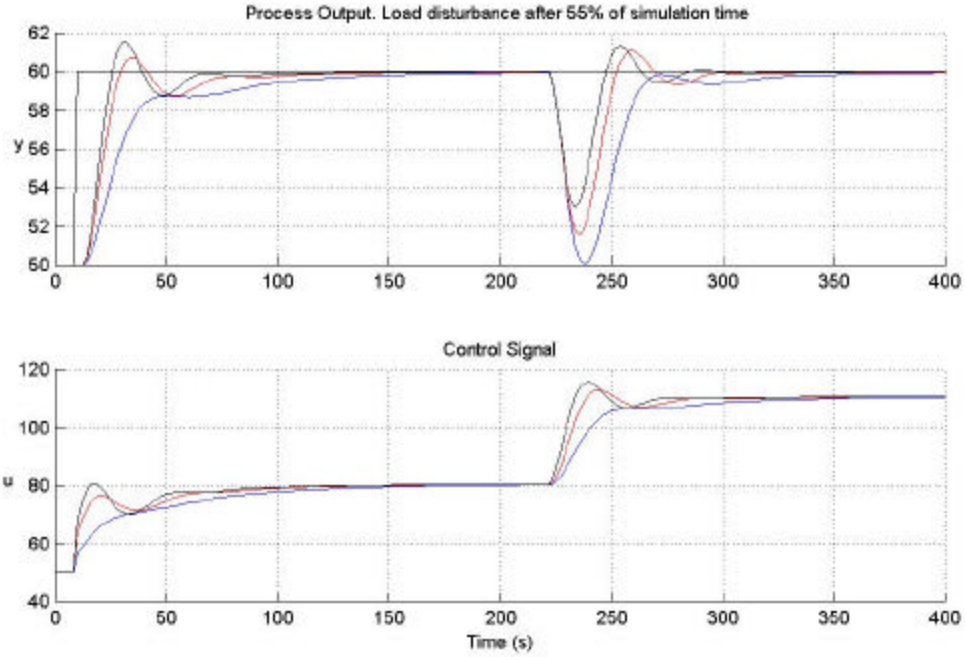

Figure 20: Simulations with PI/PID_opt with Ms=1.4 and Lambda with t=2. The slow curve belongs to the system with Lambda controller. PID_opt is the fastest as usual and PI_opt is in the middle.

The control signal reaches values above 100% and would, if it were a real control system, saturate. But since the simulation is made on a linear model it's the behaviour of the signal that is important, not the exact levels. A lower initial level would have given a control signal within measurement ranges.

4.4.6 Process four

Data from file: lev3. Level in a condensate vessel.
The process model used in the simulation is also used in all design methods:

$$G(s) = \frac{-0.00651}{s} \cdot e^{-2.00s}.$$

Simulation time = 300 s.

Measure /controller	Controller parameters	IAE	E_{max}	$u_{overshoot}$	Var(u)	φ_m (°)	L_m (s)	L_{max} (s)	A_m	Ms
Lambda λ=6	Kc=-33.63, Ti=14	4.17	0.27	0.29	4.71	47	3.6	5.6	3.33	1.61
Lambda λ=10	Kc=-23.49, Ti=22	9.37	0.35	0.21	5.21	56	6.0	8.0	5.01	1.36
Lambda λ=18	Kc=-14.6, Ti=38	25.6	0.53	0.16	6.66	64	11.4	13.4	8.06	1.20
PI_opt Ms=1.6	Kc=-28.36, Ti=10.1	4.06	0.29	0.38	5.27	41	3.5	5.5	3.93	1.60
PI_opt Ms=1.4	Kc=-21.7, Ti=13.5	6.93	0.35	0.31	5.69	47	5.3	7.3	5.16	1.40
PI_opt Ms=1.2	Kc=-12.73, Ti=23.5	20.2	0.55	0.26	7.21	55	10.2	12.2	9.24	1.20
PID_opt Ms=1.6	Kc=-46, Ti=4.9 Td=1.06	1.74	0.18	0.42	4.21	37	2.1	4.1	2.50	1.68
PID_opt Ms=1.4	Kc=-34.75, Ti=6.8 Td=1.03	2.80	0.22	0.36	4.42	43	3.1	5.1	3.31	1.44
PID_opt Ms=1.2	Kc=-20.23, Ti=12.7 Td=0.97	7.78	0.34	0.29	5.38	51	6.3	8.3	5.77	1.22

In this case all methods designed controllers that are totally unrealistic, the gain is much too high. This is due to an optimistic process model. Collected data showed almost no process time delay, the automatic parameter estimation gave L = 2.00 seconds which is short for a process of this type. The simulation shows that the control signal becomes very large, but the demands on Ms are still fulfilled.

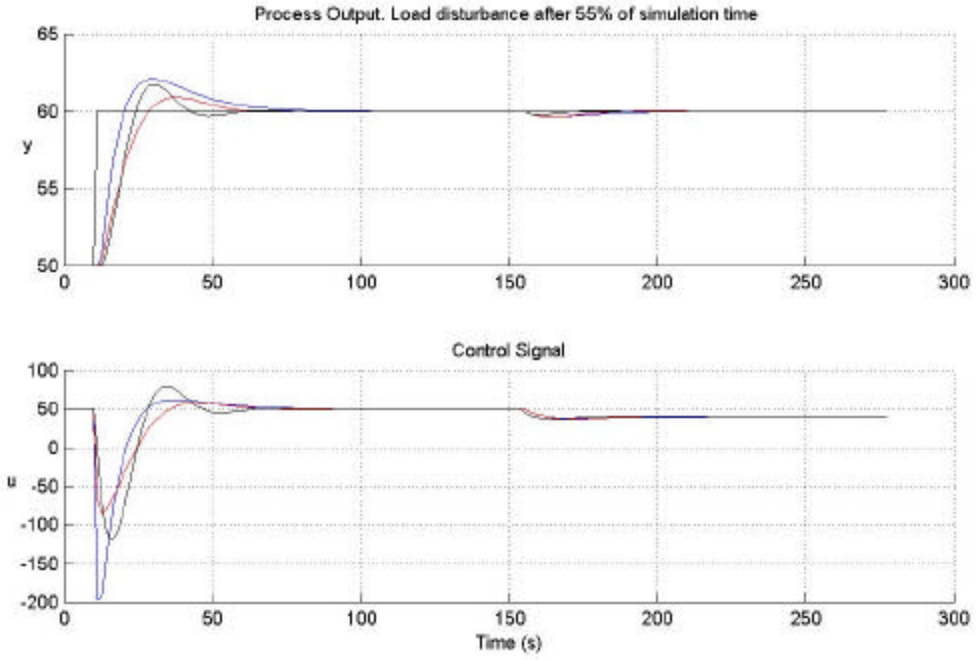

Figure 21: Simulations with PI/PID_opt with Ms=1.4 and Lambda with l=10.

A set point change at 10% gives very large control signals. The short time delay of the model and the high gain of the controller make the effects of the step load disturbance very small. λ was selected to give a closed loop system with Ms close to 1.6, 1.4 and 1.2 respectively. An experienced control engineer wouldn't choose such small λ-values for a process of this type. The rule of thumb (eq. 4 in chapter 3.3.2) with e_{max} = 25% gives λ = 38, which gives a better but still too large control signal. A more reasonable control signal is achieved with $\lambda \geq 100$ seconds.
The process time delay is in many cases longer than the estimated 2 seconds. A larger time delay gives a lower gain with any design method. Process 2, which is a similar process, had $L \approx 44$ seconds, then all design methods gave reasonable control signals.

4.4.7 Conclusions

The comparison shows that Lambda tuning is a reliable design method that is easy to understand and use in the absence of computer power. The design parameter λ has an intuitive meaning, although it's an odd meaning for integrating processes. PI/PID_opt are more scientific methods whose design parameter has a less practical meaning, the maximum value of the sensitivity function cannot be verified by simple measurements. Ms is an exact design parameter, which always gives the desired trade-off between performance and stability. Knowing that Ms=1.4 gives a stable design (if the model is correct), it is possible to design controllers without having even basic process knowledge. But as shown in the comparisons on process 4, ignorance will be punished.
The comparisons on process four taught that process knowledge is important, it is also important to make an estimation of the size of the control signal, for example in a simulation. Lambda tuning with λ according to the rule of thumb gave a smaller (although still unacceptable) control signal than PI/PID_opt with Ms = 1.4. A low Ms value does apparently not always give reasonable control signals on integrating processes.

The differences between Lambda tuning and PI_opt are quite small as long as the best possible model has one of the two structures supported by Lambda tuning, but when the process is better described by a more advanced model structure Lambda tuning shows its limitations. Since the modelling error then becomes larger than with the advanced model structure the controller has to be designed with larger stability margins, this always gives lower performance.

5 Implementational aspects

This chapter contains some interesting (?) parts of the implementation …

5.1 Graphical User Interface

The GUI is partly built in Matlab's GUI Development Environment GUIDE. GUIDE is very useful at the beginning of the design, but when the rough work is done it is faster and safer to type code by hand. See Matlab manual for further information about GUIDE.
The GUI contains not less than six windows: a main window and five sub-windows. The large number of sub-windows depends on the gradually increased number of features. When a new feature was added it was implemented in a new window.
All buttons, menus, text fields etc are represented as graphical objects with properties such as size, position, colour etc. The actions that are executed when for example a button is pressed is defined by the object's callback function. Every graphical object can have a callback function. They are all collected in a large file. Since all events are initiated by a callback this is where all the action is.

5.2 File structure

Current directory in Matlab's workspace has to contain all files below when TRA is started.

Main files

1. tra.m (initialises the tool)
2. cbtra.m (performs all actions)

Automatic estimation of different model structures

3. squareerrorg1.m
4. squareerrorg2.m
5. squareerrorg3.m
6. squareerrorg4.m
7. squareerrorg5.m

PI/PID_opt, design files

8. optpi.m
9. optpi.p
10. optpid.m
11. optpid.p
12. funpid.m
13. funpid.p

Simulation model

14. simsystem.mdl

GUIs (contains information needed to create a window)

15. guiconstants.mat (common matrices and vectors)
16. tragui.m (information about the main window)
17. modelgui.m (… model window)
18. controlgui.m (… control window)
19. sensgui.m (… sensitivity window)
20. simgui.m (… simulation window)
21. Evalgui.m (… evaluation window)

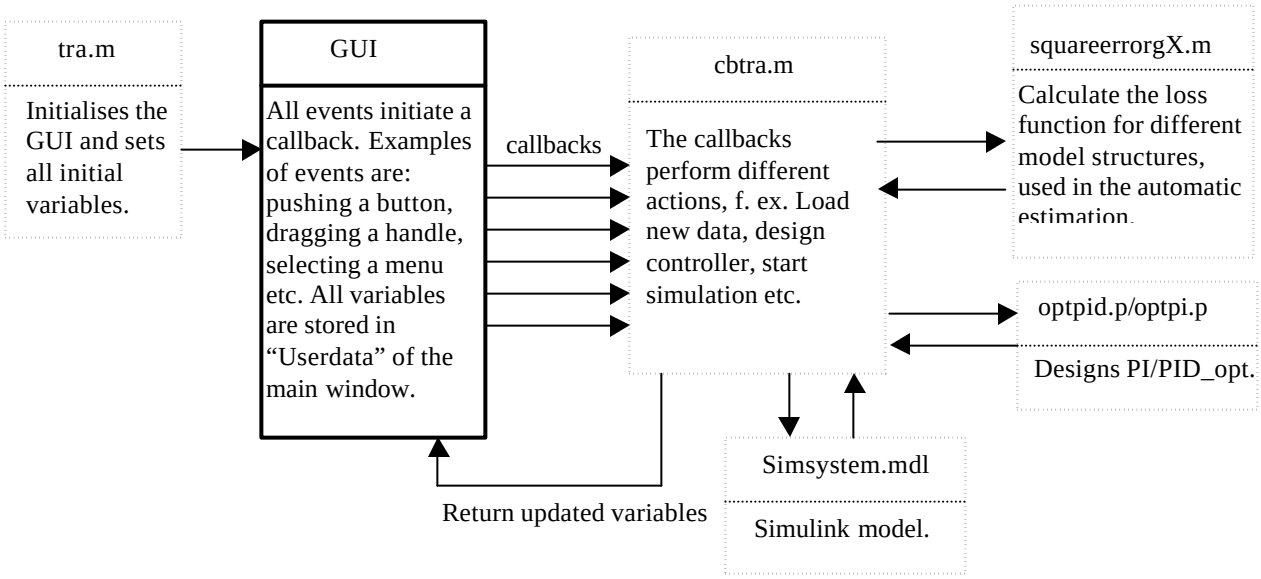

Figure 22: File structure.

The tool is initiated and started with tra.m. Most functions and calculations are implemented in the large callback file cbtra.m. A few constant matrices that are common for all GUI-windows are stored in the file guiconstants.mat. Some Matlab functions prefer an external file as input argument. The estimation function fmincon is one of those. The files squareerrorgX.m calculates the loss function for different model structures and are used as input argument to fmincon. Files 9-14 are used in the PI_opt and PID_opt designs. The file simsystem.mdl contains the simulation model. Files 16-21 are the GUI-window definitions. They can be included in either tra.m or cbtra.m when all changes in the GUI are done (tragui.m in tra.m and the rest in cbtra.m).

5.3 Data structure

One particular demand on the tool was that it had to be able to run at the same time as another Matlab application (for example another TRA) without interference. This demand made it impossible to store the variables as global variables in workspace. All TRA-variables are for that reason collected in a large struct. The struct is defined in tra.m where the variables are given their initial values. The graphical objects in Matlab have a property called "UserData". The object doesn't use UserData; it is free to fill with any variable. The struct is stored in UserData of the main window. UserData in the sub-windows contains the handle of (a pointer to) the main window. The struct is copied from UserData every time a callback is performed. At the end of the callback the new values of the struct is stored in UserData again.

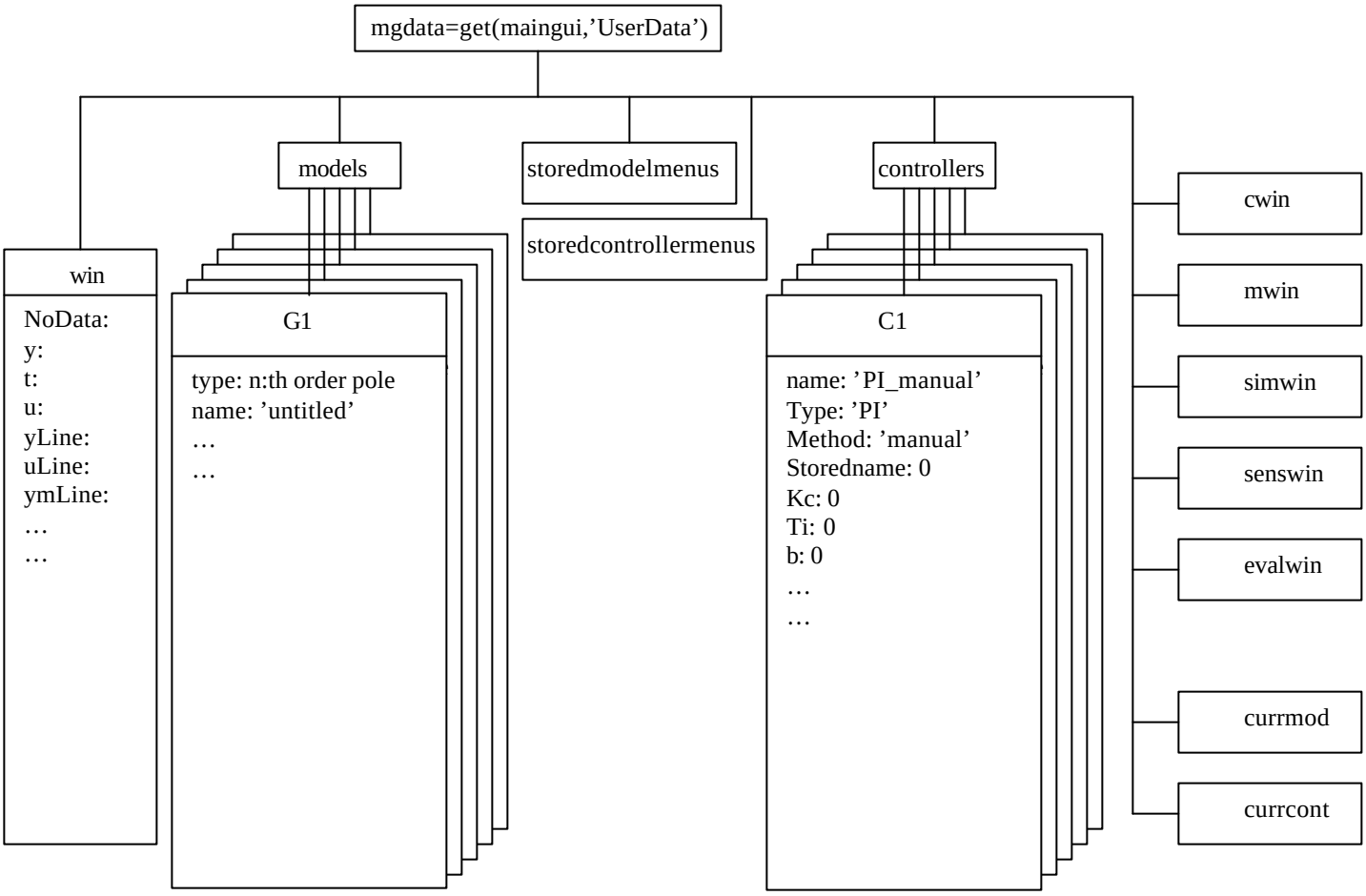

Figure 23: Data structure.

Win contain all variables concerning main window and data, models contain the five different model structures and controllers contain the five different controller types. Cwin, mwin etc contain the handle numbers to the sub windows. Currmod and currcont contains complete information about current model and current controller respectively. Storedmodelmenus and storedcontrollermenus contain handle numbers to the menu objects in Model menu and Control menu.

The data structure can be viewed in detail in tra.m where all variables get their initial values.

5.4 Model estimation

5.4.1 Handles

The handles are used to manipulate the model graphically. To enable complete manipulation of the model parameters we need as many handles as model parameters. How the handle movements should influence the model parameters is a matter of taste. The nicest way is probably to put all handles on the curve and let them affect the shape of the curve, like dragging a rubber band. The most straightforward way is to have a one to one relation; meaning only one parameter is affected when a handle is moved.
Most parameters have an intuitive placement of the handles, but some parameters in the more complex model structures don't have any intuitive placement on the model curve. The handle for Tz (the time constant of the zero) is one of those. The Tz-handle is for that reason combined with the handle for T (the time constant of the pole) and changes its value as it is moved along the time axis.

5.4.2 T-handle for model structures with n>1

It is necessary to make iterations to find the perfect position for the T-handle for model structures with n>1. Perfect position is on the model curve at 63% of the final level. It is solved with Newton Raphson iterations.

5.5 Simulation

The simulation is done in Simulink. The Simulink model is shown in the figure below.

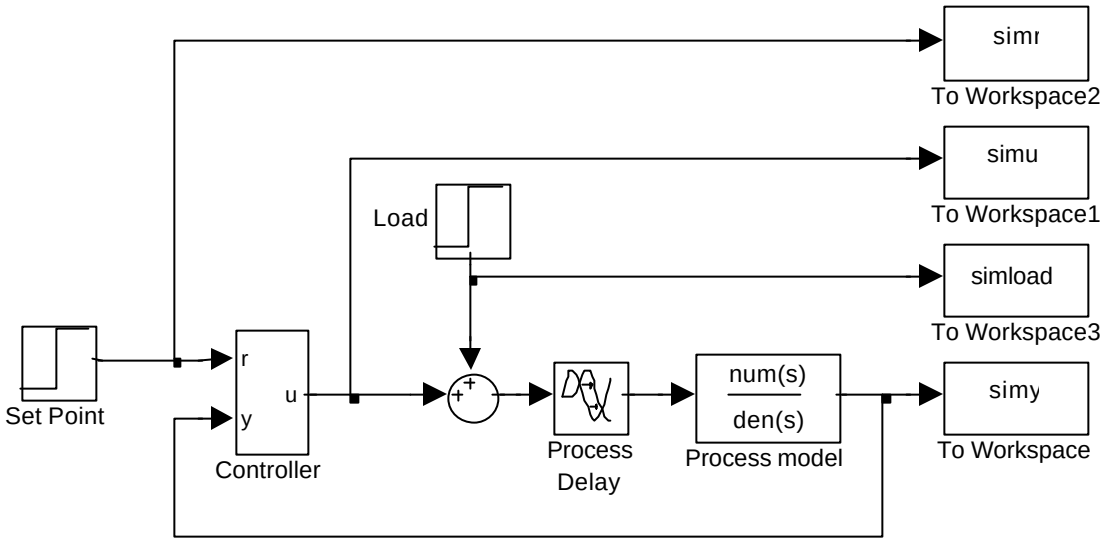

Figure 24: Simulation model in Simulink.

Set point makes a step after a few seconds (default 10 seconds), after 55% of the simulation time Load makes a step that represents a step load disturbance acting on the process. The controller is implemented with standard Simulink blocks as shown below.

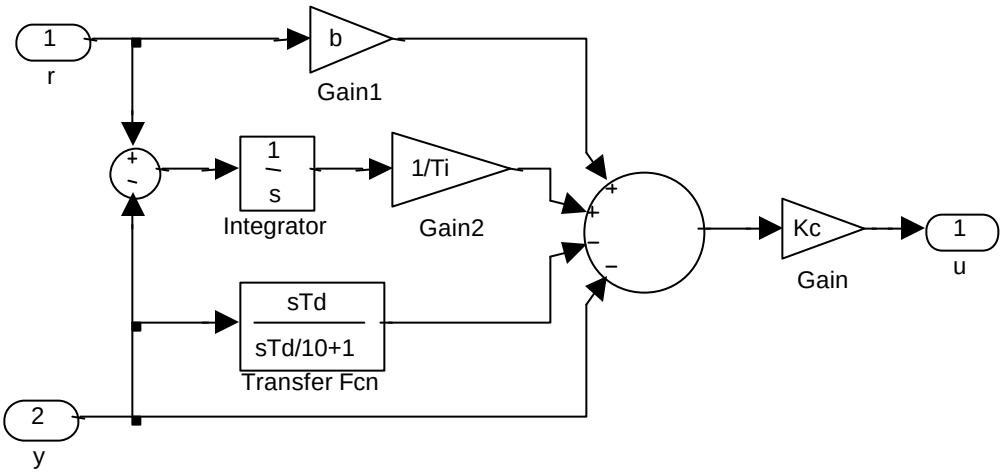

Figure 25: Controller implemented in Simulink.

Simulink does not automatically select a suitable simulation time. A time vector has to be given as input when the simulation function is called. The tool calculates the time vector with respect to current model

structure (integrating processes are treated specially), model parameters and current controller parameters. It is also possible to set the simulation time manually in simulation properties.

To make the simulation result a bit more realistic the initial level of the set point and process output is set to 50% of y-range and the initial control signal is set to 50% of u-range. The amplitude of the set point step is 10% of y-range and the amplitude of the step load disturbance is default 10% of u-range.

5.6 Special functions

5.6.1 FindSteps

This function is used to find steps of various amplitudes in the control signal. It determines where and how large the steps are. The steps that are found are used as input to the model to generate the model output. Finding steps in a nice data vector is not very difficult, but to be able to use collected data although it is not perfect the function has to perform a few checks. This can be done very sophisticated, but the present version of the function is quite simple.
It is called with [steppos, stepsizes]=FindSteps (u, stepsens), where u is the control signal and stepsens is the minimum amplitude, in percent of max (u) – min (u), of a step. The function returns two vectors. Steppos contains the element positions and stepsizes the amplitudes of the steps in vector u. The function searches for a difference between two following samples that is larger than stepsens. When a pair of samples are found where |u(k+1)-u(k)| > stepsens*(umax-umin) the position k is stored in steppos. Next step is to determine the amplitude of the step. The final level of the step is found by searching for a pair of following samples with a very small difference. When a couple of samples are found where |u(c+1)-u(c)| < 0.1*stepsens*(umax-umin) stepsize is assumed to be $u(c)-u_{level}$. U_{level} is the final level of the previous step or the initial level of u if this is the first step. The user can set stepsens in Data Properties. Default value is 5%. This means that differences between two following samples less than 5% of measurement range will not be considered as a step.

6 User's guide

6.1 System demands

This version of TRA is written for Matlab 5.3 with Simulink, Control Systems Toolbox and Optimization Toolbox. It is developed in Matlab 5.3.1 on Unix, Windows –98 and –NT with following toolbox versions:
Simulink (version 3.0, 10-september-1999)
Control Systems Toolbox (version 4.2, 10-september-1999)
Optimization Toolbox (version 2.0, 09-october-1998)

6.2 Input data

6.2.1 Tests on real process

Some tests have to be done on the real process.

- Disconnect the controller and adjust the control signal (the process input) manually.
- Make a few steps on the control signal (u), preferably in both positive and negative directions.
- Log process output (y) and the manually adjusted control signal (u). It is enough to save the sampling time if it is constant. Otherwise all sampling instants have to be logged (t) as well.
- Transform sampled data into Matlab vector format.

The tool needs one vector containing the control signal (u), one vector containing the process output (y) and one vector containing the sampling instants (t) alternately a scalar defining the sample time. All vectors have to be of the same length. Vectors can be either 1*n or n*1, all input vectors are transformed to the same format internally. The control signal must contain at least one step otherwise data is useless.

6.2.2 Measurement Ranges

The tool also demands information about measurement ranges ymin, ymax, umin and umax. Default values are min=0 and max=100.
In many control systems the controller internally scales y to values in the range 0-100. This means that when the process reaches its maximum value, whatever that may be in natural units, the controller receives a scaled signal at 100. Internally it also gives a control signal (u) in the range 0-100. If y and u are measured outside the controller's internal structure the signals will probably not be within the range 0-100.

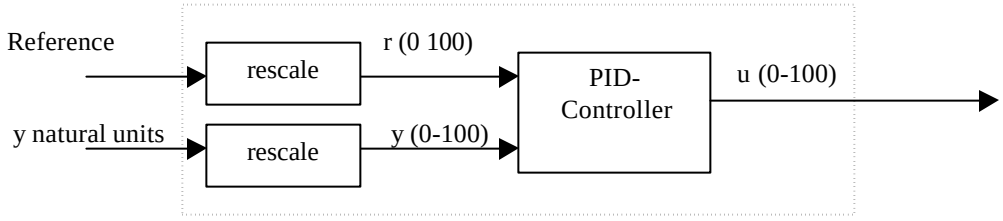

Figure 26: Explanation of measurement ranges.

If both signals were measured in the internal scaling we wouldn't have to worry, but some equipment measures the process output (y) in its natural units (volts, amps, etc) and the control signal (u) in the controllers internal scaling (0-100). For those cases the measurement ranges are ymin = $y_{\text{natural min}}$, ymax = $y_{\text{natural max}}$, umin= 0 and umax= 100. This is a very common case, so the variables umin and umax are often 0 and 100. But to be prepared for the unexpected it is possible to give umin and umax any values.

If your control system doesn't re-scale the signals as shown above or if you don't have a clue what this subchapter deals about just keep the default values.

6.2.3 Input data stored in a file

The vectors and scalars are preferably stored in a *.mat file. When loading data from a file the stored variables must be named u, y and t. It is also a good idea to include the measurement ranges in the same file, otherwise you will be asked to enter the ranges immediately after loading the file. Ranges must be scalars named umin, umax, ymin and ymax. To save the variables in a *.mat file type:

save filename y t u ymin ymax umin umax

Type “help save” (without “ ”) in Matlab to learn more about saving.

6.3 Starting the tool

- Start Matlab. Make sure that all “tra-files” are in your current directory.
- There are several ways of starting the tool.

  1. >>tra

     The tool opens as an empty shell without data. Input data can then be loaded either from a file or from Matlab workspace. This is the easiest way if your data is stored in a file. Nothing else can be done before data is loaded.

  2. >>tra(y,t,u,[ymin ymax],[umin umax])

     This is the fastest way if all variables are in workspace.

  3. >>tra(y,t,u)

     A dialog box will be displayed immediately after start-up prompting you to enter the measurement ranges. If you have data in workspace but are not sure about the measurement ranges this gives you a chance to view data before setting ranges.

- Now data (blue line) and the output of a roughly estimated model (red thick line) are showed in the axes and the tool is ready to be used. Short information messages are displayed in the white square at the bottom of the main window.

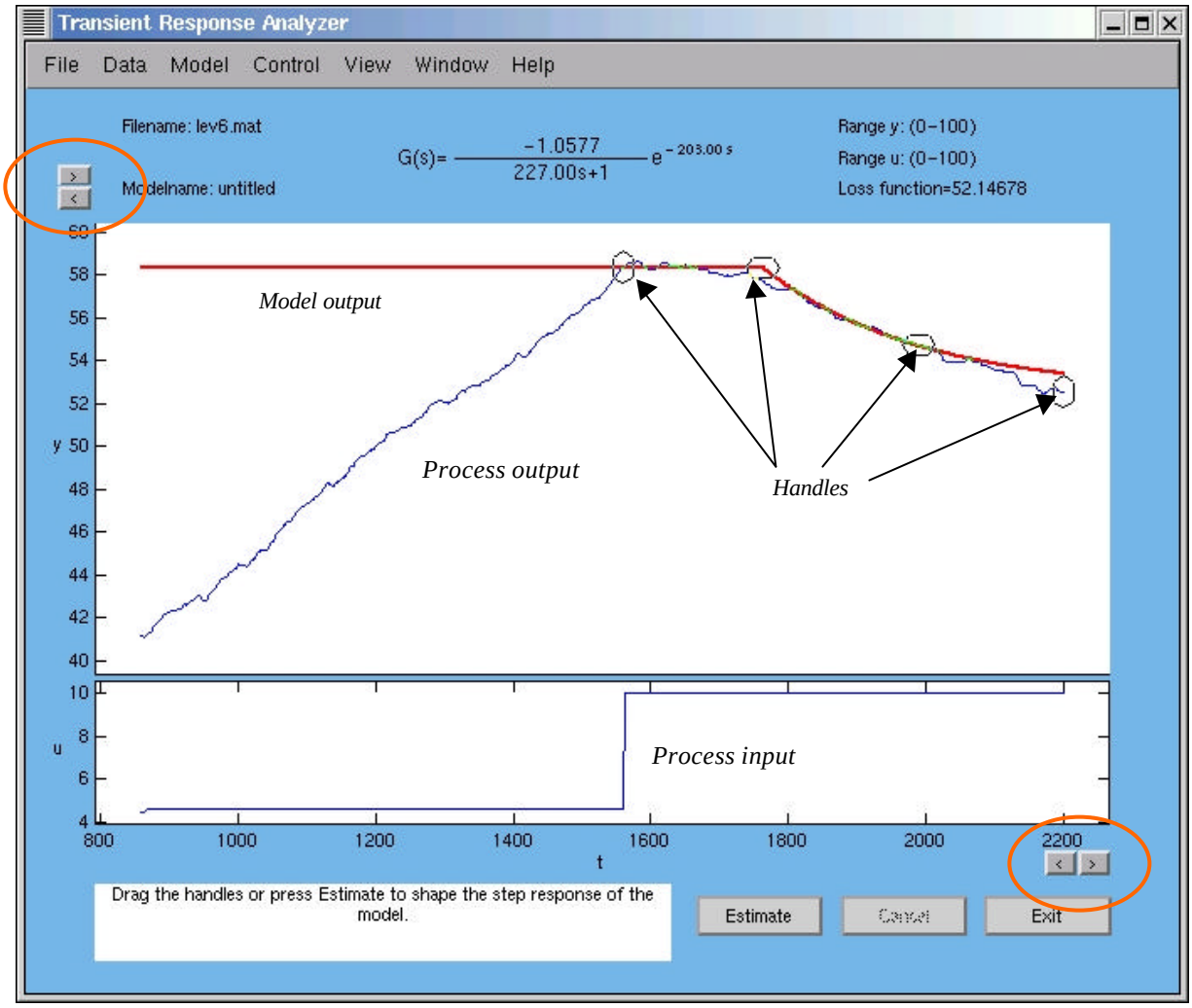

Figure 27: The main window. Default model structure is first order pole. In this case that is obviously not a good approximation of the real process. The transfer function of the model is displayed in the upper part of the window. The loss function is displayed in the upper right corner. The encircled arrow-buttons can be used to zoom in/out in either x- or y-direction. Zoom area is found in View menu.

6.4 Inspecting data

Make sure that input data is useful before too much work is done on modelling. If noise or disturbances are too large it is difficult to make a true model of the process.

6.4.1 Select region

If only a part of input data is useful choose ”Data”->”Select Region” and select area by click and drag. Then only data within the selected region is considered and displayed. When ”Data”->”Restore Original Data” is selected all original input data is considered again.

6.4.2 View

It is possible to zoom the axis in the main window. It is done with either the small arrow-buttons at the end of the y- and x-axes or from the “View” menu. Besides the normal zoom functions the menu also contains “Fit Axes”. It adjusts the axes limits so that all handles and curves are visible within the axis.
Note the difference between zoom and select region. Select region discards data outside the selected region. Zoom merely magnifies or shrinks the image.

6.4.3 Data properties

By selecting “Data” -> “Data Properties” it is possible to set new values on measurement ranges and step trig level. “Step trig level” decides how much the control signal (u) must change between to samples to be considered a step. Default value is 5% of max (u)-min (u). That is 5% of the span between the largest and the smallest actual value in the vector u. With a very small trig level the tool considers any small change in the control signal as a step. Then the computations will be much more time demanding and hence all handle movements will become slow. If the control signal also contains measurement noise the advantage of a well-balanced trig level is obvious.

6.5 Model estimation

The sampled output from the real process (y) is compared with the calculated output of a model given the same control signal (u) as the real process. The model parameters are adjusted until the calculated output agrees well with the sampled output of the real process (y). To see exactly how well the models response agrees with y the loss function is displayed in the upper right corner of the main window. A small loss function means that the process model agrees well with the real process.

6.5.1 Select Model

The user selects one among five major model structures. This is done in the “Select Model” window. The window is opened from “Model” menu. In this window it is possible to select model structure, type specific parameter values, mark/unmark parameters, make an automatic estimation of model parameters and reset the model to its initial parameters. Only marked model parameter are affected by the automatic estimation

6.5.2 Manipulation of model parameters

To manually change the model parameters it is, besides typing new values in Select Model window, also possible to drag the handles in the main window. Each handle corresponds to one parameter. The zero level is an important parameter that is not needed in the model transfer function, but it has to be set to make the model understandable. Integrating model structures also have an initial slope that has to be considered. The initial slope can be manually manipulated by dragging the broad black handle. Models with zeros have a combined handle for T and Tz, right click on the handle to switch between affecting T or Tz.

6.5.3 Automatic Estimation of model parameters

All marked parameters are optimised to minimise the loss function when the button “Estimate” is pressed. The unmarked parameters are unaffected.
If the optimization fails and the parameters are totally wrong (It could happen!) press “Reset” in Select Model window or in Model menu. The automatic estimation probably failed because the initial parameters were too far from the optimal ones, causing the optimization function to find a false minimum. Improve the initial parameter values and press “Estimate” again.

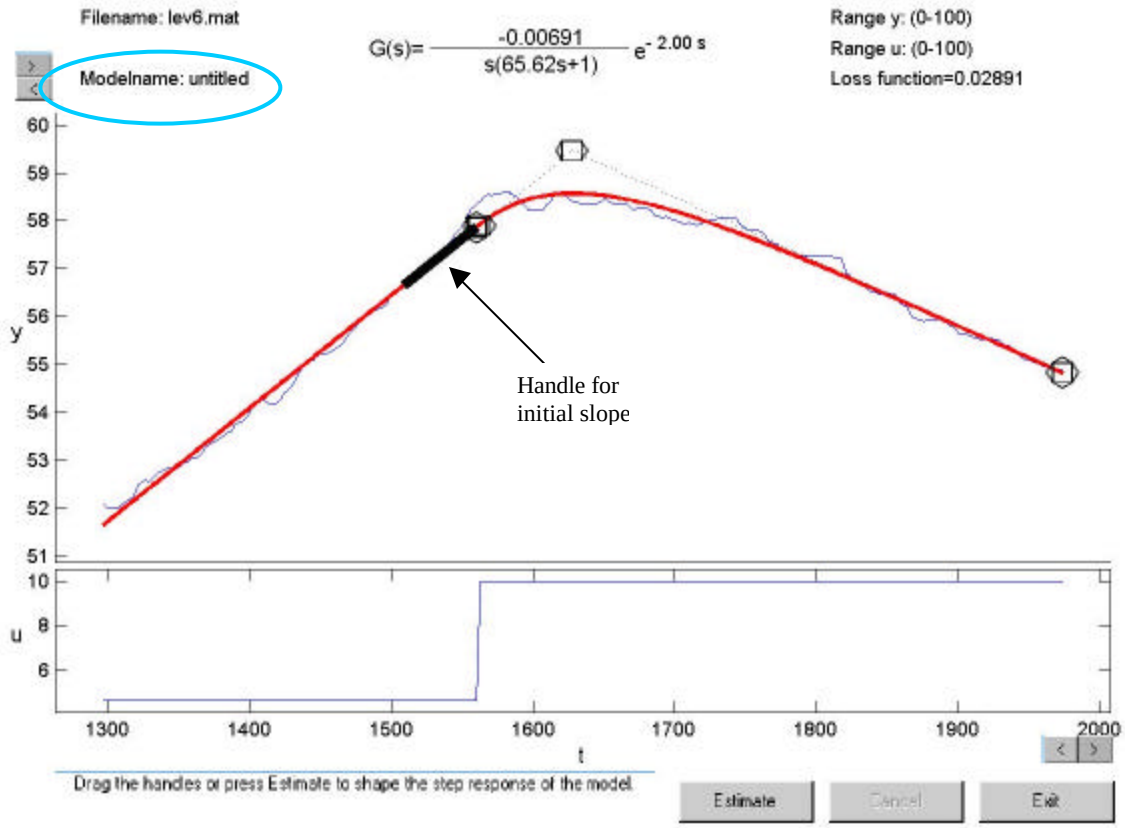

Figure 28: A closer region has now been selected. An integrating model structure with a pole (Integration + p) has been chosen and the parameters have been automatically estimated. The loss function is now considerably smaller. The model name is displayed in the upper left corner, this model have not been saved yet and has for that reason no name.

6.5.4 Saving a model

There are three ways of saving a model for later use.

1. Store Model (as…), stores the current model temporarily in TRA. The model is named by the user and is stored in Model menu. There is no limit of how many models that can be stored this way. The stored models are numbered in due order and the active model is marked in the menu. All models stored this way will disappear when TRA is terminated.
   If a stored model is useless it might as well be deleted: Model → Delete Model…→ ”model name”.
2. Export Current Model …→ … to mat-file, saves the current model in a separate *.mat file on disk, only one model can be saved in each file. Default filename is the same as the model name. Choose Model → Import Model … to load a previously exported model.
3. Export Current Model …→ …to txt-file, saves information about current model in a txt-file.

6.6 Design a controller

All control design is performed in the “Design Controller” window. Five different design methods can be selected. Two of them are manual (one for PI and one for PID) and lets the user decide all controller parameters. The other three methods are automatic; the user may set a design parameter. A default value of

the design parameter, giving a “normal” design, is displayed in the editable text field when a design method is selected for the first time. The lambda method is an old PI design method that is only specified for two model structures, the pure integrating structure and the 1:st order pole. The design parameter lambda has different meanings for different model structures. For 1:st order pole it is the desired closed loop systems time constant and for integrating models it is the desired time before the error starts to decrease after the entrance of a step load disturbance. The remaining methods, PI_opt and PID_opt are recently developed at the department of automatic control at Lund Institute of Technology. They can be used on all model structures with monotonously decreasing phase. The design parameter Ms can be given any value between 1.2-2.0 where a low value (1.2-1.4) gives a slower but more stable system than a large value (1.8-2.0). Default value is Ms = 1.4.

6.6.1 Saving a controller

This is done in the exact same ways as saving models but in the Control menu.

6.7 Evaluate the controller performance

6.7.1 Simulation

A simulation is performed when ”Simulate” is pressed in ”Design Controller” window. During the simulation the closed loop system is exposed to a set point change and a step load disturbance. The simulation is displayed in a window called ”Simulation”.

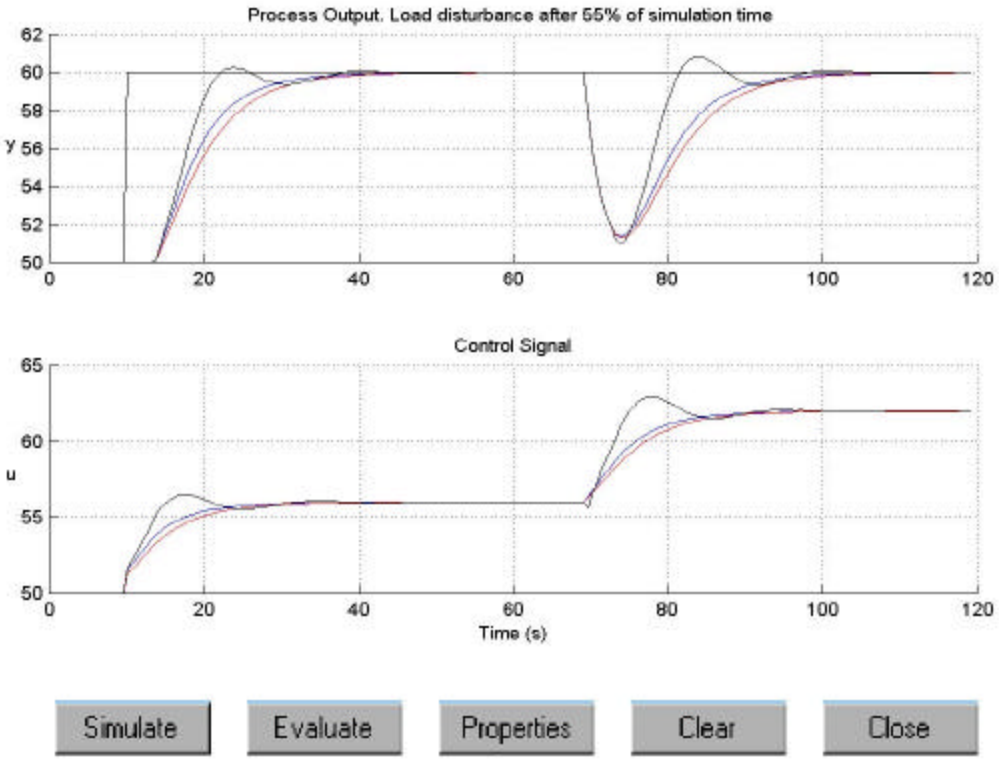

Figure 29: The simulation window.

The simulation time is calculated by a rough rule of thumb. If it is too long or too short it is possible to adjust it by pressing “Properties”. In the appearing dialog box it is also possible to set the size and occurrence time of the set point step and the size of the step load disturbance. The step load disturbance will always enter after 55% of the simulation time. If the dialog box “Simulation Properties” is closed by

pressing “OK” the changes will attend the next time “Simulate” is pressed. A good advice is to clear the simulation window before doing the new simulations.

6.7.2 Sensitivity functions

The sensitivity functions are calculated and displayed when ”Sens func” in the ”Design Controller” window is pressed. The lower plot shows the sensitivity function from n to y. The design parameter Ms in PI/D_opt is the maximum value of this sensitivity function.

6.7.3 Evaluate

To get a more scientific evaluation of the simulation and the sensitivity plots it might be a good idea to press the ”Evaluate” button in the Simulation window. The Evaluate window is then opened and some enlightening facts about the simulation and the closed loop systems stability are displayed. All measurements from the simulation begin after the entrance of the load disturbance. The diagram shows the Bode plot of the loop transfer function.

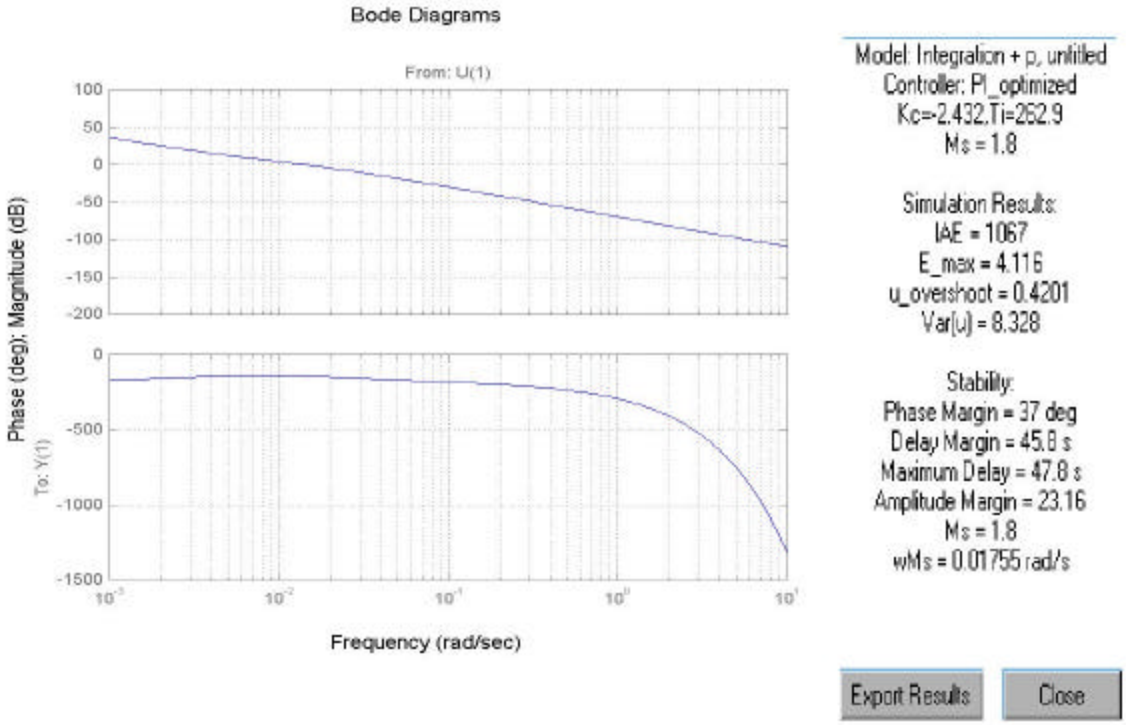

Figure 30: The Evaluation window.

It is possible to zoom the Bode Diagrams by right clicking on it. Export Results writes the evaluation results to a *.txt file. The file can then be read in an editor like Ms Word or Emacs.

6.8 Exit the tool

All sub-windows are closed and TRA is terminated when Exit in the main window is pressed. If you wish to end your Matlab session select “Quit Matlab” under File menu in the main window.

7 References

Wallén, A. (2000): “Tools for Autonomous Process Control”. PhD thesis
ISRN LUTFD2/TFRT--1058--SE, Department of automatic Control, Lund Institute of Technology, Lund, Sweden.

Panagopoulos, H. (2000): “PID Control; Design, Extension, Application”. PhD thesis ISRN LUTFD2/TFRT--1059--SE, Department of automatic Control, Lund Institute of Technology, Lund, Sweden.