Report of the Project for Bachelor

of Engineering in Electronics

Identification and Optimization PID parameters using MATLAB

Cork Institute of Technology

Gdynia Maritime University

Student: **Daniel Czarkowski**, DLX4

Supervisor: **Dr. Martin Hill**

Assessor: **Dr. Tom O'Mahony**

Cork 2002

Acknowledgements

I would like to express my gratitude to my assessor Dr. Tom O’Mahony for his support in Matlab and indeed control. Thanks, Tom.

I would like to also thank to Head of Department Paul Sliney and my supervisor in Poland, prof. Roman Śmierzchalski. They allow me to study abroad.

Contents

List of figures and tables

Abstract

The objective of this project is design a Graphical User Interface (GUI) to aid PID controller tuning. The first step in the tuning process is to identify a model for the process. This is achieved by applying a step input to the real process (a tank system) and logging the data to MATLAB via a Data Acquisition Unit. Appropriate filtering of the data is performed to remove high-frequency noise. This data is used to identify an appropriate model for the process. The PID parameters are then obtained by minimising a standard integral of error criteria where the minimisation is performed using Genetic Algorithms (GA). The results are to be compared with those obtained from tuning technique available in Nonlinear Control Design Blockset. The GUI is evaluated in real-time a single laboratory apparatus.

1 Graphic User Interface

One of the aims of the project was to implement Graphic User Interface (GUI). GUI makes that we do not have to waste a lot of time for understanding variables that someone has implemented. GUI causes easier using of Matlab. In my project, we can in easy way identify an object. (without giving details, for example: theta format) Unfortunately, universal GUI has disadvantages. Knowledge about controls is necessary.

In my project, I tried to write program, which could identify almost every object. This is possible only in limited range. For example, my Matlab code cannot be used for identify motors, however it works very well with plant such as tank.

The GUI consists of four main parts:

1. Identification
2. Tuning PID parameters
3. Display result
4. Axes

On the right, as shows in figure bellow, we have first two main parts. First, we have to identify an object, so I put it on the top of the screen. This part of the screen includes option *load data*. In this pop menu, we can choose two options: *load datafile* or *Real time: tank.* In the first option, we load default data that are saved in the file `datafile.mat` Second option *Real Time: tank* identify plant in real time. It works only on workstation with Data Acquisition Unit and *Real Time Toolbox ver. 3.0* delivered by HUMUSOFT[1]

The next part of the screen includes options tuning parameters. I have implemented two methods: Genetic Algorithm and state-of-the-art optimisation. In fact, the second method (from Optimization Toolbox) uses extra block from Nonlinear Control Design Blockset[2].

When we set parameters like type of controllers and type of criterions, we can choose optimisation methods such as Genetic Algorithm and NCD. After compute we obtain PI(D) parameters above graph.

There are some options, which make using of GUI easier. We can push button “hold on” and compare e.g. identification methods or tuning methods. Simple buttons, but useful.

1 http://www.humusoft.com/

2 more about this tuning method in chapter 4.3

After optimisation using NCD, we should update PID parameters on the main screen using “Refresh” button. Button “Grid” turns on/off grid on the graph.

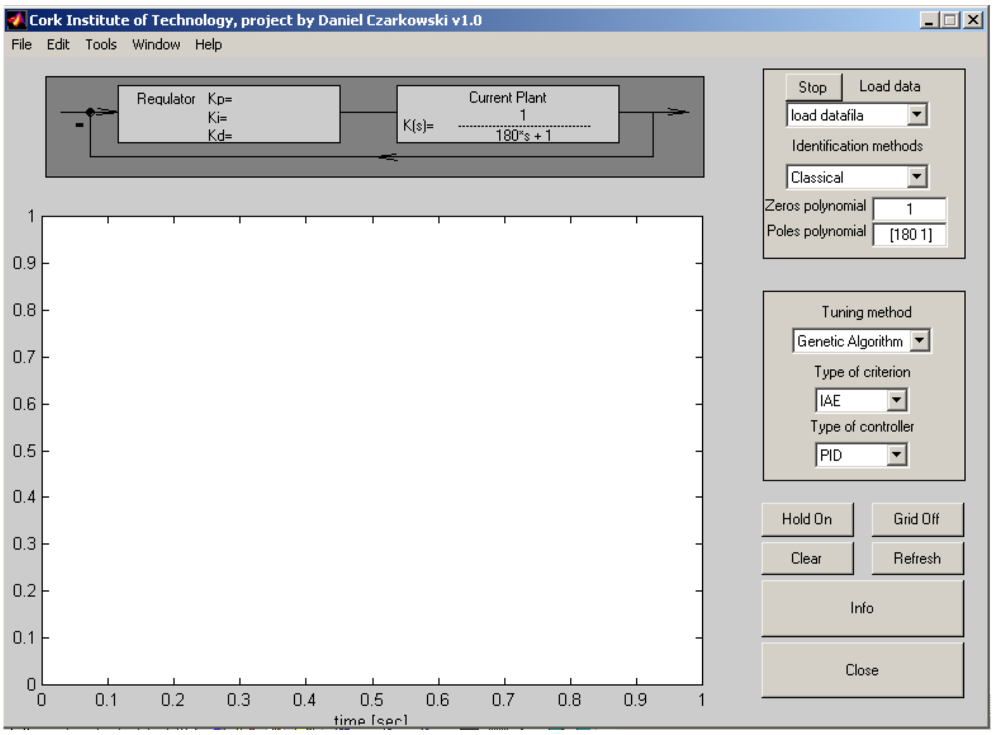


figure 1.1 Graphic User Interface

2 Identification

System identification is a problem that arises in many disciplines where a mathematical model is researched for a physical system. System identification is a basic problem in control theory, because in almost any application of control, the model is not completely specified.

In this chapter, I focus how to obtain mathematical model a tank.

2.1 Plant

In my project, I used a single tank. The **CE105 Coupled Tanks Apparatus**[3], shown in figure below, was used as my plant. In first part of my project I focused how to identify this object.

3 more information about Tecquipment Limited http://www.tq.com/index.html

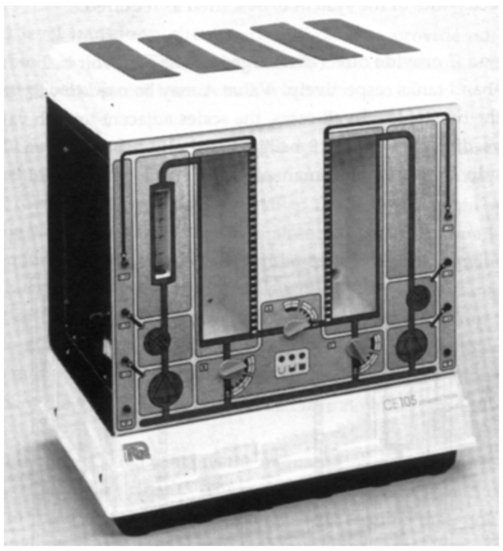


figure 2.1 Tank

The diagram of flow liquid is shown below. Pump is used to fill a tank with liquid at a constant rate. An outlet, controlled by a valve at a set point, allows water to leave the tank such that in period of time the liquid level will achieve a constant level. When this condition is achieved the system is said to be in **Equilibrium**.

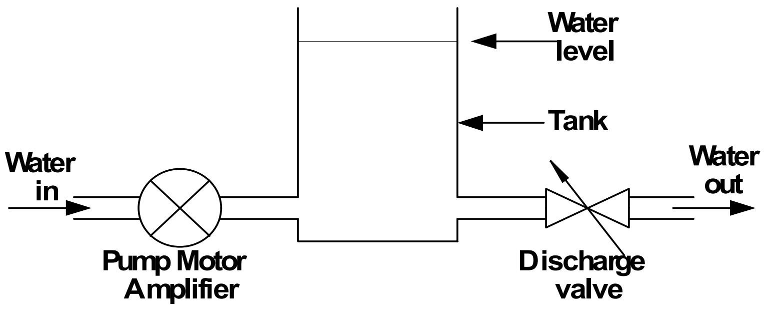


figure 2.2 Model of the tank

What happened if someone changed capacity of valve? My program is quite adapted for this situation. I assume that pump stop working when the level of the tank is the same during at least 120 sec. In other words when s(n)=b and s(n+120)=b then pump stop working. The details of the code are in file `calc_rt.m` which is attached in appendix.

2.2 Obtain data from a tank (filtering)

The goal of the identification is research mathematical model of a real plant. It is achieved by a step response. As input we have pump which works with the same efficiency all the time. It simulates a step function. In output, we have level of liquid in tank.

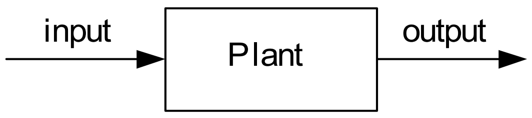


figure 2.3 Step response

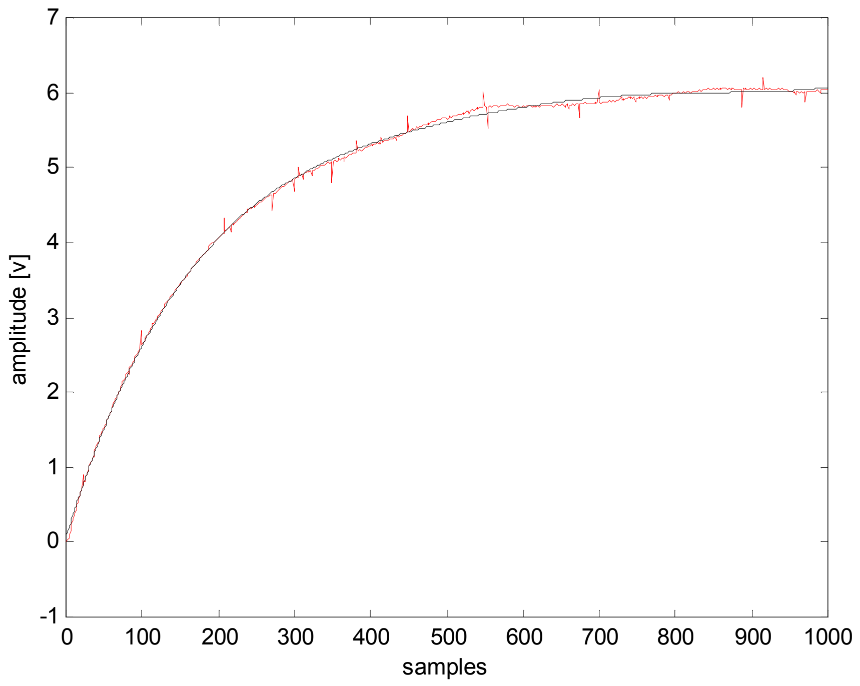


figure 2.4 Measured signal from tank (red), after approximation (black)

On the figure above are two characteristics. The first signal (red) was desired from a real plant that was a single tank.[4] The data were logging to MATLAB via a Data Acquisition Unit - AD512 produced by Humosoft. As we can notice, data are noised. In order to remove noise from real date I have used algorithm Polynomial Curve Fitting[5] the code looks like this:

```
x=[1:SamplingPeriod:N]
p = polyfit(x,Vout,5);
```

4 Described in chapter 2.1

5 more about this method on the Internet

http://www.mathworks.com/access/helpdesk/help/techdoc/ref/polyfit.shtml

```
ypoly = polyval(p,x);
```

In this case I have used fifth order polynomial which is quite good approximation as we can notice on figure above.

Relative error is counted as

$$\delta = \frac{\left|\mathrm{Re} - aprox\right|}{\mathrm{Re}} = 3.0422\%$$

2.3 Identification using classical method (The Reaction Curve Method)[6]

Once the model structure is defined, the next step is to choose the correct value for the parameters. In order to identify these parameters, a suitable stimulus must be applied to the process input. In the Reaction Curve Method a step perturbation, that is an input with wide frequency content, is applied to the process and the output is recorded and the output is recorded to fit the model the data[7]

The step response of the process is shown below.

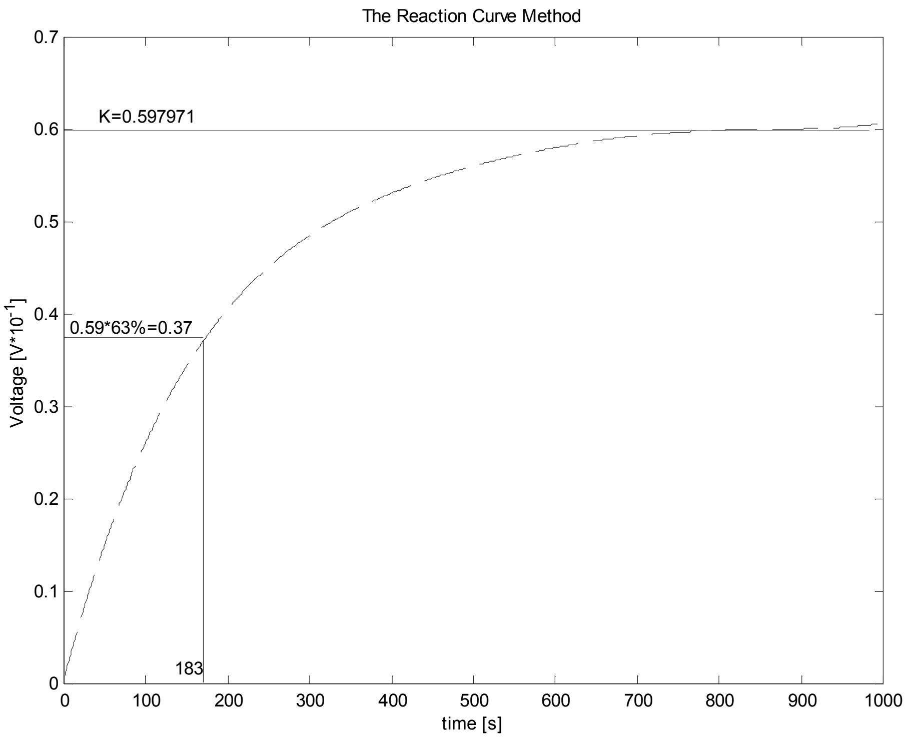


figure 2.5 The reaction curve method

Result of identification. Transfer function $K(s) = \dfrac{0.597971}{183s+1}$

`mineral` $K(s) = \dfrac{0.003268}{\mathrm{s}+0.005464}$

[6] computation in file ident_calc.m

[7] "Model Predictive control in the process industry", E.F. Camacho and C. Bordons

Relative error is counted as

$$\delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 1.9377[\%]$$

The Reaction Curve Method is probably one of the most popular methods used in industry for identify objects.

2.4 Computes the prediction error estimate of a Box-Jenkins model[8]

A general input-output linear model for a single-output system with input u and output y can be written:

$A(q)y(t) = \sum_{i=1}^{nu}[\frac{B_i(q)}{F_i(q)}]u_i(t-nk_1) + [\frac{C(q)}{D(q)}]e(t)$ here u_i denotes input #i, and A, B_i,C,D and F_i are polynomials in the shift operator (z or q)

For Box Jenkins $y(t) = [\frac{B(q)}{F_i(q)}]u(t-nk) + [\frac{C(q)}{D(q)}]e(t)$

Transfer function of measured plant $K(s) = \frac{0.003298\,\mathrm{s} + 0.003307}{\mathrm{s} + 0.005471}$

Relative error is counted as

$$\delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 0.9502[\%]$$

If we compare this method with others identification methods, the relative error is the lowest.

2.5 Computes least-squares estimation of ARX-models[9]

For ARX model $A(q)y(t) = B(q)u(t-nk) + e(t)$

Transfer function of measured plant $K(s) = \frac{0.003327}{\mathrm{s} + 0.005505}$

[8] counting in file ident_calc.m

[9] counting in file ident_calc.m

Relative error is computed as

$\delta = \frac{|\mathrm{Re} - estim|}{\mathrm{Re}} = 1.2829[\%]$, This method is worse than Box-Jenkins, but better than classical method

Using GUI we have opportunity to compare these methods with data obtained from tank. In order to do it, turn on function HOLD ON.

3 PID regulators

Despite the development of more advanced control strategies, the majority of industrial control systems still use PID controllers because they are standard industrial components, and their principle is well understood by engineers.

The **PID** controller: **P**roportional

Interval

Derivative

Parameter effects:

Close Loop Response	Rise Time	Overshoot	Settling Time	SS-Error
Proportional	Decreases	Increases	No Change	Decreases
Interval	Decreases	Increases	Increases	Eliminate
Derivative	No Change	Decreases	Decreases	No Change

table 3.1 Effects of PID regulator

On figure below, we see how the typical close loop system looks.

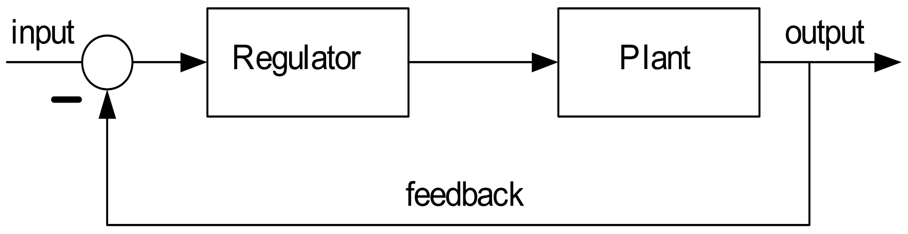


figure 3.1 Typical close loop system

"The closed-loop system behaviour is considered to be good, in the sense of having good gain and phase margins, if simultaneously crossover frequencies are as high as possible and

the gain margin is not less than 6 to 8 dB,

the phase margin is not less than $^{\Pi}/_{3}$ [rad].

The classical gain and phase margins are less conservative measures of the plant stability margin than the additive or multiplicative robust stability margins."[10]

The PID controller may be implemented in continuous or discrete time. The ideal continuous time PID controller is expressed in Laplace form as follows:

$$G(s) = K_p + \frac{K_i}{s} + K_d * s$$

Rewriting Laplace function *G(s)*

$$G(s) = \frac{Kd * s^2 + Kp * s + Ki}{s}$$

with K_p = proportional gain, K_i integral gain and K_d = derivative gain. The introduction of integral action facilitates the achievement of equality between the measured value and the desired value, as a constant error produces an increasing controller output. The introduction of derivative action means that changes in the desired value may be anticipated, and thus an appropriate correction may be added prior to the actual change. Thus, in simplified terms, the PID controller allows contributions from present, past and future controller inputs.

3.1 Does PID control have a future?[11]

It is quite reasonable to predict that PID control will continue to be used in the future. Feedback has had a revolutionary influence in practically all areas where it has been used and will continue to do so. PI(D) control is perhaps the most basic form of feedback. It is very

[10] Jerzy Mościcki Zbigniew Ogonowski „Advance Control with Matlab & Simulink" Ellis Horwood 1995 p.67

[11] Control Engineering Practice Volume 9, Issue 11 November 2001 Pages 1163-1175

effective and can be applied to a wide range of problems. The emerging features of automatic tuning have greatly simplified the use of PID control.

More knowledge about PID control has been available for a long time. Unfortunately, it has been buried in proprietary information of suppliers. There was a strong resurgence in the interest in PID control over the last 10 years. Many publications have appeared and it is typical that IFAC organized a workshop on PID control in the year 2000.

The alternatives to PID control are:

RST Discrete-time linear MISI controllers

SFO State feedback and observers

MPC Model predictive control

Fuzzy control is often mentioned as an alternative to PID control. Most fuzzy controllers used in industry have the same structure as incremental PI or PID controllers. The parameterization using rules and fuzzy membership functions makes it easy to add nonlinearities, logic, and additional input signals to the control law.

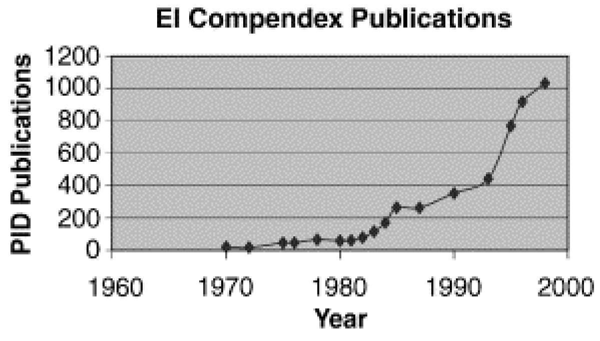


figure 3.2 PID publications historical evolution in the last 30 yr.

There has been a strong resurgence in the interest towards the PID control over the last few years. Many publications have appeared. Many of the new capabilities have been introduced by the research community. The industrial control users easily, and sometimes enthusiastically, apply these innovations. The PID control has become one of the most important ways, for the scientific and the industrial control users to work together.[12]

[12] Control Engineering Practice Volume 9, Issue 11 November 2001 Pages 1159-1161

3.2 Error Criterions

In order to select the best controller, we define a cost function. The cost function mainly derives on how the controller reacts to a given disturbance. There are many of cost functions. In fact, we can define infinitive criterions. The most popular are:

1. Integral of absolute value of error $IAE = \int_0^\infty [e(t)]\ dt$

2. Integral of error squared $ISE = \int_0^\infty [e(t)]^2 dt$

3. Time-weighted $ITAE = \int_0^\infty t[e(t)]\ dt$

Below we see step response for close loop system. The parameters of regulator are obtained using Genetic Algorithm[13].The fitness function are for different error criterions. As we notice there is no important what kind of criterions we assume.

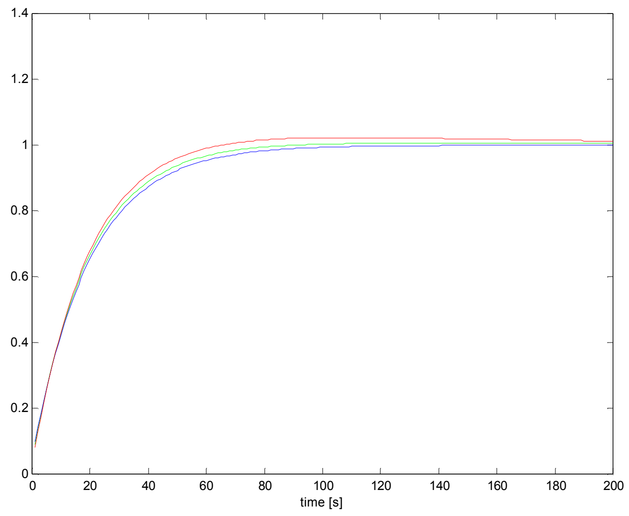


figure 3.3 Step response if cost functions is IAE (red) ISE (green) ITAE (blue)

„The question of which is the best criterion to use generally cannot be answered definitely. It depends on the structure and data of the problem. All rank tests using powers of matrices are numerically poor. Thus, if numerical algorithms to test controllability are to be use (..)"[14]

[13] more in chapter 4.2

[14] Jerzy Mościcki Zbigniew Ogonowski „Advance Control with Matlab & Simulink" Ellis Horwood 1995 p.23

4 Tuning rules

In my project I have used two methods of tuning PID. I have considered first order system, therefore classical method (Ziegler Nichols) is difficult to implement. In my case it was impossible. Gain Margin increasing to infinitive, and object is still stable.

4.1 Genetic Algorithms

When we classify different problems we can divide into two groups, first solvable and second unsolvable. In reality, what is solvable on in theory, need not to be solvable in sensible time (the time of solve can be equal to infinity). Of course complexity of calculations is very useful, because it may be used for security coding.

Searching is one of ways to solve problems For a lot of problems we are not able to construct an algorithm by definition method of searching step by step, but very often we can specify a set of potential solutions. Goal of strategy of searching is to analyze elements of set in order to fix the best one. It is easy for small sets but if the set increases it becomes more and more complicated or impossible. One of the most advanced and very modern searching methods are genetic algorithms.

Genetic algorithms (GA) are stochastic global search methods based on the mechanics of natural selection and natural genetics. They are iterative methods, widely used in optimization problems in several branches of the Sciences and Technologies. Contrary to what happens in other methods, in this methodology, in each iteration (generation), not only one point in the search space is taken into account, but a set of solutions defining a population or populations of individuals is considered. These individuals will then be ranked, according to the quality of the solution that each one can lead to.[15]

15 Mathematics and Computers in Simulation Volume 51, Issues 3-4 January 2000 Pages 287-300

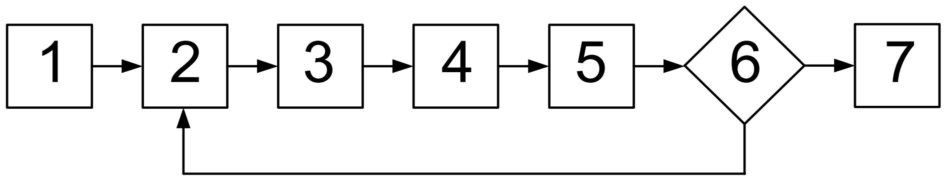


figure 4.1 Genetic algorithm loop

1) Initial population

2) Select individuals for mating

3) Mate individuals to generate offspring

4) Mutate offspring

5) Insert new individuals into population

6) Are criteria satisfied?

7) End of searching

As we can notice above a loop of evolution is closed. Replication starts from base point again and the best individuals are chosen. The selection of chromosomes is a random process, but it is very strongly directed for choosing the best individuals for reproduction. A common practice is to terminate the GA after number of generation and then test the quality of the best number of the population against the problem definition. If no acceptable solutions are found, the GA may be restarted or a fresh search initiated.

4.1.1 Chromosomal representation

Individuals are encoded as strings, chromosomes, composed over some alphabet, so that the genotypes are uniquely mapped onto the decision variable domain. The most often used for representation in Gas is the binary alphabet. Here each decision variable in the parameter set is encoded as a binary string and these are concatenated to form a chromosome.

The chromosomes are seen as binary strings of given length, and decoded into real numbers over a specified interval using either standard binary or Gray coding. The use of Gray coding for binary chromosome representation is recommended as the regular Hamming distance between quantization intervals reportedly makes the genetic search less deceptive.

4.1.2 Population

The population size is the number of chromosomes in the population. Having decided the representation, the first step is to create an initial population. This is usually achieved by generating the required number of individuals using a random number generator that uniformly distributes numbers in the desired range. For example, with the binary population of N_{ind} individuals whose chromosomes are L_{ind} bits long, N_{ind} x L_{ind} random numbers uniformly distributed from the set {0,1} would be produced.

There exist three main variations of genetic algorithms: simple, steady-state and struggle population.

4.1.3 Simple GA

The simple genetic algorithm is one of the common genetic algorithm implementations. The simple genetic algorithm uses non-overlapping population. In each generation, the entire population is replaced with new individuals. Typically the best individual is carried over from one generation to the next so that the genetic algorithm does not inadvertently forget the best that it found. Since the entire population is replaced each generation, the only "memory" the algorithm has from performance of the crossover operator.

4.1.4 Steady state GA

The steady state genetic algorithm uses overlapping population. In each generation, the newly generated individuals replace a portion of the population. At one extreme, only one or two algorithm can became a simple genetic algorithm when the whole entire population is replaced. Since the algorithm only replaced a portion of the population of each generation, the best individuals are more likely to be selected and the population quickly converges to a single individual.

4.1.5 Struggle GA

The struggle genetic algorithm is similar to steady-state GA. However, rather that replacing the worst individual, a new individual replaced the individual most similar to it, but only if the new individual has a score better than that which is the most similar.

4.1.6 Fitness function

A fitness function must be devised for each problem, given a particular chromosome, the fitness function returns a single numerical fitness value, which is proportional to the ability, or utility, of individual represented by that chromosome.

For many problems deciding upon the fitness function is very straight forward, for example for a function optimization search, the fitness is simply the value of the function \. However there are cases where may be performance measured to optimize.

Ideally, the fitness function should be smooth and regular so those chromosomes with reasonable fitness are closer in the search space, to chromosomes with slightly better fitness. However, it is not always possible to construct such an ideal fitness function. For any type of search to be successful, the fitness function must not have many local maximums, or a vary isolated global maximum.

4.1.7 Genetic Algorithms operations

4.1.7.1 Selection

The purpose of parent selection in a GA is to give more reproductive changes to those individuals that are the fit. There are many ways to do it, but one commonly used technique is *roulette wheel parent selection* (RWS), which contains three basic steps:

1. sum the fitness of all population members and name the result total fitness,
2. generate *n*, a random number between 0 and total fitness,
3. return the first population members whose fitness, added to the fitness' of the preceding population members, is greater or equal to *n.*

The effect of roulette wheel parent selection is to return a randomly selected parent. Using this selection algorithm, each parent's chance of being selected is directly proportional to its fitness, but its possibility also exists to choose the worst population member.

A second very popular way of selection is *stochastic universal sampling* (SUS). This way is a single-phase algorithm with minimum spread and zero bias.

4.1.7.2 Crossover (recombination)

The basic operator for producing new chromosomes in the GA is that of crossover. Like in nature, crossover produces new individuals, which have some parts of both parents' genetic material. The simplest form of crossover is that of single – point crossover.

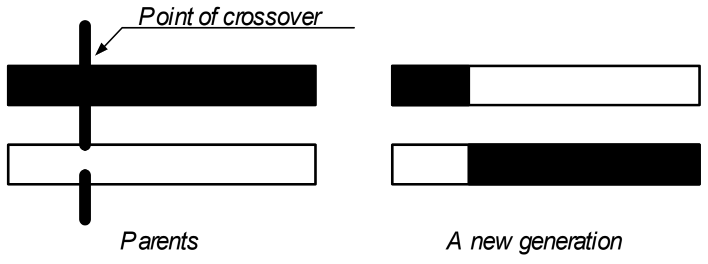


figure 4.2 Single point crossover

Of course there exist other crossover variations such as dual point, multipoint, uniform, shuffle, asexual crossover, and single child crossover.

4.1.7.3 Mutation

Mutation causes the individual genetic representation to be changed according to some probabilistic rule. In the binary string representation, mutation cause a random bit to change its state (0→1 or 1→0) and it illustrated in figure below. In natural evolution, mutation is randomly applied with low probability, typically in the range 0.001 and 0.01, and modifies element in the chromosomes. Given that mutation is generally applied uniformly to an entire population of string, it is possible that a given binary string may be mutated at more than one point.

Place of m utation

Original string: 01010110

M utated string: 01110110

figure 4.3 Binary mutation

4.1.7.4 Reinsertion

Once selection and recombination of individuals from the old population have produced a new population, the fitness of the individuals in the new population may be determinate. If fewer individuals are produced by recombination than the size of the original population, than the fractional difference between the new and old population sizes in termed a generation gap.

To maintain the size of the original population, the new individuals have to be reinserted into the old population. Similarly, if not all the new individuals are to be used at each generation or if more offspring are generated than size of old population then a reinsertion scheme must be used to determine which individuals are to exist in the new population. When selecting which members of the old population should be relocated the most apparent strategy is to replace the least fit members deterministically. The replacement should select and replace the oldest members of the population.

4.1.8 Genetic Algorithm in Matlab

The Genetic Algorithm Toolbox[16] tries to maximize a function using a simple (haploid) genetic algorithm to find a maximum of the fitness function. This toolbox consist of next ten files

B10TO2	Converts base 10 to base 2.
DECODE	Converts from binary to variable representation.
ENCODE	Converts from variable to binary representation. This script demonstrates the use of the simple genetic algorithm
GENETIC	tries to maximize a function using a simple genetic algorithm.
GENPLOT	Graphing routine for genetic algorithm.
MATE	Randomly reorders (mates) OLD_GEN.
MUTATE	Changes a gene of the OLD_GEN with probability Pm.
REPRODUC	Selects individuals proportional to their fitness.

16 download from ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/

TESTGEN Tests genetic algorithm code with f(x)=x^10.

XOVER Creates a NEW_GEN from OLD_GEN using crossover.

4.2 Result of tuning using Genetic Algorithm

4.2.1 First order system

Below are figures counted for plant such as the tank. $K(s) = \frac{1}{180s+1}$

Time of computing in both cases (PID and PI) about 30 [sec]

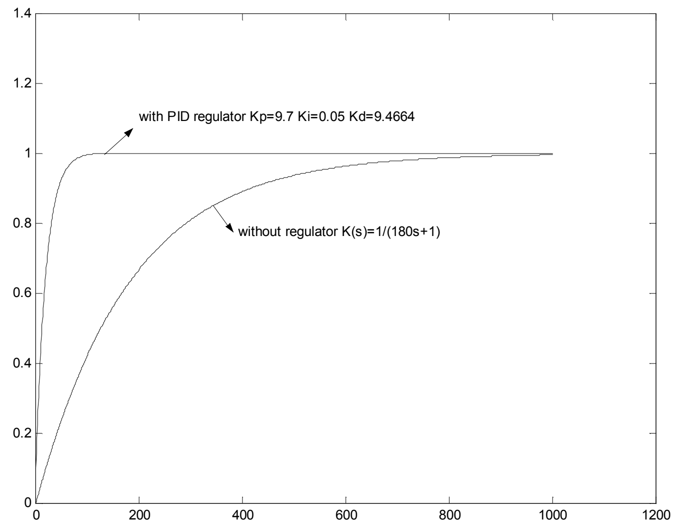


figure 4.4 Step response for PID regulator, parameters obtained by GA

For the figure above, we have next error criterions

IAE=18.2355

ISE=8.3939

ITAE= 433.2107

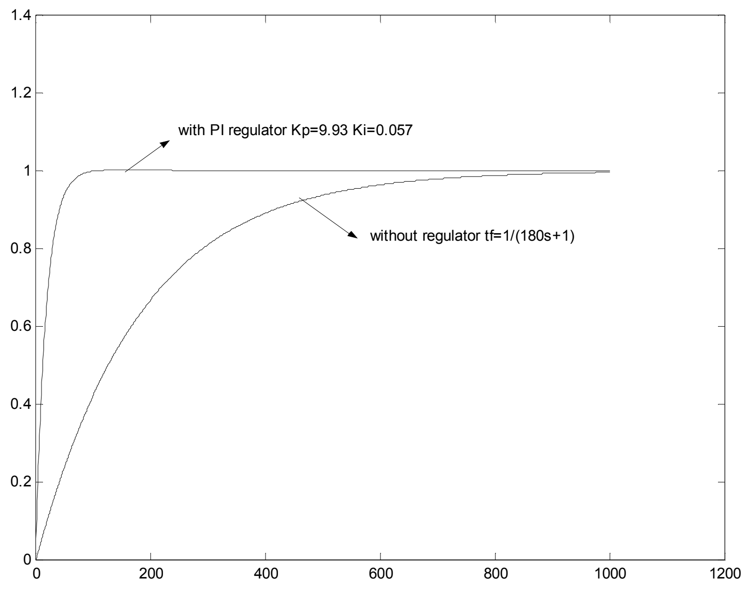


figure 4.5 Step response for PI regulator, parameters obtained by GA

For the figure above, we have next error criterions

IAE=17.7906

ISE= 8.5279

ITAE= 447.0941

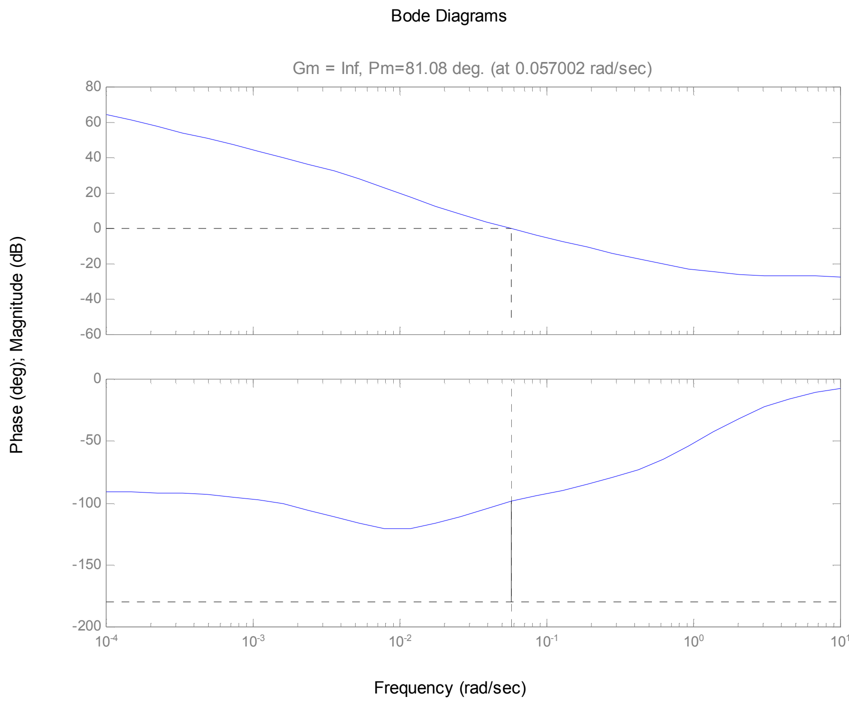


figure 4.6 Bode frequency response of the plant with regulator in open loop system

On the figure above, we see that gain margin is infinitive for this plant. It causes that classical Ziegler – Nichols method[17] does not work for this plant.

4.2.2 First order system, GA tuning without an initial population

I assume that genetic algorithm should to compute as long as conditions are satisfied, but not more than 100 iterations. It is approximately 100 [sec.]

Options in file genhaploid.m[18]

```
options = foptions([1 1e-3]); %count up to 1e-3
options(14) = 100;                          %The default maximum generations
```

No	No of iterations	Kp	Ki	Kd	IAE
range of tuning		0÷10	0÷10	0÷10	
1	21	9.1850	0.050965	1.8074	19.13
2	66	9.9454	0.0569	1.8137	17.71
3	36	9.9544	0.05874	2.3857	17.8739
4	29	9.9919	9.6333	1.2004	20.3752
5	72	10.000	0.05859	1.5390	17.73
6	100	9.9203	0.055695	2.7842	17.69
7	8	9.9963	0.0594	2.8087	17.84
8	75	9.8356	0.0544	4.0153	17.87
9	27	9.9951	0.05965	4.0008	17.56
10	85	9.9887	0.05676	6.503	17.66
Average	51.9				18.143

table 4.1 First order system, GA tuning without an initial population

I assume that if algorithm had not found solution for 100 iterations than stops. As we can notice in sixth attempt it was happened. Maximum number of generations reached without termination criterion met. The solution is that either increase maximum generations or ease termination criterion.

4.2.3 First order system, GA tuning with an initial population

What happened if give solution in an initial population?

```
options = foptions([1 1e-3]); %count up to 1e-3
options(14) = 100;                          %The default maximum generations
```

[17] Ziegler, J.G. and Nichols, N.B. (1942) Optimum settings for automatic controllers circuit. *Transaction of the ASME,* November 759-768

[18] details in appendix A

```
x0=[4.0008 9.9951 0.05965];
```

No	No of iterations	Kp	Ki	Kd	IAE
range of tuning		0÷10	0÷10	0÷10	
Initial point		9.9951	0.05965	4.0008	17.56
1	38	10	0.0555	4.9966	17.56
2	53	9.9260	0.0569	4.2506	17.79
3	74	9.9968	0.0565	3.3396	17.58
4	15	9.9151	0.0597	4.0008	17.89
5	68	9.9626	0.0574	5.8843	17.76
6	19	9.9951	0.0586	2.1282	17.75
7	33	9.9826	0.0546	2.0430	17.81
8	75	9.9951	0.0597	4.0008	17.89
9	11	9.9151	0.0589	0.9526	17.77
10	100	9.9484	0.0503	3.5195	17.85
Average	48.6				17.765

table 4.2 First order system, GA tuning with an initial population

The different between both cases (with and without initial population) is not significant. This is 6.79% more iterations in first case (without an initial population) and average error 2.15% higher than second case (with an initial population)

4.2.4 First order system, GA tuning with wide range PID parameters

In third case I assume that bounds are 0÷10000.

No	No of iterations	Kp	Ki	Kd	IAE
range of tuning		0÷10000	0÷10000	0÷10000	
1	100	6787.2	50.812	1404.1	$8.3*10^{-3}$
2	18	3378.1	9780.8	123.75	$9.2*10^{-3}$
3	25	1985.8	9822.8	247.3	$5*10^{-3}$
4	100	7088.1	39.4	327.5	$34.79*10^{-6}$
5	100	8378.1	46.5	164.8	$14.66*10^{-12}$
6	8	4794.4	4965.3	4794.4	$12.4*10^{-3}$
7	100	9009.5	50	109.7	$1.86*10^{-9}$
8	100	9890.3	54.9	437.5	$52.5*10^{-9}$
9	100	8900.4	49.4	247	$2.29*10^{-9}$
10	100	2359.1	9952.5	82.7	$43.98*10^{-6}$

table 4.3 First order system, GA tuning with wide range PID parameters

Computing for this case takes more time. The error is very small. In industry we do not use regulators with so huge gains, so even we find solution we do not implement into real object.

4.2.5 Plant with strongly oscillations

Lets consider following plant $K(s) = \dfrac{1}{50s^3 + 43s^2 + 3s + 1}$

Fitness function counted for IAE

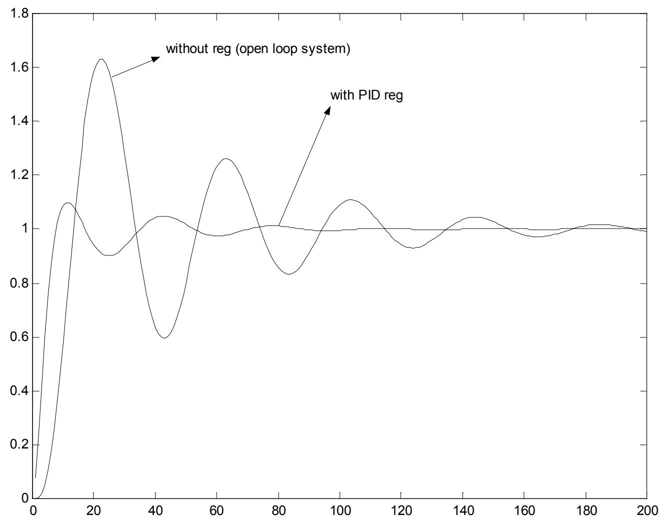


figure 4.7 Step response with and without PID regulator, fitness function IAE

Gains:

Kp=0.93935	Ki=0.2681	Kd=9.7963

Errors

IAE=5.90	ISE=2.17	ITAE=110.02

With PID regulator it is not important what kind of fitness function we assume. Genetic Algorithm finds almost the same solution (PID parameters). The different between gains is because GAs use probabilistic method.

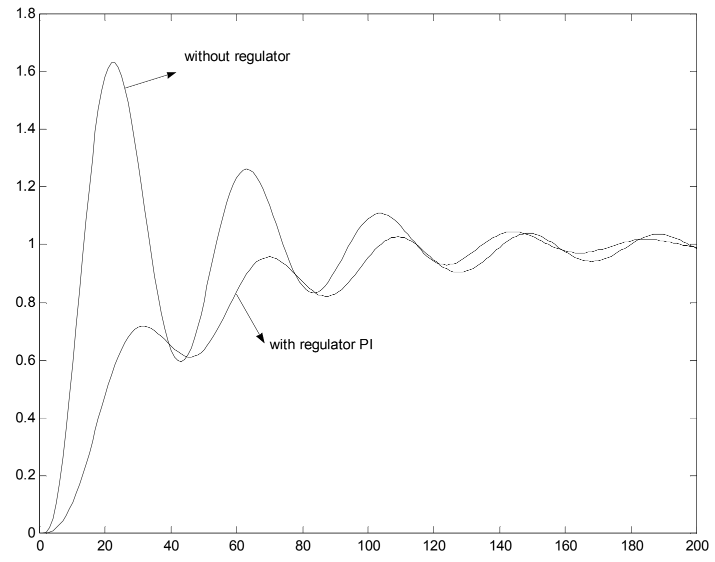


figure 4.8 Step response with and without PI regulator, fitness function IAE

Gains:

Kp=0.061 Ki=0.027

Errors

IAE=40.74 ISE=19.5981 ITAE=2569

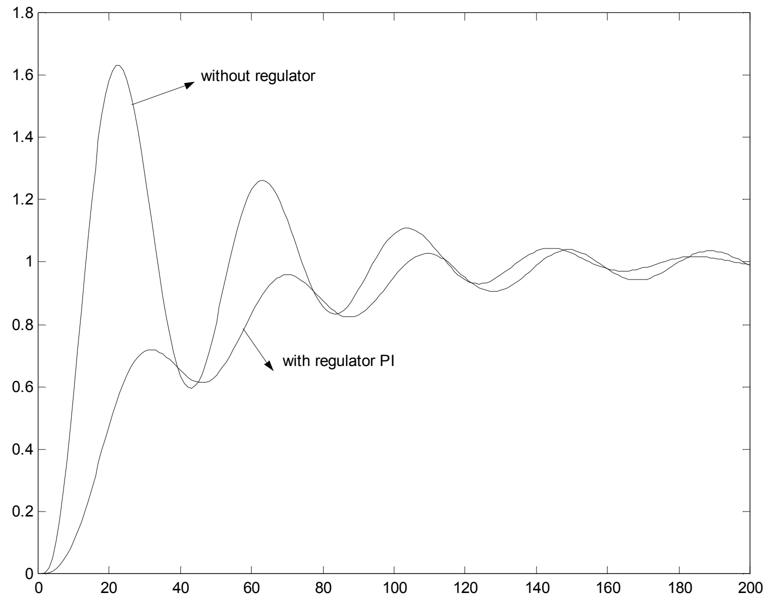


figure 4.9 Step response with and without PI regulator, fitness function ISE

Gains:

Kp=0.057984 Ki=0.027466

Errors

IAE= 40.7471 ISE= 19.6519 ITAE=2656

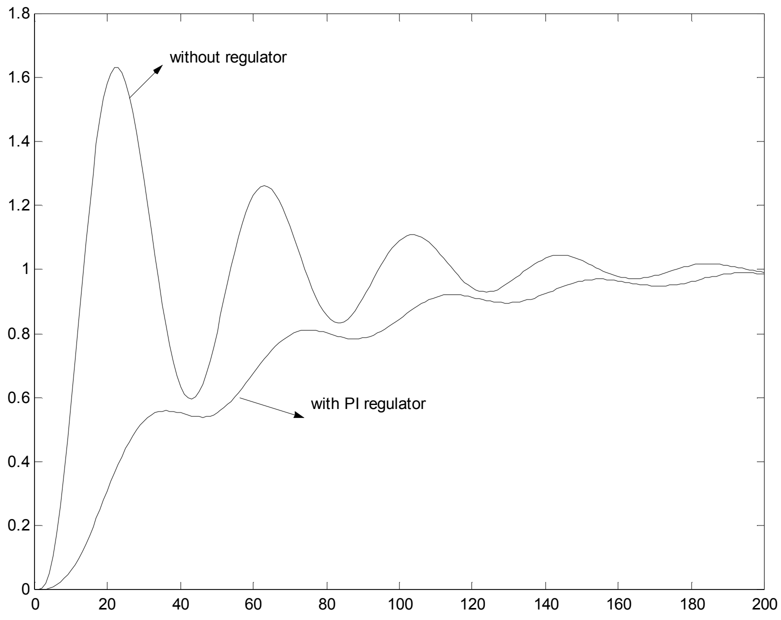


figure 4.10 Step response with and without PI regulator, fitness function ITAE

Gains:

Kp=0.020752 Ki=0.019

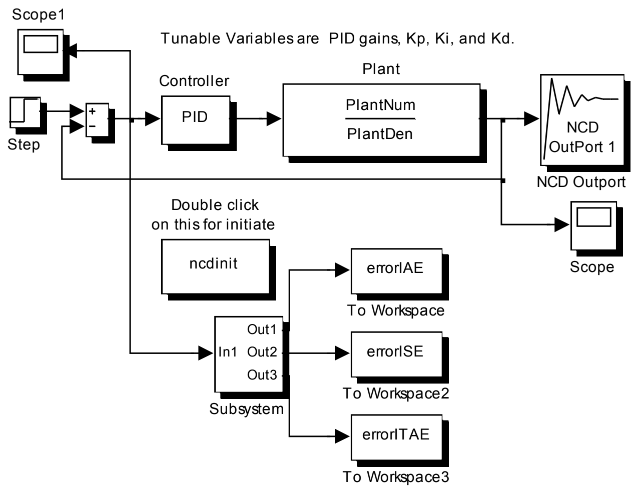
Errors

IAE= 51.9914 ISE= 27.3662 ITAE=2657

PID parameters are almost the same for fitness function IAE and ISE, only ITAE gives other (worse) solution.

4.3 Nonlinear Control Design Blockset

To use the NCD Blockset, connect the block to any signal in a Simulink model to signify that you want to place a constraint on it. The NCD Blockset automatically converts time-domain constraints into a constrained optimization problem and then solves the problem using state-of-the-art optimization routines. The optimization problem formulated by the NCD Blockset iteratively calls for simulations of the system, compares the results of the simulations with the constraint objectives, and uses gradient methods to adjust tuneable parameters to better meet the objectives.



Double click here to initialize plant data and optimization parameters.
Double click on NCD Blockset block to display constraint window.

figure 4.11 Model NCD in Simulink

4.4 Results of tuning using Nonlinear Control Design Blockset

4.4.1 First order plant

Transfer function $K(s) = \dfrac{1}{180s+1}$

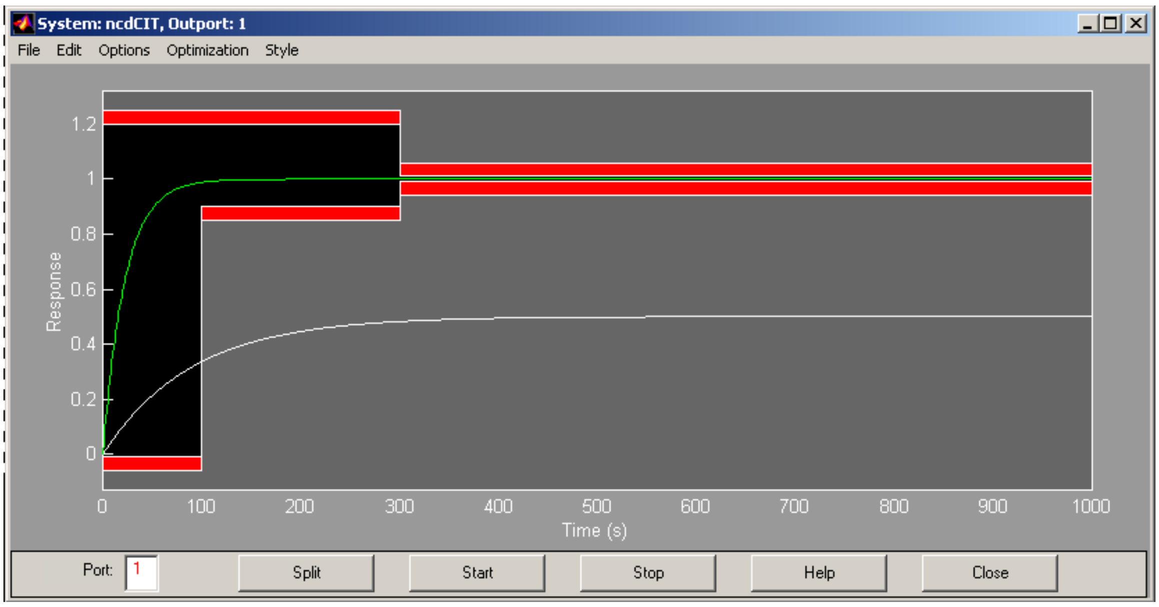

System: ncdCIT, Outport: 1
File Edit Options Optimization Style
Response
Time (s)
Port: 1
Split
Start
Stop
Help
Close

figure 4.12 NCD blockset with long settling time

Time of counting: 73.76 [sec.]

Gains:

Kp=8.0221	Ki=0.0446	Kd=-0.2544

Errors

IAE= 22.4054	ISE= 11.2029	ITAE = 502.0037[19]

4.4.2 First order plant with short settling time

In the second case I set bounded in this way that settling time is shorter. NCD finds solution with bigger Kp.

19 time of simulation 1000 [sec]

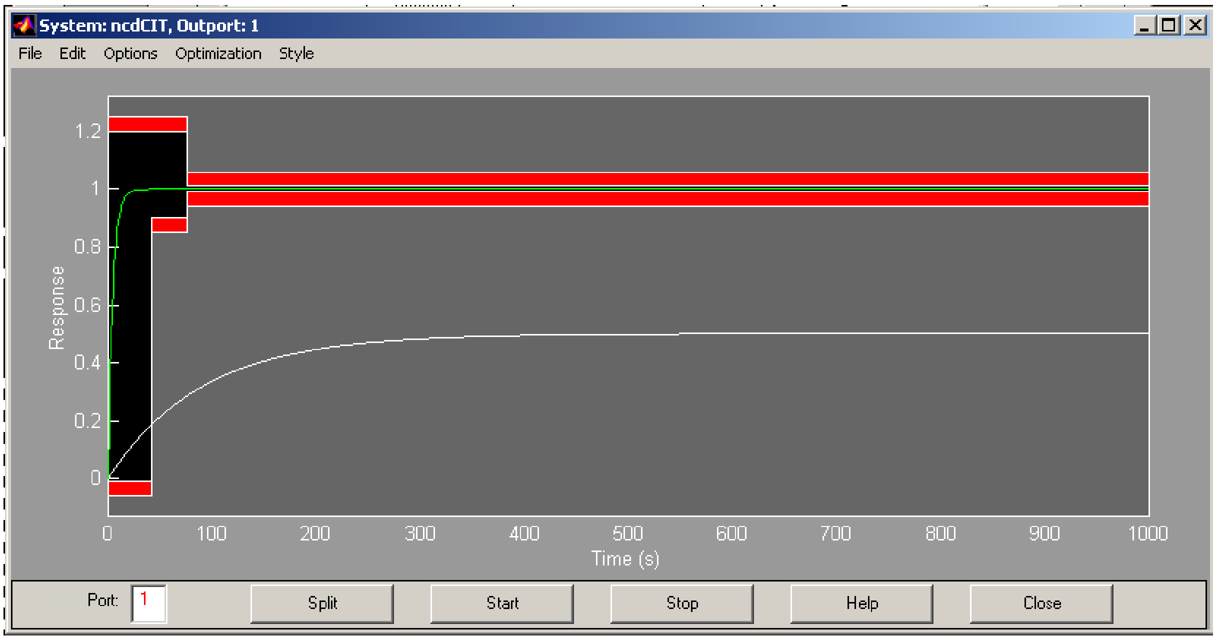


figure 4.13 NCD blockset with short settling time

Gains

Kp=38.81 Ki=0.2164 Kd=-0.59981

Errors

IAE= 4.1627 ISE= 1.8586 ITAE= 24.1821

As we can notice Errors are much lower, than first case. This is advantages NCD. We can define bounds as we want.

4.4.3 Plant with strongly oscillations

Let's consider following transfer function

Plant $K(s) = \dfrac{1}{50s^3 + 43s^2 + 3s + 1}$

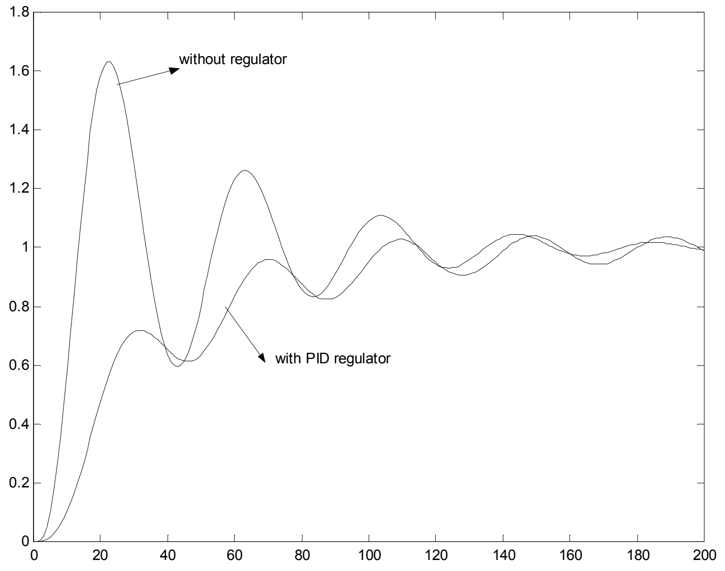


figure 4.14 Step response with and without PID regulator, NCD tuning

Gains obtained by NCD

Kp=0.694 Ki=0.0573 Kd=2.26

Errors

IAE=40.741 ISE=19.6519 ITAE=2565

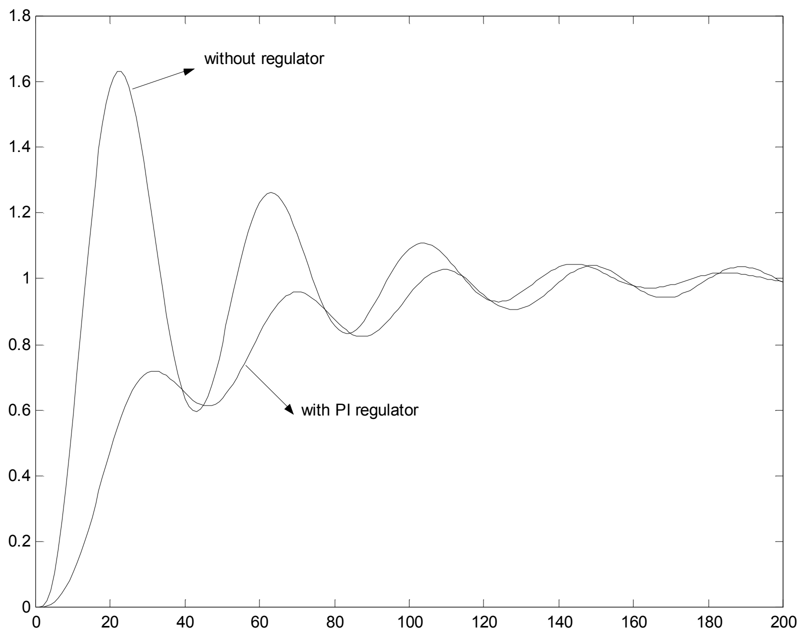


figure 4.15 Step response with and without PI regulator, NCD tuning

Gains:

Kp =0.26214 Ki=0.027916

Errors:

IAE= 40.7471 ISE=19.6519 ITAE=2565

The NCD Blockset block opens the interactive time-domain constraint figure and displays the constraints, which can be modified to shape the desired output step response. The lower and upper constraint bounds effectively define overshoot, rise time, and settling time. The response evolves and improves throughout the optimization to better achieve the constraints on system signals defined by the NCD main interface.

The NCD Blockset can aid in the design and identification of complex control systems (including gain scheduled and multimode control and repeated parameter problems).

Sequential Quadratic Programming (SQP) SQP methods represent state-of-the-art in nonlinear programming methods. The method allows you to closely mimic Newton's method for constrained optimization just as is done for unconstrained optimization. At each major iteration an approximation is made of the Hessian of the Lagrangian function using a =1quasi-Newton updating method. This is then used to generate a QP sub-problem whose solution is used to form a search direction for a line search procedure. The principal idea is the formulation of a QP sub-problem based on a quadratic approximation of the Lagrangian function.

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_1 * g_1(x)$$

The QP sub-problem is obtained by linearizing the nonlinear constraints.

4.5 Compare searching methods

It is not possible to say that one of these methods which I considered was better. Everything depends on what kind of criteria we assume. Advantages of Genetic Algorithm is that they are quicker than NCD, however disadvantages are that we have to define a fitness function, assume error criterions, and eliminate undesirable overshoot. It causes that using NCD is easier than GA According to me it is also more powerful. Let's consider prices, GA toolbox we can download for free from Mathworks server[20] until now this toolbox is not supported by Mathworks. In order to use NCD we have to purchase Matlab, Optimization Toolbox and mentioned NCD.

[20] ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/

Conclusions

In this report I have proposed the joint use of genetic algorithms and Nonlinear Control Design Blockset for PID tuning. This approach was applied to the control of a time invariant plant. It is not possible to say exactly which method is the best. It depends what kind of criteria we assume. The easiest way to find PID parameters is using NCD. This toolbox allows us to shape the function according to our requirements. Genetic Algorithms are useful for complicated problems.

At the beginning of my project I focused on identification methods. First at all I have filtered data obtained from the tank. I have used two methods available in Identification Toolbox. One method (Reaction Curve Method) was implemented by me. The different between them is not significant. Relative error is almost the same for all cases. However the winner was Box – Jenkins. The results were compared for a first order system. I assume that for second order systems, the difference between a classical method and those methods from the Identification Toolbox could be more significant.

The Graphic User Interface allows the user to easily compare all the methods. The results of Identification and tuning are computed every time we run the program. In future this program could be developed by the Astrom test i.e. we could have 11 typical methods implemented into GUI. It would result in a more interesting and more educative program.

Future work will embed the complete autotuning approach into an industrial PID controller, performing real-time control of a real plant.

References

1. Brogan W.L. “Modern Control Theory” Third edition, Prentice Hall
2. Camacho E.F. and C. Bordons, “Model Predictive control in the process industry”
3. Deasy Paula „PID Parameter optimisation using Genetic Algorithms – The Astrom Benchmark Test” DLX4 Project 2001 CIT
4. Fatla Klaudiusz „Optymalizacja nastaw regulatora wzbudzenia generatora synchronicznego, przy pomocy algorytmów genetycznych” Thesis Master of Science, Wroclaw 1999/2000
5. Fatla Klaudiusz „Using Genetic Algorithms for Optimaization of PID Controller” DLX4 Project 1999 CIT
6. http://galaxy.uci.agh.edu.pl/~ttward/iden/ Laboratory of identification object 13/03/02
7. http://www.mathworks.com
8. http://www.sciencedirect.com
9. Matlab : System Identification Toolbox User’s Guide
10. Matlab Control System Toolbox User’s Guide
11. Matlab Nonlinear Control Design Blockset User’s Guide
12. Matlab Optimization Toolbox User’s Guide
13. Mościcki Jerzy, Ogonowski Zbigniew „Advance Control with Matlab & Simulink” Ellis Horwood 1995
14. Munoz Irene Mas „System Identification Using Genetic Algorithm” DLX4 Project 2001 CIT
15. O’Mahony Tom „Real-time Control of a Single-tank System using Humusoft” Electronics Dept., CIT 2000
16. O’Mahony Tom, C.J. Dowing, Fatla Klaudiusz, “Genetic Algorithms for PID Parameter Optimisation: Minimising Error Criteria”

Appendixes

Appendix A m-files

ident_rt.m

```
% ident_rt.m
%
% measure data and filtering from a plant in real time
%
% Author:      Daniel Czarkowski DLX4
%                              daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department


%Variables
SamplingPeriod=1;             % sampling period
N=1000;                       % number of samples
u=1*zeros(N,1);               % prepare the output vector
Vout=zeros(N,1);

rtclear;                                          % clear drivers
rtload('ad512',256,[0 3 3 3 3 3 3 3 3]);  % load the HUMUSOFT AD512 driver
rtdef(1,'out',SamplingPeriod,1);     % define output timer
rtdef(2,'in',SamplingPeriod,1);      % define input timer
rtstart all;                         % start the timers


%Main loop
tic;                          % Start a stopwatch timer
disp 'busy'
for i=1:N;
   if round(rttool('Read',1,'Run'))==1
        while (rtrd(2,'t'))<i; end;   % wait for the next sample
                rtwr(1,'y',u(i));
                Vout(i)=rtrd(2,'y');  % read the input value
                Voutend=round(Vout*1000000); % round and multimle
      if i>120
         and_=and(Voutend(i-1)==Voutend(i),Voutend(i-120)==Voutend(i));
         if and_==1   % stop pump when two samples are equal
            disp 'OK'
            rtstop (1)
                Na=i;                % measure number of samples
         end;
      end;
   end;
end;
MeasurementTime=toc   % stop a stopwatch timer

rtwr (1,-1);                              % stop a pump
rtstart (1)                                      % start timer 1
try
   Na;
catch
   Na=N;
end

Vout=Vout(1:Na);                                        % initial matrix
N=Na                                                    % number of samples

Vout1=(Vout(1)+Vout(2))/2;                       % mean value saples 1 & 2
VoutOffset=Vout-Vout1;                           % subtract offset
Vout=VoutOffset;                                        % new data without offset

%http://www.mathworks.com/access/helpdesk/help/techdoc/ref/polyfit.shtml
%Polynomial curve fitting
x=[1:SamplingPeriod:N]';
p = polyfit(x,Vout,5);                                  % fit polynomial to data
ypoly = polyval(p,x);                                   % measured data after estimation
%table = [x y f y-f];
%y = erf(x);                                     % error function
clear p Na Voutend SamplingPeriod i and_ u Vout1
disp 'Done measure process'
plot (x,Vout,'r',x,ypoly,'b');
axis auto;

disp ('error of approximation[%]')
sum((abs(Vout-ypoly)./Vout))/length(ypoly)*100
title('Measured (red), Polynomial curve fitting (blue)');
save datafile;               % spare datafile is called 1000.mat
```

rtstop1.m

```
% rtstop1.m
%
% emergency stop of measuring data
%
% Author:      Daniel Czarkowski DLX4
%                             daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department

clear all
```

```
clc
rtclear;
Ts = 0.01;                          % sampling period
rtload('ad512',256,[0 3 3 3 3 3 3 3 3]);% load the HUMUSOFT AD512 driver
rtdef(1,'out',Ts,1);                                 % define output timer
rtdef(2,'in',Ts,1);                                  % define input timer
rtwr(1,-1);                        % write the initial condition
rtstart all                           % start timers
```

ident_calc.m

```
% ident_calc.m
%
% identification of a plant from datafile
%
% Author:      Daniel Czarkowski DLX4
%                            daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department



%Variables
u=ones(N,1);                          % u-input  ypoly-output
z=[ypoly u];                          % data from file datafile.mat
                                             % or script ident_rt.m

if IdentMethods==1           %identification using classical method
k=(ypoly(N)-ypoly(1))/1; %(u(N)-u(1));
stepchange=(1-exp(-1))*k;     % 63% of settling value
final_value=(stepchange+ypoly(1));
for i=1:N;                            %find the first sample
   if ypoly(i) <= final_value;
      ts=x(i);
   end
end
disp 'plant computes using a classical method'
PlantS=tf([k],[ts 1]) % transfer function of a plant
sim('pid_fine_ol');           % Simulate a simulink model
plot(yout,'r');
title('Classical method (red)');


elseif IdentMethods==2
%Computes the prediction error estimate of a Box-Jenkins model.
th=bj(z,[1 0 0 1 0]); % z-meassured data
[Num,Den]=th2tf(th);% transforms from the THETA-format to transfer functions.
disp 'Plant computes using a Box-Jenkins model'
plantbj=tf(Num,Den,1);
```

```
PlantS=d2c(plantbj,'zoh')     % conversion of discrete LTI models to continuous time.
                                     % 'zoh' Assumes zero-order hold on the inputs.
sim('pid_fine_ol');           % Simulate a simulink model
plot(yout,'g');
title('Box-Jenkins (green)');

elseif IdentMethods==3
%Computes LS-estimates of ARX-models.
disp 'plant computes using LS'
th=arx(z,[1 1 1]);            %  Computes LS-estimates of ARX-models.
[Num,Den]=th2tf(th);
plantarx=tf(Num,Den,1);
PlantS=d2c(plantarx,'zoh')    % conversion of discrete LTI models to continuous time.
                                     % 'zoh' Assumes zero-order hold on the inputs.
sim('pid_fine_ol');           % Simulate a simulink model
plot(yout,'b');
title('Least squares (blue)');
end
clear IdentMethods plantarx plantbj th ts i z u
main;
```

ncdinit.m

```
% ncdinit.m
% ncdinit Sets up necessary data files for optimization of ncdCIT.mdl.
%
%
% Author:      Daniel Czarkowski DLX4
%                            daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department
%
% Define Optimization parameters
%     - incremental step size  : ncdStruct.Tdelta
%     - tunable parameters     : ncdStruct.TvarStr
%     - optimization options   : ncdStruct.OptmOptns
%     - lower bound limits     : ncdStruct.TvlbStr
%     - upper bound limits     : ncdStruct.TvubStr


ncdglob; % declare globals

ncdStruct.Tdelta = 1;
ncdStruct.OptmOptns = [1 0.001 0.001];
ncdStruct.TvlbStr = '';
ncdStruct.TvubStr = '';
try
   contr;
catch
   contr=1
end

if contr==1
   ncdStruct.TvarStr = 'Kp Ki Kd';
   disp('PID')
else
   ncdStruct.TvarStr = 'Kp Ki';
   disp('PI')
end

try
   PlantS
catch
PlantNum=1;
PlantDen=[180 1];
end

Kp=1; % Define Tunable Variable initial values
Ki=0;
```

```
Kd=0;

ncdCIT;                                  %open simulink model
disp('Done initializing')
```

genhaploid.m

```
% genhaploid.m
%
% Genetic Algorithm implemented into PID regulator
% This script calls Genetic Algorithm Toolbox
% Download toolbox from:
% ftp://ftp.mathworks.com/pub/tech-support/solutions/s1248/genetic/
%
% Author:      Daniel Czarkowski DLX4
%                             daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Departmenttry

try
PlantNum;
catch
PlantNum=[1];               %default Numerator
PlantDen=[180 1];           %default Denumerator
end

try
   ErrorEstim;              % Type of performance index
catch                                     % 1-IAE 2-ISE 3-ITAE
   ErrorEstim=1             % default IAE
end
try
   contr;                           % Type of controller (1=PID 2=PI)
catch
   contr=1                          %default PID
end

try
   N;                                     % Number of samples
catch
   N=1000;
end

if contr==1
   disp('PID')
x=([0 1 0])';                             % initial values
vlb = [-10 -10 -10];        % lower bounds
vub = [10 10 10];                   % upper bounds
bits = [16 16 16];
Kd=x(1);
Kp=x(2);
Ki=x(3);
PIDgain=[Kd Kp Ki];
```

```
else
   disp('PI')
x=[1 0]';                          % initial values
vlb = [0 0];                 % lower bounds
vub = [10 10];               % upper bounds
bits = [16 16];
Kp=x(1);
Ki=x(2);
PIDgain=[Kp Ki];
end

PlantS=tf(PlantNum,PlantDen);
PIDden=[1 0];
PIDreg = tf(PIDgain,PIDden);        % Trasfer function of PID Controller
oltf= PlantS*PIDreg;                              % Open Loop Transfer Function

cltf=PIDreg*PlantS;                               % Close Loop Transfer Function
cltf = feedback(cltf,1,-1);
[y t] = step(cltf);          % Step response of closed-loop system in time domain
error = 1-y;
if ErrorEstim==3
   error = error.*t;
   disp('ITAE');
end
error = abs(error);
error = sum(error);
if ErrorEstim==2
   error = error.^2;
   disp('IAE')
elseif ErrorEstim ==1
   disp('IAE')
end
fitness = error                    %error without PID regulator

%options = foptions([1 -1]);
options = foptions([1 1e-4]); %count up to 1e-4
options(13) = 0.03;
options(14) = 50;                                 %The default maximum generations
tic
%x0=[50 100 0];
warning off
[x,stats,options,bf,fg,lg] =
genetic('genpid',[],options,vlb,vub,bits,N,PlantS,ErrorEstim,contr);
warning on
toc
%STATS   - [max min mean std] for each generation
%            OPTIONS - options used
%            BESTF   - Fitness of indivadual X (i.e.: best fitness)
%            FGEN    - first generation population
%            LGEN    - last generation population
% The highest point found by the genetic algorithm is
```

```
bf;

if contr == 1
   Kp=x(2);
   Ki=x(3);
   Kd=x(1);
        PIDgain=[Kd Kp Ki];
else
   Kp=x(1);
        Ki=x(2);
        PIDgain=[Kp Ki];
end

PIDreg = tf(PIDgain,PIDden);         % Trasfer function of PID Controller

cltf=PIDreg*PlantS;
cltf = feedback(cltf,1,-1);
[y t]=step(cltf,1:1:1000);
plot(y,'g');
title ('Optimalization using GA')
xlabel ('time [sec]')
axis auto;

error = 1-y;

if ErrorEstim==1
   error = abs(error);
   error = sum(error);
   errorIAE=error;
   disp('IAE')
   legend(num2str(errorIAE),'IAE');
elseif ErrorEstim==2
   error = error.^2;
   error = abs(error);
   error = sum(error);
   errorISE=error;
   disp('ISE')
   legend(num2str(errorISE),'ISE');
elseif ErrorEstim==3
   error = error.*t;
   error = abs(error);
   error = sum(error);
   errorITAE = error;
   disp('ITAE')
   legend(num2str(errorITAE),'ITAE')
end
fitness = error;
clear errorIAE errorISE errorITEA cltf oltf bf t ulb vlb
main;
```

genpid.m

```
% genpid.m
%
% Genetic Algorithm implemented into PID regulator
% This function is called by genhaploid.m
%
% Author:      Daniel Czarkowski DLX4
%                             daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Departmenttry

function fitness = genpid(x,N,PlantS,ErrorEstim,contr)

if contr == 1 %values for PID regulator
   Kd=x(1);
        Kp=x(2);
   Ki=x(3);
        PIDgain=[Kd Kp Ki];
else                  %values for PI regulator
   Kp=x(1);
        Ki=x(2);
        PIDgain=[Kp Ki];
end




PIDden=[1 0];
PIDreg = tf(PIDgain,PIDden);         % Trasfer function of PID Controller

cltf=PIDreg*PlantS;                               % open loop system
cltf = feedback(cltf,1,-1);          % Close loop system
[y t]= step(cltf,1:1:N);                     % take matrix [y]

error = 1-y;
if ErrorEstim==3                                         % ITAE
   error = error.*t;
end
if max(y) > 1.1                                          % eliminate overshoot
   error=error.*5;
end
error = abs(error);                               % IAE
error = sum(error);                               % IAE
if ErrorEstim==2                                  % ISE
   error = error.^2;
elseif ErrorEstim==3
end
fitness = 1/error;                                % minimalise error
```

main.m

```
% main.m
%
% This script causes that the results in figure with close
% loop system  are easy to undestand by user.
% The file is called by CITfig.m
% In order to see GUI run initial window.m
%
% Author:      Daniel Czarkowski DLX4
%                             daniel@czarkowski.prv.pl
%
% History:     Final Project 2002
%              Cork Institute of Technology
%              Electrical and Electronics Department%main program

try
   contr;
catch
   contr=1;    % default PID regulator
end
try
   ErrorEstim; % Type of performance index
catch                         % 1-IAE 2-ISE 3-ITAE
   ErrorEstim=1;
end

try
   Kp;
catch
   Kp=1;
   Ki=0;
   Kd=0;
end

try
   N;
catch
   N=1000;
end

try
   PlantS;
catch
   PlantS=tf(1,[180 1]);
end

[PlantNum,PlantDen] = tfdata(PlantS);                              % Get numerator and
denominator
```

```
[PlantNum] = deal(PlantNum{1});                                            % Transfer
from cell to array
[PlantDen] = deal(PlantDen{1});                                            % Transfer
from cell to array



for i=1:length(PlantNum)                    % eliminate zeros from array PlantNum
   if PlantNum(i)==0
      PlantNum(i)=[];
        if PlantNum(i)==0
        PlantNum(i)=[];
          if PlantNum(i)==0
             PlantNum(i)=[];
             if PlantNum(i)==0
                PlantNum(i)=[];
                if PlantNum(i)==0
                PlantNum(i)=[];
                       else break
                       end
                else break
                end
          else break
          end
      else break
      end
        else break
   end
end

for i=1:length(PlantDen)      % eliminate zeros from array PlantDen
   if PlantDen(i)==0
      PlantDen(i)=[];
        if PlantDen(i)==0
        PlantDen(i)=[];
          if PlantDen(i)==0
             PlantDen(i)=[];
             if PlantDen(i)==0
                PlantDen(i)=[];
                if PlantDen(i)==0
                PlantDen(i)=[];
                       else break
                       end
                else break
                end
          else break
          end
      else break
      end
        else break
   end
```

```
end


% Visualisation
HH = findobj(gcf,'Tag','PlantNum');
if length (PlantNum)==1
set(HH,'String',PlantNum(1));
elseif length (PlantNum)==2
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s + ' num2str(PlantNum(2))]));
elseif length (PlantNum)==3
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s^2 + ' num2str(PlantNum(2)), ...
               '*s + ' num2str(PlantNum(3))]));
elseif length (PlantNum)==4
   set(HH,'String',num2str([num2str(PlantNum(1)) '*s^3 + ' num2str(PlantNum(2)), ...
               '*s^2 + ' num2str(PlantNum(3)) '*s + ' num2str(PlantNum(4))]));
elseif length (PlantNum)==4
          set(HH,'String',num2str([num2str(PlantNum(1)) '*s^4 + ' num2str(PlantNum(2)),...
            '*s^3 + ' num2str(PlantNum(3)) '*s^2 + ' num2str(PlantNum(4)) ,...
            '*s + ' num2str(PlantNum(5))]));
elseif length (PlantNum)==4
                 set(HH,'String',num2str([num2str(PlantNum(1)) '*s^5 +
'num2str(PlantNum(2)),...
               '*s^4 + ' num2str(PlantNum(3)) '*s^3 + ' num2str(PlantNum(4)) ,...
               '*s^2 + ' num2str(PlantNum(5)),...
                              '*s + ' num2str(PlantNum(6))]));
else
   warning ('The plant is more than fifth order system, this program cannot accept it');
   set(HH,'String','ERROR!!!');
end



HH = findobj(gcf,'Tag','PlantDen');
if length (PlantDen)==1
set(HH,'String',PlantDen(1));
elseif length (PlantDen)==2
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s + ' num2str(PlantDen(2))]));
elseif length (PlantDen)==3
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^2 + ' num2str(PlantDen(2)),...
         '*s + ' num2str(PlantDen(3))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^3 + ' num2str(PlantDen(2)),...
         '*s^2 + ' num2str(PlantDen(3)) '*s + ' num2str(PlantDen(4))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^4 + ' num2str(PlantDen(2)),...
         '*s^3 + ' num2str(PlantDen(3)) '*s^2 + ' num2str(PlantDen(4)),...
         '*s + ' num2str(PlantDen(5))]));
elseif length (PlantDen)==4
   set(HH,'String',num2str([num2str(PlantDen(1)) '*s^5 + 'num2str(PlantDen(2)),...
         '*s^4 + ' num2str(PlantDen(3)) '*s^3 + ' num2str(PlantDen(4)),...
         '*s^2 + ' num2str(PlantDen(5)) '*s + ' num2str(PlantDen(6))]));
```

```
else
   warning ('The plant is more than fifth order system, this program cannot accept it');
   set(HH,'String','ERROR!!!');
end

% Visualisation of Kp Ki Kd in close loop system (file CITfig.m)
try
HH = findobj(gcf,'Tag','Kp');
set(HH,'String',num2str(['Kp=' num2str(Kp)]));
catch
HH = findobj(gcf,'Tag','Kp');
set(HH,'String','Kp=');
end

try
HH = findobj(gcf,'Tag','Ki');
set(HH,'String',num2str(['Ki=' num2str(Ki)]));
catch
HH = findobj(gcf,'Tag','Ki');
set(HH,'String','Ki=');
end

try
HH = findobj(gcf,'Tag','Kd');
set(HH,'String',num2str(['Kd=' num2str(Kd)]));
catch
HH = findobj(gcf,'Tag','Kd');
set(HH,'String','Kd=');
end
```

InitialWindowHelp.m

```
function InitialWindowHelp

% InitialWindowHelp.m
%
% The purpose of this function is to display a help
% window containing information on the function of each
% option available in the menu
%
% Author:      Daniel Czarkowski DLX4
%              daniel@czarkowski.prv.pl
%
% History:     Final Project 2000
%              Cork Institute of Technology
%              Electrical and Electronics Department


bodyTxt = [    ' The program displays 4 main zones :                                   '
' (1) Identification                                                                    '
' (2) Tuning Rule                                                                       '
' (3) Graphs                                                                            '
' (4) Close loop system and results of identification and tuning                        '
'                                                                                       '
' In addition, we have buttons such as "Grid on" "Hold on" "Clear" etc.      '
' These buttons are useful during comparison tuning rules or identifications            '
'                                                                                       '
' The first of these options "Load data" has a popmenu which consist of two options:    '
' a) load datafile - loads default datafile.m which has a real data obtain from a tank  '
' b) Real time identification - this option is available only in workstation which has  '
' Data Acquisition Unit. When we choose this option we can obtain data from a tank in   '
' real time. The program tries finding solution for 15 minutes. It is possible that     '
' stops measuring when settling value is constant for 2 minutes                         '
' Next we can choose what kind of identification method we want to use. All of them are '
' implemented for first order system. We can also enter system in fields Zeros & Poles  '
'                                                                                       '
' The second option - "tuning methods" we can decide what kind tuning method we want   '
' to use. If we want to change parameters such as population, ranges, time of counting  '
' etc we have to change it in file genhaploid.m Next option                             '
' (NCD - Nonlinear Control System Design) is available only in Matlab with optimization '
' toolbox and NCD toolbox. This option displays a window with close loop system.
'
' Now we can initiate values entered in the main window (GUI). Next step opens NCD block  '
' set bounds and start optimization. After optimization go to GUI, click refresh button '
' and our PID parameters will be obtained                                               '
'                                                                                       '
' Next two options (3,4) display results                                       '

 ];
```

```
h = figure('Units','points', ...
       'Color',[0.8 0.8 0.8], ...
   'FileName','C:\Edukacja\Matlab\work\project\InitialWindowHelp.m', ...
   'Name','Cork Institute of Technology, project by Daniel Czarkowski v1.0', ...
   'NumberTitle','off', ...
       'PaperPosition',[18 180 570 420], ...
       'PaperUnits','points', ...
       'Position',[173.25 71.25 447 436.5], ...
       'Tag','Fig1', ...
       'ToolBar','none');

h = uicontrol(...
   'Style','text', ...
   'String',bodyTxt,...
   'FontSize',10,...
   'ForegroundColor',[0 0 0],...
   'HorizontalAlignment','left',...
   'BackgroundColor',[0.8706 0.8588 0.8549],...
   'Position',[10 10 570 500]);                                    % Output body
text into Figure


h = uicontrol(...
   'Style','text', ...
   'String','Help',...
   'FontSize',24,...
   'ForegroundColor',[1 0 0],...
   'BackgroundColor',[0.8706 0.8588 0.8549],...
   'Position',[175 520 200 40]);                                   % Output
Title
```

InitialWindow.m

```
function fig = InitialWindow()

% InitialWindow.m
%
% This is main script which initialize Graphic User
% Interface. To edit this file run this script and next
% type guide in command window
%
% Author:      Daniel Czarkowski DLX4
%                              daniel@czarkowski.prv.pl
%
% History:     Final Project 2000
%              Cork Institute of Technology
%              Electrical and Electronics Department
load InitialWindow

h0 = figure('Units','points', ...
        'BackingStore','off', ...
        'Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'DoubleBuffer','on', ...
        'FileName','C:\Edukacja\Matlab\work\project\InitialWindow.m', ...
        'Name','Cork Institute of Technology, project by Daniel Czarkowski v1.0', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[79.5 97.5 580.5 408], ...
        'Resize','off', ...
        'ResizeFcn','legend(''ResizeLegend'')', ...
        'Tag','Fig1', ...
        'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','close(gcf)', ...
        'ListboxTop',0, ...
        'Position',[598 21.749063670412 160 44.32], ...
        'String','Close', ...
        'Tag','Close');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','InitialWindowHelp', ...
        'ListboxTop',0, ...
        'Position',[598 70.35955056179785 160 44.32], ...
        'String','Info', ...
        'Tag','Info');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat1, ...
        'ListboxTop',0, ...
        'Position',mat2, ...
        'String','Clear', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[66.75 343.5 132 34.5], ...
        'Style','frame', ...
        'Tag','RegFrame3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 366 78 9], ...
        'String','Kp=', ...
        'Style','text', ...
        'Tag','Kp');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 344.25 78 9], ...
        'String','Kd=', ...
        'Style','text', ...
        'Tag','Kd');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[120 355 78 9], ...
        'String','Ki=', ...
        'Style','text', ...
        'Tag','Ki');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[78 366 40 9], ...
        'String','Regulator', ...
        'Style','text', ...
        'Tag','RegStaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[231.75 343.5 132 34.5], ...
        'Style','frame', ...
        'Tag','PlantFrame3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[252.75 352.5 108.75 6.75], ...
        'String','----------------------------------', ...
        'Style','text', ...
        'Tag','PlantFractionText');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[252.75 356.25 108.75 9], ...
        'String','1', ...
        'Style','text', ...
        'Tag','PlantNum');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[252.75 344.25 108.75 9], ...
        'String','180*s + 1', ...
        'Style','text', ...
        'Tag','PlantDen');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[236.25 348.75 17.25 10.5], ...
        'String','K(s)=', ...
        'Style','text', ...
        'Tag','PlantStaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.8 0.8 0.8], ...
        'Enable','inactive', ...
        'ListboxTop',0, ...
        'Position',[232.5 366.75 129.75 9], ...
        'String','Current Plant', ...
        'Style','text', ...
        'Tag','PlantStaticText2');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[599 193 160 150], ...
        'Style','frame', ...
        'Tag','Frame Search', ...
        'UserData','[ ]');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[599 368 160 150], ...
        'Style','frame', ...
        'Tag','Frame Identification', ...
        'UserData','[ ]');
h1 = axes('Parent',h0, ...
        'Units','points', ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[0.501960784313725 0.501960784313725 0.501960784313725], ...
        'ColorOrder',mat3, ...
        'Position',[24 324.75 390 60], ...
        'Tag','Close loop', ...
        'XColor',[0 0 0], ...
        'XLim',[0 0.9], ...
        'XLimMode','manual', ...
        'XTickMode','manual', ...
        'YColor',[0 0 0], ...
        'YLimMode','manual', ...
        'YTickMode','manual', ...
        'ZColor',[0 0 0]);
h2 = line('Parent',h1, ...
        'Color',[0 0 0], ...
        'Interruptible','off', ...
        'Tag','ClosedLine', ...
        'XData',mat4, ...
        'YData',mat5);
h2 = line('Parent',h1, ...
        'Color',[0 0 1], ...
        'Interruptible','off', ...
        'Marker','o', ...
        'MarkerEdgeColor',[0 0 0], ...
        'MarkerFaceColor',[0 0 0], ...
        'MarkerSize',5, ...
        'Tag','ConfigurationAxesLine1', ...
        'XData',0.06, ...
        'YData',0.65);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'FontSize',16, ...
        'FontWeight','bold', ...
        'Interruptible','off', ...
        'Position',[0.04 0.54 0], ...
        'String','-', ...
        'Tag','SignText', ...
        'UserData',-1);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',mat6, ...
        'Tag','ConfigurationAxesText4', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',mat7, ...
        'Rotation',90, ...
        'Tag','ConfigurationAxesText3', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Interruptible','off', ...
        'Position',[-0.05722543352601155 1.379746835443038 17.32050807568877], ...
        'Tag','ConfigurationAxesText2', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Interruptible','off', ...
        'Position',[0.4491329479768785 1.088607594936709 17.32050807568877], ...
        'Tag','ConfigurationAxesText1', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = line('Parent',h1, ...
        'Color',[0 0 0], ...
        'Interruptible','off', ...
        'Tag','ConfigurationAxesLine2', ...
        'XData',mat8, ...
        'YData',mat9);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[600 397 50 20], ...
        'String','Zeros', ...
        'Style','text', ...
        'Tag','zero');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat10, ...
        'ListboxTop',0, ...
        'Position',[655 397 90 20], ...
        'String','1', ...
        'Style','edit', ...
        'Tag','EditPlantNum');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat11, ...
        'ListboxTop',0, ...
        'Position',[655 377 90 20], ...
        'String','[180 1]', ...
        'Style','edit', ...
        'Tag','EditPlantDen');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[600 377 50 20], ...
        'String','Poles', ...
        'Style','text', ...
        'Tag','poles');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat12, ...
        'ListboxTop',0, ...
        'Position',[640 219 75 48], ...
        'String',mat13, ...
        'Style','popupmenu', ...
        'Tag','ErrorEstim', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[615 267 124 19], ...
        'String','Type of criterion', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[614 314 134 19], ...
        'String','Tuning method', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat14, ...
        'ListboxTop',0, ...
        'Position',[640 176 75 48], ...
        'String',mat15, ...
        'Style','popupmenu', ...
        'Tag','contr', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[615 224 124 19], ...
        'String','Type of controller', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat16, ...
        'ListboxTop',0, ...
        'Position',[617 393 114 50], ...
        'String',mat17, ...
        'Style','popupmenu', ...
        'Tag','IdentMethods', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'ListboxTop',0, ...
        'Position',[603 445 142 19], ...
        'String','Identification methods', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'HorizontalAlignment','right', ...
        'ListboxTop',0, ...
        'Position',[621 492 102 19], ...
        'String','Load data', ...
        'Style','text', ...
        'Tag','LoadData');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat18, ...
        'ListboxTop',0, ...
        'Position',[617 442 114 50], ...
        'String',mat19, ...
        'Style','popupmenu', ...
        'Tag','LoadData', ...
        'Value',2);
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','main;', ...
        'ListboxTop',0, ...
        'Position',mat20, ...
        'String','Refresh', ...
        'Tag','Refresh');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback','rtstop1', ...
        'ListboxTop',0, ...
        'Position',[617 492 45 22], ...
        'String','Stop', ...
        'Tag','Pushbutton2');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat21, ...
        'ListboxTop',0, ...
        'Position',[598 149.1856375183194 73.33333333333333 26.66666666666666], ...
        'String','Hold Off', ...
        'Style','togglebutton', ...
        'Tag','Hold');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[0.831372549019608 0.815686274509804 0.784313725490196], ...
        'Callback',mat22, ...
        'ListboxTop',0, ...
        'Position',[685 148.5189708516528 73 27], ...
        'String','Grid Off', ...
        'Style','togglebutton', ...
        'Tag','Grid');
h1 = uicontrol('Parent',h0, ...
        'BackgroundColor',[1 1 1], ...
        'Callback',mat23, ...
        'ListboxTop',0, ...
        'Position',[622 266 111 48], ...
        'String',mat24, ...
        'Style','popupmenu', ...
        'Tag','SearchMethods', ...
        'Value',1);
h1 = axes('Parent',h0, ...
        'Box','on', ...
        'CameraUpVector',[0 1 0], ...
        'CameraUpVectorMode','manual', ...
        'Color',[1 1 1], ...
        'ColorOrder',mat25, ...
        'Position',[0.03746770025839794 0.05882352941176471 0.6782945736434108
0.6856617647058824], ...
        'Tag','Axes1', ...
        'XColor',[0 0 0], ...
        'YColor',[0 0 0], ...
        'ZColor',[0 0 0]);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.4980916030534351 1.018817204301075 9.160254037844386], ...
        'Tag','Axes1Text4', ...
        'VerticalAlignment','bottom');
set(get(h2,'Parent'),'Title',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[0.4980916030534351 -0.06451612903225823 9.160254037844386], ...
        'Tag','Axes1Text3', ...
        'VerticalAlignment','cap');
set(get(h2,'Parent'),'XLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','center', ...
        'Position',[-0.05534351145038167 0.4973118279569891 9.160254037844386], ...
        'Rotation',90, ...
        'Tag','Axes1Text2', ...
        'VerticalAlignment','baseline');
set(get(h2,'Parent'),'YLabel',h2);
h2 = text('Parent',h1, ...
        'Color',[0 0 0], ...
        'HandleVisibility','off', ...
        'HorizontalAlignment','right', ...
        'Position',[-0.05725190839694656 1.370967741935484 9.160254037844386], ...
        'Tag','Axes1Text1', ...
        'Visible','off');
set(get(h2,'Parent'),'ZLabel',h2);
if nargout > 0, fig = h0; end
```

Appendix B - Callbacks

SearchMethods

```
Value=get(gcbo,'Value');
SearchMethods=Value;
if SearchMethods==1
   genhaploid;
else
ncdinit;
end
clear SearchMethods
```

Grid

```
Value=get(gcbo,'Value');
HH=findobj(gcf,'Tag','Grid');
if Value==1
gca;
grid on;
set(HH,'String',num2str('Grid On'));
else
gca;
grid off;
set(HH,'String',num2str('Grid Off'));
end;
```

Hold

```
Value=get(gcbo,'Value');
if Value==1
gca;
hold on;
HH=findobj(gcf,'Tag','Hold');
set(HH,'String',num2str('Hold On'));
else
gca;
hold off;
set(HH,'String',num2str('Hold Off'));
end;
```

Pushbutton2

```
rtstop1
```

Refresh

```
main;
```

LoadData

```
Value=get(gcbo,'Value');
LoadData=Value;
if LoadData==1
   ident_rt;
elseif  LoadData==2
try
   load datafile;
catch
disp ('cannot find the datafile.mat! please change directory where is InitialWindow.m')
end
end
try
 plot (x,Vout,'r',x,ypoly,'g');
title('Measured (red), Polynomial curve fitting (blue)');
catch
disp ('datafile is incorrect, please change directory where is InitialWindow.m');
end
clear loadData
```

IdentMethods

```
Value=get(gcbo,'Value');
IdentMethods=Value;
ident_calc;
```

Contr

```
Value=get(gcbo,'Value');
contr=Value;
```

ErrorEstim

```
Value=get(gcbo,'Value');
ErrorEstim=Value;
```

EditPlantDen & EditPlantNum

```
HH = findobj(gcf,'Tag','EditPlantDen');
EditPlantDen = str2num(get(HH,'String'));

HH = findobj(gcf,'Tag','EditPlantNum');
EditPlantNum = str2num(get(HH,'String'));

PlantS=tf(EditPlantNum,EditPlantDen);
```

```
gcf;
try
N;
Catch
N=1000;
End
sim('pid_fine_ol');
plot(yout,'k');
title('Transfer function entered by user (black)');
main;
```

Clear

```
cla;
title('');
warning off
legend('')
warning on
Kp=('');
Ki=('');
Kd=('');
main;
```

Info

```
InitialWindowHelp
```

Close

```
close(gcf)
```

Appendix C - Simulink models

ncdCIT.mdl

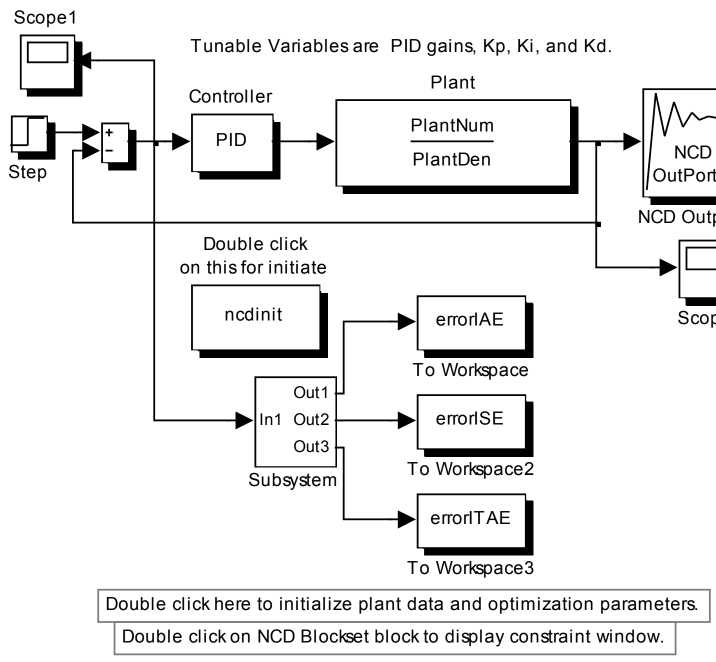


figure C1 Model in Simulink, file ncdCIT.mdl

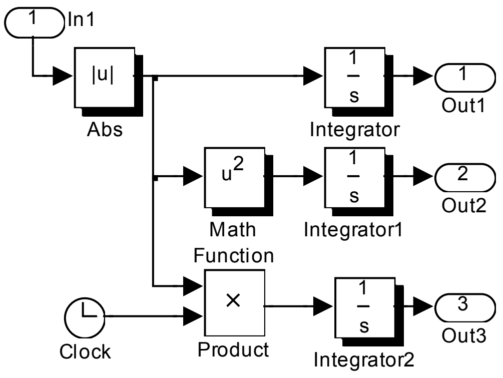


figure C2. Subsystem of ncdCIT.mdl