INTELLIGENT CONTROL

ASSIGNMENT - 2

2018-2019 Spring

Lecturer : Assoc. Prof. Gulay Oke Gunel

Name : Furkan Ayık

Student ID : 040140337

1) Neuron Based Nonlinear PID Controller

PID controllers have been widely used in industrial applications mostly for linearized systems. Classical PID controllers are bad at nonlinearities of system model or model uncertainties. One of many approaches to overcome those issues is using “Neuron based nonlinear pid controller” which is offered by Wang and Yu.

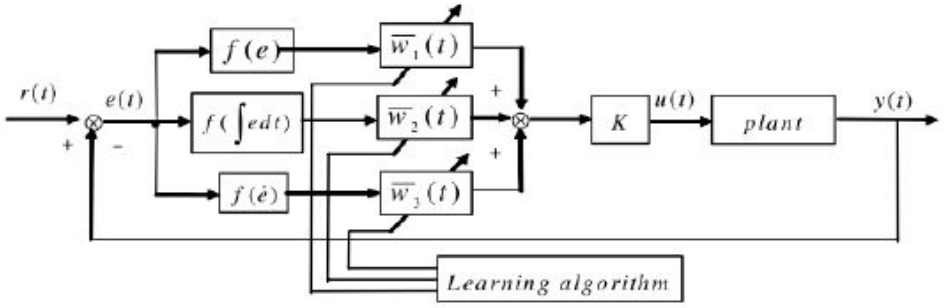


Fig 1. The neuron based nonlinear PID control system

In this control method, weights of each neuron is updated for one loop. That means it is an ‘on-line’ algorithm.

Thus control signal can be expressed as below,

$$u(t) = f(e) * w_1 + f(\int e) * w_2 + f(e')$$

Where,

$$f(x,\alpha,\delta) = \begin{cases} sign(x)\cdot|x|^{\alpha} & when\ |x| > \delta, \\ \delta^{\alpha-1}x & when\ |x| \le \delta, \end{cases}$$

This nonlinear function is coded in ‘nonl_f.m’ . The parameter α is between (0,1). When $\alpha = 1$ makes the nonlinear function linear. δ is a small positive number applied to create a small linear area in this nonlinear function when x is around zero. For each of neuron has it is own alpha and δ parameters.

1.1 Step Reference Tracking Performance

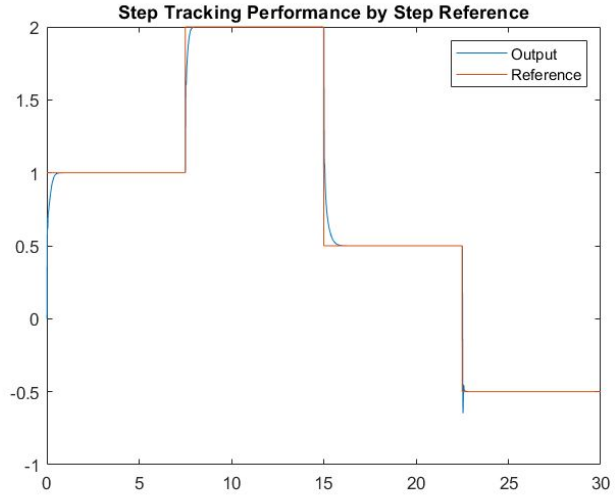


Fig 1.1.1 Unit Step Tracking

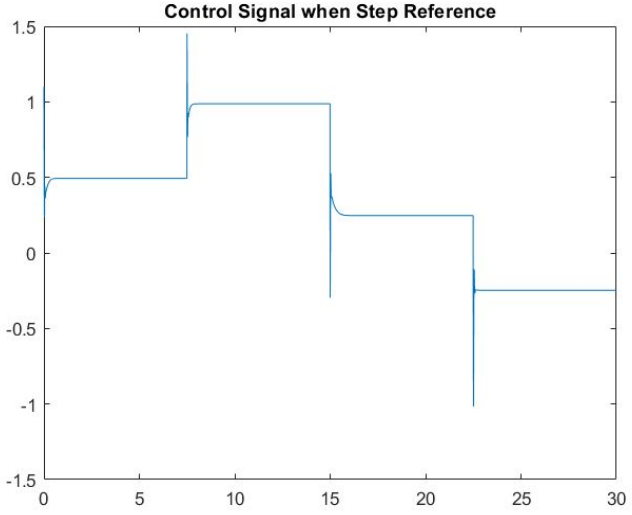


Fig 1.1.2 Control Signal for Unit Step Reference

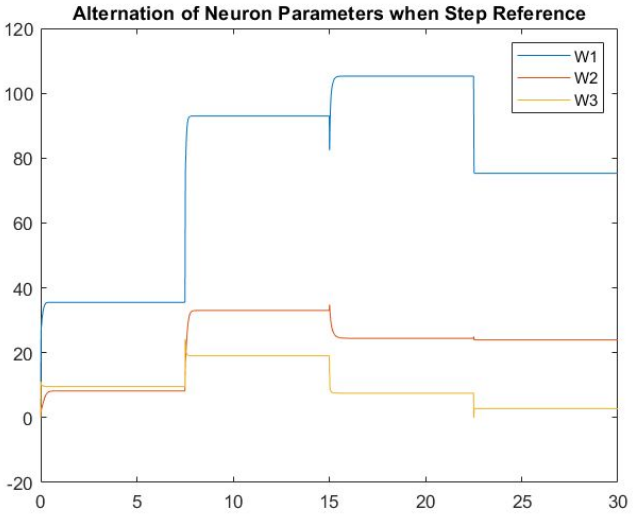


Fig 1.1.3 Parameter Alternation for Unit Step Reference

For unit step reference, controller has a remarkable performance.By the fig 1.1.2 change of control signal, it could be concluded that it reaches steady state slightly smoother for positive number references. But for references less than 0 such as -0.5, it reaches steady state with much more overshoot, that would make control signal worse and may harm the system.

From the fig 1.1.3, we can easily see the effect of α parameter, which is equal to 1 for w_3 parameter. It is a linear curve without exponential decays. This is suitable for feedforward gain. Because it directly empowering the error signal.

1.2 Exponential Reference Tracking Performance

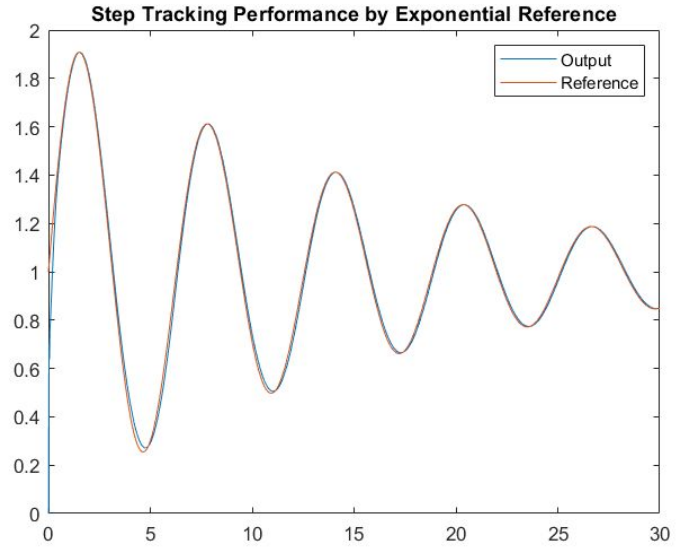


Fig 1.2.1 Exponential Reference Tracking

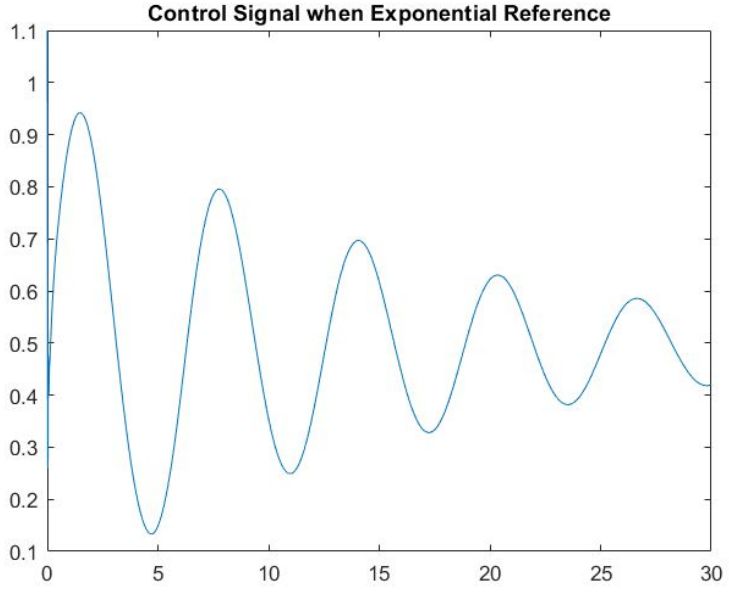


Fig 1.2.2 Control Signal for Exponential Reference

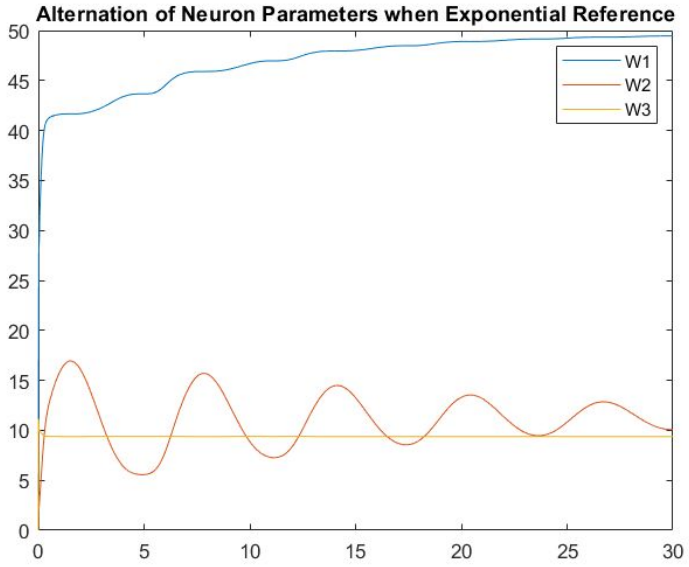


Fig 1.2.3 Parameter Alternation for Exponential Reference

For exponential reference, controller has perfect performance. No steady state error occurred and settling time is nearly zero.

Final controller parameters are shown in table.1 .

	w1	w2	w3
Step Reference	75.2915	23.9515	2.7661
Exponential Reference	49.4569	10.0830	9.3797

2) PID Type Fuzzy Controller

For this question, I created a simulink model of fuzzy PID. For fuzzy PID controller is implemented in a matlab function inside of simulink. After model is created the cost function,

$$J = \frac{1}{2}\sum_{n=1}^{\infty}[r_n - y_n]^2 + \frac{1}{2}\lambda\sum_{n=1}^{\infty}[u_n - u_{n-1}]^2$$

4 unknown parameters K_e K_{de} α β are optimized via 'optimtool' toolbox's multivariable genetic algorithm.

One of the challenges was creating the reference signal in simulink. To overcome this issue, reference signal is defined sum of 4 step function.

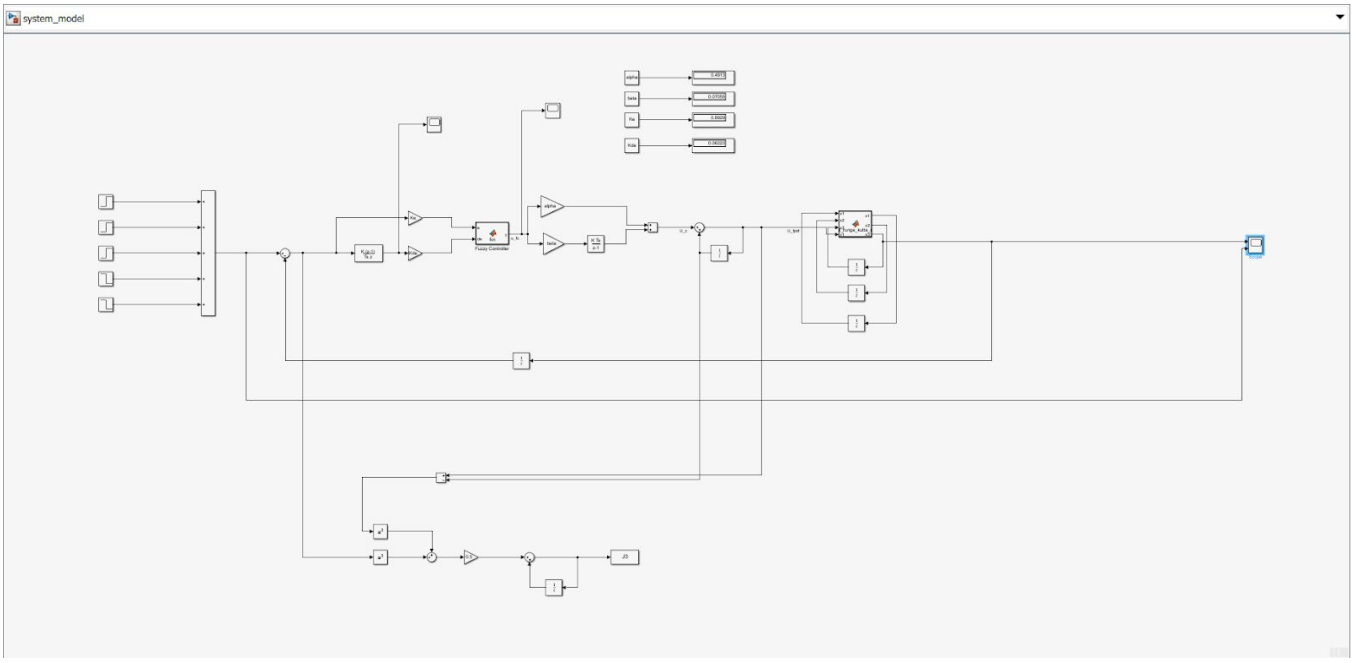


Fig.2.1 Simulink Model of Fuzzy PID

Runga-kutta model of CSTR model is created as below,

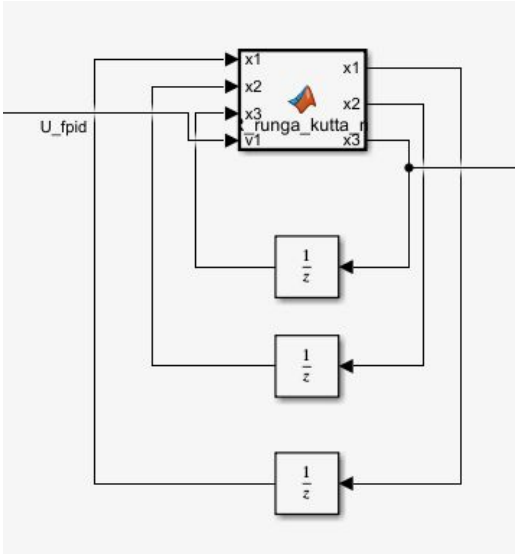


Fig 2.2 CSTR Model in Simulink

All the system can be viewed inside 'Question_2' folder.

2.1 Fuzzy PID Algorithm

1. Calculation of the error : $e_n = r_n - y_n$
2. Multiplication of the error and derivative of the error with K_e *and* K_{de} parameters respectively
3. Calculation of linguistic values : $A_i(e_n),\ A_{i+1}(e_n),\ B_j(e'_n), B_{j+1}(e'_n)$
4. Determination of the control signal U_{FC_n} by the fuzzy rules
5. Calculation of U_{c_n} ,

$$U_C(n) = \alpha * U_{FC}(n) + \beta * (U_{FC}(n) + U_{FC}(n-1))$$

6. Calculation of U_{FPID}

$$U_{FPID}(n+1) = U_{FC}(n) + U_c(n)$$

7. Applying control signal U_{FPID} to the CSTR system
8. Obtaining the output $y(n+1)$

Above algorithm does not include the genetic algorithm part.

2.2 Fuzzy PID Controller Tuning with Genetic Algorithm

In the simulink file the cost function is modeled as in fig 2.2.2

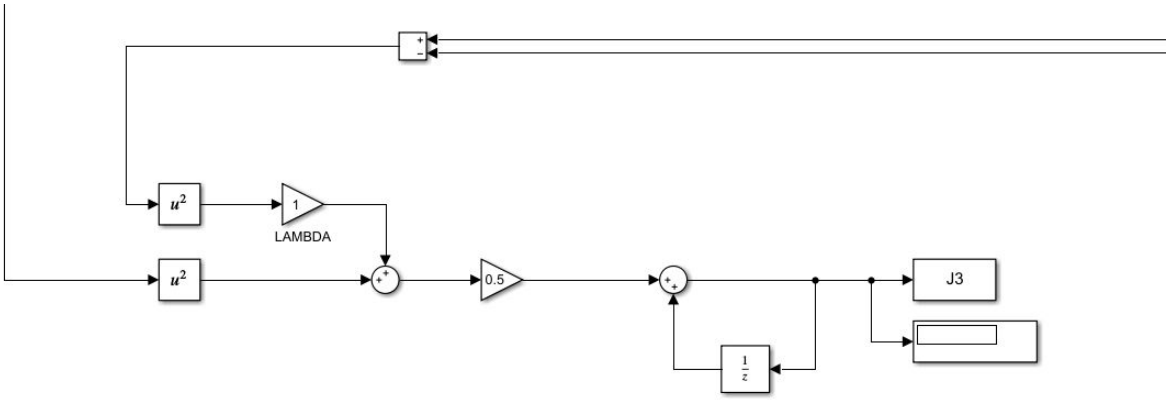


Fig 2.2.2 Cost Function in SIMULINK model

In every step cost function is obtained simply by "sim('system_model')" command in the "cost.m' matlab function file. This file is given into the genetic algorithm toolbox as a parameter. 4 different parameter is defined for fuzzy pid controller is tuned by every genetic algorithm training step. Control system and system output with reference is saved to workspace to plot later with 'plot_all.m' file. General structure of optimtool toolbox for genetic algorithm is given in fig 2.2.3.

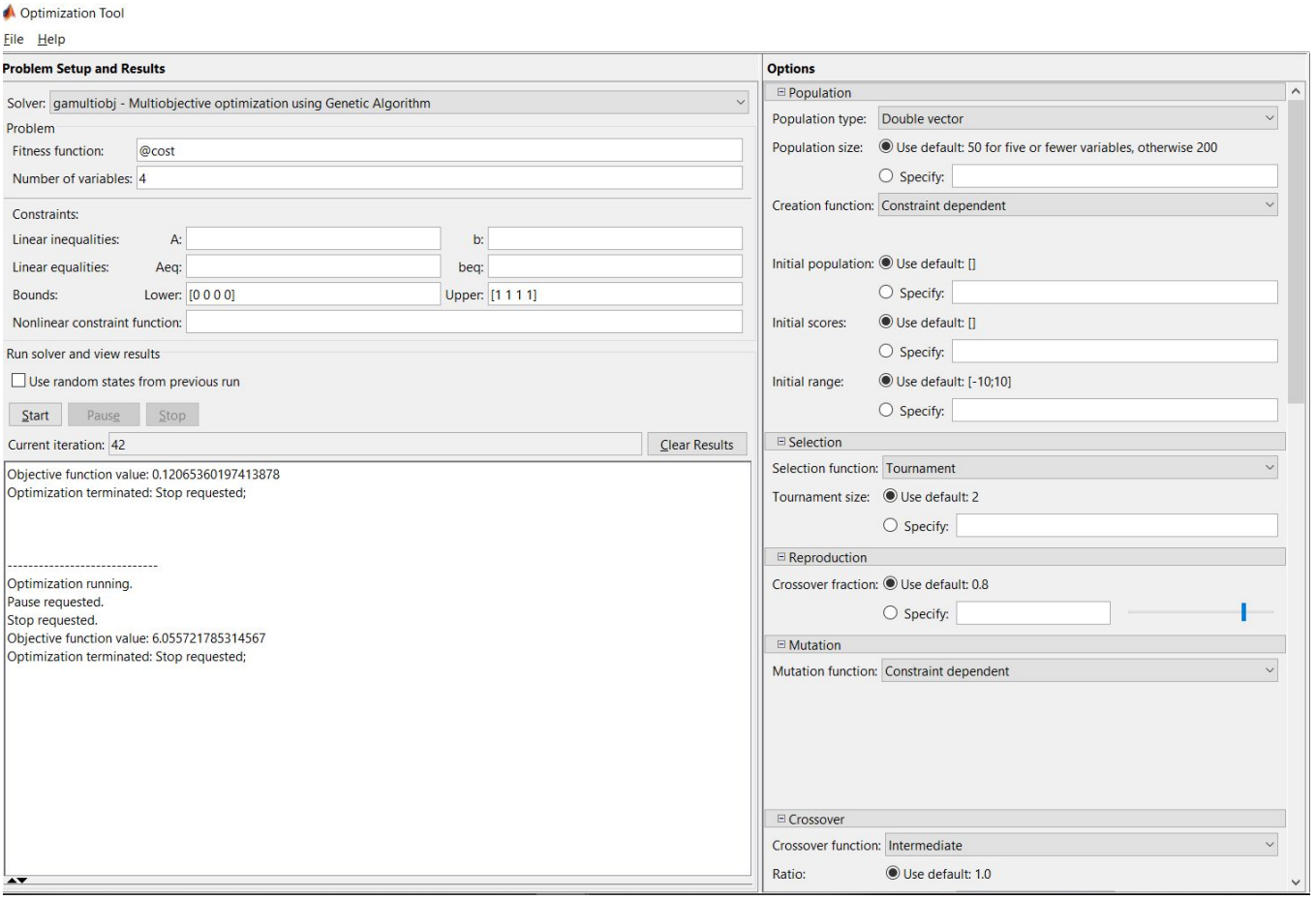
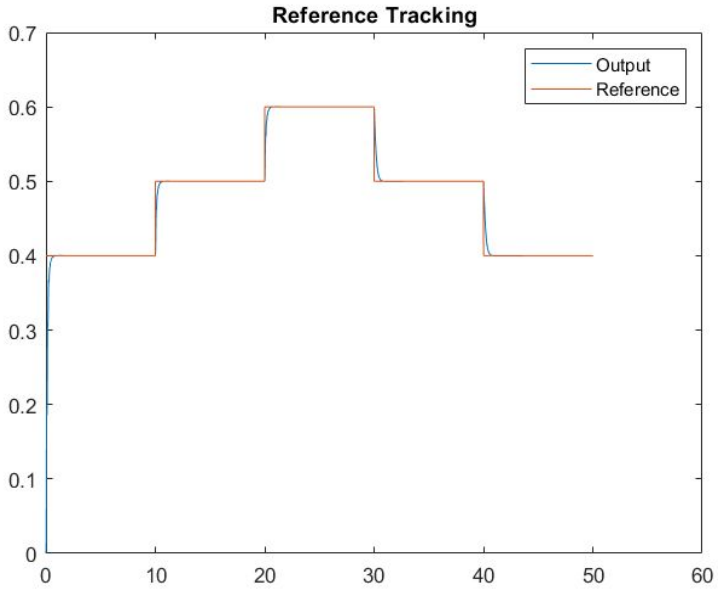
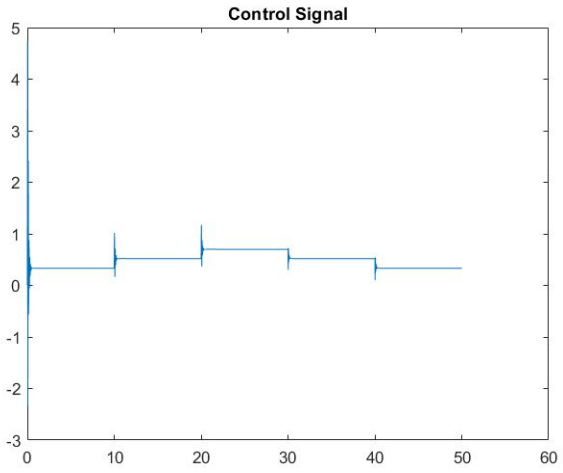


Fig 2.2.3 MATLAB Genetic Algorithm Toolbox Overview,

For $\lambda = 0.1$ **,**

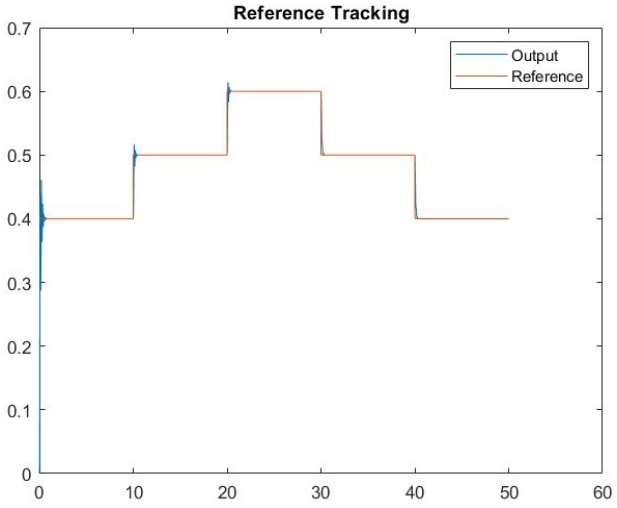


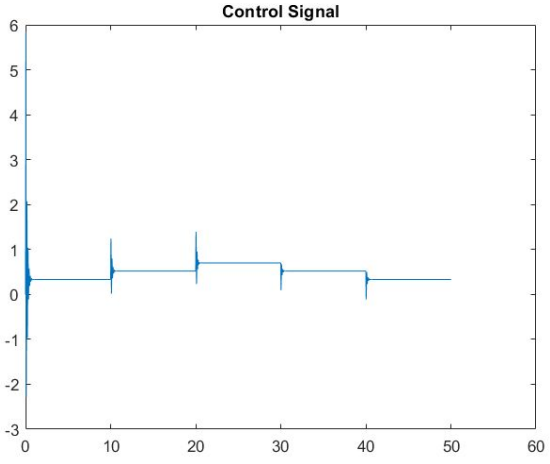


α	0.7910
β	0.7523
K_e	0.9409

K_{de}	0.1269
Cost	7.0176

For $\lambda = 0.5$ **,**

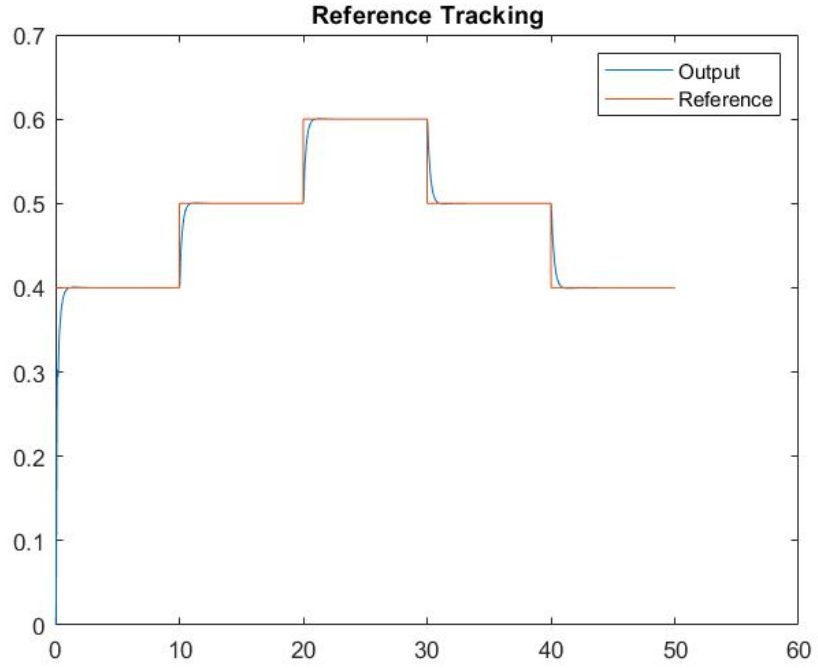


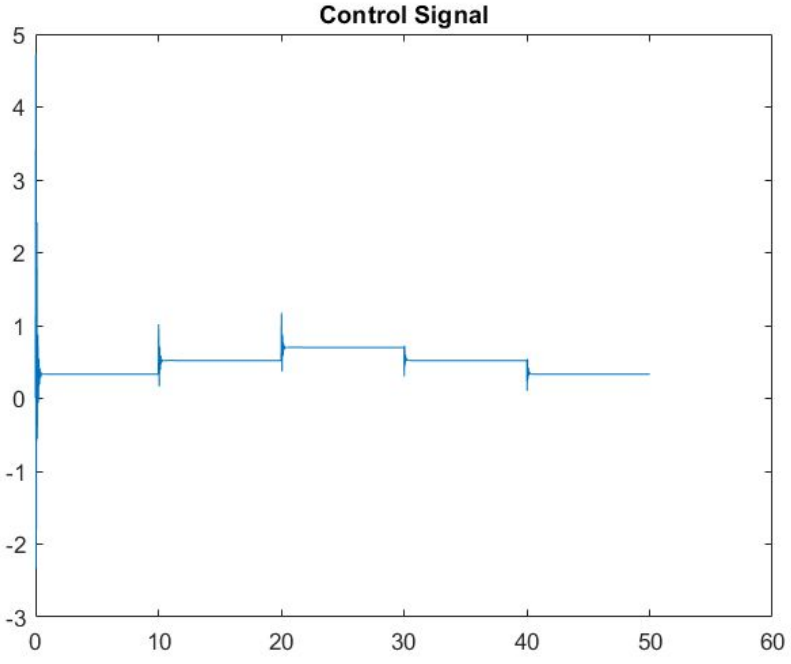


α	0.7351
β	0.0933
K_e	0.9405

K_{de}	0.0535
Cost	5.6507

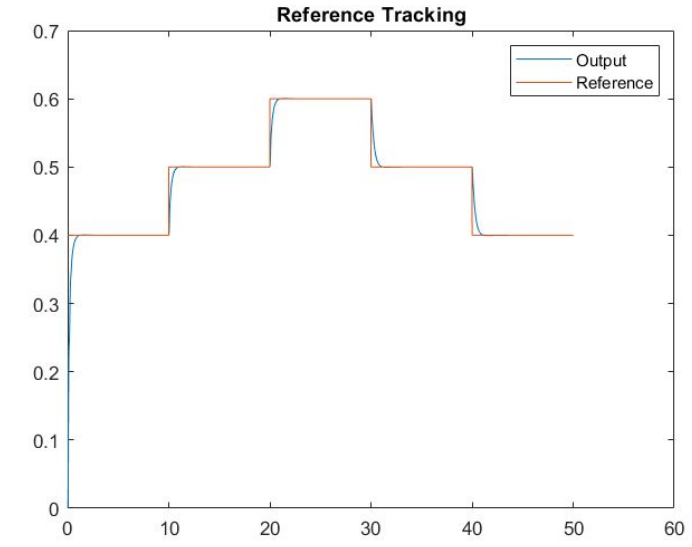
For $\lambda = 1$ **,**

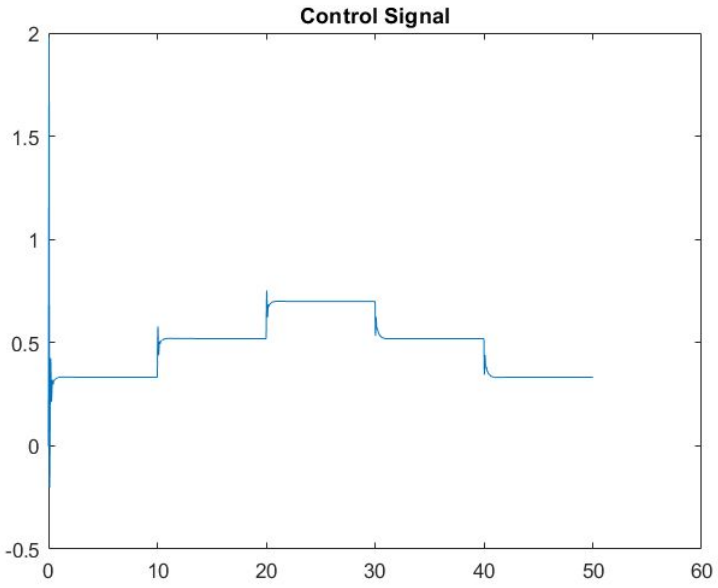




α	0.6264
β	0.9855
K_e	0.2421
K_{de}	0.0519
Cost	9.9732

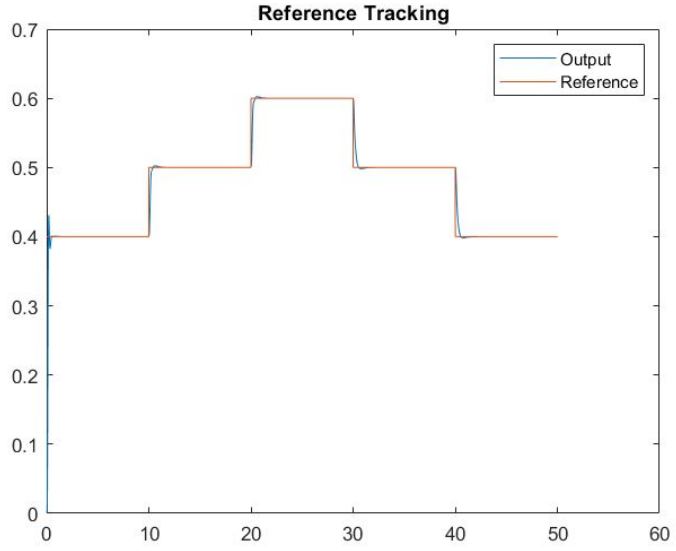
For λ = 10 **,**

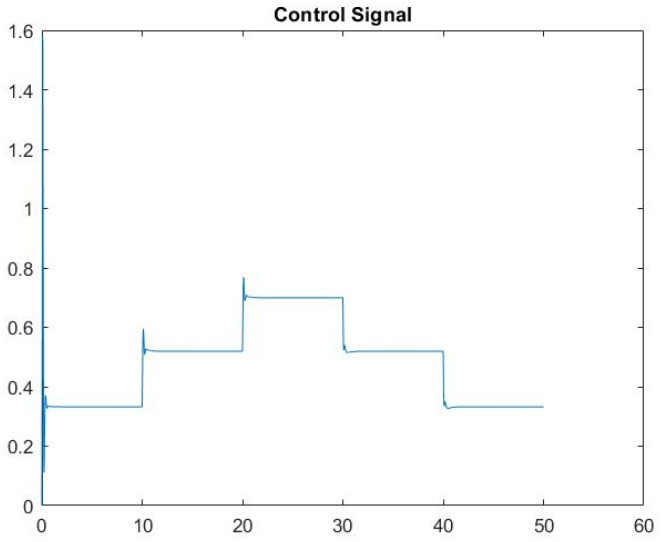




α	0.3023
β	0.5207
K_e	0.5508
K_{de}	0.1264
Cost	11.7173

For λ = 100 **,**





α	0.1151
β	0.3562
K_e	0.6763
K_{de}	0.0981
Cost	12.2835

Results

Through minimizing the cost function with genetic algorithm I achieved successful results. That was an expecting result because to minimize cost function we should have minimized two things tracking error and control effort. I could achieve better results by running the genetic algorithm longer since those results sufficiently good I decided to stop the algorithm.

I should also state that increasing λ increases the cost function and making finding right parameters harder. Because I already achieved good results by not punishing control effort that much. So that local minimas should have between minimal differences from our parameters. That is an expected phenomenon when optimizing with genetic algorithm. That algorithm totally randomly generating populations, which makes it stochastic process and finding sensitive local minimas hard sometimes.

All codes and files included in the rar file .