Optimal Control of a Linear Discrete System

Summary

In the real world, many real systems are time-discrete in nature.Even if the system is time-continuous, because the computer is based on such discrete digital technology as time and value, the implementation of computer control must be treated as a discrete system after time discrimination.Therefore, it is necessary to discuss the optimal control of discrete time systems.

To build up the optimal problem, first, we define the objective function in chapter-3 which stands for summation of the magnitude square of controls over the entire time horizon.And we want to constrain the x(T) to fall in the predetermined interval, so a penalty function is further explained in chapter-4. If the L increases which means the final state x(T) will have a greater impact on the objective function, therefore x(T) would have a much stricter constrains and this would cause the decrease of the final state value. Vice versa, if the L reduces, the value of x(T) would be larger Therefore, by changing the value of coefficient L we are able to control the X(T) to fall in the interval that we wanted.

We have used lots of methods to solve the problem in the matlab. But finally, only interior point method can match our problem. We write our own function and use interior point method to optimize it. For the problems that need to be solved, we adjusted the parameters and measured multiple sets of data. We reached a detailed conclusion, which will be introduced in detail in our article.

For the probelm 3,4,5, we change different parameters can plot the u and cost function, the change is complex but can get some rule behind it, which can be seen at Figure7,9,10. The answers are at chapter 5,6,7.

Keywords: DLQR; Interior point method; Dynamic programming; MATLAB;

Author: Dazheng Fang , Haorui Li, Yao Yao, Zhiyi Shi (Alphabetical order)

Contents

1 Introduction

1.1 Background

In the real world, many real systems are time-discrete in nature.Even if the system is time-continuous, because the computer is based on such discrete digital technology as time and value, the implementation of computer control must be treated as a discrete system after time discretization.[8]

Therefore, it is necessary to discuss the optimal control of discrete time systems.

1.2 Problem Restatement

Suppose that we have a discrete-time linear system with the following dynamic equation:

$$x(k+1) = Ax(k) + Bu(k)$$

where x(t) is the state variable at time t; u(t) is control variable at time t; A is the state matrix; and B is the control matrix. Suppose that the initial state x(0) is known. Demine the control series u(t), t=0, 1, ..., T, such that the state at the end of the time horizon x(T) falls in a predetermined interval and the energy consumed in the entire time horizon (which can be captured by the weighted summation of the magnitude square of controls over time) can be minimized. The values of the controls have upper bounds.

And the six problems are:

(1) Develop the mathematical programming model of the optimal control problem.

(2) Set up the scale of the problem (i.e., the dimension of the state vector and the control vector) by yourself, as well as the values of the matrices A and B, the initial state and the upper bounds of the controls. Solve the problem by using the tools in Matlab and obtain the optimal control policy.

(3) Change the value of A and analyze its impact on the controls and the objective function.

(4) Change the upper bounds of the controls and analyze their impact on the controls and the objective function.

(5) Change the length of the time horizon (i.e., the value of T) and analyze its impact on the controls and the objective function.

(6) Write a research report, which must include the description of the problem, definition of the notations, explanation of the data, mathematical formulation of the problem, method of solving the problem, analysis of the result, conclusion, and references. The source code must be submitted.

1.3 Our Work

First of all, we define the objective function and add a penalty function to build up the optimal problem. Then we tried DLQR, simulink, dynamic programming and interior point method to solve the problem. Finally we change the parameters like A, B, and T to track their influences on results.

2 List of Notation

Table 1: The List of Notation

Symbol	Meaning
J	System performance index, optimization function
$\bar{J}$	Cost function
L_k	The increment of performance indicator J in the Kth sampling period
λ	Lagrangian multiplier matrix
L	Penalty weight for x

3 Problem One: Build up the Model

From the problem we define the system performance index, in other words, optimization function J as:

$$J = \frac{1}{2}\sum_{k=0}^{T-1} u^T(k)u(k) + Lx(T)^T x(T)$$

We define the objective function as above. The first term stands for summation of the magnitude square of controls over the entire time horizon which also means the energy consumed by the whole system. However, if we want to constrain the x(T) to fall in the predetermined interval, a penalty function should be defined like the second term. If the L increases which means the final state x(T) will have a greater impact on the objective function, therefore x(T) would have a much stricter constrains and this would cause the decrease of the final state value. Vice versa, if the L reduces, the value of x(T) would be larger. Therefore, by changing the value of coefficient L we are able to control the X(T) to fall in the interval that we wanted.
$Notes$: Now the weight of every step is 1

The problem can be rewriten as:

$$min\ J = \frac{1}{2}\sum_{k=0}^{T-1} u^T(k)u(k) + Lx^T(T)x(T) \tag{1}$$
$$S.t.\ x(k+1) = Ax(k) + Bu(k) \tag{2}$$
$$k < T \tag{3}$$
$$k \in Z^+ \tag{4}$$
$$x(0) = x_0 \tag{5}$$
$$u(k) \leq U \tag{6}$$

4 Problem Two: Solve the problem by using thetools in Matlab

We search different MATLAB tools and finally tried the following methods:

- QLDR Function
- Interior Point Method
- Dynamic Programming
- Euler Equation

Finally the QLDR and interior point method get the best results, due to the time limit, we give up DP and Euler Equation.

4.1 DLQR Function

The linear quadratic regulator (LQR) is a well-known design technique that provides practical feedback gains.[10][1]

In the case of the Linear Quadratic Regulator (with zero terminal cost), we can get:

$$L = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u$$

where the requirement that L > 0 implies that both Q and R are positive definite. In the case of linear plant dynamics also, we have

$$L_x = x^T Q$$
$$L_u = u^T R$$
$$f_x = A$$
$$f_u = B$$

So we have:

$$\dot{x} = Ax + Bu$$
$$x(t_o) = x_o$$
$$\dot{\lambda} = -Qx - A^T\lambda$$
$$\lambda(t_f) = 0$$
$$Ru + B^T\lambda = 0$$

Since the systems are clearly linear, we try a connection = Px. Inserting this into the ˙ equation, and then using the $\dot{x}$˙ equation, and a substitution for u, we obtain:

$$PAx + A^TPx + Qx - PBR^{-1}B^TPx + \dot{P} = 0$$

This has to hold for all x, so in fact it is a matrix equation, the matrix Riccati equation. The steady-state solution is given satisfies:

$$PA + A^TP + Q - PBR^{-1}B^TP = 0$$

This equation is the matrix algebraic Riccati equation (MARE), whose solution P is needed to compute the optimal feedback gain K. The MARE is easily solved by standard numerical tools in linear algebra. The equation $Ru + B^T\lambda = 0$ gives the feedback law:

$$u = R^1B^TPx$$

The LQR generates a static gain matrix K, which is not a dynamical system.[3] Hence, the order of the closed-loop system is the same as that of the plant.

4.2 Use LQR in MATLAB

Under the derivation of the LQR, we can write the following code in order to obtain the static gain matrix K. Simultaneously, we can draw the output of the system over time where we can see clearly that the system approaching a stable value.

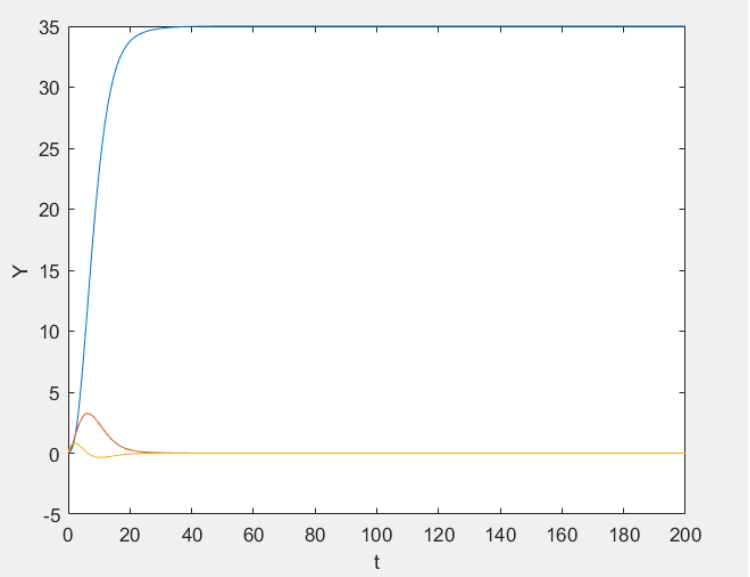
```
A=[0 1 0;0 0 1;-35 -27 -9];
B=[0;0;1];
C=[1 0 0 ;0 1 0 ;0 0 1 ];
D=[0;0;0;];
Q=[0 0 0;0 0 0;0 0 0];
R=1;
K=dlqr(A,B,Q,R)
Ac=[(A-B*K)];
Bc=[B];
Cc=[C];
Dc=[D];
t=0:0.005:200;
U=ones(size(t));
x0=[0.05 0.08 0];
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,t,x0);
plot(t,Y);
```

```
17 xlabel('t');
18 ylabel('Y');
19 u=-K*X';
```

Listing 1: LQR in MATLAB

The stable system solved by LQR is as follow:

Figure 1: Stable System Solved by LQR



The k it get is:

Figure 2: Stable System Solved by LQR

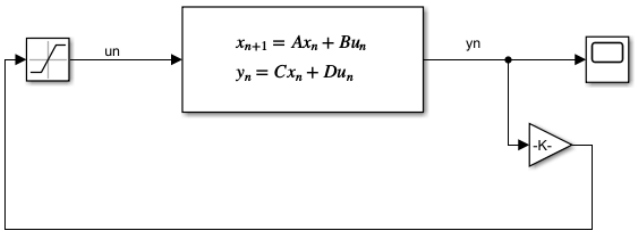
```
K =

  -34.9714  -26.7429  -8.2286
```

According to the result of LQR, We can draw the system block diagram of the problem by using Simulink in Matlab. There are totally three blocks in the whole diagram: Discrete State-Space, Gain, Scope.

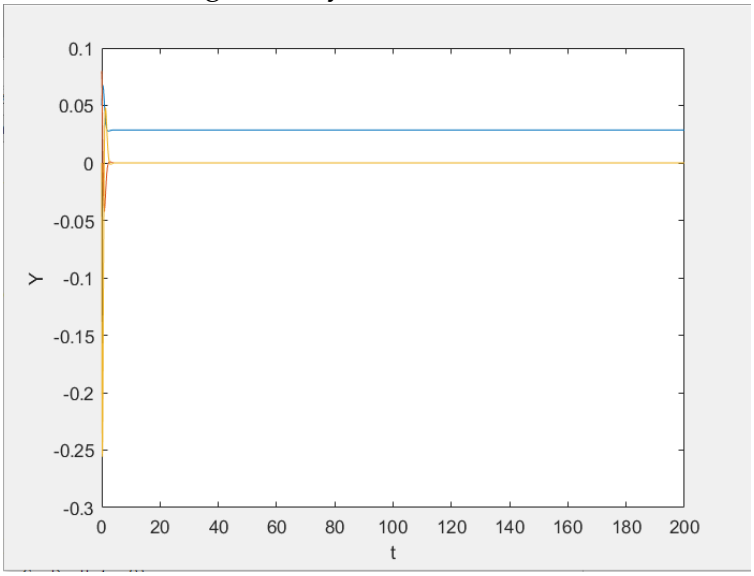
By applying Discrete State-Space block, with inputting un we can easily obtain yn which in this case is also equivalent to xn+1 as an output. According to the code, we are able to acquire the value of gain K which is mentioned in the LQR. Then we can obtain the negative feedback of the Discrete State-Space block which is -Ky(n). After connecting a scope with the output y(n), we are able to observe the system easily.

Figure 3: System in Simulink



With Simulink we get the oscillogram:

Figure 4: System in Simulink



4.3 Interior Point Method

We compare the GA[11] tool and the interior point method[6] tool in MATLAB, and finally use interior point method get best result.[5][2][9]

The primal-dual method's idea is easy to demonstrate for constrained nonlinear optimization. For simplicity, consider the all-inequality version of a nonlinear optimization problem:

minimize $f(x)$ subject to $c_i(x) \geq 0$ for $i = 1, \ldots, m, x \in \mathbb{R}^n,$ where $f : \mathbb{R}^n \to \mathbb{R}, c_i : \mathbb{R}^n \to \mathbb{R}$

The logarithmic barrier function is

$$B(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \log\left(c_i(x)\right)$$

The barrier function gradient is

$$g_b = g - \mu \sum_{i=1}^{m} \frac{1}{c_i(x)} \nabla c_i(x)$$

In addition to the original ("primal") variable x, we introduce a Lagrange multiplier inspired dual variable λ:

$$c_i(x)\lambda_i = \mu, \forall i = 1, \ldots, m$$

So we get an equation for the gradient:

$$g - A^T\lambda = 0$$

where the matrix A is the Jacobian of the constraints c(x).

Applying Newton's method we get:

$$\begin{pmatrix} W & -A^T \\ \Lambda A & C \end{pmatrix} \begin{pmatrix} p_x \\ p_\lambda \end{pmatrix} = \begin{pmatrix} -g + A^T\lambda \\ \mu 1 - C\lambda \end{pmatrix}$$

And each step is:

$$(x, \lambda) \to (x + \alpha p_x, \lambda + \alpha p_\lambda)$$

For T = 7 , upper bound = 5 low bound = -5 we get:

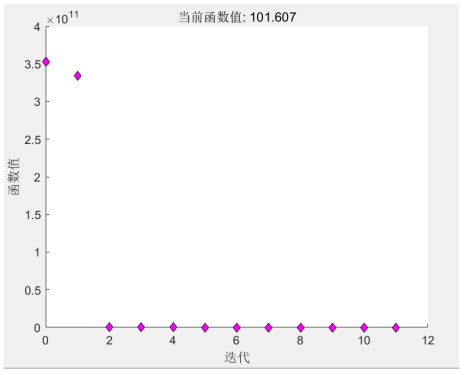


Figure 5: Optimization Routine

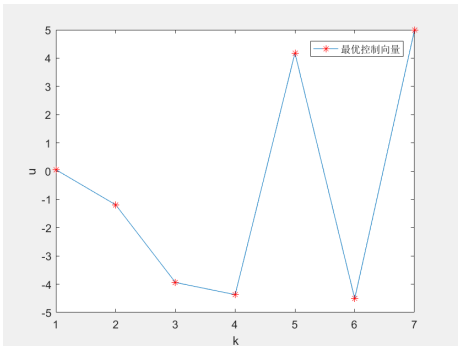


Figure 6: Step of u

You can find our source code in Appendix.

4.4 Other methods we tried

Along with Interior Point Method we tried a lot other methods in MATLAB, the program theories are as follows.

4.4.1 Dynamic Programming

The final general characteristic of the dynamic-programming approach is the development of a recursive optimization procedure, which builds to a solution of the overall N-stage problem by first solving a one-stage problem and sequentially including one stage at a time and solving one-stage problems until the overall optimum has been found. This procedure can be based on a backward induction process, where the first stage to be analyzed is the final stage of the problem and

problems are solved moving back one stage at a time until all stages are included. Alternatively, the recursive procedure can be based on a forward induction process, where the first stage to be solved is the initial stage of the problem and problems are solved moving forward one stage at a time, until all stages are included. In certain problem settings, only one of these induction processes can be applied (e.g., only backward induction is allowed in most problems involving uncertainties).

The basis of the recursive optimization procedure is the so-called principle of optimality, which has already been stated: an optimal policy has the property that, whatever the current state and decision, the remaining decisions must constitute an optimal policy with regard to the state resulting from the current decision.

Suppose we have a multistage decision process where the return (or cost) for a particular stage is:

$$f_n(d_n, s_n)$$

where dn is a permissible decision that may be chosen from the set Dn, and sn is the state of the process with n stages to go.

Noramlly, we have to solve the following problem:

$$v_n\left(s_n\right) = \operatorname{Max}\left[f_n\left(d_n, s_n\right) + f_{n-1}\left(d_{n-1}, s_{n-1}\right) + \cdots + f_0\left(d_0, s_0\right)\right]$$

subject to:

$$\begin{array}{ll} s_{m-1} = t_m\left(d_m, s_m\right) & (m = 1, 2, \ldots, n) \\ d_m \in D_m & (m = 0, 1, \ldots, n) \end{array}$$

The generally cost-to-go function is:

$$J(f, x(t)) = \min_{u(t) \in U} \left\{ h(x(\tau)) + \int_t^T g(x(\tau) \cdot u(\tau)) d\tau \right\}$$

4.4.2 Use DP to solve DLQR(Time limited and not use)

The cost function in DLQR is

$$J = \sum_{n=0}^{N-1} \left(x_n^T Q x_n + u_n^T R u_n\right) + x_N P x_N$$

Where Q,R P are positive definite matrixs.For $J_N(X_N)$ we have:

$$\begin{gathered} J_N\left(x_N\right) = x_N^T P x_N \\ J_{N-1}\left(x_{N-1}\right) = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + J_N\left(x_N\right) \end{gathered}$$

Given $x_{t+1} = Ax_k + B_{u_k}$:

$$J_{N-1}\left(x_{N-1}\right)=x_{N-1}^{T}Qx_{N-1}+u_{N-1}^{T}Ru_{N-1}+J_{N}\left(Ax_{N-1}+Bu_{N-1}\right)$$

Calculate the derivation of u and let $\nabla_u = 0$:

$$\begin{aligned}&\nabla_u\{(2)\}=2Ru+2B^TP(Ax+Bu)=0\\&u=-\left(R+B^TPB\right)^{-1}B^TPAx\end{aligned}$$

Sort out the upper formulas, let $u=-kx$ we can get:

$$\begin{aligned}J_{N-1}(x)&=x^T\hat{P}x\\\hat{P}&=Q+k^TRk+(A-Bk)^TP(A-Bk)\end{aligned}$$

So the DP iterative program is:

Algorithm 1: Use DP solve LQR

Input: x_0, A, B
Output: u_i

```
for i = N − 1, i >= 0, i − − do
    k = (R + B^T P B)^{-1} B^T P A ;
    u_i = −kx;
    P = Q + k^T R k + (A − Bk)^T P (A − Bk)
return (u_i);
```

4.4.3 Euler Function(Hard to optimize u)

By Lagrange undetermined multiplier method we can get:

$$\bar{J}=\sum_{k=0}^{T-1}\{\frac{1}{2}u^T(k)u(k)+\lambda(k+1)[-x(k+1)+Ax(k)+Bu(k)]\}$$

And this is the loss function.

Define the increment of performance indicator J in the Kth sampling period as:

$$L_k=\frac{1}{2}u^T(k)u(k)+\lambda(k+1)[-x(k+1)+Ax(k)+Bu(k)]$$

Step backward:

$$L_{k-1}=\frac{1}{2}u^T(k-1)u(k-1)+\lambda(k)[-x(k)+Ax(k-1)+Bu(k-1)]$$

Then solve the derivation:

$$\begin{aligned}
\frac{\partial L_k}{\partial x(k)} &= A^T\lambda^T(k+1) \\
\frac{\partial L_k}{\partial x(k)} &= -\lambda^T(k) \\
\frac{\partial L_k}{\partial u(k)} &= u(k) + B^T\lambda^T(k+1) \\
\frac{\partial L_k}{\partial u(k)} &= 0
\end{aligned} \tag{7}$$

By using Euler equation, we will have:

$$\begin{cases} A^T\lambda^T(k+1) = \lambda^T(k) \\ u(k) + B^T\lambda^T(k+1) = 0 \end{cases}$$

Let$\lambda^T(1) = C$, therefore:

$$\lambda^T(k) = (A^T)^{-(k-1)}C$$

$$u(k) = -B^T(A^T)^{-k}C$$

We can easily see that

$$x(k+1) = Ax(k) - BB^T(A^T)^{-k}C$$

Thus:

$$x(1) = Ax(0) - BB^TC \tag{8}$$
$$x(2) = A^2x(0) - ABB^TC - BB^T\left(A^T\right)^{-1}C \tag{9}$$
$$x(3) = A^3x(0) - A^2BB^TC - ABB^T\left(A^T\right)^{-1}C - BB^T\left(A^T\right)^{-2}C \tag{10}$$
$$\cdots \tag{11}$$
$$x(k+1) = A^{k+1}x(0) - \sum_{i=0}^{k} A^{k-i}BB^T\left(A^T\right)^{-i}C \tag{12}$$

So Euler Function constrains u into a small feasible region and we can solve it by iteration methods.

Unfortunately, this method cannot set u's limitation, it can only get the globally optimal solution, but we think this method works in optimize the cost function.

5 Problem Three: Change A and analyze its impact

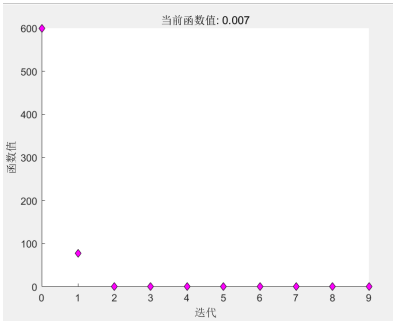
According to the gain K that we have calculated earlier, we inferred that the increase of A may lead to the raise of control series. After making trail of this method, we obtained the following experiment results which gave a visual representation of our inference.[7][4]

Our code for interior point method is:

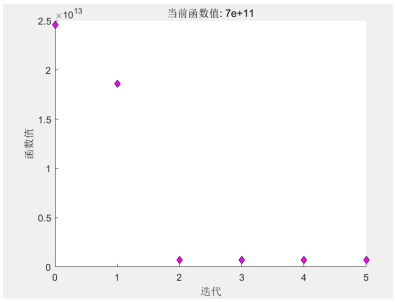
```
function y = fitness(u)
[rows,cols]=size(u);
x= [0.05; 0.08; 0]
A=[100 0 0;0 100 0;0 0 100];
B=[2;1;1];
sum=0;
L = 2
for k=1:1:cols
    x=A*x+B*u(k);
end
for k=2:1:cols
    sum=sum+u(k) * u(k);
end
y= sum + L * (x'*x);
end
```

Listing 2: Function for interior point method

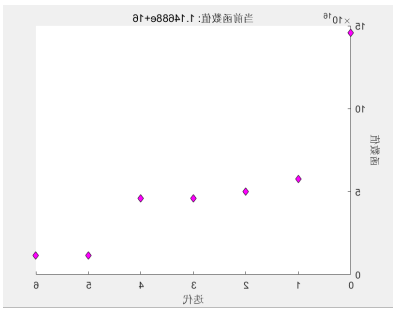
Which defines the function for optimizer. And we will change A for experiment.



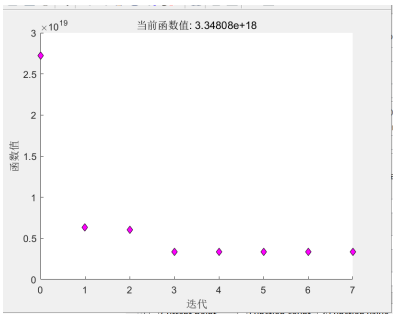
(a) A = diag(1 1 1)



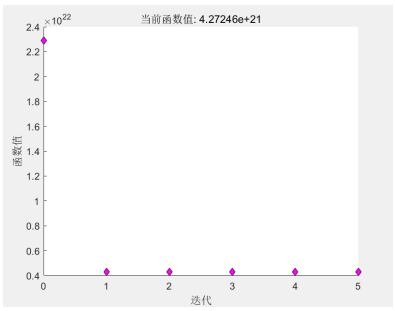
(b) A = diag(10 10 10)



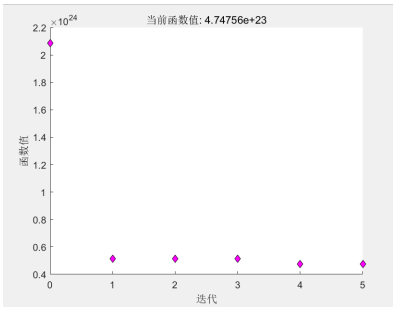
(c) A=diag(20 20 20)



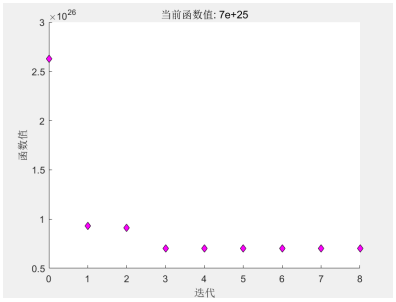
(d) A = diag(30 30 30)



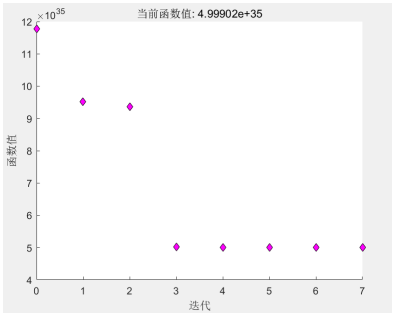
(e) A = diag(50 50 50)



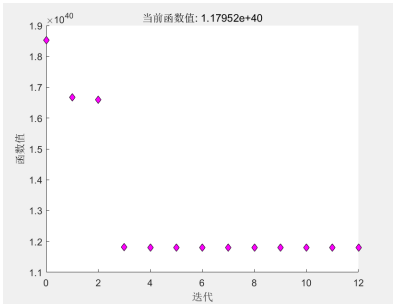
(f) A = diag(70 70 70)



(g) A = diag(100 100 100)



(h) A = diag(500 500 500)



(i) A = diag(1000 1000 1000)

Figure 7: Step of x while changing value of A

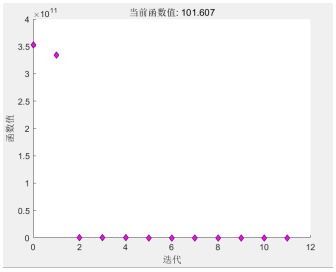
From the screenshots, we can see that the value of controls indeed increase with the growth of A at the beginning. Intuitively, we can see that when A is relatively large, the whole system will need to have a much bigger negative feedback at the beginning in order to constrain the state x(T) at the end to fall in a predetermined interval. After the x(t) is small enough, we only need small controls to adjust the system. If the controls are not big enough at the beginning, the x(t) will grow to a much larger number which may cause even bigger controls. Therefore the objective function will be able to reach the optimal value.

Figure 8: Steps of u

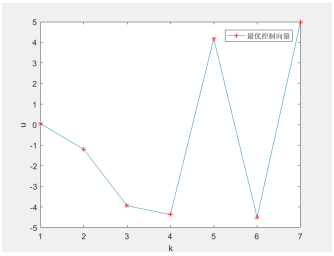
Final Point	1	2	3	4	5	6	7
a	-0.03	0	0	0	0	0	0
b	-0.45	1.326	1.518	1.539	1.507	1.459	2.758
c	-0.651	0.974	1.054	1.07	0.668	-9.577	9.999
d	-0.933	0.95	1.013	0.998	0.538	-9.945	1
e	-1.52	0.96	1.025	1.754	9.955	1	1
f	-2.113	0.921	0.686	-9.946	-8.865	1	1
g	-3.008	0.809	0.878	-9.945	1	1	1
h	-10	-10	-9.9	1	1	1	1
i	-10	-10	-10	1	1	1	1

6 Problem Four: Change upper bounds

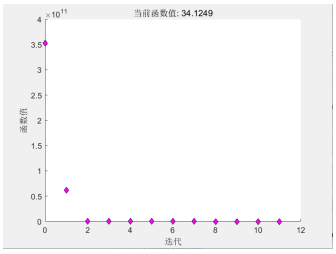
When we change the upper bounds, it can be seen from the screenshots that when the upper bound becomes smaller, the value of the objective function will become smaller. At the same time, the values of a few controls become closer to the upper bound. Obviously, when the upper bound becomes smaller, the x(T) becomes more difficult or even impossible to fall into the given range.



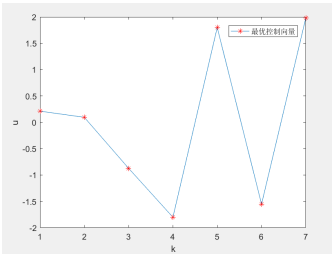
(a) T=7 upper = 5 low=-5



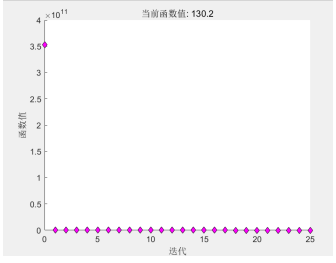
(b) T=7 upper = 5 low=-5



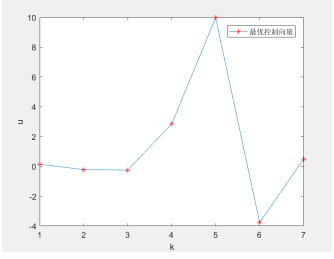
(c) T=7 upper = 2 low=-2



(d) T=7 upper = 2 low=-2



(e) T=7 upper = 10 low=-10

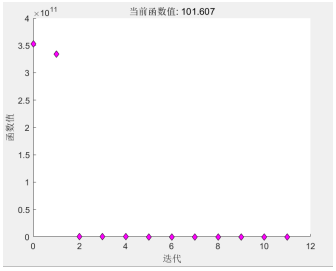


(f) T=7 upper = 10 low=-10

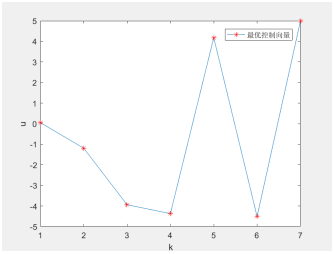
Figure 9: Change the bounds

7 Problem Five: Change the length of the time horizon: T

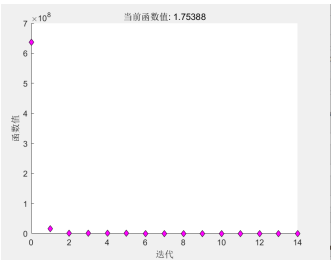
The screenshots show that when the T becomes larger, the value of the objective function will become sharply larger. When the T becomes super large, the values of the last few controls become close to the same value.



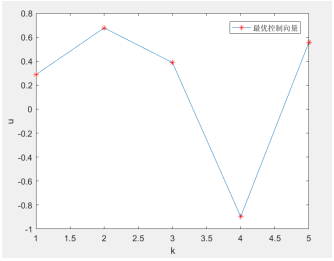
(a) T=7 upper = 5 low=-5



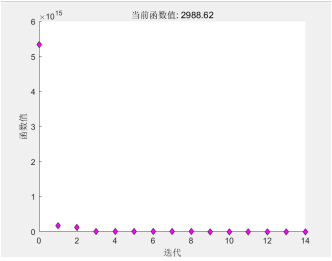
(b) T=7 upper = 5 low=-5



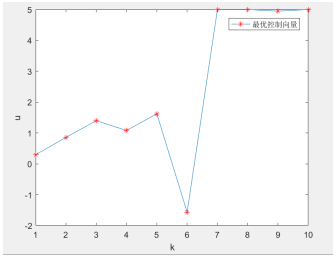
(c) T=5 upper = 5 low=-5



(d) T=5 upper = 5 low=-5



(e) T=10 upper = 5 low=-5



(f) T=7 upper = 10 low=-10

Figure 10: Change the length of the time horizon

References

[1] Barrett Ames and George Dimitri Konidaris. Bounded-error lqr-trees. In *IROS*, pages 144–150. IEEE, 2019.

[2] Greg Astfalk, Irvin Lustig, Roy E. Marsten, and David F. Shanno. The interior-point method for linear programming. *IEEE Software*, 9(4):61–68, 1992.

[3] Ravi Bhushan, Kalyan Chatterjee, and Ravi Shankar. Comparison between ga-based lqr and conventional lqr control method of dfig wind energy system. In *RAIT*, pages 214–219. IEEE, 2016.

[4] João Viana da Fonseca Neto, Ernesto Franklin Marcal Ferreira, and Patricia H. Moraes Rego. Online optimal dlqr-dfig control system design via recursive least-square approach and state heuristic dynamic programming for approximate solution of the hjb equation. In *SMC*, pages 3174–3179. IEEE, 2013.

[5] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optimization*, 6(2):342–361, 1996.

[6] Martin Mladenov, Vaishak Belle, and Kristian Kersting. The symbolic interior point method. In Satinder P. Singh and Shaul Markovitch, editors, *AAAI*, pages 1199–1205. AAAI Press, 2017.

[7] Matthew G. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, September 2005. Published online: 17 August 2005.

[8] Jan Stecha, Alena Kozáciková, Jaroslav Kozácik, and Jirí Lidický. Optimal control of a linear discrete system. *Kybernetika*, 9(5):374–388, 1973.

[9] Tao Wang, Renato D. C. Monteiro, and Jong-Shi Pang. An interior point potential reduction method for constrained equations. *Math. Program.*, 74:159–195, 1996.

[10] Yuh Shyang Wang, Nikolai Matni, and John C. Doyle. Localized lqr optimal control. *CoRR*, abs/1409.6404, 2014.

[11] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, June 1994.

Appendices

Appendix A Source Code

- Code in MATLAB for DLQR:

```
A=[0 1 0;0 0 1;-35 -27 -9];
B=[0;0;1];
C=[1 0 0 ;0 1 0 ;0 0 1 ];
D=[0;0;0;];
Q=[0 0 0;0 0 0;0 0 0];
R=1;
K=dlqr(A,B,Q,R)
Ac=[(A-B*K)];
Bc=[B];
Cc=[C];
```

```
Dc=[D];
t=0:0.005:200;
U=ones(size(t));
x0=[0.05 0.08 0];
[Y,X]=lsim(Ac,Bc,Cc,Dc,U,t,x0);
plot(t,Y);
xlabel('t');
ylabel('Y');
u=-K*X';
```

Listing 3: LQR in MATLAB

- Code in MATLAB for interior point method:

```
function y = fitness(u)
[rows,cols]=size(u);
x= [0.05; 0.08; 0]
A=[100 0 0;0 100 0;0 0 100];
B=[2;1;1];
sum=0;
L = 2
for k=1:1:cols
    x=A*x+B*u(k);
end
for k=2:1:cols
    sum=sum+u(k) * u(k);
end
y= sum + L * (x'*x);
end
```

Listing 4: Function for interior point method