**LINEAR DYNAMICAL SYSTEMS REPORT**



**School or Department:** Wu Han University

**Grade and Specialty:** Excellent engineer 2016

**Name:** Guo Yang, Kong Chuishun, Tang Rui

**Advisor:** Fangling Pu

May 7 2018

**【Abstract】**In this lab, we consider a useful application of matrix-vector multiplication, which is used to describe many systems or phenomena that change or evolve over time. And we use MATLAB to solve a problem by this way.

**【Key words】** matrix-vector multiplication, MATLAB, Linear dynamical systems

**Table of Contents**

[Abstract 1](#_Toc324279571)

[Chapter 1 Introduction 1](#_Toc324279574)

[1.1 Linear dynamical systems 1](#_Toc324279574)

[1.2 Linear dynamical system with input 1](#_Toc324279574)

[1.3 Markov model. 1](#_Toc324279574)

[1.4 Simulation 2](#_Toc324279574)

[Chapter 2 Question 2](#_Toc324279576)

[Chapter 3 Solution 3](#_Toc324279576)

[Analysis about Results 5](#_Toc324279574)

[Bibliography 5](#_Toc324279590)

[Appendix 6](#_Toc324279590)

**Chapter 1 Introduction**

**1.1 Linear dynamical systems**

Suppose x1,x2,... is a sequence of n-vectors. The index (subscript) denotes time or period, and is written as t; xt, the value of the sequence at time (or period) t, is called the state at time t. We can think of xt as a vector that changes over time, i.e., one that changes dynamically. In this context, the sequence x1,x2,... is sometimes called a trajectory or state trajectory. We sometimes refer to xt as the current state of the system (implicitly assuming the current time is t), and xt+1 as the next state, xt−1 as the previous state, and so on.

The state xt can represent a portfolio that changes daily, or the positions and velocities of the parts of a mechanical system, or the quarterly activity of an economy. If xt represents a portfolio that changes daily, (x5)3 is the amount of asset 3 held in the portfolio on (trading) day 5.

A linear dynamical system is a simple model for the sequence, in which each xt+1 is a linear function of xt:

xt+1 = Atxt, t = 1, 2, .... (1.1)

Here the n×n matrices At are called the dynamics matrices. The equation above is called the dynamics or update equation, since it gives us the next value of x, i.e., xt+1, as a function of the current value xt. Often the dynamics matrix does not depend on t, in which case the linear dynamical system is called time-invariant.

If we know xt (and At,At+1,...) we can determine xt+1,xt+2,... simply by iterating the dynamics equation (9.1). In other words: If we know the current value of x, we can ﬁnd all future values. In particular, we do not need to know the past states. This is why xt is called the state of the system. It contains all the information needed at time t to determine the future evolution of the system.

**1.2 Linear dynamical system with input.**

There are many variations on and extensions of the basic linear dynamical system model (1.1), some of which we will encounter later. As an example, we can add additional terms to the update equation:

xt+1 = Atxt + Btut + ct t = 1, 2, .... (1.2)

Here ut is an m-vector called the input, Bt is the n × m input matrix, and the n-vector ct is called the oﬀset, all at time t. The input and oﬀset are used to model other factors that aﬀect the time evolution of the state. Another name for the input ut is exogenous variable, since, roughly speaking, it comes from outside the system.

**1.3 Markov model.**

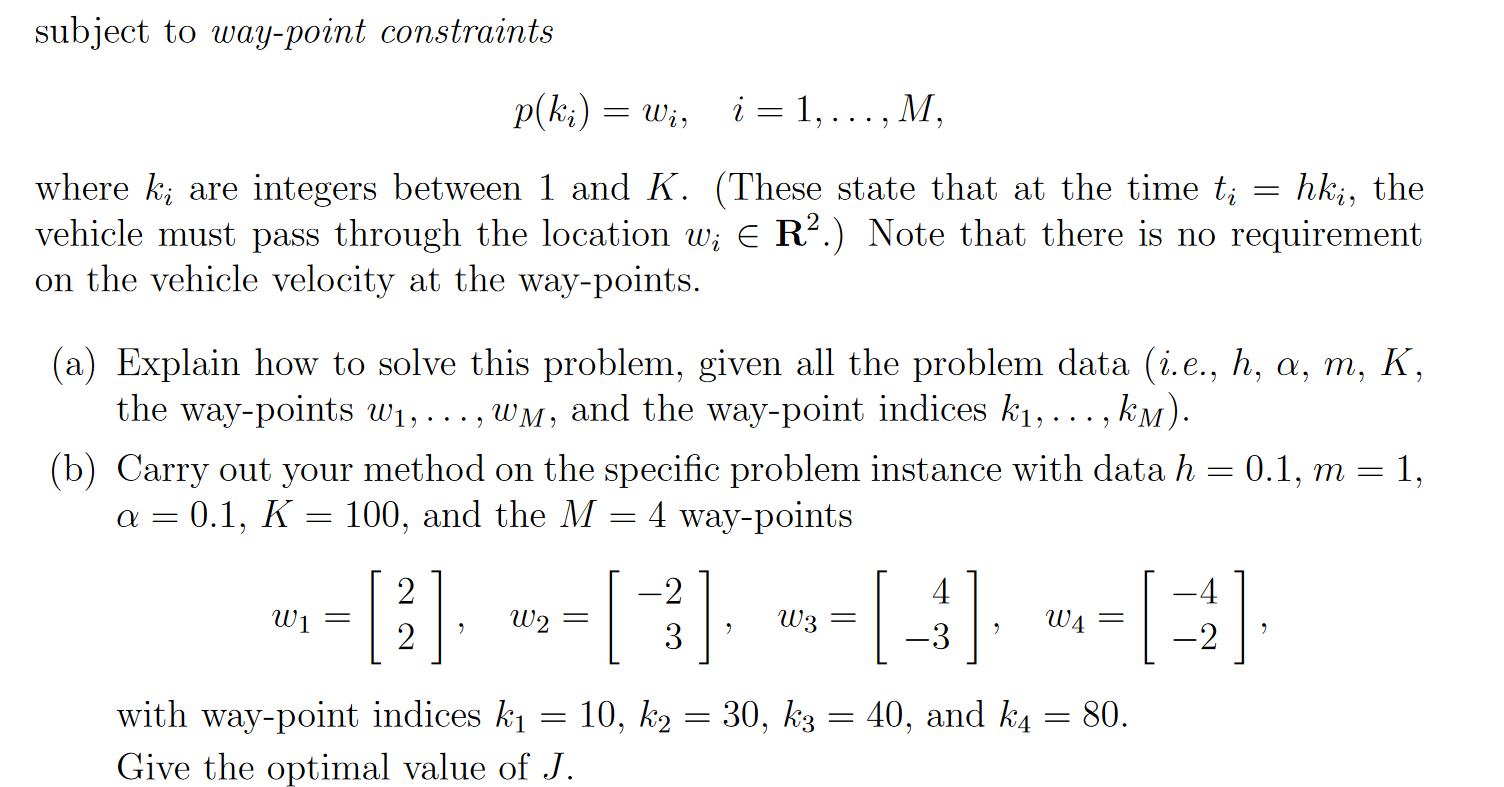
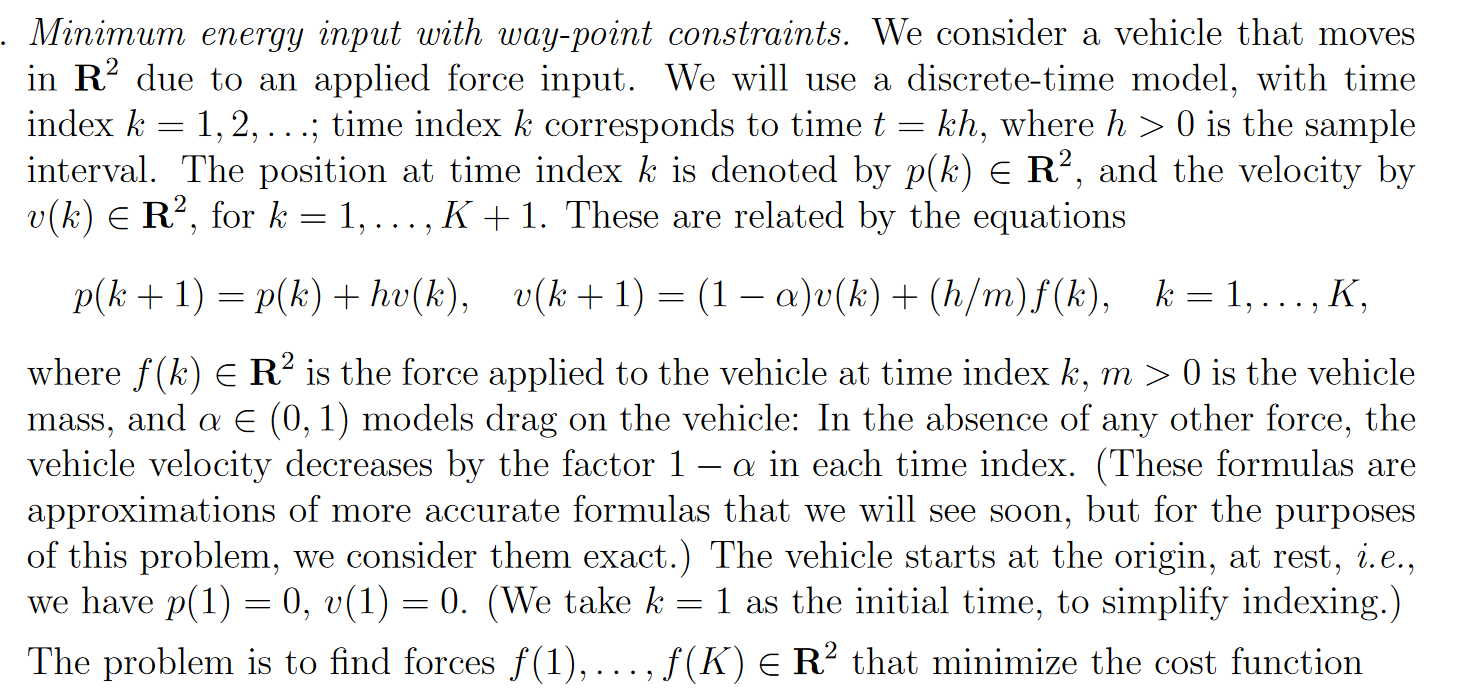
The linear dynamical system (1.1) is sometimes called a Markov model (after the mathematician Andrey Markov). Markov studied systems in which the next state value depends on the current one, and not on the previous state values xt−1,xt−2,.... The linear dynamical system (1.1) is the special case of a Markov system where the next state is a linear function of the current state. In a variation on the Markov model, called a (linear) K-Markov model, the next state xt+1 depends on the current state and K −1 previous states. Such a system has the form

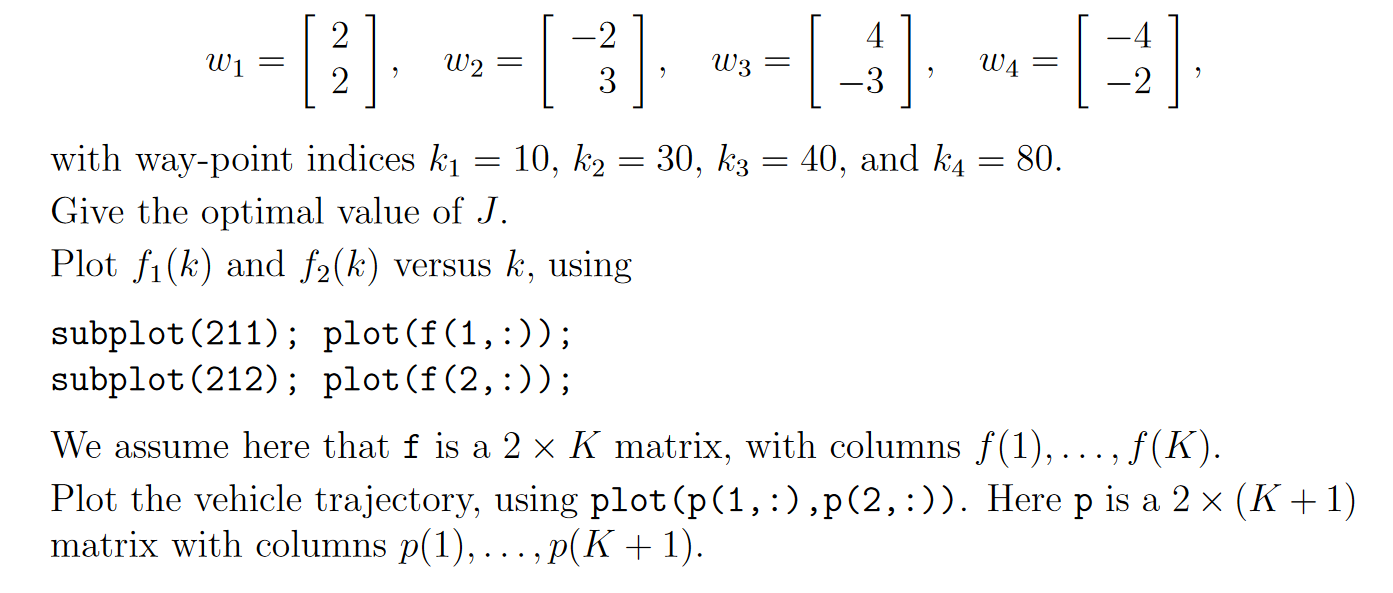
xt+1 = A1xt +···+ AKxt−K+1 t = K,K + 1,.... (1.3)

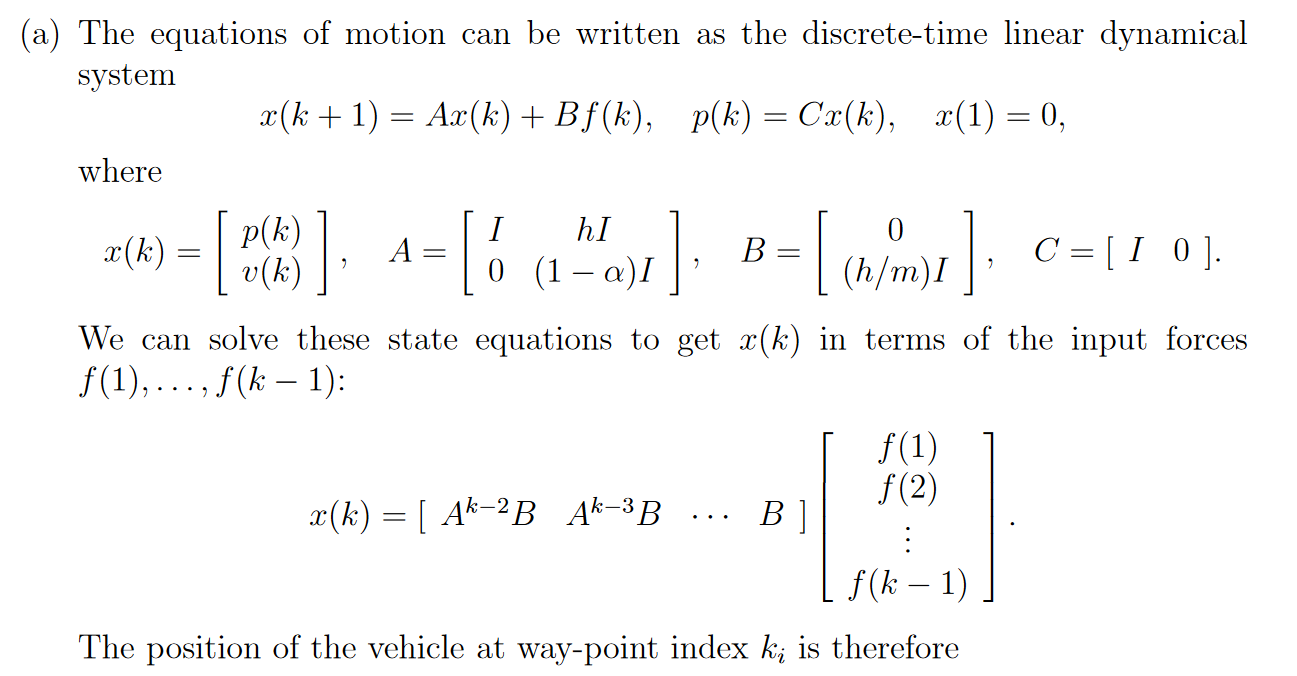
Models of this form are used in time series analysis and econometrics, where they are called (vector) auto-regressive models. When K = 1, the Markov model is the same as a linear dynamical system (1.1). When K > 1, the Markov model can be reduced to a standard linear dynamical system with an appropriately chosen state.

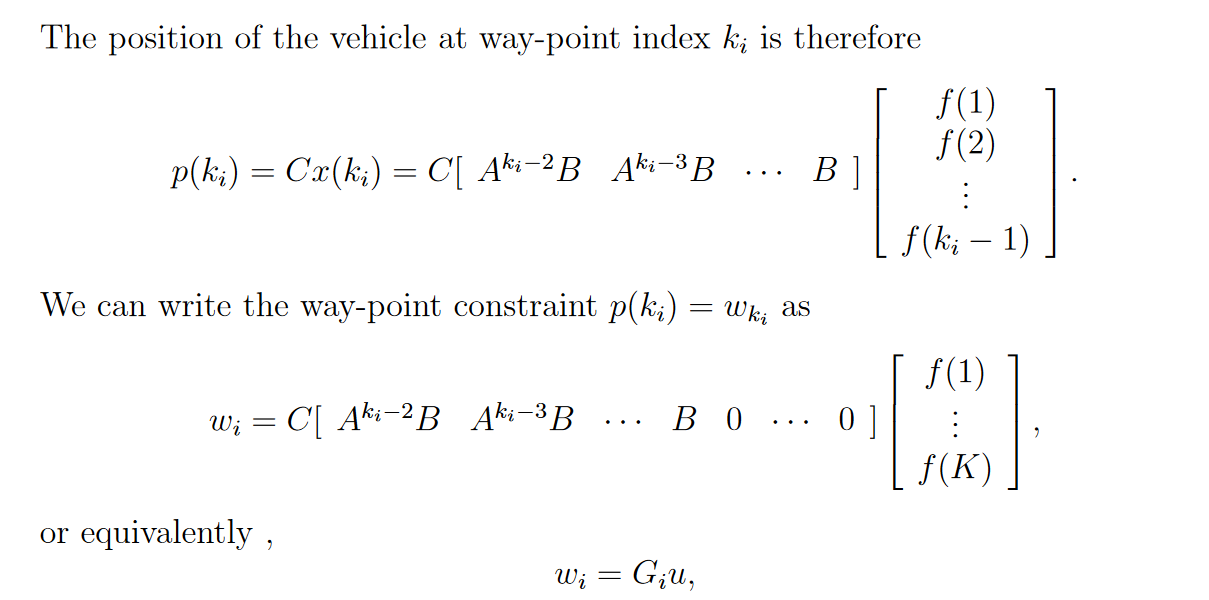
**1.4 Simulation.**

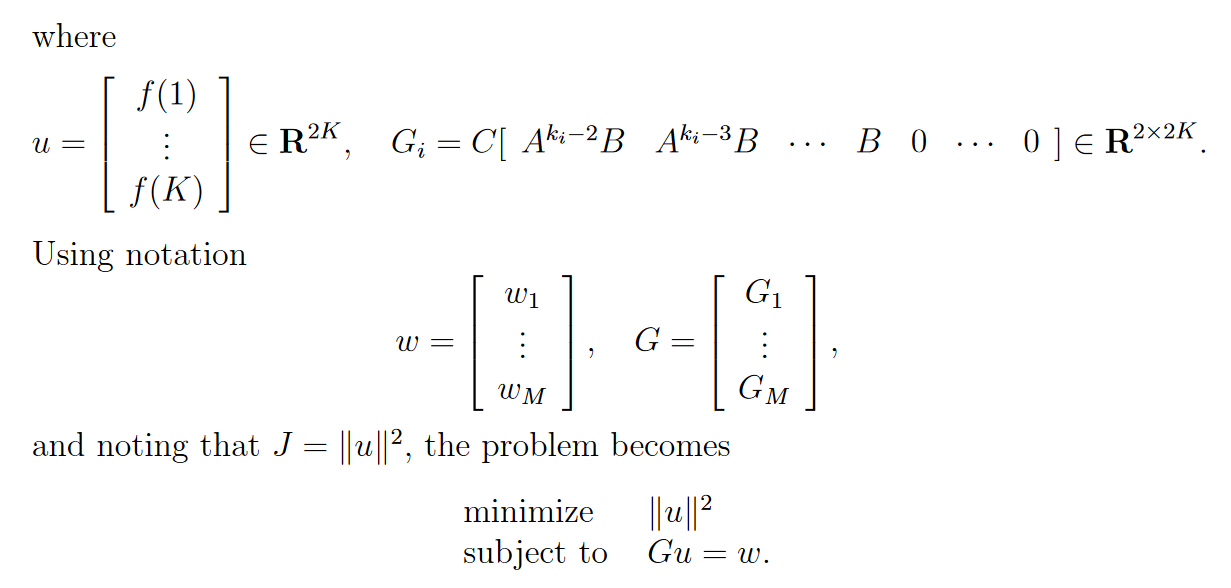
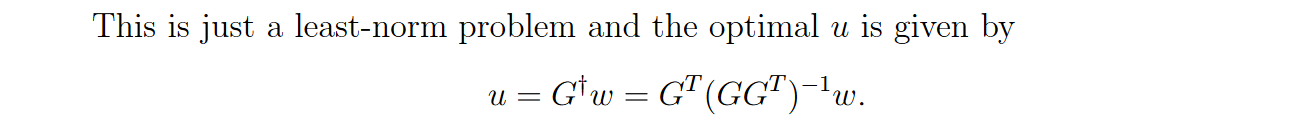
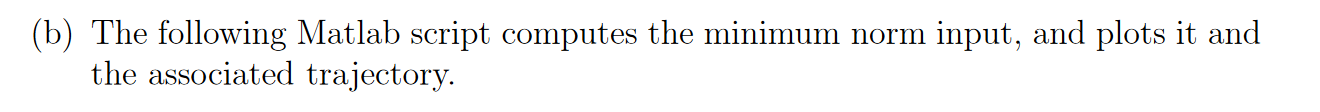
If we know the dynamics (and input) matrices, and the state at time t, we can ﬁnd the future state trajectory xt+1, xt+2,... by iterating the equation (1.1) or (1.2), provided we also know the input sequence ut, ut+1,...). This is called simulating the linear dynamical system. Simulation makes predictions about the future state of a system. (To the extent that (1.1) is only an approximation or model of some real system, we must be careful when interpreting the results.) We can carry out what-if simulations, to see what would happen if the system changes in some way, or if a particular set of inputs occurs.

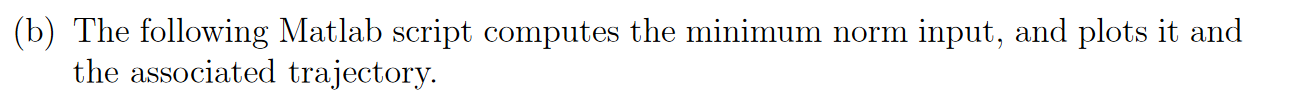
**Chapter 2 Question** 



**Chapter 3 Solution** 





% problem parameters

h = .1;

m = 1;

M=4;

alpha=0.1;

K = 100;

% way-points

k1=10; w1=[ 2; 2];

k2=30; w2=[ -2; 3];

k3=40; w3=[ 4; -3];

k4=80; w4=[-4; -2];

A = [eye(2) h\*eye(2); zeros(2) (1-alpha)\*eye(2)];

B = [zeros(2); h/m\*eye(2)];

C = [eye(2) zeros(2)];

[n, nn] = size(B);

k = [k1 k2 k3 k4];

G = [];

for i = 1:M

ABmatrix = [];

temp = B;

for j=1:k(i)-1

ABmatrix = [temp ABmatrix];

temp = A\*temp;

end

Gi = C\*[ABmatrix zeros(n, nn\*(K-k(i)+1))];

G = [G; Gi];

end

w = [w1; w2; w3; w4];

u = pinv(G)\*w;

% plotting the input

f = [u(1:2:end)’; u(2:2:end)’];

figure;

subplot(211); plot(f(1,:));

subplot(212); plot(f(2,:));

**Analysis about Results**

Figure 3.1 shows the minimum norm input forces. We see that for k ≥ 80, the optimal force is zero. This makes perfect sense: for k ≥ 80, the force f (k) does not affect the vehicle position at any of the way-points, so using any force on the

vehicle for k ≥80 just increases the cost J.

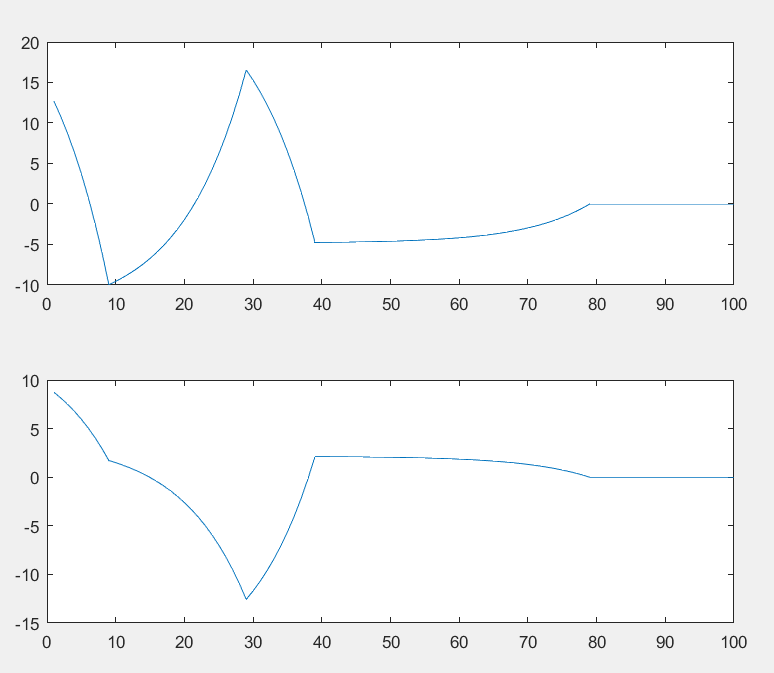
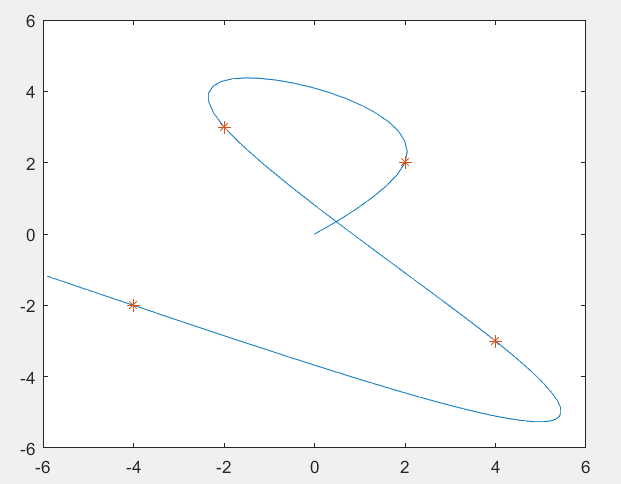
The optimal value of J is found to be 4770.5. Figure 3.2 shows the resulting trajectory.

Figure3.1 Figure3.2

**Bibliography**

[1]Bahdanau,Dzmitry,Cho,Kyunghyun,andBengio,Yoshua. Neural machine translation by jointly learning to align and translate. In ICLR’2015, arXiv:1409.0473, 2015.

**Appendix**

*code:*

close all;

clear all;

h = .1;

a = .1;

K = 100;

w1 = [2; 2];

w2 = [-2; 3];

w3 = [4; -3];

w4 = [-4; -2];

k1 = 10;

k2 = 30;

k3 = 40;

k4 = 80;

A = [1 0 h 0; 0 1 0 h; 0 0 1-a 0; 0 0 0 1-a];

B = [0 0; 0 0; h 0; 0 h];

% A1

A1 = [];

for ki = 1:k1-1,

A1 = [A1 A^(k1-1-ki)\*B];

end

z1 = [zeros(1,200-2\*(k1-1))];

z1 = [z1;z1;z1;z1];

A1 = [eye(2);0 0; 0 0]'\*[A1 z1];

%A2

A2 = [];

for ki = 1:k2-1,

A2 = [A2 A^(k2-1-ki)\*B];

end

z2 = [zeros(1,200-2\*(k2-1))];

z2 = [z2;z2;z2;z2];

A2 = [eye(2);0 0; 0 0]'\*[A2 z2];

%A3

A3 = [];

for ki = 1:k3-1,

A3 = [A3 A^(k3-1-ki)\*B];

end

z3 = [zeros(1,200-2\*(k3-1))];

z3 = [z3;z3;z3;z3];

A3 = [eye(2);0 0; 0 0]'\*[A3 z3];

%A4

A4 = [];

for ki = 1:k4-1,

A4 = [A4 A^(k4-1-ki)\*B];

end

z4 = [zeros(1,200-2\*(k4-1))];

z4 = [z4;z4;z4;z4];

A4 = [eye(2);0 0; 0 0]'\*[A4 z4];

Aall = [A1;A2;A3;A4];

W = [w1;w2;w3;w4];

f = Aall'\*inv(Aall\*Aall')\*W;

J = norm(f)^2

f1 = [];

f2 = [];

for i = 1:length(f),

if (mod(i,2)) == 0,

f2 = [f2; f(i)];

else

f1 = [f1; f(i)];

end

end

f = [f1 f2];

subplot(211); plot(f(:,1)); title('Force over time'); ylabel('Force');

subplot(212); plot(f(:,2)); ylabel('Force');

pause;

% Plotting p

x = [0;0;0;0];

for i = 1:K,

xnext = A\*x(:,i) + B\*[f(i,1); f(i,2)];

x = [x xnext];

end

close all;

p = [eye(2);0 0; 0 0]'\*x;

plot(p(1,:),p(2,:));

title('trajectory'); xlabel('p1'); ylabel('p2');