VIRTUAL CONTROL SYSTEMS LABORATORY

by

CHANNA NISANTHA BANDARA NAVARATNA, B.S.E.

A THESIS

IN

MATHEMATICS

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

Approved

May, 2001

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Professor W.P Dayawansa for his advice and invaluable support in this research and throughout my course of study at Texas Tech University. I want to thank him for not only for being my mentor but also a good friend. I wish to thank Professor Clyde F. Martin for his kindness, advice and encouragement in my course of study. I would also like to thank Professor Schovanec for his advice and serving as a member of my thesis committee. I like to extend my appreciation to Professor E. Allen for his help in clarifying many numerical problems I had during the development of this software.

I am also thankful for the undying support I have received from my parents. They have always been a constant source of encouragement in my life. I dedicate this work to my parents. Without their love and support, I would have never accomplished my goals.

CONTENTS

ACKNOWLEDGEMENTS	ii
ABSTRACT	v
LIST OF FIGURES	vi
I INTRODUCTION	1
II MATHEMATICAL MODELS	6
2.1 Simple Pendulum	6
2.2 Wedge	8
2.3 3-Link Inverted Pendulum	9
2.4 3-Link Inverted Pendulum on a Moving Cart	12
III CONTROL DESIGN TOOLS AND TECHNIQUES	17
3.1 Introduction to Linear Systems	17
3.2 PID controller	18
3.3 Linear Quadratic Regulator	19
3.4 Pole Placement	20
3.5 Observer-Based Controller Design	21
3.6 Control Tools Developed	23
3.6.1 Generalized Householder transformation	23
3.6.2 QR Decomposition	24
3.6.3 Matrix Balancing	24
3.6.4 ODE Solver	25
3.6.5 QZ algorithm	25
3.6.6 Robust Pole Assignment	26
3.6.7 Algebraic Riccati Equation	27
IV MANUAL	28
4.1 Simple Pendulum with PID Controller	28
4.2 Wedge with LQR and Pole Placement Techniques	29
4.3 3-Link Inverted Pendulum with LQR and Pole Placement	30

4.4	3-Link Inverted Pendulum on a Moving Cart	31
4.4.1	State Feedback using LQR and Pole Placement	31
4.4.2	Observer-Based Feedback using LQR and Pole Placement .	33
V	CONCLUDING REMARKS	34
	BIBLIOGRAPHY	35
	APPENDIX: JAVA ANIMATION ROUTINES	37

ABSTRACT

During the development of a control system, simulations can verify that the implemented algorithms work as expected. Smaller parts of the implementation can be tested before they are put together into a complete control system. In an existing plant, simulation can be used for optimization of the control system and control parameters can be tuned without affecting the real process. The goal of this thesis has been to develop applets to be used in a web page to simulate control systems. These web pages are to be mainly used as a tool for educational purposes to develop web-based control designs. It also provides some control algorithm routines and some routines to handle matrix operations. Control system simulation on the web environment is a research area where few studies have been done. Possibilities for simulations on the Internet are immense. With Java language's flexibility to execute on any computer, routines written in Java can easily be ported into many computers running many operating systems.

LIST OF FIGURES

2.1	Simple Pendulum	6
2.2	Wedge with masses m_1 and m_2	8
2.3	3-Link inverted pendulum	10
2.4	3-Link inverted pendulum on a moving cart	13
3.1	Plant and a controller	18
3.2	Observer-based controller design	22
4.1	Simple pendulum with PID controller	28
4.2	Wedge with a LQR and Pole Placement	29
4.3	3-link inverted Pendulum with a LQR and Pole Placement	31
4.4	3-Link inverted Pendulum on a moving cart	32
4.5	3-Link inverted pendulum with a fixed base and an observer	33

CHAPTER I

INTRODUCTION

During the last century, technology has advanced so dramatically that certain very complicated tasks can be performed with minimal human interaction. This is possible due to the emergence of advanced computer systems and the development of automatic control algorithms. When used together, solutions to complex problems can be determined in a fraction of a second. The power of a computer implementing a robust control algorithm can most of the time surpass human abilities to control a complicated system (plant).

The dynamics of a plant should be carefully studied before applying a control system. Engineers usually formulate the dynamical equations which represent the characteristics of the actual plant. The set of mathematical equations that describe a plant is called a *mathematical model*. Using this mathematical model, a control algorithm is selected so that the plant exhibits the desired output while ensuring the safety of the plant. This process is extremely important as the actual output is going to depend on this decision and the various parameters chosen.

Since one has so many ways of selecting parameters and algorithms, this process is quite complex. One cannot simply guess parameters and an algorithm and then apply them on the actual system. For example, if the plant is an airplane, a small mistake could result in a disaster. Several years ago, scientists made small-scale models and experimented with algorithms on the models. These experiments turned out to be very cumbersome and costly. Today, mathematical models are implemented, via computers, to gauge the response of the actual system to a particular control algorithm to a very high degree of accuracy. This process is called *simulation*, and it is an essential component in every practical control design task.

Simulation is an extremely important tool for every control engineer who is doing system design in industry. For students, simulations is an invaluable learning aide for studying control theory and systems. For arbitrarily nonlinear plants, there is often

no alternative to controller design by means of trial and error. through computer simulations.

Simulations can be done by many computer software programs. Many of these have been written specifically to operate on a certain computer or on a certain operating system. This is called *platform dependency*. Java is a computer language which overcame this barrier. A program written in Java is platform independent. That is, it can be executed on any computer. This opened up many avenues for Java programs. In the last few decades, the internet has become a vital part of every activity. The unprecedented growth rate of the internet has revealed a potential that has yet to be fully exploited.

A browser allows a user access a web page on the internet. There are many computer program languages that can be incorporated into a web page. Java is the most popular computer language that can be used in a web page. Java's versatility and richness in **Graphical User Interface (GUI)** environment make it an ideal candidate for web browsers. Java programs written for use in a web page are called *applets*, while programs written to be executed in a computer are called *applications*.

The internet is booming due to the fact that many things can be done using the internet without having to buy software or powerful computers. Control system simulations on the web are virtually unexplored. There are very few web sites which offer simulations in control systems. These sites do not offer much freedom to the user to change the parameters of the systems. Currently, hundreds of different simulation systems exist on the market. They are specialized for different application areas, different computing platforms, and different modelling strategies. A very popular simulation package is the SIMULINK package from MathWorks, Inc. It is an extension of Matlab, where the software depends upon the platform it runs. Another more powerful simulation package is SystemBuild from Integrated Systems Inc. [1]. Though this is very powerful, this is not available for PC/Windows or Macintosh computers [1]. Many other simulation software have many restrictions and cannot be used on the web, while many software programs allow limited freedom in changing

system parameters. Some simulation software (e.g., [4]), which has been recently developed uses Java in combination with Matlab to produce the output. This method uses an intermediate data file to exchange information. Due to the security restrictions in Java and on web servers, a web sites require permission to create/read files on a local hard disk. These techniques tend to be problematic on the web.

Though Java is an excellent choice for the web, it lacks in the area of mathematical routines. Recently, a few organizations attempted to fill this gap by providing routines for matrix operations. This thesis is an attempt to further fill this gap by developing control algorithms and also to create java applets to simulate control systems. The applets developed come with many choices of controlling the plant and provide extensive freedom in changing the parameters of the system.

One of the avenues opened with the internet is distance education. A good education program that is to be reached through the internet needs more than just lines of text. The use of sound and, more importantly, visualization are musts for education over the internet. Simulations and animations can provide a hands on experience to students, thus creating the necessity of a virtual laboratory on a web browser. Moreover, if students are given the freedom to select and change the experiment to suit their needs, a more friendly learning environment is created.

A virtual laboratory is a heterogeneous, distributed, problem-solving environment that enables a group of researchers, located around the world, to work together on a common set of projects. As with any other laboratory, the tools and techniques are specific to the domain of the research, but the basic infrastructure requirements are shared across disciplines. A good example of a virtual laboratory is a weather forecasting system that incorporates satellite data, large numbers of sensor inputs, and massive simulations for short- and medium-range weather predictions. A variation on this is predicting air quality through a virtual laboratory that couples weather models with ocean circulation models and pollution chemistry with ground and air based particulant sensors. In such a laboratory, an environmental scientist could suggest, given current conditions, when to temporarily shut down certain types of

manufacturing to avoid a potential crisis in air quality. Virtual laboratories have been proposed in many other disciplines including computational biology, radio astronomy, drug design, and materials science.

Many demonstration applets can be found under the web page <http://www.jhu.edu/signals/index.html>. These demonstrations are being developed in a project directed by Wilson J. Rugh at Johns Hopkins University. A few of these applets allow for change in the parameters, but none of the applets deal with controlling systems using modern control algorithms. The site <http://www.netzmedien.de/software/download/java/> contains six applets to demonstrate basic properties of physics. The site <http://www.univ-bechar.org/dipoleE.htm> contains an applet to plot 3D graphing electrostatic potential and lets the user select the type of graph out of 3 options. The site <http://members.nbci.com/surendranath/Applets.html> demonstrate basic math problems and basic physics problems using applets. The site http://www.nuclides.net/applets/radioactive_decay.htm simulates radioactive decay schemes for over 2600 nuclides in two different arrangements and with three different standard color schemes. The site <http://cluster.phys.uni-sofia.bg:8080/ross/ijc/comp.html#table1> demonstrates molecule cluster at different temperatures using animated applets. The site at <http://www.phy.ntnu.edu.tw/java/indexPopup.html> contains many applets to assist students in the learning of the basics of physics. The Department of Physics and Astronomy at Northwestern University at <http://www.physics.nwu.edu/ugrad/vpl/index.html>, University of Virginia. Department of Physics at http://www.phys.virginia.edu/classes/109N/more_stuff/Applets/ and the Department of Physics at the University of Oregon at <http://jersey.uoregon.edu/vlab/Piston/index.html> contain web sites with applets illustrating the basics in physics. The site <http://www.angelfire.com/wa/hurben/buff.html> provides a simulation of the famous Buffon's Needle experiment. Buffon's Needle is a Monte Carlo simulation for the determination of the value of pi. The Department of Electrical Engineering of University at Buffalo maintains a site <http://jas.eng.buffalo.edu/> with educational applets in electrical engineering. None of these sites contain applets that use control

algorithms. Many sites offer limited freedom to change parameters of the experiment.

In the next chapter, we look at some models used for the simulations in the web pages. Chapter III describes the algorithms used to control the models. Chapter IV provides explanation on how to use the developed software. The appendix includes all the Java routines developed.

CHAPTER II

MATHEMATICAL MODELS

Mathematical and computational models of several simple control systems are described in this chapter. These models are selected due to their simplicity and the fact that they are ubiquitous in undergraduate control systems laboratories throughout the country. Control design tools treated here are limited to PID control, Linear Quadratic Regulator and Pole placement. All standard texts in control theory contain detailed treatment of the theory of these design tools (e.g., [5], [7]) Java applets that animate computer models of these systems are presented in Appendix (Java Animation Routines).

2.1 Simple Pendulum

A simple pendulum as shown in Figure 2.1 is pin-jointed at one end. An external torque $\tau(t)$ is applied about the pin-joint and, the angle with vertical line $\theta(t)$ is to be controlled. In the following system, m denotes the mass of the pendulum, l denotes the length of the pendulum, and g denotes the gravitational force.

Figure 2.1: Simple Pendulum

Using Euler-Lagrange equations the dynamical system can be represented in the state space form as follows.

The kinetic energy of the pendulum is

$$T = \frac{ml^2\dot{\theta}^2}{8},$$

and the potential energy is

$$V = \frac{-mgl \cos \theta}{2}.$$

From Euler-Lagrange equations, we get,

$$\frac{ml^2\ddot{\theta}}{4} - \frac{mgl \sin \theta}{2} = \tau. \quad (2.1)$$

Assuming the angle θ is very small, we can approximate $\sin \theta$ by θ to obtain,

$$\ddot{\theta} = \frac{2g\theta}{l} + \frac{4\tau}{ml^2}. \quad (2.2)$$

This is a second-order differential equation. By introducing extra variables, a second-order differential equation can be expressed as a collection of first-order differential equations which in turn can be written in matrix form. This method will be used throughout this work to write down a higher order system of dynamic equation in matrix form.

Let

$$y = \theta$$

$$z = \dot{\theta}.$$

Then the system can be represented as,

$$\begin{bmatrix} \dot{z} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & \frac{2g}{l} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} + \begin{bmatrix} \frac{4}{ml^2} \\ 0 \end{bmatrix} \tau. \quad (2.3)$$

2.2 Wedge

Two masses of m_1 and m_2 are connected by a non-elastic cable of length L which run through a frictionless pulley at the corner of a wedge of mass M as shown in Figure 2.2. The wedge is pin-jointed to the base. The mass m_1 is applied a force F and the displacement of the mass from the equilibrium position is denoted by x . In this system, θ denotes the angular displacement from the vertical direction, x_0 denotes the distance from the corner of the wedge to mass m_1 at equilibrium, c denotes the distance from the corner of the wedge to the center of gravity of wedge, and α denotes the angle of the wedge.

Figure 2.2: Wedge with masses m_1 and m_2

Let

$$y = x_0 + x.$$

The kinetic energy of the wedge is

$$T = \frac{m_1}{2} \left[y^2 \dot{\theta}^2 + \dot{y}^2 \right] + \frac{m_2}{2} \left[(L - y)^2 \dot{\theta}^2 + \dot{y}^2 \right] + \frac{Mc^2 \dot{\theta}^2}{2},$$

and the potential energy is

$$V = m_1 g y \cos\left(\theta + \frac{\alpha}{2}\right) + m_2 g (L - y) \cos\left(\frac{\alpha}{2} - \theta\right) + M g c \cos \theta.$$

From Euler-Lagrange equations, we get

$$\ddot{x} = \frac{[m_1 y - m_2(L - y)]\dot{\theta}^2 - m_1 g \cos(\theta + \frac{\alpha}{2}) + m_2 g \cos(\frac{\alpha}{2} - \theta) - F}{m_1 + m_2}. \quad (2.1)$$

$$\ddot{\theta} = \frac{m_1 g y \sin(\theta + \frac{\alpha}{2}) - m_2 g(L - y) \sin(\frac{\alpha}{2} - \theta) + M g c \sin \theta}{m_1 y^2 + m_2(L - y)^2 + M c^2}. \quad (2.2)$$

At equilibrium position, $\theta = 0$, $\dot{\theta} = 0$, $x = 0$, $\dot{x} = 0$ and $x_0 = \frac{m_2 L}{m_1 + m_2}$.

If s represents the equilibrium position, a linear model can be obtained as follows:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{x} \\ \dot{\theta} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ddot{\theta}}{\partial \theta}|_s & \frac{\partial \ddot{\theta}}{\partial \dot{x}}|_s & \frac{\partial \ddot{\theta}}{\partial \theta}|_s & \frac{\partial \ddot{\theta}}{\partial x}|_s \\ \frac{\partial \ddot{x}}{\partial \theta}|_s & \frac{\partial \ddot{x}}{\partial \dot{x}}|_s & \frac{\partial \ddot{x}}{\partial \theta}|_s & \frac{\partial \ddot{x}}{\partial x}|_s \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{x} \\ \theta \\ x \end{bmatrix} + \begin{bmatrix} \frac{\partial \ddot{\theta}}{\partial F}|_s \\ \frac{\partial \ddot{x}}{\partial F}|_s \\ \frac{\partial \dot{\theta}}{\partial F}|_s \\ \frac{\partial \dot{x}}{\partial F}|_s \end{bmatrix} F.$$

This yields

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{x} \\ \dot{\theta} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{2Lm_1m_2g \cos \frac{\alpha}{2} + Mgc(m_1+m_2)}{m_1m_2L^2 + Mc^2(m_1+m_2)} & \frac{(m_1+m_2)^2 g \sin \frac{\alpha}{2}}{m_1m_2L^2 + Mc^2(m_1+m_2)} \\ 0 & 0 & g \sin \frac{\alpha}{2} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{x} \\ \theta \\ x \end{bmatrix} + \begin{bmatrix} \frac{-1}{m_1+m_2} \\ 0 \\ 0 \\ 0 \end{bmatrix} F. \quad (2.3)$$

2.3 3-Link Inverted Pendulum

Three links of an inverted pendulum are connected by frictionless pin joints as shown in Figure 2.3. The lengths and masses of the links from top to bottom are $2l_3$, $2l_2$, $2l_1$ and m_3 , m_2 , m_1 , respectively. One end of this structure is connected to a fixed base with a pin joint. A torque τ is applied about the uppermost pin joint. Angles θ_3 , θ_2 , θ_1 denotes the angular displacement of links from the vertical

direction. The torque is to be applied in such a way that the three pendulums reach equilibrium position of $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0$.

Figure 2.3: 3-Link inverted pendulum

The kinetic energy of the 3-link inverted pendulum is

$$T = \frac{m_1 l_1^2 \dot{\theta}_1^2}{2} + \frac{m_2}{2} [4l_1^2 \dot{\theta}_1^2 + l_2^2 \dot{\theta}_2^2 + 4l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2 - \theta_1)] + \frac{m_3}{2} [4l_1^2 \dot{\theta}_1^2 + 4l_2^2 \dot{\theta}_2^2 + l_3^2 \dot{\theta}_3^2 + 8l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2 - \theta_1) + 4l_1 l_3 \dot{\theta}_1 \dot{\theta}_3 \cos(\theta_3 - \theta_1) + 4l_2 l_3 \dot{\theta}_2 \dot{\theta}_3 \cos(\theta_3 - \theta_2)],$$

and the potential energy is

$$V = (m_1 + 2m_2 + 2m_3)gl_1 \cos \theta_1 + (m_2 + 2m_3)gl_2 \cos \theta_2 + m_3 gl_3 \cos \theta_3.$$

Let

$$\begin{aligned}
M_{12} &= 2(m_2 + 2m_3)l_1l_2 \cos(\theta_2 - \theta_1) \\
M_{13} &= 2m_3l_1l_3 \cos(\theta_3 - \theta_1) \\
M_{23} &= 2m_3l_2l_3 \cos(\theta_3 - \theta_2) \\
N_{12} &= 2(m_2 + 2m_3)l_1l_2 \sin(\theta_2 - \theta_1) \\
N_{13} &= 2m_3l_1l_3 \sin(\theta_3 - \theta_1) \\
N_{23} &= 2m_3l_2l_3 \sin(\theta_3 - \theta_2).
\end{aligned}$$

From Euler-Lagrange equations, we get

$$\begin{aligned}
&(m_1 + 2m_2 + 2m_3)l_1^2\ddot{\theta}_1 + M_{12}\ddot{\theta}_2 + M_{13}\ddot{\theta}_3 \\
&-N_{12}\dot{\theta}_2^2 - N_{13}\dot{\theta}_3^2 - (m_1 + 2m_2 + 2m_3)gl_1 \sin \theta_1 = 0,
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
&M_{12}\ddot{\theta}_1 + (m_2 + 2m_3)l_2^2\ddot{\theta}_2 + M_{23}\ddot{\theta}_3 \\
&+N_{12}\dot{\theta}_1^2 - N_{23}\dot{\theta}_3^2 - (m_2 + 2m_3)gl_2 \sin \theta_2 + \tau = 0,
\end{aligned} \tag{2.2}$$

$$M_{13}\ddot{\theta}_1 + M_{23}\ddot{\theta}_2 + m_3l_3^2\ddot{\theta}_3 + N_{13}\dot{\theta}_1^2 + N_{23}\dot{\theta}_2^2 - m_3gl_3 \sin \theta_3 - \tau = 0. \tag{2.3}$$

Let

$$\begin{aligned}
M &= \begin{bmatrix} (m_1 + 2m_2 + 2m_3)l_1^2 & M_{12} & M_{13} \\ M_{12} & (m_2 + 2m_3)l_2^2 & M_{23} \\ M_{13} & M_{23} & m_3l_3^2 \end{bmatrix}, \\
N &= \begin{bmatrix} 0 & N_{12} & N_{13} \\ -N_{12} & 0 & N_{23} \\ -N_{13} & -N_{23} & 0 \end{bmatrix},
\end{aligned}$$

and

$$D = \begin{bmatrix} (m_1 + 2m_2 + 2m_3)gl_1 & 0 & 0 \\ 0 & (m_2 + 2m_3)gl_2 & 0 \\ 0 & 0 & m_3gl_3 \end{bmatrix}.$$

The set of equations (2.1), (2.2) and (2.3) can be represented in matrix form as follows:

$$M \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = N \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \\ \dot{\theta}_3^2 \end{bmatrix} + D \begin{bmatrix} \sin \theta_1 \\ \sin \theta_2 \\ \sin \theta_3 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \tau. \quad (2.4)$$

The set of equations can be linearized by assuming in the neighborhood of equilibrium position, angles $\theta_1, \theta_2, \theta_3$ are very small thus $\sin \theta_1 \approx \theta_1, \sin \theta_2 \approx \theta_2, \sin \theta_3 \approx \theta_3$. Also $\dot{\theta}_1 \approx 0, \dot{\theta}_2 \approx 0, \dot{\theta}_3 \approx 0$.

Now equation 2.4 reduces to

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = M^{-1} D \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + M^{-1} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \tau. \quad (2.5)$$

This second-order system of equations can be written as a collection of first-order equations in matrix form,

$$\frac{d}{dt} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 & M^{-1} D \\ I & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + \begin{bmatrix} M^{-1} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tau. \quad (2.6)$$

2.4 3-Link Inverted Pendulum on a Moving Cart

Three links of an inverted pendulum are connected by frictionless pin joints as shown in Figure 2.4. One end of the series of pendulums is connected to a cart that moves on a horizontal plane. The lengths and masses of the links from top to bottom are $2l_3, 2l_2, 2l_1$ and m_3, m_2, m_1 , respectively. The cart has a mass of M and its distance from the equilibrium position is measured to be x . Angles $\theta_3, \theta_2, \theta_1$ denote

the angular displacement of links from the vertical direction. The cart is to be moved by a force F so that the the links of the pendulum reaches the stable position with $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 =$ and $x = 0$.

The kinetic energy of the system is

$$T = \frac{m_1}{2} [\dot{x}^2 + l_1^2 \dot{\theta}_1^2 - 2l_1 \dot{x} \dot{\theta}_1 \cos \theta_1] + \frac{m_2}{2} [\dot{x}^2 + 4l_1^2 \dot{\theta}_1^2 + l_2^2 \dot{\theta}_2^2 + 4l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2 - \theta_1) - 4l_1 \dot{x} \dot{\theta}_1 \cos \theta_1 - 2l_2 \dot{x} \dot{\theta}_2 \cos \theta_2] + \frac{m_3}{2} [\dot{x}^2 + 4l_1^2 \dot{\theta}_1^2 + 4l_2^2 \dot{\theta}_2^2 + l_3^2 \dot{\theta}_3^2 + 8l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2 - \theta_1) + 4l_1 l_3 \dot{\theta}_1 \dot{\theta}_3 \cos(\theta_3 - \theta_1) + 4l_2 l_3 \dot{\theta}_2 \dot{\theta}_3 \cos(\theta_3 - \theta_2) - 4l_1 \dot{x} \dot{\theta}_1 \cos \theta_1 - 4l_2 \dot{x} \dot{\theta}_2 \cos \theta_2 - 2l_3 \dot{x} \dot{\theta}_3 \cos \theta_3] + \frac{M \dot{x}^2}{2},$$

and the potential energy is

$$V = (m_1 + 2m_2 + 2m_3)gl_1 \cos \theta_1 + (m_2 + 2m_3)gl_2 \cos \theta_2 + m_3 l_3 g \cos \theta_3.$$

Figure 2.4: 3-Link inverted pendulum on a moving cart

Let

$$\begin{aligned}
M_{11} &= m_1 + m_2 + m_3 + M \\
M_{12} &= -(m_1 + 2m_2 + 2m_3)l_1 \cos \theta_1 \\
M_{13} &= -(m_2 + 2m_3)l_2 \cos \theta_2 \\
M_{14} &= -m_3 l_3 \cos \theta_3 \\
M_{22} &= (m_1 + 2m_2 + 2m_3)l_1^2 \\
M_{23} &= 2(m_2 + 2m_3)l_1 l_2 \cos(\theta_2 - \theta_1) \\
M_{24} &= 2m_3 l_1 l_3 \cos(\theta_3 - \theta_1) \\
M_{33} &= (m_2 + 2m_3)l_2^2 \\
M_{34} &= 2m_3 l_2 l_3 \cos(\theta_3 - \theta_2) \\
M_{44} &= m_3 l_3^2,
\end{aligned}$$

and

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{12} & M_{22} & M_{23} & M_{24} \\ M_{13} & M_{23} & M_{33} & M_{34} \\ M_{14} & M_{24} & M_{34} & M_{44} \end{bmatrix}.$$

Let

$$\begin{aligned}
N_{12} &= -(m_1 + 2m_2 + 2m_3)l_1 \sin \theta_1 \\
N_{13} &= -(m_2 + 2m_3)l_2 \sin \theta_2 \\
N_{14} &= -m_3 l_3 \sin \theta_3 \\
N_{23} &= 2(m_2 + 2m_3)l_1 l_2 \sin(\theta_2 - \theta_1) \\
N_{24} &= 2m_3 l_1 l_3 \sin(\theta_3 - \theta_1) \\
N_{34} &= 2m_3 l_3 l_2 \sin(\theta_3 - \theta_2),
\end{aligned}$$

$$N = \begin{bmatrix} 0 & N_{12} & N_{13} & N_{14} \\ 0 & 0 & N_{23} & N_{24} \\ 0 & -N_{23} & 0 & N_{34} \\ 0 & -N_{24} & -N_{34} & 0 \end{bmatrix},$$

and

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (m_1 + 2m_2 + 2m_3)gl_1 & 0 & 0 \\ 0 & 0 & (m_2 + 2m_3)gl_2 & 0 \\ 0 & 0 & 0 & m_3gl_3 \end{bmatrix}.$$

From Euler-Lagrange equations, we get

$$M \begin{bmatrix} \ddot{x} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = N \begin{bmatrix} \dot{x}^2 \\ \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \\ \dot{\theta}_3^2 \end{bmatrix} + D \begin{bmatrix} x \\ \sin \theta_1 \\ \sin \theta_2 \\ \sin \theta_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} F. \quad (2.1)$$

In a neighborhood of the equilibrium position, we can approximate $\sin \theta_1 \approx \theta_1$, $\sin \theta_2 \approx \theta_2$, $\sin \theta_3 \approx \theta_3$ and $\dot{\theta}_1 \approx 0$, $\dot{\theta}_2 \approx 0$, $\dot{\theta}_3 \approx 0$, $\dot{x} \approx 0$. Then,

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = M^{-1} D \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + M^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} F. \quad (2.2)$$

Representing the second-order system as a system of first-order equations, we get

$$\frac{d}{dt} \begin{bmatrix} \dot{x} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ x \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 & M^{-1}D \\ I & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ x \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + \begin{bmatrix} M^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} F. \quad (2.3)$$

CHAPTER III

CONTROL DESIGN TOOLS AND TECHNIQUES

The control techniques described below are implemented in Java language to be used in applets. Sections 3.2, 3.3, 3.4 and 3.5 introduce the concepts behind classical PID controllers, Linear Quadratic Regulator theory and Pole placement using feedback and observer based controlling when all the states of the plant cannot be measured. Section 3.6 describes the techniques used to implement those algorithms in Java language.

3.1 Introduction to Linear Systems

A linear system can be represented in the form:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad (3.1)$$

where $x(t)$ is a vector representing states, $u(t)$ is a vector with inputs to the system. n is the number of state variables, m is the number of inputs and A and B are $n \times n$ and $n \times m$ matrices, respectively.

Consider

$$\dot{x}(t) = Ax(t). \quad (3.2)$$

Its solution is $x(t) = e^{At}x(0)$. The response of (3.2) due to any non-zero initial state will approach zero (stable behavior) if and only if all eigenvalues of A have negative real parts [2].

If the input $u = Kx$, where K is an $m \times n$ matrix, then the system (3.1) can be written as

$$\dot{x} = (A + BK)x. \quad (3.3)$$

If the matrix K can be selected in such a way that the matrix $(A+BK)$ has all its eigenvalues on the left half of the complex plane, then the system (3.3) represents a stable system.

In many systems, it may not be possible to observe all the state variables which are required to implement the control law $u = Kx$. In these situations, an *observer* is used to estimate the state variables using the measured output $y = Cx$ and the input of the system.

In sections 3.3 and 3.4, we look at two methods of selecting K matrix. Section 3.5 describes the implementation of an observer, while section 3.2 looks at the classical PID controller.

3.2 PID controller

Figure 3.1 shows a linear time-invariant system (plant) and a controller for a Single Input Single Output (SISO) system. Let the input to the plant be $u(t)$, while the system states of the plant are given by $x(t)$ and the output is given by $y(t)$.

Figure 3.1: Plant and a controller

If $Y(s)$ is the Laplace transform of the system output and $U(s)$ is the Laplace transform of the input,

$$G_p(s) = \left. \frac{Y(s)}{U(s)} \right|_{\text{initial condition}=0}$$

is defined to be the *transfer function* of the plant. If the transfer function of the controller is $G_c(s)$, then the forward path transfer function is given by

$$G(s) = G_c(s)G_p(s).$$

It is easy to see that the eigenvalues of a system and the poles of the transfer function are the same. Suppose that $G_c(s)$ is chosen in such a way that the eigenvalues (poles)

of the new system $G(s)$ are in the left half of the complex plane. Consider a PID controller output,

$$u = K_I \int_{t_0}^t y dy + K_d \frac{dy}{dt} + K_p y.$$

Taking Laplace transforms,

$$G_c(s) = \frac{K_I}{s} + K_d s + K_p, \quad (3.4)$$

$$G_c(s) = \frac{K_I + K_p s + K_d s^2}{s}. \quad (3.5)$$

$G_c(s)$ introduces a pole and two possible zeros to the system and by varying the values of K_p , K_d and K_I poles and zeros of $G(s)$ can be varied.

3.3 Linear Quadratic Regulator

The linear quadratic regulator (LQR) from optimal control theory can be used to solve a family of regulator design problem. The idea is to minimize the cost J given by

$$J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) dt, \quad (3.6)$$

where Q is a semi-definite matrix and a R is a positive definite matrix. Standard assumptions are that (Q, A) is observable and (A, B) is controllable. With these standard assumptions, there exists a unique controller that minimizes the cost J in (3.6) and is given by

$$u(t) = -R^{-1} B^T K x(t),$$

where K is the unique positive definite solution of the algebraic Riccati equation

$$A^T K + K A - K B R^{-1} B^T K + Q = 0. \quad (3.7)$$

The solution to the algebraic equation (3.7) can be found by forming the Hamiltonian Matrix

$$M = \begin{bmatrix} A & -B R^{-1} B^T \\ -Q & -A^T \end{bmatrix}.$$

and computing a matrix T such that

$$T^{-1}MT = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix},$$

where a_{11} is stable. T can be computed using ordered Schur form [3]. Now partition matrix T as

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}.$$

Then the solution K is given by,

$$K = T_{21}T_{11}^{-1}.$$

3.4 Pole Placement

Consider a linear system with a single input,

$$\dot{x} = Ax + Bu, \tag{3.8}$$

$$y = Cx. \tag{3.9}$$

Let us assume that the system is controllable. Using a similarity transformation $x = Tz$ we can write the same system in the controller canonical form (see, e.g., [7])

$$\dot{z} = T^{-1}ATz + T^{-1}Bu, \tag{3.10}$$

$$y = CTz. \tag{3.11}$$

Where,

$$T^{-1}AT = \begin{bmatrix} -\alpha_1 & -\alpha_2 & \dots & \dots & -\alpha_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & & & \\ 0 & 0 & & 1 & 0 \end{bmatrix},$$

$$T^{-1}B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Since similarity transformations preserve eigenvalues, placement of eigenvalues in system (3.8) is equivalent to placing eigenvalues using system (3.10). Without loss of generality, we can assume that the system (3.8) is in the controller canonical form. We need to place poles of $(A + BK)$ at $P = [p_1, p_2, \dots, p_n]^T$ using the feedback law $u = Kx$. Let $K = [k_1, k_2, \dots, k_n]$. Then,

$$A + BK = \begin{bmatrix} -\alpha_1 + k_1 & -\alpha_2 + k_2 & \dots & \dots & -\alpha_n + k_n \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & & & \\ 0 & 0 & & 1 & 0 \end{bmatrix}.$$

The characteristic equation of $A + BK$,

$$\det|\lambda I - (A + BK)| = \lambda^n + (\alpha_1 - k_1)\lambda^{n-1} + (\alpha_2 - k_2)\lambda^{n-2} + \dots + (\alpha_n - k_n).$$

The desired characteristic equation with poles P is

$$\lambda^n + \beta_1\lambda^{n-1} + \beta_2\lambda^{n-2} + \dots + \beta_n.$$

Equating the coefficients, elements $k_1, k_2, \dots, k_n$ can be found as,

$$\alpha_1 - k_1 = \beta_1, \quad \alpha_2 - k_2 = \beta_2, \quad \dots, \quad \alpha_n - k_n = \beta_n.$$

3.5 Observer-Based Controller Design

In many control systems, only a few of the state variables are measured. Many control algorithms require all state variables to calculate the gain matrix. Thus we need to estimate the internal states of the plant using the inputs and the measured

states. Suppose in a linear system, we can only observe y given by

$$\dot{x} = Ax + Bu,$$

$$y = Cx.$$

Figure 3.2: Observer-based controller design

Assuming (A,B) controllable and (C,A) observable, we wish to estimate the states x using an observer as in Figure 3.5. Let the control input to the plant be $u = K\hat{x} + v$. A full-order observer (state estimator) with the characteristics,

$$\dot{\hat{x}} = (A + EC)\hat{x} - ECx + Bu,$$

is chosen here. The matrix E is to be determined such that $\hat{x} \rightarrow x$. By subtracting the system equation from the above equation, we obtain

$$\dot{\hat{x}} - \dot{x} = (A + EC)(\hat{x} - x).$$

If we select matrix E in such a way that all eigenvalues of $(A+EC)$ are in the left half of the complex plane, then the error $\epsilon = (\hat{x} - x) \rightarrow 0$ exponentially as $t \rightarrow \infty$. Let us write the new composite system as,

$$\frac{d}{dt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -EC & A + EC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} u. \quad (3.12)$$

Let the input to the plant be $u = K\hat{x}$. Then

$$\frac{d}{dt} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} = \begin{bmatrix} A & BK \\ -EC & A + BK + EC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}. \quad (3.13)$$

Using a similarity transformation

$$\begin{bmatrix} x \\ \epsilon \end{bmatrix} = \begin{bmatrix} I & 0 \\ -I & I \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix},$$

$$\frac{d}{dt} \begin{bmatrix} x \\ \epsilon \end{bmatrix} = \begin{bmatrix} A + BK & BK \\ 0 & A + EC \end{bmatrix} \begin{bmatrix} x \\ \epsilon \end{bmatrix}. \quad (3.14)$$

Therefore, if the eigenvalues of $(A+BK)$ and $(A+EC)$ are on the left half plane of the complex plane, system (3.12) is stable and $\hat{x} \rightarrow x$ as $t \rightarrow \infty$

3.6 Control Tools Developed

Development of software for PID, LQR and Pole placement algorithms involve development of many other auxiliary routines. Following is a description of the routines developed under this project. These use the Jama matrix routines developed by Mathworks and the National Institute of Standards and Technology (NIST). It also uses Jampack routines developed by Pete Stewart of the University of Maryland and NIST.

3.6.1 Generalized Householder transformation

A generalized Householder matrix is a symmetric, orthogonal matrix of the form

$$I + vu^T,$$

where $v^T u = -2$, and v is a scalar multiple of u . Given a vector x , a Householder matrix H_k can be found so that

$$H_k x = x + (u^T x)v$$

has one non-zero element at the desired k^{th} position.

Premultiplication of a matrix A by a Householder matrix Q_j will make all the elements of the desired i^{th} column equal to zero except for the (i,j) element. When it is required as in the QZ algorithm to make the elements of the j^{th} row equal to zero except for (i,j) element, matrix A is postmultiplied by the matrix transpose of the Householder transformation obtained using the row j .

3.6.2 QR Decomposition

Orthogonal matrix triangularization (QR decomposition) reduces a real (m,n) matrix A with $m \geq n$ and full rank to a much simpler form. It guarantees numerical stability by minimizing errors caused by machine roundoffs. A suitably chosen orthogonal matrix Q will triangularize the given matrix A ,

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Using a sequence of Householder transformations $U_1, U_2, \dots, U_r$, the matrix QA can be made upper trapezoidal where $Q = U_r U_{r-1} \dots U_1$, $r = \min(m-1, n)$. Since each U_j , $j = 1, 2, \dots, r$ is unitary, Q is also unitary.

3.6.3 Matrix Balancing

Balancing a matrix A consists of finding a diagonal similarity transform $DA D^{-1}$ such that for any i , row i and column i have the same norm. Usually done before calculating the eigenvalues of A , balancing can improve the speed and accuracy with which the eigenvalues are computed. To decrease the time taken by an eigensolver, in practice the matrix A is usually permuted before balancing to isolate diagonal blocks. Balancing can improve the accuracy of computed eigenvalues by sometimes reducing the norm of the matrix. Osborne [13] showed that balancing a matrix in the 2-norm also minimizes the Frobenius norm. Eaves et al. [12] showed that balancing in the 1-norm minimizes the sum of the elements, if A is non-negative.

3.6.4 ODE Solver

In order to solve an ordinary differential equation of the form $\dot{y} = f(t, y)$ where y is a vector of size n , at discrete time intervals, second-order and fourth-order Runge-Kutta methods are used.

Second-order Runge-Kutta method computes y_{n+1} from y_n using.

$$\begin{aligned}h &= t_{n+1} - t_n \\k_1 &= \frac{h}{2} f(t_n, y_n) \\k_2 &= h f(t_n + \frac{h}{2}, y_n + k_1) \\y_{n+1} &= k_2 + y_n\end{aligned}$$

Fourth-order Runge-Kutta method computes y_{n+1} from y_n using.

$$\begin{aligned}h &= t_{n+1} - t_n \\k_1 &= h f(t_n, y_n) \\k_2 &= h f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\k_3 &= h f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\k_4 &= h f(t_n + h, y_n + k_3) \\y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\end{aligned}$$

3.6.5 QZ algorithm

Given matrices A and B where B is not necessarily non-singular, there exist unitary matrices Q and Z such that QAZ and QBZ are both upper triangular [8]. This method does not use inversions of B or its submatrices.

The algorithm proceeds in four stages. In the first stage, a generalization of the Housholder transformations is used to reduce A to upper Hessenberg form and simultaneously B to upper triangular form. In the second step a generalization of implicit double shift QR algorithm (see [14], [8]) is used to reduce A to a quasi-upper

triangular matrix while the triangular form of B is maintained. Using the same form of Householder transformations, in the third stage the quasi-triangular matrix A is effectively reduced to triangular form in which no two consecutive subdiagonal elements are non-zero. This means that the original problem is reduced to a set of 1×1 and 2×2 subproblems. Without loss of generality, consider matrices A and B to be 2×2 submatrices from the diagonal of the original A and B . Householder transformation matrix Z is used to annihilate the first element of the larger of the two rows of $A - \lambda B$, where λ is computed as given by (5.1) on page 252 of [8]. Then a Householder matrix Q is chosen to annihilate the (2,1) element of the smaller of the two matrices AZ and λBZ . Then the (2,1) element of both QAZ and QBZ can be shown to be negligibly small.

3.6.6 Robust Pole Assignment

For the problem described in section 3.4, the algorithm described in [10], [15] is used. Let the plant be described by $\dot{x} = Ax + Bu$ and a feedback law $u = Kx$ to be determined so that the poles of the closed loop system coincides with the user specified poles $P = [\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n]^T$. This method consists of three steps.

In the first step, matrix B is decomposed using QR decomposition to determine U_0 , U_1 and Z such that

$$B = \begin{bmatrix} U_0 & U_1 \end{bmatrix} \begin{bmatrix} Z \\ 0 \end{bmatrix}.$$

Then for each λ_j QR decomposition is used as follows

$$(U_1^T (A - \lambda_j I))^T = \begin{bmatrix} \hat{S}_j & S_j \end{bmatrix} \begin{bmatrix} R_j \\ 0 \end{bmatrix}.$$

In second step, vectors $x_j = S_j w_j \in H_j = N\{U_1^T (A - \lambda_j I)\}$ are selected with $\|x_j\|_2 = 1$, $j=1, 2, \dots, n$ such that $X = [x_1, x_2, \dots, x_n]$ well conditioned. This is done by choosing vectors $x_j \in H_j$ $j=1, 2, \dots, n$ such that each vector is as orthogonal as possible to the space spanned by the remaining vectors. This can be achieved

by one of the four methods described in [10]. In the third step matrix $M = X\Lambda X^{-1}$ where $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_n\}$. The feedback gain matrix K is calculated by

$$K = Z^{-1}U_0^T(M - A).$$

3.6.7 Algebraic Riccati Equation

In order to minimize the cost J involved in the Linear Quadratic Regulator (LQR) problem described in section 3.3, it is necessary to solve the Algebraic Riccati Equation (3.7). Algorithms given in [9], [11], [20] and [15] are used to solve the equation (3.7). Using the Schur Decomposition, an orthogonal Matrix U is found such that $U^T M U = S$ where M is the Hamiltonian matrix

$$M = \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix},$$

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}, \quad U_{ij} \in \mathbb{R}^{n \times n}$$

and

$$S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$$

a quasi-upper triangular matrix. Then the feedback gain matrix K is given by

$$K = U_{21}U_{11}^{-1}.$$

CHAPTER IV

MANUAL

All applets developed in this thesis have been developed using the JDK1.2 of Sun Microsystems Corporation (<http://java.sun.com/>). Thus applets are expected to load and run on any popular browser. In certain cases, a Java plug-in may be needed to be installed as the animations were done using Java 2D Graphics. This plug-in can be downloaded from <http://java.sun.com/products/plugin/>.

Applets obtain the current time from the system clock on the computer it runs. Animations are refreshed in every $100ms$, and the calculations are done at every $10ms$ time interval in order to reduce the numerical instabilities.

4.1 Simple Pendulum with PID Controller

Once the applet is loaded and initialized, it can be simulated. The applet on the web page produces Figure 4.1

Figure 4.1: Simple pendulum with PID controller

Some abbreviations used include the following: *ref* refers to the reference or set-point, *prop* refers Proportional gain, *inte* refers to the Integral gain, *deri* refers to the Derivative gain and *Angle* refers to the angle of the pendulum with the vertical line. Initially, the PID gain settings are kept at zero value. The scroll bars let the user change the values of the fields they refer to while the animation is taking place. Thus one can experiment to see the effect of change of variables on the output of the plant. The *Stop* button temporarily halts the animation, while the *Reset* button initializes the system parameters with current values displayed on the applet.

4.2 Wedge with LQR and Pole Placement Techniques

Figure 4.2 shows the applet which shows on the web page. Though the two masses on the wedge are connected by a rigid frictionless cable which runs over a pulley at the corner of the wedge, it is not shown on the web graphics. Two control algorithms, LQR and Pole placement are available to stabilize the wedge.

Figure 4.2: Wedge with a LQR and Pole Placement

m_1 refers to the weight of the mass on the left wall of the wedge, while the m_2 refers to the weight of the mass on the right wall of the wedge. M refers to the weight of the wedge itself, and θ and x refer to the angle deviated from the equilibrium and deviation of distance of the mass on the left wall from equilibrium respectively. Once the simulation starts, values in the fields m_1 , m_2 and M cannot be changed. In order to start the animation, first a control algorithm (Radio buttons **Pole** for Pole placement or radio button **LQR** for Linear Quadratic Regulator) should be selected. The applet comes with pre-selected values for Poles and matrices Q, R . You can change these values at any time by pressing the appropriate button (Buttons **Poles**, **GetQ** and **GetR**). The button **Pause** is to temporarily halt the simulation, while the button **Continue** can be used to restart a paused simulations. If the values m_1 , m_2 or M need to be changed, the simulation must be paused to activate the scroll bars for the above fields. The initial values for θ and x can be changed when the simulation is in *pause* mode. When these values are changed, the Java graphics will redraw the wedge to reflect the changes. If buttons **Pole**, **GetQ** or **GetR** were pressed while a simulation is taking place, a window of appropriate size will pop up and the current simulation will be paused. The window which appears can be used to enter the values for Poles, Q and R matrices depending on the button user pressed. Once the values have been entered, close the window by clicking the standard close button on the upper right hand-corner of the window.

The check box **Sin** can be checked on/off at any time during the simulation. It introduces a sinusoidal disturbance on the mass m_1 . By selecting this option, once the system has been stabilized using one of the algorithms, one can see that the wedge also reflects periodic angular displacement from the stable position. The system can be stabilized again by deselecting this option.

4.3 3-Link Inverted Pendulum with LQR and Pole Placement

Figure 4.3 shows the applet once loaded using a web browser. The masses and angles deviated from vertical in counterclockwise direction of the three links are

displayed in the `m3`, `m2`, `m1` and `theta3`, `theta2`, `theta1` text fields, respectively. The fields `theta3`, `theta2`, `theta1` update during the simulations to give the precise angles. The actions of buttons `Simulate`, `Pause`, `Continue`, `GetR`, `GetQ` and `Pole` are similar to the previous applet. A user should select one of the radio buttons from `LQR` or `Pole` to specify the control algorithm to be used.

Figure 4.3: 3-link inverted Pendulum with a LQR and Pole Placement

4.4 3-Link Inverted Pendulum on a Moving Cart

4.4.1 State Feedback using LQR and Pole Placement

Figure 4.4 shows a 3-link inverted pendulum connected to a moving base. By moving the cart horizontally, the 3 links and the cart are to be brought to an equilibrium position indicated by a blue dashed line. It is assumed that all the states of the plant (3-link pendulum and the cart) are available for state feedback using either LQR or Pole placement algorithms.

Figure 4.4: 3-Link inverted Pendulum on a moving cart

A user must select either LQR or Pole Placement (radio buttons LQR and Pole) before simulation begins (button Simulate). A user can change all the system parameters on this model including the lengths by typing the values in the fields `length3`, `length2` and `length1`. As in the previous applet, masses `m3`, `m2`, `m1` refer to the three links of the pendulum from top to bottom, while angles `theta3`, `theta2` and `theta1` refer to deviation of angles from vertical in counterclockwise direction. Field `M` refers to the weight of the cart, while field `x` refers to the distance between the cart and equilibrium position. System parameters can be changed by pressing one of the following buttons: `pause`, `Pole`, `GetR` or `GetQ`. When the button `Simulate` is pressed, the applet initializes all its parameters to the contents in the fields, redraws the pendulum to fit into the graphic window by scaling the figure, and displays the gain matrix K ($u = Kx$) on the top row of the applet.

4.4.2 Observer-Based Feedback using LQR and Pole Placement

In this case, the same system as in section 4.4.1 is used with the assumption that states x and the angle θ_1 can be measured and the other states of the system are unavailable. The fields under the column *Actual* and buttons on this applet does exactly the same as the previous applet. The fields under the column heading *Obsvr* refers to the predicted system parameters of the observer. Figure 4.5 shows the applet developed.

Figure 4.5: 3-Link inverted pendulum with a fixed base and an observer

CHAPTER V

CONCLUDING REMARKS

The development of the applets is an initial step towards a virtual-control systems laboratory. This can further be improved by developing more control routines. These applets can be used as an educational supplement in standard-control theory classes, differential equation classes, or physics classes. Although the applets concentrate on mechanical systems, the control routines developed can be used to simulate any system. All the applets used in this thesis work in real time, thus correctly portraying the actual dynamics of the system.

It is observed that the ODE solvers used in the applets failed to converge to the actual results in certain situations. This was overcome by executing the ODE solver on every 10ms interval. This can be improved by adopting dynamic ODE solvers that adjust the internal sampling time to ensure the convergence of the results.

If the internet connection is poor, one may experience increased downloading time as for any internet application. If the client computer is a slow machine, the animations may not work smoothly. One way to work around this problem is to export the data to the server and a Java application on the server to collect this data. It then calculates and passes back the feedback gain matrix to the client computer, thus avoiding the need for complex calculations on the client computer.

BIBLIOGRAPHY

- [1] Edited by William S. Levine, *Control System Fundamentals*, CRC Press LLC, Boca Raton, FL, 2000
- [2] Chi-Tsong Chen, *Analog and Digital Control System Design: Transfer Function, State-Space and Algebraic Methods*, Saunders College Publishing, Orlando, FL, 1993
- [3] Stephen P. Boyd and Craig H. Barratt, *Linear Controller Design: Limits of Performance*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1991
- [4] James Conn, *Representation of Control Systems Using Java*, Master's thesis, Texas Tech University, 1998.
- [5] Eduardo D. Sontag, *Mathematical Control Theory*, Springer-Verlag, New York, 1998
- [6] Charles L. Phillips and Royce D. Harbor, *Feedback Control Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, 2000
- [7] Thomas Kailath, *Linear Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1980
- [8] C.B. Moler and G.W. Stewart, "An Algorithm for Generalized Matrix Eigenvalue Problem," *SIAM Journal on Numerical Analysis*, Vol 10, No 2, April 1973, pp 241-256.
- [9] William F. Arnold, III and Alan J. Laub, "Generalized Eigenproblem Algorithm and Software for Algebraic Riccati Equations," *Proceedings of the IEEE*, Vol 72, No 12, December 1984, pp 1746-1754
- [10] J. Kautsky, N.K. Nichols and P. Van Dooren, "Robust Pole assignment in linear state feedback," *International Journal of Control*, Vol 41, No 5, 1985, pp 1129-1155
- [11] P. Van Dooren, "A Generalized Eigenvalue Approach for Solving Riccati Equations," *SIAM Journal of Scientific and Statistical Computing*, Vol 2, No 2, June 1981, pp 121-135
- [12] B.C. Eaves, A.J. Hoffman, U.G. Rothblum and H. Schneider. "Line-sum-symmetric scalings of square nonnegative matrices," *Mathematical Programming Study*, 25 , pp 124-141. 1985.
- [13] E.E. Osborne, "On pre-conditioning of matrices," *JACM*, pp 338-345. 1960.
- [14] Kendall E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, Inc., New York, 1989
- [15] Matlab [software], The MathWorks, Inc., Natick, MA, 1998
- [16] Benjamin C. Kuo, *Automatic Control Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1995

- [17] Jonathan Knudsen, *Java 2D Graphics*, O'Reilly & Associates, Inc., Sebastopol, CA, 1999
- [18] Peter van der Linden, *Just Java 2*, Sun Microsystems, Inc., Palo Alto, CA, 1999
- [19] Laura Lemay and Charles L. Perkins, *Teach Yourself Java in 21 Days*, Sams.net Publishing, Indianapolis, IN, 1996
- [20] W.F. Arnold and Alan J. Laub, "A Software Package for the Solution of Generalized Algebraic Riccati Equations," *Proceedings of the 22nd IEEE Conference on Decision and Control*, San Antonio, TX, pp 415-417, December 1983
- [21] Rajnikant V. Patel and Pradeep Misra, "Numerical Algorithms for Eigenvalue Assignment by State Feedback," *Proceedings of the IEEE*, Vol 72, No 12, December 1984, pp 1755-1764
- [22] Delores M. Etter, C. Richard Johnson, Jr. and Geoffrey C. Orsak, "Experiences Using the Internet and the WWW to Facilitate Remote Teaming of Students and Faculty," *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, p. 3805, 1997
- [23] J.E. Luntz, W.C. Messner and D.M. Tilbury, "Web technology for controls education," *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, pp. 3798-3803, 1997
- [24] Y.C. Ho, W.B. Gong, C.G. Cassandras, J.Q. Hu and P. Vakili, "Experience with developing and dispensing research results and advanced course materials on the World Wide Web of the Internet," *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, p. 3804, 1997

APPENDIX

JAVA ANIMATION ROUTINES

SimpPend.Java file

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;
import javax.swing.*;
import Jama.*; import
java.awt.geom.*;

public class SimpPend extends JApplet implements Runnable {
    Shape mShape1;
    ModifyFunctionPid temp =new ModifyFunctionPid();
    contr controller= new contr();
    int s1v,s2v;
    JTextField tfield1,tfield2,tfield3,tfield4,tfield5;
    Thread runner;
    double ref=45;
    double er,tn,tn_1=0.0;
    double xold,yold;
    double angle=170.0;
    double [] y = new double[] {angle*Math.PI/180,0};
    Vector x =new Vector(y);
    Vector y1 ;
    int xcord,ycord;
    JScrollBar s1 = new JScrollBar(JScrollBar.HORIZONTAL,
    (int)(ref),10,-180,190);
    JScrollBar s2 = new JScrollBar(JScrollBar.HORIZONTAL,0,1,0,100);
    JScrollBar s3 = new JScrollBar(JScrollBar.HORIZONTAL,0,1,0,100);
    JScrollBar s4 = new JScrollBar(JScrollBar.HORIZONTAL,0,1,0,100);
    JPanel p2 = new JPanel();
    double error,inte_gain,errprevious=0.0,sum=0.0;
    JButton startb,endb,resetb;
    Container c;
    sg p1;
    GradientPaint gp1;
    Graphics2D g1;
    JLabel lbl1,lbl2,lbl3,lbl4,lbl5;

    public class sg extends JPanel {

        public void paintComponent(Graphics g)
        {
            super.paintComponent(g);
            g1 = (Graphics2D)g;
            g1.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
            g1.setPaint(gp1);
        }
    }
}
```

```

        g1.setStroke(new BasicStroke(5));
        g1.rotate(-y[0],200,100);
        g1.draw(mShape1);
        g1.rotate(y[0],200,100);
    }

    public void update(Graphics g)
    {
        paint(g);
    }

    public void draw()
    {
        repaint();
    }
}

public void start()
{
    if (runner == null)
    {
        runner = new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner != null)
    {
        runner.stop();
        runner = null;
    }
}

public void run()
{
    while(true)
    {
        tn=tn_1;
        tn_1=tn_1+1;
        controller.integrator_sum = sum;
        error=ref*Math.PI/180-x.get(1);
        temp.u=controller.PidController(error,tn,tn_1);
        controller.error_previous=error;
        sum = controller.integrator_sum;
        y1=temp.runge_kutta_2nd(tn,tn_1,x);
        y=y1.VectorToDouble();
        while(y[0]<-Math.PI)
        { y[0]=2*Math.PI+y[0];}
        while(y[0]>Math.PI)

```

```

        { y[0]=-2*Math.PI+y[0];}
        ycord=(int)(Math.cos(y[0])*50);
        xcord=(int)(Math.sin(y[0])*50);
        er=(ycord-yold)*(ycord-yold)+(xcord-xold)*(xcord-xold);
        if (er>1E-4)
        {
            tfield1.setText(String.valueOf((int)(y[0]*180/Math.PI)));
            p1.draw();
        }
        try { Thread.sleep(50);}
        catch (InterruptedException e) {}
        x=y1;
        xold=xcord;
        yold=ycord;
    }
}

```

```

public void buildConstraints(GridBagConstraints gbc,
int gx, int gy, int gw, int gh, int wx, int wy)
{
    gbc.gridx=gx;
    gbc.gridy=gy;
    gbc.gridwidth=gw;
    gbc.gridheight=gh;
    gbc.weightx=wx;
    gbc.weighty=wy;
}

```

```

public void init()
{
    p1=new sg();
    c=getContentPane();
    c.setLayout(new GridLayout(2,1));

    Scradj sa = new Scradj();
    s1.addAdjustmentListener(sa);
    s2.addAdjustmentListener(sa);
    s3.addAdjustmentListener(sa);
    s4.addAdjustmentListener(sa);

    s1.setUnitIncrement(10);
    s1.setBlockIncrement(10);
    s2.setUnitIncrement(1);
    s2.setBlockIncrement(5);
    s3.setUnitIncrement(1);
    s3.setBlockIncrement(5);
    s4.setUnitIncrement(1);
    s4.setBlockIncrement(5);

    startb = new JButton("start");
    endb = new JButton("stop");
    resetb = new JButton("Reset");
}

```

```

ButtonAction bact = new ButtonAction();
startb.addActionListener(bact);
endb.addActionListener(bact);
resetb.addActionListener(bact);

p2.setLayout(new GridLayout(1,3,35,1));
p2.add(startb);
p2.add(endb);
p2.add(resetb);

tfield1 = new JTextField("0.0000");
tfield2 = new JTextField(String.valueOf((int)(ref))+" ");
tfield3 = new JTextField("0.0000");
tfield4 = new JTextField("0.0000");
tfield5 = new JTextField("0.0000");

GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
c.setLayout(gridbag);
// row 1
buildConstraints(constraints, 0,0,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
lbl1=new JLabel("Ref",JLabel.LEFT);
gridbag.setConstraints(lbl1,constraints);
c.add(lbl1);
buildConstraints(constraints, 1,0,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(tfield2,constraints);
c.add(tfield2);
buildConstraints(constraints, 2,0,1,1,98,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s1,constraints);
c.add(s1);

// row 2
buildConstraints(constraints, 0,1,1,1,0,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
lbl2=new JLabel("Prop",JLabel.LEFT);
gridbag.setConstraints(lbl2,constraints);
c.add(lbl2);
buildConstraints(constraints, 1,1,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(tfield3,constraints);
c.add(tfield3);
buildConstraints(constraints, 2,1,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s2,constraints);
c.add(s2);

// row 3
buildConstraints(constraints, 0,2,1,1,0,1);
constraints.fill=GridBagConstraints.HORIZONTAL;

```

```

lbl3=new JLabel("Inte",JLabel.LEFT);
gridbag.setConstraints(lbl3,constraints);
c.add(lbl3);
buildConstraints(constraints, 1,2,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(tfield4,constraints);
c.add(tfield4);
buildConstraints(constraints, 2,2,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s3,constraints);
c.add(s3);

// row 4
buildConstraints(constraints, 0,3,1,1,0,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
lbl4=new JLabel("Deri",JLabel.LEFT);
gridbag.setConstraints(lbl4,constraints);
c.add(lbl4);
buildConstraints(constraints, 1,3,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(tfield5,constraints);
c.add(tfield5);
buildConstraints(constraints, 2,3,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s4,constraints);
c.add(s4);

// row 5
buildConstraints(constraints, 0,4,1,1,0,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
lbl5=new JLabel("Angle",JLabel.LEFT);
gridbag.setConstraints(lbl5,constraints);
c.add(lbl5);
buildConstraints(constraints, 1,4,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(tfield1,constraints);
c.add(tfield1);
buildConstraints(constraints, 2,4,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(p2,constraints);
c.add(p2);

// row 6
buildConstraints(constraints, 1,5,4,4,0,95);
constraints.fill=GridBagConstraints.BOTH;
gridbag.setConstraints(p1,constraints);
c.add(p1);

temp.c=0.01;
controller.error_previous=ref-x.get(1);
yold=(int)(Math.cos(y[0])*50);
xold=(int)(Math.sin(y[0])*50);

```

```

        mShape1 = new Line2D.Double(200,100,200,200);
        gp1=new GradientPaint(200,100,Color.green,200,0,
        Color.gray,true);
    }

class Scradj implements AdjustmentListener
{
    public void adjustmentValueChanged(AdjustmentEvent ae)
    {
        JScrollBar s = (JScrollBar) ae.getAdjustable();
        if (s==s1)
        {
            ref = ae.getValue();
            tfield2.setText(String.valueOf((int)(ref)));
        }
        else if (s==s2)
        {
            controller.prop_gain = ae.getValue()/1000.0;
            tfield3.setText(String.valueOf((int)
            (controller.prop_gain*1000))+ "E-3");
        }
        else if (s==s3)
        {
            controller.inte_gain = ae.getValue()/100000.0;
            tfield4.setText(String.valueOf((int)
            (controller.inte_gain*100000))+ "E-5");
        }
        else
        {
            controller.diff_gain = ae.getValue()/10.0;
            tfield5.setText(String.valueOf((int)
            (controller.diff_gain*10))+ "E-1");
        }
    }
}

class ButtonAction implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() instanceof JButton)
        {
            if (e.getSource() == startb)
            {
                start();
            }
            if (e.getSource() == endb)
            {
                stop();
            }
        }
    }
}

```



```

    }
    if (e.getSource() == resetb)
    {
        temp.u=0.0;
        angle=Double.parseDouble(tfield1.getText());
        x=new Vector(new double[] {angle*Math.PI/180,0});
        error=0.0;
        tn_1=0.0;
        controller.error_previous=0.0;
        sum=0.0;
    }
}
}}

```

wedge.java file

```

import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;
import javax.swing.*;
import Jama.*;
import java.awt.geom.*;
import Jampack.*;

public class wedge extends JApplet
implements Runnable
{
    Matrix A,B,Q,R,K;
    ZVector PoleVec;
    Shape mShape1,mShape2;
    double m1=19.0,m2=15.0,M=10.0,sample_width;
    double alfa=1.1,theta=-0.06,x=0.01,x0;
    double tn,tn_1=System.currentTimeMillis()/1000.0,g=10;
    double c,L=2,denominator;
    int tme=100,samples;
    JTextField mtfield,m2tfield,Mtfield,thetafield,xtfield;
    Thread runner;
    // initial conditions - vector{ thetadot, theta, xdot, x }
    Vector y=new Vector(new double[] {0,theta,0,x});
    ModifyFunction Func=new ModifyFunction(this);
    lqr LQR; // lqr algorithm
    place Plc; // Pole placement
    GradientPaint gp1,gp2;
    Container cn;
    Graphics2D offscreenG; // Graphic 2D0 screen

    // 5 scroll bars to get the user input
    JScrollBar s1 = new JScrollBar(JScrollBar.HORIZONTAL,
(int)m1,1,0,100);

```

```

JScrollBar s2 = new JScrollBar(JScrollBar.HORIZONTAL,
(int)m2,1,0,100);
JScrollBar s3 = new JScrollBar(JScrollBar.HORIZONTAL,
(int)M,1,0,100);
JScrollBar s4 = new JScrollBar(JScrollBar.HORIZONTAL,
(int)(theta*10),1,-10,10);
JScrollBar s5 = new JScrollBar(JScrollBar.HORIZONTAL,
(int)x,1,-20,20);

//p2 for JButtons ,p3 for Q matrix,
//p4 for radio JButtons, p5 for Poles
JPanel p2 = new JPanel();
JPanel p3 = new JPanel();
JPanel p5 = new JPanel();
JButton contb,simulateb,pauseb,getRb,getQb,helpb,poleb;
JLabel m1lbl,m2lbl,Mlbl,thetalbl,xlbl;
JFrame windowQ>windowR>windowP;
JTextField [][] txt=new JTextField[4][4];
JTextField [][] txtPole=new JTextField[4][2];
JTextField txtR;
windhand mwh=new windhand(this);
boolean getRpressed=true,getQpressed=true,getPpressed=true;
boolean SelectPole=true,Periodic=false,animate=false;
boolean SelectLQR=false;
ButtonGroup cbg=new ButtonGroup();
JRadioButton c1,c2;
JCheckBox c3;
DrawGraphics dg;

// initial values for Poles, Q and R matrices
double[] val2={-1,-2,-3,-4};
double[] val3={10,14,10,18};
double[] val4={1};

public class DrawGraphics extends JPanel{

    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        offscreenG = (Graphics2D)g;
        offscreenG.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
        offscreenG.setColor(Color.blue);
        offscreenG.setStroke(new BasicStroke(1,BasicStroke.CAP_BUTT,
BasicStroke.JOIN_BEVEL,0,new float[]{5,3},0));
        offscreenG.drawLine(200,10,200,190);
        offscreenG.translate(200,210);
        offscreenG.rotate((theta+alfa/2),0,0);
        offscreenG.setStroke(new BasicStroke(1));
        offscreenG.setColor(Color.black);
        offscreenG.drawLine(-8,-13,-8,(int)(-(x+x0)*100));
        offscreenG.setPaint(gp1);
    }
}

```

```

        offscreenG.setStroke(new BasicStroke(5));
        offscreenG.draw(mShape1);
        offscreenG.translate(-10,-(x+x0)*100);
        offscreenG.setPaint(gp2);
        offscreenG.draw(mShape2);
        offscreenG.translate(10,(x+x0)*100);
        offscreenG.rotate(-alfa,0,0);
        offscreenG.setPaint(gp1);
        offscreenG.draw(mShape1);
        offscreenG.setStroke(new BasicStroke(1));
        offscreenG.setColor(Color.black);
        offscreenG.drawLine(8,-13,8,(int)(-(L-x-x0)*100));
        offscreenG.translate(5,-(L-x-x0)*100);
        offscreenG.setStroke(new BasicStroke(5));
        offscreenG.setPaint(gp2);
        offscreenG.draw(mShape2);
    }

    public void update(Graphics g)
    {
        paint(g);
    }

    public Dimension getPreferredSize()
    {
        return new Dimension(400,200);
    }

    public void draw()
    {
        repaint();
    }
}

public wedge()
{
    initComponents();
}

public void start()
{
    if (runner == null)
    {
        runner = new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner != null)

```

```

    {
        runner.stop();
        runner = null;
    }
}

public void run()
{
    while(true)
    {
        tn=tn_1;
        tn_1=System.currentTimeMillis()/1000.0;
        try { Thread.sleep(tme);}
        catch (InterruptedException e) {}
        if (animate)
        {
            samples=(int)(100*(tn_1-tn)); // sampling the time
            if (samples>1)
            {
                sample_width=(tn_1-tn)/samples;

                // for higher accuracy, runge-kutta method is evaluated at
                // many sample intervals
                for (int j=0;j<samples;j++)
                y=Func.runge_kutta_4th(tn+j*sample_width,tn+(j+1)
                *sample_width,y);
                if (y.get(4)>0.88)
                {
                    // preventing ball going off the wedge
                    y.set(4,0.88);
                    // preventing positive accelerations
                    if (y.get(3)>0.0)
                        y.set(3,0.0);
                }
                if (y.get(4)<-0.65)
                {
                    // preventing ball from hiting wedge corner
                    y.set(4,-0.65);
                    // preventing negative accelerations
                    if (y.get(1)<0.0)
                        y.set(1,0.0);
                }
                theta=y.get(2);
                sahdkjshskjdghlksjdghlksfdhgslkjflhdgss
                x=y.get(4);
                if (Math.abs(y.get(1))>0.0009||
                Math.abs(theta)>0.0009||Math.abs(x)>0.0009||
                Math.abs(y.get(3))>0.0009)
                {
                    // small values are set to display "0.00"
                    if (Math.abs(theta)>0.0009)
                        thetafield.setText(String.valueOf(theta));
                }
            }
        }
    }
}

```

```

        else
            thetafield.setText("0.00");
            if (Math.abs(x)>0.001)
                xtfield.setText(String.valueOf(x));
            else
                xtfield.setText("0.00");
            if (Math.abs(theta)>alfa/2+.1)
            {
                JOptionPane.showMessageDialog(null,
                    "System uncontrollable !");
                animate=false;
            } else
                dg.draw();
        }
    }
}

public void buildConstraints(GridBagConstraints gbc, int gx,
    int gy, int gw, int gh, int wx, int wy)
{
    gbc.gridx=gx;
    gbc.gridy=gy;
    gbc.gridwidth=gw;
    gbc.gridheight=gh;
    gbc.weightx=wx;
    gbc.weighty=wy;
}

public void initComponents()
{
    dg=new DrawGraphics();
    cn=getContentPane();

    x0=m2*L/(m1+m2);           // calculating stable position
    R=EleMat.diag(new Vector(val4));
    Q=EleMat.diag(new Vector(val3));
    PoleVec=new ZVector(val2);

    c=(L/2)*Math.cos(alfa/2); // calculating the center of gravity
    Scradj sa = new Scradj();
    s1.addAdjustmentListener(sa);
    s2.addAdjustmentListener(sa);
    s3.addAdjustmentListener(sa);
    s4.addAdjustmentListener(sa);
    s5.addAdjustmentListener(sa);

    poleb=new JButton("Poles");
    getQb=new JButton("Get Q");
    helpb=new JButton("Help");
    getRb=new JButton("Get R");
}

```

```

contb = new JButton("Continue");
simulateb = new JButton ("Simulate");
pauseb = new JButton("Pause");
JButtonAction bact = new JButtonAction();
poleb.addActionListener(bact);
contb.addActionListener(bact);
simulateb.addActionListener(bact);
pauseb.addActionListener(bact);
getQb.addActionListener(bact);
getRb.addActionListener(bact);
p2.setLayout(new GridLayout(9,1,0,0));
p2.add(poleb);
p2.add(getRb);
p2.add(getQb);
p2.add(simulateb);
p2.add(pauseb);
p2.add(contb);
// p2.add(helpb);

p3.setLayout(new GridLayout(txt.length,txt[0].length,5,5));
p5.setLayout(new GridLayout(txt.length,2,5,5));
for (int i=0;i<txt.length;i++)
{
    for (int j=0;j<txt[0].length;j++)
    {
        txt[i][j]=new JTextField("0.00");
        p3.add(txt[i][j]);
        if (j<2)
        {
            txtPole[i][j]=new JTextField("0.00");
            p5.add(txtPole[i][j]);
        }
    }
    txtPole[i][0].setText(String.valueOf(val2[i]));
    txt[i][i].setText(String.valueOf(val3[i]));
}

itemListen it=new itemListen(this);
c1=new JRadioButton("LQR",false);
c2=new JRadioButton("Pole",true);
c3=new JCheckBox("Sin");
cbg.add(c1);
cbg.add(c2);
c1.addItemListener(it);
c2.addItemListener(it);
c3.addItemListener(it);
p2.add(c1);
p2.add(c2);
p2.add(c3);

windowQ=new JFrame("Enter Q Matrix");
windowQ.setSize(200,200);

```

```

windowQ.getContentPane().add(p3);
windowQ.addWindowListener(mwh);
windowR=new JFrame("R Matrix");
windowR.setSize(50,100);
txtR=new JTextField("0.00");
windowR.getContentPane().add(txtR);
txtR.setText(String.valueOf(val4[0]));
windowR.addWindowListener(mwh);
windowP=new JFrame("Poles");
windowP.setSize(100,150);
windowP.getContentPane().add(p5);
windowP.addWindowListener(mwh);

GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
cn.setLayout(gridbag);
s1.setUnitIncrement(10);
s1.setBlockIncrement(10);
s2.setUnitIncrement(1);
s2.setBlockIncrement(5);
s3.setUnitIncrement(1);
s3.setBlockIncrement(5);
s4.setUnitIncrement(1);
s4.setBlockIncrement(5);
s5.setUnitIncrement(1);
s5.setBlockIncrement(5);
m1tfield = new JTextField("000");
m2tfield = new JTextField("000");
Mtfield = new JTextField("000");
thetafield = new JTextField("0.00");
xtfield = new JTextField("0.00");
m1tfield.setEditable(false);
m2tfield.setEditable(false);
Mtfield.setEditable(false);
thetafield.setEditable(false);
xtfield.setEditable(false);

// row 1
buildConstraints(constraints, 0,0,1,1,-5,1);
constraints.fill=GridBagConstraints.NONE;
m1lbl=new JLabel("m1",JLabel.LEFT);
gridbag.setConstraints(m1lbl,constraints);
cn.add(m1lbl);
buildConstraints(constraints, 1,0,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(m1tfield,constraints);
cn.add(m1tfield);
buildConstraints(constraints, 2,0,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s1,constraints);
cn.add(s1);

```

```

// row 2
buildConstraints(constraints, 0,1,1,1,0,1);
constraints.fill=GridBagConstraints.NONE;
m2lbl=new JLabel("m2",JLabel.LEFT);
gridbag.setConstraints(m2lbl,constraints);
cn.add(m2lbl);
buildConstraints(constraints, 1,1,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(m2tfield,constraints);
cn.add(m2tfield);
buildConstraints(constraints, 2,1,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s2,constraints);
cn.add(s2);

// row 3
buildConstraints(constraints, 0,2,1,1,0,1);
constraints.fill=GridBagConstraints.NONE;
Mlbl=new JLabel("M",JLabel.LEFT);
gridbag.setConstraints(Mlbl,constraints);
cn.add(Mlbl);
buildConstraints(constraints, 1,2,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(Mtfield,constraints);
cn.add(Mtfield);
buildConstraints(constraints, 2,2,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s3,constraints);
cn.add(s3);

// row 4
buildConstraints(constraints, 0,3,1,1,0,1);
constraints.fill=GridBagConstraints.NONE;
thetalbl=new JLabel(" Theta ",JLabel.LEFT);
gridbag.setConstraints(thetalbl,constraints);
cn.add(thetalbl);
buildConstraints(constraints, 1,3,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(thetafield,constraints);
cn.add(thetafield);
buildConstraints(constraints, 2,3,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s4,constraints);
cn.add(s4);

// row 5
buildConstraints(constraints, 0,4,1,1,0,1);
constraints.fill=GridBagConstraints.NONE;
xlbl=new JLabel("x",JLabel.LEFT);
gridbag.setConstraints(xlbl,constraints);
cn.add(xlbl);

```



```

buildConstraints(constraints, 1,4,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(xtfield,constraints);
cn.add(xtfield);
buildConstraints(constraints, 2,4,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(s5,constraints);
cn.add(s5);

// row 6
buildConstraints(constraints, 0,5,1,1,0,0);
constraints.fill=GridBagConstraints.NONE;
gridbag.setConstraints(p2,constraints);
cn.add(p2);

// row 7
buildConstraints(constraints, 1,5,4,4,0,95);
constraints.fill=GridBagConstraints.BOTH;
gridbag.setConstraints(dg,constraints);
cn.add(dg);
gp1=new GradientPaint(0,-170,Color.green,0,0,Color.gray,true);
gp2=new GradientPaint(0,-170,Color.red,0,115,Color.gray,true);
mShape1 = new Line2D.Double(0,-200,0,0);
mShape2 =new Ellipse2D.Double(0,0,5,5);
m1tfield.setText(String.valueOf(m1));
m2tfield.setText(String.valueOf(m2));
Mtfield.setText(String.valueOf(M));
xtfield.setText(String.valueOf(x));
thetafield.setText(String.valueOf(theta));
}

```

```

class Scradj implements AdjustmentListener
{
    public void adjustmentValueChanged(AdjustmentEvent ae)
    {
        JScrollBar s = (JScrollBar) ae.getAdjustable();
        if (!animate)
        {
            if (s==s1)
            {
                m1 = ae.getValue();
                m1tfield.setText(String.valueOf(m1));
            }
            else if (s==s2)
            {
                m2 = ae.getValue();
                m2tfield.setText(String.valueOf(m2));
            }
            else if (s==s3)
            {
                M = ae.getValue();
                Mtfield.setText(String.valueOf(M));
            }
        }
    }
}

```

```

    }
    else if (s==s4)
    {
        animate=true;
        theta = ae.getValue()/100.0;
        thetfield.setText(String.valueOf(theta));
        dg.draw();
        animate=false;
    }
    else
    {
        animate=true;
        x = ae.getValue()/100.0;
        xtfield.setText(String.valueOf(x));
        dg.draw();
        animate=false;
    }
}
}
}

```

```

class JButtonAction implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() instanceof JButton)
        {
            if (e.getSource() == simulateb)
            {
                stop();
                animate=false;

                // disabling buttons
                getRb.setEnabled(false);
                getQb.setEnabled(false);
                poleb.setEnabled(false);
                contb.setEnabled(false);
                pauseb.setEnabled(true);
                simulateb.setEnabled(false);
                s1.setEnabled(false);
                s2.setEnabled(false);
                s3.setEnabled(false);
                s4.setEnabled(false);
                s5.setEnabled(false);
                c1.setEnabled(false);
                c2.setEnabled(false);

                //calculating matrices
                x0=m2*L/(m1+m2);
                denominator=m1*L*x0+M*c*c;
                A=new Matrix(4,4);
            }
        }
    }
}

```

```

B=new Matrix(4,1);
A.set(0,1,(2*m1*x0*g*Math.cos(alfa/2)+M*g*c)/denominator);
A.set(0,3,(m1+m2)*g*Math.sin(alfa/2)/denominator);
A.set(1,0,1);
A.set(2,1,g*Math.sin(alfa/2));
A.set(3,2,1);
B.set(2,0,-1/(m1+m2));
if (SelectLQR)
{
    LQR=new lqr(A,B,Q,R);
    K=LQR.G;
} else if (SelectPole)
{
    Plc=new place(A,B,PoleVec);
    K=Plc.K;
}
Func.a=A.minus(B.times(K));
y=new Vector(new double[] {0,theta,0,x});
animate=true;
tn_1=System.currentTimeMillis()/1000.0;
start();
}

if (e.getSource() == pauseb)
{
    animate=false;
    getRb.setEnabled(true);
    getQb.setEnabled(true);
    poleb.setEnabled(true);
    contb.setEnabled(true);
    pauseb.setEnabled(false);
    simulateb.setEnabled(true);
    s1.setEnabled(true);
    s2.setEnabled(true);
    s3.setEnabled(true);
    s4.setEnabled(true);
    s5.setEnabled(true);
    c1.setEnabled(true);
    c2.setEnabled(true);
    tn_1=System.currentTimeMillis()/1000.0;
    stop();
}

if (e.getSource() == getRb)
{
    stop();
    simulateb.setEnabled(false);
    getRb.setEnabled(false);
    animate=false;
    getQpressed=false;
    getRpressed=true;
    getPpressed=false;
}

```

```

        windowR.setVisible(true);
    }

    if (e.getSource() == poleb)
    {
        stop();
        simulateb.setEnabled(false);
        poleb.setEnabled(false);
        animate=false;
        getQpressed=false;
        getRpressed=false;
        getPpressed=true;
        windowP.setVisible(true);
    }
    if (e.getSource() == getQb)
    {
        stop();
        simulateb.setEnabled(false);
        getQb.setEnabled(false);
        animate=false;
        getQpressed=true;
        getRpressed=false;
        getPpressed=false;
        windowQ.setVisible(true);
    }
    if (e.getSource() == contb)
    {
        getRb.setEnabled(false);
        getQb.setEnabled(false);
        poleb.setEnabled(false);
        contb.setEnabled(false);
        pauseb.setEnabled(true);
        simulateb.setEnabled(false);
        s1.setEnabled(false);
        s2.setEnabled(false);
        s3.setEnabled(false);
        s4.setEnabled(false);
        s5.setEnabled(false);
        c1.setEnabled(false);
        c2.setEnabled(false);
        tn_1=System.currentTimeMillis()/1000.0;
        animate=true;
        start();
    }
    if (e.getSource() == helpb)
    {}
}
}
}
}
}

```

pend3l.java file

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;
import javax.swing.*;
import Jama.*;
import java.awt.geom.*;
import Jampack.*;

public class pend3l extends JApplet
implements Runnable
{
    int Nlinks=3,samples,tme=200;
    Matrix A,B,Q,R,K,Mass;
    ZVector PoleVec;
    Shape[] mShape=new Shape[Nlinks];
    double[] m={1.2,1.0,.8},lengths={1.4,1.0,.6},theta={0.1,-0.2,.1};
    double tn,tn_1=System.currentTimeMillis()/1000.0,x_origin;
    double y_origin,g=10.0,sample_width;
    JTextField[] mtfield=new JTextField[Nlinks];
    JTextField[] thetafield=new JTextField[Nlinks];
    Thread runner;
    double [] x1 = new double[] {0,0,0,theta[0],theta[1],theta[2]};
    private Vector y=new Vector(x1);
    ModifyFunction3l Func=new ModifyFunction3l();
    itemListen3l it;
    lqr LQR;
    place Plc;
    GradientPaint gp1,gp2;
    Graphics2D offscreenG;
    Stroke stroke;
    JPanel p3 = new JPanel();
    JPanel p5 = new JPanel();
    JButton contb,simulateb,pauseb,getRb,getQb,helpb,poleb;
    JLabel Mlbl,xlbl;
    JLabel[] mlbl=new JLabel[Nlinks],thetalbl=new JLabel[Nlinks];
    JFrame windowQ>windowR>windowP;
    TextField [][] txt=new TextField[6][6],txtPole=new TextField[6][2];
    TextField txtR;
    windhand3l mwh=new windhand3l(this);
    boolean getRpressed=true,getQpressed=true,getPpressed=true;
    boolean Periodic=false,SelectPole=true,animate=false;
    boolean SelectLQR=false;
    ButtonGroup cbg=new ButtonGroup();
    JRadioButton c1,c2;
    GraphicPanel gp;
    Container cn;
    double[] val2={-1,-2,-3,-4,-5,-6};
    double[] val3={1,1,1,1,1,1};
    double[] val4={1};
```

```

public class GraphicPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        offscreenG = (Graphics2D)g;
        offscreenG.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
        offscreenG.setColor(Color.blue);
        offscreenG.setStroke(new BasicStroke(1,BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_BEVEL,0,new float[]{5,3},0));
        offscreenG.drawLine((int)x_origin,0,(int)x_origin,
        (int)y_origin);
        offscreenG.setStroke(stroke);
        for (int i=0;i<Nlinks;i++)
        {
            offscreenG.rotate(-theta[i],x_origin,y_origin);
            offscreenG.setPaint(i%2==0?gp1:gp2);
            offscreenG.draw(mShape[i]);
            offscreenG.translate(0,-lengths[i]*100);
            offscreenG.rotate(theta[i],x_origin,y_origin);
        }
    }

    public void update(Graphics g)
    {
        paint(g);
    }

    public void draw()
    {
        repaint();
    }
}

public pend3l()
{
    initComponents();
}

public void start()
{
    if (runner == null)
    {
        runner = new Thread(this);
        runner.start();
    }
}

public void stop()
{

```

```

        if (runner != null)
        {
            runner.stop();
            runner = null;
        }
    }

    public void run()
    {
        while(true)
        {
            tn=tn_1;
            tn_1=System.currentTimeMillis()/1000.0;
            try { Thread.sleep(tme);}
            catch (InterruptedException e) {}
            if (animate)
            {
                samples=(int)(100*(tn_1-tn));
                if (samples>1)
                {
                    sample_width=(tn_1-tn)/samples;
                    for (int j=0;j<samples;j++)
                    {
                        y=Func.runge_kutta_4th(tn+j*sample_width,tn+(j+1)*
                            sample_width,y);
                        theta[0]=y.get(4);
                        theta[1]=y.get(5);
                        theta[2]=y.get(6);
                        for (int i=0;i<Nlinks;i++)
                        if (Math.abs(theta[i])>0.0009)
                            thetfield[i].setText(String.valueOf(theta[i]*
                                180/Math.PI));
                        else
                            thetfield[i].setText("0.00");
                    }
                    gp.draw();
                }
            }
        }
    }

    public void buildConstraints(GridBagConstraints gbc, int gx,
    int gy, int gw, int gh, int wx, int wy)
    {
        gbc.gridx=gx;
        gbc.gridy=gy;
        gbc.gridwidth=gw;
        gbc.gridheight=gh;
        gbc.weightx=wx;
        gbc.weighty=wy;
    }

    public void initComponents()
    {

```

```

gp=new GraphicPanel();
cn=getContentPane();

R=EleMat.diag(new Vector(val4));
Q=EleMat.diag(new Vector(val3));
PoleVec=new ZVector(val2);

poleb      = new JButton("Poles");
getQb      = new JButton("Get Q");
helpb      = new JButton("Help");
getRb      = new JButton("Get R");
contb      = new JButton("Continue");
simulateb  = new JButton("Simulate");
pauseb     = new JButton("Pause");

ButtonAction bact = new ButtonAction();
poleb.addActionListener(bact);
contb.addActionListener(bact);
simulateb.addActionListener(bact);
pauseb.addActionListener(bact);
getQb.addActionListener(bact);
getRb.addActionListener(bact);

it=new itemListen3l(this);
c1=new JRadioButton("LQR",false);
c2=new JRadioButton("Pole",true);
cbg.add(c1);
cbg.add(c2);
c1.addItemListener(it);
c2.addItemListener(it);

p3.setLayout(new GridLayout(txt.length,txt[0].length+1,5,5));
p5.setLayout(new GridLayout(txt.length,2,5,5));
for (int i=0;i<txt.length;i++)
{
    for (int j=0;j<txt[0].length;j++)
    {
        txt[i][j]=new TextField("0.00");
        p3.add(txt[i][j]);
        if (j<2)
        {
            txtPole[i][j]=new TextField("0.00");
            p5.add(txtPole[i][j]);
        }
    }
    txtPole[i][0].setText(String.valueOf(val2[i]));
    txt[i][i].setText(String.valueOf(val3[i]));
}

windowQ=new JFrame("Enter Q Matrix");
windowQ.setSize(50*txt[0].length,40*txt.length);
windowQ.getContentPane().add(p3);

```



```

windowQ.addWindowListener(mwh);
windowR=new JFrame("R Matrix");
windowR.setSize(50,100);
txtR=new TextField("0.00");
windowR.getContentPane().add(txtR);
txtR.setText(String.valueOf(val4[0]));
windowR.addWindowListener(mwh);
windowP=new JFrame("Poles");
windowP.setSize(100,40*txt.length);
windowP.getContentPane().add(p5);
windowP.addWindowListener(mwh);

GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
cn.setLayout(gridbag);

x_origin=150;
y_origin=380;

// Column 1 & Column 2
for (int i=1;i<Nlinks+1;i++)
{
    buildConstraints(constraints, 0,Nlinks-i,1,1,1,1);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    mlbl[i-1]=new JLabel("m"+i+"(kg)",JLabel.LEFT);
    gridbag.setConstraints(mlbl[i-1],constraints);
    cn.add(mlbl[i-1]);

    mtfield[i-1] = new JTextField("000");
    buildConstraints(constraints, 1,Nlinks-i,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    gridbag.setConstraints(mtfield[i-1],constraints);
    cn.add(mtfield[i-1]);
    mtfield[i-1].setText(String.valueOf(m[i-1]));

    buildConstraints(constraints, 0,2*Nlinks-i,1,1,1,1);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    thetalbl[i-1]=new JLabel("theta"+i+"(deg)",JLabel.LEFT);
    gridbag.setConstraints(thetalbl[i-1],constraints);
    cn.add(thetalbl[i-1]);

    thetafield[i-1] = new JTextField("000");
    buildConstraints(constraints, 1,2*Nlinks-i,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    gridbag.setConstraints(thetafield[i-1],constraints);
    cn.add(thetafield[i-1]);
    thetafield[i-1].setText(String.valueOf(Math.round(theta[i-1]*
18000/Math.PI)/100.0));
    mShape[i-1] = new Line2D.Double(x_origin,y_origin,x_origin,
y_origin-lengths[i-1]*100);
}

```

```

// Column 1

buildConstraints(constraints, 0,8,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(simulateb,constraints);
cn.add(simulateb);
buildConstraints(constraints, 0,9,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(pauseb,constraints);
cn.add(pauseb);
buildConstraints(constraints,0,10,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(contb,constraints);
cn.add(contb);
buildConstraints(constraints,0,11,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(helpb,constraints);
cn.add(helpb);
buildConstraints(constraints,0,12,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(c1,constraints);
cn.add(c1);

//Column 2
buildConstraints(constraints, 1,8,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getRb,constraints);
cn.add(getRb);
buildConstraints(constraints, 1,9,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getQb,constraints);
cn.add(getQb);
buildConstraints(constraints,1,10,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(poleb,constraints);
cn.add(poleb);
buildConstraints(constraints,1,12,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(c2,constraints);
cn.add(c2);

// Column 3
buildConstraints(constraints, 2,0,1,13,95,0);
constraints.fill=GridBagConstraints.BOTH;
gridbag.setConstraints(gp,constraints);
cn.add(gp);
gp1=new GradientPaint(225,115,Color.cyan,225,15,
Color.gray,true);
gp2=new GradientPaint(225,115,Color.red,225,15,Color.gray,true);
stroke=new BasicStroke(6,BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND);
contb.setEnabled(false);

```

```

    pauseb.setEnabled(false);
}

class ButtonAction implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() instanceof JButton)
    {
        if (e.getSource() == simulateb)
        {
            animate=false;
            stop();

            // disabling buttons
            getRb.setEnabled(false);
            getQb.setEnabled(false);
            poleb.setEnabled(false);
            contb.setEnabled(false);
            pauseb.setEnabled(true);
            simulateb.setEnabled(false);
            c1.setEnabled(false);
            c2.setEnabled(false);

            // read user input
            for (int i=0;i<Nlinks;i++)
            {
                m[i]=Double.parseDouble(mtfld[i].getText());
                theta[i]=Double.parseDouble(
                    thetafld[i].getText())*Math.PI/180.0;
                mtfld[i].setEditable(false);
                thetafld[i].setEditable(false);
            }
            gp.draw();

            //calculate the matrices
            Mass=new Matrix(3,3);
            B=new Matrix(3,1);
            B.set(1,0,-1);
            B.set(2,0,1);
            Mass.set(0,1,2*(m[1]+2*m[2])*lengths[0]*lengths[1]);
            Mass.set(0,2,2*m[2]*lengths[0]*lengths[2]);
            Mass.set(1,2,2*m[2]*lengths[1]*lengths[2]);
            Mass=Mass.plus(Mass.transpose());
            Mass.set(0,0,(m[0]+2*m[1]+2*m[2])*lengths[0]*lengths[0]);
            Mass.set(1,1,(m[1]+2*m[2])*lengths[1]*lengths[1]);
            Mass.set(2,2,m[2]*lengths[2]*lengths[2]);
            Mass=Mass.inverse();
            B=Mass.times(B);
            A=new Matrix(3,3);
            A.set(0,0,(m[0]+2*m[1]+2*m[2])*g*lengths[0]);
            A.set(1,1,(m[1]+2*m[2])*g*lengths[1]);

```

```

        A.set(2,2,m[2]*g*lengths[2]);
        A=Mass.times(A);
        Mass=new Matrix(6,6);
        Mass.setMatrix(0,2,3,5,A);
        Mass.setMatrix(3,5,0,2,Matrix.identity(3,3));
        A=Mass;
        Mass=new Matrix(6,1);
        Mass.setMatrix(0,2,0,0,B);
        B=Mass;
        if (SelectLQR)
        {
            LQR=new lqr(A,B,Q,R);
            K=LQR.G;
        } else if (SelectPole)
        {
            Plc=new place(A,B,PoleVec);
            K=Plc.K;
        }
        Func.A=A.minus(B.times(K));
        y=new Vector(new double[] {0,0,0,theta[0],
        theta[1],theta[2]});
        animate=true;
        tn_1=System.currentTimeMillis()/1000.0;
        start();
    }

    if (e.getSource() == pauseb)
    {
        animate=false;
        stop();
        getRb.setEnabled(true);
        getQb.setEnabled(true);
        poleb.setEnabled(true);
        contb.setEnabled(true);
        pauseb.setEnabled(false);
        simulateb.setEnabled(true);
        c1.setEnabled(true);
        c2.setEnabled(true);
        for (int i=0;i<Nlinks;i++)
        {
            mtfield[i].setEditable(true);
            thetafield[i].setEditable(true);
        }
    }

    if (e.getSource() == getRb)
    {
        animate=false;
        stop();
        simulateb.setEnabled(false);
        getRb.setEnabled(false);
        getQpressed=false;
    }

```

```

        getRpressed=true;
        getPpressed=false;
        windowR.setVisible(true);
    }

    if (e.getSource() == poleb)
    {
        animate=false;
        stop();
        simulateb.setEnabled(false);
        poleb.setEnabled(false);
        getQpressed=false;
        getRpressed=false;
        getPpressed=true;
        windowP.setVisible(true);
    }
    if (e.getSource() == getQb)
    {
        animate=false;
        stop();
        simulateb.setEnabled(false);
        getQb.setEnabled(false);
        getQpressed=true;
        getRpressed=false;
        getPpressed=false;
        windowQ.setVisible(true);
    }
    if (e.getSource() == contb)
    {
        getRb.setEnabled(false);
        getQb.setEnabled(false);
        poleb.setEnabled(false);
        contb.setEnabled(false);
        pauseb.setEnabled(true);
        simulateb.setEnabled(false);
        c1.setEnabled(false);
        c2.setEnabled(false);
        for (int i=0;i<Nlinks;i++)
        {
            mtfield[i].setEditable(false);
            thetafield[i].setEditable(false);
        }
        animate=true;
        tn_1=System.currentTimeMillis()/1000.0;
        start();
    }
    if (e.getSource() == helpb)
    {}
}
}
}
}
}

```

pend3b1.java file

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;
import javax.swing.*;
import Jama.*;
import java.awt.geom.*;
import Jampack.*;

public class pend3b1 extends JApplet
implements Runnable
{
    String Kstring;
    int Nlinks=3,samples,scale,refreshing_time=100;
    double g=10.0,sample_width;
    Matrix A,B,Q,R,K,Mass;
    ZVector PoleVec;
    Shape[] mShape=new Shape[Nlinks];
    Shape cart;
    double M=5.0,x=0.1;
    double[] m={1.2,1.0,.8},lengths={0.7,0.5,0.3};
    double theta={0.12,-0.25,0.3},y_origin;
    double tn,tn_1=System.currentTimeMillis()/1000.0,x_origin;
    JTextField Mtfield,xtfield;
    JTextField[] mtfield=new JTextField[Nlinks];
    JTextField[] thetafield=new JTextField[Nlinks];
    JTextField[] lengthfield=new JTextField[Nlinks];
    Thread runner;
    public Vector y=new Vector(new double[] {0,0,0,0,x,
theta[0],theta[1],theta[2]});
    ModifyFunction3l Func=new ModifyFunction3l();
    itemListen3b1 it;
    lqr LQR;
    place Plc;
    GradientPaint gp1,gp2;
    Graphics2D offscreenG;
    Stroke stroke;
    JPanel p3 = new JPanel();
    JPanel p5 = new JPanel();
    JButton contb,simulateb,pauseb,getRb,getQb,helpb,poleb;
    JLabel Mlbl,xlbl,Klbl;
    JLabel[] mlbl=new JLabel[Nlinks],thetalbl=new JLabel[Nlinks];
    JLabel[] lengthlbl=new JLabel[Nlinks];
    JFrame windowQ>windowR>windowP;
    TextField [][] txt=new TextField[8][8],txtPole=new TextField[8][2];
    TextField txtR;
    JTextField Kmatrix;
    windhand3b1 mwh=new windhand3b1(this);
    boolean getRpressed=true,getQpressed=true,getPpressed=true;
    boolean Periodic=false,SelectPole=true,animate=false;
```

```

boolean SelectLQR=false;
ButtonGroup cbg=new ButtonGroup();
JRadioButton c1,c2;
GraphicPanel gp;
Container cn;
double[] val2={-1,-2,-3,-4,-5,-6,-7,-8};
double[] val3={1,1,1,1,1,1,1,1};
double[] val4={1};

public class GraphicPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        offscreenG = (Graphics2D)g;
        offscreenG.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
        offscreenG.setColor(Color.blue);
        offscreenG.setStroke(new BasicStroke(1,BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_BEVEL,0,new float[]{5,3},0));
        offscreenG.drawLine((int)x_origin,0,(int)x_origin,
        (int)y_origin);
        offscreenG.translate(x*scale,0);
        offscreenG.setStroke(stroke);
        offscreenG.draw(cart);
        for (int i=0;i<Nlinks;i++)
        {
            offscreenG.rotate(-theta[i],x_origin,y_origin);
            offscreenG.setPaint(i%2==0?gp1:gp2);
            offscreenG.draw(mShape[i]);
            offscreenG.translate(0,-lengths[i]*scale);
            offscreenG.rotate(theta[i],x_origin,y_origin);
        }
    }

    public void update(Graphics g)
    {
        paint(g);
    }

    public void draw()
    {
        repaint();
    }
}

public pend3b1()
{
    initComponents();
}

```

```

public void start()
{
    if (runner == null)
    {
        runner = new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner != null)
    {
        runner.stop();
        runner = null;
    }
}

public void run()
{
    while(true)
    {
        tn=tn_1;
        tn_1=System.currentTimeMillis()/1000.0;
        try { Thread.sleep(refreshing_time);}
        catch (InterruptedException e) {}
        if (animate)
        {
            samples=(int)(100*(tn_1-tn));
            if (samples>1)
            {
                sample_width=(tn_1-tn)/samples;
                for (int j=0;j<samples;j++)
                y=Func.runge_kutta_2nd(tn+j*sample_width,tn+(j+1)
                *sample_width,y);
                theta[0]=y.get(6);
                theta[1]=y.get(7);
                theta[2]=y.get(8);
                for (int i=0;i<Nlinks;i++)
                if (Math.abs(theta[i])>0.0009)
                    thetafield[i].setText(String.valueOf(theta[i]
                    *180/Math.PI));
                else
                    thetafield[i].setText("0.000");
                x=y.get(5);
                if (Math.abs(x)>0.0009)
                    xtfield.setText(String.valueOf(x));
                else
                    xtfield.setText("0.000");
                gp.draw();
            }
        }
    }
}

```



```

    }
}

```

```

public void buildConstraints(GridBagConstraints gbc, int gx,
int gy, int gw, int gh, int wx, int wy)
{
    gbc.gridx=gx;
    gbc.gridy=gy;
    gbc.gridwidth=gw;
    gbc.gridheight=gh;
    gbc.weightx=wx;
    gbc.weighty=wy;
}

```

```

public void initComponents()
{
    gp=new GraphicPanel();
    cn=getContentPane();

    R=EleMat.diag(new Vector(val4));
    Q=EleMat.diag(new Vector(val3));
    PoleVec=new ZVector(val2);
    scale=(int)(360/(lengths[0]+lengths[1]+lengths[2]));

    poleb      = new JButton("Poles");
    getQb      = new JButton("Get Q");
    helpb      = new JButton("Help");
    getRb      = new JButton("Get R");
    contb      = new JButton("Continue");
    simulateb  = new JButton("Simulate");
    pauseb     = new JButton("Pause");

    ButtonAction bact = new ButtonAction();
    poleb.addActionListener(bact);
    contb.addActionListener(bact);
    simulateb.addActionListener(bact);
    pauseb.addActionListener(bact);
    getQb.addActionListener(bact);
    getRb.addActionListener(bact);

    it=new itemListen3b1(this);
    c1=new JRadioButton("LQR",false);
    c2=new JRadioButton("Pole",true);
    cbg.add(c1);
    cbg.add(c2);
    c1.addItemListener(it);
    c2.addItemListener(it);

    p3.setLayout(new GridLayout(txt.length,txt[0].length,5,5));
    p5.setLayout(new GridLayout(txt.length,2,5,5));
    for (int i=0;i<txt.length;i++)

```

```

{
    for (int j=0;j<txt[0].length;j++)
    {
        txt[i][j]=new TextField("0.00");
        p3.add(txt[i][j]);
        if (j<2)
        {
            txtPole[i][j]=new TextField("0.00");
            p5.add(txtPole[i][j]);
        }
    }
    txtPole[i][0].setText(String.valueOf(val2[i]));
    txt[i][i].setText(String.valueOf(val3[i]));
}

windowQ=new JFrame("Enter Q Matrix");
windowQ.setSize(50*txt[0].length,40*txt.length);
windowQ.getContentPane().add(p3);
windowQ.addWindowListener(mwh);
windowR=new JFrame("R Matrix");
windowR.setSize(50,100);
txtR=new TextField("0.00");
windowR.getContentPane().add(txtR);
txtR.setText(String.valueOf(val4[0]));
windowR.addWindowListener(mwh);
windowP=new JFrame("Poles");
windowP.setSize(100,40*txt.length);
windowP.getContentPane().add(p5);
windowP.addWindowListener(mwh);

GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
cn.setLayout(gridbag);

Mtfield  = new JTextField("000");
xtfield  = new JTextField("000");
Kmatrix  = new JTextField("                ");

x_origin=150;
y_origin=380;

// Row 1
buildConstraints(constraints, 0,0,1,1,0,0);
constraints.fill=GridBagConstraints.NONE;
Klbl=new JLabel("K Matrix",JLabel.LEFT);
gridbag.setConstraints(Klbl,constraints);
cn.add(Klbl);

buildConstraints(constraints,1,0,2,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(Kmatrix,constraints);
cn.add(Kmatrix);

```

```

// Column 1 & Column 2
for (int i=1;i<Nlinks+1;i++)
{
    buildConstraints(constraints, 0,Nlinks+1-i,1,1,0,0);
    constraints.fill=GridBagConstraints.NONE;
    mlbl[i-1]=new JLabel("m"+i+" (kg)",JLabel.LEFT);
    gridbag.setConstraints(mlbl[i-1],constraints);
    cn.add(mlbl[i-1]);

    mtfield[i-1] = new JTextField("000");
    buildConstraints(constraints, 1,Nlinks+1-i,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    gridbag.setConstraints(mtfield[i-1],constraints);
    cn.add(mtfield[i-1]);
    mtfield[i-1].setText(String.valueOf(m[i-1]));

    lengthlbl[i-1]=new JLabel("length"+i+"(m)",JLabel.LEFT);
    gridbag.setConstraints(lengthlbl[i-1],constraints);
    cn.add(lengthlbl[i-1]);

    lengthfield[i-1] = new JTextField("000");
    buildConstraints(constraints, 1,2*Nlinks-i+1,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    gridbag.setConstraints(lengthfield[i-1],constraints);
    cn.add(lengthfield[i-1]);
    lengthfield[i-1].setText(String.valueOf(2*lengths[i-1]));

    buildConstraints(constraints, 0,3*Nlinks-i+1,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    thetalbl[i-1]=new JLabel("theta"+i+"(deg)",JLabel.LEFT);
    gridbag.setConstraints(thetalbl[i-1],constraints);
    cn.add(thetalbl[i-1]);

    thetafield[i-1] = new JTextField("000");
    buildConstraints(constraints, 1,3*Nlinks-i+1,1,1,0,0);
    constraints.fill=GridBagConstraints.HORIZONTAL;
    gridbag.setConstraints(thetafield[i-1],constraints);
    cn.add(thetafield[i-1]);
    thetafield[i-1].setText(String.valueOf(Math.round(theta[i-1]*
    18000/Math.PI)/100.0));
    mShape[i-1] = new Line2D.Double(x_origin,y_origin,x_origin,
    y_origin-lengths[i-1]*scale);
}

cart=new Line2D.Double(x_origin-15,y_origin+5,
x_origin+15,y_origin+5);

// Column 1
buildConstraints(constraints, 0,10,1,1,1,1);

```

```

constraints.fill=GridBagConstraints.NONE;
Mlbl=new JLabel("M (Kg)",JLabel.LEFT);
gridbag.setConstraints(Mlbl,constraints);
cn.add(Mlbl);
buildConstraints(constraints, 0,11,1,1,1,1);
constraints.fill=GridBagConstraints.NONE;
xlbl=new JLabel("x (m)",JLabel.LEFT);
gridbag.setConstraints(xlbl,constraints);
cn.add(xlbl);

buildConstraints(constraints, 0,12,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(simulateb,constraints);
cn.add(simulateb);
buildConstraints(constraints, 0,13,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(pauseb,constraints);
cn.add(pauseb);
buildConstraints(constraints,0,14,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(contb,constraints);
cn.add(contb);
buildConstraints(constraints,0,15,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(helpb,constraints);
cn.add(helpb);
buildConstraints(constraints,0,16,1,1,1,1);
constraints.fill=GridBagConstraints.NONE;
gridbag.setConstraints(c1,constraints);
cn.add(c1);

//Column 2
buildConstraints(constraints, 1,10,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(Mtfield,constraints);
cn.add(Mtfield);
buildConstraints(constraints, 1,11,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(xtfield,constraints);
cn.add(xtfield);
buildConstraints(constraints, 1,12,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getRb,constraints);
cn.add(getRb);
buildConstraints(constraints, 1,13,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getQb,constraints);
cn.add(getQb);
buildConstraints(constraints,1,14,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(poleb,constraints);
cn.add(poleb);

```

```

buildConstraints(constraints,1,16,1,1,0,0);
constraints.fill=GridBagConstraints.NONE;
gridbag.setConstraints(c2,constraints);
cn.add(c2);

// Column 3
buildConstraints(constraints, 2,1,1,19,95,0);
constraints.fill=GridBagConstraints.BOTH;
gridbag.setConstraints(gp,constraints);
cn.add(gp);
gp1=new GradientPaint(225,115,Color.green,225,15,
Color.gray,true);
gp2=new GradientPaint(225,115,Color.red,225,15,Color.gray,true);
stroke=new BasicStroke(6,BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND);

Mtfield.setText(String.valueOf(M));
xtfield.setText(String.valueOf(x));
Kmatrix.setEditable(false);

}

class ButtonAction implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() instanceof JButton)
    {
        if (e.getSource() == simulateb)
        {
            stop();
            animate=false;
            getRb.setEnabled(false);
            getQb.setEnabled(false);
            poleb.setEnabled(false);
            contb.setEnabled(false);
            pauseb.setEnabled(true);
            simulateb.setEnabled(false);
            c1.setEnabled(false);
            c2.setEnabled(false);
            for (int i=0;i<Nlinks;i++)
            {
                m[i]=Double.parseDouble(mtfield[i].getText());
                theta[i]=Double.parseDouble(
                thetafield[i].getText())*Math.PI/180.0;
                lengths[i]=Double.parseDouble(
                lengthfield[i].getText())/2.0;
                mtfield[i].setEditable(false);
                thetafield[i].setEditable(false);
                lengthfield[i].setEditable(false);
            }
            Mtfield.setEditable(false);

```

```

xtfield.setEditable(false);
x=Double.parseDouble(xtfield.getText());
scale=(int)(360/(lengths[0]+lengths[1]+lengths[2]));
for (int i=1;i<=Nlinks;i++)
mShape[i-1] = new Line2D.Double(x_origin,y_origin,
x_origin,y_origin-lengths[i-1]*scale);
gp.draw();
Mass=new Matrix(4,4);
B=new Matrix(4,1);
B.set(0,0,1);
Mass.set(0,1,-(m[0]+2*m[1]+2*m[2])*lengths[0]);
Mass.set(0,2,-(m[1]+2*m[2])*lengths[1]);
Mass.set(0,3,-m[2]*lengths[2]);
Mass.set(1,2,(2*m[1]+4*m[2])*lengths[0]*lengths[1]);
Mass.set(1,3,2*m[2]*lengths[0]*lengths[2]);
Mass.set(2,3,2*m[2]*lengths[1]*lengths[2]);
Mass=Mass.plus(Mass.transpose());
Mass.set(0,0,m[0]+m[1]+m[2]+M);
Mass.set(1,1,(m[0]+2*m[1]+2*m[2])*lengths[0]*lengths[0]);
Mass.set(2,2,(m[1]+2*m[2])*lengths[1]*lengths[1]);
Mass.set(3,3,m[2]*lengths[2]*lengths[2]);
Mass=Mass.inverse();
B=Mass.times(B);
A=new Matrix(4,4);
A.set(1,1,(m[0]+2*m[1]+2*m[2])*g*lengths[0]);
A.set(2,2,(m[1]+2*m[2])*g*lengths[1]);
A.set(3,3,m[2]*g*lengths[2]);
A=Mass.times(A);
Mass=new Matrix(8,8);
Mass.setMatrix(0,3,4,7,A);
Mass.setMatrix(4,7,0,3,Matrix.identity(4,4));
A=Mass;
Mass=new Matrix(8,1);
Mass.setMatrix(0,3,0,0,B);
B=Mass;
if (SelectLQR)
{
    LQR=new lqr(A,B,Q,R);
    K=LQR.G;
} else if (SelectPole)
{
    Plc=new place(A,B,PoleVec);
    K=Plc.K;
}
Func.A=A.minus(B.times(K));
Kstring="";
for (int i=0;i<8;i++)
Kstring+=(int)(-K.get(0,i))+". "+(int)(10*Math.abs(
K.get(0,i)-(int)K.get(0,i)))+", ";
Kmatrix.setText(Kstring);
y=new Vector(new double[] {0,0,0,0,x,theta[0],
theta[1],theta[2]});

```

```

        animate=true;
        tn_1=System.currentTimeMillis()/1000.0;
        start();
    }
    if (e.getSource() == pauseb)
    {
        animate=false;
        stop();
        getRb.setEnabled(true);
        getQb.setEnabled(true);
        poleb.setEnabled(true);
        contb.setEnabled(true);
        pauseb.setEnabled(false);
        simulateb.setEnabled(true);
        c1.setEnabled(true);
        c2.setEnabled(true);
        for (int i=0;i<Nlinks;i++)
        {
            mtfld[i].setEditable(true);
            thetafld[i].setEditable(true);
            lengthfld[i].setEditable(true);
        }
        Mtfld.setEditable(true);
        xtfield.setEditable(true);
    }
    if (e.getSource() == getRb)
    {
        stop();
        animate=false;
        simulateb.setEnabled(false);
        getRb.setEnabled(false);
        getQpressed=false;
        getRpressed=true;
        getPpressed=false;
        windowR.setVisible(true);
    }
    if (e.getSource() == poleb)
    {
        stop();
        animate=false;
        simulateb.setEnabled(false);
        poleb.setEnabled(false);
        getQpressed=false;
        getRpressed=false;
        getPpressed=true;
        windowP.setVisible(true);
    }
    if (e.getSource() == getQb)
    {
        stop();
        animate=false;
        simulateb.setEnabled(false);

```



```

Shape[] mShape=new Shape[Nlinks];
Shape cart;
double M=3.0,x=0.1,sample_width,g=10.0;
double[] m={1.2,1.0,.8},lengths={.7,.5,.3};
double theta={0.2,-0.1,0.03},y_origin;
double tn,tn_1=System.currentTimeMillis()/1000.0,x_origin;
JTextField Mtfield,xtfield,obxfield;
JTextField[] mtfield=new JTextField[Nlinks];
JTextField[] Obtfield=new JTextField[Nlinks];
JTextField[] thetfield=new JTextField[Nlinks];
JTextField[] lengthfield=new JTextField[Nlinks];
Thread runner;
public Vector y=new Vector(new double[] {0,0,0,0,x,
theta[0],theta[1],theta[2],0,0,0,0,x,theta[0],theta[1],theta[2]});
ModifyFunction3l Func=new ModifyFunction3l();
itemListen3b12 it;
lqr LQR;
place Plc;
GradientPaint gp1,gp2;
Graphics2D offscreenG;
Stroke stroke;
JPanel p3 = new JPanel();
JPanel p5 = new JPanel();
JButton contb,simulateb,pausb,getRb,getQb,helpb,poleb;
JLabel Mlbl,xlbl,actlbl,oblbl;
JLabel[] mlbl=new JLabel[Nlinks],thetalbl=new JLabel[Nlinks];
JLabel[] lengthlbl=new JLabel[Nlinks];
JFrame windowQ>windowR>windowP;
TextField [][] txt=new TextField[8][8],txtPole=new TextField[8][2];
TextField txtR;
JTextField Kmatrix;
windhand3b12 mwh=new windhand3b12(this);
boolean getRpressed=true,getQpressed=true,getPpressed=true;
boolean Periodic=false,SelectPole=true,animate=false;
boolean SelectLQR=false;
ButtonGroup cbg=new ButtonGroup();
JRadioButton c1,c2;
GraphicPanel gp;
Container cn;
double[] val2={-1,-2,-3,-4,-5,-6,-7,-8};
double[] val3={1,1,1,1,1,1,1,1};
double[] val4={1};

```

```

public class GraphicPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        offscreenG = (Graphics2D)g;
        offscreenG.setRenderingHint(RenderingHints.KEY_ANTIALIASING,

```

```

        RenderingHints.VALUE_ANTIALIAS_ON);
        offscreenG.setColor(Color.blue);
        offscreenG.setStroke(new BasicStroke(1,BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_BEVEL,0,new float[]{5,3},0));
        offscreenG.drawLine((int)x_origin,0,(int)x_origin,
        (int)y_origin);
        offscreenG.translate(x*scale,0);
        offscreenG.setStroke(stroke);
        offscreenG.draw(cart);
        for (int i=0;i<Nlinks;i++)
        {
            offscreenG.rotate(-theta[i],x_origin,y_origin);
            offscreenG.setPaint(i%2==0?gp1:gp2);
            offscreenG.draw(mShape[i]);
            offscreenG.translate(0,-lengths[i]*scale);
            offscreenG.rotate(theta[i],x_origin,y_origin);
        }
    }

    public void update(Graphics g)
    {
        paint(g);
    }

    public void draw()
    {
        repaint();
    }
}

public pend3b12()
{
    initComponents();
}

public void start()
{
    if (runner == null)
    {
        runner = new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner != null)
    {
        runner.stop();
        runner = null;
    }
}

```

```

}

public void run()
{
    while(true)
    {
        tn=tn_1;
        tn_1=System.currentTimeMillis()/1000.0;
        try { Thread.sleep(refreshing_time);}
        catch (InterruptedException e) {}
        if (animate)
        {
            samples=(int)(100*(tn_1-tn));
            if (samples>1)
            {
                sample_width=(tn_1-tn)/samples;
                for (int j=0;j<samples;j++)
                    y=Func.runge_kutta_2nd(tn+j*sample_width,tn+(j+1)*
                    sample_width,y);
                theta[0]=y.get(6);
                theta[1]=y.get(7);
                theta[2]=y.get(8);
                for (int i=0;i<Nlinks;i++)
                {
                    if (Math.abs(theta[i])>0.0009)
                        thetafield[i].setText(String.valueOf(theta[i]*
                        180/Math.PI));
                    else
                        thetafield[i].setText("0.000");
                    if (Math.abs(y.get(14+i))>0.0009)
                        Obtfield[i].setText(String.valueOf(y.get(14+i)*
                        180/Math.PI));
                    else
                        Obtfield[i].setText("0.000");
                }
                x=y.get(5);
                if (Math.abs(x)>0.0009)
                    xtfield.setText(String.valueOf(x));
                else
                    xtfield.setText("0.000");
                if (Math.abs(y.get(13))>0.0009)
                    obxfield.setText(String.valueOf(y.get(13)));
                else
                    obxfield.setText("0.000");
                gp.draw();
            }
        }
    }
}

```

```

public void buildConstraints(GridBagConstraints gbc, int gx,
int gy, int gw, int gh, int wx, int wy)

```

```

{
    gbc.gridx=gx;
    gbc.gridy=gy;
    gbc.gridwidth=gw;
    gbc.gridheight=gh;
    gbc.weightx=wx;
    gbc.weighty=wy;
}

public void initComponents()
{
    gp=new GraphicPanel();
    cn=getContentPane();

    R=EleMat.diag(new Vector(val4));
    Q=EleMat.diag(new Vector(val3));
    PoleVec=new ZVector(val2);
    C=new Matrix(new double[][]{{0,0,0,0,1,0,0,0},
    {0,0,0,0,0,1,0,0}}).transpose();
    scale=(int)(360/(lengths[0]+lengths[1]+lengths[2]));

    poleb      = new JButton("Poles");
    getQb      = new JButton("Get Q");
    helpb      = new JButton("Help");
    getRb      = new JButton("Get R");
    contb      = new JButton("Continue");
    simulateb  = new JButton("Simulate");
    pauseb     = new JButton("Pause");

    ButtonAction bact = new ButtonAction();
    poleb.addActionListener(bact);
    contb.addActionListener(bact);
    simulateb.addActionListener(bact);
    pauseb.addActionListener(bact);
    getQb.addActionListener(bact);
    getRb.addActionListener(bact);

    it=new itemListen3b12(this);
    c1=new JRadioButton("LQR",false);
    c2=new JRadioButton("Pole",true);
    cbg.add(c1);
    cbg.add(c2);
    c1.addItemListener(it);
    c2.addItemListener(it);
    p3.setLayout(new GridLayout(txt.length,txt[0].length,5,5));
    p5.setLayout(new GridLayout(txt.length,2,5,5));
    for (int i=0;i<txt.length;i++)
    {
        for (int j=0;j<txt[0].length;j++)
        {
            txt[i][j]=new TextField("0.00");
            p3.add(txt[i][j]);
        }
    }
}

```

```

        if (j<2)
        {
            txtPole[i][j]=new TextField("0.00");
            p5.add(txtPole[i][j]);
        }
    }
    txtPole[i][0].setText(String.valueOf(val2[i]));
    txt[i][i].setText(String.valueOf(val3[i]));
}

windowQ=new JFrame("Enter Q Matrix");
windowQ.setSize(50*txt[0].length,40*txt.length);
windowQ.getContentPane().add(p3);
windowQ.addWindowListener(mwh);
windowR=new JFrame("R Matrix");
windowR.setSize(50,100);
txtR=new TextField("0.00");
windowR.getContentPane().add(txtR);
txtR.setText(String.valueOf(val4[0]));
windowR.addWindowListener(mwh);
windowP=new JFrame("Poles");
windowP.setSize(100,40*txt.length);
windowP.getContentPane().add(p5);
windowP.addWindowListener(mwh);

GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
cn.setLayout(gridbag);

Mtfield  = new JTextField("000");
xtfield  = new JTextField("000");
obxfield = new JTextField("000");

x_origin=150;
y_origin=380;

buildConstraints(constraints, 1,0,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
actlbl=new JLabel("Actual",JLabel.CENTER);
gridbag.setConstraints(actlbl,constraints);
cn.add(actlbl);

buildConstraints(constraints, 2,0,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
oblbl=new JLabel("Obsvr",JLabel.CENTER);
gridbag.setConstraints(oblbl,constraints);
cn.add(oblbl);

// Column 1 & Column 2
for (int i=1;i<Nlinks+1;i++)
{
    buildConstraints(constraints, 0,Nlinks+1-i,1,1,1,1);

```

```

constraints.fill=GridBagConstraints.HORIZONTAL;
mlbl[i-1]=new JLabel("m"+i+" (kg)",JLabel.RIGHT);
gridbag.setConstraints(mlbl[i-1],constraints);
cn.add(mlbl[i-1]);

mtfield[i-1] = new JTextField("000");
buildConstraints(constraints, 1,Nlinks+1-i,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(mtfield[i-1],constraints);
cn.add(mtfield[i-1]);
mtfield[i-1].setText(String.valueOf(m[i-1]));

buildConstraints(constraints, 0,2*Nlinks+1-i,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
lengthlbl[i-1]=new JLabel("length"+i+" (m)",JLabel.RIGHT);
gridbag.setConstraints(lengthlbl[i-1],constraints);
cn.add(lengthlbl[i-1]);

lengthfield[i-1] = new JTextField("000");
buildConstraints(constraints, 1,2*Nlinks+1-i,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(lengthfield[i-1],constraints);
cn.add(lengthfield[i-1]);
lengthfield[i-1].setText(String.valueOf(2*lengths[i-1]));

buildConstraints(constraints, 0,3*Nlinks+1-i,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
thetalbl[i-1]=new JLabel("theta"+i+" (deg)",JLabel.RIGHT);
gridbag.setConstraints(thetalbl[i-1],constraints);
cn.add(thetalbl[i-1]);

thetafield[i-1] = new JTextField("000");
buildConstraints(constraints, 1,3*Nlinks+1-i,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(thetafield[i-1],constraints);
cn.add(thetafield[i-1]);
thetafield[i-1].setText(String.valueOf(Math.round(theta[i-1]*
1800/Math.PI)/100.0));

Obtfield[i-1] = new JTextField("000");
buildConstraints(constraints, 2,3*Nlinks+1-i,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(Obtfield[i-1],constraints);
cn.add(Obtfield[i-1]);
Obtfield[i-1].setText(String.valueOf(Math.round(
theta[i-1]*18000/Math.PI)/100.0));
mShape[i-1] = new Line2D.Double(x_origin,y_origin,x_origin,
y_origin-lengths[i-1]*scale);
}

cart=new Line2D.Double(x_origin-15,y_origin+5,
x_origin+15,y_origin+5);

```

```

// Column 1
buildConstraints(constraints, 0,10,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
Mlbl=new JLabel("M(kg)",JLabel.RIGHT);
gridbag.setConstraints(Mlbl,constraints);
cn.add(Mlbl);
buildConstraints(constraints, 0,11,1,1,1,1);
constraints.fill=GridBagConstraints.HORIZONTAL;
xlbl=new JLabel("x (m)",JLabel.RIGHT);
gridbag.setConstraints(xlbl,constraints);
cn.add(xlbl);

buildConstraints(constraints, 0,12,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(simulateb,constraints);
cn.add(simulateb);
buildConstraints(constraints, 0,13,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(pauseb,constraints);
cn.add(pauseb);
buildConstraints(constraints,0,14,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(contb,constraints);
cn.add(contb);
buildConstraints(constraints,0,15,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(helpb,constraints);
cn.add(helpb);
buildConstraints(constraints,0,16,1,1,1,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(c1,constraints);
cn.add(c1);

//Column 2
buildConstraints(constraints, 1,10,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(Mtfield,constraints);
cn.add(Mtfield);
buildConstraints(constraints, 1,11,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(xtfield,constraints);
cn.add(xtfield);
buildConstraints(constraints, 1,12,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getRb,constraints);
cn.add(getRb);
buildConstraints(constraints, 1,13,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(getQb,constraints);
cn.add(getQb);
buildConstraints(constraints,1,14,1,1,0,0);

```

```

constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(poleb,constraints);
cn.add(poleb);
buildConstraints(constraints,1,16,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(c2,constraints);
cn.add(c2);

// Column 3
buildConstraints(constraints, 2,11,1,1,0,0);
constraints.fill=GridBagConstraints.HORIZONTAL;
gridbag.setConstraints(obxfield,constraints);
cn.add(obxfield);

buildConstraints(constraints, 3,0,1,17,100,0);
constraints.fill=GridBagConstraints.BOTH;
gridbag.setConstraints(gp,constraints);
cn.add(gp);

gp1=new GradientPaint(225,115,Color.green,225,15,
Color.gray,true);
gp2=new GradientPaint(225,115,Color.red,225,15,Color.gray,true);
stroke=new BasicStroke(6,BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND);
Mtfield.setText(String.valueOf(M));
xtfield.setText(String.valueOf(x));
obxfield.setText(String.valueOf(x));
for (int i=0;i<Nlinks;i++)
    Obtfield[i].setEditable(false);
obxfield.setEditable(false);
}

```

```

class ButtonAction implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() instanceof JButton)
    {
        if (e.getSource() == simulateb)
        {
            stop();
            animate=false;
            getRb.setEnabled(false);
            getQb.setEnabled(false);
            poleb.setEnabled(false);
            contb.setEnabled(false);
            pauseb.setEnabled(true);
            simulateb.setEnabled(false);
            c1.setEnabled(false);
        }
    }
}

```



```

c2.setEnabled(false);
for (int i=0;i<Nlinks;i++)
{
    m[i]=Double.parseDouble(mtfld[i].getText());
    theta[i]=Double.parseDouble(thetafld[i].getText())*
    Math.PI/180.0;
    lengths[i]=Double.parseDouble(
    lengthfld[i].getText())/2.0;
    mtfld[i].setEditable(false);
    thetafld[i].setEditable(false);
    lengthfld[i].setEditable(false);
    Obtfield[i].setEditable(false);
}
Mtfld.setEditable(false);
xtfld.setEditable(false);
x=Double.parseDouble(xtfld.getText());
scale=(int)(360/(lengths[0]+lengths[1]+lengths[2]));
for (int i=1;i<=Nlinks;i++)
mShape[i-1] = new Line2D.Double(x_origin,y_origin,
x_origin,y_origin-lengths[i-1]*scale);
gp.draw();
Mass=new Matrix(4,4);
B=new Matrix(4,1);
B.set(0,0,1);
Mass.set(0,1,-(m[0]+2*m[1]+2*m[2])*lengths[0]);
Mass.set(0,2,-(m[1]+2*m[2])*lengths[1]);
Mass.set(0,3,-m[2]*lengths[2]);
Mass.set(1,2,(2*m[1]+4*m[2])*lengths[0]*lengths[1]);
Mass.set(1,3,2*m[2]*lengths[0]*lengths[2]);
Mass.set(2,3,2*m[2]*lengths[1]*lengths[2]);
Mass=Mass.plus(Mass.transpose());
Mass.set(0,0,m[0]+m[1]+m[2]+M);
Mass.set(1,1,(m[0]+2*m[1]+2*m[2])*lengths[0]*lengths[0]);
Mass.set(2,2,(m[1]+2*m[2])*lengths[1]*lengths[1]);
Mass.set(3,3,m[2]*lengths[2]*lengths[2]);
Mass=Mass.inverse();
B=Mass.times(B);
A=new Matrix(4,4);
A.set(1,1,(m[0]+2*m[1]+2*m[2])*g*lengths[0]);
A.set(2,2,(m[1]+2*m[2])*g*lengths[1]);
A.set(3,3,m[2]*g*lengths[2]);
A=Mass.times(A);
Mass=new Matrix(8,8);
Mass.setMatrix(0,3,4,7,A);
Mass.setMatrix(4,7,0,3,Matrix.identity(4,4));
A=Mass;
Mass=new Matrix(8,1);
Mass.setMatrix(0,3,0,0,B);
B=Mass;
Plc=new place(A.transpose(),C,new ZVector(val2));
E=Plc.K;
if (SelectLQR)

```

```

{
    LQR=new lqr(A,B,Q,R);
    K=LQR.G;
} else if (SelectPole)
{
    Plc=new place(A,B,PoleVec);
    K=Plc.K;
}
Func.A=new Matrix(16,16);
Func.A.setMatrix(0,7,0,7,A);
Func.A.setMatrix(8,15,0,7,E.transpose().times(
C.transpose()));
Func.A.setMatrix(0,7,8,15,(B.times(K)).times(-1.0));
Func.A.setMatrix(8,15,8,15,A.minus(B.times(K)).minus(
E.transpose().times(C.transpose())));
y=new Vector(new double[] {0,0,0,0,x,theta[0],theta[1],
theta[2],0,0,0,0,x,theta[0],theta[1],theta[2]});
animate=true;
tn_1=System.currentTimeMillis()/1000.0;
start();
}
if (e.getSource() == pauseb)
{
    animate=false;
    stop();
    getRb.setEnabled(true);
    getQb.setEnabled(true);
    poleb.setEnabled(true);
    contb.setEnabled(true);
    pauseb.setEnabled(false);
    simulateb.setEnabled(true);
    c1.setEnabled(true);
    c2.setEnabled(true);
    for (int i=0;i<Nlinks;i++)
    {
        mtfield[i].setEditable(true);
        thetafield[i].setEditable(true);
        lengthfield[i].setEditable(true);
    }
    Mtfield.setEditable(true);
    xtfield.setEditable(true);
}
if (e.getSource() == getRb)
{
    stop();
    animate=false;
    simulateb.setEnabled(false);
    getRb.setEnabled(false);
    getQpressed=false;
    getRpressed=true;
    getPpressed=false;
    windowR.setVisible(true);
}

```

```

    }
    if (e.getSource() == poleb)
    {
        stop();
        animate=false;
        simulateb.setEnabled(false);
        poleb.setEnabled(false);
        getQpressed=false;
        getRpressed=false;
        getPpressed=true;
        windowP.setVisible(true);
    }
    if (e.getSource() == getQb)
    {
        stop();
        animate=false;
        simulateb.setEnabled(false);
        getQb.setEnabled(false);
        getQpressed=true;
        getRpressed=false;
        getPpressed=false;
        windowQ.setVisible(true);
    }
    if (e.getSource() == contb)
    {
        getRb.setEnabled(false);
        getQb.setEnabled(false);
        poleb.setEnabled(false);
        contb.setEnabled(false);
        pauseb.setEnabled(true);
        simulateb.setEnabled(false);
        c1.setEnabled(false);
        c2.setEnabled(false);
        for (int i=0;i<Nlinks;i++)
        {
            mtfield[i].setEditable(false);
            thetafield[i].setEditable(false);
            lengthfield[i].setEditable(false);
        }
        Mtfield.setEditable(false);
        xtfield.setEditable(false);
        tn_1=System.currentTimeMillis()/1000.0;
        animate=true;
        start();
    }
    if (e.getSource() == helpb)
    {}
}
}
}
}
}

```

windhand3l.java file

```
import java.awt.event.*;
import Jama.*;
import Jampack.*;

class windhand3l implements WindowListener {
    pend3l wde;
    Matrix temp;
    ZVector V;

    public windhand3l(pend3l wde1)
    {
        wde=wde1;
    }

    public void windowClosing(WindowEvent e)
    {
        if (wde.getQpressed)
        {
            wde.windowQ.setVisible(false);
            temp=new Matrix(wde.txt.length,wde.txt[0].length);
            for (int i=0;i<wde.txt.length;i++)
            for (int j=0;j<wde.txt.length;j++)
            temp.set(i,j,Double.parseDouble(wde.txt[i][j].getText()));
            wde.Q=temp;
        }else if (wde.getRpressed)
        {
            wde.windowR.setVisible(false);
            temp=new Matrix(1,1);
            temp.set(0,0,Double.parseDouble(wde.txtR.getText()));
            wde.R=temp;
        }else if (wde.getPpressed)
        {
            wde.windowP.setVisible(false);
            V=new ZVector(wde.txt.length,1);
            for (int i=1;i<=wde.txt.length;i++)
            V.set(i, new Z(Double.parseDouble(
            wde.txtPole[i-1][0].getText()),Double.parseDouble(
            wde.txtPole[i-1][1].getText())));
            wde.PoleVec=V;
        }
    }

    public void windowClosed(WindowEvent e){}
    public void windowOpened(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}
}
```

windhands3b1.java file

```
import java.awt.event.*;
import Jama.*;
import Jampack.*;

class windhands3b1 implements WindowListener {
    pend3b1 wde;
    Matrix temp;
    ZVector V;

    public windhands3b1(pend3b1 wde1)
    {
        wde=wde1;
    }

    public void windowClosing(WindowEvent e)
    {
        if (wde.getQpressed)
        {
            wde.windowQ.setVisible(false);
            temp=new Matrix(wde.txt.length,wde.txt[0].length);
            for (int i=0;i<wde.txt.length;i++)
            for (int j=0;j<wde.txt.length;j++)
            temp.set(i,j,Double.parseDouble(wde.txt[i][j].getText()));
            wde.Q=temp;
        }else if (wde.getRpressed)
        {
            wde.windowR.setVisible(false);
            temp=new Matrix(1,1);
            temp.set(0,0,Double.parseDouble(wde.txtR.getText()));
            wde.R=temp;
        }else if (wde.getPpressed)
        {
            wde.windowP.setVisible(false);
            V=new ZVector(wde.txt.length,1);
            for (int i=1;i<=wde.txt.length;i++)
            V.set(i, new Z(Double.parseDouble(wde.txtPole[i-1][0].getText()),Double.parseDouble(wde.txtPole[i-1][1].getText())));
            wde.PoleVec=V;
        }
    }

    public void windowClosed(WindowEvent e){}
    public void windowOpened(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}
}
```

windhand3b12.java file

```
import java.awt.event.*;
import Jama.*;
import Jampack.*;

class windhand3b12 implements WindowListener {
    pend3b12 wde;
    Matrix temp;
    ZVector V;

    public windhand3b12(pend3b12 wde1)
    {
        wde=wde1;
    }

    public void windowClosing(WindowEvent e)
    {
        if (wde.getQpressed)
        {
            wde.windowQ.setVisible(false);
            temp=new Matrix(wde.txt.length,wde.txt[0].length);
            for (int i=0;i<wde.txt.length;i++)
            for (int j=0;j<wde.txt.length;j++)
            temp.set(i,j,Double.parseDouble(wde.txt[i][j].getText()));
            wde.Q=temp;
        }else if (wde.getRpressed)
        {
            wde.windowR.setVisible(false);
            temp=new Matrix(1,1);
            temp.set(0,0,Double.parseDouble(wde.txtR.getText()));
            wde.R=temp;
        }else if (wde.getPpressed)
        {
            wde.windowP.setVisible(false);
            V=new ZVector(wde.txt.length,1);
            for (int i=1;i<=wde.txt.length;i++)
            V.set(i, new Z(Double.parseDouble(
            wde.txtPole[i-1][0].getText()),Double.parseDouble(
            wde.txtPole[i-1][1].getText())));
            wde.PoleVec=V;
        }
    }

    public void windowClosed(WindowEvent e){}
    public void windowOpened(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}
}
```

windhand.java file

```
import java.awt.event.*;
import Jama.*;
import Jampack.*;

class windhand implements WindowListener
{
    wedge wde;
    Matrix temp;
    ZVector V;

    public windhand(wedge wde1)
    {
        wde=wde1;
    }

    public void windowClosing(WindowEvent e)
    {
        wde.simulateb.setEnabled(true);
        if (wde.getQpressed)
        {
            wde.windowQ.setVisible(false);
            wde.getQb.setEnabled(true);
            temp=new Matrix(4,4);
            for (int i=0;i<4;i++)
            for (int j=0;j<4;j++)
            temp.set(i,j,Double.parseDouble(wde.txt[i][j].getText()));
            wde.Q=temp;
        }else if (wde.getRpressed)
        {
            wde.windowR.setVisible(false);
            wde.getRb.setEnabled(true);
            temp=new Matrix(1,1);
            temp.set(0,0,Double.parseDouble(wde.txtR.getText()));
            wde.R=temp;
        }else if (wde.getPpressed)
        {
            wde.poleb.setEnabled(true);
            wde.windowP.setVisible(false);
            V=new ZVector(4,1);
            for (int i=1;i<=4;i++)
            V.set(i, new Z(Double.parseDouble(
            wde.txtPole[i-1][0].getText()),Double.parseDouble(
            wde.txtPole[i-1][1].getText())));
            wde.PoleVec=V;
        }
    }

    public void windowClosed(WindowEvent e){}
    public void windowOpened(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
```

```

        public void windowActivated(WindowEvent e){}
        public void windowDeactivated(WindowEvent e){}
    }

```

ModifyFunction3l.java file

```

import Jama.*;

public class ModifyFunction3l extends ode {
    public Matrix A;
    Vector temp;

    public ModifyFunction3l()
    {
    }

    public Vector func(double tn,Vector yn)
    {
        temp=EleMat.times(A,yn);
        return temp;
    }
}

```

ModifyFunction3b.java file

```

import Jama.*;

public class ModifyFunction3b extends ode {
    public Matrix A,B,K;
    Vector temp;

    public ModifyFunction3b()
    {
        A=new Matrix(8,8);
        B=new Matrix(8,1);
        K=new Matrix(1,8);
    }

    public Vector func(double tn,Vector yn)
    {
        temp=EleMat.times(A.minus(B.times(K)),yn);
        return temp;
    }
}

```


itemListen.java file

```
import java.awt.event.*;

public class itemListen implements ItemListener {
    wedge wde;

    public itemListen(wedge wde1)
    {
        wde=wde1;
    }

    public void itemStateChanged(ItemEvent e)
    {
        Object src = e.getSource();
        boolean on = e.getStateChange() == ItemEvent.SELECTED;
        if (src == wde.c1) {wde.SelectLQR=true; wde.SelectPole=false;}
        else if (src == wde.c2) {wde.SelectLQR=false;
        wde.SelectPole=true;}
        else if (src == wde.c3) {wde.Periodic=e.getStateChange()
        ==ItemEvent.SELECTED;}
    }
}
```

itemListen3b1.java file

```
import java.awt.event.*;

public class itemListen3b1 implements ItemListener {
    pend3b1 wde;

    public itemListen3l(pend3b1 wde1)
    {
        wde=wde1;
    }

    public void itemStateChanged(ItemEvent e)
    {
        Object src = e.getSource();
        if (src == wde.c1) {wde.SelectLQR=true;wde.SelectPole=false;}
        else if (src == wde.c2) {wde.SelectLQR=false;
        wde.SelectPole=true;}
        else if (src == wde.c3) {wde.Periodic=e.getStateChange()
        ==ItemEvent.SELECTED;}
    }
}
```

itemListen3b12.java file

```
import java.awt.event.*;

public class itemListen3b12 implements ItemListener
{
    pend3b12 wde;

    public itemListen3b12(pend3b12 wde1)
    {
        wde=wde1;
    }

    public void itemStateChanged(ItemEvent e)
    {
        Object src = e.getSource();
        if (src == wde.c1) {wde.SelectLQR=true; wde.SelectPole=false;}
        else if (src == wde.c2) {wde.SelectLQR=false;
        wde.SelectPole=true;}
        else if (src == wde.c3) {wde.Periodic=e.getStateChange()
        ==ItemEvent.SELECTED;}
    }
}
```

itemListen3b12.java file

```
import java.awt.event.*;

public class itemListen3l implements ItemListener
{
    pend3l wde;

    public itemListen3l(pend3l wde1)
    {
        wde=wde1;
    }

    public void itemStateChanged(ItemEvent e)
    {
        Object src = e.getSource();
        if (src == wde.c1) {wde.SelectLQR=true; wde.SelectPole=false;}
        else if (src == wde.c2) {wde.SelectLQR=false;
        wde.SelectPole=true;}
        else if (src == wde.c3) {wde.Periodic=e.getStateChange()
        ==ItemEvent.SELECTED;}
    }
}
```

balance.java file

```
// Matrix Diagonal scaling to improve eigenvalue accuracy
// B=inverse(T)*A*T
// This program makes a copy of the Matrix passed and will
// hold the result in A

import Jama.*;

public class balance {
    Matrix A;
    double radix = 2.0,r,c,g,s,f;
    double b2 = radix * radix;
    int k = 1;
    int l,m,n,i,j;
    boolean jump,noconv=false,jumpout=false;

    public balance(Matrix A)
    {
        this.A=new Matrix(A.getArrayCopy());
        this.n=A.getColumnDimension();
        l=n;
        sub1();
    }

    public void sub1()
    {
        loop0:
        for (j=1; j>=1;j--)
        {
            jump=false;
            loop1:
            for (i=1;i<=l;i++)
            {
                if ((i!=j) && (A.get(j-1,i-1)!=0.0))
                {
                    jump=true;
                    break loop1;
                }
            }
            if (!jump)
            {
                m=1;
                sub2();
                if (l!=1) j=l--;
                else {jumpout=true; break loop0;}
            }
        }
        if (!jumpout)
        {
            for (j=k;j<=l;j++)
            {
```

```

    jump=false;
    loop2:
    for (i=k;i<=l;i++)
    {
        if ((i!=j)&&(A.get(i-1,j-1)!=0.0))
        {
            jump=true;
            break loop2;
        }
    }
    if (!jump)
    {
        m=k;
        sub2();
        j=k++;
    }
}

do
{
    noconv=false;
    for (i=k;i<=l;i++)
    {
        c=0.0;
        r=0.0;
        for (j=k;j<=l;j++)
        {
            if (j!=i)
            {
                c +=Math.abs(A.get(j-1,i-1));
                r +=Math.abs(A.get(i-1,j-1));
            }
        }
        if (c!=0.0 && r!=0.0)
        {
            g=r/radix;
            f=1.0;
            s=c+r;
            while (c<g)
            {
                f = f*radix;
                c = c*b2;
            }
            g=r*radix;
            while (c>=g)
            {
                f=f/radix;
                c=c/b2;
            }
            if (((c+r)/f)<(.95*s))
            {
                g=1.0/f;
            }
        }
    }
}

```

```

        noconv=true;
        for (int j=k;j<=n;j++)
            A.set(i-1,j-1,A.get(i-1,j-1)*g);
        for (int j=1;j<=l;j++)
            A.set(j-1,i-1,A.get(j-1,i-1)*f);
    }
}
} while (noconv);
}
}

public void sub2()
{
    if (j!=m)
    {
        for (i = 1;i<=l;i++)
        {
            f = A.get(i-1,j-1);
            A.set(i-1,j-1,A.get(i-1,m-1));
            A.set(i-1,m-1,f);
        }

        for (i = k;i<=n;i++)
        {
            f = A.get(j-1,i-1);
            A.set(j-1,i-1,A.get(m-1,i-1));
            A.set(m-1,i-1,f);
        }
    }
}

}

esort.java file

// Sort complex Vector (ZVector) in descending order

public class esort extends sort1 {
    Vector pr,pi,pImag,ipr,Kr,temp,prs,pis,ipi,Ki;
    Vector ndx, si, sr,isep, delims;
    ZVector S;
    int n,istart,iend;
    int [] kk,hh;

    public esort(ZVector p)
    {
        // real and imaginary parts are retrived into two real Vectors
        this.pr=new Vector(p.getReal());
        this.pImag=new Vector(p.getImag());
        n=pr.length();
        sub1();
    }
}

```

```

}

public int [] find(Vector a)
{
    int n=a.length(),m=0,s=0;
    for (int i=1;i<=n;i++)
        if (a.get(i)!=0.0)
            m++;
    int [] count=new int [m];
    for (int i=1;i<=n;i++)
        if (a.get(i)!=0.0) count[s++]=i;
    return count;
}

public Vector gtzero(Vector a)
{
    int n=a.length();
    Vector temp=new Vector(n);
    for (int i=1;i<=n;i++)
        if (a.get(i)>=0.0) temp.set(i,1);
    return temp;
}

public Vector lesszero(Vector a)
{
    int n=a.length();
    Vector temp=new Vector(n);
    for (int i=1;i<=n;i++)
        if (a.get(i)<0.0) temp.set(i,1);
    return temp;
}

public Vector diffequalto(Vector a,double s)
{
    int n=a.length();
    Vector temp =new Vector(n-1);
    for (int i=1;i<=n-1;i++)
        if ((a.get(i+1)-a.get(i))==s)
            temp.set(i,1);
    return temp;
}

public Vector diff(Vector a)
{
    Vector temp=new Vector(n-1);
    for (int i=1;i<=n-1;i++)
        temp.set(i,a.get(i+1)-a.get(i));
    return temp;
}

public Vector diffequaltozero(Vector R, Vector I)
{

```

```

    Vector temp=new Vector(n-1);
    boolean zero=false;
    for (int i=1;i<=n-1;i++)
        if ((R.get(i+1)-R.get(i))==0.0 && (Math.abs(I.get(i+1))-
            Math.abs(I.get(i)))==0.0)
            temp.set(i,1);
    return temp;
}

public Vector filter(Vector a)
{
    int n=a.length();
    Vector temp=new Vector(n);
    double sum=0.0;
    for (int i=1;i<=n;i++)
    {
        sum +=a.get(i);
        temp.set(i,sum);
    }
    return temp;
}

public Vector notequal(Vector a, double s)
{
    int n=a.length();
    Vector temp=new Vector(n);
    for (int i=1;i<=n;i++)
        if (a.get(i)!=s) temp.set(i,1);
    return temp;
}

public void sub1()
{
    pi=new Vector(n);
    for (int i=1;i<=n;i++)
        pi.set(i,Math.abs(pImag.get(i)));
    calculate(pr.multiply(-1));
    prs=sorted_vec;
    ipr=index;
    Kr=new Vector(n);
    temp=filter(notequal(diff(prs),0.0));
    for (int i=2;i<=n;i++)
        Kr.set((int)ipr.get(i),temp.get(i-1));
    calculate(pi);
    pis=sorted_vec;
    ipi=index;
    Ki=new Vector(n);
    temp=filter(notequal(diff(pis),0.0));
    for (int i=2;i<=n;i++)
        Ki.set((int)ipi.get(i),temp.get(i-1));
    calculate(Kr.plus(Ki.divide(n)));
    ndx=index;
}

```

```

si=new Vector(n);
sr=new Vector(n);
for (int i=1;i<=n;i++)
{
    si.set(i,pImag.get((int)ndx.get(i)));
    sr.set(i,pr.get((int)ndx.get(i)));
}
isep=new Vector(n);
temp=diffequaltozero(sr,si);
for (int i=2;i<=n;i++)
isep.set(i,temp.get(i-1));
kk=find(diffequalto(isep,1));
Vector temp2=new Vector(isep.length()+1);
for (int i=1;i<=isep.length();i++)
temp2.set(i,isep.get(i));
hh=find(diffequalto(temp2,-1));

for (int l=0;l<kk.length;l++)
{
    istart=kk[l];
    iend=hh[l];
    Vector clustr=new Vector(iend-istart+1);
    Vector clusti=new Vector(iend-istart+1);
    int m=1;
    for (int i=istart;i<=iend;i++)
    {
        clustr.set(m,sr.get(i));
        clusti.set(m++,si.get(i));
    }
    int [] ipos =find(gtzero(clusti));
    int [] ineg =find(lesszero(clusti));
    int npairs=Math.min(ipos.length,ineg.length);
    Vector iperm=new Vector(clustr.length());
    m=0;
    for (int i=1;i<=2*npairs;i+=2)
    {
        iperm.set(i,ipos[m]);
        iperm.set(i+1,ineg[m++]);
    }
    m=npairs;
    int tt1=ipos.length-npairs;
    for (int i=2*npairs+1;i<=2*npairs+tt1;i++)
    iperm.set(i,ipos[m++]);
    m=npairs;
    for (int i=2*npairs+tt1+1;i<=clustr.length();i++)
    iperm.set(i,ineg[m++]);
    m=1;
    for (int i=istart;i<=iend;i++)
    {
        sr.set(i,clustr.get((int)iperm.get(m)));
        si.set(i,clusti.get((int)iperm.get(m++)));
    }
}

```



```

    }
    S=new ZVector(sr,si);
}
}

```

ode.java file

```

import Jama.*;

public abstract class ode {
    public abstract Vector func(double tn,Vector yn);

    public Vector runge_kutta_2nd(double tn, double tn_1,Vector yn)
    {
        Vector k2,k1,yn_1;
        double h;
        h=tn_1-tn;
        k1=func(tn,yn).multiply(0.5*h);
        k2=func(tn+0.5*h,k1.plus(yn)).multiply(h);
        yn_1=k2.plus(yn);
        return yn_1;
    }

    public Vector runge_kutta_4th(double tn, double tn_1, Vector yn)
    {
        Vector k1,k2,k3,k4,temp,yn_1;
        double h;
        h=tn_1-tn;
        k1=func(tn,yn).multiply(h);
        k2=func(tn+0.5*h,yn.plus(k1.multiply(0.5))).multiply(h);
        k3=func(tn+0.5*h,yn.plus(k2.multiply(0.5))).multiply(h);
        k4=func(tn+h,yn.plus(k3)).multiply(h);
        temp=k1.plus(k2.multiply(2.0).plus(k3.multiply(2.0)
        .plus(k4)));
        yn_1=yn.plus(temp.divide(6.0));
        return yn_1;
    }
}

```

QRdecomp.java file

```

import Jama.*;

/* QR decomposition A=QR */
// getQ() will return the Q matrix
// getR() will return the R matrix

public class QRdecomp
{
    int m,n;
    Matrix A;
}

```

```

public QRdecomp(Matrix A)
{
    this.A=A;
    m=A.getRowDimension();
    n=A.getColumnDimension();
}

public Matrix HouseHolderTf(int k,Vector x)
{
    Matrix temp;
    Vector v,u;
    double sigma,theta;
    if (x.Empty()) temp=Matrix.identity(m-k+1,m-k+1);
    else
    {
        temp=new Matrix(m-k+1,m-k+1);
        v=x.divide(x.inf_norm());
        sigma=(v.get(1)<0.0?-1:1)*v.L2_norm();
        v.set(1,v.get(1)+sigma);
        theta=sigma*v.get(1);
        temp=v.TimesTransposeOf(v.divide(theta));
        temp=Matrix.identity(m-k+1,m-k+1).minus(temp);
    }
    return temp;
}

public Matrix getQ()
{
    int r;
    Matrix U,HouseH;
    U=Matrix.identity(m,m);
    Matrix Q=new Matrix(m,m);
    Vector VV;
    r=Math.min(m-1,n);
    for (int k=1;k<=r;k++)
    {
        if (k==1)
        {
            Matrix Uk=new Matrix(m,m);
            VV=new Vector(A.getMatrix(0,m-1,0,0).getColumnPackedCopy());
            Uk=HouseHolderTf(1,VV);
            A=Uk.times(A);
            A.setMatrix(1,m-1,0,0,new Matrix(m-1,1));
            U=Uk;
        }else
        {
            Matrix Uk=new Matrix(m,m);
            VV=new Vector(A.getMatrix(k-1,m-1,k-1,k-1)
                .getColumnPackedCopy());
            HouseH=HouseHolderTf(k,VV);
            if (k==2) Uk.set(0,0,1.0);
        }
    }
}

```

```

        else
            Uk.setMatrix(0,k-2,0,k-2,Matrix.identity(k-1,k-1));

            Uk.setMatrix(k-1,m-1,k-1,m-1,HouseH);
            U=Uk.times(U);
            A=Uk.times(A);
            A.setMatrix(k,m-1,k-1,k-1,new Matrix(m-k,1));
        }
    }
    return U.transpose();
}

public Matrix getR()
{
    return A;
}
}

```

qz.java file

Handles real matrices A and B.

/* create object qz. Constructor executes the calculation routine. Qz, Zz will hold the Unitary Q and Z while AA, BB will be Qz*A*Zz and Qz*B*Zz repectively. */

```

import Jampack.*;
import Jama.*;

public class qz {
    Matrix A,B, wa,wb;
    Zmat Qz,Zz,AA,BB;

    public qz(Matrix A,Matrix B) throws JampackException
    {
        this.A=A;
        this.B=B;
        wa=new Matrix(A.getArrayCopy());
        wb=new Matrix(B.getArrayCopy());
        cal();
    }

    public Zmat HouseHolderTf(int i,int k,int r, Zmat x)
    throws JampackException
    {
        int n=x.nr;
        Zmat temp2;
        Zmat temp=Eye.o(n);
        Z1 V=House1.genc(x,k,k+r-1,1,i);
        temp2=ZVector.house(r,V);
        temp.put(k,k+r-1,k,k+r-1,temp2);
        return temp;
    }
}

```

```

}

public Matrix HouseHolderTf(int i, Vector x)
{
    int n=x.length();
    Matrix temp;
    Vector v;
    double sigma,theta;
    if (x.Empty()||(x.inf_norm()<1E-8))
    {
        temp=Matrix.identity(n,n);
        temp.set(0,0,-1);
    }
    else
    {
        v=x.divide(x.inf_norm());
        sigma=(v.get(i)<0.0?-1:1)*v.L2_norm();
        v.set(i,v.get(i)+sigma);
        theta=sigma*v.get(i);
        temp=v.TimesTransposeOf(v.divide(theta));
        temp=Matrix.identity(n,n).minus(temp);
    }
    return temp;
}

public Matrix HouseHolderTf(int i,int k,int r,Vector x)
{
    int n=x.length();
    Matrix temp=Matrix.identity(n,n);
    temp.setMatrix(k-1,k+r-2,k-1,k+r-2,
    HouseHolderTf(i,x.getVector(k,k+r-1)));
    return temp;
}

public void cal() throws JampackException
{
    int now,previous,mm,m,n=B.getRowDimension();
    Matrix Qr=Matrix.identity(n,n),Zr=Matrix.identity(n,n);
    Matrix Uk,E;
    Zmat UU,ZZ,UUk,Atemp,Btemp,EE;
    double a10,a20,a30,p,a_12,a_22,mew,r,sigma;
    Z lambda;
    Vector V;
    Qz=Eye.o(n);
    Zz=Eye.o(n);
    boolean cont,done=true;
    for (int k=1;k<=n-1;k++)
    {
        V=new Vector(B.getMatrix(0,n-1,k-1,k-1)
        .getColumnPackedCopy());

```

```

    Uk=HouseHolderTf(1,k,n-k+1,V);
    B=Uk.times(B);
    A=Uk.times(A);
    Qr=Uk.times(Qr);
}

// step 2 Making B triangular
for (int k=1;k<=n-2;k++)
    for (int l=n-1;l>=k+1;l--)
    {
        V=new Vector(A.getMatrix(0,n-1,k-1,k-1)
            .getColumnPackedCopy());
        Uk=HouseHolderTf(1,l,2,V);
        A=Uk.times(A);
        B=Uk.times(B);
        Qr=Uk.times(Qr);
        V=new Vector(B.getMatrix(1,l,0,n-1)
            .getColumnPackedCopy());
        Uk=HouseHolderTf(2,l,2,V);
        B=B.times(Uk);
        A=A.times(Uk);
        Zr=Zr.times(Uk);
    }

// step 3 making A Hessenberg
mm=0;
m=n;
while (done)
{
    sigma=(-A.get(m-1,m-2)*B.get(m-2,m-1)/(B.get(m-2,m-2)*
        B.get(m-1,m-1))+A.get(m-1,m-1)/B.get(m-1,m-1));
    E=A.minus(B.times(sigma));
    for (int k=1+mm;k<m;k++)
    {
        V=new Vector(E.getMatrix(0,n-1,k-1,k-1)
            .getColumnPackedCopy());
        Uk=HouseHolderTf(1,k,2,V);
        E=Uk.times(E);
        B=Uk.times(B);
        Qr=Uk.times(Qr);
        V=new Vector(B.getMatrix(k,k,0,n-1)
            .getColumnPackedCopy());
        Uk=HouseHolderTf(2,k,2,V);
        E=E.times(Uk);
        B=B.times(Uk);
        Zr=Zr.times(Uk);
        A=E.plus(B.times(sigma));
    }

    // matrix size reduction process
    while ((mm+2<m)&&(Math.abs(A.get(m-1,m-2))<1E-9))
    {

```

```

        A.set(m-1,m-2,0.0);
        m =m-1;
    }
    while ((m>mm+2)&&(Math.abs(A.get(m-2,m-3))<1E-9))
    {
        A.set(m-2,m-3,0.0);
        m=m-2;
    }
    while ((mm+2<m)&&(Math.abs(A.get(mm+1,mm))<1E-9))
    {
        A.set(mm+1,mm,0.0);
        mm=mm+1;
    }
    while ((mm<m-2)&&(Math.abs(A.get(mm+2,mm+1))<1E-9))
    {
        A.set(mm+2,mm+1,0.0);
        mm=mm+2;
    }
    cont=false;
    previous=0;
    if (mm<=m-1)
    for (int ii=mm;ii<=m-1;ii++)
    {
        now=(Math.abs(A.get(ii+1,ii))<1E-9?0:1);
        if (previous*now==1) cont=true;
        else previous=now;
    }
    if (!cont) done=false;
    else
        if ((n-m+mm+1)>=n-1) done=false;
}

// real to imaginary matrix conversion
Qz=new Zmat(Qr.getArrayCopy(),new double[n][n]);
Zz=new Zmat(Zr.getArrayCopy(),new double[n][n]);
AA=new Zmat(A.getArrayCopy(),new double[n][n]);
BB=new Zmat(B.getArrayCopy(),new double[n][n]);

Zmat ttt;

for (int j=1;j<n;j++)
{
    if (Math.abs(A.get(j,j-1))>1E-9)
    {
        // 2x2 matrix formation
        Atemp=new Zmat(A.getMatrix(j-1,j,j-1,j).getArrayCopy(),
            new double[2][2]);
        Btemp=new Zmat(B.getMatrix(j-1,j,j-1,j).getArrayCopy(),
            new double[2][2]);
        if (Z.abs(Btemp.get(1,1))<2E-17)
        {
            // case when B.get(1,1)==0

```

```

        ttt=Eye.o(n);
        UUK=HouseHolderTf(1,1,2,Atemp.get(1,2,1,1));
        ttt.put(j,j+1,j,j+1,UUK);
        AA=Times.o(ttt,AA);
        BB=Times.o(ttt,BB);
        Qz=Times.o(ttt,Qz);
    }else if (Z.abs(Btemp.get(2,2))<2E-17)
    {
        // case when B.get(j,j)==0
        ttt=Eye.o(n);
        UUK=HouseHolderTf(2,1,2,H.o(Atemp.get(2,2,1,2)));
        ttt.put(j,j+1,j,j+1,UUK);
        AA=Times.o(AA,ttt);
        BB=Times.o(BB,ttt);
        Zz=Times.o(Zz,ttt);
    }else
    {
        ttt=Eye.o(n);
        mew=A.get(j-1,j-1)/B.get(j-1,j-1);
        a_12=A.get(j-1,j)-mew*B.get(j-1,j);
        a_22=A.get(j,j)-mew*B.get(j,j);
        p=0.5*(a_22/B.get(j,j)-B.get(j-1,j)*A.get(j,j-1)/
        (B.get(j-1,j-1)*B.get(j,j)));
        r=p*p+A.get(j,j-1)*a_12/(B.get(j-1,j-1)*B.get(j,j));
        if (r<0.0)
            lambda=new Z(mew+p,(p<0.0?-1:1)*Math.sqrt(-r));
        else
            lambda=new Z(mew+p+(p<0.0?-1:1)*Math.sqrt(r),0.0);
        EE=Minus.o(Atemp,Times.o(lambda,Btemp));
        if ((Z.abs(EE.get(1,1))+Z.abs(EE.get(1,2)))<
        (Z.abs(EE.get(2,1))+Z.abs(EE.get(2,2))))
            UUK=HouseHolderTf(2,1,2,H.o(EE.get(1,1,1,2)));
        else
            UUK=HouseHolderTf(2,1,2,H.o(EE.get(2,2,1,2)));
        ttt.put(j,j+1,j,j+1,UUK);
        AA=Times.o(AA,ttt);
        BB=Times.o(BB,ttt);
        Zz=Times.o(Zz,ttt);
        Atemp=Times.o(Atemp,UUK);
        Btemp=Times.o(Btemp,UUK);
        EE=Times.o(lambda,Btemp);
        if (Norm.fro(Atemp)<Norm.fro(EE))
            UUK=HouseHolderTf(1,1,2,Atemp.get(1,2,1,1));
        else
            UUK=HouseHolderTf(1,1,2,EE.get(1,2,1,1));
        ttt=Eye.o(n);
        ttt.put(j,j+1,j,j+1,UUK);
        AA=Times.o(ttt,AA);
        BB=Times.o(ttt,BB);
        Qz=Times.o(ttt,Qz);
    }
}

```

```

    }
    }
    for (int j=1;j<=AA.nr;j++)
    for (int k=1;k<j;k++)
    {
        if (Z.abs(AA.get(j,k))<1E-9) AA.put(j,k,Z.ZERO);
        if (Z.abs(BB.get(j,k))<1E-9) BB.put(j,k,Z.ZERO);
    }
}
}

```

qzZ.java file

Handles complex matrices A and B.

```

import Jama.*;
import Jampack.*;

public class qzZ {
    Zmat A,B,U,ZZ;

    public qzZ(Zmat A,Zmat B) throws JampackException
    {
        this.A=new Zmat(A);
        this.B=new Zmat(B);
        cal();
    }

    public Zmat HouseHolderTf(int i,int k,int r, Zmat x)
    throws JampackException
    {
        int n=x.nr;
        Zmat temp2;
        Zmat temp=Eye.o(n);
        Z1 V=House1.genc(x,k,k+r-1,1,i);
        temp2=ZVector.house(r,V);
        temp.put(k,k+r-1,k,k+r-1,temp2);
        return temp;
    }

    public void cal() throws JampackException
    {
        int m,mm,now,previous,n=B.nr;
        Z mew,p,r,q;
        Zops zt=new Zops();
        Zmat E,Uk,temp,Atemp,Btemp,EE;
        Z a10,a20,a30,a_12,a_22;
        Z lambda,sigma;
        Z xt=new Z();
        U=Eye.o(n);
        ZZ=Eye.o(n);
    }
}

```



```

boolean cont,done=true;
for (int k=1;k<=n-1;k++)
{
    temp=new Zmat(B.get(1,n,k,k));
    Uk=HouseHolderTf(1,k,n-k+1,temp);
    B=Times.o(Uk,B);
    for(int i=k+1;i<=n;i++)
    // making exact zeros
    B.put(i,k,Z.ZERO);
    A=Times.o(Uk,A);
    U=Times.o(Uk,U);
}

for (int k=1;k<=n-2;k++)
    for (int l=n-1;l>=k+1;l--)
    {
        temp=new Zmat(A.get(1,n,k,k));
        Uk=HouseHolderTf(1,l,2,temp);
        A=Times.o(Uk,A);
        // making exact zeros
        A.put(l+1,k,Z.ZERO);
        B=Times.o(Uk,B);
        U=Times.o(Uk,U);
        temp=new Zmat(H.o(B).get(1,n,l+1,l+1));
        Uk=HouseHolderTf(2,l,2,temp);
        B=Times.o(B,Uk);
        // making exact zeros
        B.put(l+1,l,Z.ZERO);
        A=Times.o(A,Uk);
        ZZ=Times.o(ZZ,Uk);
    }

mm=0;
m=n;
while (done)
{
    sigma=zt.Plus(zt.Div(zt.Times(new Z(-1,0),
    zt.Times(A.get(m,m-1),
    B.get(m-1,m))),zt.Times(B.get(m-1,m-1),B.get(m,m))),
    zt.Div(A.get(m,m),B.get(m,m)));
    E=Minus.o(A,Times.o(sigma,B));
    for (int k=1+mm;k<m;k++)
    {
        temp=new Zmat(E.get(1,n,k,k));
        Uk=HouseHolderTf(1,k,2,temp);
        E=Times.o(Uk,E);
        B=Times.o(Uk,B);
        U=Times.o(Uk,U);
        temp=new Zmat(H.o(B).get(1,n,k+1,k+1));
        Uk=HouseHolderTf(2,k,2,temp);
        E=Times.o(E,Uk);
        B=Times.o(B,Uk);
    }
}

```

```

        ZZ=Times.o(ZZ,Uk);
        A=Plus.o(E,Times.o(sigma,B));
    }
    while ((mm+2<m)&&(Z.abs(A.get(m,m-1))<1E-9))
    {
        A.put(m,m-1,Z.ZERO);
        m=m-1;
    }
    while ((m>mm+2)&&(Z.abs(A.get(m-1,m-2))<1E-9))
    {
        A.put(m-1,m-2,Z.ZERO);
        m=m-2;
    }
    while ((mm+2<m)&&(Z.abs(A.get(mm+2,mm+1))<1E-9))
    {
        A.put(mm+2,mm+1,Z.ZERO);
        mm=mm+1;
    }
    while ((mm<m-2)&&(Z.abs(A.get(mm+3,mm+2))<1E-9))
    {
        A.put(mm+3,mm+2,Z.ZERO);
        mm=mm+2;
    }
    cont=false;
    previous=0;
    if (mm<=m-1)
    for (int ii=mm+1;ii<=m-1;ii++)
    {
        now=(Z.abs(A.get(ii+1,ii))<1E-9?0:1);
        if (previous*now==1) cont=true;
        else previous=now;
    }
    if (!cont) done=false;
    else
        if ((n-m+mm+1)>=n-1) done=false;
}

for (int j=1;j<=A.nr;j++)
for (int k=1;k<j;k++)
{
    if (Z.abs(A.get(j,k))<1E-8) A.put(j,k,Z.ZERO);
    if (Z.abs(B.get(j,k))<1E-8) B.put(j,k,Z.ZERO);
}

Zmat ttt;
for (int j=2;j<=n;j++)
{
    if (Z.abs(A.get(j,j-1))>1E-9)
    {
        // 2x2 matrix formation
        Atemp=A.get(j-1,j,j-1,j);
        Btemp=B.get(j-1,j,j-1,j);
    }
}

```

```

if (Z.abs(Btemp.get(1,1))<2E-17)
{
    // case when B.get(1,1)==0
    ttt=Eye.o(n);
    Uk=HouseHolderTf(1,1,2,Atemp.get(1,2,1,1));
    ttt.put(j-1,j,j-1,j,Uk);
    A=Times.o(ttt,A);
    B=Times.o(ttt,B);
    U=Times.o(ttt,U);
}else if (Z.abs(Btemp.get(2,2))<2E-17)
{
    // case when B.get(j,j)==0
    ttt=Eye.o(n);
    Uk=HouseHolderTf(2,1,2,H.o(Atemp.get(2,2,1,2)));
    ttt.put(j-1,j,j-1,j,Uk);
    A=Times.o(A,ttt);
    B=Times.o(B,ttt);
    ZZ=Times.o(ZZ,ttt);
}else
{
    ttt=Eye.o(n);
    mew=zt.Div(A.get(j-1,j-1),B.get(j-1,j-1));
    a_12=zt.Minus(A.get(j-1,j),zt.Times(mew,B.get(j-1,j)));
    a_22=zt.Minus(A.get(j,j),zt.Times(mew,B.get(j,j)));
    p=zt.Times(new Z(0.5,0),(zt.Minus(zt.Div(a_22,
    B.get(j,j)),zt.Div(zt.Times(B.get(j-1,j),A.get(j,j-1))
    ,zt.Times(B.get(j-1,j-1),B.get(j,j))))));
    r=zt.Plus(zt.Times(p,p),zt.Div(zt.Times(
    A.get(j,j-1),a_12),zt.Times(B.get(j-1,j-1)
    ,B.get(j,j)))));
    if (r!=Z.ZERO)
        lambda=zt.Plus(zt.Plus(mew,p),xt.Sqrt(r));
    else
        lambda=zt.Plus(mew,p);
    EE=Minus.o(Atemp,Times.o(lambda,Btemp));
    if((Z.abs(EE.get(1,1))+Z.abs(EE.get(1,2)))<
    (Z.abs(EE.get(2,1))+Z.abs(EE.get(2,2))))
        Uk=HouseHolderTf(2,1,2,H.o(EE.get(1,1,1,2)));
    else
        Uk=HouseHolderTf(2,1,2,H.o(EE.get(2,2,1,2)));
    ttt.put(j-1,j,j-1,j,Uk);
    A=Times.o(A,ttt);
    B=Times.o(B,ttt);
    ZZ=Times.o(ZZ,ttt);
    Atemp=Times.o(Atemp,Uk);
    Btemp=Times.o(Btemp,Uk);
    EE=Times.o(lambda,Btemp);
    if (Norm.fro(Atemp)<Norm.fro(EE))
        Uk=HouseHolderTf(1,1,2,Atemp.get(1,2,1,1));
    else
        Uk=HouseHolderTf(1,1,2,EE.get(1,2,1,1));
    ttt=Eye.o(n);

```

```

        ttt.put(j-1,j,j-1,j,Uk);
        A=Times.o(ttt,A);
        B=Times.o(ttt,B);
        U=Times.o(ttt,U);
    }

    }
}
for (int j=1;j<=A.nr;j++)
for (int k=1;k<j;k++)
{
    if (Z.abs(A.get(j,k))<1E-9) A.put(j,k,Z.ZERO);
    if (Z.abs(B.get(j,k))<1E-9) B.put(j,k,Z.ZERO);
}
}
}

```

ZVector.java file

Handles Complex vector operations

```

import Jampack.*;

public class ZVector {
    private int n;
    private Z[] V;

    public ZVector(int m, double s)
    {
        this.n=m;
        V=new Z[n];
        for (int i=0;i<n;i++)
            V[i]=new Z(s,0.0);
    }

    public ZVector(int m)
    {
        this.n=m;
        V=new Z[n];
        for (int i=0;i<n;i++)
            V[i]=new Z(0.0,0.0);
    }

    public ZVector(Vector R, Vector I)
    {
        this.n=R.length();
        V=new Z[n];
        for (int i=0;i<n;i++)
            V[i]=new Z(R.get(i+1),I.get(i+1));
    }

    public ZVector(double [] U)

```

```

{
    this.n=U.length;
    V=new Z[n];
    for (int i=0;i<n;i++)
        V[i]=new Z(U[i],0.0);
}

public ZVector(double [] U,double [] K)
{
    this.n=U.length;
    V=new Z[n];
    for (int i=0;i<n;i++)
        V[i]=new Z(U[i],K[i]);
}

public ZVector(Z[] U)
{
    this.n=U.length;
    V=new Z[n];
    V=U;
}

public boolean Empty()
{
    boolean ans=true;
    for (int i=0;i<n;i++)
        if (V[i].re!=0.0 && V[i].im!=0.0) ans=false;
    return ans;
}

public void print()
{
    for(int i=0;i<n;i++)
        System.out.println(V[i].re+"\t"+V[i].im);
}

public double[] getReal()
{
    double [] K =new double [n];
    for (int j=0;j<n;j++)
        K[j]=V[j].re;
    return K;
}

public double[] getImag()
{
    double [] K =new double [n];
    for (int j=0;j<n;j++)
        K[j]=V[j].im;
    return K;
}

public double getReal(int i)

```

```

{
    return (V[i-1].re);
}
public double getImag(int i)
{
    return (V[i-1].im);
}

public int length()
{
    return this.n;
}

public Z get(int i)
{
    return V[i-1];
}

public void set(int i,Z s)
{
    V[i-1]=s;
}

public void setReal(int i,double s)
{
    V[i-1].re=s;
}

public void setImag(int i,double s)
{
    V[i-1].im=s;
}

public double RealMin(ZVector s)
{
    double temp=s.getReal(1);
    for (int i=2;i<=s.length();i++)
        temp=Math.min(temp,s.getReal(i));
    return temp;
}

public ZVector plus(ZVector s)
{
    Z[] temp =new Z[this.length()];
    for (int i=0;i<this.length();i++)
    {
        temp[i].re=V[i].re+s.getReal(i+1);
        temp[i].im=V[i].im+s.getImag(i+1);
    }
    return new ZVector(temp);
}

```

```

public ZVector AddElement(Z s)
{
    ZVector temp =new ZVector(n+1);
    for (int i=1;i<=n;i++)
        temp.set(i,V[i-1]);
    temp.set(n+1,s);
    return temp;
}

public ZVector AddElement(double s)
{
    ZVector temp =new ZVector(n+1);
    for (int i=1;i<=n;i++)
        temp.set(i,V[i-1]);
    temp.set(n+1,new Z(s,0.0));
    return temp;
}

public ZVector multiply(Z s)
{
    Z[] temp = new Z[this.length()];
    for (int i=0; i<this.length();i++)
        temp[i].Times(s,V[i]);
    return new ZVector(temp);
}

public ZVector multiply(ZVector S)
{
    Z[] temp =new Z[this.length()];
    if (this.length() !=S.length())
    {
        throw new IllegalArgumentException("Vector lengths
            must be equal.");
    }
    for (int i=0; i<this.length();i++)
        temp[i].Times(S.get(i+1),V[i]);
    return new ZVector(temp);
}

public double L2_norm()
{
    double sum=0.0;
    for (int i=0;i<n;i++)
        sum +=V[i].re*V[i].re+V[i].im*V[i].im;
    return(Math.sqrt(sum));
}

public ZVector divide(double s) throws JampackException
{
    Z t=new Z(0,0);
    if (Math.abs(s)<0.00000001)

```

```

    {
        throw new IllegalArgumentException("Division by
            a small number.");
    }
    Z[] temp = new Z[this.length()];
    for (int i=0; i<this.length();i++)
        temp[i]=t.Div(V[i],s);
    return new ZVector(temp);
}

public ZVector divide(Z s) throws JampackException
{
    Z t=new Z(0,0);
    if (Z.abs(s)<0.00000001)
    {
        throw new IllegalArgumentException("Division by
            a small number.");
    }
    Z[] temp = new Z[this.length()];
    for (int i=0; i<this.length();i++)
        temp[i]=t.Div(V[i],s);
    return new ZVector(temp);
}

public double inf_norm()
{
    double sum=Z.abs(V[0]);
    for (int i=1;i<n;i++)
        if (Z.abs(V[i])>sum) sum=Z.abs(V[i]);
    return sum;
}

public ZVector getVector(int k,int r)
{
    int j=1;
    ZVector temp=new ZVector(r-k+1);
    for (int i=k;i<=r;i++)
    {
        temp.set(j,V[i-1]);
        j++;
    }
    return temp;
}

public Zmat TimesTransposeOf(ZVector s)
{
    Z t=new Z(1,1);
    Zmat temp=new Zmat(n,n);
    for (int j=1;j<=n;j++)
        for (int k=1;k<=n;k++)

```



```

        temp.put(j,k,t.Times(V[j-1],s.get(k)));
        return temp;
    }

    public Z SumElements()
    {
        Z sum=new Z();
        for (int i=0; i<this.length();i++)
        {
            sum.im += V[i].im;
            sum.re += V[i].re;
        }
        return sum;
    }

    public static ZVector ColumnVector(int k,Zmat A)
    {
        int n=A.nr;
        ZVector V=new ZVector(n);
        for (int i=1;i<=n;i++)
            V.set(i,A.get(i,k));
        return V;
    }

    public static Zmat house(int n,Z1 V) throws JampackException
    {
        Zmat temp1=new Zmat(n,1);
        Zmat temp2;
        for (int i=0;i<n;i++)
            temp1.put(i+1,1,V.get(i));
        temp2=Times.o(temp1,H.o(temp1));
        return (Minus.o(Eye.o(n),temp2));
    }
}

```

Vector.java file

Handles Real vector operations

```

import Jama.*;

public class Vector implements java.io.Serializable {
    private int n;
    private double [] V;

    public Vector(int m, double s)
    {
        this.n=m;
        V=new double[n];
        for (int i=0;i<n;i++)
            V[i]=s;
    }
}

```

```

public Vector(int m)
{
    this.n=m;
    V=new double[n];
    for (int i=0;i<n;i++)
        V[i]=0.0;
}

public Vector(double [] U)
{
    this.n=U.length;
    V=U;
}

public boolean Empty()
{
    boolean ans=true;
    for (int i=0;i<n;i++)
        if (V[i]!=0.0) ans=false;
    return ans;
}

public void print()
{
    for(int i=0;i<n;i++)
        System.out.println(V[i]);
}

public double[] VectorToDouble()
{
    double [] K =new double [n];
    for (int j=0;j<n;j++)
        K[j]=V[j];
    return K;
}

public int length()
{
    double[] temp = this.VectorToDouble();
    return temp.length;
}

public double get(int i)
{
    return V[i-1];
}

public void set(int i,double s)
{
    V[i-1]=s;
}

```

```

public double min()
{
    double temp=V[0];
    for (int i=1;i<n;i++)
        temp=Math.min(temp,V[i]);
    return temp;
}

public Vector plus(Vector s)
{
    double [] temp =new double[this.length()];
    for (int i=0;i<this.length();i++)
        temp[i]=V[i]+s.get(i+1);
    return new Vector(temp);
}

public Vector minus(Vector s)
{
    double [] temp =new double[this.length()];
    for (int i=0;i<this.length();i++)
        temp[i]=V[i]-s.get(i+1);
    return new Vector(temp);
}

public Vector AddElement(double s)
{
    Vector temp =new Vector(n+1);
    for (int i=1;i<=n;i++)
        temp.set(i,V[i-1]);
    temp.set(n+1,s);
    return temp;
}

public Vector element_square()
{
    Vector temp=new Vector(n);
    for (int i=0;i<n;i++)
        temp.set(i+1,V[i]*V[i]);
    return temp;
}

public Vector concatenate(Vector s)
{
    Vector temp=new Vector(n+s.length());
    for (int i=1;i<=n;i++)
        temp.set(i,V[i-1]);
    for (int i=1;i<=s.length();i++)
        temp.set(i+n,s.get(i));
    return temp;
}

```

```

public Vector multiply(double s)
{
    double[] temp = new double[this.length()];
    for (int i=0; i<this.length();i++)
        temp[i]=s*V[i];
    return new Vector(temp);
}

public Vector multiply(Vector S)
{
    double[] temp =new double[this.length()];
    if (this.length() !=S.length())
    {
        throw new IllegalArgumentException("Vector
        lengths must be equal.");
    }
    for (int i=0; i<this.length();i++)
        temp[i]=S.get(i)*V[i];
    return new Vector(temp);
}

public Vector divide(double s)
{
    if (Math.abs(s)<0.00000001)
    {
        throw new IllegalArgumentException("Division by
        a small number.");
    }
    double[] temp = new double[this.length()];
    for (int i=0; i<this.length();i++)
        temp[i]=V[i]/s;
    return new Vector(temp);
}

public Matrix TimesTransposeOf(Vector s)
{
    Matrix temp=new Matrix(n,n);
    for (int j=0;j<n;j++)
        for (int k=0;k<n;k++)
            temp.set(j,k,V[j]*s.get(k));
    return temp;
}

public double SumElements()
{
    double sum=0.0;
    for (int i=0; i<this.length();i++)
        sum += V[i];
    return sum;
}

```

```

    public Vector getVector(int k,int r)
    {
        int j=1;
        Vector temp=new Vector(r-k+1);
        for (int i=k;i<=r;i++)
        {
            temp.set(j,V[i-1]);
            j++;
        }
        return temp;
    }

    public double DotProduct(Vector s)
    {
        return this.multiply(s).SumElements();
    }

    public double L2_norm()
    {
        double sum=0.0;
        for (int i=0;i<n;i++)
            sum +=V[i]*V[i];
        return(Math.sqrt(sum));
    }

    public double inf_norm()
    {
        double sum=Math.abs(V[0]);
        for (int i=1;i<n;i++)
            if (Math.abs(V[i])>sum) sum=Math.abs(V[i]);
        return sum;
    }
}

```

shurorder.java file

```

import Jampack.*;

public class schurorder {
    Zmat Qin,Ttemp_in,Qout,Tout;
    int[] index;

    public schurorder(Zmat Q, Zmat T, int[] indexi)
        throws JampackException
    {
        this.Qin=Q;
        this.Ttemp_in=T;
        this.index=indexi;
        cal();
    }
}

```

```

public Zmat InvGivens(Z x,Z y) throws JampackException
{
    double absx = Z.abs(x);
    double absy = Z.abs(y);
    double c, nrm;
    Z s;
    Zops zt=new Zops();
    Zmat Gtemp=new Zmat(2,2);
    if (absx == 0.0)
    {
        Gtemp.put(2,1,Z.ZERO);
        Gtemp.put(1,2,Z.ZERO);
        Gtemp.put(2,2,Z.ONE);
        Gtemp.put(1,1,Z.ONE);
    }else
    {
        nrm = Math.sqrt(absx*absx+absy*absy);
        c = absx/nrm;
        s = zt.Div(zt.Times(x,zt.Conj(y)),new Z(absx*nrm,0));
        Gtemp.put(2,1,new Z(c,0));
        Gtemp.put(1,2,new Z(c,0));
        Gtemp.put(2,2,s);
        Gtemp.put(1,1,zt.Times(new Z(-1,0),zt.Conj(s)));
    }

    return Gtemp;
}

private void cal() throws JampackException
{
    Zops zt=new Zops();
    Zmat t;
    int n,l1,l2,k,move,ix;
    n=Ttemp_in.nr;
    Tout=Ttemp_in;
    Qout=Qin;
    for (int i=1;i<n-1;i++)
    {
        move=0;
        k=i;
        for (int j=i+1;j<=n;j++)
            if (index[j-1]<index[k-1])
            {
                k=j;
                move=1;
            }
        if (move==1)
        {
            for (int l=k;l>=i+1;l--)
            {

```



```

    {
        return(Div(a,new Z(b,0)));
    }

    public Z Div(Z a, Z b)
    throws JampackException{
        Z temp=new Z();
        double avi, t, tre, tim;
        avi = abs(b);
        if (avi == 0){
            throw new JampackException
                ("Divide by zero.");
        }
        avi = 1./avi;
        tre = b.re*avi;
        tim = -b.im*avi;
        t = (a.re*tre - a.im*tim)*avi;
        temp.im = (a.im*tre + a.re*tim)*avi;
        temp.re = t;
        return temp;
    }

    public Z Minus(Z a, Z b){
        Z temp=new Z();
        temp.re = a.re - b.re;
        temp.im = a.im - b.im;
        return temp;
    }

    public Z Plus(Z a, Z b){
        Z temp=new Z();
        temp.re = a.re + b.re;
        temp.im = a.im + b.im;
        return temp;
    }
}

```

sortC.java file

```

public class sortC {
    Vector index,sorted_vec;
    int n;

    public Vector sortr(Vector m, Vector n)
    {
        Vector a=new Vector(m.VectorToDouble());
        Vector b=new Vector(n.VectorToDouble());
        double s,tr,ti,q,angle;
        int k;
    }
}

```



```

for (int i=n;i>=1;i--)
{
    s=a.get(i)*a.get(i)+b.get(i)*b.get(i);
    k=i;
    for (int j=1;j<i;j++)
    {
        q=a.get(j)*a.get(j)+b.get(j)*b.get(j);
        if (s==q)
        {
            if (b.get(j)!=0.0)
            {
                angle=Math.atan(a.get(j)/b.get(j));
                angle=angle*(a.get(j)<0.0?-1:1);
            }
        } else
        if (s<q)
        {
            s=q;
            k=j;
        }
    }
    if (k!=i)
    {
        tr=a.get(i); ti=b.get(i);
        a.set(i,a.get(k)); b.set(i,b.get(k));
        a.set(k,tr); b.set(k,ti);
    }
}
return a;
}

```

```

public Vector index(Vector a, Vector b)
{
    index=new Vector(n);
    double s;
    int last_used=-1;
    double prev=b.get(1);
    for (int i=1;i<=n;i++)
    {
        s=b.get(i);
        loop1:
        for (int j=1;j<=n;j++)
        {
            if (s==a.get(j))
            {
                if (s==prev && j>last_used)
                {
                    index.set(i,j);
                    last_used=j;
                    break loop1;
                } else if (s!=prev)

```

```

        {
            index.set(i,j);
            last_used=j;
            prev=s;
            break loop1;
        }
    }
}
return index;
}

public void calculate(Vector Real, Vector Imag)
{
    n=Real.length();
    sorted_vec=sortr(Real,Imag);
    index(Real,Imag,sorted_vecR,sorted_vecI);
}
}

```

place.java file

```

import Jama.*;
import Jampack.*;

public class place extends EleMat {
    public Matrix K;
    public Zmat Kk;
    private int nx,na,n,m,nmmp1,mp1,i,M,nn,nnx;
    private Matrix Bin,Y;
    private Zmat A,B,C1,C2,Zz,Qb,u1,u0,Bx,X,Q,R,S,Pr,Xf;
    private Zmat Bv,Bu,Bs,Yj;
    private ZVector P;
    private Vector cmplx,pr,pi;
    private Z z;
    private Zsvd svd;
    private double cnd;

    public place(Matrix Aa,Matrix Bb, ZVector Pp)
    {
        this.A=new Zmat(Aa.getArrayCopy());
        this.Bin=Bb;
        this.P=Pp;
        try {sub1();}
        catch (JampackException e)
        {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}

```

```

public void sub1() throws JampackException
{
    Bx=new Zmat(0,0);
    pi=new Vector(0);
    pr=new Vector(0);
    cmplx=new Vector(0);
    nx=A.nr;
    na=A.nc;
    n=Bin.getRowDimension();
    m=Bin.getColumnDimension();
    esort est=new esort(P);
    P=est.S;
    if (na!=nx)
    {
        System.out.println("A matrix must be square.");
        return;
    }
    else if (nx!=n)
    {
        System.out.println("B matrix must have as many rows as A.");
        return;
    }
    else if (P.length()!=nx)
    {
        System.out.println("P must have the same number
        of states as A.");
        return;
    }
    else if (nx==0)
    {
        System.out.println("Matrix A empty ");
        return;
    }
    B=new Zmat(Bin.getArrayCopy());
    M=Bin.rank();
    if (M<m)
    {
        svd= new Zsvd(B);
        Bu=(svd.U).get(1,n,1,m);
        Bs=new Zmat(svd.S);
        Bv=svd.V;
        if ((Bs.get(M+1,M+1)).re!=0.0)
        {
            System.out.println("Expected svd to return
            rank "+M+" S matrix");
            return;
        }
        B=Times.o(Bu,Bs.get(1,Bs.nr,1,M));
        m=M;
    }
    nx=0;
    i=1;
}

```

```

while (i<=n)
{
    if (P.getImag(i)!=0.0)
    {
        pr=pr.AddElement(P.getReal(i));
        pi=pi.AddElement(P.getImag(i));
        cmplx=cmplx.AddElement(1.0);
        i +=2;
    }else
    {
        pr=pr.AddElement(P.getReal(i));
        pi=pi.AddElement(0.0);
        cmplx=cmplx.AddElement(0.0);
        i++;
    }
    nx++;
}

nmmp1=n-m+1;
mp1=m+1;
Zqrd qrd = new Zqrd(B);
Qb=qrd.Q;
Zz=qrd.R;
u0=Qb.get(1,n,1,m);
u1=Qb.get(1,n,m+1,n);
Zz=Zz.get(1,m,1,m);
if (m==n)
{
    A=Minus.o(A,Elemat.diag(P.getReal()));
    i=0;
    for (int j=1;j<=nx;j++)
    {
        i++;
        if (cmplx.get(j)!=0.0)
        {
            A.put(i,i+1,z.Plus(A.get(i,i+1),new Z(pi.get(j),0.0)));
            A.put(i+1,i,z.Minus(A.get(i+1,i),new Z(pi.get(j),0.0)));
            i++;
        }
    }
    Kk=Times.o(Solve.aib(Zz,H.o(u0)),A);
} else
{
    X=stepX(u1,R,Q,qrd,svd,Y,A,Yj,Bx,S,nnx,nn,m,nx,
    n,nmmp1,pr,pi,cmplx,Pr);
    cnd=Elemat.cond(X);
    if (cnd*2.22044604925031/1.0e+16>=1.0)
    System.out.println("Can not place eigen values there");

    C1=Times.o(X,Elemat.diag(P));
    C2=new Zmat(H.o(Solve.aib(H.o(X),H.o(C1))).getRe());
}

```

```

        Kk=Times.o(Solve.aib(Zz,H.o(u0)),Minus.o(A,C2));
        K=new Matrix(Kk.getRe());
        // K hold the gain matrix
    }
}
}

```

exchange.java file

/* Once called the A,B,z matrices will hold the result If the A,B, z should be intact, then they should be copied at the constructor as follows; this.A=new Zmat(A); */

```

import Jampack.*;

public class exchange {
    int[] q;
    Zmat Ain,Bin,z;

    public exchange(Zmat A, Zmat B, Zmat z, int[] p)
    throws JampackException
    {
        this.q=p;
        this.Ain=A;
        this.Bin=B;
        this.z=z;
        cal();
    }

    public Zmat rot(Zmat x)
    {
        Zmat G=new Zmat(2,2);
        double r;
        Z zt=new Z();
        if (x.get(2,1)!=Z.ZERO)
        {
            r = Norm.fro(x);
            G.put(1,1,1,2,H.o(x));
            G.put(2,1,zt.Times(new Z(-1,0),x.get(2,1)));
            G.put(2,2, x.get(1,1));
            G=Times.o(new Z(1/r,0),G);
        }else
            G = Eye.o(2);
        return G;
    }

    public void cal() throws JampackException
    {
        boolean loop,altb;
        int hh,ii,n;
        Z zt=new Z();
    }
}

```

```

int n2 = q.length;
int[] p=new int[n2];
Zmat s,G;

for (int m=1;m<=n2;m++)
p[q[m-1]-1]=m;

for (int jj = n2;jj>=2;jj--)
{
    ii=0;
    loop=true;
    while (loop && ii<p.length)
    if (p[ii++]==jj) loop=false;
    if (ii<=jj-1)
    for (int k=ii;k<=jj-1;k++)
    {
        altb=(Z.abs(Ain.get(k+1,k+1))<Z.abs(Bin.get(k+1,k+1)));
        s =H.trans(Minus.o(Times.o(Ain.get(k+1,k+1),
        Bin.get(k,k,k,k+1)),Times.o(Bin.get(k+1,k+1),
        Ain.get(k,k,k,k+1))));

        if (s.get(1,1) !=Z.ZERO)
        {
            s = Times.o(zt.Div(Z.ONE,s.get(1,1)),s);
            G = EleMat.getMatrix(rot(s),2,1,-1,1,2,1);

            Ain.put(1,k+1,k,k+1,Times.o(Ain.get(1,k+1,k,k+1),G));
            Bin.put(1,k+1,k,k+1,Times.o(Bin.get(1,k+1,k,k+1),G));
            z.put(1,z.nr,k,k+1,Times.o(z.get(1,z.nr,k,k+1),G));
        }
        n = Ain.nc;
        if (altb)
            G = rot(Bin.get(k,k+1,k,k));
        else
            G = rot(Ain.get(k,k+1,k,k));
        Ain.put(k,k+1,k,n,Times.o(G,Ain.get(k,k+1,k,n)));
        Bin.put(k,k+1,k,n,Times.o(G,Bin.get(k,k+1,k,n)));
        Ain.put(k+1,k,Z.ZERO);
        Bin.put(k+1,k,Z.ZERO);
    }
    q=p;
    p=new int[q.length-1];
    hh=0;
    for (int mm=1;mm<=q.length-1;mm++)
    {
        if (mm==ii) hh=1;
        p[mm-1]=q[mm+hh-1];
    }
}
}
}

```

sort1.java file

```
public class sort1 {
    Vector index,sorted_vec;
    int n;

    public Vector sortr(Vector m)
    {
        Vector a=new Vector(m.VectorToDouble());
        double s,t;
        int k;

        for (int i=n;i>=1;i--)
        {
            s=a.get(i);
            k=i;
            for (int j=1;j<i;j++)
            {
                if (s<a.get(j))
                {
                    s=a.get(j);
                    k=j;
                }
            }
            if (k!=i)
            {
                t=a.get(i);
                a.set(i,s);
                a.set(k,t);
            }
        }
        return a;
    }

    public Vector index(Vector a, Vector b)
    {
        index=new Vector(n);
        double s;
        int last_used=-1;
        double prev=b.get(1);
        for (int i=1;i<=n;i++)
        {
            s=b.get(i);
            loop1:
            for (int j=1;j<=n;j++)
            {
                if (s==a.get(j))
                {
                    if (s==prev && j>last_used)
                    {
                        index.set(i,j);
                    }
                }
            }
        }
    }
}
```

```

        last_used=j;
        break loop1;
    }else if (s!=prev)
    {
        index.set(i,j);
        last_used=j;
        prev=s;
        break loop1;
    }
    }
}
return index;
}

public void calculate(Vector s)
{
    n=s.length();
    sorted_vec=sortr(s);
    index(s,sorted_vec);
}
}

```

EleMat.java file

```

// Collection of routines to do matrix operations not
// found in standard packages.

import Jama.*;
import Jampack.*;

public class EleMat {

// Matrix, vector multiplication

    public static Vector times(Matrix A, Vector V)
    {
        Vector temp=new Vector(A.getRowDimension());
        double sum;
        for (int j=0;j<A.getRowDimension();j++)
        {
            sum=0.0;
            for (int i=0;i<A.getColumnDimension();i++)
            sum +=A.get(j,i)*V.get(i+1);
            temp.set(j+1,sum);
        }
        return temp;
    }

// rcond of a matrix

```



```

    public static double rcond(Zmat A) throws JampackException
    {
        return(1/(norm1(A)*norm1(Inv.o(A))));
    }

// L-inf norm of matrices

    public static double norm1(Zmat A)
    {
        double norm=-1.0;
        double sum;
        for (int i=1;i<=A.nc;i++)
        {
            sum=0.0;
            for (int j=1;j<=A.nr;j++)
                sum +=Z.abs(A.get(j,i));
            if (sum>norm) norm=sum;
        }
        return norm;
    }

// row pivoting of the identity matrix

    public static Zmat Pivot(int[] p)
    {
        int n=p.length;
        Zmat temp1=Eye.o(n);
        Zmat temp2;
        for (int k=0;k<n;k++)
        {
            temp2=temp1.get(k+1,k+1,1,n);
            temp1.put(k+1,k+1,1,n,temp1.get(p[k]+1,p[k]+1,1,n));
            temp1.put(p[k]+1,p[k]+1,1,n,temp2);
        }
        return temp1;
    }

    public static Matrix getMatrix(Matrix x,int rs,int re,
    int ri,int cs,int ce,int ci)
    {
        Matrix temp=new Matrix((re-rs)/ri+1,(ce-cs)/ci+1);
        int rowindex=0,colindex;
        if (ri>0)
        {
            for (int i=rs-1;i<re;i+=ri)
            {
                colindex=0;
                if (ci<0)
                {
                    for (int j=cs-1;j>ce-2;j+=ci)
                    {

```



```

// Given a column matrix this will return the # of negative elements

public static int MatrixNegLength(Matrix x)
{
    // x has to be a column matrix
    int n=x.getRowDimension();
    int length=0;
    for (int i=0;i<n;i++)
        if (x.get(i,0)<0.0) length++;
    return length;
}

// Given a column matrix this will return an array of integers
// which represent a zero element by zero, negative elements by
// -1 and positive elements by 1

public static int[] signMatrix(Matrix x)
{
    // x should be a column matrix
    int n=x.getRowDimension();
    int[] V=new int[n];
    for (int i=0;i<n;i++)
        if (x.get(i,0)==0.0) V[i]=0;
        else V[i]=(x.get(i,0)<0?-1:1);
    return V;
}

// condition of a Complex Matrix

public static double cond(Zmat A) throws JampackException
{
    Zdiagmat S;
    Zsvd svd=new Zsvd(A);
    S=svd.S;
    double Max=S.get(1).re;
    double Min=S.get(1).re;
    for (int i=2;i<=S.n;i++)
    {
        Max=Math.max(Max,S.get(i).re);
        Min=Math.min(Min,S.get(i).re);
    }
    return (Max/Min);
}

// Complex conjugate of a Complex Matrix

public static Zmat conj(Zmat A)
{
    Zmat temp =new Zmat(A.nr,A.nc);
    for (int i=1;i<=A.nr;i++)

```

```

        for (int j=1;j<=A.nc;j++)
            temp.put(i,j,A.get(i,j).Conj(A.get(i,j)));
    return temp;
}

// norm of a Complex Matrix

public static double norm(Zmat A) throws JampackException
{
    Zdiagmat S;
    Zsvd svd=new Zsvd(A);
    S=svd.S;
    double t=S.get(1).re;
    for (int i=2;i<=S.n;i++)
        t=Math.max(t,S.get(i).re);
    return t;
}

// add two Complex matrices

public static Zmat AddZmat(Zmat Zm1,Zmat Zm2)
{
    int nr1,nr2,mc1,mc2;
    nr1=Zm1.nr;
    nr2=Zm2.nr;
    mc1=Zm1.nc;
    mc2=Zm2.nc;
    Zmat temp;
    if (nr1==0)
    {
        temp= new Zmat(Zm2);
    }else
    {
        if (nr1!=nr2) {System.out.println("Error No of
rows mismatch !");}
        temp= new Zmat(nr1,mc1+mc2);
        temp.put(1,nr1,1,mc1,Zm1);
        temp.put(1,nr1,mc1+1,mc1+mc2,Zm2);
    }
    return temp;
}

// Augment a matrix by putting a column into the nc th column

public static Matrix PutColumn(int nc,Matrix A,double[][] Bb)
{
    // Bb should be double[*][1]
    Matrix temp;
    Matrix B=new Matrix(Bb);
    int n=A.getRowDimension();
    int m=A.getColumnDimension();
    if (m<nc)

```

```

{
    temp=new Matrix(n,nc);
    temp.setMatrix(0,n-1,0,m-1,A);
}else
{
    temp=new Matrix(A.getArrayCopy());
}
temp.setMatrix(0,n-1,nc-1,nc-1,B);
return temp;
}

public Zmat stepX(Zmat u1,Zmat R,Zmat Q,Zqrd qrd,Zsvd svd,Matrix Y,
Zmat A, Zmat Yj,Zmat Bx,Zmat S, int nnx,int nn,int m,int nx,int n,
int nmmp1,Vector pr, Vector pi, Vector cmplx, Zmat Pr)
throws JampackException
{
    Zmat X,Xtemp;
    Zmat I=new Zmat(new Zdiagmat(n,new Z(1,0)));
    for (int i=1;i<=nx;i++)
    {
        Zmat tz1=Times.o(H.o(Minus.o(Times.o(new Z(pr.get(i-1),
pi.get(i-1)),I),Q)),Y);
        qrd = new Zqrd(tz1);
        Q =qrd.Q;
        R=qrd.R;
        Bx=AddZmat(Bx,Q.get(1,nmmp1,Q.nr,n));
    }
    nn=1;
    Xtemp=new Zmat(Bx.nr,nx);
    Y=new Matrix(Xtemp.nr,1);
    for (int i=1;i<=nx;i++)
    {
        Xtemp.put(1,Bx.nr,i,i,Bx.get(1,Bx.nr,(i-1)*m+1,
(i-1)*m+1));
        if (m>1)
        {
            lp1:
            for(int ii=2;ii<=m;ii++)
            {
                nnx=nn+(int)cmplx.get(i);
                if (nnx!=nn) Q=PutColumn(nnx,Y,Xtemp.get(1,
Xtemp.nr,i,i).getIm());
                Y=PutColumn(nn,Y,Xtemp.get(1,Xtemp.nr,i,i).getRe());
                if (Y.rank()>=nnx) break lp1;
                Xtemp=PutColumn(i,Ytemp,Bx.get(1,Bx.nr,(i-1)
*m+ii,(i-1)*m+ii));
            }
            nn=nn+1+(int)cmplx.get(i);
        }
    }
    if(m>1)
    {

```

```

        for (int k=1;k<=5;k++)
        {
            for (int i=1;i<=nx;i++)
            {
                if (i==1) S=Xtemp.get(1,Xtemp.nr,i+1,nx);
                else S=AddZmat(Xtemp.get(1,Xtemp.nr,1,i-1),
                    Xtemp.get(1,Xtemp.nr,i+1,nx));
                S=AddZmat(S, conj(S));
                svd=new Zsvd(S);
                Yj=svd.U;
                Pr=Bx.get(1,Bx.nr,(i-1)*m+1,i*m);
                Pr=Times.o(Pr,H.o(Pr));
                Xtemp=PutColumn(i+1,Xtemp,Times.o(Pr,Yj.get(1,
                    Yj.nr,n,n)));
                Xtemp=PutColumn(i,Xtemp,Times.o(new Z(1/norm(
                    Xtemp.get(1,Xtemp.nr,i,i)),0.0),Xtemp.get(1,
                    Xtemp.nr,i,i)));
            }
        }
        X=new Zmat(0,0);
        for (int i=1;i<=nx;i++)
        {
            if (cmplx.get(i)!=0.0)
            {
                X=EleMat.AddZmat(X,Xtemp.get(1,Xtemp.nr,i,i));
                X=EleMat.AddZmat(X,EleMat.conj(Xtemp.get(1,
                    Xtemp.nr,i,i)));
            }else
            {
                X=EleMat.AddZmat(X,Xtemp.get(1,Xtemp.nr,i,i));
            }
        }
        return X;
    }

// Augment a Complex matrix by putting a column into the nc th column

public static Zmat PutColumn(int nc,Zmat A,Zmat B)
{
    // B should be a Column matrix
    Zmat temp;
    int n=A.nr;
    int m=A.nc;
    if (m<nc)
    {
        temp=new Zmat(n,nc);
        temp.put(1,n,1,m,A);
    }else
    {
        temp=new Zmat(A);
    }
}

```

```

    }
    temp.put(1,n,nc,nc,B);
    return temp;
}

// Augment a Matrix by adding another as a column

public static Matrix setMColumn(Matrix A,Matrix B)
{
    Matrix temp;
    int n=A.getRowDimension();
    if (n!=B.getRowDimension())
    {
        System.out.println("\nShould have same number of Rows !");
        temp=A;
    }else
    {
        int bc=B.getColumnDimension();
        int ac=A.getColumnDimension();
        temp=new Matrix(n,(ac+bc));
        temp.setMatrix(0,n-1,0,ac-1,A);
        temp.setMatrix(0,n-1,ac,(ac+bc-1),B);
    }
    return temp;
}

// Augment a Matrix by adding another as a row

public static Matrix setMRow(Matrix A,Matrix B)
{
    Matrix temp;
    int n=A.getColumnDimension();
    if (n!=B.getColumnDimension())
    {
        System.out.println("Should have same number of Columns");
        temp=A;
    }else
    {
        int bc=B.getRowDimension();
        int ac=A.getRowDimension();
        temp=new Matrix((ac+bc),n);
        temp.setMatrix(0,ac-1,0,n-1,A);
        temp.setMatrix(ac,(ac+bc-1),0,n-1,B);
    }
    return temp;
}

// division of a scalar(s) and a Matrix(x)
// element by element division

```



```

public static Matrix rdivide(double s, Matrix x)
{
    int n=x.getRowDimension();
    int m=x.getColumnDimension();
    Matrix a=new Matrix(n,m);
    double temp;
    for (int i=0;i<n;i++)
    for (int j=0;j<m;j++)
    {
        temp=x.get(i,j);
        if (temp!=0.0)
            a.set(i,j,s/temp);
    }
    return a;
}

// for a 2x2 Matrix scale= B(1,2)/A(1,2)

public static double getbalanceScale(Matrix A)
{
    balance bla=new balance(A);
    Matrix B=bla.A;
    double val2;
    // B should be a 2x2 Matrix for lqr problem
    return (B.get(0,1)/A.get(0,1));
}

public void stepK(Matrix Q,Matrix S,Matrix A,Matrix B,Matrix R,
Matrix E,Matrix r,Zmat L,Matrix q,Vector ignore,Matrix X,Zmat aa,
Zmat bb,Matrix a, Matrix b ,Zmat z,int[] index,Vector V,
boolean noerror,boolean done,int m,int n) throws JampackException
{
    qz qzalgo;
    double Scalefact;
    a=setMRow(setMRow(setMColumn(A,setMColumn(new Matrix(n,n)
,B)),setMColumn(Q.times(-1),setMColumn(A.transpose()
.times(-1),S))),setMColumn(S.transpose(),setMColumn(
B.transpose(),R)));
    Scalefact=1;
    QRdecomp qrd=new QRdecomp(a.getMatrix(0,
a.getRowDimension()-1,n*2,n*2+m-1));
    q=qrd.getQ();
    r=qrd.getR();
    a=getMatrix(q,1,q.getRowDimension(),1,n*2+m,m+1,-1)
.transpose().times(a.getMatrix(0,
a.getRowDimension()-1,0,n*2-1));
    b=getMatrix(q,1,n*2,1,n*2+m,m+1,-1).transpose();
    qzalgo=new qz(a,b);
    z=qzalgo.Zz;
    aa=qzalgo.AA;
}

```

```

        bb=qzalgo.BB;
        L=eledivide(diag(aa),diag(bb));
        double[][] Lreal=L.getRe();
        double[] lreal=new double[Lreal.length];
        for (int i=0;i<Lreal.length;i++)
            lreal[i]=(Lreal[i][0]>0?1.0:0.0);
        V=new Vector(lreal);
        sort isort=new sort(V);
        index=isort.index;
        ignore=isort.sorted_vec;
        if (ignore.SumElements()!=n)
            if (noerror)
            {
                X=new Matrix(0,0);
                done=true;
            }else
            {
                System.out.println("Can not order eign values");
                System.exit(0);
                noerror=false;
            }
        if (!done && noerror)
        {
            exchange qzh=new exchange(aa,bb,z,index);
            // aa,bb,z returns the answer
        }

    }

// new matrix is returned with elements replaced
// by their absolute value.

public static Matrix abs(Matrix X)
{
    Matrix temp=new Matrix(X.getArrayCopy());
    for(int i=0;i<temp.getRowDimension();i++)
        for (int j=0; j<temp.getColumnDimension();j++)
            temp.set(i,j,Math.abs(temp.get(i,j)));
    return(temp);
}

// Matrix is returned with the Vector V in its main diagonal

public static Matrix diag(Vector V)
{
    int l = V.length();
    Matrix Temp = new Matrix(l,l);
    for(int i=1;i<=l;i++)
        Temp.set(i-1,i-1,V.get(i));
    return Temp;
}

```

```

// Complex Matrix is returned with the Vector V in its main diagonal
public static Zmat diag(ZVector V)
{
    int l = V.length();
    Zmat Temp = new Zmat(l,l);
    for(int i=1;i<=l;i++)
        Temp.put(i,i,V.get(i));
    return Temp;
}

// Complex Matrix is returned with the Column Matrix x
// in its main diagonal
public static Zmat diagZmat(Zmat x)
{
    if (x.nc>1) System.out.println("error not a
column matrix");
    Zmat Temp = new Zmat(x.nr,x.nr);
    for(int i=1;i<=x.nr;i++)
        Temp.put(i,i,x.get(i,1));
    return Temp;
}

// Complex Matrix is returned with the array of double
// V in its main diagonal
public static Zmat diag(double[] V)
{
    int l = V.length;
    Zmat Temp = new Zmat(l,l);
    for(int i=1;i<=l;i++)
        Temp.put(i,i,new Z(V[i-1],0.0));
    return Temp;
}

// Matrix is returned with Matrix A in its diagonal
public static Matrix diagColumnMatrix(Matrix A)
{
    if (A.getColumnDimension()>1) System.out.println("error
not a column matrix");
    int n = A.getRowDimension();
    Matrix Temp = new Matrix(n,n);
    for(int i=0;i<n;i++)
        Temp.set(i,i,A.get(i,0));
    return Temp;
}

// retrieve the max of abs(element) in a Matrix
public static double getZmatMax(Zmat x)

```

```

{
    double temp=-1E64;
    for (int i=1;i<=x.nr;i++)
    for (int j=1;j<=x.nc;j++)
    if (temp<Z.abs(x.get(i,j))) temp=Z.abs(x.get(i,j));
    return temp;
}

// return a Matrix with Vector V in its sub diagonal N

public static Matrix diag(Vector V,int N)
{
    int l= V.length();
    Matrix Temp = new Matrix(l+Math.abs(N),l+Math.abs(N));
    if (N==0)
        Temp=diag(V);
    else if (N>0)
    {
        for (int i=1;i<=l;i++)
            Temp.set(i,i+N,V.get(i));
    }
    else
    {
        for (int i=1;i<=l;i++)
            Temp.set(i-N,i,V.get(i));
    }
    return Temp;
}

// Return a Matrix with diagno1 replaced by the diagonal
// elements of array V

public static Matrix diag(double[][] V)
{
    int l=Math.min(V.length,V[0].length);
    Matrix Temp = new Matrix(l,1);
    for (int i=0;i<l;i++)
        Temp.set(i,0,V[i][i]);
    return Temp;
}

// Return a Matrix with diagno1 replaced by the diagonal
// elements of Matrix X

public static Matrix diag(Matrix X)
{
    int l = Math.min(X.getRowDimension(),X.getColumnDimension());
    Matrix Temp = new Matrix(l,1);
    for(int i=1;i<=l;i++)
        Temp.set(i-1,0,X.get(i-1,i-1));
    return Temp;
}

```

```

// Return a Complex diagonal Matrix which has
// diagonal elements of Complex Matrix X

public static Zmat diag(Zmat X)
{
    int l = Math.min(X.nr,X.nc);
    Zmat Temp = new Zmat(l,1);
    for(int i=1;i<=l;i++)
        Temp.put(i,1,X.get(i,i));
    return Temp;
}

public static Matrix diag(Matrix X,int N)
{
    Matrix Temp;
    if (N==0)
        Temp=diag(X);
    else
    {
        int l = Math.min(X.getRowDimension(),X.getColumnDimension())
            -Math.abs(N);
        Temp = new Matrix(l,1);
        if (N>0)
        {
            for (int i=1;i<=l;i++)
                Temp.set(i-1,0,X.get(i+N-1,i-1));
        }
        else
        {
            for (int i=1;i<=l;i++)
                Temp.set(i-1,0,X.get(i-1,i-N-1));
        }
    }
    return Temp;
}
}

```

lqr.java file

```

import Jama.*;
import Jampack.*;

public class lqr extends EleMat {
    public Matrix G;
    private Matrix A,B,Q,R,X;
    EigenvalueDecomposition eignv;
    Vector V;
    int n,m;
    double eps=2.220446049250313E-16;

    public lqr(Matrix a,Matrix b,Matrix q,Matrix r)

```

```

{
    this.A=a;
    this.B=b;
    this.Q=q;
    this.R=r;
    this.n=b.getRowDimension();
    this.m=b.getColumnDimension();
    if ((A.getColumnDimension()!=Q.getColumnDimension())||
        (A.getRowDimension()!=Q.getRowDimension()))
    {
        System.out.println("A and Q must be the same size ! ");
        System.exit(0);
    }
    else if ((R.getRowDimension()!=m)||
        (R.getColumnDimension()!=m))
    {
        System.out.println("R must be "+m+"x"+m+" square matrix ");
        System.exit(0);
    }
    else
    try { findk();}
    catch(JampackException Je){System.out.println(
        "LQR method Error "+Je.getMessage());}
}

public double minVecInMat(Matrix x)
{
    int n=x.getRowDimension();
    double min=1E64;
    for (int i=0;i<n;i++)
        if (x.get(i,0)<min)
            min=x.get(i,0);
    return min;
}

public double maxVecInMat(Matrix x)
{
    int n=x.getRowDimension();
    double max=-1E64;
    for (int i=0;i<n;i++)
        if (x.get(i,0)>max)
            max=x.get(i,0);
    return max;
}

public boolean isDiagonal(Matrix x)
{
    boolean Btemp=true;
    int n=x.getRowDimension();
    for (int i=0;i<n;i++)
        for (int j=0;j<n;j++)
            if ((i!=j)&&(x.get(i,j)!=0.0)) Btemp=false;
}

```

```

    return Btemp;
}

public void findk() throws JampackException
{
    Matrix S,E,U,D,Da,a,b,q,r;
    Matrix H12,H21,BU,DINV,Hh,realL;
    Zmat shur,UU,TT,z,t,aa,bb,L,T11,T21,XLt,p,uu,vv;
    Zutmat u;
    Zltmat l;
    qz qzalgo;
    qzZ qzZalgo;
    Vector Dv,ignore;
    double min,max, Scalefact,Asym,condx1;
    int leng;
    boolean noerror=true,done=false,invertR=false,isDiag;
    int[] index;
    X=new Matrix(0,0);
    Q=(Q.plus(Q.transpose())).times(0.5);
    R=(R.plus(R.transpose())).times(0.5);
    eignv=new EigenvalueDecomposition(R);
    V=new Vector(eignv.getRealEigenvalues());
    if (V.min()<=0.0)
    {
        System.out.println("The R matrix must be positive definite");
        System.out.println(" Program terminated !");
        System.exit(0);
        noerror=false;
    }else
    {

        S=new Matrix(n,m);
        E=Matrix.identity(n,n);
        // to check "can R be safely inverted ?",
        isDiag=isDiagonal(R);
        if (isDiag)
        {
            U=Matrix.identity(1,1);
            D=diag(R);
            invertR=true;
        }else
        {
            shur=new Zmat(R.getArrayCopy(),new double[n][n]);
            Schur sch=new Schur(shur);
            UU=sch.U;
            TT=sch.T;
            U=new Matrix(UU.getRe());
            D=new Matrix(TT.getRe());
            D=diag(D);
            Da=abs(D);
            min=minVecInMat(Da);
            max=maxVecInMat(Da);

```

```

    if (min<=eps*max)
    {
        noerror=false;
        System.out.println("R must be non singular");
        System.exit(0);
    }
    else
    invertR=(min>(Math.sqrt(eps)*max)?true:false);
}
if (invertR)
{
    BU=B.times(U);
    DINV=diagColumnMatrix(rdivide(1,D));
    H12=BU.times(-1.0).times(DINV.times(BU.transpose()));
    H21=Q.times(-1.0);
    double[][] vall={{0, H12.norm1()},{H21.norm1(),0}};
    Scalefact=getbalanceScale(new Matrix(vall));
    Hh=setMRow(setMColumn(A,H12.times(Scalefact)),
    setMColumn(H21.times(1/Scalefact),A.transpose()
    .times(-1.0)));
    shur=new Zmat(Hh.getArrayCopy(),new double
    [Hh.getRowDimension()][Hh.getColumnDimension()]);
    Schur sch=new Schur(shur);
    z=sch.U;
    t=sch.T;
    reall=diag(t.getRe());
    index=signMatrix(reall);
    leng=MatrixNegLength(reall);
    if (leng!=n)
    {
        if (noerror)
        {
            done=true;
            X=new Matrix(0,0);
        }
        else
        {
            noerror=false;
            System.out.println("Can not order eign values");
            System.exit(0);
        }
    }
    if (!done && noerror)
    {
        schurorder schurorder1=new schurorder(z,t,index);
    }
}
}else
stepK(Q,S,A,B,R,E,r,L,q,ignore,X,aa,bba,b,z,
V,index,noerror,done,m,n);
if (!done && noerror)
{
    // check for symmetry of solution

```



```

T11=z.get(1,n,1,n);
T21=z.get(n+1,n*2,1,n);
XLt=Times.o(H.o(X1),X2);
Asym=getZmatMax(Minus.o(XLt,H.o(XLt)))-Math.sqrt(eps);
if (Asym>0.1*getZmatMax(XLt))
{
    if (noerror)
    {
        done=true;
        X=new Matrix(0,0);
    }else
    {
        noerror=false;
        System.out.println("Can not order eign values");
        System.exit(0);
    }
}
else if (Asym>Math.sqrt(eps))
System.out.println("Solution may be inacurate");

if (!done && noerror)
{
    Zludpp luDecomp=new Zludpp(T11);
    p=Pivot(luDecomp.pvt);
    u=luDecomp.U;
    l=luDecomp.L;
    condx1=1/(Norm.fro(u)*Norm.fro(Inv.o(u)));
    if (condx1>Math.sqrt(eps))
    X=new Matrix((Times.o(Solve.bai(Solve.bai(T21,
    u),l),p)).getRe());
    else if (condx1>eps)
    {
        qzZalgo=new qzZ(T11,T21);
        // qzZ handles Complex Matrices !!
        uu=qzZalgo.A;
        vv=qzZalgo.B;
        XLt=qzZalgo.U; // use of XLt in place of u
        X=new Matrix((Times.o(XLt,Times.o(diagZmat(
        eldivide(diag(vv),diag(uu))),XLt))).getRe());

    } else if (noerror)
    {
        done=true;
        X=new Matrix(0,0);
    }else
    {
        noerror=false;
        System.out.println("Solution not finite");
        System.exit(0);
    }
}
if (!done && noerror)
{
    X=X.plus(X.transpose()).times(Scalefact/2);
}

```

```
G=R.solve(B.transpose().times(X));  
}  
}  
}  
}  
}
```

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University or Texas Tech University Health Sciences Center, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Agree (Permission is granted.)

Student Signature

Date

Disagree (Permission is not granted.)

Student Signature

Date