R Textbook Companion for Introduction to Linear Algebra by Gilbert Strang[1]

Created by
Malapati Venkata Dharun Raghava
B.Tech.
Computer Science and Engineering
Sri Venkateswara University College of Engineering
Cross-Checked by
R TBC Team

May 23, 2020

[1]Funded by a grant from the National Mission on Education through ICT - http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and R codes written in it can be downloaded from the "Textbook Companion Project" section at the website - https://r.fossee.in.

Book Description

Title: Introduction to Linear Algebra

Author: Gilbert Strang

Publisher: Wellesley - Cambridge Press, Wellesley MA USA

Edition: 4

Year: 2009

ISBN: 978-0-9802327-1-4

R numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means an R code whose theory is explained in Section 2.3 of the book.

Contents

List of R Codes

Chapter 1

Introduction to Vectors

R code Exa 1.1 Sum of two vectors

```
#Example : 1  Chapter : 1  pageno : 1
v<-matrix(c(1,1),2,1,TRUE)
w<-matrix(c(2,3),2,1,TRUE)
z<-v+w
z
```

R code Exa 1.2.a Schwartz and Traingular Inequality

```
#Example : 1.2A  Chapter : 1.2  Pageno : 17
#Finds whether Schwartz and traingular inequality
   between the given vectors are satisfied
#Finds the Cosine of the angle between the given
   vectors
v<-c(3,4)
w<-c(4,3)
dp=sum(v*w)
magn_v=sqrt(sum(v*v))
magn_w=sqrt(sum(w*w))
```

```
if(dp<=magn_v*magn_w){
  print("Schwartz inequality is satisfied for the given vectors")
}
z=v+w
magn_sum=sqrt(sum(z*z))
if(magn_sum<=magn_v+magn_w){
  print("Traingular inequality is satisfied for the given vectors")
}
print(paste("Cosine of the angle between the given vectors is ",dp/(magn_v*magn_w)))
```

R code Exa 1.2.b Unit Vector and checking orthogonaity

```
#Example : 1.2B   Chapter : 1.2    Pageno:-18
v<-c(3,4)
V<-v/sqrt(sum(v*v))
print(paste("The Unit vector in the direction of the given vector is ", V[1]," ",V[2]))
u<-c(-4,3)
if(sum(u*v)==0){
  print("The vectors u and v are perpendicular to each other")
}
U = u/sqrt(sum(u*u))
print(paste("The unit vector in the direction of u is",U[1]," ",U[2]))
```

R code Exa 1.2.c Solution of Linear System

```
#Example : 1.2C   Chapter : 1.2   Pageno : 18
A<-matrix(c(2,-1,-1,2),ncol=2)
```

```
b<-c(1,0)
x<-solve(A,b)
print("The solution of system is :")
print(x)
```

R code Exa 1.2.1 Dot Product of Vectors

```
#Example : 1     Chapter : 1.2      pgno:-11

#Computing the dotproduct of two vectors
dotproduct<-function(x,y){
  res<-0
  for(i in 1:length(x)){
    res<-res+x[i]*y[i]
  }
  res
}
v<-matrix(c(4,2),nrow=2,ncol=1,byrow = T)
w<-matrix(c(-1,2),nrow=2,ncol=1,byrow=T)
r<-dotproduct(v,w)
print(paste("Dot product of given vectors is",r))
```

R code Exa 1.2.2 Dot Product of Vectors

```
#Example : 2     Chapter : 1.2     pgno:-11

#Computing the dotproduct of two vectors
dotproduct<-function(x,y){
  dp<-sum(x*y)
  return(dp)
}
weight<-matrix(c(4,2),nrow=2,ncol=1,byrow = T)
distance<-matrix(c(-1,2),nrow=2,ncol=1,byrow=T)
```

```
center_point<-dotproduct(weight,distance)
center_point
```

R code Exa 1.2.5 Angle betwen two vectors

```
#Example : 5     Chapter : 1.2    pageno : 16

v<-c(2,1)
w<-c(1,2)
dotproduct<-sum(v*w)
magn_v<-sqrt(sum(v*v))
magn_w<-sqrt(sum(w*w))
cos_angle<-dotproduct/(magn_v*magn_w)
print(paste("Cosine of the angle between the given
    vectors is",cos_angle))
```

R code Exa 1.3.a Inverse of the matrix

```
#Example : 1.3A    Chapter : 1.3    Pageno : 27
A=matrix(c(1,0,0,-1,1,0,1,-1,1),ncol=3,byrow=T)
A1=solve(A) # to find the inverse of the matrix
A1
```

Chapter 2

Solving Linear Equations

R code Exa 2.1.a Solving the equations by Column picture

```
# Example : 2.1A  Chapter : 2.1   Pageno : 39
#Solving the equations by Column picture
#Column Picture : solution is the linear combination
    of columns of A that makes b
A=matrix(c(1,2,3,3,2,5,2,2,6),ncol=3)
b=c(-3,-2,-5)
x<-solve(A,b)
x
#if b=(4,4,8)
b=c(4,4,8)
x<-solve(A,b)
x
```

R code Exa 2.1.1 Multiplication of vector as dotproducts of rows and cols

```
#Example : 1   Chapter : 2.1  Pageno : 37
#Multiplication by rows and cols dotproduct.
multiply<-function(A,x){
```

```
  b<-c()
  for(i in 1:3){
    b<-c(b,sum(A[i,]*x[,1]))
  }
  b<-matrix(b,ncol=1)
  print(paste("Multiplying matrices by dotproduct of
      rows and cols"))
  print(A)
  print("*")
  print(x)
  print("=")
  print(b)
}


I<-matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=T)
x<-matrix(c(4,5,6),ncol=1)
A=matrix(c(1,1,1,0,0,0,0,0,0),ncol=3)
multiply(A,x)
A<-matrix(c(1,0,0,0,1,0,0,0,1),nrow=3,ncol=3,byrow=T
  )
multiply(A,x)
```

R code Exa 2.2.a Pivots and Multipliers in converting matrix to upper traingular system

```
#Example : 2.2A   Chapter : 2.2   page no :
   50
#Pivots and Multipliers in converting matrix to
   upper traingular system
matrix(c(1,-1,0,-1,2,-1,0,-1,2),ncol=3)->A
A
print(paste("First pivot is",A[1,1]))
l21<-A[2,1]/A[1,1]
print(paste("Multiplier L21 to convert the second
```

```
    row first element to 0 is",l21))
A[2,]<-A[2,]-l21*A[1,]
A
print(paste("The second pivot is ",A[2,2]))
l32<-A[3,2]/A[2,2]
A[3,]<-A[3,]-l32*A[2,]
print("The equivalent Upper traingular system for
    the matrix A is ")
A
```

R code Exa 2.3.b Multiplication with Elimination and Permutation matricies

```
#Example : 2.3b     Chapter : 2.3    Pageno : 61
#Multiplication with Elimination and Permutation
    matricies
Ab<-matrix(c(1,4,0,2,8,3,2,9,2,1,3,1),ncol=4)
E21<-matrix(c(1,0,0,-4,1,0,0,0,1),ncol=3,byrow=T)
P32<-matrix(c(1,0,0,0,0,1,0,1,0),ncol=3,byrow=T)
E21Ab<-E21%*%Ab
print(E21Ab)
P32E21Ab<-P32%*%E21Ab
print(P32E21Ab)
P32E21<-P32%*%E21
print(P32E21)
P32E21Ab<-P32E21%*%Ab
print(P32E21Ab)
#Solution for this system is
b<-P32E21Ab[,4]
P32E21Ab<-P32E21Ab[,-4]
x<-solve(P32E21Ab,b)
print(x)
```

R code Exa 2.3.c Multiplying Matrices in two different ways

```
# Example : 2.3C    Chapter : 2.3    Pageno  :  62
# Multiplying Matrices in two different ways
A<-matrix(c(3,1,2,4,5,0),ncol=2)
B<-matrix(c(2,1,4,1),ncol=2)
AB<-A%*%B
print(AB)
#Multiplying matrices A and B as Rows of A times
    columns of B as dot product
for(r in 1:dim(A)[1]){
  for(c in 1:dim(B)[2]){
    AB[r,c]<-sum(A[r,]*B[,c])
  }
}
print(AB)
#Multiplying Matrices A and B as Columns of A times
    rows of B
AB1<-matrix(A[,1],ncol=1)%*%matrix(B[1,],nrow=1)
AB2<-matrix(A[,2],ncol=1)%*%matrix(B[2,],nrow=1)
AB<-AB1+AB2
print(AB)
```

R code Exa 2.3.1 Matrix multiplication as dotproduct of row and column

```
# Example : 1     Chapter  :  2.3    page no  :  57
#Matrix multiplication as dotproduct of row and
    column
A<-matrix(c(3,4,5,6),nrow=2,ncol=2,byrow=T)
x<-c(2,1)
B<-A%*%x
B
Ax<-c()
for(i in 1:2){
  Ax<-c(Ax,sum(A[i,]*x))
```

```
}
Ax<-matrix(Ax,ncol=1)
Ax
```

R code Exa 2.3.2 Purpose of Elimination matrices

```
#Example : 2 Chapter : 2.3 pageno : 58
#Purpose of Elimination matrices
I<-matrix(c(1,0,0,0,1,0,0,0,1),ncol=3)
E31<-matrix(c(1,0,-4,0,1,0,0,0,1),ncol=3)
b<-matrix(c(1,3,9),ncol=1)
Ib<-I%*%b
print(Ib)
Eb<-E31%*%b
print(Eb)
```

R code Exa 2.4.a Square of Pascal Matrix is HyperCube Matrix

```
#Example : 2.4A Chapter : 2.4 Page no
  : 72
#Square of Pascal Matrix is HyperCube Matrix
L<-matrix(c(1,1,1,1,0,1,2,3,0,0,1,3,0,0,0,1),ncol=4)
H<-L%*%L
print("Square of Pascal Matrices is:")
print(H)
H1<-H%*%matrix(c(1,1,1,1),ncol=1)
print(H1)
```

R code Exa 2.4.b Commutative property does not hold for matrix multiplication

```
#Example : 2.4B            Chapter:2.4          Pageno  :
    74

B<-matrix(c(1,0,1,1),ncol=2)
C<-matrix(c(0,0,1,0),ncol=2)
BC<-B%*%C
CB<-C%*%B
print(BC)
print(CB)
print("Commutative property does not hold for matrix
     multiplication but by chance here it happened BC
    =CB")
```

R code Exa 2.4.c 3step paths for the given directed graph

```
#Example  :  2.4C    Chapter  :  2.4      Pageno  :
    74
#3-step paths for the given directed graph
A<-matrix(c(1,1,1,0),ncol=2)
A3<-A%*%A%*%A
print("3 step paths between each pair of nodes in
    the given directed graph is :")
print(A3)
```

R code Exa 2.4.1 Multiplication of Square matrices

```
# Example : 1           Chapter : 2.4
    Pageno : 68
#Multiplication of Square matrices
A<-matrix(c(1,2,1,-1),ncol=2)
B<-matrix(c(2,3,2,4),ncol=2)
AB<-A%*%B
print(AB)
```

R code Exa 2.4.2 Column times Row

```
#Example : 2          Chapter : 2.4          Page no :
    68
#Column times Row
A<-matrix(c(0,1,2),ncol=1)
B<-matrix(c(1,2,3),nrow=1)
AB<-A%*%B
print(AB)
```

R code Exa 2.5.a Inverse of difference matrix is Sum matrix

```
# Example : 2.5A   Chapter : 2.5   Pageno : 87
# Inverse of difference matrix is Sum matrix
A<-matrix(c(1,-1,0,0,1,-1,0,0,1),ncol=3)
A1<-solve(A)
print("Inverse of the difference matrix is singular
    ")
print(A1)
```

R code Exa 2.5.b Inverse of the given matrices

```
# Example : 2.5B   Chapter : 2.5   Pageno : 88
# Inverse of the given matrices
B<-matrix(c(4,8,3,7),ncol=2)
C<-matrix(c(6,6,6,0),ncol=2)
S<-matrix(c(1,1,1,0,1,1,0,0,1),ncol=3)
B1<-solve(B)
C1<-solve(C)
```

```
S1<-solve(S)
print("Inverses of given matrices ")
print(B1)
print(C1)
print(S1)
```

R code Exa 2.5.c Inverse of the Pascal matrix by gauss jordan elimination method

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 2.5C   Chapter : 2.5   Pageno : 88
# Inverse of the Pascal matrix by gauss jordan
    elimination method
library(pracma)
L<-matrix(c(1,1,1,1,0,1,2,3,0,0,1,3,0,0,0,1),ncol=4)
I<-eye(4)
LI<-cbind(L,I)
IL1<-rref(LI)
L1<-IL1[,c(5:8)]
print("Inverse of given Pascal matrix is")
print(L1)
```

R code Exa 2.5.2 Inverse of an Elimination Matrix

```
# Example : 2   Chapter : 2.5   Pageno : 82
```

```
# Inverse of an Elimination Matrix
E<-matrix(c(1,-5,0,0,1,0,0,0,1),ncol=3)
E1<-solve(E)
print("The inverse of the given elimination matrix is")
print(E1)
```

R code Exa 2.5.3 Inverse of product of matrices

```
# Example : 3  Chapter : 2.5  Pageno : 83
# Inverse of product of matrices
E<-matrix(c(1,-5,0,0,1,0,0,0,1),ncol=3)
E1<-solve(E)
F<-matrix(c(1,0,0,0,1,-4,0,0,1),ncol=3)
F1<-solve(F)
print(F1)
FE<-F%*%E
print(FE)
print("Inverse of FE ")
FE1<-solve(FE)
print(FE1)
print("Inverse of FE can also be E-1*F-1 ")
print(E1%*%F1)
```

R code Exa 2.5.4 Inverse of matrix by guass jordan elimination matrix

```
# Packages used : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
```

```
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 4   Chapter : 2.5   Pageno : 85
# Inverse of matrix by guass jordan elimination
    matrix
library(pracma)
A<-matrix(c(2,4,3,7),ncol=2)
I<-eye(2)
AI<-cbind(A,I)
R<-rref(AI)
X<-R[,c(3:4)]
print("Inverse of A is")
print(X)
```

R code Exa 2.5.5 Inverse of Lower traingular matrix is also a lower traingular matrix

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 5   Chapter : 2.5   Pageno : 86
# Inverse of Lower traingular matrix is also a lower
     traingular matrix
library(pracma)
L<-matrix(c(1,3,4,0,1,5,0,0,1),ncol=3)
I<-eye(3)
LI<-cbind(L,I)
R<-rref(LI)
```

```
L1<-R[,c(4:6)]
print("Inverse of L is")
print(L1)
```

R code Exa 2.6.a LU Factorisation

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 2.6A  Chapter : 2.6  Pageno : 101
# LU factorisation
library(pracma)
P<-matrix(c(1,1,1,1,1,2,3,4,1,3,6,10,1,4,10,20),ncol
    =4)
print("LU factorisation of P")
print(lu(P))
```

R code Exa 2.6.b Forward Elimination

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html
```

```
# Example : 2.6B    Chapter : 2.6    Pageno : 102
# Forward Elimination
library(pracma)
P<-matrix(c(1,1,1,1,1,2,3,4,1,3,6,10,1,4,10,20),ncol
   =4)
b<-c(1,0,0,0)
L<-lu(P)$L
U<-lu(P)$U
c<-solve(L,b)
x<-solve(U,c)
print("The solution is ")
print(x)
```

R code Exa 2.6.1 LU Factorisation

```
# Packages used  : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 1    Chapter : 2.6    Pageno : 96
# LU Factorisation
library(pracma)
A<-matrix(c(2,1,0,1,2,1,0,1,2),ncol=3)
print("LU factorisation of A is")
print(lu(A))
# The answers may vary due to rounding off values
```

R code Exa 2.6.2 LU Factorisation

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 2   Chapter : 2.6   Pageno : 96
# LU Factorisation
library(pracma)
A<-matrix(c(1,1,0,0,1,2,1,0,0,1,2,1,0,0,1,2),ncol=4)
print("LU factorisation of A is")
print(lu(A))
# The answers may vary due to rounding off values
```

R code Exa 2.6.3 Forward Elimination

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 3   Chapter : 2.6   Pageno : 98
# Forward Elimination
library(pracma)
A<-matrix(c(1,4,2,9),ncol=2)
b<-c(5,21)
```

```
L<-lu(A)$L
U<-lu(A)$U
c<-solve(L,b)
x<-solve(U,c)
print("The solution is ")
print(x)

# The answers may vary due to rounding off values
```

R code Exa 2.7.a Multiplication by Permutation matrices

```
# Example : 2.7A    Chapter : 2.7    Pageno : 114
# Multiplication by Permutation matrices
P<-matrix(c(0,0,1,1,0,0,0,1,0),ncol=3)
A<-matrix(c(1,4,5,4,2,6,5,6,3),ncol=3)
Q<-t(P)
PA<-P%*%A
PAQ<-PA%*%Q
print("P*A")
print(PA)
print("P*A*Q")
print(PAQ)
```

R code Exa 2.7.b Symmetric Factorisation

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html
```

```r

# Example : 2.7B   Chapter : 2.7   Pageno : 115
# Symmetric Factorisation
library(pracma)
A<-matrix(c(1,4,5,4,2,6,5,6,3),ncol=3)
L<-lu(A)$L
D<-lu(A)$U
for(i in 1:3){
  j<-i+1
  while(j<=3){
    D[i,j]<-0
    j<-j+1
  }
}
LT<-t(L)
print("Symmetric factorisation of A=LDLT")
print(L)
print(D)
print(LT)
print(L%*%D%*%LT)
```

R code Exa 2.7.1 Inverses and Transposes

```r
# Example : 1   Chapter : 2.7   Pageno : 108
# Inverses and Transposes
A<-matrix(c(1,6,0,1),ncol=2)
AT<-t(A)
A1<-solve(A)
print("The inverse and transpose of the matrix are "
  )
print(A1)
print(AT)
print("Transpose of A-1 is ")
print(t(A1))
print("Invere of A transpose is")
```

```
print(solve(AT))
```

R code Exa 2.7.4 Product of matrix and its trnspose gives symmetric matrix

```
# Example : 4    Chapter : 2.7    Pageno : 110
# Product of matrix and its trnspose gives symmetric matrix
R<-matrix(c(-1,0,1,-1,0,1),ncol=3)
RT<-t(R)
print("R * t(R)")
print(R%*%RT)
print("t(R) * R")
print(RT%*%R)
```

Chapter 3

Vector Spaces and Subspaces

R code Exa 3.2.a Find the Matrix having the given special solutions

```
# Example : 3.2A   Chapter : 3.2     Page No: 139
# Find the Matrix having the given special solutions
s1<-c(-3,1,0,0)
s2<-c(-2,0,-6,1)
R<-diag(4)
R<-R[-4,]
#As 1st column is pivot column it is not needed to
   change
#As 3rd column is next pivot column , row reduced
   echolon form has 1 in second row of third column
R[,3]<-R[,2]
#Two free columns are modified acording to special
   solutions
R[,2]<-c(-1*s1[1],0,0)
R[,4]<-c(-1*s2[1],-1*s2[3],0)
print("matrix having given special solutions")
print(R)
```

R code Exa 3.2.b Special solution pivot columns free columns and reduced row echelon form of the given matrix

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 3.2B   Chapter : 3.2     Page No: 140
# Find the Specialsolution , Pivotcolumns ,
    freecolumns and Reduced row echelon form for each
     given matrix

library(pracma)
solution <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
     free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
```

```
    j<-j+1
  }
}
y<-length(pivotcol)
freecol<-c(1:n)
freecol<-freecol[!freecol%in%pivotcol]
x<-length(freecol)
N<-c()
#find the basis for null space based on Row reduced echelon form of given matrix
if(y==n){
  N<-c()
}
else{
for(i in 1:x){
  temp<-c(1:n)
  for(j in 1:x){
    temp[freecol[j]]<-0
  }
  temp[freecol[i]]<-1
  temp[freecol[i]]
  for(j in 1:y){
    temp[pivotcol[j]]<-R[j,freecol[i]]*-1
  }
  N<-c(N,temp)
}
N<-matrix(N,nrow=n,ncol=x)
}
#Basis for the nullspace of given matrix
print("Special solutions are given by the basis vectors of null space")
print(N)
print("Pivot columns are")
print(pivotcol)
print("Free columns or free variables are")
print(freecol)
print("Row reduced echelon form is")
print(R)
```

```
}
A1<-matrix(c(0,0,0,0,0,0,0,0),nrow=2)
print("For the matrix A1")
solution(A1)
A2<-matrix(c(3,1,6,2),nrow=2)
print("For the matrix A2")
solution(A2)
A3<-cbind(A2,A2)
print("For the matrix A3")
solution(A3)
```

R code Exa 3.2.2 Nullspace of given Singular Matrix

```
# Packages used : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 2   Chapter : 3.2     Page No: 132
# Find the Nullspace of given singular matrix

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers of pivot columns
  freecol<-c()  #vector to store the column numbers of free columns
  i<-1
```

```
j<-1

# to find which columns are pivot and which are free
while(i<=m & j<=n){
  if(R[i,j]==1){
    pivotcol<-c(pivotcol,j)
    i<-i+1
    j<-j+1
  }
  else{
    j<-j+1
  }
}
y<-length(pivotcol)
freecol<-c(1:n)
freecol<-freecol[!freecol%in%pivotcol]
x<-length(freecol)
N<-c()
#find the basis for null space based on Row reduced echelon form of given matrix
if(y==n){
  return(N)
}
for(i in 1:x){
  temp<-c(1:n)
  for(j in 1:x){
    temp[freecol[j]]<-0
  }
  temp[freecol[i]]<-1
  temp[freecol[i]]
  for(j in 1:y){
    temp[pivotcol[j]]<-R[j,freecol[i]]*-1
  }
  N<-c(N,temp)
}
N<-matrix(N,nrow=n,ncol=x)
#Basis for the nullspace of given matrix
```

```
  return(N)
}
A<-matrix(c(1,3,2,6),nrow=2,ncol=2)
N<-nullspacebasis(A)
print("Basis vectors for nullspace of given matrix is")
N
```

R code Exa 3.2.3 Nullspaces of given three matrices

```
# Packages used  : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 3   Chapter : 3.2     Page No: 133
# Find the Nullspace of given matrices A,B,C

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers of pivot columns
  freecol<-c()  #vector to store the column numbers of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
```

```
    free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row
    reduced echelon form of given matrix
  if(y==n){
    return(N)
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  #Basis for the nullspace of given matrix
  return(N)
}
```

```
A<-matrix(c(1,3,2,8),ncol=2,nrow=2)
N<-nullspacebasis(A)
print("Basis vectors of the nullspace of matrix A is
  ")
N
B<-rbind(A,2*A)
N1<-nullspacebasis(B)
print("Basis vectors of the nullspace of matrix B is
  ")
N1
C<-cbind(A,2*A)
C
N2<-nullspacebasis(C)
print("Basis vectors of the nullspace of matrix C is
  ")
N2
```

R code Exa 3.2.4 Nullspace of given upper traingular Matrix

```
# Packages used : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 4   Chapter : 3.2     Page No: 136
# Find the Nullspace of given Upper traingular
   matrix

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
```

```
m<-nrow(A)
n<-ncol(A)
pivotcol<-c() #vector to store the column numbers of pivot columns
freecol<-c()  #vector to store the column numbers of free columns
i<-1
j<-1

# to find which columns are pivot and which are free
while(i<=m & j<=n){
  if(R[i,j]==1){
    pivotcol<-c(pivotcol,j)
    i<-i+1
    j<-j+1
  }
  else{
    j<-j+1
  }
}
y<-length(pivotcol)
freecol<-c(1:n)
freecol<-freecol[!freecol%in%pivotcol]
x<-length(freecol)
N<-c()
#find the basis for null space based on Row reduced echelon form of given matrix
if(y==n){
  return(N)
}
for(i in 1:x){
  temp<-c(1:n)
  for(j in 1:x){
    temp[freecol[j]]<-0
  }
  temp[freecol[i]]<-1
  temp[freecol[i]]
```

```
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  #Basis for the nullspace of given matrix
  return(N)
}

U<-matrix(c(1,0,5,0,7,9),nrow=2)
N<-nullspacebasis(U)
print("Basis vectors for the Nullspace of given
   upper traingular matrix is ")
N
```

R code Exa 3.3.a Row reduced echelon form rank and special solutions of given matrix

```
# Packages used  : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 3.3A   Chapter : 3.3     Page No: 149
# Find the row reduced echelon form, rank and
   special solution for given matrix

library(pracma)
solution <- function(A){
  R<-rref(A)
```

```
m<-nrow(A)
n<-ncol(A)
pivotcol<-c() #vector to store the column numbers of pivot columns
freecol<-c()  #vector to store the column numbers of free columns
i<-1
j<-1

# to find which columns are pivot and which are free
while(i<=m & j<=n){
  if(R[i,j]==1){
    pivotcol<-c(pivotcol,j)
    i<-i+1
    j<-j+1
  }
  else{
    j<-j+1
  }
}
y<-length(pivotcol)
freecol<-c(1:n)
freecol<-freecol[!freecol%in%pivotcol]
x<-length(freecol)
N<-c()
#find the basis for null space based on Row reduced echelon form of given matrix
if(y==n){
  N<-c()
}
else{
for(i in 1:x){
  temp<-c(1:n)
  for(j in 1:x){
    temp[freecol[j]]<-0
  }
  temp[freecol[i]]<-1
```

```
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  }
  print("Row reduced echelon form is")
  print(R)
  print("Rank of the Matrix is")
  print(y)
  print("Special solutions for given matrix are given by")
  print(N)
}

A<-matrix(c(1,-1,0,0,-1,2,-1,0,0,-1,2,-1,0,0,-1,1),nrow=4)
solution(A)
```

R code Exa 3.3.c Special solutions and row reduced echelon forms

```
# Packages used : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 3.3C   Chapter : 3.3     Page No: 151
# Find the special solutions and row reduced echelon forms
```

```
library(pracma)
solution <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
     free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row
     reduced echelon form of given matrix
  if(y==n){
    N<-c()
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
```

```
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  print("row reduced echelon form is")
  print(R)
  print("Special solution is ")
  print(N)
}
#c is not equal to 4
A<-matrix(c(1,3,4,2,6,8,1,3,2),ncol=3)
solution(A)
#c=4
A4<-matrix(c(1,3,4,2,6,8,1,3,4),ncol=3,nrow=3)
solution(A4)
#c is not equal to 0
B<-matrix(c(1,1,1,1),nrow=2)
solution(B)
#c = 0
B0<-matrix(c(0,0,0,0),nrow=2)
solution(B0)
```

R code Exa 3.4.a Complete solution of the given system

```
# Packages used : pracma
# To install pracma,type following in command line
  while connected to internet
# install.packages("pracma")
# package can be included by command " library(
```

```
pracma)"
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 3.4A   Chapter : 3.4     Page No: 160
# Find the Complete solution of the given system

library(pracma)
completesolution <- function(A,b){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers of pivot columns
  freecol<-c()  #vector to store the column numbers of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row reduced echelon form of given matrix
```

```
  if(y==n){
    N<-c()
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  s<-N
  Ab<-cbind(A,b)
  Rd<-rref(Ab)
  temp<-Rd[,n+1]
  p<-c(1:n)
  for(i in 1:length(freecol)){
    p[freecol[i]]<-0
  }
  for(i in 1:length(pivotcol)){
    p[pivotcol[i]]<-temp[i]
  }
  print("The special solution is")
  print(s)
  print("The particular solution is")
  print(p)
  print("Complete solution = Particular solution + Special solution")
}
A<-matrix(c(1,2,3,2,4,6,3,8,7,5,12,13),nrow=3)
b<-c(0,6,-6)
completesolution(A,b)
```

R code Exa 3.4.c Complete solution of the given system

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 3.4C   Chapter : 3.4     Page No: 162
# Find the Complete solution of the given system

library(pracma)
completesolution <- function(A,b){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
     free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
```

```
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row
     reduced echelon form of given matrix
  if(y==n){
    N<-c()
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  s<-N
  Ab<-cbind(A,b)
  Rd<-rref(Ab)
  temp<-Rd[,n+1]
  p<-c(1:n)
  for(i in 1:length(freecol)){
    p[freecol[i]]<-0
  }
  for(i in 1:length(pivotcol)){
    p[pivotcol[i]]<-temp[i]
  }
```

```
  print("The special solution is")
  print(s)
  print("The particular solution is")
  print(p)
  print("Complete solution = particular solution +
     Special solution")
}
A<-matrix(c(1,2,4,2,4,8,1,4,6,0,8,8),nrow=3)
b<-c(4,2,10)
completesolution(A,b)
```

R code Exa 3.4.2 particular solution and special solution

```
# Packages used : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 2   Chapter : 3.4     Page No: 158
# Find the particular solution and special solution
   of the given system

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
```

```
i<-1
j<-1

# to find which columns are pivot and which are free
while(i<=m & j<=n){
  if(R[i,j]==1){
    pivotcol<-c(pivotcol,j)
    i<-i+1
    j<-j+1
  }
  else{
    j<-j+1
  }
}
y<-length(pivotcol)
freecol<-c(1:n)
freecol<-freecol[!freecol%in%pivotcol]
x<-length(freecol)
N<-c()
#find the basis for null space based on Row reduced echelon form of given matrix
if(y==n){
  return(N)
}
for(i in 1:x){
  temp<-c(1:n)
  for(j in 1:x){
    temp[freecol[j]]<-0
  }
  temp[freecol[i]]<-1
  temp[freecol[i]]
  for(j in 1:y){
    temp[pivotcol[j]]<-R[j,freecol[i]]*-1
  }
  N<-c(N,temp)
}
N<-matrix(N,nrow=n,ncol=x)
```

```
  #Basis for the nullspace of given matrix
  return(N)
}
A<-matrix(c(1,1,1,2,1,-1),nrow=2)
b<-c(3,4)
s<-nullspacebasis(A)
Ab<-cbind(A,b)
Rd<-rref(Ab)
p<-Rd[,4]
p<-c(p,0)
print("Particualar solution is")
print(p)
print("special solution is ")
print(s)
print("The complete solution = particular solution +
    special solution")
```

R code Exa 3.5.1 Columns of given matrix are linearly dependent

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 1   Chapter : 3.5     Page No: 170
# Columns of given matrix are dependent
library(pracma)
A<-matrix(c(1,2,1,0,1,0,3,5,3),nrow=3)
if(Rank(A)!=ncol(A)){
  print("The columns of A are dependent")
}
```

```
print("As the columns of A are dependent ,there is a
    nonzero solution to Ax=0")
x<-matrix(c(-3,1,1),ncol=1)
print(A%*%x)
```

R code Exa 3.6.a Four Fundamental Spaces of given matrix

```
# Packages used : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 3.6A   Chapter : 3.6     Page No: 190
# Four Fundamental Spaces of given matrix

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
     free
  while(i<=m & j<=n){
    if(R[i,j]==1){
```

```
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row reduced echelon form of given matrix
  if(y==n){
    return(N)
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  #Basis for the nullspace of given matrix
  return(N)
}
l<-matrix(c(1,2,5,0,1,0,0,0,1),ncol=3)
u<-matrix(c(1,0,0,3,0,0,0,1,0,5,6,0),ncol=4)
A<-l%*%u
print("Row space Basis of A")
```

```
print(u[1,])
print(u[2,])
print("COlumn space Basis of A ")
print(l[,1])
print(l[,2])
print("Null space Basis of A")
N<-nullspacebasis(A)
print(N)
print("Null space Basis of A transpose")
NT<-nullspacebasis(t(A))
print(NT)
```

R code Exa 3.6.1 Dimensions and rank of matrix

```
# Packages used : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 1   Chapter : 3.6      Page No: 188
# Dimensions and rank of matrix
library(pracma)
solution<-function(A){
  print(paste("Number of rows , m=",nrow(A)))
  print(paste("Number of columns,n=",ncol(A)))
  print(paste("Rank of the given matrix",Rank(A)))
}
A<-matrix(c(1,2,3),nrow=1)
solution(A)
```

R code Exa 3.6.2 Dimensions and rank of matrix

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 2   Chapter : 3.6     Page No: 188
# Dimensions and rank of matrix
library(pracma)
solution<-function(A){
  print(paste("Number of rows , m=",nrow(A)))
  print(paste("Number of columns,n=",ncol(A)))
  print(paste("Rank of the given matrix",Rank(A)))
}
A<-matrix(c(1,2,2,4,3,6),nrow=2)
solution(A)
```

Chapter 4

Orthogonality

R code Exa 4.1.a Dimensions of the subspaces in the given space

```
# Example : 4.1A    Chapter : 4.1       Page No: 201
# Dimensions of the subspaces in the given space
dim_R<-9
dim_S<-6
print("Possible dimensions of the subspaces
   orthogonal to S")
x<-dim_R-dim_S
orthogonal_dimensions<-c(0:x)
print(orthogonal_dimensions)
print(paste("possible dimensions of orthogonal
   complement subspaces to S",dim_R-dim_S))
print(paste("The smallest matrix A in S is ",dim_S,"
    by ",dim_R))
print(paste("The Null space matrix N is ",dim_R," by
    ",dim_R-dim_S))
```

R code Exa 4.1.b Null space Basis of a plane subspace

```
# Packages used  : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 4.1B   Chapter : 4.1     Page No: 201
# Null space Basis of a plane subspace
library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers of pivot columns
  freecol<-c()  #vector to store the column numbers of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
```

```
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row reduced echelon form of given matrix
  if(y==n){
    return(N)
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  #Basis for the nullspace of given matrix
  return(N)
}

A<-matrix(c(1,-3,-4),ncol=3)
print("Given Plane x-3y-4z=0 is a null space of following 1*3 matrix")
print(A)
#to make matrix copatible for our function
A<-rbind(A,c(0,0,0),c(0,0,0))
N<-nullspacebasis(A)
print("SPecial solutions or nullspace basis of given plane subspace is")
print(N)
A<-A[-c(2,3),]
print("Row space is ")
print(A)
temp<-cbind(N,c(1,-3,-4))
```

```
x<-c(1,1,-1)
print("vector 6,4,5 is split into vn + vs as 1 of
   each vector in nullspace basis and −1 of rowspace
    basis")
v<-temp%*%x
print(v)
```

R code Exa 4.1.3 Rows of matrix are perpendicualr to the vectors in nullspace

```
# Example : 3   Chapter : 4.1      Page No: 197
# Rows of given matrix are perpendicular to the
   vector in nullspace
A<-matrix(c(1,5,3,2,4,7),nrow=2)
x<-c(1,1,-1)
for(i in 1:nrow(A)){
  dot_product<-sum(A[i,]*x)
  if(dot_product==0){
    print(paste("Row ",i,"is perpendicular to x"))
  }
}
```

R code Exa 4.2.a Projection onto the line and the plane

```
# Example : 4.2A   Chapter : 4.2      Page No: 213
# Projection onto the line and onto the plane

projection_line<-function(a,b){
  p<-((sum(a*b))/(sum(a*a)))*a
  e<-b-p
  print("The projection vector is ")
  print(p)
  print("The error vector is ")
  print(e)
```

```
}

projection_plane<-function(A,b){
  b<-matrix(c(b),ncol=1)
  ATA<-t(A)%*%A
  ATA1<-solve(ATA)
  P<-A%*%ATA1
  P<-P%*%t(A)
  p<-P%*%b
  e<-b-p
  print("The projection vector is ")
  print(p)
  print("The error vector is ")
  print(e)
}

b<-c(3,4,4)
a<-c(2,2,1)
projection_line(a,b)
A<-matrix(c(a,1,0,0),ncol=2)
projection_plane(A,b)
#The answer may slightly vary due to rounding off
   values
```

R code Exa 4.2.b Best possible solution

```
# Example : 4.2B   Chapter : 4.2      Page No: 213
# Find best Possible solution

solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  x<-solve(ATA,ATb)
  return(x)
}
```

```
A<-matrix(c(1,1,1),ncol=1)
b<-matrix(c(70,80,120),ncol=1)
x<-solution(A,b)
print("The best possible solution is ")
print(x)
```

R code Exa 4.2.1 Projection of vector onto line

```
# Example : 1    Chapter : 4.2      Page No: 208
# Projection of the vector onto line

#Answers to this problem are displayed in the form
    of x/y in textbook
#Here the same answers are in decimal formats

projection<-function(b,a){
  xhat<-(sum(a*b))/(sum(a*a))
  p<-xhat*a
  return(p)
}
b<-c(1,1,1)
a<-c(1,2,2)
p<-projection(b,a)
print("The projection vector p i.e., b on a is ")
print(p)
e<-b-p
print("The error vector is ")
print(e)
if(sum(e*a)==0){
  print("Vector e is perpendicular to a")
}
#The  answer may slightly vary due to  rounding off
    values
```

R code Exa 4.2.2 Projection Matrix of the line

```
# Example : 2   Chapter : 4.2     Page No: 209
# Find the projection matrix onto the line

projection_matrix<-function(a){
  a<-matrix(c(a),ncol=1)
  P<-a%*%t(a)
  temp<-t(a)%*%a
  temp<-1/temp
  t<-temp[1,1]
  P<-t*P
  return(P)
}
a<-c(1,2,2)
P<-projection_matrix(a)
print("The projection matrix is ")
print(P)
#The  answer may slightly vary due to  rounding off
   values
```

R code Exa 4.2.3 Best possible solution projection vector and Projection Matrix

```
# Example : 3   Chapter : 4.2     Page No: 211
# Find the best possible solution , projection vector
    and projection matrix

solution<-function(A,b){
  ATA<-t(A)%*%A
  b<-matrix(c(b),ncol=1)
  ATb<-t(A)%*%b
```

```
  xhat<-solve(ATA,ATb)
  p<-A%*%xhat
  e<-b-p
  ATA1<-solve(ATA)
  P<-A%*%ATA1
  P<-P%*%t(A)
  print("The best possible solution is ")
  print(xhat)
  print("The projection vector is ")
  print(p)
  print("The error vector e is")
  print(e)
  print("The projection matrix is ")
  print(P)
}

A<-matrix(c(1,1,1,0,1,2),ncol=2)
b<-c(6,0,0)
solution(A,b)
#The  answer may slightly vary due to  rounding off
   values
```

R code Exa 4.3.a Fit a straight line

```
# Example : 4.3A   Chapter : 4.3     Page No: 225
# Fit a straight line

#1,2,3 solutions can be done without any need of
   computation
solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  xhat<-solve(ATA,ATb)
  return(xhat)
}
```

```
fit_line<-function(D){
  num_of_points<-nrow(D)
  t<-c()
  for(i in 1:num_of_points){
    t<-c(t,1)
  }
  t<-c(t,D[,1])
  A<-matrix(c(t),ncol=2)
  b<-D[,2]
  b<-matrix(c(b),ncol=1)
  x<-solution(A,b)# The system has no solution , we need to find the best solution
  return(x)
}
t<-c(1:10)
b<-c(0,0,0,0,0,0,0,0,0,40)
Data<-matrix(c(t,b),ncol=2)
x<-fit_line(Data)
C<-x[1]
D<-x[2]
print(paste("The best straight line is b= ",C," + ",D,"t"))
#The  answer may slightly vary due to  rounding off values.
```

R code Exa 4.3.b Fit a Parabola

```
# Example : 4.3B   Chapter : 4.3      Page No: 226
# Fit a Parabola
solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  xhat<-solve(ATA,ATb)
  return(xhat)
}
```

```
fit_parabola<-function(D){
  num_of_points<-nrow(D)
  t<-c()
  for(i in 1:num_of_points){
    t<-c(t,1)
  }
  t<-c(t,D[,1])
  t<-c(t,D[,1]*D[,1])
  A<-matrix(c(t),ncol=3)
  b<-D[,2]
  b<-matrix(c(b),ncol=1)
  x<-solution(A,b)# The system has no solution , we need to find the best solution
  return(x)
}
t<-c(-2,-1,0,1,2)
b<-c(0,0,1,0,0)
Data<-matrix(c(t,b),ncol=2)
x<-fit_parabola(Data)
C<-x[1]
D<-x[2]
E<-x[3]
print(paste("The best Parabola that fitt in is b= ",C,"+",D,"t+",E,"t2"))
#The answer may slightly vary due to rounding off values.
```

R code Exa 4.3.1 Fit a straight line

```
# Example : 1   Chapter : 4.3     Page No: 218
# Fit a straight line
solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  xhat<-solve(ATA,ATb)
```

```
  return(xhat)
}
fit_line<-function(D){
  num_of_points<-nrow(D)
  t<-c()
  for(i in 1:num_of_points){
    t<-c(t,1)
  }
  t<-c(t,D[,1])
  A<-matrix(c(t),ncol=2)
  b<-D[,2]
  b<-matrix(c(b),ncol=1)
  x<-solution(A,b)# The system has no solution , we
     need to find the best solution
  return(x)
}
Data<-matrix(c(0,6,1,0,2,0),ncol=2,byrow=T)
x<-fit_line(Data)
C<-x[1]
D<-x[2]
print(paste("The best straight line is b= ",C," + ",
   D,"t"))
```

R code Exa 4.3.2 Fit a straight line

```
# Example : 2    Chapter : 4.3      Page No: 222
# Fit a straight line
solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  xhat<-solve(ATA,ATb)
  return(xhat)
}
fit_line<-function(D){
  num_of_points<-nrow(D)
```

```
  t<-c()
  for(i in 1:num_of_points){
    t<-c(t,1)
  }
  t<-c(t,D[,1])
  A<-matrix(c(t),ncol=2)
  b<-D[,2]
  b<-matrix(c(b),ncol=1)
  x<-solution(A,b)# The system has no solution , we
     need to find the best solution
  return(x)
}
Data<-matrix(c(-2,0,2,1,2,4),ncol=2)
x<-fit_line(Data)
C<-x[1]
D<-x[2]
print(paste("The best straight line is b= ",C," + ",
   D,"t"))
#The  answer may slightly vary due to  rounding off
   values.
```

R code Exa 4.3.3 Fit a Parabola

```
# Example : 3   Chapter : 4.3      Page No: 224
# Fit a Parabola
solution<-function(A,b){
  ATA<-t(A)%*%A
  ATb<-t(A)%*%b
  xhat<-solve(ATA,ATb)
  return(xhat)
}
fit_parabola<-function(D){
  num_of_points<-nrow(D)
  t<-c()
  for(i in 1:num_of_points){
```

```
    t<-c(t,1)
  }
  t<-c(t,D[,1])
  t<-c(t,D[,1]*D[,1])
  A<-matrix(c(t),ncol=3)
  b<-D[,2]
  b<-matrix(c(b),ncol=1)
  x<-solution(A,b)# The system has no solution , we
     need to find the best solution
  return(x)
}
Data<-matrix(c(0,1,2,6,0,0),ncol=2)
x<-fit_parabola(Data)
C<-x[1]
D<-x[2]
E<-x[3]
print(paste("The best Parabola that fitt in is b= ",
   C,"+",D,"t+",E,"t2"))
```

R code Exa 4.4.4 Projections of vector onto the line plane if the basis are given as orthonormal vectors

```
# Example : 4   Chapter : 4.4      Page No: 233
# Projections of the vector onto line,plane if basis
    are given as orthonormal vectors
Q<-matrix(c(-1,2,2,2,-1,2,2,2,-1),ncol=3)
Q<-(1/3)*Q
q1<-Q[,1]
q2<-Q[,2]
q3<-Q[,3]
b<-c(0,0,1)
p1<-sum(q1*b)*q1
p2<-sum(q2*b)*q2
p3<-sum(q3*b)*q3
print("Projection of b onto q1")
```

```
print(p1)
print("Projection of b onto q2")
print(p2)
print("Projection of b onto q3")
print(p3)
print("Projection of b onto plane of q1 and q2")
print(p1+p2)
print("Projection of b onto space of q1,q2, and q3")
print(p1+p2+p3) # same as vector b
```

R code Exa 4.4.5 Gram Schmidt method to convert matrix to its orthogonal form

```
# Example : 5   Chapter : 4.4     Page No: 233
# Gram-Schmidt method to convert matrix into its orthogonal form
magnitude<-function(a){
  x<-0
  for(i in 1:length(a)){
    x<-x+a[i]*a[i]
  }
  x<-sqrt(x)
  return(x)
}
a<-c(1,-1,0)
b<-c(2,0,-2)
c<-c(3,-3,3)
A<-a
B<-b-(sum(A*b)/sum(A*A))*A
C<-c-(sum(A*c)/sum(A*A))*A-(sum(B*c)/sum(B*B))*B
print("Orthogonal vectors corresponding to a,b,c are")
print(A)
print(B)
print(C)
```

```
q1<-(1/magnitude(A))*A
q2<-(1/magnitude(B))*B
q3<-(1/magnitude(C))*C
Q<-matrix(c(q1,q2,q3),ncol=3)
print("Orthogonal matrix with orthonormal vectors of
    a,b,c ")
print(Q)
#The  answer may slightly vary due to  rounding off
    values
```

Chapter 5

Determinants

R code Exa 5.2.1 Determinant of matrix by Product of pivots

```
# Example : 1    Chapter : 5.2      Page No: 255
# Determinant of matrices by multiplying pivots
A<-matrix(c(0,0,4,0,2,5,1,3,6),ncol=3)
P<-matrix(c(0,0,1,0,1,0,1,0,0),ncol=3)
PA<-P%*%A
print(PA)
detA<--1*PA[1,1]*PA[2,2]*PA[3,3]
print(detA)
det(A)
print("detA is the determinant of the given matrix")
```

R code Exa 5.2.5 Determinants of matrices

```
# Example : 5    Chapter : 5.2      Page No: 259
# Determinants of matrices
A4<-matrix(c(0,1,0,0,1,0,1,0,0,1,0,1,0,0,1,0),ncol
    =4)
P4<-matrix(c(0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0),ncol
    =4)
```

```
print(det(A4))
print(det(P4))
```

R code Exa 5.2.7 Determinants of matrices

```
# Example : 7   Chapter : 5.2     Page No: 261
# Determinants of matrices
B4<-matrix(c(1,-1,0,0,-1,2,-1,0,0,-1,2,-1,0,0,-1,2),
   ncol=4)
print(det(B4))
```

R code Exa 5.3.a Null space of matrix is the transpose of cofactor matrix

```
# Example : 5.3A   Chapter : 5.3     Page No: 277
# Nullspace of matrix as transpose of Cofactor
   matrix
nullspacebasis<-function(A){
  C<-matrix(c(1:9),ncol=3)
  for(i in 1:3){
    for(j in 1:3){
      if((i+j)%%2==0){
        x<-1
      }
      else{
        x<--1
      }
      C[i,j]<-x*det(A[-i,-j])
    }
  }
  C<-t(C)
  return(C)
}

```

```
A1<-matrix(c(1,2,2,4,3,2,7,9,8),ncol=3)
N1<-nullspacebasis(A1)
print("The null space basis are given by columns of
    transpose of cofactor matrix")
print("Null space of A1 is")
print(N1)
A2<-matrix(c(1,1,1,1,1,1,2,1,1),ncol=3)
N2<-nullspacebasis(A2)
print("Null space of A2 is ")
print(N2)
```

R code Exa 5.3.b Solve by Crammers rule and inverse of matrix

```
# Example : 5.3B    Chapter : 5.3       Page No: 278
# Solve by crammers rule and inverse of the matrix

solve_by_crammersrule<-function(A,b){
  B1<-A
  B2<-A
  B3<-A
  B1[,1]<-b
  B2[,2]<-b
  B3[,3]<-b
  x1<-det(B1)/det(A)
  x2<-det(B2)/det(A)
  x3<-det(B3)/det(A)
  x<-c(x1,x2,x3)
  return(x)
}
inverse<-function(A){
  C<-matrix(c(1:9),ncol=3)
  for(i in 1:3){
    for(j in 1:3){
      if((i+j)%%2==0){
        x<-1
```

```
        }
        else{
          x<--1
        }
        C[i,j]<-x*det(A[-i,-j])
      }
    }
    CT<-t(C)
    A1<-(1/det(A))*CT
    return(A1)
}

A<-matrix(c(2,1,5,6,4,9,2,2,0),ncol=3)
b<-c(0,0,1)
x<-solve_by_crammersrule(A,b)
print("x is ")
print(x)
A1<-inverse(A)
print("Inverse of A is ")
print(A1)
print("A * inverse of A is Identity matrix")
I<-A%*%A1
print(I)
#The  answer may slightly vary due to  rounding off values
```

R code Exa 5.3.1 Crammers rule for solving system of equations

```
# Example : 1   Chapter : 5.3     Page No: 269
# Cramers rule for solving system of equations
A<-matrix(c(3,5,4,6),ncol=2)
b<-c(2,4)
B1<-A
B1[,1]<-b
B2<-A
```

```
B2[,2]<-b
x1<-det(B1)/det(A)
x2<-det(B2)/det(A)
print("SOlution is ")
print(x1)
print(x2)
```

R code Exa 5.3.3 The inverse of triangular matrix is triangular matrix

```
# Example : 3    Chapter : 5.3      Page No: 271
# The inverse of the traingular matrix is traingular
A<-matrix(c(1,1,1,1,0,1,1,1,0,0,1,1,0,0,0,1),ncol=4)
A1<-solve(A)
print("The inverse of A is")
print(A1)
```

R code Exa 5.3.7 Cross product of vectors

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 7    Chapter : 5.3      Page No: 275
# CrossProduct of Vectors
library(pracma)
u<-c(3,2,0)
v<-c(1,4,0)
cp<-cross(u,v)
```

```
print("u * v is ")
print(cp)
```

R code Exa 5.3.8 Cross product of vectors

```
# Packages used  : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 8   Chapter : 5.3     Page No: 276
# CrossProduct of Vectors

library(pracma)
u<-c(1,1,1)
v<-c(1,1,2)
cp<-cross(u,v)
print("u * v is ")
print(cp)
```

R code Exa 5.3.9 Right hand rule

```
# Packages used  : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
```

```
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 9    Chapter : 5.3      Page No: 276
# The right hand rule - cross product of x-axis and y-axis is z-axis

library(pracma)
u<-c(1,0,0)
v<-c(0,1,0)
cp<-cross(u,v)
print("u * v is ")
print(cp)
```

Chapter 6

Eigen Values and Eigen Vectors

R code Exa 6.1.a Eigen values and eigen vectors

```r
# Example : 6.1A   Chapter : 6.1     Page No: 291
# Eigen values and eigen vectors
solution<-function(A){
  sol<-eigen(A)
  lambda<-sol$values
  x<-sol$vectors #these are normalised eigen vectors
  #to get the eigen vectors as in texxt book multiply these normalised vectors with scalars
  x[,1]<-x[,1]*(1/x[1,1])
  x[,2]<-x[,2]*(1/x[1,2])
  print("The eigen values of the matrix are")
  print(lambda)
  print("The eigen vecotrs of the matrix respective to above eigen values are")
  print(x)
}

A<-matrix(c(2,-1,-1,2),ncol=2)
print("For A")
solution(A)
A2<-A%*%A
```

```
print("For square of A")
solution(A2)
A1<-solve(A)
print("For inverse of A")
solution(A1)
I<-matrix(c(1,0,0,1),ncol=2)
A4I<-A+4*I
print("For A+4I")
solution(A4I)
#The  answer may slightly vary due to  rounding off
   values
#The answers provided in the text book may vary
   because of the computation process
#Both answers are correct , here it is taken -Ax+b=0
    , In the text book it is considered as Ax-b=0
```

R code Exa 6.1.b Eigen values and eigen vectors

```
# Example : 6.1B   Chapter : 6.1      Page No: 292
# Eigen values and eigen vectors
solution<-function(A){
  sol<-eigen(A)
  lambda<-round(sol$values)
  x<-sol$vectors #these are normalised eigen vectors
  #to get eigen vectors in text book multiply
     normalised eigen vectors with scalars
  x[,1]<-x[,1]*(1/x[1,1])
  x[,2]<-round(x[,2]*(1/x[1,2]))
  x[,3]<-x[,3]*(1/x[1,3])
  print("The eigen values of the matrix are")
  print(lambda)
  print("The eigen vectors of the matrix respective
     to above eigen values are")
  print(x)
}
```

```
A<-matrix(c(1,-1,0,-1,2,-1,0,-1,1),ncol=3)
solution(A)
#The  answer may slightly vary due to  rounding off values
```

R code Exa 6.1.1 Eigen values and eigen vectors

```
# Example : 1   Chapter : 6.1      Page No: 284
# Eigen values and eigen vectors
A<-matrix(c(0.8,0.2,0.3,0.7),ncol=2)
sol<-eigen(A)
lambda<-sol$values
x<-sol$vectors
print("The eigen values of the matrix are")
print(lambda)
print("The eigen vectors of the matrix in normalised form are")
print(x)
#to get eigen vectors in the textbook multiply normalised vectors by scalars
x[,1]<-x[,1]*(0.6/x[1,1])
x[,2]<-x[,2]*(1/x[1,2])
print("Eigen vectors with respect to the above eigen values respectively are")
print(x)
print(A%*%x)
#The  answer may slightly vary due to  rounding off values
#The answers provided in the text book may vary because of the computation process
#Both answers are correct , here it is taken -Ax+b=0 , In the text book it is considered as Ax-b=0
```

R code Exa 6.1.2 Eigen values and eigen vectors of Projection matrix

```
# Example : 2   Chapter : 6.1     Page No: 285
# Eigen values and eigen vectors of Projection
    matrix
A<-matrix(c(0.5,0.5,0.5,0.5),ncol=2)
sol<-eigen(A)
lambda<-sol$values
x<-sol$vectors #These are normalised eigen vectors
#to get eigen vectors in text book multiply them
    with scalars
x[,1]<-x[,1]*(1/x[1,1])
x[,2]<-x[,2]*(1/x[1,2])
print("The eigen values of the matrix are")
print(lambda)
print("The eigen vectors of the matrix are")
print(x)
```

R code Exa 6.1.3 Eigen values and eigen vectors of Reflection matrix

```
# Example : 3   Chapter : 6.1     Page No: 286
# Eigen values and eigen vectors of Reflection
    matrix
A<-matrix(c(0,1,1,0),ncol=2)
sol<-eigen(A)
lambda<-sol$values
print("The eigen values of the matrix are")
print(lambda)
#The  answer may slightly vary due to  rounding off
    values
```

R code Exa 6.1.4 Eigen values and eigen vectors of Singular matrix

```
# Packages used  : pracma
# To install pracma,type following in command line
    while connected to internet
# install.packages("pracma")
# package can be included by command " library(
    pracma) "
# for more information about pracma visit https://
    cran.r-project.org/web/packages/pracma/index.html

# Example : 4   Chapter : 6.1     Page No: 287
# Eigen values and eigen vectors of Singualar matrix

library(pracma)
nullspacebasis <- function(A){
  R<-rref(A)
  m<-nrow(A)
  n<-ncol(A)
  pivotcol<-c() #vector to store the column numbers
     of pivot columns
  freecol<-c()  #vector to store the column numbers
     of free columns
  i<-1
  j<-1

  # to find which columns are pivot and which are
     free
  while(i<=m & j<=n){
    if(R[i,j]==1){
      pivotcol<-c(pivotcol,j)
      i<-i+1
      j<-j+1
    }
```

```
    else{
      j<-j+1
    }
  }
  y<-length(pivotcol)
  freecol<-c(1:n)
  freecol<-freecol[!freecol%in%pivotcol]
  x<-length(freecol)
  N<-c()
  #find the basis for null space based on Row reduced echelon form of given matrix
  if(y==n){
    return(N)
  }
  for(i in 1:x){
    temp<-c(1:n)
    for(j in 1:x){
      temp[freecol[j]]<-0
    }
    temp[freecol[i]]<-1
    temp[freecol[i]]
    for(j in 1:y){
      temp[pivotcol[j]]<-R[j,freecol[i]]*-1
    }
    N<-c(N,temp)
  }
  N<-matrix(N,nrow=n,ncol=x)
  #Basis for the nullspace of given matrix
  return(N)
}
A<-matrix(c(1,2,2,4),ncol=2)
sol<-eigen(A)
print(sol)
lambda<-sol$values
print("The eigen values of the matrix are")
print(lambda)
I<-matrix(c(1,0,0,1),ncol=2)
E1<-A-lambda[1]*I
```

```
E1
rref(E1)
x1<-nullspacebasis(E1)
E2<-A-lambda[2]*I
rref(E2)
x2<-nullspacebasis(E2)
print("The eigen vectors of the matrix in normalized
    form are")
print(x1)
print(x2)
#to get eigen vectors in the textbook multiply
   normalised vectors by scalars
x1<-2*x1
x2<--1*x2
print("The eigen vectors of the matrix are ")
print(x2)
print(x1)
#The answer may slightly vary due to rounding off
   values
#The answers provided in the text book may vary
   because of the computation process
#Both answers are correct , here it is taken -Ax+b=0
    , In the text book it is considered as Ax-b=0
```

R code Exa 6.2.b Inverse Eigen values Determinant and eigen vector matrix

```
# Packages used : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html
```

```
# Example : 6.2B    Chapter : 6.2       Page No: 306
# Inverse Eigen values Determinant and eigen vector
  matrix
library(pracma)
A<-5*eye(4)-ones(4)
eigenvalues<-eigen(A)$values
print("Eigen Values of A ")
print(eigenvalues)
A1<-solve(A)
print("Inverse of A ")
print(A1)
eigenvalues1<-eigen(A1)$value
print("Eigen Values of A inverse ")
print(eigenvalues1)
detA<-det(A)
print("Determinant of A")
print(detA)
x<-eigen(A)$vectors[,4] #normalized  eigen vector of
   eigen value=1
#toget eigen vector in text book
x<-x*(1/x[1])
print("Eigen vector of A for eigen value 1")
print(x)
print("The other eigen vectors are perpendicular to
  x since A is Symmetric so eigen vector matrix
  contains x with different signs as follows")
S<-matrix(c(1,-1,1,-1,1,1,-1,-1,1,-1,-1,1,x),ncol=4)
print(S)
print("To get normalized matrix multiply by
  magnitude of vectors which is same for all and is
   0.5")
S<-0.5*S
print(S) # The eigen vectors are with respect to
  5,5,5,1
```

R code Exa 6.2.1 Diagonilizing a Matrix and Power of matrix computed from power of its diagonal matrix

```
# Example : 1   Chapter : 6.2      Page No: 299
# Diagonilizing a Matrix and Power of matrix computed from power of its diagonal matrix
A<-matrix(c(1,0,5,6),ncol=2)
lambda<-eigen(A)$values
S<-eigen(A)$vectors
S<-round(S)
S1<-solve(S)
Diag_matrix<-round(S1%*%A%*%S)
print("Diagonal Matrix is ")
print(Diag_matrix)
A2<-A%*%A
print("The square of matrix")
print(A2)
print("The Power of matrix can also be computed from its diagonal matrix as follows")
A2_diag<-S%*%Diag_matrix%*%Diag_matrix%*%S1
print("The square of matrix computed from its diagonal matrix")
print(A2_diag)
#The  answer may slightly vary due to  rounding off values
#The answers provided in the text book may vary because of the computation process
#Both answers are correct , here it is taken -Ax+b=0 , In the text book it is considered as Ax-b=0
```

R code Exa 6.2.2 The diagonal matrix of any matrix contains the eigen values in its main diagonal

```
# Example : 2    Chapter : 6.2       Page No: 300
# The diagonal matrix of any matrix contains the
   eigen values in its main diagonal
A<-matrix(c(0.8,0.2,0.3,0.7),ncol=2)
lambda<-eigen(A)$values
print(lambda)
S<-eigen(A)$vectors #Normlised eigen vectors , Answer
    can also be validated with normalised eigen
   vectors
#to get eigen vector matrix in text book
S[,1]<-S[,1]*(0.6/S[1,1])
S[,2]<-S[,2]*(1/S[1,2])
S1<-solve(S)
Diag_matrix<-diag(2)*lambda
print("S*diag_matrix(A)*S-1 is A")
print(S%*%Diag_matrix%*%S1)
#The  answer may slightly vary due to  rounding off
   values
#The answers provided in the text book may vary
   because of the computation process
#Both answers are correct , here it is taken -Ax+b=0
    , In the text book it is considered as Ax-b=0
```

R code Exa 6.3.b Eigen values and eigen vectors

```
# Example : 6.3B    Chapter : 6.3       Page No: 322
# Eigen Values and Eigen vectors of A
A<-matrix(c(-2,1,0,1,-2,1,0,1,-2),ncol=3)
eigenvalues<-eigen(A)$values
x<-eigen(A)$vectors
print("Eigen values of A are ")
print(eigenvalues)
print("Eigen vectors of A in normalised form")
print(x)
#to get eigen vectors in the textbook multiply
```

```
    normalised vectors by scalars
x[,1]<-x[,1]*(1/x[1,1])
x[,2]<-x[,2]*(1/x[1,2])
x[,3]<-x[,3]*(1/x[1,3])
print("Eigen vectors with respect to the above eigen
     values respectively are")
print(x)
#The  answer may slightly vary due to  rounding off
    values
#The answers provided in the text book may vary
    because of the computation process
#Both answers are correct , here it is taken -Ax+b=0
     , In the text book it is considered as Ax-b=0
```

R code Exa 6.3.1 Solve Differential equation

```
# Example : 1   Chapter : 6.3     Page No: 313
# Solve Differential equation

# lambda1,lamda2,x1,x2,c,d are computed here.. for
    remaining details look textbook
A<-matrix(c(0,1,1,0),ncol=2)
lambda<-eigen(A)$values
x<-eigen(A)$vectors #These are normalised eigen
    vectors
#to get eigen vectors in textbook .. Multiply them
    with the scalars
x[,1]<-x[,1]*(1/x[1,1])
x[,2]<-x[,2]*(1/x[1,2])
print(x)
u<-c(4,2)
cd<-solve(x,u)
C<-cd[1]
D<-cd[2]
print("Lambda 1 and Lambda 2")
```

```
print(lambda)
print("x1 and x2")
print(x)
print("C and D are")
print(C)
print(D)
```

R code Exa 6.3.2 Solve Differential equation

```
# Example : 2   Chapter : 6.3     Page No: 314
# Solve Differential equation

# lambda1,lamda2,lambda3,x1,x2,x3,c1,c2,c3 are computed here.. for remaining details look textbook
A<-matrix(c(1,0,0,1,2,0,1,1,3),ncol=3)
lambda<-eigen(A)$values
x<-round(eigen(A)$vectors)
u<-c(9,7,4)
c<-solve(x,u)
print("Lambda 3 and Lambda 2 and Lambda 1 are")
print(lambda)
print("x3 and x2 and x1")
print(x)
print("c3,c2 and c1 are")
print(c)
#The  answer may slightly vary due to  rounding off values
#The answers provided in the text book may vary because of the computation process
#Both answers are correct , here it is taken -Ax+b=0 , In the text book it is considered as Ax-b=0
```

R code Exa 6.3.6 Solve Differential equation

```
# Example : 6    Chapter : 6.3      Page No: 321
# Solve Differential equation

# lambda1,lamda2,x1,x2,c,d are computed here.. for
    remaining details look textbook
A<-matrix(c(1,0,1,2),ncol=2)
lambda<-eigen(A)$values
x<-round(eigen(A)$vectors)
u<-c(2,1)
c<-solve(x,u)
print("Lambda 1 and Lambda 2")
print(lambda)
print("x1 and x2")
print(x)
print("c1 and c2 are")
print(c)
#The  answer may slightly vary due to  rounding off
    values
#The answers provided in the text book may vary
    because of the computation process
#Both answers are correct , here it is taken -Ax+b=0
     , In the text book it is considered as Ax-b=0
```

R code Exa 6.4.b Eigen values and eigen vectors

```
# Example : 6.4B    Chapter : 6.4      Page No: 336
# Eigen Values and Eigen vectors of given matrices
A3<-matrix(c(2,-1,0,-1,2,-1,0,-1,2),ncol=3)
eigenvalues<-eigen(A3)$values
x<-eigen(A3)$vectors
print("Eigen values of A3 are ")
print(eigenvalues)
print("Eigen vectors of A3 in normalised form are ")
```

```
print(x)
#to get eigen vectors in the textbook multiply normalised vectors by scalars
x[,1]<-x[,1]*(1/x[1,1])
x[,2]<-x[,2]*(sqrt(2)/x[1,2])
x[,3]<-x[,3]*(1/x[1,3])
print("Eigen vectors with respect to the above eigen values respectively are")
print(x)
B4<-matrix(c(1,-1,0,0,-1,2,-1,0,0,-1,2,-1,0,0,-1,1),ncol=4)
B4eigenvalues<-eigen(B4)$values
B4x<-eigen(B4)$vectors
print("Eigen values of B4 are ")
print(B4eigenvalues)
print("Eigen vectors of B4 in normalised form are")
print(B4x)
#to get eigen vectors in the textbook multiply normalised vectors by scalars
B4x[,1]<-B4x[,1]*(1/B4x[1,1])
B4x[,2]<-B4x[,2]*(1/B4x[1,2])
B4x[,3]<-B4x[,3]*(1/B4x[1,3])
B4x[,4]<-B4x[,4]*(1/B4x[1,4])
print("Eigen vectors with respect to the above eigen values respectively are")
print(B4x)
#The  answer may slightly vary due to  rounding off values
#The answers provided in the text book may vary because of the computation process
#Both answers are correct , here it is taken -Ax+b=0 , In the text book it is considered as Ax-b=0
```

R code Exa 6.4.1 Eigen values and eigen vectors

```
# Example : 1   Chapter : 6.4     Page No: 331
# Eigen Values and Eigen vectors of A
A<-matrix(c(1,2,2,4),ncol=2)
lambda<-eigen(A)$values
x<-eigen(A)$vectors #These are Normalised eigen vectors
#to get eigen vectors in textbook
x[,1]<-x[,1]*(1/x[1,1])
x[,2]<-x[,2]*(2/x[1,2])
print("Eigen values and eigen vectors of respective eigen values of A")
print(lambda)
print(x)
#The  answer may slightly vary due to  rounding off values
#The answers provided in the text book may vary because of the computation process
#Both answers are correct , here it is taken -Ax+b=0 , In the text book it is considered as Ax-b=0
```

R code Exa 6.4.4 pivots and eigen values have same signs for symmetric matrices

```
# Packages used  : pracma
# To install pracma,type following in command line while connected to internet
# install.packages("pracma")
# package can be included by command " library(pracma) "
# for more information about pracma visit https://cran.r-project.org/web/packages/pracma/index.html

# Example : 4   Chapter : 6.4     Page No: 334
# Pivots and Eigen values have same signs for the Symmetric matrices
```

```
library(pracma)
A<-matrix(c(1,3,3,1),ncol=2)
u<-lu(A)$U
pivots<-c(u[1,1],u[2,2])
eigenvalues<-eigen(A)$values
print("Eigen values and pivots have same signs for
   symmetric matrix")
print(pivots)
print(eigenvalues)
print("This is not true if the matrix is not
   symmetric")
B<-matrix(c(1,-1,6,-4),ncol=2)
Bu<-lu(B)$U
Bpivots<-c(Bu[1,1],Bu[2,2])
Beigenvalues<-eigen(B)$values
print(Bpivots)
print(Beigenvalues)
```

R code Exa 6.5.1 Tests for positive definiteness

```
# Packages used : pracma
# To install pracma,type following in command line
   while connected to internet
# install.packages("pracma")
# package can be included by command " library(
   pracma) "
# for more information about pracma visit https://
   cran.r-project.org/web/packages/pracma/index.html

# Example : 1   Chapter : 6.5     Page No: 344
# Tests for positive definitenes
library(pracma)
A<-matrix(c(2,-1,0,-1,2,-1,0,-1,2),ncol=3)
u<-lu(A)$U
pivots<-c(u[1,1],u[2,2],u[3,3])
```

```
upper_left_deteminants<-c(A[1,1],det(A[-3,-3]),det(A
   ))
eigenvalues<-eigen(A)$values
print("All pivots, upper left determinants and eigen
     values are positive")
print(pivots)
print(upper_left_deteminants)
print(eigenvalues)
```

R code Exa 6.6.a Pascals Matrix and its inverse are similar

```
# Example : 3    Chapter : 6.6      Page No: 357
# Pascals matrix and its inverse are similar
A<-matrix(c(1,1,1,1,0,1,2,3,0,0,1,3,0,0,0,1),ncol=4)
A1<-solve(A)
Aev<-eigen(A)$values
A1ev<-eigen(A1)$values
print("Both pascal matrix and its inverse have same
   eigen values")
print(Aev)
print(A1ev)
```

R code Exa 6.6.1 Similar matrices are matrices with same eigen values

```
# Example : 1    Chapter : 6.6      Page No: 356
# Similar matrices are matrices with same eigen
   values
A<-matrix(c(0.5,0.5,0.5,0.5),ncol=2)
Aev<-eigen(A)$values
print("Eigen values of A ")
print(Aev)
Lambda<-matrix(c(1,0,0,0),ncol=2)
Lev<-eigen(Lambda)$values
```

```
print("eigen values of lambda matrix")
print(Lev)
M<-matrix(c(1,1,0,2),ncol=2)
M1<-solve(M)
M1AM<-M1%*%A%*%M
M1AMev<-eigen(M1AM)$values
print("EIgen values of M-1*A*M")
print(M1AMev)
print("A and M-1*A*M are similar matrices")
```

R code Exa 6.6.2 SImilar matrices with repeated eigen values

```
# Example : 2   Chapter : 6.6      Page No: 356
# Similar matrices with repeated eigen values
A<-matrix(c(0,0,1,0),ncol=2)
Aev<-eigen(A)$values
A1<-matrix(c(1,1,-1,-1),ncol=2)
A1ev<-round(eigen(A1)$values)
print("Both th egiven matrices are similar because
   their eigen values are same")
print(Aev)
print(A1ev)
#The answer may slightly vary due to rounding off
   values
```

R code Exa 6.6.3 Jordans Theorem and jordan Matrix

```
# Example : 3   Chapter : 6.6      Page No: 357
# Jordans theorem and Jordan Matrix
J<-matrix(c(5,0,0,1,5,0,0,1,5),ncol=3)
Jev<-eigen(J)$values
JT<-t(J)
JTev<-eigen(JT)$values
```

```
print("Jordans theorem says that both J and
   transpose of J are similar")
print(Jev)
print(JTev)
```

R code Exa 6.7.3 Singular Value Decomposition

```
# Example : 3   Chapter : 6.7     Page No: 366
# Singular Value decomposition

A<-matrix(c(2,-1,2,1),ncol=2)
print("Singular value decomposition is given by")
print(svd(A))
#The  answer may slightly vary due to  rounding off
   values
#The answers provided in the text book may vary
   because of the computation method followed.
```

R code Exa 6.7.4 Singular Value Decomposition

```
# Example : 3   Chapter : 6.7     Page No: 366
# Singular Value decomposition

A<-matrix(c(2,1,2,1),ncol=2)
print("Singular value decomposition is given by")
print(svd(A))
#The  answer may slightly vary due to  rounding off
   values
#The answers provided in the text book may vary
   because of the computation method followed.
```

Chapter 7

Linear Transformations

R code Exa 7.3.a leftinverse rightinverse Pseduoinverse of given matrices

```
# Example : 7.3A    Chapter : 7.3       Page No: 405
# leftinverse ,rightinverse ,Pseduoinverse of given
    matrices
A1<-matrix(c(2,2),ncol=1)
A2<-matrix(c(2,2),ncol=2)
A3<-matrix(c(2,2,2,2),ncol=2)
A1T<-t(A1)
A2T<-t(A2)
A1inv<-solve(A1T%*%A1)%*%A1T
print("Left inverse of A1")
print(A1inv)
print(A1inv%*%A1)
A2inv<-A2T%*%solve(A2%*%A2T)
print("right inverse of A2")
print(A2inv)
print(A2%*%A2inv)

#The answers given in the text book is wrong it is 1
    /8 ..  not 1/sqrt(8)
V1<-svd(A3)$v
U1T<-t(svd(A3)$u)
```

```
d<-svd(A3)$d
sigma1<-matrix(c(1/d[1],0,0,0),ncol=2)
A3inv<-2*(V1%*%sigma1%*%U1T)
print("The Pseduo inverse of the given matrix")
print(A3inv)
```

R code Exa 7.3.1 Diagonilization of matrix

```
# Example : 1   Chapter : 7.3     Page No: 400
# Diagonaliazation of matrix
A<-matrix(c(0.5,-0.5,-0.5,0.5),ncol=2)
ev<-eigen(A)$values
D<-matrix(c(ev[1],0,0,ev[2]),ncol=2)
print("The diagonialized matrix")
print(D)
```

R code Exa 7.3.2 Similar Projection Matrices

```
# Example : 2   Chapter : 7.3     Page No: 401
# Similar Projection Matrices
A<-matrix(c(0.5,-0.5,-0.5,0.5),ncol=2)
Aev<-eigen(A)$values
W<-matrix(c(2,0,1,1),ncol=2)
W1<-solve(W)
B<-W1%*%A%*%W
print("Matrix B = W-1 * A * W")
print(B)
Bev<-eigen(B)$values
print("A and B are similar matrices")
print(Aev)
print(Bev)
```

R code Exa 7.3.3 Polar Decomposition

```
# Example : 3   Chapter : 7.3     Page No: 402
# Polar Decomposition
A<-matrix(c(2,-1,2,1),ncol=2)
Q<-round(svd(A)$u)%*%t(svd(A)$v)
H<-t(Q)%*%A
print("Polar Decomposition A=QH")
print("Q is ")
print(Q)
print("H is ")
print(H)
#The  answer may slightly vary due to  rounding off
    values
#The answers provided in the text book may vary
    because of the computation method followed.
```

R code Exa 7.3.4 Pseduo Inverse of a matrix

```
# Example : 4   Chapter : 7.3     Page No: 404
# Pseduoinverse of matrix
A<-matrix(c(2,1,2,1),ncol=2)
V<-svd(A)$v
UT<-t(svd(A)$u)
d<-svd(A)$d
sigma1<-matrix(c(1/d[1],0,0,0),ncol=2)
A1<-V%*%sigma1%*%UT
print("The Pseduo inverse of the given matrix")
print(A1)
#The  answer may slightly vary due to  rounding off
    values
```

```
#The answers provided in the text book may vary
    because of the computation method followed.
```

Chapter 8

Applications

R code Exa 8.1.1 Movements tensions elongations of spring

```
# Example : 1   Chapter : 8.1      Page No: 413
# Find movements , tensions , elongations of spring
K<-matrix(c(2,-1,0,-1,2,-1,0,-1,2),ncol=3)
K1<-solve(K)
f<-c(1,1,1)#since all mi=m
u<-K1%*%f
print("Movements are given by mg/c * u and u  is")
print(u)
A<-matrix(c(1,-1,0,0,0,1,-1,0,0,0,1,-1),ncol=3)
e<-A%*%u
print("Elongations are given by mg/c * e and e is")
print(e)
print("Tensions are given by mg * e and e is")
print(e)
```

R code Exa 8.1.2 Movements tensions elongations of spring

```
# Example : 2   Chapter : 8.1      Page No: 414
```

```
# Find movements , tensions , elongations of spring
K1<-matrix(c(2,-1,0,-1,2,-1,0,-1,1),ncol=3)
K11<-solve(K1)
f<-c(1,1,1)
u<-K11%*%f
print("Movements are given by mg/c * u and u  is")
print(u)
A<-matrix(c(1,-1,0,0,1,-1,0,0,1),ncol=3)
e<-A%*%u
print("Elongations are given by mg/c * e and e is")
print(e)
y<-solve(t(A))%*%f
print("Tensions are given by mg * y and y is")
print(y)
```

R code Exa 8.2.1 Find the currents

```
# Example : 1   Chapter : 8.2     Page No: 427
# Find the currents
A<-matrix(c
   (-1,-1,0,-1,0,0,1,0,-1,0,-1,0,0,1,1,0,0,-1,0,0,0,1,1,1)
   ,ncol=4)
AT<-t(A)
Laplacian_matrix<-AT%*%A
Laplacian_matrix<-Laplacian_matrix[-4,-4]
b<-c(1,0,0)
x<-solve(Laplacian_matrix,b)
x<-c(x,0)
print("Voltages are given by S * ")
print(x)
print("S is source")
y<--1*A%*%x
print("Currents are given by S * ")
print(y)
```

R code Exa 8.3.1 Positive Markov matrix application

```
# Example : 1    Chapter : 8.3       Page No: 432
# Positive Markov matrix Application.
A<-matrix(c(0.8,0.2,0.05,0.95),ncol=2)
u0<-c(0.02,0.98)
u1<-A%*%u0
print(u1)
u2<-A%*%u1
print(u2)
```

R code Exa 8.3.3 Markov matrix application

```
# Example : 3    Chapter : 8.3       Page No: 433
# Markov matrix Application.
A<-matrix(c(0,0.5,0.5,0.5,0,0.5,0.5,0.5,0),ncol=3)
u0<-c(8,16,32)
u1<-A%*%u0
u2<-A%*%u1
u3<-A%*%u2
print("Populations in three groups in subsequent
    months")
print(u1)
print(u2)
print(u3)
```

R code Exa 8.3.4 Linear Algebra in economy

```
# Example : 4    Chapter : 8.3       Page No: 437
```

```
# Linear Algebra in Economy
A<-matrix(c(0.2,0.4,0.5,0.3,0.4,0.1,0.4,0.1,0.3),
   ncol=3)
lambda<-eigen(A)$values
print("Lambda max is ")
print(lambda[1])
I<-matrix(c(1,0,0,0,1,0,0,0,1),ncol=3)
A1<-solve(I-A)
print(A1)
#The answers may vary due to rounding off values
```

R code Exa 8.3.5 Linear Algebra in economy

```
# Example : 5    Chapter : 8.3      Page No: 437
# Linear Algebra in Economy
A<-matrix(c(0,1,4,0),ncol=2)
lambda<-eigen(A)$values
print("Lambda max is ")
print(lambda[1])
I<-matrix(c(1,0,0,1),ncol=2)
A1<-solve(I-A)
print(A1)
#The answers may vary due to rounding off values
```

R code Exa 8.5.2 length of function sinx and sinx and cosx are orthogonal in function space

```
#integrate is a function from package stats which is
    included in R by default
#if not install package stats
# Example : 2    Chapter : 8.5      Page No: 448
# length of function sinx and sinx and cosx are
   orthogonal in function space
```

```
f1<-function(x){
  return (sin(x)*sin(x))
}
f2<-function(x){
  return (sin(x)*cos(x))
}
print("Inner product of sinx and sinx is ")
lsin2x<-integrate(f1,0,2*pi)
print(lsin2x)
print("Length of sinx is square root of above inner
   product")
print(sqrt(lsin2x$value))
print("Cos x and sinx are orthogonal in functional
   space because their inner product equals zero")
lsinxcosx<-integrate(f2,0,2*pi)
print(lsinxcosx)
#The  answer may slightly vary due to  rounding off
   values
```

R code Exa 8.6.2 Singular Value Decomposition

```
# Example : 6   Chapter : 8.6     Page No: 457
# Singualar Value Decomposition
gradematrixA<-matrix(c
   (-6,0,4,2,2,4,0,-6,0,-6,2,4,4,2,-6,0),ncol=4)
print("SVD of Grade matrix is")
print(svd(gradematrixA))
#The answers may vary due to rounding off values
```

Chapter 9

Numerical Linear Algebra

R code Exa 9.2.2 Norm of a diagonal matrix

```
# Example : 2   Chapter : 9.2      Page No: 475
# Norm of diagonal matrix
A<-matrix(c(2,0,0,3),ncol=2)
print("The norm of the diagonal matrix is its
   largest entry")
print(A[2,2])
print("Largest eigen value and norm are equal for
   this matrix")
lambdamax<-eigen(A)$values[1]
print(lambdamax)
print("Eigen vectors of this matrix are")
ev<-round(eigen(A)$vectors)
print(ev)
```

R code Exa 9.2.3 Condition number of Positive definite Matrix

```
# Example : 3   Chapter : 9.2      Page No: 478
# Condition number for positive definite matrix
```

```
A<-matrix(c(6,0,0,2),ncol=2)
print("Condition number for positive definite matrix
    is max.eigen value/min.eign value")
condition_number=eigen(A)$values[1]/eigen(A)$values
   [2]
print("Condition number for A")
print(condition_number)
```

R code Exa 9.3.1 Powers of some matrices can be calculated easily with max eigen value

```
# Example : 1   Chapter : 9.3      Page No: 482
# Powers of some matrices can be calculated easily
   with max eigen value
B<-matrix(c(0.6,0.6,0.5,0.5),ncol=2)
B1<-matrix(c(0.6,0,1.1,0.5),ncol=2)
lambda<-eigen(B)$values
lambda1<-eigen(B1)$values
lmax<-lambda[1]
l1max<-lambda1[1]
print("Lambda max of B is")
print(lmax)
print("Lambda max of B1 is ")
print(l1max)
B2<-B%*%B
print("B square is")
print(B2)
B2<-lmax*B
print(B2)
print("B1 square has 0.6*0.6 and 0.5*0.5 on its
   diagonal")
B12<-B1%*%B1
print(B12)
```

Chapter 10

Complex numbers and Matrices

R code Exa 10.1.1 r for complex numbers

```
# Example : 1   Chapter : 10.1     Page No: 496
# r of complex numbers
rad2deg <- function(rad) {
  (rad * 180) / (pi)
}
z<-complex(real=1,imaginary=1)
z1<-Conj(z)
print("r of z and its conjugate are")
print(Mod(z))
print(Mod(z1))
print("The argument of z and its conjugate in
   degrees are")
print(paste(rad2deg(Arg(z)),"degrees")) # in radians
    which is equal to 45 degree
print(paste(rad2deg(Arg(z1)),"degrees"))
```

R code Exa 10.2.1 Orthogonal Complex vectors

```
# Example : 1    Chapter : 10.2       Page No: 502
# Orthogonal Complex vectors
i<-complex(real=0,imaginary=1)
u<-matrix(c(1,i),ncol=1)
ut<-t(u)
#take conjugate of ut
ut[1,2]<-Conj(ut[1,2])
v<-matrix(c(i,1),ncol=1)
print("inner product of u and v is conj(u tanspose ) * v is  ")
innerproduct<-ut%*%v
print(innerproduct)
print("As innerproduct is zero , they are orthogonal complex vectors")
```