# R Textbook Companion for Introductory Linear Algebra: An Applied First Course by Bernard Kolman & David R. Hill[1]

Created by
Ashiq Mehmood Asharaf
R programming
Electrical & Electronics Engineering
Government Engineering College Barton Hill
Cross-Checked by
R TBC Team

November 14, 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:** Introductory Linear Algebra: An Applied First Course

**Author:** Bernard Kolman & David R. Hill

**Publisher:** Dorling Kindersley, India

**Edition:** 8

**Year:** 2008

**ISBN:** 978-8131723227

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

# Linear Equations and Matrices

**R code Exa 1.1.1** Linear Systems

```
#Page No. 2

A <- matrix(c(1,1,0.05,0.09),2,2,T)
b <- matrix(c(100000,7800),2,1,T)

ans <-solve(A,b)

cat("value of x :" , ans[1], "\n")
cat("value of y :" , ans[2])
```

**R code Exa 1.1.3** Linear Systems

```
#Page No. 3

A= matrix(c(1,2,3,2,-3,2,3,1,-1),3,3,T)
b= matrix(c(6,14,-2),3,1,T)

ans <-solve(A,b)
```

```
cat("value of x :" , ans[1], "\n")
cat("value of y :" , ans[2], "\n")
cat("value of z :", ans[3])
```

**R code Exa 1.1.4** Linear Systems

```
#Page No. 4

A=matrix(c(1,2,-3,2,1,-3),2,3,T)
A
b=matrix(c(-4,4),2,1,T)

asvd <-svd(A)
adiag <- diag(1/asvd$d)
adiag[2,1] =0
solution<- asvd$v %*% adiag %*% t(asvd$u)  %*% b

print(round(solution,0))

#The answer may vary due to difference in representation.
```

**R code Exa 1.1.5** Linear Systems

```
#Page No. 4

A= matrix(c(1,2,2,-2,3,5),3,2,T)
b= matrix(c(10,-4,26),3,1,T)

First_term <- t(A) %*% A
Second_term <- t(A) %*% b

ans <- solve(First_term, Second_term)
```

```

cat(ans[1] , "\n")
cat(ans[2])
```

**R code Exa 1.1.6** Linear Systems

```
#Page No. 5

library(matlib)
A <- matrix(c(1, 2, 2, -2, 3, 5), 3, 2, byrow = T)
b <- c(10,-4,20)
showEqn(A, b)
Solve(A, b, fractions = TRUE)
print("No common solution")
```

**R code Exa 1.2.3** Matrices

```
#Page No. 11


A <- matrix(c
    (0,785,3469,5959,785,0,3593,6706,3469,3593,0,6757,5959,6706,6757,0
    ,c(4,4), dimnames = list(c("LONDON","MADRID","
    NEWYORK","TOKYO"),c("LONDON","MADRID","NEWYORK","
    TOKYO ")))

print(A)
```

**R code Exa 1.2.10** Matrices

```
#Page No. 14


A<-matrix(c(1,2,-2,-1,4,3), c(2,3))
B<-matrix(c(0,1,2,3,-4,1),c(2,3))

ans <- A + B
print(ans)
```

**R code Exa 1.2.12** Matrices

```
#Page No. 15

A<- matrix(c(2,4,3,2,-5,1),c(2,3))
B<-matrix(c(2,3,-1,5,3,-2),c(2,3))

ans <- A - B

print(ans)
```

**R code Exa 1.2.14** Matrices

```
#Page No. 16

A1 <- matrix(c(0,2,1,-3,3,-2,5,4,-3),c(3,3))
A2 <- matrix(c(5,6,-1,2,2,-2,3,3,3),c(3,3))

solution <- (3 *A1) - (0.5 *A2)
print(solution)

#The answer may vary due to difference in
   representation.
```

**R code Exa 1.2.15** Matrices

```
#Page No. 16

A<-matrix(c(4,0,-2,5,3,-2),c(2,3))
B<-matrix(c(6,3,0,2,-1,4,-4,2,3),c(3,3))
C<-matrix(c(5,-3,2,4,2,-3),c(3,2))
D<-matrix(c(3,-5,1),c(1,3))
E<-matrix(c(2,-1,3),c(3,1))

Atrans = t(A)
Btrans = t(B)
Ctrans = t(C)
Dtrans = t(D)
Etrans = t(E)

print(Atrans)
print(Btrans)
print(Ctrans)
print(Dtrans)
print(Etrans)
```

---

**R code Exa 1.3.1** Dot Product and Matrix Multiplication

```
#Page No. 22


DOT =function(U,V)
{
      w<-0
      product_matrix <- U*V
      for(num in product_matrix)
```

```
        {
          w<-w+num
        }
        return (w)
}

U<- c(1,-2,3,4)
V<- c(2,3,-2,1)

ans <- DOT(U,V)
cat("u.v is : ", ans)
```

**R code Exa 1.3.4** Dot Product and Matrix Multiplication

```
#Page No. 23

A<- matrix(c(1,3,2,1,-1,4), nrow= 2, ncol= 3)
B<- matrix(c(-2,4,2,5,-3,1), nrow= 3, ncol=2 )

ans <- A%*%B
print(ans)
```

**R code Exa 1.3.5** Dot Product and Matrix Multiplication

```
#Page No. 24

A<-matrix(c(1,4,0,-2,2,1,3,1,-2),nrow=3,ncol=3)
B<-matrix(c(1,3,-2,4,-1,2),nrow=3,ncol=2)

x<-A[c(3),c(1,2,3)]
y<-B[c(1,2,3),c(2)]

solution = x%*%y
```

```
print(solution)
```

**R code Exa 1.3.10** Dot Product and Matrix Multiplication

```
#Page No. 25

A<-matrix(c(1,-1,2,3),c(2,2))
B<-matrix(c(2,0,1,1),c(2,2))

AB<- A%*%B
BA<- B%*%A

Check=function(x,y)
{
  w<-identical(x,y)
  return(w)
}

ans <- Check(AB,BA)
print(ans)

#The answer may vary due to difference in
   representation.
```

**R code Exa 1.3.21** Dot Product and Matrix Multiplication

```
#Page No. 30

A<- matrix(c(1,0,2,0,0,2,0,1,1,3,-4,0,0,-1,0,3),c
   (4,4))
B<- matrix(c
   (2,0,1,-3,0,1,3,-1,0,1,0,2,1,-1,0,1,1,2,1,0,-1,2,0,-1)
   ,nrow=4,ncol=6)
```

```
C<- A%*%B

Aii= A[c(1,2),c(1,2)]
Bii= B[c(1,2),c(1,2,3)]
Aij= A[c(1,2),c(3,4)]
Bji= B[c(3,4),c(1,2,3)]

Cii <- Aii %*% Bii + Aij %*% Bji
print(Cii)
```

**R code Exa 1.3.23** Dot Product and Matrix Multiplication

```
#Page No. 32

A<- array(c(3,4,5,8))
sum<-0
for(num in A)
{
  sum= sum+num
}
print(sum)
```

**R code Exa 1.4.2** Properties of Matrix operations

```
#Page No. 41

A<- matrix(c(2,-4,3,5,4,-2),c(2,3))
minus= (-A)

solution <- A + minus
print(solution)

```

```
#The answer may vary due to difference in
    representation.
```

**R code Exa 1.4.5** Properties of Matrix operations

```
#Page No. 42


A<-matrix(c(2,3,2,-1,3,2),c(2,3))
B<-matrix(c(1,2,3,0,2,-1),c(3,2))
C<-matrix(c(-1,1,2,2,0,-2),c(3,2))

First_method = A %*% (B + C)
print(First_method)

Second_method = A%*%B + A%*%C
print(Second_method)

if(identical(First_method, Second_method) == TRUE){
  print("Condition satisfied")
} else
{
  print(("Not satisfied"))
}
```

**R code Exa 1.4.10** Properties of Matrix Operations

```
#Page No. 45

r<- -2
A<-matrix(c(1,-2,2,0,3,1),c(2,3))
B<-matrix(c(2,1,0,-1,4,-2),c(3,2))

```

```
First_method = A %*% (r * B)

Second_method = r * (A%*%B)

Third_method = (r*A) %*% B

if (identical(First_method , Second_method) &&
    identical(First_method , Third_method) &&
    identical(Second_method ,Third_method) == TRUE )
       {
    print("Theorem 1.3d is satisfied")
}else {
   print("Theorem 1.3d is not satisfied")
}
```

**R code Exa 1.4.11** Properties of Matrix Operations

```
#Page No. 46

A<-matrix(c(1,2,3,-1,2,3),c(2,3))
B<-matrix(c(0,2,3,1,2,-1),c(3,2))

Method_one = t(A%*%B)
Method_two = t(B) %*% t(A)

if(identical(Method_one, Method_two) == TRUE ){
  print("Both methods are equal")

}else
{
  print("Both methods are different")
}
```

**R code Exa 1.5.3** Matrix Transformations

```
#Page No. 55

A<- matrix(c(1,-2,2,3),c(2,2))
v<- c('v1','v2')
W<- matrix(c(4,-1),c(2,1))
x<-solve(A,W)

cat(x[c(1),c(1)] , "\n")
cat(x[c(2),c(1)])
```

**R code Exa 1.6.4** Solutions of Linear Systems of Equations

```
#Page No. 64


A<-matrix(c(1,2,1,2,1,-2,4,3,2,3,2,3),c(3,4))

add<- 2* A[c(3),c(1,2,3,4)] + A[c(2),c(1,2,3,4)]
A[c(2),c(1,2,3,4)]<-add;A

B=A
temp=B[c(2),c(1,2,3,4)]
tem2=B[c(3),c(1,2,3,4)]
B[c(3),c(1,2,3,4)] = temp
B[c(2),c(1,2,3,4)] = tem2

C=B
adon<- 2* C[c(1),c(1,2,3,4)]

print("Row echelon matrix : ")
C[c(1),c(1,2,3,4)]<-adon;C
```

**R code Exa 1.6.5** Solutions of Linear Systems of Equations

```
#Page No. 65

A<-matrix(c
    (0,0,2,2,2,0,2,0,3,2,-5,-6,-4,3,2,9,1,4,4,7),c
    (4,5))

Pcol = function(x,y)
{
count<-0
var<-0
arr1= A[c(1),c(1,2,3,4,5)]
for(num in arr1 )
     { if(num==0)
       {count=count+1
       next}
      else
        return (count)
        break
     }

  }

pivot = function(x,y)
    {
      arr<-A[c(1,2,3,4),c(1)]
      for(num in arr)
        if(num ==0)
          next
      else
        return (num)

}
```

```
c<-Pcol(A)
p<-pivot(A)

A2=A
  a<-A[c(1),c(1,2,3,4,5)]
  c<-A[c(3),c(1,2,3,4,5)]

A2[c(1),c(1,2,3,4,5)]<-c
A2[c(3),c(1,2,3,4,5)]<-a

mul<- A2[c(1),c(1,2,3,4,5)] * 1/p
A2[c(1),c(1,2,3,4,5)]<-mul;A2

A3<-A2
A3[c(4),c(1,2,3,4,5)]<- (-2)*A3[c(1),c(1,2,3,4,5)] + A3[c(4),c(1,2,3,4,5)]

B<- A3[c(2,3,4),c(1,2,3,4,5)]

c<-Pcol(B)
p2<-pivot(B)

a<-B[c(1),c(1,2,3,4,5)]
b<-B[c(2),c(1,2,3,4,5)]
B[c(1),c(1,2,3,4,5)]<-b
B[c(2),c(1,2,3,4,5)]<-a

app12<-B[c(1),c(1,2,3,4,5)]* 1/p2
B[c(1),c(1,2,3,4,5)]<-app12;B
B[c(3),c(1,2,3,4,5)]<- (B[c(1),c(1,2,3,4,5)]*2 )+ B[c(3),c(1,2,3,4,5)]

C<- B[c(2,3),c(1,2,3,4,5)]
c<-Pcol(C)
p3<-pivot(C)

adik<- C[c(1),c(1,2,3,4,5)] * 1/p3
C[c(1),c(1,2,3,4,5)]<-adik;C
```

```
C[c(2),c(1,2,3,4,5)]<- ( C[c(1),c(1,2,3,4,5)]*(-2) ) + C[c(2),c(1,2,3,4,5)]
D<-C[c(2),c(1,2,3,4,5)]
first<-A3[c(1),c(1,2,3,4,5)]
second<-B[c(1),c(1,2,3,4,5)]
third<-C[c(1),c(1,2,3,4,5)]
fourth<-D

H<-matrix(first)
l<-cbind(H,second)
cbind(l,third)
k<-cbind(l,third)
cbind(k,fourth)
z<-cbind(k,fourth)
solution<-t(z)

print(solution)

#The answer may vary due to difference in representation.
```

**R code Exa 1.7.4** The Inverse of a Matrix

```
#Page No. 93

A<-matrix(c(1,3,2,4), c(2,2))

first<-solve(A)
trans<-t(first)

second<-t(A)
inv<-solve(second)


if(all.equal(inv,trans) == TRUE){
```

```
  print(trans)
  print(inv)
  print("Equal")
}else
{
  print("Not Equal")
}

#The answer may vary due to difference in
   representation.
```

**R code Exa 1.7.5** The Inverse of a Matrix

```
#Page No. 96

A<-matrix(c(1,0,5,1,2,5,1,3,1),c(3,3))

Ainv<- solve(A)

print(Ainv)

#The answer may vary due to difference in
   representation.
```

**R code Exa 1.7.6** The Inverse of a Matrix

```
#Page No. 97

 A<-matrix(c(1,1,5,2,-2,-2,-3,1,-3),c(3,3))

 A1<-A
  A1[c(2),c(1,2,3)]<- (A1[c(1),c(1,2,3)]*(-1)) + A1[
     c(2),c(1,2,3)]
```

```
  A2<-A1
  A2[c(3),c(1,2,3)]<- (A2[c(1),c(1,2,3)]*(-5)) + A2[
     c(3),c(1,2,3)]
  A3<-A2
  A3[c(3),c(1,2,3)]<- (A3[c(2),c(1,2,3)]*(-3)) + A3[
     c(3),c(1,2,3)]


arr<-t(A3)
count<-0

if(arr[7]==0 && arr[8]==0 && arr[9]==0){
      count=count+1

if(count==1){
     print(A3)
     print("It is a singular matrix")
 }
 else
   print("Not singular matrix")

}
```

# Chapter 2

# Applications of Linear Equations and Matrices

**R code Exa 2.2.5** Graph Theory

```
#Page No. 127

library(igraph ,quietly=TRUE)
simple.graph <- graph_from_literal(P4-+P2,P3-+P2,P1
   -+P5,P3-+P1 ,P2-+P5, P2-+P5,P6-+P2,P6-+P4,P4-+P5,
   P3-+P5)
plot.igraph(simple.graph)
get.adjacency(simple.graph)

#The answer may vary due to difference in
   representation.
```

---

**R code Exa 2.2.6** Graph Theory

```
#Page No. 127

```

```
library(igraph, quietly=TRUE)
simple.graph <- graph_from_literal(P3-+P1,P1-+P5,P2
   -+P6,P6-+P3,P4-+P2,P4-+P5,P4-+P6,P5-+P6,P2++P5,P6
   -+P1,P3-+P2,P3-+P5)
plot.igraph(simple.graph)
get.adjacency(simple.graph)

#The answer may vary due to difference in
   representation.
```

**R code Exa 2.2.14** Graph Theory

```
#Page No. 133

library(igraph)

simple.graph<- graph_from_literal(P1-+P2,P2-+P3,P3++
   P4, P2-+P4,P3-+P5,P5-+P4,P1++P4,P5-+P2)
plot.igraph(simple.graph)
A<-get.adjacency(simple.graph)
square=A%*%A
cube= square%*% A
fourth= cube%*% A
solution<- A + square + cube + fourth

check = function(M)
{
  y<-0
  count<-0
  M<-matrix(1:25,nrow=5,ncol=5)

  for(num in M)
      if(num>0)
       {
        count=count+1
```

```
            if(count==25)
            {

              y<-TRUE
            }
            else
            { next }
        }
      else
      {
        break
      }

  return(y)

}


if(check(solution) == TRUE){
  print(solution)
  print("Strongly connected")
}else{
  print("Not connected strongly")
}


#The answer may vary due to difference in
   representation.
```

**R code Exa 2.4.1** Electrical Circuits

```
#Page No. 146

library(igraph,quietly = TRUE)

```

```
simple.graph<- graph_from_literal(a-b,b-c,c-d,d-e,e-f,f-a,c-f)
plot.igraph(simple.graph, mark.shape = -0.6)

E1<-40
E2<-120
E3<-80
R1<-5
R2<-10
R3<-10
R4<-30

coeff<-matrix(c(1,1,0,1,-2,1,-1,0,5),c(3,3))
const<-matrix(c(0,-16,20),c(3,1))
solution<-solve(coeff,const)
print(solution)

cat(solution[c(1)] ,"\n")
cat(solution[c(2)] , "\n")
cat(solution[c(3)])
```

**R code Exa 2.5.1** Markov Chains

```
#Page No. 150


T<- matrix(c(1/2,1/2, 2/3,1/3),c(2,2),dimnames=list(c("D","R"),c("R","D")) )
print(T)


#The answer may vary due to difference in representation.
```

**R code Exa 2.5.3** Markov Chains

```
#Page No. 151

x0<-matrix(c(1,0))

T<- matrix(c(2/3,1/3,1/2,1/2),c(2,2),dimnames=list(c
   ("D","R"),c("D","R")) )
print(T)
x1<- T %*% x0
round(x1,4)
x2<- T %*% x1
round(x2,4)
x3<- T %*% x2
round(x3,4)
x4<- T %*% x3
first_term<-round(x4,3)
x5<- T %*% x4
second_term <-round(x5,3)

if(identical(first_term, second_term) == TRUE){
  cat(second_term[1]*100, "\n" )
  cat(second_term[2]*100, "\n")
}

#The answer may slightly vary due to rounding off
   values.
```

**R code Exa 2.5.7** Markov Chains

```
#Page No. 154

```

```
T<-matrix(c(0.2,0.8,1,0),c(2,2))
square<- T %*% T

check =function(x,y)
{
  count<-0
for (num in square)
  if(num>0)
   {
    count=count+1
      if(count==4)
      {
        y<-TRUE
        return(y)
        next
      }
     }else
     {
        break
     }
  }

K<-check(square)
if(K==TRUE)
  {
  print(square)
  print("Regular")
}else{
   print("Not Regular")
}
```

# Chapter 3

# Determinants

**R code Exa 3.1.5** Definition and Properties

```
#Page No. 184

a11<-2
a21<-4
a12<- -3
a22<- 5
A<-matrix(c(a11,a21,a12,a22),c(2,2))

F_Determinant= function(x,y)
{
  x<- a11*a22
  y<- a12*a21
  Delta<-x - y
  return(Delta)

}

cat(F_Determinant(A))
```

**R code Exa 3.1.7** Definition and Properties

```
#Page No. 185

 A<-matrix(c(1,2,3,2,1,1,3,3,2),c(3,3))
 print(A)
 cat(det(A))
```

**R code Exa 3.1.9** Definition and Properties

```
#Page No. 186

A<-matrix(c(1,2,3,2,1,3,3,1,2),c(3,3))
trans<- t(A)
DetA<-det(A)
Dtrans<- det(trans)

cat(DetA,"\n")
cat(Dtrans, "\n")
```

**R code Exa 3.1.10** Definition and Properties

```
#Page No. 186

A<-matrix(c(2,3,-1,2),c(2,2))
B<-matrix(c(3,2,2,-1),c(2,2))

Det_A<-det(A)
Det_B<-det(B)

cat(Det_A, "\n")
cat(Det_B, "\n")
```

**R code Exa 3.1.11** Definition and Properties

```
#Page No. 187

A<-matrix(c(1,-1,1,2,0,2,3,7,3) , nrow=3, ncol=3)

check = function(a,b,d)
{
  a<- A[c(1),c(1,2,3)]
  b<- A[c(2),c(1,2,3)]
  d<- A[c(3),c(1,2,3)]

      x=identical(a,d)
      y=identical(a,b)
      z=identical(b,d)

      if(x==TRUE || y==TRUE || z==TRUE)
      {
        return(det(A))
      }
      else
       return("Non-identical rows")
}
cat(check(A))
```

---

**R code Exa 3.1.14** Definition and Properties

```
#Page No. 187

A<-matrix(c(1,1,2,2,5,8,3,3,6),c(3,3))

first_Det<-det(A)
```

```
f<- 2
A[c(3),c(1,2,3)] <- A[c(3),c(1,2,3)]/f
f2<- 3
A[c(1,2,3),c(3)]<-  A[c(1,2,3),c(3)]/f2

second_Det<- det(A)*f*f2

if(identical(first_Det , second_Det)){
  cat(first_Det , "\n")

}
```

**R code Exa 3.1.18** Definition and Properties

```
#Page No. 191

A<-matrix(c(1,3,2,4),c(2,2))
B<-matrix(c(2,1,-1,2),c(2,2))

mul_AB<- A%*%B
det_AB= det(mul_AB)

det_A= det(A)
det_B= det(B)
det_product = det_A * det_B

x<-all.equal.numeric(det_AB,det_product)

if(x==TRUE){
  cat(det_AB, "\n")
  cat(det_product, "\n")

}
```

**R code Exa 3.1.19** Definition and Properties

```
#Page No. 191

A<-matrix(c(1,3,2,4),c(2,2))
Ainv <- solve(A)

Det_A    = det(A)
Det_Ainv = det(Ainv)

Det_A_reciprocal = 1/Det_A

x<-all.equal.numeric(Det_A_reciprocal , Det_Ainv)
if(x==TRUE){
  cat(Det_A_reciprocal, "\n")
  cat(Det_Ainv, "\n")
}

# The answer may vary due to difference in
   representation.
```

---

**R code Exa 3.2.1** Cofactor Expansion and Applications

```
#Page No. 196

library(matlib)

A<-matrix(c(3,4,7,-1,5,1,2,6,2),c(3,3))

A12 <- cofactor(A,1,2)
A23 <- cofactor(A,2,3)
A31 <- cofactor(A,3,1)
```

```
solution <- array(c(A12, A23, A31), dim = c(3,1))
cat(solution)
```

**R code Exa 3.2.2** Cofactor Expansion and Applications

```
#Page No. 197

library(matlib)
A<-matrix(c(1,-4,3,2,2,2,0,0,-3,1,0,-2,4,3,-3,3),c
   (4,4))

C11<- cofactor(A,3,1)
C12<- cofactor(A,3,2)
C13<- cofactor(A,3,3)
C14<- cofactor(A,3,4)

solution<- C11*A[c(3),c(1)] + C12*A[c(3),c(2)] + C13
   *A[c(3),c(3)] + C14*A[c(3),c(4)]

cat(solution)
```

**R code Exa 3.2.5** Cofactor Expansion and Applications

```
#Page No. 200

library(matlib)

A<- matrix(c(3,-2,1,5,6,2,1,0,-3),nrow=3,byrow =
   TRUE)

print(adjoint(A))
```

**R code Exa 3.2.7** Cofactor Expansion and Applications

```
#Page No. 202

library(matlib)
A<-matrix(c(3,-2,1,5,6,2,1,0,-3),nrow = 3,byrow =
    TRUE)
print(A)
det(A)
X<-solve(A)
print(X)

Check.inv = function(A.inverse)
{
  det_A <- 1/det(A)
Adj_A <- adjoint(A)
A.inverse = Adj_A *  det_A
return(A.inverse)
}

Check.inv(A)

#The answer may vary due to difference in
    representation.
```

---

**R code Exa 3.2.9** Cofactor Expansion and Applications

```
#Page No. 206

library(matlib)

```

```r
coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow = TRUE)
b<- matrix(c(1,4,-3),c(3,1))
first_D <- det(coeff)

coeff[,1]<- b;coeff
Second_D <-det(coeff)
x1<- Second_D / first_D
coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow = TRUE)

coeff[,2]<- b;coeff
third_D <- det(coeff)
x2<- third_D / first_D
coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow = TRUE)

coeff[,3] <- b;coeff
fourth_D<-det(coeff)
x3<- fourth_D / first_D

cat(x1, "\n")
cat(x2, "\n")
cat(x3)
```

# Chapter 4

# Vectors in Rn

**R code Exa 4.1.3** Vectors in the Plane

```
#Page No. 216

library(graphics)
x <- c(0,2)
y <- c(0,3)

plot.new()
plot.default(c(0,4),c(0,4))
arrows(x0=0,y0=0,x1=2,y1=3,length=0.15,angle=20,code
   =2 ,lwd=2)
```

---

**R code Exa 4.1.6** Vectors in the Plane

```
#Page No. 219

 x1 <- matrix(c(2,-5), ncol = 1, byrow = T)
 sum<-0
 sq<-0
```

```

 for(num in x1)
 {
  sq<- num^2
  sum<- sq+sum
  next
 }

cat(sqrt(sum))

#The answer may vary due to difference in
   representation.
```

---

**R code Exa 4.1.7** Vectors in the Plane

```
#Page No. 219

P<- matrix(c(3,2),c(2,1))
Q<-matrix(c(-1,5),c(2,1))

x<- (Q[c(1)] - P[c(1)])^2
y<- (Q[c(2)] - P[c(2)])^2

sol<- sqrt(x+y)
cat(sol)
```

---

**R code Exa 4.1.8** Vectors in the Plane

```
#Page No. 220


tri_angle<- matrix(c(-1,3,2,4,1,6),c(3,2))

```

```
shape <- cbind(tri_angle, 1)


area_calc = function(a)
{
  sol=0
  sol<- det(a) / 2
  return(sol)

}

cat(area_calc(shape))
```

**R code Exa 4.1.9** Vectors in the Plane

```
#Page No. 221

u<-c(1,2)
v<-c(3,-4)

x0<- u[1] + v[1]
y0<- u[2] + v[2]

sol<- c(x0,y0)
plot(x0,y0)

cat(sol)
```

**R code Exa 4.1.14** Vectors in the Plane

```
#Page No. 225

library(Matrix)
```

```
u<-c(2,4)
v<-c(-1,2)

DOT= function(u,v)
{
  sis<- u[1]*v[1] + u[2]*v[2]
  return(sis)

}
upper<- DOT(u,v)

magnitude =function(u)
{
  sum<-0
  s1<-0
  m<-0
  for(num in u)
    { s1<- num^2
      sum=sum+s1
      next }
 m<-sqrt(sum)
 return(m)
}

mag_u<-magnitude(u)
mag_v<-magnitude(v)

cos_theta<- upper/(mag_u*mag_v)

cat(cos_theta)
```

**R code Exa 4.1.15** Vectors in the Plane

```
#Page No. 225
```

```
u<-c(2,-4)
v<-c(4,2)

DOT= function(u,v)
{
  sis<- u[1]*v[1] + u[2]*v[2]
  return(sis)

}
sol<- DOT(u,v)

if(sol==0){
  cat(sol, "\n")
}
```

**R code Exa 4.1.16** Vectors in the Plane

```
#Page No. 226

library(matlib)

x<-c(-3,4)

unit_vector =function(u)
{
  sum<-0
  s1<-0
  m<-0
  for(num in u)
  { s1<- num^2
  sum=sum+s1
  next }
  m<-sqrt(sum)
  sol<- sqrt( (u[1]/m)^2 + (u[2]/m)^2 )
```

```r
  return(sol)
}

cat(unit_vector(x))
```

**R code Exa 4.2.2** n Vectors

```r
#Page No. 230

u<-c(1,-2,3)
v<-c(2,3,-3)

sum<- c(u+v)

cat(sum)
```

**R code Exa 4.2.3** n Vectors

```r
#Page No. 230

u<-c(2,3,-1,2)
c=-2
final<- c*u

cat(final)
```

**R code Exa 4.2.10** n Vectors

```r
#Page No. 237

```

```
library(matlib)

u<-c(2,3,2,-1)
v<-c(4,2,1,3)

DOT= function(u,v)
{
  sis<- u[1]*v[1] + u[2]*v[2] + u[3]*v[3] + u[4]*v[4]
  return(sis)

}
sol<- DOT(u,v)

magnitude =function(u)
{
  sum<-0
  s1<-0
  m<-0
  for(num in u)
  {  s1<- num^2
     sum=sum+s1
      next
     }
  m<-sqrt(sum)
  return(m)
}

mag_u<- magnitude(u)
mag_v<- magnitude(v)
mag_UV <- mag_u * mag_v

if(sol <= mag_UV)
{
  cos_angle<- abs(sol/mag_UV )

  if(cos_angle >= -1 && cos_angle <= 1)
  {
```

```
    cat(cos_angle, "\n")

  }

}

#The answer may vary due to difference in
   representation.
```

---

**R code Exa 4.2.13** n Vectors

```
#Page No. 239

library(matlib)

u<-c(1,0,0,1)
v<-c(0,1,1,0)

magnitude =function(u)
{
  sum<-0
  s1<-0
  m<-0
  for(num in u)
  { s1<- num^2
    sum=sum+s1
    next
    }
  m<-sqrt(sum)
  return(m)
}

mag_u<- magnitude(u)
mag_v<- magnitude(v)
```

```
magU_V <- mag_u + mag_v
mag_UV <- magnitude(u+v)

if(mag_UV <= magU_V)
 {
   cat(mag_UV, "\n")
  cat(magU_V, "\n")

}

#The answer may vary due to difference in
   representation.
```

**R code Exa 4.3.2** Linear Transformations

```
#Page No. 249

Li<-c(2,-1)
Lj<-c(3,1)
Lk<-c(-1,2)

sol<- c(-3,4,2)
sol<- -3*Li + 4*Lj + 2*Lk

cat(sol)
```

# Chapter 5

# Applications of Vectors in R2 and R3

**R code Exa 5.1.1** Cross Product in R3

```
#Page No. 259

vector.cross <- function(a, b)
  {
    if(length(a)!=3 || length(b)!=3){
      stop("Cross product is only defined for 3D
         vectors.");
    }
    i1 <- c(2,3,1)
    i2 <- c(3,1,2)
    sol<-(a[i1]*b[i2] - a[i2]*b[i1])
    return(sol)
  }

u<-c(2,1,2)
v<-c(3,-1,-3)

cat(vector.cross(u,v), "\n")

```

```
#The answer may vary due to difference in
   representation.
```

**R code Exa 5.1.4** Cross Product in R3

```
#Page No. 262

library(utils)
library(matlib)

P1<-matrix(c(2,2,4),byrow = T)
P2<-matrix(c(-1,0,5),byrow= T)
P3<-matrix(c(3,4,3),byrow=T)
vector.cross <- function(a, b)
{

   cal<- b[c(1)]-a[c(1)]
   cal2<- b[c(2)]-a[c(2)]
   cal3<- b[c(3)]-a[c(3)]
   cout<- matrix(c(cal,cal2,cal3),byrow = T)
   print(cout)
   dd<-matrix(c(cal),byrow=T)
   ff<-rbind(dd,cal2)
   gg<-rbind(ff,cal3)
   return(gg)

}
u= vector.cross(P1, P2)
v= vector.cross(P1, P3)

CrossProduct<- function(x, y, i=1:3)
  {

  To3D <- function(x) head(c(x, rep(0, 3)), 3)
  x <- To3D(x)
```

```
  y <- To3D(y)

  Index3D <- function(i) (i - 1) %% 3 + 1

  return (x[Index3D(i + 1)] * y[Index3D(i + 2)] -
            x[Index3D(i + 2)] * y[Index3D(i + 1)])
}

new<-CrossProduct(u,v)

Area.triangle= function(x)
{
  half<-0
  for(num in x)
  {
    half<-half+ (num/2)^2
    half

  }
  return(sqrt(half))
}

cat(Area.triangle(new), "\n")

#The answer may vary due to difference in
   representation.
```

**R code Exa 5.1.6** Cross Product in R3

```
#Page No. 263

u<-c(1,-2,3)
v<-c(1,3,1)
w<-c(2,1,2)
mat<-matrix(c(u,v,w),nrow=3,byrow = T)
```

```

CrossProduct= function(x, y, i=1:3)
{
  To3D <- function(x) head(c(x, rep(0, 3)), 3)
  x <- To3D(x)
  y <- To3D(y)

  Index3D <- function(i) (i - 1) %% 3 + 1

  return (x[Index3D(i + 1)] * y[Index3D(i + 2)] -
          x[Index3D(i + 2)] * y[Index3D(i + 1)])
}

vw_cross<-CrossProduct(v,w)

DOT= function(u,v)
{
  sis<- u[1]*v[1] + u[2]*v[2] + u[3]*v[3]
  return(sis)

}
uv_dot<-DOT(u,vw_cross)

vol1<- abs(uv_dot)
vol2<- abs(det(mat))

cat(vol1, "\n")
cat(vol2)
```

# Chapter 6

# Real Vector Spaces

**R code Exa 6.1.7** Vector Spaces

```
#Page No. 275

library(polynom)
P1<-polynomial(coef = c(-1,5,-2,0,3))
P2<-polynomial(coef=c(1,2))
P3<-polynomial(coef=c(4))

coeff.p1<-coef(P1)
coeff.p2<-coef(P2)
coeff.p3<-coef(P3)

degree=function(coeff_p1)
                {count=0

                  for(num in coeff_p1)
                  {
                    count=count+1
                  }
                 c<- count-1
                 return(c)
                }
```

```
degree(coeff.p1)
degree(coeff.p2)
degree(coeff.p3)
```

**R code Exa 6.2.11** Subspaces

```
#Page No. 283


v<-matrix(c(2,1,5),nrow=3,byrow=T)
A<-matrix(c(1,1,1,2,0,1,1,2,0),nrow=3,byrow=T)
v1<-A[c(1,2,3),c(1)]
v2<-A[c(1,2,3),c(2)]
v3<-A[c(1,2,3),c(3)]

c_matrix<-solve(A,v)
cat(c_matrix, "\n")
```

**R code Exa 6.2.12** Subspaces

```
#Page No. 284

s1<-matrix(c(1,0,0,0,0,0),c(2,3))
s2<-matrix(c(0,0,1,0,0,0),c(2,3))
s3<-matrix(c(0,0,0,1,0,0),c(2,3))
s4<-matrix(c(0,0,0,0,0,1),c(2,3))

matrix_model<-matrix(c('a',0,'b','c', '0', 'd'),c
    (2,2))

a<-1
b<-3
c<-5
```

```
d<-7
span_S <- (a * s1) + (b * s2) + (c * s3) + (d * s4)

print(span_S)

#The answer may vary due to difference in
   representation.
```

**R code Exa 6.3.2** Linear Independence

```
#Page No. 292

s1<-matrix(c(1,0,0,0),c(2,2))
s2<-matrix(c(0,1,1,0),c(2,2))
s3<-matrix(c(0,0,0,1),c(2,2))

matrix_model<-matrix(c('a','b','b','c'),c(2,2))

a<-3
b<-6
c<-9

span_S <- (a * s1) + (b * s2) + (c * s3)

print(span_S)

#The answer may vary due to difference in
   representation.
```

**R code Exa 6.3.8** Linear Independence

```
#Page No. 295

```

```
v1<- c(1,0,1,2)
v2<- c(0,1,1,2)
v3<- c(1,1,1,3)

A<-matrix(c(v1,v2,v3),nrow=4)
b<-matrix(c(0,0,0,0))

first<- t(A)%*%A
second<- t(A)%*%b
ans_matrix <-solve(first,second)

count<-0
for(num in ans_matrix)
{
  num=num+1
  if(num == 1)
    count= count+1

  else
    print("Linearly dependent")

}

if(count == 3){
  cat(ans_matrix, "\n")

}
```

**R code Exa 6.4.2** Basis and Dimension

```
#Page No. 303

v1<-c(1,0,1,0)
v2<-c(0,1,-1,2)
v3<-c(0,2,2,1)
```

```
v4<-c(1,0,0,1)

A<-matrix(c(v1,v2,v3,v4),c(4,4))
b<-matrix(c(0,0,0,0),c(4,1))
Sol<-solve(A,b)

count<-0
for(num in Sol)
{
    num=num+1
    if(num == 1)
      count= count+1

    else
      print("Linearly dependent")

}



if(count == 4){
  cat(Sol, "\n")
}
```

**R code Exa 6.4.5** Basis and Dimension

```
#page 309

v1<-c(1,2,-2,1)
v2<-c(-3,0,-4,3)
v3<-c(2,1,1,-1)
v4<-c(-3,3,-9,-6)
v5<-c(9,3,7,-6)

library(matlib)
```

```
A<-matrix(c(v1,v2,v3,v4,v5),c(4,5))
A
b<-matrix(c(0,0,0,0),c(4,1))
echelForm<-echelon(A,b)
echelForm<-echelForm[,-4]
echelForm

V_1<-echelForm[c(1,2,3,4),c(1)]
V_2<-echelForm[c(1,2,3,4),c(2)]
V_3<-echelForm[c(1,2,3,4),c(3)]
V_4<-echelForm[c(1,2,3,4),c(4)]
V_5<-echelForm[c(1,2,3,4),c(5)]

check= function(x)
{
  p<-0
  count<-0
  ans<-0
  for(num in x)
  {
    p=p+num
    count=count+1
        if(count==4)
        {
          ans<-p*p*p

          if(ans==1)
            print("Leading one")
          else
            print("Not Leading one")
        }
      else
        {
          next
        }


  }
```

```
}

check(V_1)
check(V_2)
check(V_3)
check(V_4)
check(V_5)
```

**R code Exa 6.5.3** Homogeneous Systems

```
#Page No. 322

library(matlib)

A<-matrix(c(1,3,5,-1),c(2,2))
lamda<- -10:10

for(num in lamda)
{
  MAT<-matrix(c(num-1,-3,-5,num+1),c(2,2))
  d<-det(MAT)

  if(d==0)
  {
    print(num)
    print(-num)

  }
  next
}
```

**R code Exa 6.6.5** The Rank of a Matrix and Applications

```
#Page No. 333

library(matlib)

A<-matrix(c(1,0,0,1,2,1,1,0,-1,1,4,2,0,0,6,1,1,1,0,0,2,1,2,2,1),c(5,5))
E<-echelon(A)

N<-nrow(E)-c(R(E))

cat(c(R(E)), "\n")
cat(N)
```

**R code Exa 6.6.8** The Rank of a Matrix and Applications

```
#Page No. 336

library(matlib)
A<-matrix(c(1,1,1,2,1,3,0,-3,3),c(3,3))
E<-echelon(A)

cat(c(R(E), "\n"))
```

**R code Exa 6.7.1** Coordinates and Changes of Basis

```
#Page No. 341

library(matlib)

v1=c(1,1,0,0)
v2=c(2,0,1,0)
v3=c(0,1,2,-1)
```

```
v4=c(0,1,-1,0)

mat<- matrix(c(v1,v2,v3,v4),c(4,4))
v<-matrix(c(1,2,-6,2),c(4,1))
E<-echelon(mat,v)

Vs<- matrix(c(E[c(1),c(5)], E[c(2),c(5)], E[c(3),c
  (5)], E[c(4),c(5)]), c(4,1))

print(Vs)
```

# Chapter 7

# Applications of Real vector Spaces

**R code Exa 7.2.1** Least Squares

```
#Page No. 380

A<-matrix(c
    (1,2,-2,4,0,1,2,1,3,2,2,-1,-1,1,4,1,1,2,3,2,1,0,3,0)
    ,c(6,4))
b<-matrix(c(1,5,-2,1,3,5),c(6,1))

trans<-t(A)
LHS<-trans%*%A
RHS<-trans%*%b

solve(LHS,RHS)

#The answer may slightly vary due to rounding off
    values.
```

**R code Exa 7.2.2** Least Squares

```
#Page No. 381

library(matlib)

A<-matrix(c
    (1,2,-2,4,0,1,2,1,3,2,2,-1,-1,1,4,1,1,2,3,2,1,0,3,0)
    ,c(6,4))
b<-matrix(c(1,5,-2,1,3,5),c(6,1))

qr<-QR(A)

Qknown<-qr$Q
Qtrans<-t(Qknown)
RHS<-Qtrans%*%b
Rknown<-qr$R

solution_1<-solve(Rknown,RHS)

trans<-t(A)
LHS<-trans%*%A
RHS<-trans%*%b
solution_2<-solve(LHS,RHS)

print(solution_1)

if(all.equal(solution_1 , solution_2)){
  print("Equal")
}else{
  print("Unequal")
}

#The answer may slightly vary due to rounding off
    values.
```

**R code Exa 7.2.4** Least Squares

```
#Page No. 384

library(polynom)

b=matrix(c(4.5,5.5,5.7,6.6,7.0,7.7,8.5,8.7,9.5,9.7),c(10,1))
A=matrix(c(3,4,5,6,7,8,9,10,11,12,1,1,1,1,1,1,1,1,1,1),c(10,2))
x=c('b1','b0')

trans<-t(A)
LHS<-trans %*% A
RHS<-trans %*% b
sol<-solve(LHS,RHS)
sol
b1<-sol[c(1)]
b0<-sol[c(2)]
y<-polynomial(coef=c(b0,b1))

x<-30
y<- b1*(x) + b0

cat(round(y, 3), "\n")

#The answer may slightly vary due to rounding off values.
```

# Chapter 8

# Eigenvalues Eigenvectors and Diagonalization

**R code Exa 8.1.2** Eigenvalues and Eigenvectors

```
#Page No. 409

A<-matrix(c(0,1/2,1/2,0),c(2,2))

ans<- eigen(A,only.values=FALSE, EISPACK = FALSE)

print(ans$vectors)
cat(ans$values, "\n")

#The answer may vary due to difference in
    representation.
```

**R code Exa 8.1.3** Eigenvalues and Eigenvectors

```
#Page No. 410

```

```
A<-matrix(c(0,0,0,1),c(2,2))

ans<- eigen(A, only.value=FALSE, EISPACK = FALSE)

cat(ans$values, "\n")
print(ans$vectors)

#The answer may vary due to difference in
    representation.
```

**R code Exa 8.1.5** Eigenvalues and Eigenvectors

```
#Page No. 412

library(pracma)
library(polynom)

lamda<-0
A<-matrix(c(lamda-1,-1,-4,-2,lamda-0,4,1,-1,lamda-5)
    ,c(3,3))

cpol<-charpoly(A)
roots<-polyroot(cpol)
print(roots)
polynomial(coef=c(-cpol[c(4)],cpol[c(3)],-cpol[c(2)
    ],cpol[c(1)]))

#The answer may vary due to difference in
    representation.
```

**R code Exa 8.1.6** Eigenvalues and Eigenvectors

```
#Page No. 414
```

```
A<-matrix(c(1L,1L,4L,2L,0L,-4L,-1L,1L,5L),c(3,3))
ans<- eigen(A)

cat(ans$values, "\n")
print(ans$vectors)

#The answer may vary due to difference in
    representation.
```

**R code Exa 8.1.8** Eigenvalues and Eigenvectors

```
#Page No. 419

A<- matrix(c(0,1/2,0,0,0,1/3,6,0,0),c(3,3))

ans<-eigen(A)
values<-ans$values

vect<- ans$vectors

solution_vector<- matrix(data=Re(vect[,3]),c(3,1),T)

cat(Re(values[c(3)]), "\n")
print(solution_vector)

#The answer may vary due to difference in
    representation.
```

**R code Exa 8.2.1** Diagonalization

```
#Page No. 422

```

```
A<-matrix(c(1,-2,1,4),c(2,2))
P<-matrix(c(1,1,1,2),c(2,2))
invo<- solve(P)

B<- invo %*% A %*% P
print(B)
```

**R code Exa 8.2.3** Diagonalization

```
#Page No. 424

A<-matrix(c(1,-2,1,4),c(2,2))
ev<-eigen(A)
vect<- ev$vectors
print(vect)
count<-0
for(num in vect){
  if(num != 0)
    count = count + 1
}

if(count == 4){
  P<-matrix(c(-vect[c(1,2),c(1)],-vect[c(1,2),c(2)])
     ,nrow=2)
  invo<- solve(P)

  ans<- invo %*% A %*% P
  print(ans)

}
```

**R code Exa 8.3.1** Diagonalization of Symmetric Matrices

```
#Page No. 434


A<- matrix(c(0,0,-2,0,-2,0,-2,0,3),c(3,3))
ev<-eigen(A)

v<-ev$values
D<-diag(v)

print(D)
```

**R code Exa 8.3.5** Diagonalization of Symmetric Matrices

```
#Page No. 439

A<-matrix(c(1,2,0,0,2,1,0,0,0,0,1,2,0,0,2,1),c(4,4))

ev<-eigen(A)
vect<- ev$vectors

x1<- vect[c(1,2,3,4),c(4)]
x2<- vect[c(1,2,3,4),c(3)]
x3<- vect[c(1,2,3,4),c(2)]
x4<- vect[c(1,2,3,4),c(1)]

ans<- matrix(c(x1,x2,x3,x4), c(4,4))
print(ans)
```

# Chapter 9

# Applications of Eigenvalues and Eigenvectors

**R code Exa 9.2.5** Differential Equations

```
#Page No. 457

library(matlib)

X<-matrix(c(4,6,8),c(3,1))
P<-matrix(c(1,1,1,1,2,4,1,4,16),c(3,3))

E<-echelon(P,X)
b<-E[c(1,2,3),c(4)]

cat(b[c(1)], "\n")
cat(b[c(2)], "\n")
cat(b[c(3)], "\n")
```

**R code Exa 9.2.6** Differential Equations

```
#Page No. 457

A<-matrix(c(1,0,0,0,3,-2,0,-2,3),c(3,3))

ev<-eigen(A)
vect<- round(ev$vectors, 0)
value<- Re(ev$values)
print(vect)

cat(value, "\n")
```

**R code Exa 9.4.7** Quadratic Forms

```
#Page No. 481

library(matrixcalc)

A<-matrix(c(0,0,0,0,3,4,0,4,-3),c(3,3))
E<-eigen(A)

value<- E$values
x<-value[c(2)]
y<-value[c(3)]
value[c(2)]=y
value[c(3)]=x
D<-diag(value)

k<-1/sqrt(5)
H<-matrix(c(k,0,0,0,k,0,0,0,1),c(3,3))

D1<- t(H) %*% D %*% H
rank<-matrix.rank(D1)

cat(value, "\n")
cat(rank, "\n")
```

# Chapter 10

# Linear Transformations and Matrices

**R code Exa 10.1.2** Definition and Examples

```
#Page No. 503

p<-c(3,2,4)
q<-c(4,3,3)
t<-5

LHS = t*(p+q) + t^2
RHS = (t*p + t^2) + (t*q + t^2)

check =function(methodX,methodY)
{
     result<-identical(methodX, methodY)
     if(result==FALSE)
     {
       print("Non-Linear transformation")

       }else
     {
       print("linear transformation")
```

```
        }

}

check(LHS, RHS)
```

**R code Exa 10.2.12** The Kernel and Range of a Linear Transformation

```
#Page No. 513

library(matlib)

a1<- c(1,0,1)
a2<- c(1,0,0)
a3<- c(0,1,1)
a4<- c(0,1,0)

S<-matrix(c(a1,a2,a3,a4),c(3,4))
b<-matrix(c(0,0,0),c(3,1))
E<-echelon(S,b)

basis =function(q)
{
  sum<-0
  for(num in q)
  {
     sum<-num^2 +sum

  }
  if(sum==1 )
     cat("basis: ", q, "\n")
  else
     cat("Not basis", "\n")


```

```
}

basis(E[c(1,2,3),c(1)])
basis(E[c(1,2,3),c(2)])
basis(E[c(1,2,3),c(3)])
basis(E[c(1,2,3),c(4)])
```