

R Textbook Companion for
Introductory Linear Algebra: An Applied
First Course
by Bernard Kolman & David R. Hill¹

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

November 14, 2020

¹Funded 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	4
1 Linear Equations and Matrices	5
2 Applications of Linear Equations and Matrices	21
3 Determinants	27
4 Vectors in \mathbb{R}^n	35
5 Applications of Vectors in \mathbb{R}^2 and \mathbb{R}^3	44
6 Real Vector Spaces	48
7 Applications of Real vector Spaces	57
8 Eigenvalues Eigenvectors and Diagonalization	60
9 Applications of Eigenvalues and Eigenvectors	65
10 Linear Transformations and Matrices	68

List of R Codes

Exa 1.1.1	Linear Systems	5
Exa 1.1.3	Linear Systems	5
Exa 1.1.4	Linear Systems	6
Exa 1.1.5	Linear Systems	6
Exa 1.1.6	Linear Systems	7
Exa 1.2.3	Matrices	7
Exa 1.2.10	Matrices	7
Exa 1.2.12	Matrices	8
Exa 1.2.14	Matrices	8
Exa 1.2.15	Matrices	9
Exa 1.3.1	Dot Product and Matrix Multiplication	9
Exa 1.3.4	Dot Product and Matrix Multiplication	10
Exa 1.3.5	Dot Product and Matrix Multiplication	10
Exa 1.3.10	Dot Product and Matrix Multiplication	11
Exa 1.3.21	Dot Product and Matrix Multiplication	11
Exa 1.3.23	Dot Product and Matrix Multiplication	12
Exa 1.4.2	Properties of Matrix operations	12
Exa 1.4.5	Properties of Matrix operations	13
Exa 1.4.10	Properties of Matrix Operations	13
Exa 1.4.11	Properties of Matrix Operations	14
Exa 1.5.3	Matrix Transformations	15
Exa 1.6.4	Solutions of Linear Systems of Equations	15
Exa 1.6.5	Solutions of Linear Systems of Equations	16
Exa 1.7.4	The Inverse of a Matrix	18
Exa 1.7.5	The Inverse of a Matrix	19
Exa 1.7.6	The Inverse of a Matrix	19
Exa 2.2.5	Graph Theory	21
Exa 2.2.6	Graph Theory	21

Exa 2.2.14	Graph Theory	22
Exa 2.4.1	Electrical Circuits	23
Exa 2.5.1	Markov Chains	24
Exa 2.5.3	Markov Chains	25
Exa 2.5.7	Markov Chains	25
Exa 3.1.5	Definition and Properties	27
Exa 3.1.7	Definition and Properties	28
Exa 3.1.9	Definition and Properties	28
Exa 3.1.10	Definition and Properties	28
Exa 3.1.11	Definition and Properties	29
Exa 3.1.14	Definition and Properties	29
Exa 3.1.18	Definition and Properties	30
Exa 3.1.19	Definition and Properties	31
Exa 3.2.1	Cofactor Expansion and Applications	31
Exa 3.2.2	Cofactor Expansion and Applications	32
Exa 3.2.5	Cofactor Expansion and Applications	32
Exa 3.2.7	Cofactor Expansion and Applications	33
Exa 3.2.9	Cofactor Expansion and Applications	33
Exa 4.1.3	Vectors in the Plane	35
Exa 4.1.6	Vectors in the Plane	35
Exa 4.1.7	Vectors in the Plane	36
Exa 4.1.8	Vectors in the Plane	36
Exa 4.1.9	Vectors in the Plane	37
Exa 4.1.14	Vectors in the Plane	37
Exa 4.1.15	Vectors in the Plane	38
Exa 4.1.16	Vectors in the Plane	39
Exa 4.2.2	n Vectors	40
Exa 4.2.3	n Vectors	40
Exa 4.2.10	n Vectors	40
Exa 4.2.13	n Vectors	42
Exa 4.3.2	Linear Transformations	43
Exa 5.1.1	Cross Product in \mathbb{R}^3	44
Exa 5.1.4	Cross Product in \mathbb{R}^3	45
Exa 5.1.6	Cross Product in \mathbb{R}^3	46
Exa 6.1.7	Vector Spaces	48
Exa 6.2.11	Subspaces	49
Exa 6.2.12	Subspaces	49
Exa 6.3.2	Linear Independence	50

Exa 6.3.8	Linear Independence	50
Exa 6.4.2	Basis and Dimension	51
Exa 6.4.5	Basis and Dimension	52
Exa 6.5.3	Homogeneous Systems	54
Exa 6.6.5	The Rank of a Matrix and Applications	54
Exa 6.6.8	The Rank of a Matrix and Applications	55
Exa 6.7.1	Coordinates and Changes of Basis	55
Exa 7.2.1	Least Squares	57
Exa 7.2.2	Least Squares	57
Exa 7.2.4	Least Squares	59
Exa 8.1.2	Eigenvalues and Eigenvectors	60
Exa 8.1.3	Eigenvalues and Eigenvectors	60
Exa 8.1.5	Eigenvalues and Eigenvectors	61
Exa 8.1.6	Eigenvalues and Eigenvectors	61
Exa 8.1.8	Eigenvalues and Eigenvectors	62
Exa 8.2.1	Diagonalization	62
Exa 8.2.3	Diagonalization	63
Exa 8.3.1	Diagonalization of Symmetric Matrices	63
Exa 8.3.5	Diagonalization of Symmetric Matrices	64
Exa 9.2.5	Differential Equations	65
Exa 9.2.6	Differential Equations	65
Exa 9.4.7	Quadratic Forms	66
Exa 10.1.2	Definition and Examples	68
Exa 10.2.12	The Kernel and Range of a Linear Transformation	69

Chapter 1

Linear Equations and Matrices

R code Exa 1.1.1 Linear Systems

```
1 #Page No. 2
2
3 A <- matrix(c(1,1,0.05,0.09),2,2,T)
4 b <- matrix(c(100000,7800),2,1,T)
5
6 ans <-solve(A,b)
7
8 cat("value of x :", ans[1], "\n")
9 cat("value of y :", ans[2])
```

R code Exa 1.1.3 Linear Systems

```
1 #Page No. 3
2
3 A= matrix(c(1,2,3,2,-3,2,3,1,-1),3,3,T)
4 b= matrix(c(6,14,-2),3,1,T)
5
6 ans <-solve(A,b)
```



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

R code Exa 1.1.4 Linear Systems

```
1 #Page No. 4
2
3 A=matrix(c(1,2,-3,2,1,-3),2,3,T)
4 A
5 b=matrix(c(-4,4),2,1,T)
6
7 asvd <-svd(A)
8 adiaq <- diag(1/asvd$d)
9 adiaq[2,1] =0
10 solution<- asvd$v %*% adiaq %*% t(asvd$u) %*% b
11
12 print(round(solution,0))
13
14 #The answer may vary due to difference in
   representation.
```

R code Exa 1.1.5 Linear Systems

```
1 #Page No. 4
2
3 A= matrix(c(1,2,2,-2,3,5),3,2,T)
4 b= matrix(c(10,-4,26),3,1,T)
5
6 First_term <- t(A) %*% A
7 Second_term <- t(A) %*% b
8
9 ans <- solve(First_term, Second_term)
```

```
10
11 cat(ans[1] , "\n")
12 cat(ans[2])
```

R code Exa 1.1.6 Linear Systems

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

R code Exa 1.2.3 Matrices

```
1 #Page No. 11
2
3
4 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 ")))
5
6 print(A)
```

R code Exa 1.2.10 Matrices

```
1 #Page No. 14
2
3
4 A<-matrix(c(1,2,-2,-1,4,3), c(2,3))
5 B<-matrix(c(0,1,2,3,-4,1), c(2,3))
6
7 ans <- A + B
8 print(ans)
```

R code Exa 1.2.12 Matrices

```
1 #Page No. 15
2
3 A<- matrix(c(2,4,3,2,-5,1), c(2,3))
4 B<-matrix(c(2,3,-1,5,3,-2), c(2,3))
5
6 ans <- A - B
7
8 print(ans)
```

R code Exa 1.2.14 Matrices

```
1 #Page No. 16
2
3 A1 <- matrix(c(0,2,1,-3,3,-2,5,4,-3), c(3,3))
4 A2 <- matrix(c(5,6,-1,2,2,-2,3,3,3), c(3,3))
5
6 solution <- (3 *A1) - (0.5 *A2)
7 print(solution)
8
9 #The answer may vary due to difference in
  representation.
```

R code Exa 1.2.15 Matrices

```
1 #Page No. 16
2
3 A<-matrix(c(4,0,-2,5,3,-2),c(2,3))
4 B<-matrix(c(6,3,0,2,-1,4,-4,2,3),c(3,3))
5 C<-matrix(c(5,-3,2,4,2,-3),c(3,2))
6 D<-matrix(c(3,-5,1),c(1,3))
7 E<-matrix(c(2,-1,3),c(3,1))
8
9 Atrans = t(A)
10 Btrans = t(B)
11 Ctrans = t(C)
12 Dtrans = t(D)
13 Etrans = t(E)
14
15 print(Atrans)
16 print(Btrans)
17 print(Ctrans)
18 print(Dtrans)
19 print(Etrans)
```

R code Exa 1.3.1 Dot Product and Matrix Multiplication

```
1 #Page No. 22
2
3
4 DOT =function(U,V)
5 {
6     w<-0
7     product_matrix <- U*V
8     for(num in product_matrix)
```

```

9      {
10         w<-w+num
11     }
12     return (w)
13 }
14
15 U<- c(1,-2,3,4)
16 V<- c(2,3,-2,1)
17
18 ans <- DOT(U,V)
19 cat("u.v is : ", ans)

```

R code Exa 1.3.4 Dot Product and Matrix Multiplication

```

1 #Page No. 23
2
3 A<- matrix(c(1,3,2,1,-1,4), nrow= 2, ncol= 3)
4 B<- matrix(c(-2,4,2,5,-3,1), nrow= 3, ncol=2 )
5
6 ans <- A%*%B
7 print(ans)

```

R code Exa 1.3.5 Dot Product and Matrix Multiplication

```

1 #Page No. 24
2
3 A<-matrix(c(1,4,0,-2,2,1,3,1,-2), nrow=3, ncol=3)
4 B<-matrix(c(1,3,-2,4,-1,2), nrow=3, ncol=2)
5
6 x<-A[c(3),c(1,2,3)]
7 y<-B[c(1,2,3),c(2)]
8
9 solution = x%*%y

```

```
10 print(solution)
```

R code Exa 1.3.10 Dot Product and Matrix Multiplication

```
1 #Page No. 25
2
3 A<-matrix(c(1,-1,2,3),c(2,2))
4 B<-matrix(c(2,0,1,1),c(2,2))
5
6 AB<- A%*%B
7 BA<- B%*%A
8
9 Check=function(x,y)
10 {
11   w<-identical(x,y)
12   return(w)
13 }
14
15 ans <- Check(AB,BA)
16 print(ans)
17
18 #The answer may vary due to difference in
   representation.
```

R code Exa 1.3.21 Dot Product and Matrix Multiplication

```
1 #Page No. 30
2
3 A<- matrix(c(1,0,2,0,0,2,0,1,1,3,-4,0,0,-1,0,3),c
   (4,4))
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)
```

```

5 C<- A**%B
6
7 Aii= A[c(1,2),c(1,2)]
8 Bii= B[c(1,2),c(1,2,3)]
9 Aij= A[c(1,2),c(3,4)]
10 Bji= B[c(3,4),c(1,2,3)]
11
12 Cii <- Aii **% Bii + Aij **% Bji
13 print(Cii)

```

R code Exa 1.3.23 Dot Product and Matrix Multiplication

```

1 #Page No. 32
2
3 A<- array(c(3,4,5,8))
4 sum<-0
5 for(num in A)
6 {
7   sum= sum+num
8 }
9 print(sum)

```

R code Exa 1.4.2 Properties of Matrix operations

```

1 #Page No. 41
2
3 A<- matrix(c(2,-4,3,5,4,-2),c(2,3))
4 minus= (-A)
5
6 solution <- A + minus
7 print(solution)
8

```

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

R code Exa 1.4.5 Properties of Matrix operations

```
1 #Page No. 42
2
3
4 A<-matrix(c(2,3,2,-1,3,2),c(2,3))
5 B<-matrix(c(1,2,3,0,2,-1),c(3,2))
6 C<-matrix(c(-1,1,2,2,0,-2),c(3,2))
7
8 First_method = A %*% (B + C)
9 print(First_method)
10
11 Second_method = A%*%B + A%*%C
12 print(Second_method)
13
14 if(identical(First_method, Second_method) == TRUE){
15   print("Condition satisfied")
16 } else
17 {
18   print(("Not satisfied"))
19 }
```

R code Exa 1.4.10 Properties of Matrix Operations

```
1 #Page No. 45
2
3 r<- -2
4 A<-matrix(c(1,-2,2,0,3,1),c(2,3))
5 B<-matrix(c(2,1,0,-1,4,-2),c(3,2))
6
```



```

7 First_method = A %*% (r * B)
8
9 Second_method = r * (A%*%B)
10
11 Third_method = (r*A) %*% B
12
13 if (identical(First_method , Second_method) &&
14     identical(First_method , Third_method) &&
15     identical(Second_method ,Third_method) == TRUE )
16     {
17     print("Theorem 1.3d is satisfied")
18 }else {
19     print("Theorem 1.3d is not satisfied")
20 }

```

R code Exa 1.4.11 Properties of Matrix Operations

```

1 #Page No. 46
2
3 A<-matrix(c(1,2,3,-1,2,3),c(2,3))
4 B<-matrix(c(0,2,3,1,2,-1),c(3,2))
5
6 Method_one = t(A%*%B)
7 Method_two = t(B) %*% t(A)
8
9 if(identical(Method_one, Method_two) == TRUE ){
10     print("Both methods are equal")
11 }
12 }else
13 {
14     print("Both methods are different")
15 }

```

R code Exa 1.5.3 Matrix Transformations

```
1 #Page No. 55
2
3 A<- matrix(c(1,-2,2,3),c(2,2))
4 v<- c('v1','v2')
5 W<- matrix(c(4,-1),c(2,1))
6 x<-solve(A,W)
7
8 cat(x[c(1),c(1)], "\n")
9 cat(x[c(2),c(1)])
```

R code Exa 1.6.4 Solutions of Linear Systems of Equations

```
1 #Page No. 64
2
3
4 A<-matrix(c(1,2,1,2,1,-2,4,3,2,3,2,3),c(3,4))
5
6 add<- 2* A[c(3),c(1,2,3,4)] + A[c(2),c(1,2,3,4)]
7 A[c(2),c(1,2,3,4)]<-add;A
8
9 B=A
10 temp=B[c(2),c(1,2,3,4)]
11 tem2=B[c(3),c(1,2,3,4)]
12 B[c(3),c(1,2,3,4)] = temp
13 B[c(2),c(1,2,3,4)] = tem2
14
15 C=B
16 adon<- 2* C[c(1),c(1,2,3,4)]
17
18 print("Row echelon matrix : ")
19 C[c(1),c(1,2,3,4)]<-adon;C
```

R code Exa 1.6.5 Solutions of Linear Systems of Equations

```
1 #Page No. 65
2
3 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))
4
5 Pcol = function(x,y)
6 {
7   count<-0
8   var<-0
9   arr1= A[c(1),c(1,2,3,4,5)]
10  for(num in arr1 )
11    { if(num==0)
12      {count=count+1
13        next}
14      else
15        return (count)
16        break
17    }
18
19  }
20
21 pivot = function(x,y)
22 {
23   arr<-A[c(1,2,3,4),c(1)]
24   for(num in arr)
25     if(num ==0)
26       next
27     else
28       return (num)
29
30 }
```

```

31 c<-Pcol(A)
32 p<-pivot(A)
33
34 A2=A
35 a<-A[c(1),c(1,2,3,4,5)]
36 c<-A[c(3),c(1,2,3,4,5)]
37
38 A2[c(1),c(1,2,3,4,5)]<-c
39 A2[c(3),c(1,2,3,4,5)]<-a
40
41 mul<- A2[c(1),c(1,2,3,4,5)] * 1/p
42 A2[c(1),c(1,2,3,4,5)]<-mul;A2
43
44 A3<-A2
45 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)]
46
47 B<- A3[c(2,3,4),c(1,2,3,4,5)]
48
49 c<-Pcol(B)
50 p2<-pivot(B)
51
52 a<-B[c(1),c(1,2,3,4,5)]
53 b<-B[c(2),c(1,2,3,4,5)]
54 B[c(1),c(1,2,3,4,5)]<-b
55 B[c(2),c(1,2,3,4,5)]<-a
56
57 app12<-B[c(1),c(1,2,3,4,5)]* 1/p2
58 B[c(1),c(1,2,3,4,5)]<-app12;B
59 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)]
60
61 C<- B[c(2,3),c(1,2,3,4,5)]
62 c<-Pcol(C)
63 p3<-pivot(C)
64
65 adik<- C[c(1),c(1,2,3,4,5)] * 1/p3
66 C[c(1),c(1,2,3,4,5)]<-adik;C

```

```

67 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)]
68 D<-C[c(2),c(1,2,3,4,5)]
69 first<-A3[c(1),c(1,2,3,4,5)]
70 second<-B[c(1),c(1,2,3,4,5)]
71 third<-C[c(1),c(1,2,3,4,5)]
72 fourth<-D
73
74 H<-matrix(first)
75 l<-cbind(H,second)
76 cbind(l,third)
77 k<-cbind(l,third)
78 cbind(k,fourth)
79 z<-cbind(k,fourth)
80 solution<-t(z)
81
82 print(solution)
83
84 #The answer may vary due to difference in
    representation.

```

R code Exa 1.7.4 The Inverse of a Matrix

```

1 #Page No. 93
2
3 A<-matrix(c(1,3,2,4), c(2,2))
4
5 first<-solve(A)
6 trans<-t(first)
7
8 second<-t(A)
9 inv<-solve(second)
10
11
12 if(all.equal(inv,trans) == TRUE){

```

```

13  print(trans)
14  print(inv)
15  print(" Equal")
16 }else
17 {
18  print(" Not Equal")
19 }
20
21 #The answer may vary due to difference in
    representation.

```

R code Exa 1.7.5 The Inverse of a Matrix

```

1 #Page No. 96
2
3 A<-matrix(c(1,0,5,1,2,5,1,3,1),c(3,3))
4
5 Ainv<- solve(A)
6
7 print(Ainv)
8
9 #The answer may vary due to difference in
    representation.

```

R code Exa 1.7.6 The Inverse of a Matrix

```

1 #Page No. 97
2
3 A<-matrix(c(1,1,5,2,-2,-2,-3,1,-3),c(3,3))
4
5 A1<-A
6 A1[c(2),c(1,2,3)]<- (A1[c(1),c(1,2,3)]*(-1)) + A1[
    c(2),c(1,2,3)]

```

```
7   A2<-A1
8   A2[c(3),c(1,2,3)]<- (A2[c(1),c(1,2,3)]*(-5)) + A2[
    c(3),c(1,2,3)]
9   A3<-A2
10  A3[c(3),c(1,2,3)]<- (A3[c(2),c(1,2,3)]*(-3)) + A3[
    c(3),c(1,2,3)]
11
12
13  arr<-t(A3)
14  count<-0
15
16  if(arr[7]==0 && arr[8]==0 && arr[9]==0){
17      count=count+1
18
19  if(count==1){
20      print(A3)
21      print("It is a singular matrix")
22  }
23  else
24      print("Not singular matrix")
25
26 }
```

Chapter 2

Applications of Linear Equations and Matrices

R code Exa 2.2.5 Graph Theory

```
1 #Page No. 127
2
3 library(igraph ,quietly=TRUE)
4 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)
5 plot.igraph(simple.graph)
6 get.adjacency(simple.graph)
7
8 #The answer may vary due to difference in
   representation.
```

R code Exa 2.2.6 Graph Theory

```
1 #Page No. 127
2
```



```

3 library(igraph, quietly=TRUE)
4 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)
5 plot.igraph(simple.graph)
6 get.adjacency(simple.graph)
7
8 #The answer may vary due to difference in
  representation.

```

R code Exa 2.2.14 Graph Theory

```

1 #Page No. 133
2
3 library(igraph)
4
5 simple.graph<- graph_from_literal(P1--P2,P2--P3,P3++
  P4, P2--P4,P3--P5,P5--P4,P1++P4,P5--P2)
6 plot.igraph(simple.graph)
7 A<-get.adjacency(simple.graph)
8 square=A%*%A
9 cube= square%*% A
10 fourth= cube%*% A
11 solution<- A + square + cube + fourth
12
13 check = function(M)
14 {
15   y<-0
16   count<-0
17   M<-matrix(1:25,nrow=5,ncol=5)
18
19   for(num in M)
20     if(num>0)
21       {
22         count=count+1

```

```

23         if(count==25)
24         {
25
26             y<-TRUE
27         }
28         else
29         { next }
30     }
31     else
32     {
33         break
34     }
35
36     return(y)
37
38 }
39
40
41 if(check(solution) == TRUE){
42     print(solution)
43     print("Strongly connected")
44 }else{
45     print("Not connected strongly")
46 }
47
48
49 #The answer may vary due to difference in
    representation.

```

R code Exa 2.4.1 Electrical Circuits

```

1 #Page No. 146
2
3 library(igraph,quietly = TRUE)
4

```

```

5 simple.graph<- graph_from_literal(a-b,b-c,c-d,d-e,e-
  f,f-a,c-f)
6 plot.igraph(simple.graph, mark.shape = -0.6)
7
8 E1<-40
9 E2<-120
10 E3<-80
11 R1<-5
12 R2<-10
13 R3<-10
14 R4<-30
15
16 coeff<-matrix(c(1,1,0,1,-2,1,-1,0,5),c(3,3))
17 const<-matrix(c(0,-16,20),c(3,1))
18 solution<-solve(coeff,const)
19 print(solution)
20
21 cat(solution[c(1)] ,"\n")
22 cat(solution[c(2)] , "\n")
23 cat(solution[c(3)])

```

R code Exa 2.5.1 Markov Chains

```

1 #Page No. 150
2
3
4 T<- matrix(c(1/2,1/2, 2/3,1/3),c(2,2),dimnames=list(
  c("D","R"),c("R","D"))) )
5 print(T)
6
7
8 #The answer may vary due to difference in
  representation.

```

R code Exa 2.5.3 Markov Chains

```
1 #Page No. 151
2
3 x0<-matrix(c(1,0))
4
5 T<- matrix(c(2/3,1/3,1/2,1/2),c(2,2),dimnames=list(c
  ("D", "R"),c("D", "R"))) )
6 print(T)
7 x1<- T %*% x0
8 round(x1,4)
9 x2<- T %*% x1
10 round(x2,4)
11 x3<- T %*% x2
12 round(x3,4)
13 x4<- T %*% x3
14 first_term<-round(x4,3)
15 x5<- T %*% x4
16 second_term <-round(x5,3)
17
18 if(identical(first_term, second_term) == TRUE){
19   cat(second_term[1]*100, "\n" )
20   cat(second_term[2]*100, "\n")
21 }
22
23 #The answer may slightly vary due to rounding off
  values.
```

R code Exa 2.5.7 Markov Chains

```
1 #Page No. 154
2
```

```

3  T<-matrix(c(0.2,0.8,1,0),c(2,2))
4  square<- T %*% T
5
6  check =function(x,y)
7  {
8      count<-0
9      for (num in square)
10         if(num>0)
11            {
12                count=count+1
13                if(count==4)
14                    {
15                        y<-TRUE
16                        return(y)
17                        next
18                    }
19                }else
20                {
21                    break
22                }
23            }
24
25  K<-check(square)
26  if(K==TRUE)
27      {
28          print(square)
29          print("Regular")
30      }else{
31          print("Not Regular")
32      }

```

Chapter 3

Determinants

R code Exa 3.1.5 Definition and Properties

```
1 #Page No. 184
2
3 a11<-2
4 a21<-4
5 a12<- -3
6 a22<- 5
7 A<-matrix(c(a11 , a21 , a12 , a22) , c(2 , 2))
8
9 F_Determinant= function(x,y)
10 {
11   x<- a11*a22
12   y<- a12*a21
13   Delta<-x - y
14   return(Delta)
15
16 }
17
18 cat(F_Determinant(A))
```

R code Exa 3.1.7 Definition and Properties

```
1 #Page No. 185
2
3 A<-matrix(c(1,2,3,2,1,1,3,3,2),c(3,3))
4 print(A)
5 cat(det(A))
```

R code Exa 3.1.9 Definition and Properties

```
1 #Page No. 186
2
3 A<-matrix(c(1,2,3,2,1,3,3,1,2),c(3,3))
4 trans<- t(A)
5 DetA<-det(A)
6 Dtrans<- det(trans)
7
8 cat(DetA, "\n")
9 cat(Dtrans, "\n")
```

R code Exa 3.1.10 Definition and Properties

```
1 #Page No. 186
2
3 A<-matrix(c(2,3,-1,2),c(2,2))
4 B<-matrix(c(3,2,2,-1),c(2,2))
5
6 Det_A<-det(A)
7 Det_B<-det(B)
8
9 cat(Det_A, "\n")
10 cat(Det_B, "\n")
```

R code Exa 3.1.11 Definition and Properties

```
1 #Page No. 187
2
3 A<-matrix(c(1,-1,1,2,0,2,3,7,3) , nrow=3, ncol=3)
4
5 check = function(a,b,d)
6 {
7   a<- A[c(1),c(1,2,3)]
8   b<- A[c(2),c(1,2,3)]
9   d<- A[c(3),c(1,2,3)]
10
11   x=identical(a,d)
12   y=identical(a,b)
13   z=identical(b,d)
14
15   if(x==TRUE || y==TRUE || z==TRUE)
16   {
17     return(det(A))
18   }
19   else
20     return("Non-identical rows")
21 }
22 cat(check(A))
```

R code Exa 3.1.14 Definition and Properties

```
1 #Page No. 187
2
3 A<-matrix(c(1,1,2,2,5,8,3,3,6) , c(3,3))
4
5 first_Det<-det(A)
```



```

6
7 f<- 2
8 A[c(3),c(1,2,3)] <- A[c(3),c(1,2,3)]/f
9 f2<- 3
10 A[c(1,2,3),c(3)]<- A[c(1,2,3),c(3)]/f2
11
12 second_Det<- det(A)*f*f2
13
14 if(identical(first_Det , second_Det)){
15   cat(first_Det , "\n")
16
17 }

```

R code Exa 3.1.18 Definition and Properties

```

1 #Page No. 191
2
3 A<-matrix(c(1,3,2,4),c(2,2))
4 B<-matrix(c(2,1,-1,2),c(2,2))
5
6 mul_AB<- A%*%B
7 det_AB= det(mul_AB)
8
9 det_A= det(A)
10 det_B= det(B)
11 det_product = det_A * det_B
12
13 x<-all.equal.numeric(det_AB,det_product)
14
15 if(x==TRUE){
16   cat(det_AB, "\n")
17   cat(det_product, "\n")
18
19 }

```

R code Exa 3.1.19 Definition and Properties

```
1 #Page No. 191
2
3 A<-matrix(c(1,3,2,4),c(2,2))
4 Ainv <- solve(A)
5
6 Det_A = det(A)
7 Det_Ainv = det(Ainv)
8
9 Det_A_reciprocal = 1/Det_A
10
11 x<-all.equal.numeric(Det_A_reciprocal , Det_Ainv)
12 if(x==TRUE){
13   cat(Det_A_reciprocal , "\n")
14   cat(Det_Ainv , "\n")
15 }
16
17 # The answer may vary due to difference in
    representation.
```

R code Exa 3.2.1 Cofactor Expansion and Applications

```
1 #Page No. 196
2
3 library(matlib)
4
5 A<-matrix(c(3,4,7,-1,5,1,2,6,2),c(3,3))
6
7 A12 <- cofactor(A,1,2)
8 A23 <- cofactor(A,2,3)
9 A31 <- cofactor(A,3,1)
```

```
10
11 solution <- array(c(A12, A23, A31), dim = c(3,1))
12 cat(solution)
```

R code Exa 3.2.2 Cofactor Expansion and Applications

```
1 #Page No. 197
2
3 library(matlib)
4 A<-matrix(c(1,-4,3,2,2,2,0,0,-3,1,0,-2,4,3,-3,3),c
   (4,4))
5
6 C11<- cofactor(A,3,1)
7 C12<- cofactor(A,3,2)
8 C13<- cofactor(A,3,3)
9 C14<- cofactor(A,3,4)
10
11 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)]
12
13 cat(solution)
```

R code Exa 3.2.5 Cofactor Expansion and Applications

```
1 #Page No. 200
2
3 library(matlib)
4
5 A<- matrix(c(3,-2,1,5,6,2,1,0,-3),nrow=3,byrow =
   TRUE)
6
7 print(adjoint(A))
```

R code Exa 3.2.7 Cofactor Expansion and Applications

```
1 #Page No. 202
2
3 library(matlib)
4 A<-matrix(c(3,-2,1,5,6,2,1,0,-3),nrow = 3,byrow =
      TRUE)
5 print(A)
6 det(A)
7 X<-solve(A)
8 print(X)
9
10 Check.inv = function(A.inverse)
11 {
12   det_A <- 1/det(A)
13   Adj_A <- adjoint(A)
14   A.inverse = Adj_A * det_A
15   return(A.inverse)
16 }
17
18 Check.inv(A)
19
20 #The answer may vary due to difference in
    representation.
```

R code Exa 3.2.9 Cofactor Expansion and Applications

```
1 #Page No. 206
2
3 library(matlib)
4
```

```

5  coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow
    = TRUE)
6  b<- matrix(c(1,4,-3),c(3,1))
7  first_D <- det(coeff)
8
9  coeff[,1]<- b;coeff
10 Second_D <-det(coeff)
11 x1<- Second_D / first_D
12 coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow
    = TRUE)
13
14 coeff[,2]<- b;coeff
15 third_D <- det(coeff)
16 x2<- third_D / first_D
17 coeff<-matrix(c(-2,3,-1,1,2,-1,-2,-1,1),nrow=3,byrow
    = TRUE)
18
19 coeff[,3] <- b;coeff
20 fourth_D<-det(coeff)
21 x3<- fourth_D / first_D
22
23 cat(x1, "\n")
24 cat(x2, "\n")
25 cat(x3)

```

Chapter 4

Vectors in R^n

R code Exa 4.1.3 Vectors in the Plane

```
1 #Page No. 216
2
3 library(graphics)
4 x <- c(0,2)
5 y <- c(0,3)
6
7 plot.new()
8 plot.default(c(0,4),c(0,4))
9 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

```
1 #Page No. 219
2
3 x1 <- matrix(c(2,-5), ncol = 1, byrow = T)
4 sum<-0
5 sq<-0
```

```

6
7  for(num in x1)
8  {
9    sq<- num^2
10   sum<- sq+sum
11   next
12  }
13
14  cat(sqrt(sum))
15
16  #The answer may vary due to difference in
    representation.

```

R code Exa 4.1.7 Vectors in the Plane

```

1  #Page No. 219
2
3  P<- matrix(c(3,2),c(2,1))
4  Q<-matrix(c(-1,5),c(2,1))
5
6  x<- (Q[c(1)] - P[c(1)])^2
7  y<- (Q[c(2)] - P[c(2)])^2
8
9  sol<- sqrt(x+y)
10 cat(sol)

```

R code Exa 4.1.8 Vectors in the Plane

```

1  #Page No. 220
2
3
4  tri_angle<- matrix(c(-1,3,2,4,1,6),c(3,2))
5

```

```
6 shape <- cbind(tri_angle, 1)
7
8
9 area_calc = function(a)
10 {
11   sol=0
12   sol<- det(a) / 2
13   return(sol)
14
15 }
16
17 cat(area_calc(shape))
```

R code Exa 4.1.9 Vectors in the Plane

```
1 #Page No. 221
2
3 u<-c(1,2)
4 v<-c(3,-4)
5
6 x0<- u[1] + v[1]
7 y0<- u[2] + v[2]
8
9 sol<- c(x0,y0)
10 plot(x0,y0)
11
12 cat(sol)
```

R code Exa 4.1.14 Vectors in the Plane

```
1 #Page No. 225
2
3 library(Matrix)
```



```

4
5 u<-c(2,4)
6 v<-c(-1,2)
7
8 DOT= function(u,v)
9 {
10   sis<- u[1]*v[1] + u[2]*v[2]
11   return(sis)
12
13 }
14 upper<- DOT(u,v)
15
16 magnitude =function(u)
17 {
18   sum<-0
19   s1<-0
20   m<-0
21   for(num in u)
22     { s1<- num^2
23       sum=sum+s1
24       next }
25   m<-sqrt(sum)
26   return(m)
27 }
28
29 mag_u<-magnitude(u)
30 mag_v<-magnitude(v)
31
32 cos_theta<- upper/(mag_u*mag_v)
33
34 cat(cos_theta)

```

R code Exa 4.1.15 Vectors in the Plane

1 #Page No. 225

```

2
3 u<-c(2,-4)
4 v<-c(4,2)
5
6 DOT= function(u,v)
7 {
8   sis<- u[1]*v[1] + u[2]*v[2]
9   return(sis)
10
11 }
12 sol<- DOT(u,v)
13
14 if(sol==0){
15   cat(sol, "\n")
16 }

```

R code Exa 4.1.16 Vectors in the Plane

```

1 #Page No. 226
2
3 library(matlib)
4
5 x<-c(-3,4)
6
7 unit_vector =function(u)
8 {
9   sum<-0
10  s1<-0
11  m<-0
12  for(num in u)
13  { s1<- num^2
14    sum=sum+s1
15    next }
16  m<-sqrt(sum)
17  sol<- sqrt( (u[1]/m)^2 + (u[2]/m)^2 )

```

```
18   return(sol)
19 }
20
21 cat(unit_vector(x))
```

R code Exa 4.2.2 n Vectors

```
1 #Page No. 230
2
3 u<-c(1,-2,3)
4 v<-c(2,3,-3)
5
6 sum<- c(u+v)
7
8 cat(sum)
```

R code Exa 4.2.3 n Vectors

```
1 #Page No. 230
2
3 u<-c(2,3,-1,2)
4 c=-2
5 final<- c*u
6
7 cat(final)
```

R code Exa 4.2.10 n Vectors

```
1 #Page No. 237
2
```

```

3 library(matlib)
4
5 u<-c(2,3,2,-1)
6 v<-c(4,2,1,3)
7
8 DOT= function(u,v)
9 {
10   sis<- u[1]*v[1] + u[2]*v[2] + u[3]*v[3] + u[4]*v
      [4]
11   return(sis)
12
13 }
14 sol<- DOT(u,v)
15
16 magnitude =function(u)
17 {
18   sum<-0
19   s1<-0
20   m<-0
21   for(num in u)
22   { s1<- num^2
23     sum=sum+s1
24     next
25   }
26   m<-sqrt(sum)
27   return(m)
28 }
29
30 mag_u<- magnitude(u)
31 mag_v<- magnitude(v)
32 mag_UV <- mag_u * mag_v
33
34 if(sol <= mag_UV)
35 {
36   cos_angle<- abs(sol/mag_UV )
37
38   if(cos_angle >= -1 && cos_angle <= 1)
39   {

```

```
40
41     cat(cos_angle, "\n")
42
43   }
44
45 }
46
47 #The answer may vary due to difference in
  representation.
```

R code Exa 4.2.13 n Vectors

```
1 #Page No. 239
2
3 library(matlib)
4
5 u<-c(1,0,0,1)
6 v<-c(0,1,1,0)
7
8 magnitude =function(u)
9 {
10   sum<-0
11   s1<-0
12   m<-0
13   for(num in u)
14     { s1<- num^2
15       sum=sum+s1
16       next
17     }
18   m<-sqrt(sum)
19   return(m)
20 }
21
22 mag_u<- magnitude(u)
23 mag_v<- magnitude(v)
```

```
24 magU_V <- mag_u + mag_v
25 mag_UV <- magnitude(u+v)
26
27 if(mag_UV <= magU_V)
28   {
29     cat(mag_UV, "\n")
30     cat(magU_V, "\n")
31
32   }
33
34 #The answer may vary due to difference in
    representation.
```

R code Exa 4.3.2 Linear Transformations

```
1 #Page No. 249
2
3 Li<-c(2,-1)
4 Lj<-c(3,1)
5 Lk<-c(-1,2)
6
7 sol<- c(-3,4,2)
8 sol<- -3*Li + 4*Lj + 2*Lk
9
10 cat(sol)
```

Chapter 5

Applications of Vectors in R2 and R3

R code Exa 5.1.1 Cross Product in R3

```
1 #Page No. 259
2
3 vector.cross <- function(a, b)
4   {
5     if(length(a)!=3 || length(b)!=3){
6       stop("Cross product is only defined for 3D
7         vectors.");
8     }
9     i1 <- c(2,3,1)
10    i2 <- c(3,1,2)
11    sol<-(a[i1]*b[i2] - a[i2]*b[i1])
12    return(sol)
13  }
14 u<-c(2,1,2)
15 v<-c(3,-1,-3)
16
17 cat(vector.cross(u,v), "\n")
18
```

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

R code Exa 5.1.4 Cross Product in R3

```
1 #Page No. 262
2
3 library(utils)
4 library(matlib)
5
6 P1<-matrix(c(2,2,4),byrow = T)
7 P2<-matrix(c(-1,0,5),byrow= T)
8 P3<-matrix(c(3,4,3),byrow=T)
9 vector.cross <- function(a, b)
10 {
11
12     cal<- b[c(1)]-a[c(1)]
13     cal2<- b[c(2)]-a[c(2)]
14     cal3<- b[c(3)]-a[c(3)]
15     cout<- matrix(c(cal,cal2,cal3),byrow = T)
16     print(cout)
17     dd<-matrix(c(cal),byrow=T)
18     ff<-rbind(dd,cal2)
19     gg<-rbind(ff,cal3)
20     return(gg)
21
22 }
23 u= vector.cross(P1, P2)
24 v= vector.cross(P1, P3)
25
26 CrossProduct<- function(x, y, i=1:3)
27 {
28
29     To3D <- function(x) head(c(x, rep(0, 3)), 3)
30     x <- To3D(x)
```



```

31   y <- To3D(y)
32
33   Index3D <- function(i) (i - 1) %% 3 + 1
34
35   return (x[Index3D(i + 1)] * y[Index3D(i + 2)] -
36           x[Index3D(i + 2)] * y[Index3D(i + 1)])
37 }
38
39 new<-CrossProduct(u,v)
40
41 Area.triangle= function(x)
42 {
43   half<-0
44   for(num in x)
45   {
46     half<-half+ (num/2)^2
47     half
48   }
49 }
50 return(sqrt(half))
51 }
52
53 cat(Area.triangle(new), "\n")
54
55 #The answer may vary due to difference in
    representation.

```

R code Exa 5.1.6 Cross Product in R3

```

1 #Page No. 263
2
3 u<-c(1,-2,3)
4 v<-c(1,3,1)
5 w<-c(2,1,2)
6 mat<-matrix(c(u,v,w),nrow=3,byrow = T)

```

```

7
8 CrossProduct= function(x, y, i=1:3)
9 {
10   To3D <- function(x) head(c(x, rep(0, 3)), 3)
11   x <- To3D(x)
12   y <- To3D(y)
13
14   Index3D <- function(i) (i - 1) %% 3 + 1
15
16   return (x[Index3D(i + 1)] * y[Index3D(i + 2)] -
17           x[Index3D(i + 2)] * y[Index3D(i + 1)])
18 }
19
20 vw_cross<-CrossProduct(v,w)
21
22 DOT= function(u,v)
23 {
24   sis<- u[1]*v[1] + u[2]*v[2] + u[3]*v[3]
25   return(sis)
26
27 }
28 uv_dot<-DOT(u,vw_cross)
29
30 vol1<- abs(uv_dot)
31 vol2<- abs(det(mat))
32
33 cat(vol1, "\n")
34 cat(vol2)

```

Chapter 6

Real Vector Spaces

R code Exa 6.1.7 Vector Spaces

```
1 #Page No. 275
2
3 library(polynom)
4 P1<-polynomial(coef = c(-1,5,-2,0,3))
5 P2<-polynomial(coef=c(1,2))
6 P3<-polynomial(coef=c(4))
7
8 coeff.p1<-coef(P1)
9 coeff.p2<-coef(P2)
10 coeff.p3<-coef(P3)
11
12 degree=function(coeff_p1)
13     {count=0
14
15         for(num in coeff_p1)
16         {
17             count=count+1
18         }
19         c<- count-1
20         return(c)
21     }
```

```
22 degree(coeff.p1)
23 degree(coeff.p2)
24 degree(coeff.p3)
```

R code Exa 6.2.11 Subspaces

```
1 #Page No. 283
2
3
4 v<-matrix(c(2,1,5),nrow=3,byrow=T)
5 A<-matrix(c(1,1,1,2,0,1,1,2,0),nrow=3,byrow=T)
6 v1<-A[c(1,2,3),c(1)]
7 v2<-A[c(1,2,3),c(2)]
8 v3<-A[c(1,2,3),c(3)]
9
10 c_matrix<-solve(A,v)
11 cat(c_matrix, "\n")
```

R code Exa 6.2.12 Subspaces

```
1 #Page No. 284
2
3 s1<-matrix(c(1,0,0,0,0,0),c(2,3))
4 s2<-matrix(c(0,0,1,0,0,0),c(2,3))
5 s3<-matrix(c(0,0,0,1,0,0),c(2,3))
6 s4<-matrix(c(0,0,0,0,0,1),c(2,3))
7
8 matrix_model<-matrix(c('a',0,'b','c','0','d'),c
9 (2,2))
10 a<-1
11 b<-3
12 c<-5
```

```
13 d<-7
14 span_S <- (a * s1) + (b * s2) + (c * s3) + (d * s4)
15
16 print(span_S)
17
18 #The answer may vary due to difference in
    representation.
```

R code Exa 6.3.2 Linear Independence

```
1 #Page No. 292
2
3 s1<-matrix(c(1,0,0,0),c(2,2))
4 s2<-matrix(c(0,1,1,0),c(2,2))
5 s3<-matrix(c(0,0,0,1),c(2,2))
6
7 matrix_model<-matrix(c('a','b','b','c'),c(2,2))
8
9 a<-3
10 b<-6
11 c<-9
12
13 span_S <- (a * s1) + (b * s2) + (c * s3)
14
15 print(span_S)
16
17 #The answer may vary due to difference in
    representation.
```

R code Exa 6.3.8 Linear Independence

```
1 #Page No. 295
2
```

```

3 v1<- c(1,0,1,2)
4 v2<- c(0,1,1,2)
5 v3<- c(1,1,1,3)
6
7 A<-matrix(c(v1,v2,v3),nrow=4)
8 b<-matrix(c(0,0,0,0))
9
10 first<- t(A)%*%A
11 second<- t(A)%*%b
12 ans_matrix <-solve(first,second)
13
14 count<-0
15 for(num in ans_matrix)
16 {
17   num=num+1
18   if(num == 1)
19     count= count+1
20
21   else
22     print("Linearly dependent")
23
24 }
25
26 if(count == 3){
27   cat(ans_matrix, "\n")
28
29 }

```

R code Exa 6.4.2 Basis and Dimension

```

1 #Page No. 303
2
3 v1<-c(1,0,1,0)
4 v2<-c(0,1,-1,2)
5 v3<-c(0,2,2,1)

```

```

6 v4<-c(1,0,0,1)
7
8 A<-matrix(c(v1,v2,v3,v4),c(4,4))
9 b<-matrix(c(0,0,0,0),c(4,1))
10 Sol<-solve(A,b)
11
12 count<-0
13 for(num in Sol)
14 {
15     num=num+1
16     if(num == 1)
17         count= count+1
18
19     else
20         print("Linearly dependent")
21
22 }
23
24
25
26 if(count == 4){
27     cat(Sol, "\n")
28 }

```

R code Exa 6.4.5 Basis and Dimension

```

1 #page 309
2
3 v1<-c(1,2,-2,1)
4 v2<-c(-3,0,-4,3)
5 v3<-c(2,1,1,-1)
6 v4<-c(-3,3,-9,-6)
7 v5<-c(9,3,7,-6)
8
9 library(matlib)

```

```

10 A<-matrix(c(v1,v2,v3,v4,v5),c(4,5))
11 A
12 b<-matrix(c(0,0,0,0),c(4,1))
13 echelForm<-echelon(A,b)
14 echelForm<-echelForm[,-4]
15 echelForm
16
17 V_1<-echelForm[c(1,2,3,4),c(1)]
18 V_2<-echelForm[c(1,2,3,4),c(2)]
19 V_3<-echelForm[c(1,2,3,4),c(3)]
20 V_4<-echelForm[c(1,2,3,4),c(4)]
21 V_5<-echelForm[c(1,2,3,4),c(5)]
22
23 check= function(x)
24 {
25     p<-0
26     count<-0
27     ans<-0
28     for(num in x)
29     {
30         p=p+num
31         count=count+1
32         if(count==4)
33         {
34             ans<-p*p*p
35
36             if(ans==1)
37                 print("Leading one")
38             else
39                 print("Not Leading one")
40         }
41     else
42     {
43         next
44     }
45
46
47 }

```



```
48
49 }
50
51 check(V_1)
52 check(V_2)
53 check(V_3)
54 check(V_4)
55 check(V_5)
```

R code Exa 6.5.3 Homogeneous Systems

```
1 #Page No. 322
2
3 library(matlib)
4
5 A<-matrix(c(1,3,5,-1),c(2,2))
6 lamda<- -10:10
7
8 for(num in lamda)
9 {
10   MAT<-matrix(c(num-1,-3,-5,num+1),c(2,2))
11   d<-det(MAT)
12
13   if(d==0)
14   {
15     print(num)
16     print(-num)
17
18   }
19   next
20 }
```

R code Exa 6.6.5 The Rank of a Matrix and Applications

```

1 #Page No. 333
2
3 library(matlib)
4
5 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))
6 E<-echelon(A)
7
8 N<-nrow(E)-c(R(E))
9
10 cat(c(R(E)), "\n")
11 cat(N)

```

R code Exa 6.6.8 The Rank of a Matrix and Applications

```

1 #Page No. 336
2
3 library(matlib)
4 A<-matrix(c(1,1,1,2,1,3,0,-3,3),c(3,3))
5 E<-echelon(A)
6
7 cat(c(R(E)), "\n"))

```

R code Exa 6.7.1 Coordinates and Changes of Basis

```

1 #Page No. 341
2
3 library(matlib)
4
5 v1=c(1,1,0,0)
6 v2=c(2,0,1,0)
7 v3=c(0,1,2,-1)

```

```
8 v4=c(0,1,-1,0)
9
10 mat<- matrix(c(v1,v2,v3,v4),c(4,4))
11 v<-matrix(c(1,2,-6,2),c(4,1))
12 E<-echelon(mat,v)
13
14 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))
15
16 print(Vs)
```

Chapter 7

Applications of Real vector Spaces

R code Exa 7.2.1 Least Squares

```
1 #Page No. 380
2
3 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))
4 b<-matrix(c(1,5,-2,1,3,5),c(6,1))
5
6 trans<-t(A)
7 LHS<-trans**%A
8 RHS<-trans**%b
9
10 solve(LHS,RHS)
11
12 #The answer may slightly vary due to rounding off
    values.
```

R code Exa 7.2.2 Least Squares

```
1 #Page No. 381
2
3 library(matlib)
4
5 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))
6 b<-matrix(c(1,5,-2,1,3,5),c(6,1))
7
8 qr<-QR(A)
9
10 Qknown<-qr$Q
11 Qtrans<-t(Qknown)
12 RHS<-Qtrans%*%b
13 Rknown<-qr$R
14
15 solution_1<-solve(Rknown,RHS)
16
17 trans<-t(A)
18 LHS<-trans%*%A
19 RHS<-trans%*%b
20 solution_2<-solve(LHS,RHS)
21
22 print(solution_1)
23
24 if(all.equal(solution_1 , solution_2)){
25   print("Equal")
26 }else{
27   print("Unequal")
28 }
29
30 #The answer may slightly vary due to rounding off
    values.
```

R code Exa 7.2.4 Least Squares

```
1 #Page No. 384
2
3 library(polynom)
4
5 b=matrix(c(4.5,5.5,5.7,6.6,7.0,7.7,8.5,8.7,9.5,9.7),
6         c(10,1))
7 A=matrix(c
8         (3,4,5,6,7,8,9,10,11,12,1,1,1,1,1,1,1,1,1,1),c
9         (10,2))
10 x=c('b1','b0')
11
12 trans<-t(A)
13 LHS<-trans %*% A
14 RHS<-trans %*% b
15 sol<-solve(LHS,RHS)
16 sol
17
18 b1<-sol[c(1)]
19 b0<-sol[c(2)]
20
21 y<-polynomial(coef=c(b0,b1))
22
23 x<-30
24 y<- b1*(x) + b0
25
26 cat(round(y, 3), "\n")
27
28 #The answer may slightly vary due to rounding off
29 values.
```

Chapter 8

Eigenvalues Eigenvectors and Diagonalization

R code Exa 8.1.2 Eigenvalues and Eigenvectors

```
1 #Page No. 409
2
3 A<-matrix(c(0,1/2,1/2,0),c(2,2))
4
5 ans<- eigen(A,only.values=FALSE, EISPACK = FALSE)
6
7 print(ans$vectors)
8 cat(ans$values, "\n")
9
10 #The answer may vary due to difference in
    representation.
```

R code Exa 8.1.3 Eigenvalues and Eigenvectors

```
1 #Page No. 410
2
```

```

3 A<-matrix(c(0,0,0,1),c(2,2))
4
5 ans<- eigen(A, only.value=FALSE, EISPACK = FALSE)
6
7 cat(ans$values, "\n")
8 print(ans$vectors)
9
10 #The answer may vary due to difference in
    representation.

```

R code Exa 8.1.5 Eigenvalues and Eigenvectors

```

1 #Page No. 412
2
3 library(pracma)
4 library(polynom)
5
6 lamda<-0
7 A<-matrix(c(lamda-1,-1,-4,-2,lamda-0,4,1,-1,lamda-5)
    ,c(3,3))
8
9 cpol<-charpoly(A)
10 roots<-polyroot(cpol)
11 print(roots)
12 polynomial(coef=c(-cpol[c(4)],cpol[c(3)],-cpol[c(2)
    ],cpol[c(1)]))
13
14 #The answer may vary due to difference in
    representation.

```

R code Exa 8.1.6 Eigenvalues and Eigenvectors

```

1 #Page No. 414

```



```

2
3 A<-matrix(c(1L,1L,4L,2L,0L,-4L,-1L,1L,5L),c(3,3))
4 ans<- eigen(A)
5
6 cat(ans$values, "\n")
7 print(ans$vectors)
8
9 #The answer may vary due to difference in
  representation.

```

R code Exa 8.1.8 Eigenvalues and Eigenvectors

```

1 #Page No. 419
2
3 A<- matrix(c(0,1/2,0,0,0,1/3,6,0,0),c(3,3))
4
5 ans<-eigen(A)
6 values<-ans$values
7
8 vect<- ans$vectors
9
10 solution_vector<- matrix(data=Re(vect[,3]),c(3,1),T)
11
12 cat(Re(values[c(3)]), "\n")
13 print(solution_vector)
14
15 #The answer may vary due to difference in
  representation.

```

R code Exa 8.2.1 Diagonalization

```

1 #Page No. 422
2

```

```

3 A<-matrix(c(1,-2,1,4),c(2,2))
4 P<-matrix(c(1,1,1,2),c(2,2))
5 invo<- solve(P)
6
7 B<- invo %*% A %*% P
8 print(B)

```

R code Exa 8.2.3 Diagonalization

```

1 #Page No. 424
2
3 A<-matrix(c(1,-2,1,4),c(2,2))
4 ev<-eigen(A)
5 vect<- ev$vectors
6 print(vect)
7 count<-0
8 for(num in vect){
9   if(num != 0)
10     count = count + 1
11 }
12
13 if(count == 4){
14   P<-matrix(c(-vect[c(1,2),c(1)],-vect[c(1,2),c(2)])
15             ,nrow=2)
16   invo<- solve(P)
17   ans<- invo %*% A %*% P
18   print(ans)
19
20 }

```

R code Exa 8.3.1 Diagonalization of Symmetric Matrices

```

1 #Page No. 434
2
3
4 A<- matrix(c(0,0,-2,0,-2,0,-2,0,3),c(3,3))
5 ev<-eigen(A)
6
7 v<-ev$values
8 D<-diag(v)
9
10 print(D)

```

R code Exa 8.3.5 Diagonalization of Symmetric Matrices

```

1 #Page No. 439
2
3 A<-matrix(c(1,2,0,0,2,1,0,0,0,0,1,2,0,0,2,1),c(4,4))
4
5 ev<-eigen(A)
6 vect<- ev$vectors
7
8 x1<- vect[c(1,2,3,4),c(4)]
9 x2<- vect[c(1,2,3,4),c(3)]
10 x3<- vect[c(1,2,3,4),c(2)]
11 x4<- vect[c(1,2,3,4),c(1)]
12
13 ans<- matrix(c(x1,x2,x3,x4),c(4,4))
14 print(ans)

```

Chapter 9

Applications of Eigenvalues and Eigenvectors

R code Exa 9.2.5 Differential Equations

```
1 #Page No. 457
2
3 library(matlib)
4
5 X<-matrix(c(4,6,8),c(3,1))
6 P<-matrix(c(1,1,1,1,2,4,1,4,16),c(3,3))
7
8 E<-echelon(P,X)
9 b<-E[c(1,2,3),c(4)]
10
11 cat(b[c(1)], "\n")
12 cat(b[c(2)], "\n")
13 cat(b[c(3)], "\n")
```

R code Exa 9.2.6 Differential Equations

```

1 #Page No. 457
2
3 A<-matrix(c(1,0,0,0,3,-2,0,-2,3),c(3,3))
4
5 ev<-eigen(A)
6 vect<- round(ev$vectors, 0)
7 value<- Re(ev$values)
8 print(vect)
9
10 cat(value, "\n")

```

R code Exa 9.4.7 Quadratic Forms

```

1 #Page No. 481
2
3 library(matrixcalc)
4
5 A<-matrix(c(0,0,0,0,3,4,0,4,-3),c(3,3))
6 E<-eigen(A)
7
8 value<- E$values
9 x<-value[c(2)]
10 y<-value[c(3)]
11 value[c(2)]=y
12 value[c(3)]=x
13 D<-diag(value)
14
15 k<-1/sqrt(5)
16 H<-matrix(c(k,0,0,0,k,0,0,0,1),c(3,3))
17
18 D1<- t(H) %*% D %*% H
19 rank<-matrix.rank(D1)
20
21 cat(value, "\n")
22 cat(rank, "\n")

```


Chapter 10

Linear Transformations and Matrices

R code Exa 10.1.2 Definition and Examples

```
1 #Page No. 503
2
3 p<-c(3,2,4)
4 q<-c(4,3,3)
5 t<-5
6
7 LHS = t*(p+q) + t^2
8 RHS = (t*p + t^2) + (t*q + t^2)
9
10 check =function(methodX,methodY)
11 {
12     result<-identical(methodX, methodY)
13     if(result==FALSE)
14     {
15         print("Non-Linear transformation")
16
17     }else
18     {
19         print("linear transformation")

```

```
20     }
21
22 }
23
24 check(LHS , RHS)
```

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

```
1 #Page No. 513
2
3 library(matlib)
4
5 a1<- c(1,0,1)
6 a2<- c(1,0,0)
7 a3<- c(0,1,1)
8 a4<- c(0,1,0)
9
10 S<-matrix(c(a1,a2,a3,a4),c(3,4))
11 b<-matrix(c(0,0,0),c(3,1))
12 E<-echelon(S,b)
13
14 basis =function(q)
15 {
16     sum<-0
17     for(num in q)
18     {
19         sum<-num^2 +sum
20     }
21 }
22 if(sum==1 )
23     cat(" basis: ", q, "\n")
24 else
25     cat("Not basis", "\n")
26
27
```



```
28 }  
29  
30 basis(E[c(1,2,3),c(1)])  
31 basis(E[c(1,2,3),c(2)])  
32 basis(E[c(1,2,3),c(3)])  
33 basis(E[c(1,2,3),c(4)])
```
