R Textbook Companion for Matrices and Linear Transformations by Charles G. Cullen[1]

Created by
Aswin S
B.Tech.
Mechanical Engineering
Government Engineering College, Barton Hill
Cross-Checked by
R TBC Team

September 21, 2021

[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: Matrices and Linear Transformations

Author: Charles G. Cullen

Publisher: Dover Publications, Inc., New York

Edition: 2

Year: 1990

ISBN: 978-0-486-66328-9

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

Matrices and Linear Systems

R code Exa 1.1 Matrix multiplication

```
#page - 20
#section - 1.3 MATRICES
#example 1


#first matrix A
A <- matrix(c(2,-1,0,4,3,-2), 2, 3, byrow=TRUE)
A
#second matrix B
B <- matrix(c(-4,0,6,-4,1,6), 3, 2, byrow=TRUE)
B
#multiplying A and B to AB
AB <- A %*% B

#printing AB
AB

#multiplying B and A to BA
BA <- B %*% A

#printing BA
```

```
BA
```

R code Exa 1.2 Matrix multiplication

```
#page - 21
#section - 1.3 MATRICES
#example 2


#first matrix A
A <- matrix(c(1,2,3,4), 2, 2, byrow=TRUE)
A
#second matrix B
B <- matrix(c(5,6,7,8), 2, 2, byrow=TRUE)
B
#multiplying A and B to AB
AB <- A %*% B

#printing AB
AB

#multiplying B and A to BA
BA <- B %*% A

#printing BA
BA

#function to compare two matrices
matequal <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y)
        && all(x == y)

#checking if AB and BA are equal
matequal(AB, BA)
```

R code Exa 1.3 Matrix multiplication

```
#page - 23
#section - 1.3 MATRICES
#example 4


#first matrix A
A <- matrix(c(2,-1,1,-1,2,-1,1,-1,2), 3, 3, byrow=
   TRUE)
A
#second matrix B
B <- matrix(c(6,-5,5,-5,6,-5,5,-5,6), 3, 3, byrow=
   TRUE)
B
#multiplying A and B to AB
AB <- A %*% B

#printing AB
AB

#multiplying B and A to BA
BA <- B %*% A

#printing BA
BA

#function to compare two matrices
matequal <- function(x, y)
       is.matrix(x) && is.matrix(y) && dim(x) == dim(y)
            && all(x == y)

#checking if AB and BA are equal
matequal(AB, BA)
```

R code Exa 1.4 Matrix multiplication

```r
#page - 23
#section - 1.3 MATRICES
#example 4


#first matrix A
A <- matrix(c(3,1,0,1,-1,2,1,1,1), 3, 3, byrow=TRUE)
A
#second matrix B
B <- matrix(c(3/8,1/8,-2/8,-1/8,-3/8,6/8,-2/8,2/8,4/8), 3, 3, byrow=TRUE)
B
#multiplying A and B to AB
AB <- A %*% B

#printing AB
AB

#multiplying B and A to BA
BA <- B %*% A

#printing BA
BA

#function to compare two matrices
matequal <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y) && all(x == y)

#checking if AB and BA are equal
matequal(AB, BA)

```

```
#identity matrix I
I <-diag(3)
I

#checking if AB and I are equal
matequal(AB, I)
```

R code Exa 1.5 Matrix addition

```
#page - 27
#section - 1.4 MATRIX ADDITION AND SCALAR MULTIPLICATION
#example 5


#first matrix A
A <- matrix(c(3,-2,4,6,0,-3,-1,7,10,9,5,0), 3, 4, byrow=TRUE)
A
#second matrix B
B <- matrix(c(2,4,6,8,0,1,3,5,7,9,-2,-4), 3, 4, byrow=TRUE)
B
#adding A and B to C
C <- A + B

#printing C
C
```

R code Exa 1.6 Matrix addition

```
#page - 27
```

```
#section - 1.4 MATRIX ADDITION AND SCALAR MULTIPLICATION
#example 6


#first matrix M
M <- matrix(c(3,0,7,-2,4,0,5,6,0), 3, 3)
M
#second matrix S
S <- matrix(c(-3,0,-7,2,-4,0,-5,-6,0), 3, 3)
S
#adding M and S to C
C <- M + S

#printing C
C
```

R code Exa 1.7 Matrix addition

```
#page - 30
#section - 1.4 MATRIX ADDITION AND SCALAR MULTIPLICATION
#example 7


#first matrix A
A <- matrix(c(1,4,2,0,-3,5), 2, 3)
A
#second matrix B
B <- matrix(c(4,8,-7,7,3,0), 2,3)
B
#performing the action 2A-3B and equating it to C
C <- (A*2) - (B*3)


```

```
#printing C
C
```

R code Exa 1.8 Transposition

```
#page - 32
#section - 1.5 TRANSPOSITION
#example 8

#matrix A
A <- matrix(c(1,2,3,4,5,6), 2, 3, byrow=TRUE)
A

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

#tanspose of matrix A - AT
AT = t(A)
AT

#tanspose of matrix B - BT
BT = t(B)
BT
```

R code Exa 1.9 Conjugate of complex matrix

```
#page - 34
#section - 1.5 TRANSPOSITION
#example 9

#complex matrix A
```

```
A <- matrix(c(3, 5+6*1i , 2, 0+3*1i, 2+3*1i , 2-1i )
    , 3, 2, byrow=TRUE)
A

#conjugate of complex matrix A, A_
A_ = Conj(A)
A_

#transpose of conjugate of complex matrix A, Astar
Astar = Conj(t(A))
Astar
```

R code Exa 1.10 Triangular matrix

```
#page - 40
#section - 1.7 SPECIAL KINDS OF MATRICES
#example 10

#matrix L
L <- matrix(c(2,0,0,3,0,0,-1,0,4), 3, 3, byrow=TRUE)
L

#matrix U
U <- matrix(c(0,7,9,0,0,-1,0,0,0), 3, 3, byrow=TRUE)
U

#creating Lcheck to create a lower triangular matrix
    of L
Lcheck <- L
lower.tri(Lcheck)
Lcheck[upper.tri(Lcheck)] <- 0
Lcheck

#creating Ucheck to create a upper triangular matrix
    of U
```

```
Ucheck <- U
lower.tri(Ucheck)
Ucheck[lower.tri(Ucheck)] <- 0
Ucheck

#function to compare two matrices
matequal <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y)
        && all(x == y)

#checking if L and U are lower and upper triangular
   matrix respectively
matequal(L, Lcheck)
matequal(U, Ucheck)
```

R code Exa 1.11 Symmetric and skew symmetric

```
#page - 41
#section - 1.7 SPECIAL KINDS OF MATRICES
#example 11

#matrix A
A <- matrix(c(1,2,3,4,2,5,-6,7,3,-6,8,-9,4,7,-9,0),
   4, 4, byrow=TRUE)
A

#matrix B
B <- matrix(c
   (0,1,2,3,-1,0,-4,5,-2,4,0,6,-3,-5,-6,-0), 4, 4,
   byrow=TRUE)
A

AT = t(A)
AT

```

```
BT = t(B)
BT

#function to compare two matrices

#symmetric check function
matsym <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y)
        && all(x == y)

#skew symmetric check function
matskew <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y)
        && all(x == -y)

#condition check
if(matsym(AT, A)){
    print("A is a symmetric matrix")
}else if(matskew(AT, A)){
    print("A is a skew symmetric matrix")
}else{
    print("none")
}
if(matsym(BT, B)){
    print("B is a symmetric matrix")
}else if(matskew(BT, B)){
    print("B is a skew symmetric matrix")
}else{
    print("none")
}
```

R code Exa 1.12 Hermitian matrix

```
#page - 42
#section - 1.5 TRANSPOSITION
```

```r
#example 12

#complex matrix A
A <- matrix(c(2, 5*1i , 2-3*1i, -5*1i, 3 , 4, 2+3*1i , 4, 0), 3, 3, byrow=TRUE)
A

#complex matrix B
B <- matrix(c(1i, 3 , 5*1i, -3, 0 , 2+3*1i, 5*1i, , 0), -2+3*1i, 3*1i), 3, 3, byrow=TRUE)
B

#conjugate transpose of complex matrix A, A_
Astar = Conj(t(A))
Astar

#conjugate transpose of complex matrix B, B_
Bstar = Conj(t(B))
Bstar

#function to compare two matrices

#Hermitian matrix check function
matHer <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y) && all(x == y)

#skew Hermitian matrix check function
matskewHer <- function(x, y)
    is.matrix(x) && is.matrix(y) && dim(x) == dim(y) && all(x == -y)

#condition check
if(matHer(Astar, A)){
    print("A is a Hermitian matrix")
}else if(matskewHer(Astar, A)){
    print("A is a skew Hermitian matrix")
}else{
```

```
    print("none")
}
if(matHer(Bstar, B)){
    print("B is a Hermitian matrix")
}else if(matskewHer(Bstar, B)){
    print("B is a skew Hermitian matrix")
}else{
    print("none")
}
```

R code Exa 1.13 Echelon

```
#page - 45
#section - 1.8 ROW EQUIVALENCE
#example 13

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(1,2,3,1,3,2,1,1,0,2,4,1,1,1,1,1), 4,
    4, byrow=TRUE)

#column matrix k
K <- c(3,7,1,4)

#reduced row-echelon form
echelon(A, K, verbose=TRUE, fractions=TRUE)
```

R code Exa 1.14 Inverse of matrix

```
#page - 53
#section - 1.9 ELEMENTARY MATRICES AND MATRIX INVERSES
#example 14

#included package - matlib

#for inverse functions
library(matlib)

#matrix A
A <- matrix(c(1,-1,2,3,2,-1,0,2,4,1,-11,-1,1,2,3,83), 4, 4, byrow=TRUE)
A

#determinant of A
det(A)

#inverse matrix of A, AI
(AI <- inv(A))
```

R code Exa 1.15 Inverse of matrix

```
#page - 55
#section - 1.9 ELEMENTARY MATRICES AND MATRIX INVERSES
#example 15

#included package - matlib

#for inverse functions
library(matlib)

#matrix B
B <- matrix(c(3,-1,2,2,1,1,1,-3,0), 3, 3, byrow=TRUE
```

```
    )
B

#determinant of matrix B
det(B)

#inverse of matrix B, BI
(BI  <- inv(B))
```

Chapter 2

Vector Spaces

R code Exa 2.1 Subspaces

```
#page - 74
#section - 2.2 SUBSPACES
#example 1

#let us represent 'a' with a value as we cannot
    multiply character matrix with numerical matrix
a = 7

#let unit vectors of F4*1 be E1, E2, E3, E,4
E1 <- matrix(c(1,0,0,0), 4, 1)
E2 <- matrix(c(0,1,0,0), 4, 1)
E3 <- matrix(c(0,0,1,0), 4, 1)
E4 <- matrix(c(0,0,0,1), 4, 1)

#thus vector A is
A = (E1 %*% a) + (E2 %*% a) + (E3 %*% a) + (E4 %*% a
    )
A
```

R code Exa 2.2 Subspaces

```
#page - 75
#section - 2.2 SUBSPACES
#example 2

#included package - matlib

#for echelon function
library(matlib)

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

#column matrix k
K <- c(-6,10,15)

#reduced row-echelon form
echelon(A, K, reduced=FALSE, verbose=TRUE, fractions
    =TRUE)
```

R code Exa 2.3 Echelon

```
#page - 80
#section - 2.3 LINEAR INDEPENDENCE AND BASES
#example 3

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(1,-2,1,2,-5,0,-1,3,1,2,0,10), 4, 3,
```

```
    byrow=TRUE)

#column matrix k
K <- c(0,0,0,0)

#reduced row-echelon form
echelon(A, K, reduced=FALSE, verbose=TRUE, fractions
    =TRUE)
```

R code Exa 2.4 Echelon

```
#page - 81
#section - 2.3 LINEAR INDEPENDENCE AND BASES
#example 4

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(-1,-2,2,0,1,3,1,1,1), 3, 3, byrow=TRUE
    )

#column matrix k
K <- c(0,0,0)

#reduced row-echelon form
echelon(A, K, reduced=FALSE, verbose=TRUE, fractions
    =TRUE)
```

R code Exa 2.5 Echelon

```
#page - 82
#section - 2.3 LINEAR INDEPENDENCE AND BASES
#example 5

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(1,2,3,3,2,1,0,2,4,1,1,1), 4, 3, byrow=
    TRUE)

#column matrix k
K <- c(1,1,1,1)

#reduced row-echelon form
B = echelon(A, K, reduced=TRUE, verbose=TRUE,
    fractions=TRUE)
```

R code Exa 2.6 Echelon

```
#page - 82
#section - 2.3 LINEAR INDEPENDENCE AND BASES
#example 6

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(3,-1,-1,-2,2,-2,-1,-1,3), 3, 3, byrow=
    TRUE)

```

```
#column matrix k
K <- c(-3,2,1)

#reduced row-echelon form
B = echelon(A, K, reduced=TRUE, verbose=TRUE,
    fractions=TRUE)
```

R code Exa 2.7 Rank of a matrix

```
#page - 88
#section - 2.4 THE RANK OF A MATRIX
#example 7

#included package - matlib
#included package - matrixcalc

#for echelon function
library(matlib)

#for rank calculation
library(matrixcalc)

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

#column matrix k
K <- c(2,-3,8)

#reduced row-echelon form
B = echelon(A, K, reduced=TRUE, verbose=TRUE,
    fractions=TRUE)

#rank of A
matrix.rank(A)
```

Chapter 3

Determinants

R code Exa 3.1 Determinant

```
#page - 103
#section - 3.1 DEFINITION OF THE DETERMINANT
#example 1

#matrix A
A <- matrix(c(1,2,3,2,4,1,1,3,0), 3, 3, byrow=TRUE)
A

#determinant of A
det(A)
```

R code Exa 3.2 Minor and cofactor

```
#page - 106
#section - 3.2 THE LAPLACE EXPANSION
#example 2

#matrix A
```

```
A <- matrix(c(1,2,3,4,5,6,7,8,9), 3, 3, byrow=TRUE)
A

# Minor and cofactor functions
minor <- function(A, i, j) A[-i,-j]
cofactor <- function(A, i, j) (-1)^(i+j) * det(minor
    (A,i,j))

#calculating Minor and cofactor

minor(A, 1, 2)
cofactor(A, 1, 2)

minor(A, 3, 3)
cofactor(A, 3, 3)
```

R code Exa 3.3 Determinant

```
#page - 108
#section - 3.2 THE LAPLACE EXPANSION
#example 3

#matrix A
A <- matrix(c(2,4,6,1,2,3,1,4,9), 3, 3, byrow=TRUE)
A

#determinant of A
det(A)
```

R code Exa 3.4 Determinant

```
#page - 108
#section - 3.2 THE LAPLACE EXPANSION
```

```
#example 4

#matrix A
A <- matrix(c(1,-1,2,3,2,2,0,2,4,1,-1,-1,1,2,3,0),
    4, 4, byrow=TRUE)
A

#determinant of A
det(A)
```

R code Exa 3.5 Adjoints

```
#page - 111
#section - 3.3 ADJOINTS AND INVERSES
#example 5

#matrix A
A <- matrix(c(2,-1,1,4,2,4,6,3,9), 3, 3)

# Minor and cofactor functions
minor <- function(A, i, j) det(A[-i,-j])
cofactor <- function(A, i, j) (-1)^(i+j) * minor(A,i
    ,j)

#Adjoint functions
adjoint <- function(x) {
  n <- nrow(x)
  B <- matrix(NA, n, n)
  for( i in 1:n )
    for( j in 1:n )
      B[j,i] <- cofactor(x, i, j)
  B
}
adjoint(A)
```

R code Exa 3.6 Laplace expansion

```
#page - 114
#section - 3.2 THE LAPLACE EXPANSION
#example 6

#column matrices
c1 <- matrix(c(1,1,2), 3, 1, byrow=TRUE)
c2 <- matrix(c(2,-1,3), 3, 1, byrow=TRUE)
c3 <- matrix(c(1,1,-1), 3, 1, byrow=TRUE)
c4 <- matrix(c(4,5,1), 3, 1, byrow=TRUE)

A <- matrix(c(c1,c2,c3), 3, 3)
B <- matrix(c(c4,c2,c3), 3, 3)
C <- matrix(c(c1,c4,c3), 3, 3)
D <- matrix(c(c1,c2,c4), 3, 3)
A
B
C
D
#solution of r,s and t
r = det(B)/det(A)
r

s=det(C)/det(A)
s

t=det(D)/det(A)
t
```

R code Exa 3.7 Rank of a matrix

```r
#page - 115
#section - 3.4 DETERMINANTS AND RANK
#example 7

#included package - matrixcalc

#for rank calculation
library(matrixcalc)

#matrix A
A <- matrix(c
  (1,-1,1,1,1,2,-1,-1,2,-2,1,-1,0,-3,-1,-1), 4, 4,
  byrow=TRUE)

#matrix N
N <- matrix(c(0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0), 4,
  4, byrow=TRUE)

#rank of A
matrix.rank(A)

#rank of N
matrix.rank(N)
```

Chapter 4

Linear Transformations

R code Exa 4.1 w coordinate matrices

```
#page - 123
#section - 4.2 MATRIX REPRESENTATION
#example 1

#w-coordinate matrices
Crdwb1 <- matrix(c(-2,5,6,-4), 4, 1, byrow=TRUE)
Crdwb2 <- matrix(c(-5,12,11,-5), 4, 1, byrow=TRUE)
Crdwb3 <- matrix(c(3,-5,-5,3), 4, 1, byrow=TRUE)

#matrix T
Tmat <- matrix(c(Crdwb1,Crdwb2,Crdwb3), 4, 3)
Tmat

Crdwv <- matrix(c(3,-5,7), 3, 1)
Crdwv

CrdwTv = Tmat %*% Crdwv
CrdwTv
```

R code Exa 4.2 Change of basis and similarity

```
#page - 131
#section - 4.4 CHANGE OF BASIS AND SIMILARITY
#example - 2

#included package - matlib

#for inverse functions
library(matlib)

#matrix M
M <- matrix(c(2,4,6,-1,2,3,1,4,9), 3, 3, byrow=TRUE)
M

#matrix P
P <- matrix(c(1,0,1,1,1,1,1,1,0), 3, 3, byrow=TRUE)
P

#inverse of matrix P, PI
(PI  <- inv(P))

Mtx = PI %*% M %*% P
Mtx
```

R code Exa 4.3 Change of basis and similarity

```
#page - 132
#section - 4.4 CHANGE OF BASIS AND SIMILARITY
#example - 3

#included Package - matlib

#for inverse functions
library(matlib)
```

```

#matrix M
M <- matrix(c(17,12,18,-16,-9,-24,-5,-4,-4), 3, 3,
    byrow=TRUE)
M

#matrix Q
Q <- matrix(c(10,-3,-3,-8,3,2,-3,1,1), 3, 3, byrow=
    TRUE)
Q

#inverse of matrix Q, QI
(QI  <- inv(Q))

Mtx = QI %*% M %*% Q
Mtx
```

R code Exa 4.4 Eigenvalues and eigenvectors

```
#page - 135
#section - 4.5 CHARACTERISTIC VECTORS AND
    CHARACTERISTIC VALUES
#example - 4

#matrix A
A <- matrix(c(7,-8,-8,9,-16,-18,-5,11,13), 3, 3,
    byrow=TRUE)
A

#matrix X
X <- matrix(c(1,3,-2), 3, 1, byrow=TRUE)
X

A %*% X

```

```r
#eigenvalues and eigenvectors
eigen(A)
```

R code Exa 4.5 Characteristic polynomial

```r
#page - 137
#section - 4.5 CHARACTERISTIC VECTORS AND
    CHARACTERISTIC VALUES
#example - 5

#included package - pracma

#for charpoly function
library(pracma)

#matrix A
A <- matrix(c(2,1,1,2,3,2,1,1,2), 3, 3, byrow=TRUE)
A

#characteristic polynomial
charpoly(A, info = FALSE)
```

R code Exa 4.6 Characteristic polynomial

```r
#page - 156
#section - 4.8 SCHUR'S THEOREM AND NORMAL MATRICES
#example - 6

#included package - pracma

#for charpoly function
library(pracma)

```

```r
#matrix A
B <- matrix(c(4,-1,1,-1,4,-1,1,-1,4), 3, 3, byrow=
  TRUE)
B

#characteristic polynomial
charpoly(B, info = FALSE)
```

Chapter 5

Similarity Part I

R code Exa 5.1 Minimum polynomial

```
#page - 167
#section - 5.1 THE CAYLEY-HAMILTON THEOREM
#example - 1

#included package - polynom

#for minimum polynomial function
library(polynom)

#matrix B
B <- matrix(c(17,-8,-12,14,46,-22,-35,41,-2,1,4,-4,4,-2,-2,3), 4, 4, byrow = TRUE)
B

eigVals <- eigen(B)$values
multEig <- table(eigVals)
k <- length(multEig)
minPoly <- 1
for(i in 1:k){
  poly.i <- polynomial(c(-as.numeric(names(multEig)[
```

```
      i]), 1))
  minPoly <- (minPoly*poly.i)
}

#minimum polynomial
minPoly
```

R code Exa 5.2 Characteristic polynomial

```
#page - 168
#section - 5.1 THE CAYLEY-HAMILTON THEOREM
#example - 2

#included packages - pracma, polynom

#for charpoly function
library(pracma)

#for minimum polynomial function
library(polynom)

#matrix S
S <- matrix(c(4,-1,1,-1,4,-1,1,-1,4), 3, 3, byrow =
    TRUE)
S

eigVals <- eigen(S)$values
multEig <- table(eigVals)
k <- length(multEig)
minPoly <- 1
for(i in 1:k){
  poly.i <- polynomial(c(-as.numeric(names(multEig)[
      i]), 1))
  minPoly <- (minPoly*poly.i)
}
```

```
#characteristic polynomial
charpoly(S, info = FALSE)

#minimum polynomial
minPoly
```

Chapter 6

Polynomials and Polynomial Matrices

R code Exa 6.1 Left and right functional value

```
#page - 208
#section - 6.4 MATRICES WITH POLYNOMIAL ELEMENTS
#example 1

#included package - exmp

#for power of a matrix
library(expm)

#constant matrices
c1 <- matrix(c(0,0,1,1), 2, 2, byrow=TRUE)
c2 <- matrix(c(0,1,0,1), 2, 2, byrow=TRUE)
c3 <- matrix(c(1,1,0,1), 2, 2, byrow=TRUE)

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

#square of A
```

```
A2 = A %^% 2
A2

#right functional value
Pr = c1%*%A2 + c2%*%A + c3
Pr

#left functional value
Pi = A2%*%c1 + A%*%c2 + c3
Pi

#function to compare two matrices
matequal <- function(x, y)
  is.matrix(x) && is.matrix(y) && dim(x) == dim(y) &
    & all(x == y)

matequal(Pr,Pi)
```

Chapter 8

Matrix Analysis

R code Exa 8.1 Primary functions

```
#page - 242
#section - 8.2 PRIMARY FUNCTIONS
#example 1


#first matrix A
A <- matrix(c(2,1,1,2,3,2,1,1,2), 2, 3, byrow=TRUE)
A

fn <- function(z)
  sin((pi/2)*z)

fn(A)
```

Chapter 9

Numerical Methods

R code Exa 9.1 Echelon

```
#page - 253
#section - 9.2 EXACT METHODS FOR SOLVING AX = K
#example 1

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c(2,1,-1,2,1,3,2,-3,-1,2,1,-1,2,-3,-1,4)
    , 4, 4, byrow=TRUE)

#column matrix k
K <- c(1,0,1,0)

#reduced row-echelon form
echelon(A, K, reduced=TRUE, verbose=TRUE, fractions=
   FALSE)
```

R code Exa 9.2 Echelon

```
#page - 254
#section - 9.2 EXACT METHODS FOR SOLVING AX = K
#example 2

#included package - matlib

#for echelon function
library(matlib)

#matrix A
A <- matrix(c
   (7,9,2,-1,4,-5,-7,2,3,-2,-5,-1,1,6,-4,-3), 4, 4,
   byrow=TRUE)

#column matrix k
K <- c(1,2,4,3)

#reduced row-echelon form
echelon(A, K, reduced=TRUE, verbose=TRUE, fractions=
   TRUE)
```

R code Exa 9.3 Inverse of matrix

```
#page - 259
#section - 9.2 EXACT METHODS FOR SOLVING AX = K
#example 3

#included package - matlib

#for inverse functions
```

```
library(matlib)

#matrix A
A <- matrix(c(1,0,2,-1,4,5,3,-1,0,1,8,5,-3,1,4,6,2,0,0,1,0,1,4,2,0), 5, 5, byrow=TRUE)
A

#determinant of matrix A
det(A)

#inverse of matrix A, AI
(AI <- inv(A))
```

R code Exa 9.4 Eigenvalues and eigenvectors

```
#page - 265
#section - 9.4 CHARACTERISTIC VALUES AND VECTORS
#example - 4

#matrix A
A <- matrix(c(1,1,3,1,-2,1,3,1,3), 3, 3, byrow=TRUE)
A

#matrix X
X <- matrix(c(1,1,1), 3, 1, byrow=TRUE)
X

#eigenvalues and eigenvectors
eigen(A)
```