Your score is 79 points out of 110 points. You have done good and please try to do more in the future.

Computational physics project two on linear algebra and project three on interpolation and curve fitting by using R-language

By Graduate student Melessew Nigussie
Bahir Dar University, Department of physics
January09,2007

1. Content

Linear Algebra

1. Matrices and vectors
2. Matrix and vector operations in R
3. Linear equations
4. Unsolvable problems
5. Ill-conditioned problems
6. Gauss elimination
7. Gauss-Jordan elimination and matrix inversion
8. LU decomposition
9. Matrix eigenvalue
10. Iterative solution

Interpolation and Curve Fitting

1. Linear interpolation and extrapolation

2. Lagrange Interpolation
3. Cubic spline interpolation
4. Curve fitting
5. Spectral analysis

2. Abstract

The purpose of this project is

- to exercise different problems on matrix, interpolation and curve fitting by using hand and R- language
- to practice how to analysis data
- to program functions which are useful to calculate different problems on matrix, interpolation and curve fitting in R-language

3. Introduction

Linear algebra is fundamental in numerical methods. In addition, R-software capabilities are built upon the matrix and vector operation. In R it is possible use basic operations and solve different problems like, system of lea[illegible]ns,unsolvable problems,Ill-conditioned problems,Gauss elimi[illegible]nvalue and iterative solutions based on matrix and other techniques.Finally, it is possible to analysis data based on the principle of interpolation(estimating the value the function from already measured data),curve fitting-reconciling the function to the given data and spectral analysis based on the discrete and continuous frequency spectrum generated by Fourier series and Fourier transform.

to

linear

4. Linear algebra

4.1. Matrix and Vector

4 points

Matrix

- Definition:Matrix is a rectangular array of numbers enclosed in a pair of brackets or parenthesis: $\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}$ this matrix is called ***m*** by ***n*** matrix. The subscript ***m*** shows the row of the matrix and ***n*** shows the column of the matrix. In,R,however the matrices are print out without brackets or parenthesis.

Vector

- Definition: Vectors are special forms of matrix. If $m > 1$ and $n = 1$, the above matrix becomes $\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{m1} \end{pmatrix}$

- with only one column, which is called ***a column vector*** or ***column matrix***.
- On the other hand, if $m = 1$and $n > 1$, the matrix becomes $\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \end{pmatrix}$ which has only one row and is called a ***row vector*** or ***row matrix.***
- Note in expressing vector as a matrix , the elements of row or column vectors are known as the ***component of vectors***

4.2. Special Matrices

Square Matrix

1. Definition: A matrix having the number rows and columns equal is called a ***square matrix***. Ex.A= $\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}$

2. Identity matrix. A square matrix of order n having unit elements in the principal or leading diagonal and zero elements everywhere

What is this symbol

else,is called a identity matrix and is denoted by I.EgI= $\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$

```
# This is the function that gives n rowed by n column square
 #identity matrix
II=function(nrow,ncol){
A=matrix(,nrow,ncol)
for(i in 1:nrow){
for(j in 1:ncol){
if(i==j){
 A[i,j]=1
}else{
A[i,j]=0
A
}
}
}
list(A,A)
}
    Example
 source("II.R")
```

Very good program

```
 nrow=4;ncol=4
 II(nrow,ncol)
[[1]]
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

3. Upper and Lower triangular matrix,Diagonal and Scalar matrices Definition A square matrix A of any orde space [illegible] said to be an upper triangular matrix if its elements aij= [illegible] where i,j are positive integers ranging over from 1 to n. Eg. A=

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a1n \\ 0 & a_{22} & a_{2n} & \dots & a_{2n} \\ o & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & ann \end{pmatrix}$$

4. A square matrix A of any order n, A=$[a_{ij}]$ is said to be a lower triangular matrix if its elementsa_{ij}=0 for $i < j$ where i,j are positive integers

ranging over from 1 to n. E.g., A= $\begin{pmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}$

5. A square matrix of order n which is both upper and lower triangular is called a diagonal matrix.Example, $\begin{pmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & o & \dots & 0 \end{pmatrix}$

6. In a diagonal matrix if all the diagonal elements are equal to the scalar quantity say λ that is a_{11}=a_{22}=...=a_{nn}=λ, then the matrix is called a scalar matrix. E.g. $\begin{pmatrix} \lambda & 0 & 0 & \dots & 0 \\ 0 & \lambda & 0 & \dots & 0 \\ 0 & 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \lambda \end{pmatrix}$

7. A square matrix A=$[a_{ij}]$ is known as singular matrix if its determinant is equal to zero.

8. A square matrix is A=$[a_{ij}]$ is known as non-singular matrix if its determinant is different from zero.

9. The transposed matrix
The transpose of $A_{mxn}$is defined to be the n x m matrix A^Tobtained by interchanging rows and columns in
A.Example, $A^T = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

```
 # this is a function that transposed the given matrix
T=function(A){
 n=nrow(A)# number of row of matrix A
 m=ncol(A) # number of column of matrix A
 tra=matrix(,m,n) # empty m by n matrix(the transpose of matrix A)
for(i in 1:n){
for(j in 1:m){
T=A[i,j];A[i,j]=tra[j,i];tra[j,i]=T
}
}
list(tra=tra)
}
   Example,the transpose of the above matrix by this function is

 getwd()
[1] "C:/Program Files/R/R-2.4.0rc"
```

```
 getwd()
[1] "C:/Documents and Settings/Tigist/My Documents/Melessew/function"
 source("T.R")
 A=matrix(c(1,3,5,2,4,6),3,2)
 A
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
 T(A)
$tra
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matrix and Vector Operations in R

9 points

In R ,matrix ,column and row vectors can be entered by the same rule. For example, the matrix B= $\begin{pmatrix} 1 & 6 \\ 5 & 2 \end{pmatrix}$ is entered into R by:

```
B=matrix(c(1,5,6,2),nrow=2,ncol=2)
         [,1] [,2]
      [1,] 1     6
      [2,] 5     2
```

A column or a row vector may be defined as a matrix of one column or one row ,respectively . For example,

```
A=matrix(c(1,2),nrow=1,ncol=2);(row vector)
    [,1] [,2]
[1,] 1  2
B=matrix(c(1,7),nrow=2,ncol=1);(column vector)
      [,1]
 [1,]  1
 [2,]  7
```

- The addition and subtraction of two arrays or vectors of the same length are defined as follows:

  by hand A=$\begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$, B=$\begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix}$
  A+B=$\begin{pmatrix} a_1+b_1 & a_2+b_2 & \dots & a_n+b_n \end{pmatrix}$
  A-B=$\begin{pmatrix} a_1-b_1 & a_2-b_2 & \dots & a_n-b_n \end{pmatrix}$

```
by R, example
A=matrix(c(1,2,3),nrow=1,ncol=3)
B=matrix(c(4,5,6),nrow=1,ncol=3)
A+B  [,1] [,2] [,3]
 [1,] 5    7    9
A-B   [,1] [,2] [,3]
   [1,] -3  -3  -3
```

- In R element by element multiplication of vectors can be performed by the operator '*'.

```
Example, A*B= [,1] [,2] [,3]
          [1,] 4   10   18
```

- In R element by element division calculated by the operator'/'.

```
Example,  A/B= [,1] [,2] [,3]
               [1,]0.25 0.4   0.5
```

Example.Program a function that calculates the work done by constant force.

```
# this is a function that calculates the work done by the
constant force
dotp=function(F,S){  # F and S are force and displacement as
                     #the row vector of the same dimension
 W=F%*%t(S)               # this is the work done
 list(W=W) }
F=matrix(c(2,3,4),1,3)
F [,1] [,2] [,3]
[1,]    2    3    4
S=matrix(c(5,6,8),1,3)
S [,1] [,2] [,3]
[1,]    5    6    8
dotp(F,S)
$W
     [,1]
[1,]   60
```

Very good

Addition,Subtraction and Multiplication of Matrix in R

1. The sum and difference of two n-by-m matrices A and B, whose general elemnents are a_{ij} and b_{ij} is a new n-by-m matrix, say S, whose general elements is s_{ij}=$a_{ij} + b_{ij}$ or s_{ij}=$a_{ij} - b_{ij}$
   Example:A=matrix(c(2,3,4,5),nrow=2,ncol=2)
   B=matrix(c(6,7,8,9),nrow=2,ncol=2)

```
S=A+B   [,1] [,2]
    [1,] 8    12
    [2,] 10   14
S=A-B [,1] [,2]
   [1,] -4  -4
   [2,]-4   -4
```

2. An n-by-m matrix can be multiplied by an m-by-p matrix ,yielding an n-by-p matrix.

```
Example, Let A is 2-by-2 matrix
             B is 2-by-2 matrix
A%*%B=   [,1] [,2]
     [1,] 40   52
     [2,] 53   69
```

6 points

Linear Algebra

consider a set of m equations with n unknowns given by: $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = y_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = y_2$

... ... where a_{ij} are coefficients

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = y_m$ x_j are unknowns y_i are known terms

You should write this part properly

the above equation can be written as the product of matrix.

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & am2 & a_{m3} & \ldots & a_{mn} \end{pmatrix}$$

$$X=\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ \vdots \\ x_n \end{pmatrix}, Y=\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{pmatrix}$$

Therefore, the linear system can be written in matrix form. AX=Y

Linear equations expressed in matrix form may be classified into three cases:

- case1: m=n(the number of equations is equal to number unknowns)
- case 2 n(number of equations is less than the number of unknowns ,which is called an ***under-determined problems***)
- case 3 n(the number of equations is greater than the number of s ,which is called an ***over-determined problems)***

when m=n, the matrix is square matrix,therefor it is possible to find the solution matrix by R $AX = Y \Rightarrow X = A^{-1}Y$
Example, In the figure1 below the electric network connected to three terminals with known voltages . Find the voltage of at nodes a,b,and c.

Solution by R

use i_{ab} in latex

- the current i from a to b can be denoted by iab,and related to the voltage as
- $iab = \frac{Va-Vb}{Rab} where$
- Rab is resistance between a and b and
- Va and Vb are voltage.The total current leaving node a must be zero according to Kirchhoff's point rule.

- Hence$\sum_{j=b,c,\ldots k} i_{aj} = 0, \sum_{j=b,c,\ldots k} \frac{Va-Vj}{Raj} = 0$

Now it is possible to find the unknown values of the voltage at each node

- $\frac{Va-20}{2} + \frac{Va-Vb}{4} + \frac{Va-Vc}{3} = 0$ when i leaves point a
- $\frac{Vb-Va}{4} + \frac{Vb-0}{3} + \frac{Vb-Vc}{5} = 0$ when i leaves point b
- $\frac{Vc-5}{3} + \frac{Vc-va}{3} + \frac{Vc-Vb}{5} = 0$ when i leaves point c

these equations can be rewritten as

- $\frac{13Va}{12} - \frac{Vb}{4} - \frac{Vc}{3} = 10$
- $\frac{-Va}{4} + \frac{47Vb}{60} - \frac{Vc}{3} = 0$
- $\frac{-Va}{3} - \frac{Vb}{5} + \frac{13Vc}{15} = \frac{5}{3}$

Good program and good example

A=$\begin{pmatrix} 13/12 & -1/4 & -1/3 \\ -1/4 & 47/60 & -1/3 \\ -1/3 & -1/5 & 13/15 \end{pmatrix}$,X=$\begin{pmatrix} Va \\ Vb \\ Vc \end{pmatrix}$, Y=$\begin{pmatrix} 10 \\ 0 \\ 5/3 \end{pmatrix}$
it possible to find the solution of this equation by using the following function.

```
# this is the function that evaluates the solution of n equation
and n unknowns # which is written in the form of AX=Y
SOL=function(A,Y){  # where A square matrix and Y is column matrix
   X=solve(A)%*%Y # the solution of the linear equation
list(X=X) }
```

```
 A=matrix(c(13/12,-1/4,-1/3,-1/4,47/60,-1/5,-1/3,-1/3,13/15),3,3)
 A
           [,1]       [,2]       [,3]
[1,]  1.0833333 -0.2500000 -0.3333333
[2,] -0.2500000  0.7833333 -0.3333333
[3,] -0.3333333 -0.2000000  0.8666667
 Y=matrix(c(10,0,5/3),3,1)
 Y
          [,1]
[1,] 10.000000
[2,]  0.000000
[3,]  1.666667
 SOL(A,Y)
$X
          [,1]
[1,] 14.020938
[2,]  8.414114
[3,]  9.257464
```

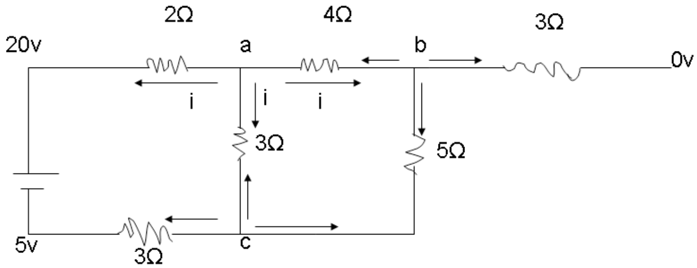


Figure 1: Write the figure caption in here. You can say for example, Simple circuit diagram

5. Unsolvable problems

2 points

A set of linear equations is not always numerically solvable.

1. If one equation is a multiple of another or can be obtained by adding or subtracting other equations, that equation is said to be linear dependant

```
    -x+y=1     the number of solutions of this equation is infinite.
    -2x+2y=2  Let us apply R to see what will happen the answer
 A=matrix(c(-1,-2,1,2),2,2)
 A
     [,1] [,2]
[1,]   -1    1
[2,]   -2    2
 Y=matrix(c(1,2),2,1)
 Y
     [,1]
[1,]    1
[2,]    2
 SOL(A,Y)
Error in solve.default(A) :
 Lapack routine dgesv: system is exactly singular
```

2. An inconsistent system of equations are unsolvable.
   A set of equations is inconsistent if the left side of at least one equation can be completely eliminated by adding or subtracting other equations, while the right hand still remains nonzero.

```
    -x+y=1    Let us see it by R
    -x+y=0

     A=matrix(c(-1,-1,1,1),2,2)
 A
     [,1] [,2]
[1,]   -1    1 [2,]   -1    1
 Y=matrix(c(1,0),2,1)
 Y
     [,1]
[1,]    1 [2,]    0
 SOL(A,Y)
Error in solve.default(A) : Lapack routine dgesv: system is
exactly singular
```

3. The three independent equations for two unknowns can never be simultaneously satisfied.Example,

```
   x+2y=-2
-x+y=1
 2x-y=0

      Let us see the solution by R
 A=matrix(c(1,-1,2,2,1,-1),3,2)
 A
     [,1] [,2]
[1,]    1    2
[2,]   -1    1
[3,]    2   -1
 Y=matrix(c(-2,1,0),3,1)
 Y
     [,1]
[1,]   -2
[2,]    1
[3,]    0
 SOL(A,Y)
Error in solve.default(A) : only square matrices can be inverted
```

2 points

Ill-conditioned problems

Definition: Ill-conditioned systems are those where small changes in coefficients result in large change in the solution . An alternative interpretation of ill-conditioning is that a wide range of answers can approximately satisfy the equations. The round-off errors can induce small changes in the coefficients , but these artificial changes can lead to large solution errors for ill-conditioned systems.

Example,
x+2y=10
1.1x+2y=10.4 the solution is
x=4 and y=3 However, the slight change of coefficients 1.1 to 1.05 the solution becomes x=8 and y=1 How do we determine whether a matrix is ill-conditioned or not? The slope of ill-conditioned equations are almost equal.

$a_{11}x + a_{12}y$=b_1 =>y=$b_1/a_{12} - (a_{11}/a_{12})x$
$a_{21}x + a_{22}y$=b_2 =>y=$b_2/a_{22} - (a_{21}/a_{22})x$
therefore, $\frac{a_{11}}{a_{12}} \approx \frac{a_{21}}{a_{22}} \Rightarrow a_{11}a_{22} \approx a_{12}a_{21}$
$\Rightarrow a_{11}a_{22} - a_{12}a_{21} \approx 0$
$a_{11}a_{22} - a_{12}a_{21}$ is the determinant of a two dimensional matrix
Therefore an ill-conditioned system is one with a ***determinant close to zero.*** In fact if the determinant is exactly zero the two slopes are identical,which connote either no solution or an infinite number of solutions in the case of ***unsolvable problems.***

Gauss elimination

6 points

Gauss-elimination consists of

- Forward elimination
  when m=n in the m equations and n unknowns, the forward elimination proceeds as follows. The first equation times a_{21}/a_{11} is subtracted from the second equation to eliminate the first term of the second equation.Likewise, the first term of every equation therefore, $i > 2$,is eliminated by subtracting the first equation times a_{i1}/a_{11}. Then the equation looks like

  $a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = y_1$

  $a'_{22}x_2 + a'_{23}x_3 + ... + a'_{2n}x_n = y'_2$

  ... ...

  $a'_{n2}x_2 + a'_{n3}x_3 + ... + a'_{nn}x_n = y'_n$

  where a'_{ij}=$a_{ij} - (a_{i1}/a_{11})a_{1j}$ y'=$y_i - (a_{i1}/a_{11})y_1$

Computational . . .
Content
Abstract
Introduction
Linear algebra
Interpolation and . . .
summey

Home Page
Title Page
Page 28 of 89
Go Back
Full Screen
Close
Quit

next

Nex,the leading term of every equation in the third through the last equation ,$i > 2$, is eliminated by subtracting the second equation times a'_{i2}/a'_{22}. After this step is completed , the leading terms of the fourth through the last equations are eliminated .When the forward elimination process is finished ,the set of the equations will be in the form

$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}=y_1$
$a'_{22}x_2 + ... + a'_{2n}x_n=y'_2$
. .. .
$a^{(n-1)}_{nn}x_n=y^{(n-1)}_n$

The leading term in each equation is called pivot.

- backward substitution The backward substitution procedure starts with the last equation. The solution of xn is obtained from the last equation by

  Please use x_{n} in latex

  $x_n = \frac{y_{n^{n-1}}}{a^{n-1}{}_{nn}}$
  $x_{n-1} = (y_{n-1} - a^{n-2}_{n-1} x_n)/a^{n-2}_{n-1,n-1}$
  $\ldots$
  $x_1 = (y_1 - \sum_{i=2}^{n} a_{1i} x_i)/a_{11}$
  So far we assumed an ideally simple situation that no pivot(diagonal coefficients),a_{ii},becomes zero. If any pivot becomes zero in the process of solution ,however, the forward elimination process cannot proceed.
  Pivoting is to exchange order of equations so that pivot coefficient a_{ii}becomes larger in the magnitude than any coefficients which are below in the same column and are to be eliminated.
  If absolute value of $a_{11} \geqslant$ absolute valuea_{i1}, $i > 1$,no pivoting is necessary .Otherwise the first equation is exchanged with the one with largest value. This process should be tested in each row .Then after it is possible to eliminate.

Example ,Solve the following equation by Gauss-elimination step by step by R

```
B=matrix(c(-0.04,0.56,-0.24,0.04,-1.56,1.24,0.12,0.32,-0.28),3,3)
 B
      [,1]  [,2]  [,3]
[1,] -0.04  0.04  0.12
[2,]  0.56 -1.56  0.32
[3,] -0.24  1.24 -0.28
 Y=matrix(c(3,1,0),3,1)
 Y
     [,1]
[1,]    3
[2,]    1
[3,]    0
 A=cbind(B,Y)
 A
      [,1]  [,2]  [,3] [,4]
[1,] -0.04  0.04  0.12    3
[2,]  0.56 -1.56  0.32    1
[3,] -0.24  1.24 -0.28    0

 T=A[2,];A[2,]=A[1,];A[1,]=T # row one and row two should be exchanged
 A
      [,1]  [,2]  [,3] [,4]
[1,]  0.56 -1.56  0.32    1
[2,] -0.04  0.04  0.12    3
[3,] -0.24  1.24 -0.28    0
```

```
 A[2,]=A[2,]-A[1,]*(A[2,1]/A[1,1]);A[3,]=A[3,]-A[1,]*(A[3,1]/A[1,1])
 A
     [,1]        [,2]       [,3]      [,4]
[1,] 0.56 -1.56000000  0.3200000 1.0000000
[2,] 0.00 -0.07142857  0.1428571 3.0714286
[3,] 0.00  0.57142857 -0.1428571 0.4285714
 T=A[3,];A[3,]=A[2,];A[2,]=T # row three and row two should be pivoted
 A
     [,1]        [,2]       [,3]      [,4]
[1,] 0.56 -1.56000000  0.3200000 1.0000000
[2,] 0.00  0.57142857 -0.1428571 0.4285714
[3,] 0.00 -0.07142857  0.1428571 3.0714286
 A[3,]=A[3,]-A[2,]*(A[3,2]/A[2,2]) # eliminating number below the 2nd pivot
 A
     [,1]       [,2]       [,3]      [,4]
[1,] 0.56 -1.5600000  0.3200000 1.0000000
[2,] 0.00  0.5714286 -0.1428571 0.4285714
[3,] 0.00  0.0000000  0.1250000 3.1250000
 X[3]=A[3,4]/A[3,3]   # backward substitution since forward elimination compl
 X[3]
[1] 25
X[2]=(A[2,4]-A[2,3]*X[3])/A[2,2]
X[2]
[1] 7
   X[1]=(A[1,4]-A[1,2:3]%*% X[2:3])/A[1,1]
  X[1]
  [1] 7
```

```
The above equation can be solved by a function below.
#This is a function that will perform Gaussian elimination on the
#square matrix  A and column vector b
GaussE=function(A,b){
n=nrow(A)
  # forward elimination
for(k in 1:(n-1)){  # go to column by column
for(i in (k+1):n){  # go to row by row
coeff=A[i,k]/A[k,k]
for(j in (k+1):n){
A[i,j]=A[i,j]-coeff*A[k,j]
}
A[i,k]=coeff
b[i]=b[i]-coeff*b[k]
}
}
     # backward substitution
x=matrix(,n,1)# empty column matrix to be replaced
x[n]=b[n]/A[n,n]# the bottom solution
for(i in (n-1):1){
sum=b[i]
for(j in (i+1):n){
sum=sum-A[i,j]%*%x[j]
x[i]=sum/A[i,i]
}
}
list(x=x)
}
```

Very good

```
 A=matrix(c(-0.04,0.56,-0.24,0.04,-1.56,1.24,0.12,0.32,-0.28),3,3)
 A
      [,1]  [,2]  [,3]
[1,] -0.04  0.04  0.12
[2,]  0.56 -1.56  0.32
[3,] -0.24  1.24 -0.28
 b=matrix(c(3,1,0),3,1)
 b
     [,1]
[1,]    3
[2,]    1
[3,]    0
 getwd()
[1] "C:/Program Files/R/R-2.4.0rc"
 getwd()
[1] "C:/Documents and Settings/Tigist/My Documents/Melessew/function"
 source("GaussE.R")
 GaussE(A,b)
$x
     [,1]
[1,]    7
[2,]    7
[3,]   25
```

5 points

please give space after comma and fullstop

Gauss-Jordan elimination

Gauss-Jordan elimination is a variation of Gauss elimination. The Gauss-Jordan elimination eliminates the number above and below a pivot without distinguishing the forward elimination and backward substitution separably.Pivoting is necessary ,however, for the same reason as far Gauss-elimination.Example ,

```
 B=matrix(c(-0.04,0.56,-0.24,0.04,-1.56,1.24,0.12,0.32,-0.28),3,3)
 B
      [,1]  [,2]  [,3]
[1,] -0.04  0.04  0.12
[2,]  0.56 -1.56  0.32
[3,] -0.24  1.24 -0.28
 Y=matrix(c(3,1,0),3,1)
 A=cbind(B,Y)
 A
      [,1]  [,2]  [,3] [,4]
[1,] -0.04  0.04  0.12    3
[2,]  0.56 -1.56  0.32    1
[3,] -0.24  1.24 -0.28    0
 T=A[2,];A[2,]=A[1,];A[1,]=T # the first and the second row should be exchang
 A
      [,1]  [,2]  [,3] [,4]
[1,]  0.56 -1.56  0.32    1
[2,] -0.04  0.04  0.12    3
[3,] -0.24  1.24 -0.28    0
```

```
 A[1,]=A[1,]/A[1,1]   # normalizing the first row
 A
      [,1]      [,2]       [,3]     [,4]
[1,]  1.00 -2.785714  0.5714286 1.785714
[2,] -0.04  0.040000  0.1200000 3.000000
[3,] -0.24  1.240000 -0.2800000 0.000000
 for(i in 2:3){
 + A[i,]=A[i,]-A[i,1]*A[1,]
 + }    #the elements below A[1,1] are
eliminated
 A
     [,1]        [,2]       [,3]      [,4]
[1,]    1 -2.78571429  0.5714286 1.7857143
[2,]    0 -0.07142857  0.1428571 3.0714286
[3,]    0  0.57142857 -0.1428571 0.4285714
 T=A[3,];A[3,]=A[2,];A[2,]=T  # pivoting is necessary
 A
     [,1]        [,2]       [,3]      [,4]
[1,]    1 -2.78571429  0.5714286 1.7857143
[2,]    0  0.57142857 -0.1428571 0.4285714
[3,]    0 -0.07142857  0.1428571 3.0714286
 A[2,]=A[2,]/A[2,2]  # normalizing the second pivot
 A
     [,1]        [,2]       [,3]     [,4]
[1,]    1 -2.78571429  0.5714286 1.785714
[2,]    0  1.00000000 -0.2500000 0.750000
[3,]    0 -0.07142857  0.1428571 3.071429
```

```
for(i in 1:3){# elements above and below second pivot are
                 #eliminated
+ if(i!=2){ + A[i,]=A[i,]-A[i,2]*A[2,] + } + }
A
     [,1] [,2]   [,3]  [,4]
[1,]    1    0 -0.125 3.875
[2,]    0    1 -0.250 0.750
[3,]    0    0  0.125 3.125
A[3,]=A[3,]/A[3,3]# normalizing third row
A
     [,1] [,2]   [,3]   [,4]
[1,]    1    0 -0.125  3.875
[2,]    0    1 -0.250  0.750
[3,]    0    0  1.000 25.000
for(i in 1:3){# elements above and below are eliminated
+ if(i!=3){
+ A[i,]=A[i,]-A[i,3]*A[3,]
+ }
+ }
A
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    7
[2,]    0    1    0    7
[3,]    0    0    1   25
```

the first three columns are an identity matrix,while the last column is the solution.

Matrix Inversion

Example,find the inverse of matrix B below. The Gauss-Jordan elimination may be used to find the inverse of a matrix. To do this , an augmented matrix is written ,in which the first three columns are the original matrix B, and the next three columns are the identity matrix.

```
 B=matrix(c(-0.04,0.56,-0.24,0.04,-1.56,1.24,0.12,0.32,-0.28),3,3)
 I=matrix(c(1,0,0,0,1,0,0,0,1),3,3)
 A=cbind(B,I)
 A
      [,1]  [,2]  [,3] [,4] [,5] [,6]
[1,] -0.04  0.04  0.12    1    0    0
[2,]  0.56 -1.56  0.32    0    1    0
[3,] -0.24  1.24 -0.28    0    0    1
 T=A[2,];A[2,]=A[1,];A[1,]=T  # pivoting is necessary
 A
      [,1]  [,2]  [,3] [,4] [,5] [,6]
[1,]  0.56 -1.56  0.32    0    1    0
[2,] -0.04  0.04  0.12    1    0    0
[3,] -0.24  1.24 -0.28    0    0    1
 A[1,]=A[1,]/A[1,1]  # normalizing the first row
 A
      [,1]      [,2]       [,3] [,4]     [,5] [,6]
[1,]  1.00 -2.785714  0.5714286    0 1.785714    0
[2,] -0.04  0.040000  0.1200000    1 0.000000    0
[3,] -0.24  1.240000 -0.2800000    0 0.000000    1
```

```
 for(i in 2:3){
+ A[i,]=A[i,]-A[i,1]*A[1,]
+ }
 A
     [,1]        [,2]       [,3] [,4]       [,5] [,6]
[1,]    1 -2.78571429  0.5714286    0 1.78571429    0
[2,]    0 -0.07142857  0.1428571    1 0.07142857    0
[3,]    0  0.57142857 -0.1428571    0 0.42857143    1
 T=A[3,];A[3,]=A[2,];A[2,]=T  # pivoting the second row by third row
 A
     [,1]        [,2]       [,3] [,4]       [,5] [,6]
[1,]    1 -2.78571429  0.5714286    0 1.78571429    0
[2,]    0  0.57142857 -0.1428571    0 0.42857143    1
[3,]    0 -0.07142857  0.1428571    1 0.07142857    0
 A[2,]=A[2,]/A[2,2]  # normalizing the second row
 A
     [,1]        [,2]       [,3] [,4]       [,5] [,6]
[1,]    1 -2.78571429  0.5714286    0 1.78571429 0.00
[2,]    0  1.00000000 -0.2500000    0 0.75000000 1.75
[3,]    0 -0.07142857  0.1428571    1 0.07142857 0.00
 for(i in 1:3){ #eliminating elements above and below second
                             #pivot
+ if(i!=2){
+ A[i,]=A[i,]-A[i,]*A[2,]
+ }
+ }
```

```
 A
     [,1] [,2]       [,3] [,4]       [,5] [,6]
[1,]    1    0  0.7142857    0 0.44642857 0.00
[2,]    0    1 -0.2500000    0 0.75000000 1.75
[3,]    0    0  0.1785714    1 0.01785714 0.00
 A[3,]=A[3,]/A[3,3] # normalizing the third row
 A
     [,1] [,2]       [,3] [,4]      [,5] [,6]
[1,]    1    0  0.7142857  0.0 0.4464286 0.00
[2,]    0    1 -0.2500000  0.0 0.7500000 1.75
[3,]    0    0  1.0000000  5.6 0.1000000 0.00
 for(i in 1:3){
+ if(i!=3){
+ A[i,]=A[i,]-A[i,3]*A[3,]
+ }
+ }
 A
     [,1] [,2] [,3] [,4]  [,5] [,6]
[1,]    1    0    0 -4.0 0.375 0.00
[2,]    0    1    0  1.4 0.775 1.75
[3,]    0    0    1  5.6 0.100 0.00
 B_inv=A[,4:6]
 B_inv
     [,1]  [,2] [,3]
[1,] -4.0 0.375 0.00
[2,]  1.4 0.775 1.75
[3,]  5.6 0.100 0.00
```

4 points

LU-Decomposition

The Lu-decomposition scheme transforms a matrix A to a product of two matrices.
A=LU where-L is the lower triangular matrix
U is the upper triagular matrix
with A=LU, the equation AX=Y is equivalently as LUX=Y which can be solved as LZ=Y; Z=UX

Solving equations using the LU decomposition is significantly more efficient than solving each equation individually by Gauss elimination when one has to solve a number of linear equation sets with the same nondifferentiation but different inhomogeneous (source terms). A matrix may be decomposed to L and U using Gauss elimination .The matrix after forward elimination may be regarded as transform of matrix A to U and the transformation is regarded equivalently by premultiplication of a matrix F which is obtained by the forward elimination of identity matrix in the augmented ma[illegible] what is this FT thing
FTherefore, $F = L^{-1}$. This implies that the lower triangular matrix is the inverse of F. Example. By LU decomposition , solve the linear equation

BX=Y with B= $\begin{pmatrix} 2 & 1 & -3 \\ -1 & 3 & 2 \\ 3 & 1 & -3 \end{pmatrix}$ Y= $\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$

Home Page
Title Page
Page 40 of 89
Go Back
Full Screen
Close
Quit

```
B=matrix(c(2,-1,3,1,3,1,-3,2,-3),3,3)
B
     [,1] [,2] [,3]
[1,]    2    1   -3
[2,]   -1    3    2
[3,]    3    1   -3
I=matrix(c(1,0,0,0,1,0,0,0,1),3,3)
I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
A=cbind(B,I)
A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    1   -3    1    0    0
[2,]   -1    3    2    0    1    0
[3,]    3    1   -3    0    0    1
A[2,]=A[2,]-A[1,]*(A[2,1]/A[1,1])
A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2  1.0 -3.0  1.0    0    0
[2,]    0  3.5  0.5  0.5    1    0
[3,]    3  1.0 -3.0  0.0    0    1
```

```
A[3,]=A[3,]-A[1,]*(A[3,1]/A[1,1])
A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2  1.0 -3.0  1.0    0    0
[2,]    0  3.5  0.5  0.5    1    0
[3,]    0 -0.5  1.5 -1.5    0    1
A[3,]=A[3,]-A[2,]*(A[3,2]/A[2,2])
A
     [,1] [,2]      [,3]      [,4]      [,5] [,6]
[1,]    2  1.0 -3.000000  1.000000 0.0000000    0
[2,]    0  3.5  0.500000  0.500000 1.0000000    0
[3,]    0  0.0  1.571429 -1.428571 0.1428571    1

F=A[,4:6]
F
          [,1]      [,2] [,3]
[1,]  1.000000 0.0000000    0
[2,]  0.500000 1.0000000    0
[3,] -1.428571 0.1428571    1
U=F%*%B
U
     [,1] [,2]      [,3]
[1,]    2  1.0 -3.000000
[2,]    0  3.5  0.500000
[3,]    0  0.0  1.571429
```

```
 L=solve(F)
 L
     [,1]       [,2] [,3]
[1,]  1.0  0.0000000    0
[2,] -0.5  1.0000000    0
[3,]  1.5 -0.1428571    1
 Y=matrix(c(2,0,1),3,1)
 Y
     [,1]
[1,]    2
[2,]    0
[3,]    1
 Z=solve(L)%*%Y
 Z
          [,1]
[1,]  2.000000
[2,]  1.000000
[3,] -1.857143
 X=solve(U)%*%Z
 X
           [,1]
[1,] -1.0000000
[2,]  0.4545455
[3,] -1.1818182
```

```
# This is the function that use the LU decomposition and
#transform the give  function into the upper and lower triangular
matrix LU=function(A,I){
 A=cbind(A,I) # the augmented matrix of A and the identity matrix
 n=nrow(A) # the number row of the augmented matrix
 m=ncol(A) # the number column of the augmented matrix
for(j in 2:n){
for(i in j:n){
C=A[i,(j-1)]/A[(j-1),(j-1)]# the ratio of the elements below the row of the f
A[i,]=A[i,]-A[(j-1),]*C # making zero the elements below the diagonal element
}
}
U=A[,1:n] # the upper triangular matrix of the given matrix
 F=A[,(n+1):m]# the matrix times by the identity matrix that is itself
 L=solve(F)# the lower triangular matrix of the given matrix
 B=L%*%U
list(U=U,L=L,B=B)
 }
  source("LU.R")
 A=matrix(c(2,-1,3,1,3,1,-3,2,-3),3,3)
 A
     [,1] [,2] [,3]
[1,]    2    1   -3
[2,]   -1    3    2
[3,]    3    1   -3
```

Excellent work!!

```
 I=matrix(c(1,0,0,0,1,0,0,0,1),3,3)
 I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
 LU(A,I)
$U
     [,1] [,2]      [,3]
[1,]    2  1.0 -3.000000
[2,]    0  3.5  0.500000
[3,]    0  0.0  1.571429

$L
     [,1]       [,2] [,3]
[1,]  1.0  0.0000000    0
[2,] -0.5  1.0000000    0
[3,]  1.5 -0.1428571    1

$B
     [,1] [,2] [,3]
[1,]    2    1   -3
[2,]   -1    3    2
[3,]    3    1   -3
```

4 points

Matrix Eigenvalue

A non-zero vector X is called a characterstic vector of a matrix A, if there is a number λ such that AX=λ X.where λ is called a characteristic roots of A.The characteristic roots are often known as ***Eigenvalue***. AX=λX=λIX,I being a unit matrix $\Rightarrow$ (A-λI)X=0, since $X \neq 0$,the matrix (A-λI) is singular so that det(A-λI)=0 which follows that every characteristic root of λ of a matrix A is a root of its characteristic equation det(A-λI)=0.

Every root of the charactrstic equation of a matrix is a characteristic roots of the matrix.Thus if $A = [a_{ij}]$ be an n-rowed square matrix and λ an indeterminate,then the characteristic equation det(A-λI)=0 gives

det $\begin{pmatrix} a_{11}-\lambda & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22}-\lambda & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33}-\lambda & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a^{n1} & a_{n2} & a_{n3} & \dots & a_{nn}-\lambda \end{pmatrix} = 0$ which is an ordiary polynomial in λ of degree n and hence will give n values of λ on simplification.These n values of λ are n eigenvalues of this equation.

The polynomial function can be defined by f(λ)=det(A-λI) is called the characteristic polynomial of matric A.Example,find the eigenvalue of the matrixA=$\begin{pmatrix} 1 & 3 \\ -1 & 2 \end{pmatrix}$

solution f(λ)=det($\begin{pmatrix} 1-\lambda & 3 \\ -1 & 2-\lambda \end{pmatrix} = \lambda^2 - 3\lambda + 5$the solution f($\lambda$)=0 are called characteristic value and are the same as eigenvalue of the matrix A.λ=(3+$\sqrt{11}$i)/2 or λ=(3-$\sqrt{11}$i)/2

```
 # this is function that calculates the real eigenvalue of 2 by 2 matrix
EIG=function(A){
  E1=((A[1,1]+A[2,1])-sqrt((A[1,1]+A[2,1])^2-4*(A[1,1]*A[2,1]-A[2,1]*A[1,2]))
  E2=((A[1,1]+A[2,1])+sqrt((A[1,1]+A[2,1])^2-4*(A[1,1]*A[2,1]-A[2,1]*A[1,2]))
 list(E1=E1,E2=E2)
}
```

Example.Consider a system consisting of masses and springs as shown below in figure2 . The equation of the net force are given by

$m_1\frac{d^2}{dt^2}(y_1(t)) = -(k_1+k_2)+k_2y_2$

$m_2\frac{d^2}{dt^2}(y_2(t)) = k_2y_1-k_2y_2$

solution Given $K_1 = 0.3N/m$,$k_2 = 0.1N/m$,$m_1 = 0.1kg$,$m_2 = 0.2kg$

For harmonic oscillation the solution may be written as $y_k(t)$=$e^{2\Pi\lambda it}f_k$,k=1,2

where λ is the frequency and i =$\sqrt{-1}$

by using the above equations -γf_1=-$(\frac{k_1+k_2}{m_1})f_1+\frac{k2}{m1}f_2$

-γf_2=$\frac{k_2}{m_2}f_1-\frac{k_2}{m_2}f_2$where$\gamma$=$(2\Pi\lambda)^2$

these two equations can be written in matrix form Af=γf

A=$\begin{pmatrix} \frac{k_1+k_2}{m_1} & -\frac{k_2}{m_1} \\ \frac{-k_2}{m_2} & \frac{k_2}{m_2} \end{pmatrix}$, f=$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$

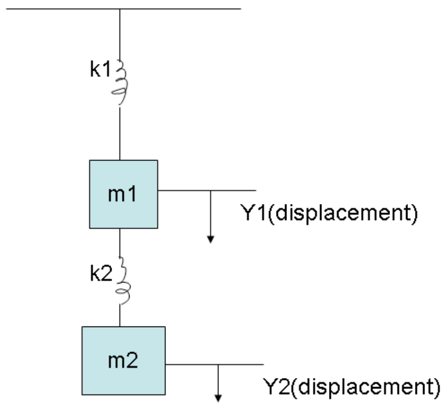


Figure 2: mass-spring system

```
 A=matrix(c(4,-0.5,-1,0.5),2,2)
 A
     [,1] [,2]
[1,]  4.0 -1.0
[2,] -0.5  0.5
 EIG(A)
$E1
[1] -0.6084953

$E2
[1] 4.108495
```

9 points

Iterative solution

As opposed to the direct method of solving a set of linear equations by elimination is the iterative methods.It is prefered over the direct method when

-the number of unknowns is very large but the coefficient matrix is sparce(has many zeros)

Please use fullstop and comma to separate things

-the equestions are not lin[illegible] an initial guess for iteration to get starte[illegible].If a good guess is available, convergence of the iterative solution becomes fast;however,if not the initial guess may be set to a null vector.There are three iterative methods

space

1.Jacobie iterative method is written as $x_i^{(t)}=(y_i-\sum_{j=1,j\neq i}^{n} a_{ij}$ [illegible] here superscript t is the iteration count. Consider a linear equatio[illegible] where A is a squere matrix,X is an unknown and Y is the source term.Example.

$6x_1 - 2x_2 + x_3$=11

$x_1 + 2x_2 - 5x_3$=-1

$-2x_1 + 7x_2 + 2x_3$=5 The true solution is x_1=2,x_2=1,x_3=1

to see the iteration scheme, solve each equation for one of the vaeriables,

x_1=$1.8333 + 0.3333x_2 - 0.1667x_3$

x_2=$0.7143 + 0.2857x_1 - 0.2857x_3$

x_3=$0.2 + 0.2x_1 + 0.4x_2$ Let the initial guess be zero,substituting this approximations into the right hand sides of the set of equations generate new approximations that we hope , are close to the true value. The new values are substituted in the right hand sides to generate a second approximation , and the process is repeated untill successive values of each of the variable are sufficiently alike.

x_1^{n+1}=$1.8333 + 0.3333x_2^n - 0.1667x_3^n$
$x_2^{(n+1)}$=$0.7143 + 0.2855x_1^n - 0.2857x_3^n$
$x_3^{(n+1)}$=$0.2 + 0.2x_1^n + 0.4x_2^n$ Let the initial guess vector be $x^{(0)}$=(0,0,0)

```
successive estimates of solution(Jacobi method)
   1st      2nd       3rd       4th      5th        6th ...  9th
 x1  0        1.833   2.038       2.085   2.004    1.994 ...  2
 x2 0       0.714     1.181      1.053    1.001    0.999...  1
 x3 0       0.2       0.852      1.08     1.038     1.008... 1
```

```
 # this is the program that iterates n times the given system of equestion by
 # Jacobie method
 X=c(0,0,0)# the initial guess
 x=c(12,20,30)# tentative new solution
 for(i in 1:9){ # number of iteration
+ x[1]=1.8333+0.3333*X[2]-0.1667*X[3] +
x[2]=0.7143+0.2857*X[1]-0.2857*X[3] + x[3]=0.2+0.2*X[1]+0.4*X[2] +
X=x + }
 x
[1] 2.000022 1.000035 1.000244
```

Example 2,

$12x_1 + 3x_2 - 5x_3$=1

$x_1 + 5x_2 + 3x_3$=28

$3x_1 + 7x_2 + 13x_3$=76

```
 # this is the program that iterates n times the given system of question
 #  by Jacobie method
 X=c(1,0,1)# the initial guess
 x=c(12,20,30)# tentative new solution
 for(i in 1:6){ # number of iteration
+ x[1]=(1/12)-3*(X[2]/12)+5*(X[3]/12)
+ x[2]=(28/5)-X[1]/5-3*(X[3]/5)
+ x[3]=(76/13)-3*(X[1]/13)-7*(X[2]/13)
+ X=x
+ }
 x
[1] 1.014267 2.943251 3.943382
```

Computational...
Content
Abstract
Introduction
Linear algebra
Interpolation and . . .
summey

2.The Gauss-Seidel method

Its advantage over the Jacobie method is speed of evaluating the problem because whenever a new value is obtained it can be written over the old values unlike the Jacobie method. And it is given by $x_i^{(t)}=(y_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(t)} - \sum_{j=i+1}^{n} a_{ij}x_j^{t-1})/a_{ii}$ The above example solved as follow by the Gauss-Seidel method

	1st	2nd	3rd	4th	5th	6th
x1	0	1.833	2.069	1.998	1.999	2
x2	0	1.238	1.002	0.995	1.00	1
x3	0	1.062	1.015	0.998	0.998	1

```
 # this is the program that iterates n times the given system of question
# by Gauss-Seidel method
X=c(0,0,0)# the initial guess
x=c(12,20,30)# tentative new solution
for(i in 1:9){ # number of iteration
+ x[1]=1.8333+0.3333*X[2]-0.1667*X[3]
+x[2]=0.7143+0.2857*x[1]-0.2857*X[3]
+ x[3]=0.2+0.2*x[1]+0.4*x[2]
X=x
+ }
 x
[1] 1.9998979 0.9999790 0.9999712
```

Home Page
Title Page
Page 53 of 89
Go Back
Full Screen
Close
Quit

```
 The above example 2 by Gauss-Seidel method
  # this is the program that iterates n times the given system of question
 # by Gauss-Seidel method
 X=c(1,0,1)# the initial guess
 x=c(12,20,30)# tentative new solution
 for(i in 1:6){ # number of iteration
+ x[1]=(1/12)-3*(X[2]/12)+5*(X[3]/12)
+x[2]=(28/5)-x[1]/5-3*(X[3]/5)
+x[3]=(76/13)-3*(x[1]/13)-7*(x[2]/13)
+ X=x
+ }
 x
[1] 0.9991948 3.0001089 4.0001272
```

3.The successive -over- relaxation (SOR) is further improvement of the Gauss-seidal scheme, and is written as $x_i^t = w(y_i - \sum_{j=1}^{i-1} a_{ij}x_j^t - \sum_{j=i+1}^{n} a_{ij}x_j^{t-1})/a_{ii} + (1-w)x_i^{t-1}$ where w is an over-relaxation parameter satisfying $1 \leqslant w \leqslant 2$ As a rule of thumb, w may be set to a value between 1.2 and 1.7[6]

6. Interpolation and curve fitting

In numerous applications areas ,we may faced with the task of describing data, often measured ,with ***analytical function.*** There are two approaches to solve this problem.

4 points

- Interpolation
  Interpolation is a way of estimating values of the function between those given by some set of data points.Interpolation is the valuable tool when one cannot quickly evaluate the function at the desired intermediate points.Example, when the data points are the results of some experimental measuremnts.The strategy used in approximating the unknown value of the function is ***finding a polynomial that fits a selected set of points***(x,f(x)) and assume that the polynomial and the unknown function behave nearly the same over the interval in equations.

  1. Linear interpolation
     The linear interpolation is a line fitted to two data points and is given by:g(x)=$\frac{f(b)-f(a)}{b-a}(x-a)+f(a)$ where f(a) and f(b) are the known values of f(x) at x=a and x=b respectively.

```
 # This is the function that determines the functional values for a
 #specified abscissa denoted by xi by linear interpolation from
 #function table.In LIP,linear interpolation is applied to each
 #data interval.
 # x is a 4 by 1 column array
  # y is a 4 by 1column array
# xi is a 3 by 1 column array of x values for which
 #yi,i=1,2,3 values are to be evaluated by linear interpolation.
LIP=function(x,y,xi){
   y1=y[1,1]+((y[2,1]-y[1,1])/(x[2,1]-x[1,1]))*(xi[1,1]-x[1,1])
   y2=y[2,1]+((y[3,1]-y[2,1])/(x[3,1]-x[2,1]))*(xi[2,1]-x[2,1])
   y3=y[3,1]+((y[4,1]-y[3,1])/(x[4,1]-x[3,1]))*(xi[3,1]-x[3,1])
 list(y1=y1,y2=y2,y3=y3)
}
```

Example, The following data are the observed lake Tana water level(meter) in 2001 for 7 consecutive days.

Date	2001
1/Jan	1786.90
2/Jan	1786.92
3/Jan	1786.92
4/Jan	1786.91
5/Jan	1786.91
6/Jan	1786.89
7/Jan	1786.90

were not

Now, assume that the measuremen[illegible]n on 2/Jan,4/Jan and 6/Jan due to the unknown problem.If we need the missed data ,it impossible to find them by using the above function.

```
 x=matrix(c(1,3,5,7),4,1)# 1/Jan,3/Jan,5/Jan,7/Jan
 x
     [,1]
[1,]    1
[2,]    3
[3,]    5
[4,]    7
 y=matrix(c(1786.90,1786.92,1786.91,1786.90),4,1) # observed lake T
 y   [,1]
[1,] 1786.90
[2,] 1786.92
[3,] 1786.91
[4,] 1786.90
 xi=matrix(c(2,4,6),3,1) # missed dates
 xi   [,1]
[1,]    2
[2,]    4
[3,]    6
 LIP(x,y,xi)
$y1
[1] 1786.91

$y2
[1] 1786.915

$y3 [1] 1786.905
```

```
 The error during approximating in the
above problem can be found by the following function
 # This is the function that calculate
 # the relative error  during interpolation
 # TV is the true value of the data
# AV is the approximation value
 # RE is the relative error value in percentile
Err=function(TV,AV){
  RE=abs((TV-AV)/TV*100)
list(RE=RE) }
 TV=1786.92;AV=1786.91
 Err(TV,AV)
$RE
[1] 0.0005596221

 TV=1786.91;AV=1786.915
 Err(TV,AV)
$RE
[1] 0.0002798126
 TV=1786.89;AV=1786.905
 Err(TV,AV)
$RE
[1] 0.0008394473
```

Linear extrapolation is the process of estimating a value of the function f(x) that lies outside the range of the known base points, x(i),i=0,1,2...,n.

6 points

2. Lagrange Interpolation Polynomial
When the data are not "smooth" mea
irregularities ,there are problems with interpolating polynomials. To solve such problems, higher degree polynomial are necessary to follow such irregularities.
Lagrange polynomial is the simplest way to exhibit the existence of a polynomial for interpolation with unevenly spaced data. The data where the x-values are not equi-spaced often occurs as the result of the experimental observation.
Consider a polynomial interpolation formula that passes via the data points.

please write it correctly

$x_1 x_2 \ldots x_{n+1}$
$y_1 y_2 \ldots y_{n+1}$

To introduce the basic principle of the Lagrange formula,consider the product of factors $u_1 = (x - x_2)(x - x_3)\ldots(x - x_{n+1})$

Home Page
Title Page
Page 59 of 89
Go Back
Full Screen
Close
Quit

This function is an n-th order polynomial of x and becomes zero at x=$x_2, x_3, ..., x_{n+1}$ but not zero for x=x_1. If we divide u_1(x) by $u_1(x_1)$, the resulting function is $v_1(x)$=$\frac{u_1(x)}{u_1(x_1)}$=$\frac{(x-x_2)(x-x_3)...(x-x_{n+1})}{(x_1-x_2)(x_1-x_3)...(x_1-x_{n+1}}$ satisfies $v_1(x_1) = 1 and v_1(x_i)$=0 for i=2,3,...,n+1 In general terms ,we can write $v_i$by $v_1(x)$=$\frac{u_i(x)}{u_i(x_i)}$=$\prod_{j=1,j\neq i}^{n+1}(\frac{x-x_j)}{(x_{i-x_j})}$ The function $v_i(x)$ is an n^{th}order polynomial which satisfies $v_i(x_i)$=1 and $v_i(x_j)$=0, for all $j \neq i$ we call $v_i(x)$ a coefficient polinomial.If we multiply $v_1(x), v_2(x), ..., v_{n+1}$by $y_1, y_2, ..., y_{n+1}$respectively and add them together ,the summation becomes a polynomial of order n again. Hence the Lagrange interpolation formula of order n is written by g(x)=$\sum_{i=1}^{n+1} v_i(x) y_i$

```
# This is the function that interpolate  data by Lagrange interpola
#x,y:are data points 4 by 1 column array
 # xi;  array of abscissas for which y values are to be
 #computed
  # yi: array of y values computed by Lagrange interpolation.

LG=function(x,y,xi){
 yi=((((xi[,1]-x[2,1])*(xi[,1]-x[3,1])*(xi[,1]-x[4,1]))/
 ((x[1,1]-x[2,1])*(x[1,1]-x[3,1])*(x[1,1]-x[4,1])))*y[1,1]
 +(((xi[,1]-x[1,1])*(xi[,1]-x[3,1])*(xi[,1]-x[4,1]))/
 ((x[2,1]-x[1,1])*(x[2,1]-x[3,1])*(x[2,1]-x[4,1])))*y[2,1]
 +(((xi[,1]-x[1,1])*(xi[,1]-x[2,1])*(xi[,1]-x[4,1]))/
 ((x[3,1]-x[1,1])*(x[3,1]-x[2,1])*(x[3,1]-x[4,1])))*y[3,1]
 +(((xi[,1]-x[1,1])*(xi[,1]-x[2,1])*(xi[,1]-x[3,1]))/
 ((x[4,1]-x[1,1])*(x[4,1]-x[2,1])*(x[4,1]-x[3,1])))*y[4,1])
   list(yi=yi)
}
```

Example,Take the observed lake Tana water level(meter) in 2001 on the given days below.

Date	level(meter)
1/Jan	1786.90
2/Jan	1786.92
3/Jan	1786.92
4/Jan	1786.91
5/Jan	1786.91
6/Jan	1786.89
7/Jan	1786.90

Assume that the data on 2/Jan,4/Jan and 6/Jan did not observed due to unknown problem but now the data are needed.
Find the left data by using Lagrange interpolation and compare the result with the linear interpolation.

```
x=matrix(c(1,3,5,7),4,1)
x
     [,1]
[1,]    1
[2,]    3
[3,]    5
[4,]    7
```

```
 y=matrix(c(1786.90,1786.92,1786.91,1786.89),4,1)
 y
        [,1]
[1,] 1786.90
[2,] 1786.92
[3,] 1786.91
[4,] 1786.89
 xi=matrix(c(2,4,6),3,1)
 xi [,1]
[1,]    2
[2,]    4
[3,]    6
 LG(x,y,xi)
$yi [1] 1786.915 1786.918 1786.900
  The relative error in Lagrange interpolation
 TV=1786.92;AV=1786.915
 Err(TV,AV)
$RE
[1] 0.0002798111
 TV=1786.91;AV=1786.918
 Err(TV,AV)
$RE
[1] 0.0004477002

 TV=1786.89;AV=1786.90
 Err(TV,AV)
$RE [1] 0.0005596315
```

6 points

The relative … than in linear interpolati… pproximates better than …

3. Spline interpolation The n-th order polynomial can be used to interpolate between n+1 data points. However, there are cases where these functions can lead to erroneous(mistake)results because of round-off error and overshoot(deviating widely). An alternative approach is to apply lower order polynomials to subsets of data points. Such connecting polynomials are called **spline functions.**
For example, the third ordered curves employed to connect each pair of data points are called ***cubic splines***. This functions can be constructed so that the connection between adjacent cubic equations are visually smooth. The objective in cubic splines is to derive a third order polynomial for each interval between knots, as in $f_i(x)$=$a_ix^3 + b_ix^2 + c_ix + d_i$....1 Thus, for n+1 data points (i=0,1,2,...,n), there are n intervals and ,consequently, 4n unknowns constants to evaluate.In order to fit a cubic spline, function $f_i(x)$ to data points (x_i, f_i), the following quantities are involved.
x_i, i=1,2,3,...,n
f_i,i=1,2,3,...,n
f_i'i=1,2,3,...,n
f_i'',i=1,2,3,...,n The function becomes straight line at the end points ,therefore the second derivative at the end point becomes zero such condition leads to what is termed a ”natural ” spline.

Please write it properly using proper latex commands

By performing different evaluations we can get the following cubic equations: $f_i(x)=\frac{f_i^{''}(x_{i-1})(x_i-x)^3}{6(x_i-x_{i-1})}+\frac{f_i^{''}(x_i(x-x_{i-1})^3}{6(x_i-x_{i-1})}+(\frac{f(x_{i-1})}{(x_i-x_{i-1})}$ - $\frac{f^{''}(x_{i-1})(x_i-x_{i-1}}{6})(x_i-x)$ +$(\frac{f(x_i)}{(x_i-x_{i-1})}-\frac{f^{''}(x_i)(x_i-x_{i-1}}{6})(x-x_{i-1})$....2

Note, this equation contains only two unknown coefficient, the second derivative at the beginning and end of the interval - $f^{''}(x_{i-1})$ and $f^{''}(x_i)$ These unknowns can be determined by the following equation.

$(x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + (x_{i+1} - x_i)f''(x_i) = (\frac{6}{x_{i+1}-x_i})(f(x_{i+1}) - f(x_i)) + (\frac{6}{x_i-x_{i-1}})(f(x_{i-1}) - f(x_i))$...3 After we determine these unknowns equations 2,is the third order polynomial that can be used to interpolate with in the intervals. Example, Fit cubic splines to the following data and utilize the results to estimate the value at x=4,5,8. (x_o=3, $f(x_o)$=2.5), (x_1=4.5 , $f(x_1)$=1), (x_2=7 ,$f(x_2)$=2.5), (x_3=9 , $f(x_3)$=0.5) Solution : First determine the unknown second derivatives at the knots by using equation(3) Because of the natural spline condition, $f''(3)$=0 Hence, for the first interval $8f''(4.5) + 2.5f''(7)$=9.6 By similar fashion equation (3) can be applied to the second interval,hence $2.5f''(4.5) + 9f''(7)$=-9.6 These two equations can be solved simultaneously for $f''(4.5)$=1.679009 $f''(7)$=-1.53308 and these values can be substituted into equation(2) along with values for $x's$ and $f(x)'s$ to yield: at x=4,f(4)=1.3;x=5,f(5)=1.002;x=8,f(8)=1.7. In R it is possible to interpolate by using the function "cubic".

```
# This is the function that performs a cubic spline interpolation
    # by using four data points
 # x and y are data points in column vector form
# xi is a column vector of abscissa for which the
interpolation is performed
# yi is the interpolated y value for xi
cubic=function(x,y,xi){
   a=2*(x[3,1]-x[1,1]) # the coefficients of the second derivative
   b=(x[3,1]-x[2,1])
   c=(x[3,1]-x[2,1])
  d=2*(x[4,1]-x[2,1])
 A=matrix(c(a,c,b,d),2,2)# coefficient matrix of the second derivat
  s=(6/(x[3,1]-x[2,1]))*(y[3,1]-y[2,1])
  r=(6/(x[2,1]-x[1,1]))*(y[1,1]-y[2,1]);m=s+r
 t=(6/(x[4,1]-x[3,1]))*(y[4,1]-y[3,1])
 g=(6/(x[3,1]-x[2,1]))*(y[2,1]-y[3,1])
 n=t+g ;z=matrix(c(m,n),2,1)
 X=solve(A)%*%z # the value of the second derivative
f=X[1,1]  # the value of the second derivative at x[2,1]
ff=X[2,1]# the value of the second derivative at x[3,1]
y1=((f/6*(x[2,1]-x[1,1]))*(xi[1,1]-x[1,1])^3+
      (y[1,1]/(x[2,1]-x[1,1]))*(x[2,1]-xi[1,1])
    +(y[2,1]/(x[2,1]-x[1,1])-(f*(x[2,1]-x[1,1])/6))*(xi[1,1]-x[1,1]
y2=((f*(x[3,1]-xi[2,1])^3)/(6*(x[3,1]-x[2,1])))+
   ((ff*(xi[2,1]-x[2,1])^3)/(6*(x[3,1]-x[2,1])))
   +(y[2,1]/(x[3,1]-x[2,1])-(f/6)*(x[3,1]-x[2,1]))*(x[3,1]-xi[2,1])
 +((y[3,1]/(x[3,1]-x[2,1]))-((ff/6)*
(x[3,1]-x[2,1])))*(xi[2,1]-x[1,1])
```

```
y3=((ff*((x[4,1]-xi[3,1])^3/6))*(x[4,1]-x[3,1])
 +((y[3,1]/(x[4,1]-x[3,1])-(ff*((x[4,1]-x[3,1])/6)))*(x[4,1]-xi[3,1
   +(y[4,1]/(x[4,1]-x[3,1]))*(xi[3,1]-x[3,1]))
list(y1=y1,y2=y2,y3=y3)}
 x=matrix(c(3,4.5,7,9),4,1)
 x  [,1]
[1,]  3.0
[2,]  4.5
[3,]  7.0
[4,]  9.0
 y=matrix(c(2.5,1,2.5,0.5),4,1)
 y [,1]
[1,]  2.5
[2,]  1.0
[3,]  2.5
[4,]  0.5
 xi=matrix(c(4,5,8),3,1)
 xi [,1]
[1,]    4
[2,]    5
[3,]    8
 cubic(x,y,xi)
$y1
[1] 1.5
$y2
[1] 0.8827376
$y3
[1] 1.5
```

Curve fitting means to fit a function g(x) to a set of give data ,(x_i, y_i),i=1,2,3…,L
The function g(x) can be a ***polynomial*** , ***non linear function,*** or a ***linear combination of known functions.*** The function g(x) selected for curve fitting must have a certain number of undetermined coefficients. In general ,the number of data points to be fitted , L, is much greater than the number of undetermined coefficients, k. Therefor, the principle of determining the coefficients is based on minimization of the discrepancies between the determined function and the data points and named the ***least square method.***

1.Line fitting

Suppose we desire to fit a linear function to the data set, as shown below.

i	xi	yi
1	0.1	0.61
2	0.4	0.92
3	0.5	0.99
4	0.7	1.52
5	0.7	1.47
6	0.9	2.03

The line fitted to a data set is called a regression line.The linear function is expressed by g(x)=c1x+c2 where c1 and c2 are undetermined constants.Since the number of data points is greater than two, the line can not be fitted to every point ,but is determined by minimizing the discrepancies between the line and data.

Home Page
Title Page

Go Back
Full Screen
Close
Quit

Definition: The deviation of the line from data points is given by r_i=$y_i - g(x_i)$=$y_i - (c1x_i + c2)$, i=1,2,...,L
The sum of the squared devaition is R=$\sum_{i=1}^{L} r_i^2$=$\sum_{i=1}^{L}(y_i - c1x_i - c2)$
The minimum value of the deviation occurs if partial deviations of R with respect to c1 and c2 becomes zerc

$\partial R \partial c1$=$-2\sum_{i=1}^{L} x_i(y_i - c1x_i - c2)$=0
$\partial R \partial c2$=$-2\sum_{i=1}^{L}(y_i - c1x_i - c2)$=0
Let a11=$\sum_{i=1}^{L} x_i^2$;a12=a21=$\sum_{i=1}^{L} x_i$;a22=
z1=$\sum_{i=1}^{L} x_i y_j$;z2=$\sum_{i=1}^{L} y_i$

please correct it using proper latex command

Hence we can express this equation in matrix form as A=$\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix}$, c=$\begin{pmatrix} c1 \\ c2 \end{pmatrix}$,y=$\begin{pmatrix} z1 \\ z2 \end{pmatrix}$ To determine the coefficients consider the equation as an over-determined linear equations (because the number of data points are greater than the coefficients - the number of equations are greater than the coefficients). The coefficients by hand can be found by c1=$(a_{22}z_1 - a_{12}z_2)/(a_{11}a_{22} - a_{12}a_{21})$ c2=$(a_{11}z_2 - a_{21}z_1)/(a_{11}a_{22} - a_{12}a_{21})$ In R it is possible to find the coefficients by the function"Lift".

```
# this is a function determines the coefficients of line fitted to
the data points.
 # xi and yi are column vector data set
Lift=function(xi,yi){
 a12= a21=sum(xi)
  a22=length(xi)
  a11=sum(xi*xi)
  z1=sum(xi*yi)
  z2=sum(yi)
A=matrix(c(a11,a21,a12,a22),2,2) z=matrix(c(z1,z2),2,1)
c=solve(A)%*%z   # the coefficients of the line fitted
list(c=c) }
 Example 1. Determine the regression line for the data
above.
 xi=matrix(c(0.1,0.4,0.5,0.7,0.7,0.9),6,1)
 xi [,1]
[1,]  0.1
[2,]  0.4
[3,]  0.5
[4,]  0.7
[5,]  0.7
[6,]  0.9
```

```
 yi=matrix(c(0.61,0.92,0.99,1.52,1.47,2.03),6,1)
 yi
     [,1]
[1,] 0.61
[2,] 0.92
[3,] 0.99
[4,] 1.52
[5,] 1.47
[6,] 2.03
 Lift(xi,yi)
$c      [,1]
[1,] 1.7645570
[2,] 0.2861603
 Therefor,g(x)=1.764557x+0.28616
```

Computational . . .
Content
Abstract
Introduction
Linear algebra
Interpolation and . . .
summey
Home Page
Title Page
Page 72 of 89
Go Back
Full Screen
Close
Quit

Example 2. Let us fit the line to wind speed measured on 1/1/2006 in each minute.And then find the speed of the wind at 10th minute by the fited line.

```
source("Lift.R")
 Lift(xi,yi)  #xi is the minute as the column vector and yi is the
                # wind speed in each minute
 $c
              [,1]    # the coefficient of the line fitted
[1,] -0.0005906586
[2,]  3.7310073546
 t=10
 g=-0.0005906586*t+3.7310073546 # the equation of the line
 g
[1] 3.725101  # the speed of the wind at the 10th minite
  Let as find the relative error when Av(approximation
  value)=3.725101m/s and TV(true value)=4m/s
 source("Err.R")
 TV=4;AV=3.725101
 Err(TV,AV)
$RE [1] 6.872475 # the relative error
```

Non-linear curve fitting with a power function

For some data types , fitting the power function given by g(x)=βx^{α}, may be suitable where α and β are undetermined coefficients. To determine the coefficients, we have to take the logarithm of the data set.

log(g(x))=αlog(x)+log(β) Let G=log(g(x));c1=α
c2=log(β)$\Rightarrow$ β=exp(c2) ;X=log(x) This becomes G=c1X +c2 hence the problem is reduced to line regression.This equation is fitted to the data set($log(x_i)$,$log(y_i)$) In R , non- linear curve fitting with power function can be done by "Nlcf" function.

```
# This is the function fitting the non-linear curve on the given
data # x and y are column vector data points Nlcf=function(x,y){
   xi=log(x)
   yi=log(y)
 a12= a21=sum(xi)
  a22=length(xi)
  a11=sum(xi*xi)
  z1=sum(xi*yi)
  z2=sum(yi)
A=matrix(c(a11,a21,a12,a22),2,2)
z=matrix(c(z1,z2),2,1)
c=solve(A)%*%z
B=exp(c[2,1]) # the value of beta
t=c[1,1] # the value of alpha
list(B=B,t=t) }
```

```
 x=matrix(c(0.15,0.4,0.6,1.01,1.5),5,1)
 x
     [,1]
[1,] 0.15
[2,] 0.40
[3,] 0.60
[4,] 1.01
[5,] 1.50
 y=matrix(c(4.4964,5.1284,5.6931,6.2884,7.5507),5,1)
 y
       [,1]
[1,] 4.4964
[2,] 5.1284
[3,] 5.6931
[4,] 6.2884
[5,] 7.5507
 Nlcf(x,y)
$B
[1] 6.506493

$t
 [1] 0.2153211
```

Therefor,g(x)=$6.506493x^{0.2153211}$

Curve fitting with a higher-order polynomial

Example. curve fitting with quadratic polinomial

suppose we desire to fit a quadratic functio to the data set. The quadratic equation can be expressed as g(x)=$c_1x^2 + c_2x + c_3$ where c_1,$c_2$and $c_3$are undetermined constant. Since the number of data points is greater than three, the quadratic function can not be fitted to every point,but it is possible by minimizing the discrepancies between the function and data. The deviation of the line from the data points is defined by r_i=$y_i - g(x_i)$=$y_i - (c_1x_i^2 + c_2x_i + c_3)$,i=1,2,3,...,L where L is the total number of data points The sum of the squared deviations is R=$\sum_{i=1}^{L} r_i^2$=$\sum_{i=1}^{L}(y_i - c_1x_i^2 - c_2x_i - c_3)$ The minimum of R occurs if partial derivatives of R with respect to c_1,c_2 and $c_3$becomes zero.

$\frac{\partial R}{\partial c_1} = -2\sum_{i=1}^{L} x_i^2(y_i - c_1x_i^2 - c_2x_i - c_3)$=0
$\frac{\partial R}{\partial c_2} = -2\sum_{i=1}^{L} x_i(y_i - c_1x_i^2 - c_2x_i - c_3)$=0
$\frac{\partial R}{\partial c_3} = -2\sum_{i=1}^{L}(y_i - c_1x_i^2 - c_2x_i - c_3)$=0 From this it is possible to write as

a_{11}=$\sum_{i=1}^{L} x_i^4$;
a_{21}=a_{12}=$\sum_{i=1}^{L} x_i^3$;
a_{13}=a_{31}=a_{22}=$\sum_{i=1}^{L} x_i^2$
a_{23}=a_{32}=$\sum_{i=1}^{L} x_i$;
a_{33}=$\sum_{i=1}^{L}$=L
z_1=$\sum_{i=1}^{L} x_i^2y_i$;
z_2=$\sum_{i=1}^{L} x_iy_i$;
z_3=$\sum_{i=1}^{L} y_i$

```
# This is the function that fit the given data to the quadratic function
# where xi and yi are the input data points
Qft=function(xi,yi){
a11=sum(xi^4)
a21=a12=sum(xi^3)
a13=a22=a31=sum(xi^2)
a23=a32=sum(xi)
a33=length(xi)
z1=sum((xi^2)*yi)
z2=sum(xi*yi)
z3=sum(yi)
A=matrix(c(a11,a21,a31,a12,a22,a32,a13,a23,a33),3,3)
z=matrix(c(z1,z2,z3),3,1)
c=solve(A)%*%z
list(c=c)
}
  Example1. Calculate the value of h(t)=v*t-(g/2)*(t^2) for each
  100 minutes where v=20m/s,g=10m/s^2 and then add noise that is
  rnorm(100). After that fit  value to the quadratic equation and approximate
   the value v and -g/2 from the fitted function.
```

```
 getwd()
[1] "C:/Documents and Settings/Tigist/My Documents/Melessew/function"
 source("Qft.R")
 t=matrix(1:100,100,1)# time in minute as column vector
 y=matrix((20*t-5*(t^2)+rnorm(100)),100,1) # the value of h in
                                            #each minute as column vector
 Qft(t,y)
$c
           [,1]
[1,] -5.0001388
[2,] 20.0141247
[3,] -0.5358718
```

These are the coefficients of the fitted quadratic equation. Therefore, c_2=v=20.0141247m/s;c_1=(-g/2)=-5.0001388m/s^2. Therefore the fitted function becomes
y=20.141247*t-5.0001388*(t^2)
The graph of this function is shown in the figure(3) below

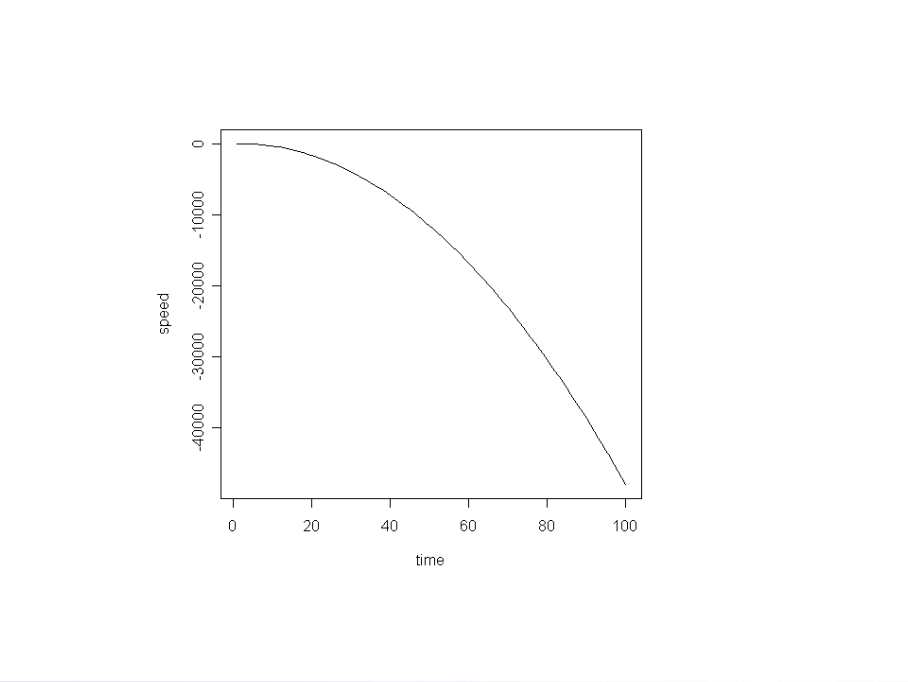


Figure 3: Figure captions

12 points

Spectral Analysis

Fourer analysis is extremely useful for data analysis, as it breaks down a signal into constituent sinusoids of different frequencies. For the sampled vector data ,Fourier analysis is performed using the discrete Fourier transform(DFT).The fourier integral of f(t) is given by F(ω)=$\int_{-\infty}^{\infty} f(t)exp(-i\omega t)dt$ or this equation is called **Fourier transform** of f(t).

But, the functions are often represented by finite sets of discrete values. In this case the fourier transform is said to be discrete fourier transform which can be given by $F(\omega)$=$\sum_{\Delta t=0}^{T-1} g(\Delta t)exp(i\omega\Delta t)$

Each point $F(\omega)$of the transform has an associate frequency, given by f_k=$\frac{k-1}{\tau N}$

where k=0,1,2,...,N-1

τ is the sampling interval

The discrete Foureir Transform decompose the input signal into sine and cosine wave forms. The purpose of decomposition is to obtain easier wave form to deal with than the original signal. The discrete Fourier transform changes an N point input signal into two N/2 +1 point output signals.

The input signal contains the signal being decomposed ,while the two output signals contain the amplitude of components of sine and cosine waves.

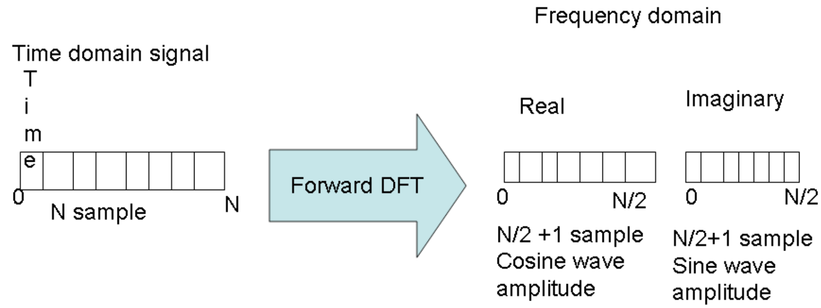

Frequency domain
Time domain signal
T
i
m
e
0
N sample
N
Forward DFT
Real
Imaginary
0
N/2
N/2 +1 sample
Cosine wave
amplitude
0
N/2
N/2+1 sample
Sine wave
amplitude

Figure 4:

1. The time domain signal is the input signal being decomposed by DFT
2. The frequency domain signal is the input signal after being decomposed by DFT

The frequency domain consists:

1. real part that is the amplitude of cosine wave
2. imaginary part that is the amplitude of the sine wave

The fast Fourier transform (FFT) is an efficient algorithm for computing the DFT of a sequence. It is particularly useful in areas such as signal and image processing, where its uses range from filtering ,frequency analysis to power spectrum estimates.

Example 1 Using fft analyze the variation of wind speed measured in each minute in Bahir Dar on 1/1/2006. Analysis: Although the sequence of velocity time series is real,the results of Fourier transform is complex(yf=fft(v)). Notice that the ***time domain*** and the ***frequency domain*** provides an alternative perspective for characterizing the behavior of the oscillating function.

1. The wave form(figure6) Not necessarily. Waveform can also tell you something as little information which is note co ehavior of the time domain function be by the amplitude ,phase angle ,and frequency but not by the shape of the wave.
2. If figure6 is multiplied by by exp(iwt) by taking different w and adding e result is shown in figure7. This is e the speed of wind(time domain). T e domain wind speed in to the continuous frequency domain(figure7).
   (a) As figure7,between frequencies about point10 -point1400, the signal consists of relatively flat region. This is called the White noise,because it contains an equal amount of all the frequencies ,the same as White light. It results from the noise on the time domain waveform being ***uncorrelated*** from sample to sample. That is knowing the noise value present on any one sample provides no information on the noise value present on any other sample.

Home Page
Title Page
Page 83 of 89
Go Back
Full Screen
Close
Quit

The possible source of the noise(error) this wind speed is disturbed wind due to different factors.

(b) There is sharp peak at about frequency points0-9 as shown in figure7. This frequency is the frequency of spectrum of a non-sinusoidal waveform(wind speed). The frequency at this peak value is called the ***fundamental frequency***which is the frequency of time domain data.

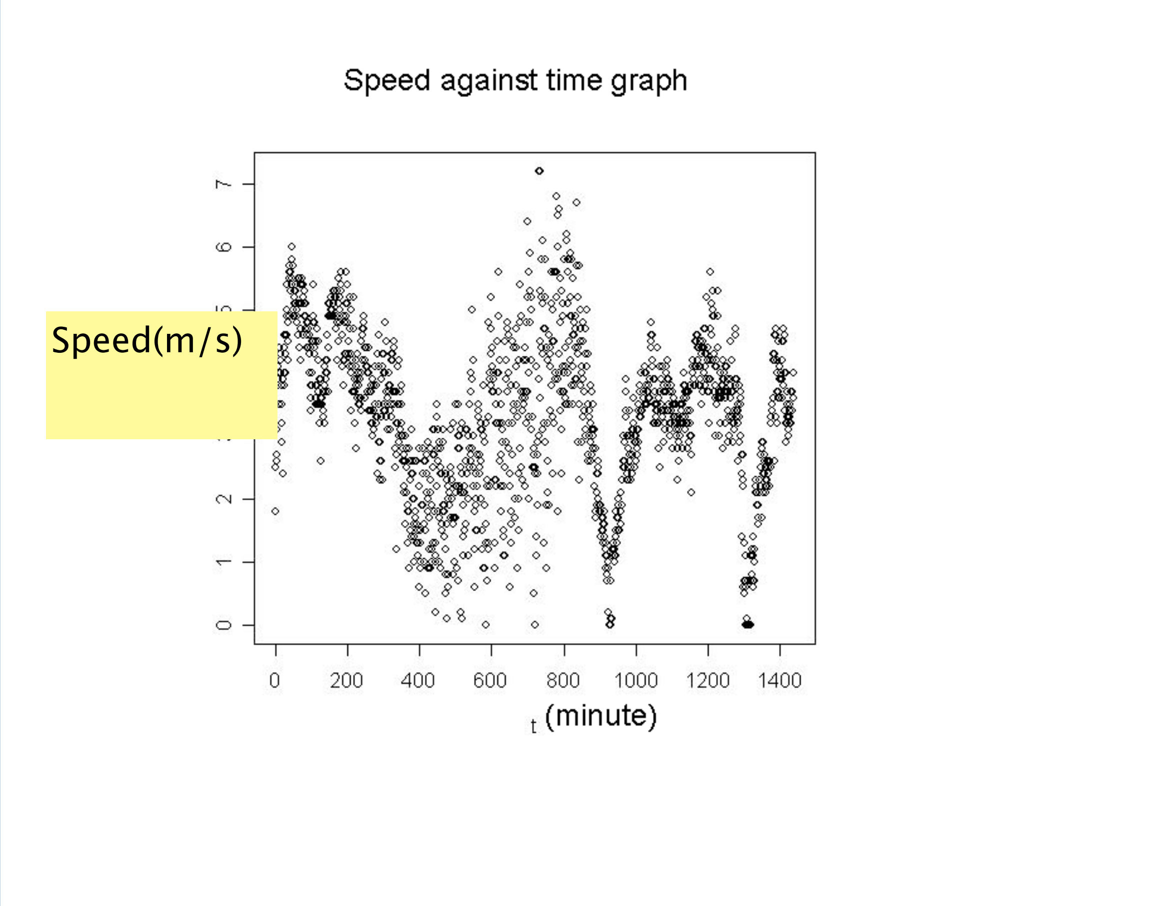

Speed against time graph
Speed(m/s)
0
1
2
6
7
0
200
400
600
800
1000
1200
1400
t (minute)

Figure 5:

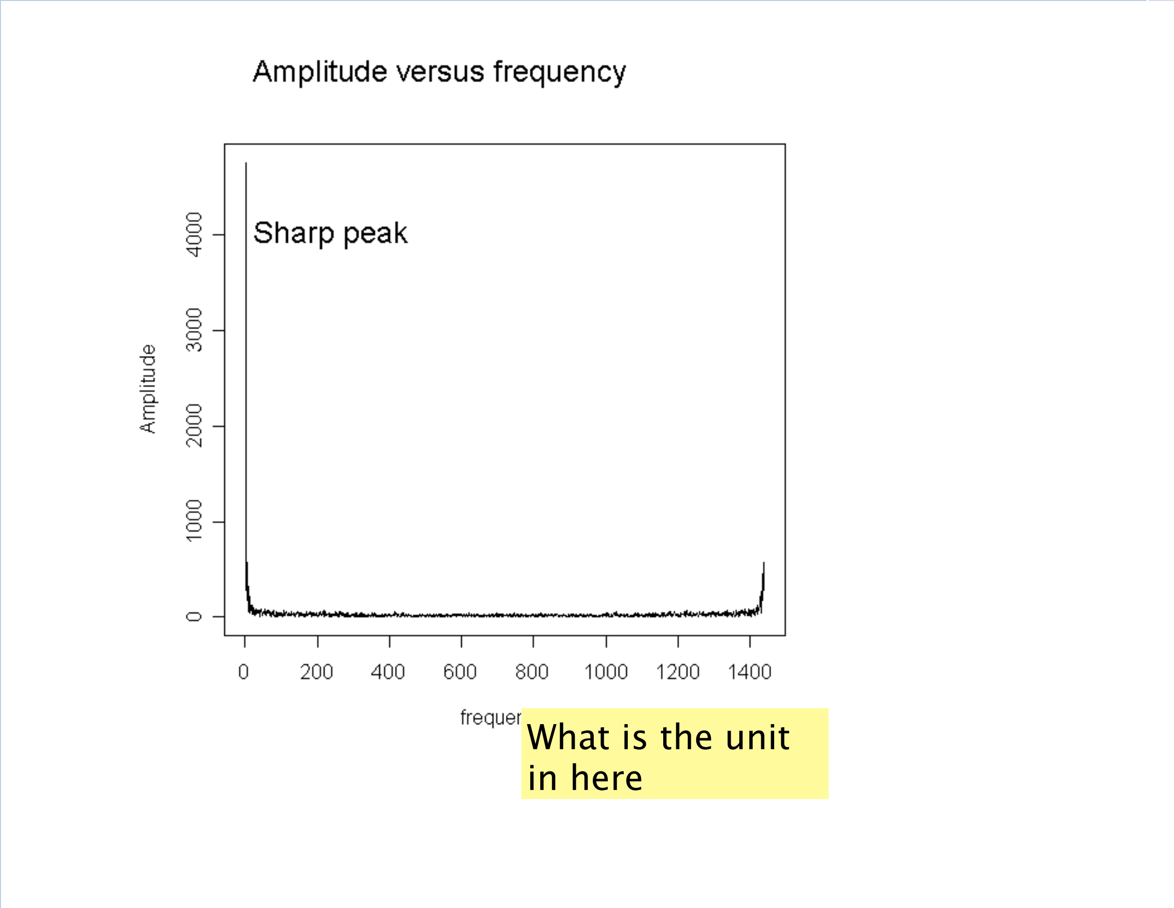


Figure 6: Put figure caption

Spectral analysis by programming

```
#This is program for the  spectral analysis
deltat=0.001 # here we choose the sampling interval, we assume 1000 sample in
t=seq(0,5,deltat) # Here we assume that we take 5000 samples
f1=50 # frequency of the first signal in Hertz
f2=150 # frequency of the second signal in Hertz
f3=100 # frequency of the third signal in Hertz
y1=sin(2*pi*f1*t) # here we create sine curve with 50 Hertz frequency
y2=sin(2*pi*f2*t) # Here we create sine curve with 150 Hertz frequency
y3=sin(2*pi*f3*t) # Here we create cosine curve with 100 Hertz frequency
y=y1+y2+y3 # Here we create signal with two frequencies
f=seq(0,1/(2*(0.001)),1/(length(y)*0.001)) # Here we create frequency axis
Yf=abs(fft(y))/max(abs(fft(y))) # Here I calculate the normalized Fourier tra
par(mfrow=c(3, 1)) # Here we create the row panel for the plots
plot(t,y)
lines(t,y)
plot(f, Yf[1:length(f)],xlim=c(0,max(f)),ylim=c(0,1.5))
lines(f, Yf[1:length(f)],xlim=c(0,length(f)-1),ylim=c(0,1.5))
plot(f, Yf[1:length(f)],xlim=c(0, f3),ylim=c(0,1.5))
lines(f, Yf[1:length(f)])
```

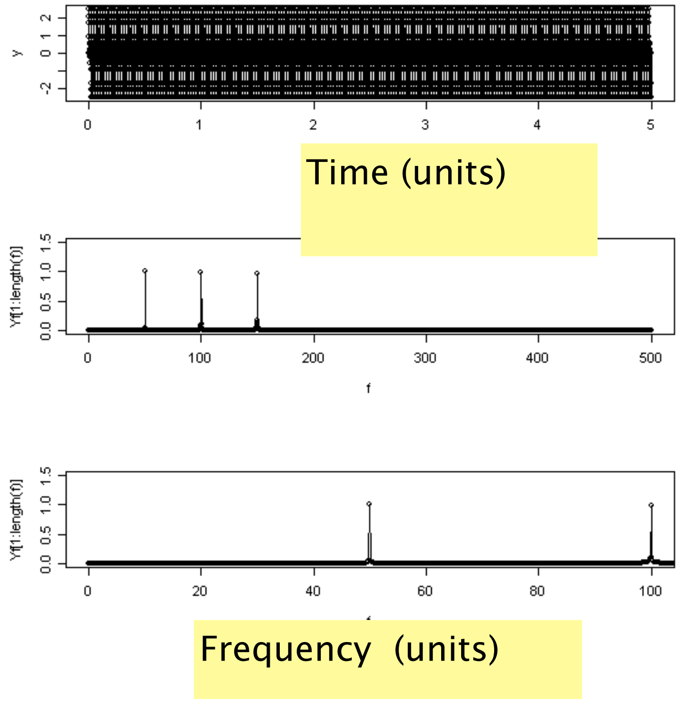
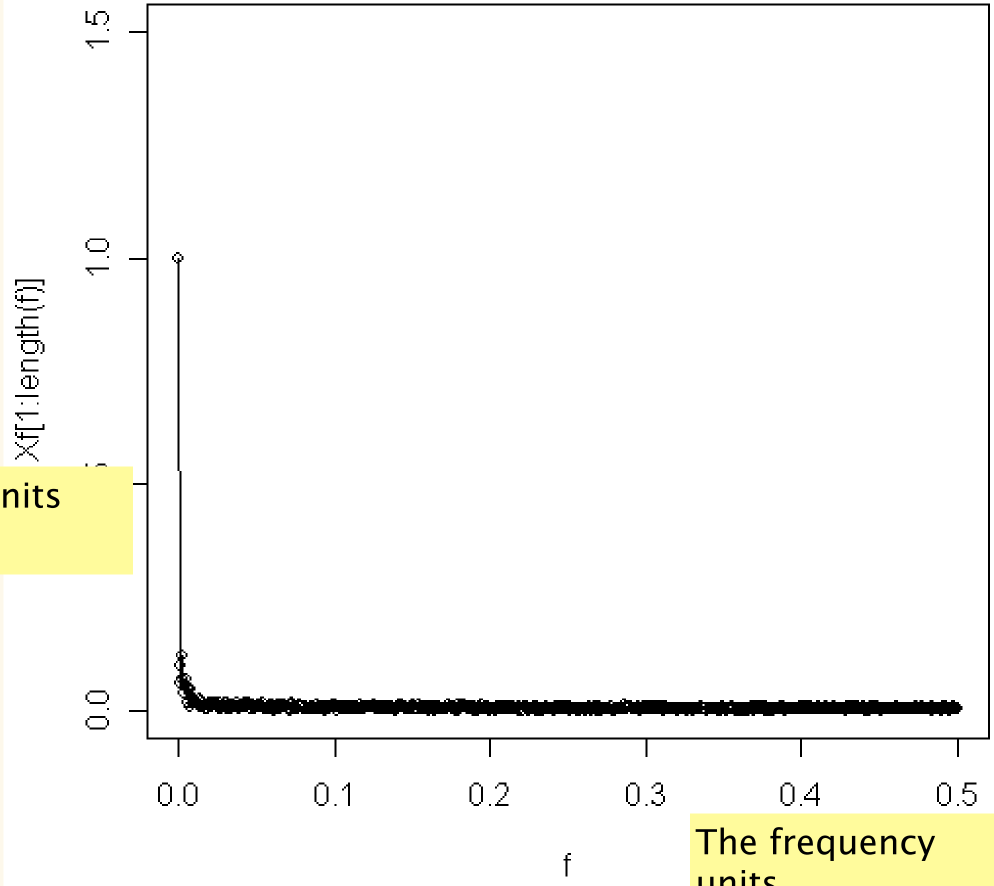


Figure 7:

I used the program written by Dr.Baylie Damtie and analyze the speed of the wind which have 1440 data points which is collected in one minute interval.

```
 x=A[,2]# the speed of the wind
 deltat=1 # time interval between each data observation
 spectrum_analyzer=function(x,deltat){
+ f=seq(0,1/(2*(deltat)),1/(length(x)*deltat))
+ Xf=abs(fft(x))/max(abs(fft(x)))
+ plot(f, Xf[1:length(f)],xlim=c(0,max(f)),ylim=c(0,1.5))
+ lines(f, Xf[1:length(f)],xlim=c(0,length(f)-1),ylim=c(0,1.5))
+ }
 spectrum_analyzer(x,deltat)
```



units

The frequency units