CHE/CEN138

COMPUTER PROGRAMMING

BASIC NUMERICAL METHODS ALGORITHMS, BISECTION METHOD, GAUSS ELIMINATION, ODE SOLVING

References

1.Pratap, R. “Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers”Oxford University Press, 2010.

2.Hunt, B.R., Lipsman, L.R. and Rosemberg J. M. “A guide to MATLAB for Beginners and ExperiencedUsers"Cambridge University Press, 2001.

3.Kubat, C. “MATLAB Yapay Zeka ve Mühendislik Uygulamaları” İkinci Baskı, Pusula Yayıncılık, 2014McGraw Hill, International Edition 2012.

BISECTION METHOD

Explanation: In general, if f (x) is real and continuous in the interval from xl to xu and f (x_L) and f (x_U) have opposite signs, that is,

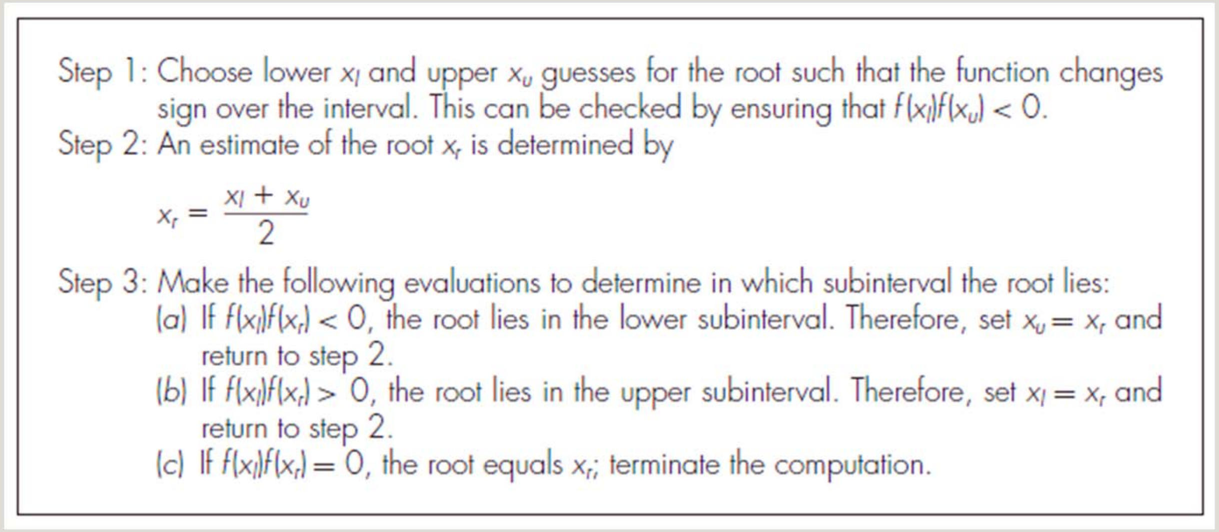
$f(x_L)\ f(x_U) < 0$

then there is at least one real root between xl and xu.

Incremental search methods capitalize on this observation by locating an interval where the function changes sign. Then the location of the sign change (and consequently, the root) is identified more precisely by dividing the interval into a number of subintervals. Each of these subintervals is searched to locate the sign change. The process is repeated and the root estimate refined by dividing the subintervals into finer increments.

The bisection method, which is alternatively called binary chopping, interval halving, or Bolzano's method, is one type of incremental search method in which the interval is always divided in half. If a function changes sign over an interval, the function value at the midpoint is evaluated. The location of the root is then determined as lying at the midpoint of the subinterval within which the sign change occurs. The process is repeated to obtain refined estimates. A simple algorithm for the bisection calculation is given below.

BISECTION METHOD



Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

$$x_r = \frac{x_l + x_u}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

(a) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.

(b) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.

(c) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

BISECTION METHOD

```
xyz=input('Enter your function: ','s')
xl=input('Enter xl:')
xu=input('Enter xu:')
tol=input('Enter stopping criterion:')
kmax=input('Enter max iter:')
f=inline(xyz);
fid=fopen('bisection.txt','w');
fprintf(fid,'Roots of Equation f(x)=x^3-12*x^2+47*x-60 \n\n');
fprintf(fid,'iter     xl       xu       xr       f(xl)       f(xu)       f(xr) \n');
fprintf(fid,'-------------------------------------------------------------------\n');
for i=1:kmax
    ya=f(xl);
    yb=f(xu);
    xr=0.5*(xl+xu);
    yr=f(xr);
    fprintf(fid,'%4.1f %7.4f %7.4f %7.4f   %7.4f   %7.4f   %7.4f\n',i,xl,xu,xr,ya,yb,yr);
    if ya*yr<0
        xu=xr;
    else xl=xr;
    end
    if abs(yr)<tol;
        break
    end
end
fclose(fid);
disp('The root of the equation is')
xr
```

GAUSS ELIMINATION METHOD

Gaussian elimination, also known as **row reduction**, is an algorithm in linear algebra for solving a system of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. To perform row reduction on a matrix, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. Using these operations, a matrix can always be transformed into an upper triangular matrix, and in fact one that is in row echelon form. The process of row reduction makes use of elementary row operations, and can be divided into two parts. The first part (sometimes called forward elimination) reduces a given system to *row echelon form*, from which one can tell whether there are no solutions, a unique solution, or infinitely many solutions. The second part (sometimes called back substitution) continues to use row operations until the solution is found; in other words, it puts the matrix into *reduced* row echelon form.

GAUSS ELIMINATION METHOD

$$A=\begin{bmatrix}144 & 12 & 1\\ 64 & 8 & 1\\ 25 & 5 & 1\end{bmatrix}, X=\begin{Bmatrix}x_1\\ x_2\\ x_3\end{Bmatrix}, B=\begin{Bmatrix}279.2\\ 177.2\\ 106.8\end{Bmatrix} \Rightarrow [AB]=\begin{bmatrix}144 & 12 & 1 & 279.2\\ 64 & 8 & 1 & 177.2\\ 25 & 5 & 1 & 106.8\end{bmatrix}$$

$$[64 \quad 8 \quad 1 \quad 177.2]-\left(\frac{64}{144}\right)[144 \quad 12 \quad 1 \quad 279.2]$$
$$=[0 \quad 2.6667 \quad 0.55555 \quad 53.1111]$$

$$[25 \quad 5 \quad 1 \quad 106.8]-\left(\frac{25}{144}\right)[144 \quad 12 \quad 1 \quad 279.2]$$
$$=[0 \quad 2.9167 \quad 0.8264 \quad 58.3278]$$

$$[AB]=\begin{bmatrix}144 & 12 & 1 & 279.2\\ 0 & 2.6667 & 0.55555 & 53.1111\\ 0 & 2.9167 & 0.8264 & 58.3278\end{bmatrix}$$

GAUSS ELIMINATION METHOD

$$[0 \quad 2.9167 \quad 0.8264 \quad 58.3278] - \left(\frac{2.9167}{2.6667}\right)[0 \quad 2.6667 \quad 0.55555 \quad 53.1111]$$
$$= [0 \quad 0 \quad 0.2188 \quad 0.2376]$$

$$\begin{bmatrix} 144 & 12 & 1 \\ 0 & 2.6667 & 0.55555 \\ 0 & 0 & 0.2188 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 279.2 \\ 53.1111 \\ 0.2376 \end{Bmatrix}$$

$$\begin{array}{lcl} 0.2188x_3 = 0.2376 & \rightarrow & x_3 = 1.0859 \\ 2.6667x_2 + 0.55555x_3 = 53.1111 & \rightarrow & x_2 = 19.6902 \\ 144x_1 + 12x_2 + x_3 = 279.2 & \rightarrow & x_1 = 0.2905 \end{array}$$

GAUSS ELIMINATION METHOD

```
x=zeros(3,1);
AB=[A B]
n=size(A,1);
for i=1:n-1
    P=-(AB(i+1:end,i)/AB(i,i))
    AB(i+1:end,i:end)=AB(i+1:end,i:end)+P*AB(i,i:end)
end
for i=n:-1:1
    x(i,:)=(AB(i,n+1:end)-AB(i,i+1:n)*x(i+1:n,:))/AB(i,i)
end
```

ODE SOLVING

- Solve the following ordinary differential equation using MATLAB's Ordinary Differential Equation (ODE) solver with the initial condition of x(0) = 0 for a time span between 0 and 2.

$$\frac{dx}{dt} = x + t$$

```
function dxdt=difdenk(t,x)
dxdt=x+t;
```

```
[t,x]=ode23('difdenk',[0 2],0)
plot(t,x)
xlabel('t')
ylabel('x')
```

ODE SOLVING

```
t =               x =


        0                 0
   0.0250            0.0003
   0.0500            0.0013
   0.0810            0.0034
   0.1234            0.0079
   0.1791            0.0170
   0.2496            0.0339
   0.3363            0.0634
   0.4400            0.1127
   0.5615            0.1918
   0.7009            0.3146
   0.8579            0.5002
   1.0319            0.7743
   1.2221            1.1716
   1.4221            1.7228
   1.6221            2.4402
   1.8221            3.3606
   2.0000            4.3865
```

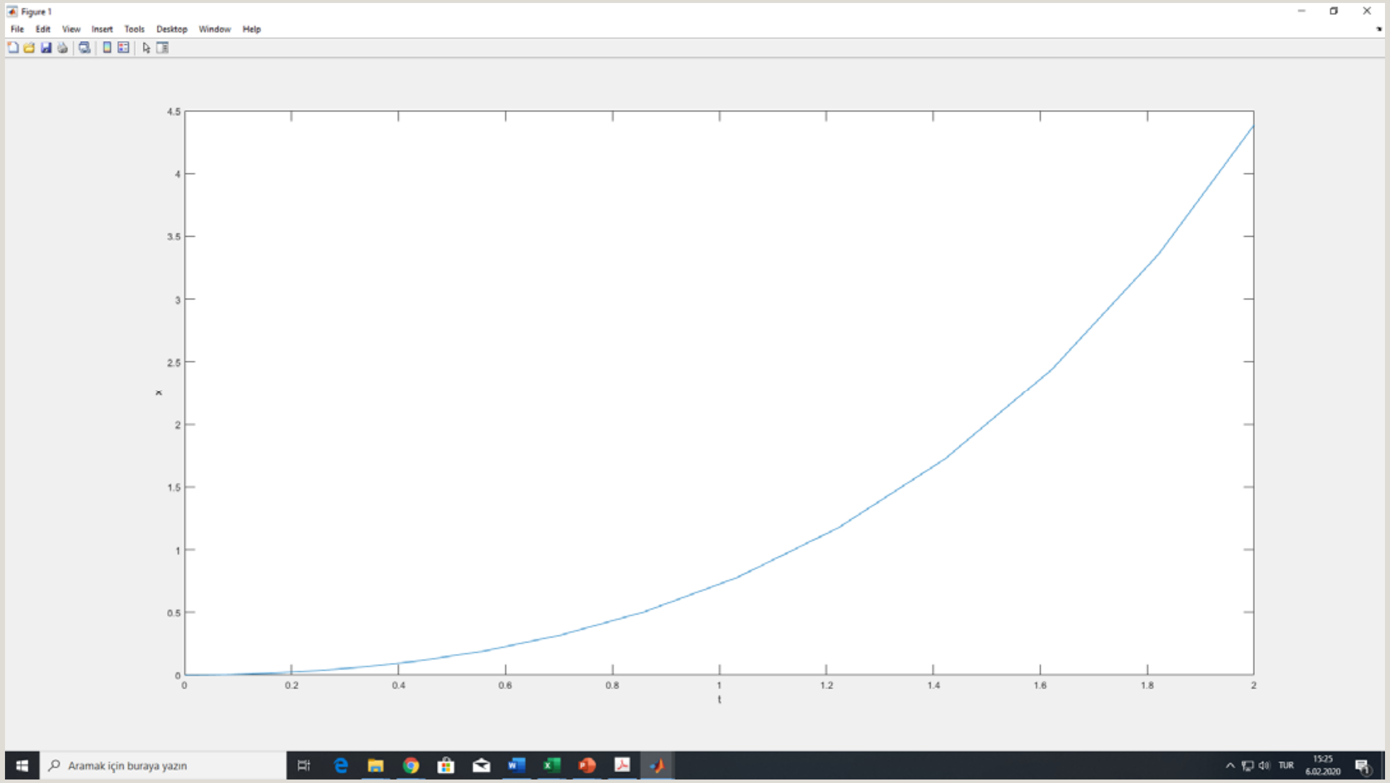


Figure 1
File Edit View Insert Tools Desktop Window Help
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5
x
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
t
Aramak için buraya yazın
TUR
15:25
6.02.2020