The MATLAB Programs for Some Numerical Methods and Algorithms (II)

D.A.GISMALLA
Department of Mathematics & Computer Studies
Faculty of Science & Technology
Gezira University, Wadi medani,
Sudan

Abstract- **In[1] , We have listed and described some numerical methods and techniques for the reader so that he can be acquainted with them and then a description program flow chart is mentioned without the Matlab Software Program. Here , the Matlab Software Program for the Methods and Algorithms mentioned in [1] . The SOFTWARE PROGRAMS are for Differentiation, Integration and Ordinary Differential Equations.**

Keywords: **Trapezoidal Rule**, **Midpoint Method**, **Simpson Rule, Gaussian Quatrature, Euler's Method**

I. NUMERICAL DIFFERENTIATION

A. Forward and Backward-Difference Formula

```
% Example 1.1  the forward and
% backward-difference formula
% In Matlab command window
% h=[0.1 0.01  .001];
% xo=1.8;
% syms x
% f=inline('log(x)')
% diff=fdiff1(f,h,xo)

function diff=fdiff1(f,h,xo)
n=length(h);
for i=1:n
d=xo+h(i);
fexact(i) = feval(f,d);
diff(i)= (feval(f,d)-feval(f,xo))/h(i);
error(i) = abs(fexact(i)- diff(i));
end
disp('    h      fexact   diff(f(h))  error ')
disp([h'    fexact'          diff'    error' ] )
```

Manuscript received September 8, 2016
D.A.GISMALLA, Department of Mathematics & Computer Studies Faculty of Science & Technology Gezira University, Wadi medani, Sudan

B. Three –Point Differentiation Formula for f'

```
% Example 1.2 the three point
% differentiation formula for f'
% In Matlab command window Type
% h=[0.1];
% xo=2.0;
% syms x
% f=inline('x*exp(x)')
% diff=fdif1(f,h,xo)

function  diff=fdif1(f,h,xo)
n=length(h);
for i=1:n
d=xo+h(i);
d0=xo-h(i);
diffexact(i) = feval(f,xo) + exp(xo);
diff(i)= 1/(2*h(i))*(feval(f,d)-feval(f,d0));
error(i) = abs(diffexact(i)- diff(i));
end
disp('    h    diffexact   diff(f(h))  error ')
disp([h'      diffexact'      diff'    error' ] )
```

C. The Five –Point Differentiation Formula for f'

```
% Eample 1.3 the five point differentiation
% formula for f'
% In Matlab command window Type
% h=[0.1];
% xo=2.0;
% syms x
% f=inline('x*exp(x)')
% diff=fdif(f,h,xo);

function  diff=fdif(f,h,xo)
n=length(h);
for i=1:n
d_2=xo- 2*h(i);
```

```
d_1=xo-h(i);
d1=xo+h(i);
d2=xo+ 2*h(i);
diffexact(i) = feval(f,xo) + exp(xo);
diff(i)= 1/(12*h(i))*(feval(f,d_2)-8*feval(f,d_1)
            + 8*feval(f,d1) - feval(f,d2));
error(i) = abs(diffexact(i)- diff(i));
end
disp('      h    diffexact   diff(f(h))  error ')
disp([h'     diffexact'           diff'     error' ] )
```

D. Three-point Differentiation Formula for f''

```
%  Example 1.4 the three point differentiation
%  formula for f''
%  In Matlab command window Type
%  h=[0.1  0.2];
%  xo=2.0;
%  syms x
%  f=inline('x*exp(x)')
%  diff=fdif2(f,h,xo);

function  diff=fdif2(f,h,xo)
n=length(h);
for i=1:n
d=xo+h(i);
d0=xo-h(i);
diffexact(i) = feval(f,xo) + 2*exp(xo);
diff(i)=(1/(h(i)^2))*(feval(f,d0)-
          2*feval(f,xo)+feval(f,d));
error(i) = abs(diffexact(i)- diff(i));
end
disp('      h    diffexact   diff(f(h))  error ')
disp([h'     diffexact'          diff'    error' ] )
```

II. NUMERICAL INTEGRATION

A. Integration by Using Taylor's Expansion

```
%  Expand the function by Taylor's Expansion   %   and
then integrate
%   Example 3.1 Integrate F(x) =√ (1+x)
%    on    the     interval (0, 0.1)
%   In the Matlab Command Window  Type
%    format long
%    syms x ;
%    f=inline (’sqrt(1+x)’) ;
%    f2=taylor(f(x),3,0) ;
%    Ie=int(f2,x,0,0.1) ;
%    Ir=int(’sqrt(1+x)’,x,0,0.1) ;
%    RealError=abs(Ir-Ie) ;
%    disp('     f2     Ie    Ir     realerror ')
%    disp([     f2     Ie   Ir     RealError])
```

B. Integrate Using the Composite Trapezoidal Rule

```
% Example 3.2   Integrate using the Composite
% Trapezoidal Rule
% In Matlab command window
% a=1.1;
% b=1.6;
% n =5 ;
% syms x;
% f=inline('exp(x)') ;
% Ie=compTrapezoidal(f,a,b,n)

function [Ie ,Ir ,Error]=compTrapezoidal(f,a,b,n)
h=(b-a)/n;
x=a:h:b;
sum=0;
for i=2:n
sum=sum+feval(f,x(i));
end
% the approximate value of integral Ie
Ie=(h/2)*(feval(f,a)+feval(f,b)+2*sum);
% The exact value  of In tegral Ir
 Ir=int('exp(x)' ,a,b);
% The absolute Actual Error
%  = abs(Approximation - Exact)
 Error = abs(Ie-Ir);
```

C. Integrate Using the Composite Simpson Rule

```
%  In Matlab command window
%  a=1.1;
%  b=1.6;
%  n = 3 ;
%  n is even i.e. n=2*n divisions
%            in the formula
%  syms x;
%  f=inline('exp(x)');
%  Ie=compSimpson(f,a,b,n)

function[Ie,Ir,Error]=
compSimpson(f,a,b,n)
h=(b-a)/(2*n);
x=a:h:b;
sum=feval(f,a);
for i=2:2:2*n
sum=sum+4*feval(f,x(i))
+2*feval(f,x(i+1));
end
sum=sum-feval(f,b);
% The approximate Integral Ie
Ie = (h/3)*sum;
%  The exact value  of Integral Ir
Ir=int('exp(x)',a,b);
% The absolute Actual Error =
 %  abs(Appoximation - Exact)
Error = abs(Ie-Ir);
```


D. Gaussian Quatrature rules

a. Example 3.4(a) Integration using Gaussian Quatrature rules

```
% In Matlab command window
% syms x;
% f=inline('exp(-x^2)');
% c=[1.0000000000 1.0000000000];
% x=[0.5773502692 -0.5773502692];
% a=1; b=1.5 ;
% n=2 ;
% I=quassian(f,c,x,a,b,n)

function Ie=quassian(f,c,x,a,b,n)
sum=0;
for j=1:n
t=((b-a)*x(j)+a+b)/2;
sum=sum+c(j)*feval(f,t)*(b-a)/2;
end
Ie=sum;
```

b. Example 3.4(b) Integration Using Guassian Quatrature rules

```
% In Matlab command window Type
% syms x;
% f=inline('exp(-x^2)');
% a=1; b=1.5 ;
% n=2 ; % Gaussian-Table is given
% and n runs from 2 to 5
% I=quassianTable(f,a,b,n)

function [Ie,Ir,Error]=quassianTable(f,a,b,n)
c=[1.00000000    0.5555555556    0.3478548451
0.2369268850
1.0000000000    0.8888888889    0.6521451549
0.4786286705
0.0000000000    0.5555555556    0.6521451549
0.5688888889
0.0000000000    0.0000000000    0.3478548451
0.4786286705
0.0000000000    0.0000000000    0.0000000000
0.2369268850];
x=[0.5773502692  0.7745966692  0.8611363116
      0.9061798459
-0.5773502692   0.0000000000   0.3399810436
      0.5384693101
0.0000000000  -0.7745966692  -0.3399810436
      0.0000000000
0.0000000000   0.0000000000  -0.8611363116
      -0.5384693101
0.0000000000   0.0000000000   0.0000000000
      -0.9061798459];
sum=0;
for j=1:n
t=((b-a)*x(j,n-1)+a+b)/2;
sum=sum+c(j,n-1)*feval(f,t)*(b-a)/2;
end
Ie=sum;
Ir=int('exp(-x^2)',a,b);
Error = abs(Ie-Ir);
```

III. ORDINARY DIFFERENTIAL EQUATIONS

A. Euler's Method

```
% Eample 4.1 Euler's Method
% In the Command Window
% Submit the derivative as function f
% & the approximation is yappr
% number of Iteration n
% Submit the exact soultion as function yexact
% h=0.1;
% y0=1.0;
% n=4;
% syms t ;
% syms yappr
% f=inline('3*exp (-4*t)-2*yappr');
% y=inline('2.5*exp (-2*t)-1.5*exp (-4*t)');
function [yappr , yexact , error] = EULERMAT(f,y ,h, y0,n);

% initialize time variable.
t=0:h:n*h;
% initial condition (same for approximation).
yappr(1)=y0;
for i=1:n-1     % Set up "for" loop.
% Calculate derivative;
k1= feval(f,t(i),yappr(i));
% Estimate new value of y;
yappr(i+1)= yappr(i)+h*k1;
end
for i=1:n
% exact solution
yexact(i)=feval(y,t(i));
error(i)= abs(yappr(i)-yexact(i));
end
disp( '    yappr    yexact     error')
disp([ yappr'  yexact'  error'])
```

B. Midpoint Method

```
% Eample 4.2 MIDPOINT Method
% In the Command Window
% Submit the derivative as function f
% & the approximation is yappr
% number of Iteration n
```



```
% Submit the exact soultion as function yexact
% h=0.1;
% y0=1.0;
% n=4;
% syms t ;
% syms yappr
% f=inline('3*exp (-4*t)-2*yappr');
% y=inline('2.5*exp (-2*t)-1.5*exp (-4*t)');

function [yappr , yexact , error] = MIDPOINT(f,y ,h, y0,n);

% initialize time variable
t=0:h:n*h;.
% initial condition (same for approximation).
yappr(1)=y0;
for i=1:n-1     % Set up "for" loop.
% Calculate derivative;
k1= h*feval(f,t(i),yappr(i));
% Calculate midpoint derivative;

k2= h*feval(f,(t(i)+0.5*h),(yappr(i)+0.5*k1));
% Estimate new value of y;
yappr(i+1)= yappr(i)+h*k2;
end
for i=1:n
yexact(i)=feval(y,t(i));  % exact solution
error(i)= abs(yappr(i)-yexact(i));
end
disp( '     yappr    yexact     error')
disp([ yappr'  yexact'  error'])
```

C. RungeKutta Method

```
% Eample 4.2 RUNGEKUTTA Method
% In the Command Window
% Submit the derivative as function f
%  & the approximation is yappr
%  number of Iteration n
%  Submit the exact soultion as function yexact
%  h=0.1;
%  y0=1.0;
%  n=4;
%  syms t ;
%  syms yappr
%  f=inline('3*exp (-4*t)-2*yappr');
%  y=inline('2.5*exp (-2*t)-1.5*exp (-4*t)');

function [yappr , yexact , error] = RUNGEKUTTA(f,y ,h,
y0,n);

t=0:h:n*h;  % initialize time variable.
% initial condition (same for approximation).
yappr(1)=y0;
for i=1:n-1     % Set up "for" loop.
% Calculate derivative;
k1= h*feval(f,t(i),yappr(i));
% Calculate midpoint derivative;
k2= h*feval(f,(t(i)+0.5*h),(yappr(i)+0.5*k1));
% Calculate midpointderivative;
k3= h*feval(f,(t(i)+0.5*h),(yappr(i)+0.5*k2));
% Calculate final_point derivative;
k4= h*feval(f,(t(i)+h),(yappr(i)+k3));
k=(k1+2*k2+3*k3+k4)/6;
% Estimate new value of y;
yappr(i+1)= yappr(i)+k;
end
for i=1:n
yexact(i)=feval(y,t(i));  % exact solution
error(i)= abs(yappr(i)-yexact(i));
end
   disp( '     yappr    yexact     error')
     disp([ yappr'  yexact'  error'])
```

IV. CONCLUSION

The Software MATLAB Programs for paper(I) and paper(II) were copied from the original M_files and then pasted on Microsoft Word Documented Format.

The MATLAB that can be OPENS directly and runs in the MATLAB MODE INVIROMENT can be found in on separate attached FILE DIRECTORY CALLED MATLABNUM containing the four sub directly that We have described in [1].

REFERENCES

[1] International Journal of Algorithms, Computing and Mathematics, Volume 5 , number 1 pp.58-68 , Feb. 2012 India www.eashwarpublications.com

[2] Richard L. Burden, J. Douglas Faires, Numerical Analysis Prindle ,Weber & Schmidt

[3] Birkhoff,G and G. Rota(1978) Ordinary Differential Equation John Wiley & Sons, New York; 342 PP.QA372,B58

[4] www.swarthmore.edu/NatSci/echeeve1/Ref/-NumericInt/RK2.html

