Program Ordinary Differential Equation 2 - ODE2

```
function ode2
%**********************************************************************
%                                                                     *
%                                                                     *
%                    FINITE ELEMENT METHOD (FEM)                      *
%                                                                     *
%                                                                     *
%      PROGRAM ODE2                                                   *
%                                                                     *
%      A FINITE ELEMENT PROGRAM FOR THE SOLUTION OF THE LINEAR        *
%      SECOND ORDER DIFFERENTIAL EQUATION                             *
%                                                                     *
%         D( A DU/DX)/DX + B DU/DX + C U + D = 0                      *
%                                                                     *
%      WHERE A, B, C, AND D CAN BE ANY SPECIFIED FUNCTION OF X        *
%                                                                     *
%        EQUIVALENT FINITE ELEMNET EQUATION:                          *
%                                                                     *
%             SK(I,J)*U(J)=Q(I)                                       *
%                                                                     *
%                                                                     *
%        NUMEL    = NUMBER OF ELEMENTS                               *
%        XDIS     = LENGTH IN X-DIRECTION                            *
%        XORD(I)  = X-COORDINATE OF EACH NODE                        *
%        U(I)     = NODAL VALUE OF DEPENDENT VARIABLE U(X)           *
%        Q(I)     = EQUIVALENT NODAL VALUE OF D(X).                  *
%                   WHEN SPECIFIED AS BOUNDARY CONDITION IT ALSO     *
%                   INCLUDES (A DU/DX) EVALUATED AT BOUNDARY POINT   *
%        NPBC(I)  = NODAL POINT BOUNDARY CONDITION                   *
%                     NPBC = 0: Q OR RHS IS KNOWN                    *
%                               U IS UNKNOWN                         *
%                     NPBC = 1: Q IS UNKNOWN                         *
%                               U IS KNOWN                           *
%        GPTS(I)  = ARRRAY OF GUASSIAN QUADRATURE POINTS             *
%        GWTS(I)  = ARRAY OF WEIGHTS FOR ABOVE POINTS                *
%                                                                    *
%        DIMENSIONS: MXNN = NUMEL + 1                                *
%                                                                    *
%                                                                    *
%**********************************************************************
```

Program Ordinary Differential Equation 2 - ODE2

```
    clc

%**********************************************************************
%  Gaussian   Quadrature Points and Weights                          *
%**********************************************************************

gpts(1) = 0.93246951; gpts(2) = 0.66120938; gpts(3) = 0.23861018;
gpts(4) = -gpts(3); gpts(5) = -gpts(2); gpts(6) = -gpts(1);

gwts(1) = 0.17132449; gwts(2) = 0.36076157; gwts(3) = 0.46791393;
gwts(4) = gwts(3); gwts(5) = gwts(2); gwts(6) = gwts(1);

%=========================================
%      OPEN INPUT FILES
%=========================================
fileID = -1;
errmsg = '';
while fileID < 0
   disp(errmsg);
   file = input('Open file: ', 's');
   filein  = strcat(file,'.in');
   [fileID,errmsg] = fopen(filein);
end

fileout = strcat(file,'.out');

%============================
%      ECHO INPUT DATA     =
%============================
numnp = fscanf(fileID,'%d',1);
numel = numnp-1;

X = ['Number of nodes    : ',num2str(numnp)];
disp(X)
X = ['Number of elements : ',num2str(numel)];
disp(X)
```

6-point Gaussian quadrature from -1 < x < 1

Gaussian quadrature points

Gaussian quadrature weights

Opens input file *.in (must exist) and *.out file for results.

Program Ordinary Differential Equation 2 - ODE2

```
xord(1:numnp,1)=0;
npbc(1:numnp,1)=0;
u(1:numnp,1)=0;
q(1:numnp,1)=0;

s1 = zeros(numnp);
s2 = zeros(numnp);
s3 = zeros(numnp);
sq = zeros(numnp,1);

l = 1;

while feof(fileID) == 0
    A = fscanf(fileID,'%d %d %f %f %f',5);
    np=A(1);
    npbc(np)=A(2);
    xord(np)=A(3);
    u(np)=A(4);
    q(np)=A(5);
    ll = np - l;
    if ll > 0
        diff = np + 1 - l;
        dx = (xord(np) - xord(l - 1)) / diff;
        dq = (q(np) - q(l - 1)) / diff;

        for i = l:np - 1
                npbc(i) = npbc(i - 1);
                xord(i) = xord(i - 1) + dx;
                q(i) = q(i - 1) + dq;
                l = l + 1;
        end %for loop
    end %if
    l = l + 1;
end %while loop

fclose(fileID);
```

Allocate arrays

Read input file *.in

Generate values for nodes, u(x), and q(x)

Program Ordinary Differential Equation 2 - ODE2

```
fileID = fopen(fileout,'w');
fprintf(fileID,'%s\r\n',' ');
fprintf(fileID,'%s\r\n', ' ***************************************************************');
fprintf(fileID,'%s\r\n', ' **                                                           **');
fprintf(fileID,'%s\r\n', ' ***           ODE2 - The University of Memphis              ***');
fprintf(fileID,'%s\r\n', ' **                                                           **');
fprintf(fileID,'%s\r\n', ' ***************************************************************');
fprintf(fileID,'%s\r\n', '');
fprintf(fileID,'%s\r\n', 'Input Data');
fprintf(fileID,'%s\r\n', '------------------------------------------------------------------------');
fprintf(fileID,'%s\r\n', 'Node         Code              x             u             q ');
fprintf(fileID,'%s\r\n', '------------------------------------------------------------------------');

for i = 1:numnp
    fprintf(fileID,'%4i %11i %18.5f %13.5f %13.5f\r\n', i, npbc(i), xord(i),u(i), q(i));
end % Next i

%==================================
%      BEGIN INITIALIZATION     =
%==================================
    nqpts = 6;
```

Echo input file data to output file *.out

Program Ordinary Differential Equation 2 - ODE2

```
%====================================
%      FORMATION OF SK MATRIX      =
%====================================
    for i = 1:numel

        rl = xord(i + 1) - xord(i);
        xi = xord(i);
        dxds = rl / 2;
        dsdx = 1 / dxds;

%=======================================
%      BEGIN GUASSIAN QUADRATURE      =
%=======================================
        for j = 1:nqpts
            xl = (rl / 2) + (rl * gpts(j)) / 2;
            xg = xi + xl;
            s = gpts(j);
            sw = gwts(j);
            dndx(1) = dsn(1, s) * dsdx;
            dndx(2) = dsn(2, s) * dsdx;

            for k = 1:2
                for l = 1:2
                    ke = (i-1)+k;
                    le = (i-1)+l;
                    s1(ke, le) = ********** BLEEP **********;
                    s2(ke, le) = s2(ke, le) + sn(k, s) * b(xg) * dndx(l) * sw * dxds;
                    s3(ke, le) = ********** BLEEP **********;
                end % l
                sq(ke) = sq(ke) + sn(k, s) * d(xg) * sw * dxds;
            end % k
        end % j
    end %i = 1:numel

    sk = s1-s2-s3;
```

Length of element

$\frac{d\xi}{dx}$

Maps Gauss point into global coordinates

Interpolation Function defined in $-1 < \xi < 1$

Terms associated with a(x)

Terms associated with c(x)

Position in **sk** array as a function of the element number *i*

Program Ordinary Differential Equation 2 - ODE2

```
%=====================================================================
%      ALL ELEMENTS ARE NOW ASSEMBLED IN GLOBAL MATRIX EQUATION.     =
%      KNOWN VALUES OF U(I) MUST BE ACCOUNTED FOR.                   =
% ====================================================================

 for i = 1:numnp
        if npbc(i) == 1
            for j = 1:numnp
                if j ~= i
                    sq(j) = sq(j) - u(i)*sk(j,i);
                end
            end
            sk(i,1:numnp) = 0;
            sk(1:numnp,i) = 0;
            sk(i,i) = 1;
            sq(i) = u(i);
        end

    end %i

    u = sk\sq;
```

Apply boundary conditions

Solve equations

Program Ordinary Differential Equation 2 - ODE2

```
%==================
% Plot results
%==================
    plot(xord,u,'b-o','LineWidth',2,'MarkerFaceColor','b');
    title('ODE2')
    xlabel('x')
    ylabel('u(x)')

%============================
% PRint to output file *.out
%============================
    fprintf(fileID,'%s\r\n',' ');
    fprintf(fileID,'%s\r\n', 'Output Data');
    fprintf(fileID,'%s\r\n', '-------------------------------------------------------------------------');
    fprintf(fileID,'%s\r\n', 'Node             x             u ');
    fprintf(fileID,'%s\r\n', '-------------------------------------------------------------------------');

    for i = 1:numnp
        fprintf(fileID,'%4i %16.5f %13.5f\r\n', i, xord(i),u(i) );
    end %i

    fprintf(fileID,'%s\r\n',' ');
    fprintf(fileID,'%s\r\n', 'ODE2 analysis complete');

    fclose(fileID);

disp('ODE2 analysis complete')

%End of ODE2 function ====================================================
end
```

Plot results

Print results to *.out file

Program Ordinary Differential Equation 2 - ODE2

```
function sn1 = sn(i, x)
    if i == 1
        sn1 = *********** BLEEP **********;
    else
        sn1 = 0.5 * x + 0.5;
    end
end

function dsn1 = dsn(i)
    if i == 1
        dsn1 = *********** BLEEP **********;
    else
        dsn1 = 0.5;
    end
end

function a1 = a(x)
    a1 = 29000;
end

function b1 = b(x)
    b1 = 0;
end

function c1 = c(x)
    c1 = 0;
end

function d1 = d(x)
    d1 = 10*(1-x/120);
end
```

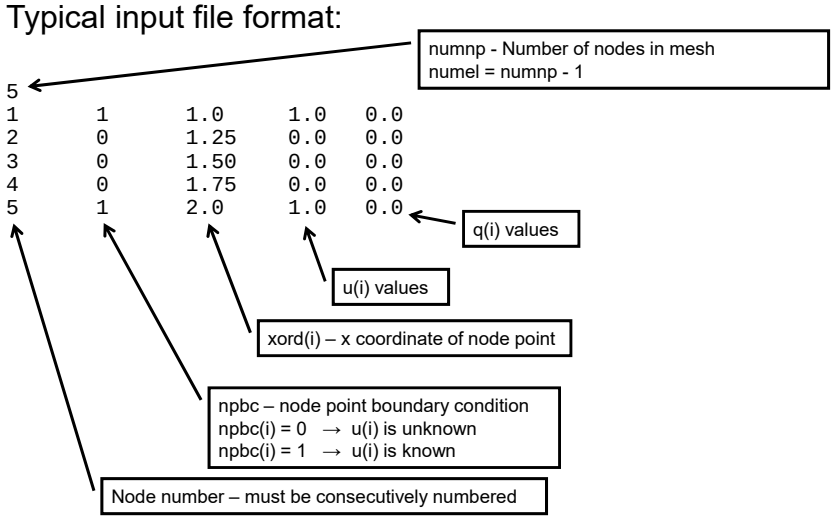
N - linear interpolation functions -1 < x < 1

N' - linear interpolation functions -1 < x < 1

General function for:
d(A dU/dX)/dX + B dU/dX + C U + D = 0

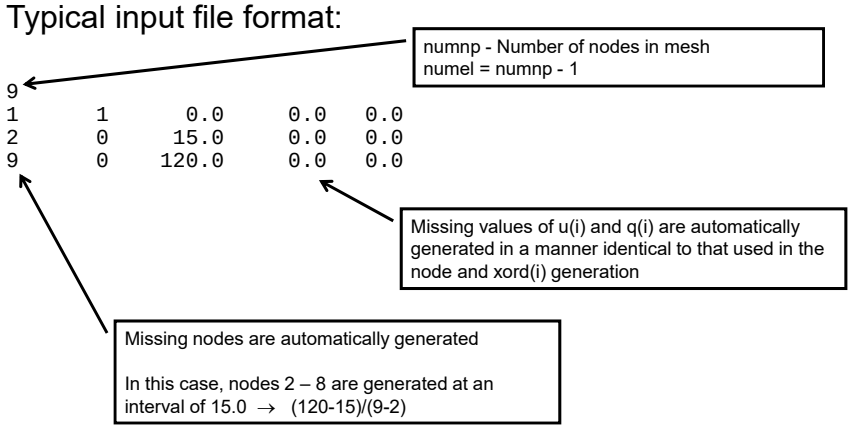
Program Ordinary Differential Equation 2 - ODE2

Typical input file format:



Program Ordinary Differential Equation 2 - ODE2

Typical input file format:



Program Ordinary Differential Equation 2 - ODE2

Typical output file format:

```
 ************************************************************
 **                                                        **
 ***          ODE2 - The University of Memphis            ***
 **                                                        **
 ************************************************************

Input Data
------------------------------------------------------------------------
Node         Code             x             u             q
------------------------------------------------------------------------
 1            1            0.00000       0.00000       0.00000
 2            0            15.00000      0.00000       0.00000
 3            0            30.00000      0.00000       0.00000
 4            0            45.00000      0.00000       0.00000
 5            0            60.00000      0.00000       0.00000
 6            0            75.00000      0.00000       0.00000
 7            0            90.00000      0.00000       0.00000
 8            0            105.00000     0.00000       0.00000
 9            0            120.00000     0.00000       0.00000

Output Data
------------------------------------------------------------------------
Node             x             u
------------------------------------------------------------------------
 1           0.00000       0.00000
 2           15.00000      0.27317
 3           30.00000      0.47845
 4           45.00000      0.62554
 5           60.00000      0.72414
 6           75.00000      0.78394
 7           90.00000      0.81466
 8           105.00000     0.82597
 9           120.00000     0.82759
```

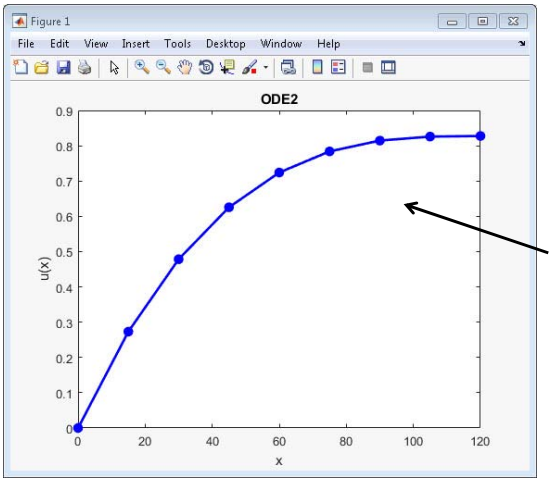
Listing of generated values of xord(i), u(i), and q(i)

FEM solution for the u(x) at the listed nodes

Program Ordinary Differential Equation 2 - ODE2

```
>>ode2
Open file: test1
Number of nodes    : 9
Number of elements : 8
ODE2 analysis complete
>>
```

Problem has been solved



Plot of solution

End of
ODE2 Program