**Solution of a 1-D differential equation using Finite Element Method (code in FORTRAN90)**

**Created By**:

Ajay Malkoti

**Description of the problem**

The problem is to code for solution of 1D differential equation of following form:

Given boundary conditions are : y(0)=0 and y(L)=0

**Important Variables and sample inputs to them**

**Parameters used as Global Parameters.**

X0=0, Domain Length XA and XB (REAL)

X1=1

ACOEF=1,

BCOEF=-3 Differential Equation Coefficients (REAL)

CCOEF=2

NEL=10 No Of Element (INTEGER)

NNEL=2 No Of Node Per Element (INTEGER)

NDOF=1 Degree Of Freedom (INTEGER)

NNODE=11 No Of Nodes In System (INTEGER)

EDOF=1 Element Degree Of Freedom(DOF)

SDOF Total DOF of System (abbr. SDOF) = NNODE\*NDOF

**Global Matrices and corresponding subroutines**

COORD(:) Coordinate Is a Vector/ 1-D Matrix Which Stores The Coordinate Of Each Node

**Subroutine** to generate COORD is COORDINATE()

TOPO(:,:) Topology Is The Matrix Which Keeps Account Of Each Element Node

**Subroutine** to generate TOPO is TOPOLOGY()

BC(:,:) Boundary Conditions Flag Prescribed At Which Node The Bc Are Prescribed

BCV(:,:) Boundary Condition Value At The Prescribed Node

**Subroutine** to generate BC and BCV is BOUNDARY\_CONDITION()

K(:,:) Set Up Element matrix (REAL)

F(:,:) Set Up RHS function Matrix (REAL)

**Subroutine** to generate K and F is ELEMENT()

KK(:,:) Set Up Global Elemental Matrx (REAL)

FF(:,:) Set Up Global RHS Function Matrix (REAL)

**Subroutine** to generate/assemble KK and FF is ASSEMBLE()

U(:,:) Solution Vector

**Subroutine** to generate sol vector U is SOLVE()

**Organization of program**

Line 1 – 25: Define variable etc

Line 36 : Setting up the coordinate matrix

Line 39 : Call TOPOLOGY to set up topology matrix

Line 42 : Call BOUNDARY\_CONDITION to set BC and BCV

Line 63 : Initializing the global matrix KK and FF

Line 68-99: **LOOP:** Set up the global matrix using do loop

Line 110: Call APP\_BC to apply boundary condition

Line 124: Solve the matrix for the final solution

Line 126: Call ENP to evaluate and print the solution with exact answer.

**Function and Subroutines**

FUNCTION RHS\_FUN(X)

SUBROUTINE COORDINATE()

SUBROUTINE TOPOLOGY() ! To find the nodes connected to the respective element

SUBROUTINE BOUNDARY\_CONDITION()

SUBROUTINE SYS\_DOF(IEL)

SUBROUTINE ELEMENT(I,XE) !for Ith element, WE ARE COMPUTING K OR STIFFNES MATRIX

SUBROUTINE NSHAPE(X,P,N)\* ! Shape Function

SUBROUTINE DSHAPE(X,P,B)\* !Derivative Of Shape Function

SUBROUTINE GAUSS(NGP,GP,W)\* !Compute The Gasussian Points And Weights

SUBROUTINE ASSEMBLE(K,F)

SUBROUTINE APP\_BC()

SUBROUTINE SOLVE(U)

SUBROUTINE SOLVE\_TRIDIAG(A,B,C,D,X,N)

SUBROUTINE FSOL(V,L,FF)\* !FOR UPPER TRAINGUALR L(NxN)\*U(Nx1)=Y(Nx1)

SUBROUTINE BSOL(U,LT,V)\* !FOR LOWER TRAINGUALR L'(NxN)\*U(Nx1)=Y(Nx1)

SUBROUTINE ENP()

Some of above subroutine are either not fully developed or used so they might be ignored ( e.g. NSHAPE, DSHAPE,GAUSS, FSOL,BSOL)

**Output of the program**

* Output of the program, ie solution for U is stored in “OUTPUT\_FEM.TXT”.
* Following is the example when we selected the parameter same as shown in above description, which are

|  |  |  |
| --- | --- | --- |
|  | **Physical interpretation** | **Representation in code** |
| Differential eqn | a=1, b=-3,c=2 | ACOEF=1,BCOEF=-3,CCOEF=2 |
| Domain Length | 0<x<1 | X0=0, X1=1 |
|  | | |
| No Of Element | No of pieces into which domain is to break | NEL=10 |
| No Of Node Per Element | Linear Element | NNEL=2 |
| Degree Of Freedom |  | NDOF=1 |

Following parameters are to be set according to above paramers

NNODE=11 No Of Nodes In System (for linear) = NEL+1

EDOF=1 Element Degree Of Freedom (DOF)

SDOF Total DOF of System (abbr. SDOF) = NNODE\*NDOF

A sample of the output file OUTPUT\_FEM.TXT for above parameters is given below

|  |
| --- |
| NODE NO X U U EXACT DIFFERENCE  1 0.000 0.000 0.000 0.000  2 0.100 -0.040 -0.031 -0.009  3 0.200 -0.080 -0.061 -0.019  4 0.300 -0.116 -0.088 -0.028  5 0.400 -0.146 -0.111 -0.035  6 0.500 -0.164 -0.128 -0.036  7 0.600 -0.165 -0.136 -0.029  8 0.700 -0.143 -0.131 -0.011  9 0.800 -0.092 -0.111 0.019  10 0.900 -0.005 -0.069 0.064  11 1.000 0.000 0.000 0.000 |

**PROGRAM**

|  |
| --- |
| ! --------------------------------------------------  ! Free Format FTN95 Source File  ! --------------------------------------------------  PROGRAM FEM  IMPLICIT NONE  ! 1D DIFFERENTIAL EQN IS:: a\*D''u+ b\*D'u + c\*u= f(x)  ! BC are : u(0)=0 and u(L)=0  REAL , ALLOCATABLE :: COORD(:), TOPO(:,:), BC(:,:), BCV(:,:)  ! COORDINATE, TOPOLOGY, BOUNDARY CONDITION, BC VALUE  !ABOVE VAR ARE GLOBAL VARIABLE AS THEY DON'T NEED TO BE CHANGED FURTHER ONCE ASSIGNED  REAL , ALLOCATABLE :: K(:,:),F(:,:) ! SET UP DIFFERENT ELEMENT AND OTHER MATRICES  REAL , ALLOCATABLE :: KK(:,:),FF(:,:), U(:,:) ! SET UP GLOBAL MATRICES  INTEGER,ALLOCATABLE :: NE(:) , INDEX(:) ! NODAL NO OF ITH ELEMENT  REAL ,ALLOCATABLE :: XE(:) !COORDINATE OF ITH ELEMENT  INTEGER,PARAMETER :: NEL=10 !NO OF ELEMENT  INTEGER,PARAMETER :: NNEL=2 !NO OF NODE PER ELEMENT  INTEGER,PARAMETER :: NDOF=1 !DOF  INTEGER,PARAMETER :: NNODE=11 !NO OF NODES IN SYSTEM  INTEGER :: SDOF,EDOF !TOTAL SYSTEM DOF = NNODE\*NDOF  REAL , PARAMETER :: X0=0,X1=1 !DOMAIN LENGTH XA AND XB  REAL , PARAMETER :: ACOEF=1,BCOEF=-3,CCOEF=2 !DIFFERENTIAL EQN COFF  INTEGER :: I,J,OK,TEMP !OTHER  PRINT\*, "\*\*\*\*\*\*\*\*\*\*\*\*\*\* STARTING FEM MODULE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  ALLOCATE (COORD(NNODE), TOPO(NNODE,NNEL), BC(NNODE,NDOF), BCV(NNODE,NDOF), STAT=OK)  IF(OK/=0) THEN  PRINT\*, 'ALLOCATION IS NOT SUCCESSFUL, PROGRAM IS TERMINATED'  STOP  END IF  PRINT\*, "COORD, TOPO, BC, BCV ALLOCATED"  !SETUP GEOMETRY  CALL COORDINATE() ! TO FIND THE COORDINATE OF EACH NODE = COORD(:,:)  PRINT\*, "COORDNATE MATRIX DONE. COORDINATES ARE"  PRINT\*, COORD  CALL TOPOLOGY() ! TO FIND THE ELEMENT CONNECTIVITY TO NODES= TOPO(:,:)  PRINT\*, "TOPOLOGY MATRIX DONE. TOPOLOGY MATRIX IS"  PRINT\*, TOPO  CALL BOUNDARY\_CONDITION() ! TO FIND THE BC WHICH WILL BE IMPOSED LATER ON EQN DURING ASSEMBLY= BC(:,:)  PRINT\*, "BC DONE, BC ARE"  PRINT\*, BC  PRINT\*, "BCV DONE, BC ARE"  PRINT\*, BCV  SDOF = NNODE\*NDOF !COMPUTE THE DOF ASSOCIATED TO WHOLE SYSTEM  EDOF = NNEL\* NDOF !COMPUTE THE DOF ASSOCIATED TO EACH ELEMENT  PRINT\*, "SDOF=",SDOF, " EDOF=", EDOF  !ALLOCATION AND INITIALIZATION OF MARTICES AND VECTORS  ALLOCATE(NE(NNEL),XE(NNEL), STAT = OK)  IF (OK /= 0) THEN  PRINT \*, 'ALLOCATION OF NE, XE IS NOT SUCCESSFUL, PROGRAM IS TERMINATED'  STOP  END IF  ALLOCATE (FF(SDOF,1), KK(SDOF,SDOF),INDEX(EDOF),STAT = OK)  IF (OK /= 0) THEN  PRINT \*, 'ALLOCATION IS NOT SUCCESSFUL, PROGRAM IS TERMINATED'  STOP  END IF  FF=0  KK=0  INDEX=0  PRINT\*,"\*\*\*\*\*\*\*\*\*\*\* ASSEMBLY PROCESS STARTS \*\*\*\*\*\*\*\*\*\*\*\*"  DO I=1,NEL  WRITE(\*,\*)  PRINT\*,"FOR THE ELEMENT = ", I  PRINT\*," \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  NE=TOPO(I,:) !FETCH ALL NODES NO LYING WITHIN ELEMENT I  DO J=1,NNEL  TEMP=NE(J)  XE(J)=COORD(TEMP) !FIND THE COORDINATE OF ALL THOSE NODES  END DO  WRITE(\*,\*)  PRINT\*, " ELEMEMNT COORDINATE FETCHED ", XE  ! SET UP ELEMENTAL MATRICES  CALL ELEMENT(I,XE) !CALCULATE STIFFNESS MATRIX K AND F  !COMPUTATION OF ELEMENT MATRIX AND VECTOR AND THEIR ASSEMBLY  WRITE(\*,\*)  PRINT\*, " ELEMENT CALCULATED K IS", K  WRITE(\*,\*)  PRINT\*, " ELEMENT CALCULATED F IS", F  !CALL FOR INDEXING  CALL SYS\_DOF(I) !RETURN "INDEX" OF SYSTEM DOF ASSOCIATED WITH ELEMENT I  WRITE(\*,\*)  PRINT\*, "PRINT THE INDEX MATRIX",INDEX  !CALL FOR ASSEMBLEY OF MATRIX  CALL ASSEMBLE(K,F)  WRITE(\*,\*)  PRINT\*, "STIFNESS MATIRX K IS",  PRINT 10, KK    WRITE(\*,\*)  PRINT\*, "RHS MATRIX F IS",  PRINT 10, FF  END DO  PRINT\*, "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  WRITE(\*,\*)  PRINT\*, "GLOBAL MATRICES HAVE BEEN OBTAINED"  PRINT\*, "STIFNESS MATIRX K IS",  PRINT 10, KK  PRINT\*, "RHS MATRIX F IS"  PRINT\*, FF  WRITE(\*,\*)  10 FORMAT(5X, 6F10.3)  CALL APP\_BC() !APPLY BC ON GLOBAL K AND GLOBAL F MATRIX  PRINT\*, "APPLIED BOUNDARY CONDITIONS, GLOBAL MATRICES ARE"  PRINT\*, "STIFNESS MATIRX K IS",  PRINT 10, TRANSPOSE(KK)  PRINT\*, "RHS MATRIX F IS"  PRINT\*, FF  WRITE(\*,\*)  ALLOCATE(U(SIZE(KK,1),1))  U=0  PRINT\*, "SOL MATRIX U ALLOTED AND INITIALIZED"  WRITE(\*,\*)  PRINT\*, "SOLVE FOR K\*U=F"  CALL SOLVE(U) !SOLVE KK\*U=FF FOR U  CALL Enp()!CALCULATE THE EXACT SOLUTION AND ERROR  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CONTAINS  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  FUNCTION RHS\_FUN(X)  REAL , INTENT(IN) :: X  REAL :: RHS\_FUN  RHS\_FUN=1  END FUNCTION RHS\_FUN  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE COORDINATE()  INTEGER :: I  REAL ::L  L= (X1-X0)/NEL  DO I=1,NNODE !LOOP FOR FINDING THE COORDINATE OF EACH NODE  COORD(I)= X0+L\*(I-1)  END DO  END SUBROUTINE COORDINATE  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE TOPOLOGY() ! TO FIND THE NODES CONNECTED TO THE RESPECTIVE ELEMENT  INTEGER:: I,J  TOPO=0  DO I=1,NEL  DO J=1,NNEL  TOPO(I,J) = J + (I-1)\*(NNEL - 1) !NODE CONNECTED TO LEFT OF Ith ELEMENT  ! TOPO(I,J) = I+1 !NODE CONNECTED TO RIGHT OF Ith ELEMENT  END DO  END DO  END SUBROUTINE TOPOLOGY  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE BOUNDARY\_CONDITION()  INTEGER:: N  N=NNODE\*NDOF  BC=0  BCV=0  BC(1,1)=1 !SET THE FLAG FOR THE NODE AT WHICH BC ARE SPECIFIED  BC(N,1)=1 !SET THE FLAG  BCV(1,1)=0 !VALUE CORROSPONDING TO SPECIFIED NODE AND DOF  BCV(N,1)=0  END SUBROUTINE BOUNDARY\_CONDITION  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE SYS\_DOF(IEL)  INTEGER, INTENT(IN) :: IEL  INTEGER :: START, I    START= (IEL-1)\*(NNEL-1)\*NDOF !INXEXING START WITH THIS FOR FIRST UNK OF Ith ELEMENT  DO I=1,EDOF  INDEX(I)= START+I  END DO  END SUBROUTINE SYS\_DOF  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE ELEMENT(I,XE) !for Ith element, WE ARE COMPUTING K OR STIFFNES MATRIX  REAL , INTENT(IN) :: XE(:)  INTEGER, INTENT(IN) :: I  REAL , ALLOCATABLE :: M1(:,:),M2(:,:),M3(:,:),GP(:),W(:), N(:,:),B(:,:)  REAL :: H,JACOB,XT  INTEGER :: NGP, DIMEN, J, SELECTION    JACOB =(XE(1)-XE(NNEL))/2 !COMPUTE JACOBIAN  IF (NNEL==2)THEN ! DEFAULT ELEMENT SPECIFICATION  H=XE(1)-XE(NNEL)  IF (I==1)THEN  ALLOCATE(M1(2,2),M2(2,2),M3(2,2),K(2,2),F(2,1))  M1 = reshape((/1, -1, -1, 1 /),(/2,2/))  M2 = reshape((/-1, 1, -1, 1/),(/2,2/))  M3 = reshape((/2, 1, 1, 2 /),(/2,2/))  K= -(ACOEF/H)\*M1+ (BCOEF/2)\*M2 + (CCOEF\*H/2)\*M3  F= RESHAPE((/1, 1 /),(/2,1/))  F= H/2\*F  END IF  ELSE IF (NNEL/=2)THEN !COMPUTING THE ELEMENT SPECIFICLY  NGP=NNEL  IF(I==1)THEN  ALLOCATE(GP(NNEL),W(NNEL))  ALLOCATE(N(NNEL,1),B(NNEL,1),STAT=OK) !ALLOCATE SHAPE AND DERIVATIVE MATRIX ITS SPACE  END IF  CALL GAUSS(NGP,GP,W)    DO J=1,NGP !LOOP OVER N GAUSSIAN POINTS FOR INTEGRATION  XT=JACOB\*GP(J)+ (XE(1)+XE(NNEL))/2 !COMPUTE GAUSS POINT IN PHYSICAL COORDINATE  CALL NSHAPE(XT,XE,N)  CALL DSHAPE(XT,XE,N)  !NOW CONCIDERING N,B ARE COLOUMN VECTOR, FIND THE DIMENTION OF THE ELEMENT MATRIX  DIMEN = SIZE(N,1)  ALLOCATE(K(DIMEN,DIMEN),F(DIMEN,1), STAT=OK)  K= -ACOEF\*MATMUL(B,TRANSPOSE(B))+ BCOEF\*MATMUL(B,TRANSPOSE(N))+ CCOEF\*MATMUL(N,TRANSPOSE(N))  K= W(J)\*K ! MULTIPLY THE FUNCTION TO RESPECTIVE WEIGHT  F=RHS\_FUN(XT)\*N  END DO  K=K\*JACOB  F=F\*JACOB  END IF  END SUBROUTINE ELEMENT  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE NSHAPE(X,P,N) ! SHAPE FUNCTION  REAL ,INTENT(IN) :: X,P(:)  REAL , INTENT(OUT) :: N(:,:)    IF(NNEL==2) THEN  N(1,1)=(X-P(2))/(P(1)-P(2))  N(2,1)=(X-P(1))/(P(2)-P(1))  ELSE IF (NNEL==3)THEN  N(1,1)=(X-P(2))\*(X-P(3))/((P(1)-P(2))\*(P(1)-P(3)))  N(2,1)=(X-P(1))\*(X-P(3))/((P(2)-P(1))\*(P(2)-P(3)))  N(3,1)=(X-P(1))\*(X-P(2))/((P(3)-P(1))\*(P(3)-P(2)))  END IF  END SUBROUTINE NSHAPE  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE DSHAPE(X,P,B) !DERIVATIVE OF SHAPE FUNCTION  REAL , INTENT(IN) :: X,P(:)  REAL , INTENT(OUT) :: B(:,:)    IF (NNEL==2) THEN  B(1,1)=-1/(P(2)-P(1))  B(2,1)=1/(P(2)-P(1))  ELSEIF (NNEL==3)THEN  B(1,1)=(2\*X-P(2)-P(3))/((P(1)-P(2))\*(P(1)-P(3)))  B(2,1)=(2\*X-P(1)-P(3))/((P(2)-P(1))\*(P(2)-P(3)))  B(3,1)=(2\*X-P(1)-P(2))/((P(3)-P(1))\*(P(3)-P(2)))  END IF  END SUBROUTINE DSHAPE  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE GAUSS(NGP,GP,W) !COMPUTE THE GASUSSIAN POINTS AND WEIGHTS  REAL , INTENT(OUT) :: GP(:),W(:)  INTEGER, INTENT(IN) :: NGP  IF (NGP==1)THEN  GP(1)=0  W(1)=2  ELSEIF (NGP==2)THEN  GP = (/-0.5773502691896257,0.5773502691896257/)  W = (/1,1/)  ELSEIF (NGP==3)THEN  GP=(/-0.7745966692,0.7745966692,0.0/)  W=(/.5555555556, -.5555555556, 0.8888888889/)  END IF  END SUBROUTINE GAUSS  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE ASSEMBLE(K,F)  REAL , INTENT(IN) :: K(:,:),F(:,:)  INTEGER :: I,J,II,JJ  DO I = 1,EDOF  II=INDEX(I)  FF(II,1)=FF(II,1)+ F(I,1)  DO J=1,EDOF  JJ = INDEX(J)  KK(II,JJ) = KK(II,JJ) + K(I,J)  END DO  END DO  END SUBROUTINE ASSEMBLE  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE APP\_BC()  INTEGER :: KSIZE,I    KSIZE=SIZE(KK,1)  DO I=1,KSIZE  IF(BC(I,1)==1)THEN  KK(I,1:KSIZE)=0  KK(I,I)= 1  FF(I,1)= BCV(I,1)  END IF  END DO  END SUBROUTINE APP\_BC  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE SOLVE(U)  REAL , INTENT(OUT) :: U(:,:)  REAL , ALLOCATABLE :: V(:,:), L(:,:),D1(:),D2(:),D3(:)  REAL :: LSUM1, LSUM2  INTEGER :: I,J,N,SELECTION, OK    N=SIZE(KK,1)  PRINT\*,N  SELECTION=2  IF (SELECTION==1) THEN  ALLOCATE(L(N,N),V(N,1), STAT = OK)  PRINT\*,"ALLOCATED SPACE FOR L MATRIX"  !MATRIX DECOMPOSITION OF KK  !IT TAKES KK MATRIX AND DECOMPOSE IT INTO L\*L'. HERE L'= TRANSPOSE OF L  !OUTPUT OF FOLLOWING CODE LOOP IS L MATRIX  L=0  DO I=1,N !ROW NO  DO J=1,I !COLOUMN NO  PRINT\*, "LOOP NO ",I,J  IF(I==1.AND.J==1) THEN  L(I,J)= SQRT(K(1,1))  ELSEIF (J==1) THEN  L(I,1) = KK(I,1)/L(1,1)  ELSEIF (I==J) THEN  LSUM1 = DOT\_PRODUCT(L(I,1:I-1),L(I,1:I-1))  PRINT\*, ".....FLAG....",KK(I,J)-LSUM1    L(I,J)=SQRT(KK(I,J)-LSUM1)  ELSE  LSUM2 = DOT\_PRODUCT(L(I,1:J-1),L(J,1:J-1))  L(I,J)=(KK(I,J)-LSUM2)/L(J,J)  END IF  END DO  END DO  PRINT\*,"COMPUTED L MATRIX IS"  PRINT\*, L    ALLOCATE(V(N,1), STAT=OK)  PRINT\*,"ALLOCATED SPACE FOR V (INTERMEDIATE SOL)"  !\*\*\*\*\*\*SOLUTION OF THE EQUATION BY FORWARD AND BACKWARD SUBSTITUTION \*\*\*\*\*\*  CALL FSOL(V,L,FF) !FOR LOWER TRIANGUALR MATRIX, SOLUTION(U) L(NxN)\*U = FF(Nx1)  WRITE(\*,\*)  PRINT\*,"COMPUTED SOL FOR V IS"  PRINT\*,V  L=TRANSPOSE(L) !CHANGING L to upper triangual form  PRINT\*,"TRANSPOSE OF L MATRIX"  PRINT\*,L  CALL BSOL(U,L,V) ! FOR UPPER TRIANGULAR MATRIX SOLUTION (U) L\*U=U' HERE U'=U EARLIER OBTAINED  ELSE IF (SELECTION==2)THEN  PRINT\*, "SOL FOR TRIDIGONAL MATRIX APPROACH OPTED"  ALLOCATE(D1(N),D2(N),D3(N))  D1=0  D2=0  D3=0  PRINT\*, "THREE DIAGONALS ARE"  DO I=1,N-1  D1(I)=KK(I+1,I)  D2(I)=KK(I,I)  D3(I)=KK(I,I+1)  END DO  D2(N)=KK(N,N)  print\*, D1  print\*, D2  print\*, D3  CALL solve\_tridiag(D1,D2,D3,FF,U,N)  END IF    PRINT\*,"FINAL COMPUTED SOL FOR U"  PRINT\*,U  END SUBROUTINE SOLVE  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE SOLVE\_TRIDIAG(A,B,C,D,X,N)  IMPLICIT NONE  ! A - SUB-DIAGONAL (MEANS IT IS THE DIAGONAL BELOW THE MAIN DIAGONAL)  ! B - THE MAIN DIAGONAL  ! C - SUP-DIAGONAL (MEANS IT IS THE DIAGONAL ABOVE THE MAIN DIAGONAL)  ! D - RIGHT PART  ! X - THE ANSWER  ! N - NUMBER OF EQUATIONS    INTEGER,INTENT(IN) :: N  REAL ,DIMENSION(N),INTENT(IN) :: A,B,C,D  REAL ,DIMENSION(N),INTENT(OUT) :: X  REAL ,DIMENSION(N) :: CP,DP  REAL :: M  INTEGER I    ! INITIALIZE C-PRIME AND D-PRIME  CP(1) = C(1)/B(1)  DP(1) = D(1)/B(1)  ! SOLVE FOR VECTORS C-PRIME AND D-PRIME  DO I = 2,N  M = B(I)-CP(I-1)\*A(I)  CP(I) = C(I)/M  DP(I) = (D(I)-DP(I-1)\*A(I))/M  ENDDO  ! INITIALIZE X  X(N) = DP(N)  ! SOLVE FOR X FROM THE VECTORS C-PRIME AND D-PRIME  DO I = N-1, 1, -1  X(I) = DP(I)-CP(I)\*X(I+1)  END DO    END SUBROUTINE SOLVE\_TRIDIAG  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE FSOL(V,L,FF) !FOR UPPER TRAINGUALR L(NxN)\*U(Nx1)=Y(Nx1)  REAL , INTENT(IN) :: L(:,:),FF(:,1)  REAL , INTENT(OUT) :: V(:,1)  REAL :: SUM  INTEGER :: I,N  N= SIZE(L,1)  SUM=0  I=1  DO  V(I,1)=(FF(I,1)-SUM)/L(I,I)  I=I+1  IF (I>N) EXIT  SUM = DOT\_PRODUCT(L(I,1:I-1),V(1:I-1,1))  END DO  END SUBROUTINE FSOL  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE BSOL(U,LT,V) !FOR LOWER TRAINGUALR L'(NxN)\*U(Nx1)=Y(Nx1)  REAL , INTENT(IN) :: LT(:,:),V(:,1) !LT= L' = UPPER TRIANGULAR MATRIX  REAL , INTENT(OUT) :: U(:,1)  INTEGER :: I,N  REAL :: SUM  N=SIZE(LT)  SUM=0  I=N  DO  U(I,1)=(V(I,1)-SUM)/LT(I,I)  I=I-1  IF (I>N) EXIT  SUM = DOT\_PRODUCT(LT(I,I:N),U(I:N,1))  END DO  END SUBROUTINE BSOL  !\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  SUBROUTINE ENP()  REAL ,ALLOCATABLE :: UE(:)  REAL :: C1,C2,C3, X,DIFF, PERROR  INTEGER :: I,N  N=SIZE(U,1)  ALLOCATE(UE(N))  C1= 0.5/EXP(1.0)  C2= -0.5\*(1 + EXP(1.0))/EXP(1.0)  C3= 0.5  WRITE(\*,50)"NODE NO","X","U","U EXACT", "DIFFERENCE"  OPEN(999,FILE= "OUTPUT\_FEM.TXT")  WRITE(999,50)"NODE NO","X","U","U EXACT", "DIFFERENCE",  50 FORMAT(A10,5A12)  DO I=1,NNODE  X=COORD(I)  UE(I)=C1\*EXP(2\*X) + C2\*EXP(X) + C3  DIFF=U(I,1)-UE(I)  PERROR= DIFF\*100.0/UE(I)  WRITE(\*,100)I, X,U(I,1), UE(I), DIFF  WRITE(999,100)I,X, U(I,1), UE(I), DIFF  END DO  100 FORMAT(I10,5F12.3)  CLOSE(999)  END SUBROUTINE ENP  END PROGRAM FEM |