MODULE MESHDATA

!

! MODULE STORING DATA DEFINING THE MESH.

!

REAL :: X(10000),Y(10000),AI(20000),AJ(20000),AK(20000)

REAL :: BI(20000),BJ(20000),BK(20000),AREA(20000)

INTEGER :: NEL,NNP,NBP,MAXNEL,MAXNNP,MAXNBP

INTEGER :: NPI(20000),NPJ(20000),NPK(20000),NPB(400)

INTEGER :: NPTS1,NPTS2,NPTS3,NPTS4

END MODULE MESHDATA

MODULE EQNSDATA

!

! MODULE STORING EQUATION DATA: RECTANGULARISED OVERALL STIFFNESS

! MATRIX, FORCE VECTOR AND SOLUTION VECTOR.

!

REAL :: OKXX(10000,13),OKXY(10000,13),OKYX(10000,13),

& OKYY(10000,13),U(10000),V(10000),FX(10000),FY(10000),

& FXMOD(10000),FYMOD(10000),SFXX(10000),SFXY(10000),

& SFYX(10000),SFYY(10000)

INTEGER :: NPA(10000,13),NAP(10000)

END MODULE EQNSDATA

MODULE MATLDATA

!

! MODULE STORING MATERIAL DATA.

!

REAL :: E(10),NU(10),ALPHA(10),RHOG(10)

INTEGER :: NMAT,MATM(20000)

END MODULE MATLDATA

MODULE LOADDATA

!

! MODULE STORING LOAD DATA.

!

REAL :: DELTAT(20000),XBAR(20000),YBAR(20000)

END MODULE LOADDATA

MODULE RESTDATA

!

! MODULE STORING BOUNDARY CONDITION RESTRAINT DATA.

!

REAL :: TANG(400),UPRES(400),VPRES(400)

INTEGER :: NCOND(400),NBC3P

END MODULE RESTDATA

MODULE EMODDATA

!

! MODULE STORING ELEMENT MODULUS DATA FOR A NONLINEAR PROBLEM.

!

REAL :: EE(20000)

END MODULE EMODDATA

PROGRAM PLANE

!

! PROGRAM FOR FINITE ELEMENT ANALYSIS OF TWO-DIMENSIONAL PROBLEMS OF

! THE BIHARMONIC PLANE STRAIN OR PLANE STRESS TYPE, USING CONSTANT

! STRAIN TRIANGULAR ELEMENTS.

!

USE MESHDATA

USE EQNSDATA

USE MATLDATA

USE LOADDATA

REAL :: B(3,6),D(3,3),BTD(6,3),ESTIFF(6,6),ET(3),THETAM(6)

INTEGER :: IJK(3)

CHARACTER(80) :: TITLE

CHARACTER(6) :: CASE

OPEN(5,FILE="DATA")

OPEN(6,FILE="RESULTS")

OPEN(7,FILE="MESHRES")

!

! DEFINE MAXIMUM PROBLEM SIZE PERMITTED BY THE ARRAY DIMENSIONS.

MAXNEL=20000

MAXNNP=10000

MAXNBP=400

!

! INPUT THE PROBLEM TITLE AND TYPE.

READ(5,FMT="(A80)") TITLE

READ(5,FMT="(A6)") CASE

WRITE(6,61) CASE,TITLE

61 FORMAT("CST FINITE ELEMENT SOLUTION FOR PLANE ",A," PROBLEM"

& // A)

!

! INPUT OR GENERATE THE MESH DATA, MATERIAL PROPERTIES, TEMPERATURE

! CHANGES AND BODY FORCES.

CALL MESH

CALL MODIFY

CALL MATLS

CALL TEMPS

CALL BODYF

!

! COMPUTE THE ELEMENT GEOMETRIES.

Each element in turn: DO M=1,NEL

I=NPI(M)

J=NPJ(M)

K=NPK(M)

AI(M)=-X(J)+X(K)

AJ(M)=-X(K)+X(I)

AK(M)=-X(I)+X(J)

BI(M)=Y(J)-Y(K)

BJ(M)=Y(K)-Y(I)

BK(M)=Y(I)-Y(J)

AREA(M)=0.5\*(AK(M)\*BJ(M)-AJ(M)\*BK(M))

IF(AREA(M) <= 0.) THEN

WRITE(6,62) M

62 FORMAT(/"ELEMENT NUMBER",I6," HAS NEGATIVE AREA - STOP")

STOP

END IF

END DO Each element in turn

!

! OUTPUT THE MESH DATA.

CALL MSHOUT

!

! SET INITIAL VALUES OF STIFFNESSES, EXTERNAL FORCES AND UNKNOWNS.

Each overall row in turn: DO IROW=1,NNP

Each overall column in turn: DO IC=1,13

OKXX(IROW,IC)=0.

OKXY(IROW,IC)=0.

OKYX(IROW,IC)=0.

OKYY(IROW,IC)=0.

NPA(IROW,IC)=0

END DO Each overall column in turn

NPA(IROW,1)=IROW

FXMOD(IROW)=0.

FYMOD(IROW)=0.

U(IROW)=0.

V(IROW)=0.

END DO Each overall row in turn

!

! MODIFY MATERIAL PROPERTIES IF CASE IS ONE OF PLANE STRAIN.

IF(CASE == "STRAIN") THEN

Each material in turn: DO MAT=1,NMAT

E(MAT)=E(MAT)/(1.-NU(MAT)\*\*2)

ALPHA(MAT)=ALPHA(MAT)\*(1.+NU(MAT))

NU(MAT)=NU(MAT)/(1.-NU(MAT))

END DO Each material in turn

END IF

!

! SET UP THE OVERALL ASSEMBLY LOOP.

Each element in turn: DO M=1,NEL

!

! STORE THE ELEMENT NODE NUMBERS IN ORDER IN ARRAY IJK.

IJK(1)=NPI(M)

IJK(2)=NPJ(M)

IJK(3)=NPK(M)

!

! COMPUTE THE BODY FORCE COMPONENTS ON EACH NODE OF THE ELEMENT.

GXM=XBAR(M)\*AREA(M)/3.

GYM=YBAR(M)\*AREA(M)/3.

!

! FORM THE ELEMENT DIMENSION MATRIX.

B(1,1)=BI(M)

B(1,2)=0.

B(1,3)=BJ(M)

B(1,4)=0.

B(1,5)=BK(M)

B(1,6)=0.

B(2,1)=0.

B(2,2)=AI(M)

B(2,3)=0.

B(2,4)=AJ(M)

B(2,5)=0.

B(2,6)=AK(M)

Each element column in turn: DO ICE=1,6

IF(MOD(ICE,2) == 0) B(3,ICE)=B(1,ICE-1)

IF(MOD(ICE,2) == 1) B(3,ICE)=B(2,ICE+1)

END DO Each element column in turn

!

! FORM THE ELASTIC PROPERTY MATRIX.

MAT=MATM(M)

FACT=E(MAT)/(1.-NU(MAT)\*\*2)

D(1,1)=FACT

D(1,2)=FACT\*NU(MAT)

D(1,3)=0.

D(2,1)=D(1,2)

D(2,2)=D(1,1)

D(2,3)=0.

D(3,1)=0.

D(3,2)=0.

D(3,3)=FACT\*0.5\*(1.-NU(MAT))

!

! MULTIPLY THE TRANSPOSE OF MATRIX B BY MATRIX D.

Each element row in turn: DO IRE=1,6

Each element column in turn: DO ICE=1,3

BTD(IRE,ICE)=0.

Sum products: DO ISUM=1,3

BTD(IRE,ICE)=BTD(IRE,ICE)+B(ISUM,IRE)\*D(ISUM,ICE)

END DO Sum products

END DO Each element column in turn

END DO Each element row in turn

!

! FORM THE THERMAL STRAIN AND THERMAL FORCE VECTORS.

ET(1)=ALPHA(MAT)\*DELTAT(M)

ET(2)=ET(1)

ET(3)=0.

Each element row in turn: DO IRE=1,6

SUM=0.

Sum products: DO ISUM=1,3

SUM=SUM+BTD(IRE,ISUM)\*ET(ISUM)

END DO Sum products

THETAM(IRE)=0.5\*SUM

END DO Each element row in turn

!

! FORM THE ELEMENT STIFFNESS MATRIX.

Each element row in turn: DO IRE=1,6

Each element column in turn: DO ICE=1,6

SUM=0.

Sum products: DO ISUM=1,3

SUM=SUM+BTD(IRE,ISUM)\*B(ISUM,ICE)

END DO Sum products

ESTIFF(IRE,ICE)=0.25\*SUM/AREA(M)

END DO Each element column in turn

END DO Each element row in turn

!

! ADD ELEMENT STIFFNESS TO OVERALL STIFFNESS.

Each element row in turn: DO IRE=1,3

Each element column in turn: DO ICE=1,3

IROW=IJK(IRE)

ICOL=IJK(ICE)

!

! STORE STIFFNESS COEFFICIENTS IN RECTANGULAR FORM OF OVERALL MATRICES.

IFLAG=0

Each overall column in turn: DO IC=1,13

IF(NPA(IROW,IC) == 0) THEN

NPA(IROW,IC)=ICOL

NAP(IROW)=IC

END IF

IF(NPA(IROW,IC) == ICOL) THEN

OKXX(IROW,IC)=OKXX(IROW,IC)+ESTIFF(2\*IRE-1,2\*ICE-1)

OKXY(IROW,IC)=OKXY(IROW,IC)+ESTIFF(2\*IRE-1,2\*ICE)

OKYX(IROW,IC)=OKYX(IROW,IC)+ESTIFF(2\*IRE,2\*ICE-1)

OKYY(IROW,IC)=OKYY(IROW,IC)+ESTIFF(2\*IRE,2\*ICE)

IFLAG=1

EXIT

END IF

END DO Each overall column in turn

IF(IFLAG == 0) THEN

WRITE(6,63) IROW

63 FORMAT(/"NODE ",I6," HAS MORE THAN 12 ADJACENT NODES - STOP")

STOP

END IF

!

END DO Each element column in turn

END DO Each element row in turn

!

! ASSEMBLE THE EXTERNAL FORCES ON THE NODES.

Each element row in turn: DO IRE=1,3

IROW=IJK(IRE)

FXMOD(IROW)=FXMOD(IROW)+GXM+THETAM(2\*IRE-1)

FYMOD(IROW)=FYMOD(IROW)+GYM+THETAM(2\*IRE)

END DO Each element row in turn

!

END DO Each element in turn

!

! COMPUTE THE SELF-FLEXIBILITY SUBMATRICES.

Each node in turn: DO I=1,NNP

DENOM=OKXX(I,1)\*OKYY(I,1)-OKXY(I,1)\*OKYX(I,1)

SFXX(I)=OKYY(I,1)/DENOM

SFXY(I)=-OKXY(I,1)/DENOM

SFYX(I)=-OKYX(I,1)/DENOM

SFYY(I)=OKXX(I,1)/DENOM

END DO Each node in turn

!

! APPLY THE BOUNDARY CONDITIONS.

CALL BCS

!

! SOLVE THE LINEAR EQUATIONS.

CALL SOLVE2

!

! OUTPUT THE REQUIRED RESULTS.

CALL OUTPUT

STOP

END PROGRAM PLANE

SUBROUTINE MESH

!

! SUBPROGRAM TO READ OR GENERATE A MESH OF TRIANGULAR FINITE ELEMENTS.

! THIS VERSION READS IN THE NECESSARY DATA.

!

USE MESHDATA

!

! INPUT THE NUMBERS OF NODES AND ELEMENTS.

READ(5,\*) NNP,NEL

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,61) NNP,NEL

61 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! INPUT THE NODAL POINT CO-ORDINATES.

READ(5,\*) (I,X(I),Y(I),N=1,NNP)

!

! INPUT THE ELEMENT NODE DATA.

READ(5,\*) (M,NPI(M),NPJ(M),NPK(M),N=1,NEL)

RETURN

END SUBROUTINE MESH

SUBROUTINE MESH

!

! SUBPROGRAM TO READ OR GENERATE A MESH OF TRIANGULAR FINITE ELEMENTS.

! THIS VERSION GENERATES A SQUARE MESH OF RIGHT-ANGLED TRIANGLES.

!

USE MESHDATA, NXPT=>NPTS1, NYPT=>NPTS2

!

! INPUT AND STORE THE NUMBERS OF POINTS REQUIRED IN THE X AND Y

! DIRECTIONS.

READ(5,\*) NXPT,NYPT

!

! COMPUTE AND TEST THE NUMBERS OF NODES AND ELEMENTS.

NNP=NXPT\*NYPT

NEL=(NXPT-1)\*(NYPT-1)\*2

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,61) NNP,NEL

61 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! DEFINE THE NODAL POINT CO-ORDINATES.

Each horizontal row of nodes in turn: DO IY=1,NYPT

Each node along the row in turn: DO IX=1,NXPT

I=(IY-1)\*NXPT+IX

X(I)=FLOAT(IX-1)/FLOAT(NXPT-1)

Y(I)=FLOAT(IY-1)/FLOAT(NYPT-1)

END DO Each node along the row in turn

END DO Each horizontal row of nodes in turn

!

! DEFINE THE NUMBERS OF THE THREE NODES OF EACH ELEMENT.

NXEL=NXPT-1

NYEL=NYPT-1

Each horizontal row of elements in turn: DO IY=1,NYEL

Each pair of elements along the row in turn: DO IX=1,NXEL

NSQ=(IY-1)\*NXEL+IX

M1=NSQ\*2-1

M2=M1+1

I=(IY-1)\*NXPT+IX

NPI(M1)=I

NPJ(M1)=I+NXPT+1

NPK(M1)=I+NXPT

NPI(M2)=I

NPJ(M2)=I+1

NPK(M2)=I+1+NXPT

END DO Each pair of elements along the row in turn

END DO Each horizontal row of elements in turn

RETURN

END SUBROUTINE MESH

SUBROUTINE MESH

!

! SUBPROGRAM TO READ OR GENERATE A MESH OF TRIANGULAR FINITE ELEMENTS.

! THIS VERSION GENERATES A UNIFORM EQUILATERAL TRIANGULAR MESH.

!

USE MESHDATA, NSPT=>NPTS1

!

! INPUT AND STORE THE NUMBER OF POINTS ON EACH SIDE OF THE MESH.

READ(5,\*) NSPT

!

! COMPUTE AND TEST THE NUMBERS OF NODES AND ELEMENTS.

NNP=NSPT\*(NSPT+1)/2

NEL=(NSPT-1)\*\*2

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,61) NNP,NEL

61 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! DEFINE THE NODAL POINT CO-ORDINATES.

HX=1./FLOAT(NSPT-1)

HY=HX\*0.5\*SQRT(3.)

I=0

Each horizontal row of nodes in turn: DO IY=1,NSPT

NXPT=NSPT-IY+1

Each node along the row in turn: DO IX=1,NXPT

I=I+1

X(I)=FLOAT(IX-1)\*HX+FLOAT(IY-1)\*0.5\*HX

Y(I)=FLOAT(IY-1)\*HY

END DO Each node along the row in turn

END DO Each horizontal row of nodes in turn

!

! DEFINE THE NUMBERS OF THE THREE NODES OF EACH ELEMENT,

! FIRST FOR THE UPWARD POINTING ELEMENTS.

M=0

NYEL=NSPT-1

Each horizontal row of elements in turn: DO IY=1,NYEL

NXEL=NSPT-IY

Each element along the row in turn: DO IX=1,NXEL

M=M+1

NPI(M)=M+IY-1

NPJ(M)=NPI(M)+1

NPK(M)=M+NSPT

END DO Each element along the row in turn

END DO Each horizontal row of elements in turn

!

! THEN FOR THE DOWNWARD POINTING ELEMENTS.

M1=M

NYEL=NYEL-1

Each horizontal row of elements in turn: DO IY=1,NYEL

NXEL=NSPT-IY-1

Each element along the row in turn: DO IX=1,NXEL

M=M+1

NPI(M)=M-M1+2\*IY-1

NPJ(M)=M-M1+NSPT+IY

NPK(M)=NPJ(M)-1

END DO Each element along the row in turn

END DO Each horizontal row of elements in turn

RETURN

END SUBROUTINE MESH

SUBROUTINE MESH

!

! SUBPROGRAM TO READ OR GENERATE A MESH OF TRIANGULAR FINITE ELEMENTS.

! THIS VERSION GENERATES A SQUARE MESH OF MAINLY ISOSCELES ELEMENTS.

!

USE MESHDATA, NXPT=>NPTS1, NYPT=>NPTS2

!

! INPUT AND STORE THE NUMBERS OF POINTS ALONG THE X AND Y AXES.

READ(5,\*) NXPT,NYPT

!

! COMPUTE AND TEST THE NUMBERS OF NODES AND ELEMENTS.

MODNY=MOD(NYPT,2)

IF(MODNY == 0) NNP=NYPT\*(2\*NXPT+1)/2

IF(MODNY == 1) NNP=(NYPT-1)\*(2\*NXPT+1)/2+NXPT

NEL=(NYPT-1)\*(2\*NXPT-1)

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,61) NNP,NEL

61 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! DEFINE THE NODAL POINT CO-ORDINATES.

I=0

Each horizontal row of nodes in turn: DO IY=1,NYPT

MODIY=MOD(IY,2)

Each node along the row in turn: DO IX=1,NXPT

I=I+1

X(I)=FLOAT(IX-1)/FLOAT(NXPT-1)

Y(I)=FLOAT(IY-1)/FLOAT(NYPT-1)

IF(MODIY == 0 .AND. IX > 1) X(I)=X(I)-0.5/FLOAT(NXPT-1)

END DO Each node along the row in turn

IF(MODIY == 0) THEN

I=I+1

Y(I)=Y(I-1)

X(I)=1.

END IF

END DO Each horizontal row of nodes in turn

!

! DEFINE THE NUMBERS OF THE THREE NODES OF EACH ELEMENT,

! FIRST FOR THE UPWARD POINTING ELEMENTS.

M=0

NYEL=NYPT-1

Each horizontal row of elements in turn: DO IY=1,NYEL

NXEL=NXPT-1

IF(MOD(IY,2) == 0) NXEL=NXPT

Each element along the row in turn: DO IX=1,NXEL

M=M+1

NPI(M)=M+IY-1

NPJ(M)=NPI(M)+1

NPK(M)=NPJ(M)+NXPT

END DO Each element along the row in turn

END DO Each horizontal row of elements in turn

!

! THEN FOR THE DOWNWARD POINTING ELEMENTS.

M1=M

Each horizontal row of elements in turn: DO IY=1,NYEL

NXEL=NXPT

IF(MOD(IY,2) == 0) NXEL=NXPT-1

Each element along the row in turn: DO IX=1,NXEL

M=M+1

NPI(M)=M-M1+IY-1

NPJ(M)=NPI(M)+NXPT+1

NPK(M)=NPJ(M)-1

END DO Each element along the row in turn

END DO Each horizontal row of elements in turn

RETURN

END SUBROUTINE MESH

SUBROUTINE MESH

!

! SUBPROGRAM TO READ OR GENERATE A MESH OF TRIANGULAR FINITE ELEMENTS.

! THIS VERSION GENERATES A CIRCULAR MESH.

!

USE MESHDATA, NCEL=>NPTS1, NRPT=>NPTS2

!

! INPUT NUMBER OF ELEMENTS AT CENTRE AND POINTS ALONG A RADIUS.

READ(5,\*) NCEL,NRPT

!

! COMPUTE AND TEST THE NUMBERS OF NODES AND ELEMENTS.

NNP=NCEL\*NRPT\*(NRPT-1)/2+1

NEL=NCEL\*(NRPT-1)\*\*2

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,61) NNP,NEL

61 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! DEFINE THE NODAL POINT CO-ORDINATES.

X(1)=0.

Y(1)=0.

PI=4.\*ATAN(1.)

I=1

NREL=NRPT-1

Each ring of nodes in turn: DO IR=1,NREL

R=FLOAT(IR)/FLOAT(NREL)

NTHPT=NCEL\*IR

Each node along the ring in turn: DO ITH=1,NTHPT

THETA=FLOAT(ITH-1)\*2.\*PI/FLOAT(NTHPT)

I=I+1

X(I)=R\*COS(THETA)

Y(I)=R\*SIN(THETA)

END DO Each node along the ring in turn

END DO Each ring of nodes in turn

!

! DEFINE THE NUMBERS OF THE THREE NODES OF EACH ELEMENT.

!

! INWARD POINTING ELEMENTS.

M=0

I=1

Each ring of inward pointing elements in turn: DO IR=1,NREL

NTHPT=NCEL\*IR

Each element along the ring in turn: DO ITH=1,NTHPT

M=M+1

IF(ITH == 1) NPI(M)=I

IF(ITH > 1) NPI(M)=NPI(M-1)+1

IF(ITH > 1 .AND. MOD(ITH-1,IR) == 0) NPI(M)=NPI(M-1)

NPJ(M)=M+1

NPK(M)=M+2

IF(ITH == 1) THEN

I=NPI(M)

K=NPJ(M)

END IF

END DO Each element along the ring in turn

NPI(M)=I

NPK(M)=K

I=K

END DO Each ring of inward pointing elements in turn

!

! OUTWARD POINTING ELEMENTS.

M1=M

J=NCEL+3

Each ring of outward pointing elements in turn: DO IR=2,NREL

NTHPT=NCEL\*(IR-1)

Each element along the ring in turn: DO ITH=1,NTHPT

M=M+1

NPI(M)=M-M1+1

IF(ITH == 1) NPJ(M)=J

IF(ITH > 1) NPJ(M)=NPJ(M-1)+1

IF(ITH > 1 .AND. MOD(ITH-1,IR-1) == 0) NPJ(M)=NPJ(M-1)+2

NPK(M)=NPI(M)+1

IF(ITH == 1) K=NPI(M)

END DO Each element along the ring in turn

NPK(M)=K

J=NPJ(M)+2

END DO Each ring of outward pointing elements in turn

RETURN

END SUBROUTINE MESH

SUBROUTINE MODIFY

!

! SUBPROGRAM TO MODIFY THE MESH.

! THIS VERSION APPLIES LINEAR SCALING TO THE NODE CO-ORDINATES.

!

USE MESHDATA

!

! INPUT THE DEPTH (Y-DIRECTION) AND WIDTH (X-DIRECTION).

READ(5,\*) H,W

!

! MODIFY THE CO-ORDINATES OF THE NODAL POINTS.

Each node in turn: DO I=1,NNP

X(I)=X(I)\*W

Y(I)=Y(I)\*H

END DO Each node in turn

RETURN

END SUBROUTINE MODIFY

SUBROUTINE MODIFY

!

! SUBPROGRAM TO MODIFY THE MESH TO SUIT A PARTICULAR PROBLEM.

! THIS VERSION ADAPTS A SQUARE MESH TO STRESS CONCENTRATION PROBLEM.

!

USE MESHDATA

NXPT=NPTS1

NYPT=NPTS2

!

! INPUT THE MESH SCALE FACTOR AND PLATE DIMENSIONS.

READ(5,\*) S,A,B

!

! TEST FOR ACCEPTABLE BASIC MESH.

IF(MOD(NXPT,2) == 0 .OR. MOD(NYPT,2) == 0) THEN

WRITE(6,61)

61 FORMAT(/"MESH UNSUITABLE FOR PRESENT MODIFICATION")

STOP

END IF

!

! PERFORM FIRST MODIFICATION OF Y CO-ORDINATES.

HR=(B-A)\*(S-1.)/(S\*\*(NYPT-1)-1.)

I=0

Each horizontal row in turn: DO IY=1,NYPT

YIMOD=HR\*(S\*\*(IY-1)-1.)/(S-1.)

IXMAX=NXPT

IF(MOD(IY,2) == 0) IXMAX=NXPT+1

Each point along the row in turn: DO IX=1,IXMAX

I=I+1

Y(I)=YIMOD

END DO Each point along the row in turn

END DO Each horizontal row in turn

!

! PERFORM SECOND MODIFICATION TO INTRODUCE CURVATURE.

PI=4.\*ATAN(1.)

Each node in turn: DO I=1,NNP

R=A+Y(I)

PHI=X(I)\*0.5\*PI

X(I)=R\*SIN(PHI)

Y(I)=R\*COS(PHI)

END DO Each node in turn

!

! MODIFY CO-ORDINATES OF POINTS NEXT TO THE END POINTS OF THE OUTERMOST

! CIRCUMFERENTIAL ROW.

I1=NNP-NXPT+2

I2=NNP-1

Y(I1)=B

X(I2)=B

!

! DEFINE AND TEST NEW TOTAL NUMBERS OF NODES AND ELEMENTS.

I=NNP

NNP=NNP+NXPT-2

M=NEL

NEL=NEL+2\*NXPT-6

IF(NNP > MAXNNP .OR. NEL > MAXNEL) THEN

WRITE(6,62) NNP,NEL

62 FORMAT(/"EXCESSIVE SIZE OF MESH, NNP =",I6,", NEL =",I6)

STOP

END IF

!

! DEFINE THE CO-ORDINATES OF THE ADDITIONAL NODES.

IXMAX=NXPT-3

Each node in turn around the edge: DO IX=1,IXMAX

I=I+1

II=I1+IX

IF(IX <= (NXPT-3)/2) THEN

X(I)=X(II)

Y(I)=B

END IF

IF(IX > (NXPT-3)/2) THEN

X(I)=B

Y(I)=Y(II-1)

END IF

END DO Each node in turn around the edge

X(NNP)=B

Y(NNP)=B

!

! DEFINE THE NODES OF THE ADDITIONAL ELEMENTS, OUTWARD POINTING

! THEN INWARD POINTING.

M1=M

Each outward element in turn: DO IX=1,IXMAX

M=M+1

NPI(M)=I1+M-M1-1

NPJ(M)=NPI(M)+1

NPK(M)=NPI(M)+NXPT-1

END DO Each outward element in turn

M2=M

IXMAX=IXMAX-1

Each inward element in turn: DO IX=1,IXMAX

M=M+1

NPI(M)=I1+M-M2

NPJ(M)=NPI(M)+NXPT-1

NPK(M)=NPJ(M)-1

END DO Each inward element in turn

!

! LAST ELEMENT.

NPI(NEL)=I2+(NXPT-3)/2+1

NPJ(NEL)=NPI(NEL)+1

NPK(NEL)=NNP

!

! ELLIPTICAL HOLE

Each node: DO I=1,NNP

Y(I)=Y(I)\*1.0

END DO Each node

RETURN

END SUBROUTINE MODIFY

SUBROUTINE MATLS

!

! SUBPROGRAM FOR DEFINING THE MATERIAL PROPERTIES OF THE ELEMENTS.

!

USE MATLDATA

USE MESHDATA

!

! INPUT THE MATERIAL PROPERTIES - MAXIMUM 10 DIFFERENT MATERIALS.

READ(5,\*) NMAT

IF(NMAT > 10) THEN

WRITE(6,61) NMAT

61 FORMAT(/"TOO MANY MATERIALS - NMAT =",I5)

STOP

END IF

READ(5,\*) (MAT,E(MAT),NU(MAT),ALPHA(MAT),RHOG(MAT),N=1,NMAT)

WRITE(6,62) (MAT,E(MAT),NU(MAT),ALPHA(MAT),RHOG(MAT),MAT=1,NMAT)

62 FORMAT(/"MATERIAL PROPERTIES" //

& " MATL E NU ALPHA RHOG" /

& (I5,E12.4,F8.3,2E12.4))

!

! DEFINE THE MATERIAL FOR EACH ELEMENT.

! THIS VERSION ASSUMES ALL ELEMENTS ARE OF FIRST MATERIAL.

Each element in turn: DO M=1,NEL

MATM(M)=1

END DO Each element in turn

RETURN

END SUBROUTINE MATLS

SUBROUTINE TEMPS

!

! SUBPROGRAM FOR DEFINING MEAN TEMPERATURE CHANGES FOR THE ELEMENTS.

! THIS VERSION READS AND ASSIGNS A UNIFORM CHANGE.

!

USE LOADDATA

USE MESHDATA

!

READ(5,\*) TEMP

Each element in turn: DO M=1,NEL

DELTAT(M)=TEMP

END DO Each element in turn

RETURN

END SUBROUTINE TEMPS

SUBROUTINE BODYF

!

! SUBPROGRAM FOR DEFINING THE BODY FORCE COMPONENTS (PER UNIT VOLUME)

! FOR THE ELEMENTS.

! THIS VERSION ASSUMES GRAVITY ACTS IN THE NEGATIVE Y-DIRECTION.

!

USE MATLDATA

USE LOADDATA

USE MESHDATA

!

Each element in turn: DO M=1,NEL

MAT=MATM(M)

XBAR(M)=0.

YBAR(M)=-RHOG(MAT)

END DO Each element in turn

RETURN

END SUBROUTINE BODYF

SUBROUTINE MSHOUT

!

! SUBPROGRAM TO WRITE OUT THE MESH DATA.

!

USE MESHDATA

USE MATLDATA

USE LOADDATA

!

! OUTPUT THE NUMBER OF ELEMENTS AND NODES, AND THE NODE CO-ORDINATES.

WRITE(7,71) NEL,NNP,(I,X(I),Y(I),I=1,NNP)

71 FORMAT(/"GEOMETRIC DATA FOR THE MESH" //

& 10X,"NUMBER OF ELEMENTS =",I6 //

& 10X,"NUMBER OF NODAL POINTS =",I6 //

& "NODAL POINT CO-ORDINATES" //

& 3(" I X Y ")/(3(I7,2F9.4)))

!

! OUTPUT THE ELEMENT NODE AND MATERIAL NUMBERS, AREAS, TEMPERATURE

! CHANGES AND BODY FORCE COMPONENTS.

WRITE(7,72) (M,NPI(M),NPJ(M),NPK(M),MATM(M),AREA(M),DELTAT(M),

& XBAR(M),YBAR(M),M=1,NEL)

72 FORMAT(/"ELEMENT DATA" // " M I J K MAT",

& " AREA DELTAT XBAR YBAR"/(4I7,I3,4E12.4))

RETURN

END SUBROUTINE MSHOUT

SUBROUTINE BCS

!

! SUBPROGRAM TO APPLY THE BOUNDARY CONDITIONS.

!

USE MESHDATA

USE EQNSDATA

USE MATLDATA

USE LOADDATA

USE RESTDATA

!

Each node in turn: DO I=1,NNP

FX(I)=0.

FY(I)=0.

END DO Each node in turn

!

! INPUT THE NUMBERS OF SETS OF DATA FOR EACH TYPE OF BOUNDARY CONDITION

READ(5,\*) NBC1P,NBC2F,NBC3P

!

! INPUT AND APPLY POINT FORCE DATA.

IF(NBC1P > 0) THEN

READ(5,\*) (I,FX(I),FY(I),N=1,NBC1P)

END IF

!

! INPUT AND APPLY DISTRIBUTED FORCE DATA.

IF(NBC2F > 0) THEN

Each distributed force in turn: DO IF=1,NBC2F

READ(5,\*) NBP,PX,PY

READ(5,\*) (NPB(N),N=1,NBP)

NS=NBP-1

Each boundary point in turn: DO IS=1,NS

I1=NPB(IS)

I2=NPB(IS+1)

SIDE=SQRT((X(I1)-X(I2))\*\*2+(Y(I1)-Y(I2))\*\*2)

FXM=0.5\*PX\*SIDE

FX(I1)=FX(I1)+FXM

FX(I2)=FX(I2)+FXM

FYM=0.5\*PY\*SIDE

FY(I1)=FY(I1)+FYM

FY(I2)=FY(I2)+FYM

END DO Each boundary point in turn

END DO Each distributed force in turn

END IF

!

! DEFINE FINAL MODIFIED EXTERNAL FORCES ON THE NODES.

Each node in turn: DO I=1,NNP

FXMOD(I)=FXMOD(I)+FX(I)

FYMOD(I)=FYMOD(I)+FY(I)

END DO Each node in turn

!

! INPUT AND APPLY THE RESTRAINED NODE DATA.

READ(5,\*) (NPB(N),NCOND(N),TANG(N),UPRES(N),VPRES(N),N=1,NBC3P)

Each restrained node in turn: DO N=1,NBC3P

I=NPB(N)

!

! NODAL POINT DISPLACEMENTS PRESCRIBED.

IF(NCOND(N) == 1) THEN

U(I)=UPRES(N)

V(I)=VPRES(N)

SFXX(I)=0.

SFXY(I)=0.

SFYX(I)=0.

SFYY(I)=0.

END IF

!

! NODE RESTRAINED TO MOVE IN DIRECTION WHOSE SLOPE IS GIVEN BY TANG.

IF(NCOND(N) == 2) THEN

U(I)=UPRES(N)

V(I)=VPRES(N)

SFXX(I)=(SFXX(I)\*SFYY(I)-SFXY(I)\*SFYX(I))/

& (SFXX(I)\*TANG(N)\*\*2-(SFXY(I)+SFYX(I))\*TANG(N)+SFYY(I))

SFXY(I)=SFXX(I)\*TANG(N)

SFYX(I)=SFXY(I)

SFYY(I)=SFXY(I)\*TANG(N)

END IF

!

! NODE RESTRAINED TO MOVE IN Y-DIRECTION ONLY.

IF(NCOND(N) == 3) THEN

U(I)=UPRES(N)

SFYY(I)=SFYY(I)-SFYX(I)\*SFXY(I)/SFXX(I)

SFXX(I)=0.

SFXY(I)=0.

SFYX(I)=0.

END IF

END DO Each restrained node in turn

RETURN

END SUBROUTINE BCS

SUBROUTINE FEQUIV(BSXX,BSYY,BSXY)

!

! SUBPROGRAM TO FIND THE NODAL POINT FORCES EQUIVALENT TO

! A SET OF EXTERNAL STRESSES. THESE STRESSES ARE APPLIED

! OVER A PART OF THE BOUNDARY FORMED BY A NUMBER OF SEGMENTS,

! EACH OF WHICH IS THE SIDE OF AN ELEMENTS.

!

USE MESHDATA

USE EQNSDATA

INTEGER :: IJK(3)

!

NS=NBP-1

Each boundary segment in turn: DO IS=1,NS

!

! IDENTIFY THE TWO POINTS AT THE ENDS OF THE SEGMENT.

I1=NPB(IS)

I2=NPB(IS+1)

!

! FIND THE ELEMENT HAVING THESE TWO POINTS AS NODES, AND FOR EACH

! POINT WHETHER IT IS THE FIRST, SECOND OR THIRD NODE OF THE ELEMENT.

Each element in turn: DO M=1,NEL

IJK(1)=NPI(M)

IJK(2)=NPJ(M)

IJK(3)=NPK(M)

N1=0

N2=0

NN=0

Each node of the element: DO IN=1,3

IF(I1 == IJK(IN)) THEN

N1=IN

NN=NN+1

END IF

IF(I2 == IJK(IN)) THEN

N2=IN

NN=NN+1

END IF

END DO Each node of the element

!

! SEARCH COMPLETE WHEN BOTH POINTS ARE FOUND IN AN ELEMENT.

IF(NN == 2) EXIT

END DO Each element in turn

IF(NN < 2 .OR. N1 == N2) THEN

WRITE(6,61) I1,I2

61 FORMAT(/"IN APPLIED BOUNDARY STRESS INPUT DATA,"/

& "POINTS ",I6," AND ",I6," ARE NOT TWO DIFFERENT",

& "NODES OF THE SAME ELEMENT - STOP")

STOP

END IF

!

! FIND THE FORCES AT THE POINTS EQUIVALENT TO THE STRESSES

! APPLIED TO THE BOUNDARY SEGMENT.

IF(N1 /= 1 .AND. N2 /= 1) THEN

A=AI(M)

B=BI(M)

END IF

IF(N1 /= 2 .AND. N2 /= 2) THEN

A=AJ(M)

B=BJ(M)

END IF

IF(N1 /= 3 .AND. N2 /= 3) THEN

A=AK(M)

B=BK(M)

END IF

FXM=-0.5\*(BSXX\*B+BSXY\*A)

FX(I1)=FX(I1)+FXM

FX(I2)=FX(I2)+FXM

FYM=-0.5\*(BSYY\*A+BSXY\*B)

FY(I1)=FY(I1)+FYM

FY(I2)=FY(I2)+FYM

END DO Each boundary segment in turn

END SUBROUTINE FEQUIV

SUBROUTINE SOLVE2

!

! SUBPROGRAM FOR SOLVING BY GAUSS-SEIDEL METHOD THE EQUATIONS

! OBTAINED FROM THE FINITE ELEMENT FORMULATION OF BIHARMONIC PROBLEMS.

!

! THIS VERSION ALLOWS FOR NONLINEAR MATERIAL PROPERTIES TO BE UPDATED

! PERIODICALLY.

!

USE EQNSDATA

USE MESHDATA

!

! INPUT THE SOLUTION PARAMETERS.

READ(5,\*) NCYCLE,IFREQ,ORELAX,TOLER

WRITE(6,61) ORELAX

61 FORMAT(/"SOLUTION OF EQUATIONS BY GAUSS-SEIDEL ITERATION" //

& "OVER-RELAXATION FACTOR = ",F6.3)

!

! SET UP ITERATION LOOP.

IF(IFREQ /= 0) WRITE(\*,62)

62 FORMAT(" ITER ERROR ")

Each cycle of iteration in turn: DO ITER=1,NCYCLE

SUMD=0.

SUMDD=0.

!

! OBTAIN NEW ESTIMATE FOR EACH UNKNOWN IN TURN.

Each pair of equations in turn: DO IROW=1,NNP

IF(SFXX(IROW)+SFYY(IROW) /= 0.) THEN

SUMX=FXMOD(IROW)

SUMY=FYMOD(IROW)

ICMAX=NAP(IROW)

Each overall column in turn: DO IC=1,ICMAX

ICOL=NPA(IROW,IC)

SUMX=SUMX-OKXX(IROW,IC)\*U(ICOL)-OKXY(IROW,IC)\*V(ICOL)

SUMY=SUMY-OKYX(IROW,IC)\*U(ICOL)-OKYY(IROW,IC)\*V(ICOL)

END DO Each overall column in turn

DELU=SFXX(IROW)\*SUMX+SFXY(IROW)\*SUMY

DELV=SFYX(IROW)\*SUMX+SFYY(IROW)\*SUMY

SUMDD=SUMDD+ABS(DELU)+ABS(DELV)

U(IROW)=U(IROW)+ORELAX\*DELU

V(IROW)=V(IROW)+ORELAX\*DELV

SUMD=SUMD+ABS(U(IROW))+ABS(V(IROW))

END IF

END DO Each pair of equations in turn

!

! TEST FOR CONVERGENCE.

ERROR=SUMDD/SUMD

IF(ERROR < TOLER) EXIT

!

! OUTPUT PROGRESS INFORMATION AND UPDATE MATERIAL PROPERTIES

! EVERY IFREQ CYCLES, UNLESS IFREQ=0.

IF(IFREQ /= 0) THEN

IF(MOD(ITER,IFREQ) == 0) THEN

WRITE(\*,63) ITER,ERROR

63 FORMAT(I8,E15.4)

CALL NONLIN

END IF

END IF

!

END DO Each cycle of iteration in turn

!

! WARN OF FAILURE TO CONVERGE.

IF(ERROR > TOLER) THEN

WRITE(6,64) NCYCLE

64 FORMAT(/"NO CONVERGENCE AFTER",I8," CYCLES")

RETURN

END IF

!

! OUTPUT NUMBER OF ITERATIONS AND TOLERANCE FOR CONVERGED SOLUTION.

WRITE(6,65) TOLER,ITER

65 FORMAT(/"ITERATION CONVERGED TO A TOLERANCE OF",E12.4,

& " AFTER",I8," CYCLES")

RETURN

END SUBROUTINE SOLVE2

SUBROUTINE NONLIN

!

! SUBPROGRAM FOR UPDATING MATERIAL PROPERTIES FOR A NONLINEAR

! PROBLEM DURING THE ITERATIVE SOLUTION OF THE EQUATIONS.

!

USE MESHDATA

USE EQNSDATA

USE MATLDATA

USE LOADDATA

USE RESTDATA

USE EMODDATA

!

REAL :: B(3,6),D(3,3),BTD(6,3),ESTIFF(6,6)

INTEGER :: IJK(3),IFLAG=0

!

! INITIALISE ELEMENT VALUES OF MODULUS, AND CHECK FOR TEMPERATURE

! CHANGES.

IF(IFLAG == 0) THEN

SUMDELT=0.

Each element in turn: DO M=1,NEL

MAT=MATM(M)

EE(M)=E(MAT)

SUMDELT=SUMDELT+ABS(DELTAT(M))

END DO Each element in turn

WRITE(6,61)

61 FORMAT(/"NONLINEAR MODULUS")

IF(SUMDELT > 0.) THEN

WRITE(6,62)

62 FORMAT(/"PROGRAM NOT VALID FOR TEMPERATURE CHANGES")

STOP

END IF

IFLAG=1

END IF

!

! SET INITIAL VALUES OF STIFFNESSES.

Each overall row in turn: DO IROW=1,NNP

Each overall column in turn: DO IC=1,13

OKXX(IROW,IC)=0.

OKXY(IROW,IC)=0.

OKYX(IROW,IC)=0.

OKYY(IROW,IC)=0.

END DO Each overall column in turn

END DO Each overall row in turn

!

! SET UP THE OVERALL ASSEMBLY LOOP.

Each element in turn: DO M=1,NEL

I=NPI(M)

J=NPJ(M)

K=NPK(M)

IJK(1)=I

IJK(2)=J

IJK(3)=K

!

! ELEMENT STRAINS AND STRESSES.

EXX=0.5\*(BI(M)\*U(I)+BJ(M)\*U(J)+BK(M)\*U(K))/AREA(M)

EYY=0.5\*(AI(M)\*V(I)+AJ(M)\*V(J)+AK(M)\*V(K))/AREA(M)

EXY=0.5\*(AI(M)\*U(I)+BI(M)\*V(I)+AJ(M)\*U(J)+BJ(M)\*V(J)+AK(M)\*U(K)

& +BK(M)\*V(K))/AREA(M)

MAT=MATM(M)

ET=ALPHA(MAT)\*DELTAT(M)

FACT=EE(M)/(1.-NU(MAT)\*\*2)

SIGXX=FACT\*((EXX-ET)+NU(MAT)\*(EYY-ET))

SIGYY=FACT\*(NU(MAT)\*(EXX-ET)+(EYY-ET))

SIGXY=FACT\*0.5\*(1.-NU(MAT))\*EXY

!

! NONLINEAR MODULUS.

EE(M)=ENONLIN(MAT,EXX,EYY,EXY,SIGXX,SIGYY,SIGXY)

!

! FORM THE ELEMENT DIMENSION MATRIX.

B(1,1)=BI(M)

B(1,2)=0.

B(1,3)=BJ(M)

B(1,4)=0.

B(1,5)=BK(M)

B(1,6)=0.

B(2,1)=0.

B(2,2)=AI(M)

B(2,3)=0.

B(2,4)=AJ(M)

B(2,5)=0.

B(2,6)=AK(M)

Each element column in turn: DO ICE=1,6

IF(MOD(ICE,2) == 0) B(3,ICE)=B(1,ICE-1)

IF(MOD(ICE,2) == 1) B(3,ICE)=B(2,ICE+1)

END DO Each element column in turn

!

! FORM THE ELASTIC PROPERTY MATRIX.

FACT=EE(M)/(1.-NU(MAT)\*\*2)

D(1,1)=FACT

D(1,2)=FACT\*NU(MAT)

D(1,3)=0.

D(2,1)=D(1,2)

D(2,2)=D(1,1)

D(2,3)=0.

D(3,1)=0.

D(3,2)=0.

D(3,3)=FACT\*0.5\*(1.-NU(MAT))

!

! MULTIPLY THE TRANSPOSE OF MATRIX B BY MATRIX D.

Each element row in turn: DO IRE=1,6

Each element column in turn: DO ICE=1,3

BTD(IRE,ICE)=0.

Sum products: DO ISUM=1,3

BTD(IRE,ICE)=BTD(IRE,ICE)+B(ISUM,IRE)\*D(ISUM,ICE)

END DO Sum products

END DO Each element column in turn

END DO Each element row in turn

!

! FORM THE ELEMENT STIFFNESS MATRIX.

Each element row in turn: DO IRE=1,6

Each element column in turn: DO ICE=1,6

SUM=0.

Sum products: DO ISUM=1,3

SUM=SUM+BTD(IRE,ISUM)\*B(ISUM,ICE)

END DO Sum products

ESTIFF(IRE,ICE)=0.25\*SUM/AREA(M)

END DO Each element column in turn

END DO Each element row in turn

!

! ADD ELEMENT STIFFNESS TO OVERALL STIFFNESS.

Each element row in turn: DO IRE=1,3

Each element column in turn: DO ICE=1,3

IROW=IJK(IRE)

ICOL=IJK(ICE)

!

! STORE STIFFNESS COEFFICIENTS IN RECTANGULAR FORM OF OVERALL MATRICES.

Each overall column in turn: DO IC=1,13

IF(NPA(IROW,IC) == ICOL) THEN

OKXX(IROW,IC)=OKXX(IROW,IC)+ESTIFF(2\*IRE-1,2\*ICE-1)

OKXY(IROW,IC)=OKXY(IROW,IC)+ESTIFF(2\*IRE-1,2\*ICE)

OKYX(IROW,IC)=OKYX(IROW,IC)+ESTIFF(2\*IRE,2\*ICE-1)

OKYY(IROW,IC)=OKYY(IROW,IC)+ESTIFF(2\*IRE,2\*ICE)

EXIT

END IF

END DO Each overall column in turn

END DO Each element column in turn

END DO Each element row in turn

END DO Each element in turn

!

! COMPUTE THE SELF-FLEXIBILITY SUBMATRICES.

Each node in turn: DO I=1,NNP

DENOM=OKXX(I,1)\*OKYY(I,1)-OKXY(I,1)\*OKYX(I,1)

SFXX(I)=OKYY(I,1)/DENOM

SFXY(I)=-OKXY(I,1)/DENOM

SFYX(I)=-OKYX(I,1)/DENOM

SFYY(I)=OKXX(I,1)/DENOM

END DO Each node in turn

!

! APPLY THE BOUNDARY RESTRAINT CONDITIONS.

Each restrained node in turn: DO N=1,NBC3P

I=NPB(N)

!

! NODAL POINT DISPLACEMENTS PRESCRIBED.

IF(NCOND(N) == 1) THEN

SFXX(I)=0.

SFXY(I)=0.

SFYX(I)=0.

SFYY(I)=0.

END IF

!

! NODE RESTRAINED TO MOVE IN DIRECTION WHOSE SLOPE IS GIVEN BY TANG.

IF(NCOND(N) == 2) THEN

SFXX(I)=(SFXX(I)\*SFYY(I)-SFXY(I)\*SFYX(I))/

& (SFXX(I)\*TANG(N)\*\*2-(SFXY(I)+SFYX(I))\*TANG(N)+SFYY(I))

SFXY(I)=SFXX(I)\*TANG(N)

SFYX(I)=SFXY(I)

SFYY(I)=SFXY(I)\*TANG(N)

END IF

!

! NODE RESTRAINED TO MOVE IN Y-DIRECTION ONLY.

IF(NCOND(N) == 3) THEN

SFYY(I)=SFYY(I)-SFYX(I)\*SFXY(I)/SFXX(I)

SFXX(I)=0.

SFXY(I)=0.

SFYX(I)=0.

END IF

END DO Each restrained node in turn

END SUBROUTINE NONLIN

FUNCTION ENONLIN(MAT,EXX,EYY,EXY,SIGXX,SIGYY,SIGXY)

!

! SUBPROGRAM TO FIND THE MODULUS OF A NONLINEAR MATERIAL FROM

! THE LOCAL STATE OF STRESS OR STRAIN.

! THIS VERSION FOR PLANE STRESS ONLY.

!

USE MATLDATA

USE EMODDATA

!

REAL, PARAMETER :: E0=50.E9, SIG0=10.E6, N=1.4

!

IF(MAT > 1) THEN

ENONLIN=E(MAT)

RETURN

END IF

!

! EQUIVALENT STRESS IN SIMPLE TENSION, ASSUMING PLANE STRESS.

SIGE=SQRT(SIGXX\*\*2+SIGYY\*\*2-SIGXX\*SIGYY+3.\*SIGXY\*\*2)

IF(SIGE/SIG0 < 1.E-3) SIGE=SIG0\*1.E-3

!

! NONLINEAR MODULUS WITH POWER-LAW DEPENDENCE ON STRESS.

ENONLIN=E0\*(SIGE/SIG0)\*\*(1.-N)

END FUNCTION ENONLIN

SUBROUTINE OUTPUT

!

! SUBPROGRAM TO OUTPUT THE FINAL RESULTS FOR A NONLINEAR

! MATERIAL PROBLEM.

!

USE MESHDATA

USE EQNSDATA

USE MATLDATA

USE LOADDATA

USE RESTDATA

USE EMODDATA

!

! OUTPUT THE DISPLACEMENT BOUNDARY CONDITIONS.

WRITE(6,61) (NPB(IB),NCOND(IB),TANG(IB),IB=1,NBC3P)

61 FORMAT(/"DISPLACEMENT BOUNDARY CONDITIONS" //

& " NODE COND TANG NODE COND TANG NODE",

& " COND TANG" / (3(I7,I5,F10.4)))

!

! OUTPUT THE NODAL POINT FORCES AND DISPLACEMENTS.

WRITE(6,62) (I,FX(I),FY(I),FXMOD(I),FYMOD(I),U(I),V(I),I=1,NNP)

62 FORMAT(/"NODAL POINT FORCES AND DISPLACEMENTS" //

& " NODE FX FY FXMOD FYMOD",

& " U V" / (I6,6E12.4))

!

! COMPUTE AND OUTPUT THE ELEMENT STRAIN AND STRESS COMPONENTS.

WRITE(6,63)

63 FORMAT(/" M EXX EYY EXY ET",

& " SIGXX SIGYY SIGXY SIGE",

& " E")

Each element in turn: DO M=1,NEL

I=NPI(M)

J=NPJ(M)

K=NPK(M)

EXX=0.5\*(BI(M)\*U(I)+BJ(M)\*U(J)+BK(M)\*U(K))/AREA(M)

EYY=0.5\*(AI(M)\*V(I)+AJ(M)\*V(J)+AK(M)\*V(K))/AREA(M)

EXY=0.5\*(AI(M)\*U(I)+BI(M)\*V(I)+AJ(M)\*U(J)+BJ(M)\*V(J)+AK(M)\*U(K)

& +BK(M)\*V(K))/AREA(M)

MAT=MATM(M)

ET=ALPHA(MAT)\*DELTAT(M)

FACT=EE(M)/(1.-NU(MAT)\*\*2)

SIGXX=FACT\*((EXX-ET)+NU(MAT)\*(EYY-ET))

SIGYY=FACT\*(NU(MAT)\*(EXX-ET)+(EYY-ET))

SIGXY=FACT\*0.5\*(1.-NU(MAT))\*EXY

SIGE=SQRT(SIGXX\*\*2+SIGYY\*\*2-SIGXX\*SIGYY+3.\*SIGXY\*\*2)

WRITE(6,64) M,EXX,EYY,EXY,ET,SIGXX,SIGYY,SIGXY,SIGE,EE(M)

64 FORMAT(I6,9E12.4)

END DO Each element in turn

RETURN

END SUBROUTINE OUTPUT