Examples' manual for *USER_LOADING option

Authors:

Melissa Adoum, Vincent Lapoujade
CRIL Technology

Correspondence:
Melissa Adoum
2, Impasse Henry Pitot
31500 Toulouse, FRANCE

Tel : +33.5.62.47.39.12
Fax : +33.5.61.20.47.89
e-mail : melissa.adoum@criltechnology.com

Keywords:
User loading, load blast

ABSTRACT

The aim of this study is to understand how the *USER_LOADING option works and to give some examples of its use. This option uses the subroutine *loadud,* its usage will be described step by step and it is illustrated with four examples :

- Plate loaded with time dependant pressure,
- Plate loaded with displacement dependant pressure,
- Randers-Pehrson & Bannister Tests,
- Plate loaded by an explosion of 5kg of TNT.

Some of those examples use loadings that can be also applied through LS-DYNA existing options, thus the results obtained with the user loading option could be compared and validated.

1 File *dyn21.f* and the *USER_LOADING option

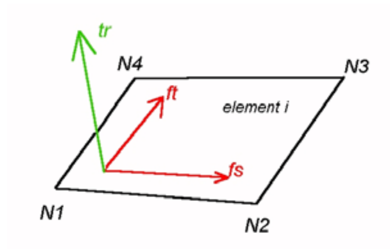
The file *dyn21.f* contains a collection of subroutines to be defined by the user, such as user defined materials, user defined loadings, …The keyword *USER_LOADING calls the subroutine *loadud* which is included in *dyn21.f*.
In this subroutine, the user has access to several variables like nodal displacement, velocity or acceleration as well as to his own parameters, whose values have been defined in the input deck. Pressure can be defined as a function of these user parameters or/and other variables of the model.
Once *loadud* is completed, *dyn21.f* should be compiled with LS-DYNA objects, yielding an executable LS-DYNA. File *dyn21.f* and LS-DYNA objects can be obtained from your local distributor or directly from LSTC.

2 Subroutine *loadud*

For a load applied on shell elements,
Block 1: declaration
As in almost all programs, the variable declaration is done first.

Block 2 : Read input & echo
The user parameters are read from the input deck and written in the echo file d3hsp.

Block 3 : Element evaluation
For each element, connectivity is determined. The nodes of each element and their coordinates are read. Then, the elementary coordinate system, the area of the element and its normal vector are computed.



Block 4 : pressure computation and nodal efforts
Here, the user must define the pressure as a function of his parameters, nodal data, ...

pressure = f(parm(1), parm(2), nod_displ, nod_accel,…)

The resultant force on the current element due to the pressure is then computed as :

Resultant_F = pressure x area.

Finally, the corresponding nodal force is estimated as :

Nodal_F = Resultant_F/4

Block 5 : Nodal force update
At current time, the nodal force is updated.

3 INPUT DECK

The input file for LS-DYNA contains the keyword *USER_LOADING followed by the user parameters. Up to 160 parameters can be defined, the next * ends this card.

```
*USER_LOADING
$        PARAM1          PARAM2          PARAM3
. . .
```

4 EXAMPLES

4.1 Pressure as a function of time

The system considered is a steel plate whose borders are fixed (translational degrees of freedom constrained). The square plate is 2m on a side and 1 mm thick. The model contains 400 Belytschko-Tsay shell elements with the warping option activated.

We apply a time dependent pressure that follows the relation : $p(t) = 320 - 64t^2$.

This load can also be applied to the system using the keyword *LOAD_SEGMENT. We will compare the results to those obtained using a user loading.

**LOAD_SEGMENT*

This is the definition of pressure using the *LOAD_SEGMENT option. LCID is the parabolic load curve pressure vs. Time :

```
*LOAD_SEGMENT
$     LCID        SF        AT        N1        N2        N3   N4
         1 1.0000000 0.0000000         7         6        27   28
*DEFINE_CURVE
         1
            .0000000  320.0000
...
```

**USER_LOADING*

The subroutine *loadud* is modified as follows : *lcid* and *scalfact* are introduced and correspond to the user parameters

```
LCID = INT(PARM(1))
SCALFACT = PARM(2).
```

The pressure computation is done as

```
PRESSURE = SCALFACT . FVAL(LCDI)
```

fval(lcid) : value of load curve *lcid* at current time

In the input file, the command is :

```
*USER_LOADING
$    PARM1     PARM2
$     LCDI  SCALFACT
         1      1.00
*DEFINE_CURVE
         1
            .0000000  320.0000
 …
```

4.2 Results

In both cases, the load applied to the whole system is a pressure which changes as a parabolic function of time.

The results for both loading methods (*LOAD_SEGMENT and *USER_LOADING) can be compared.

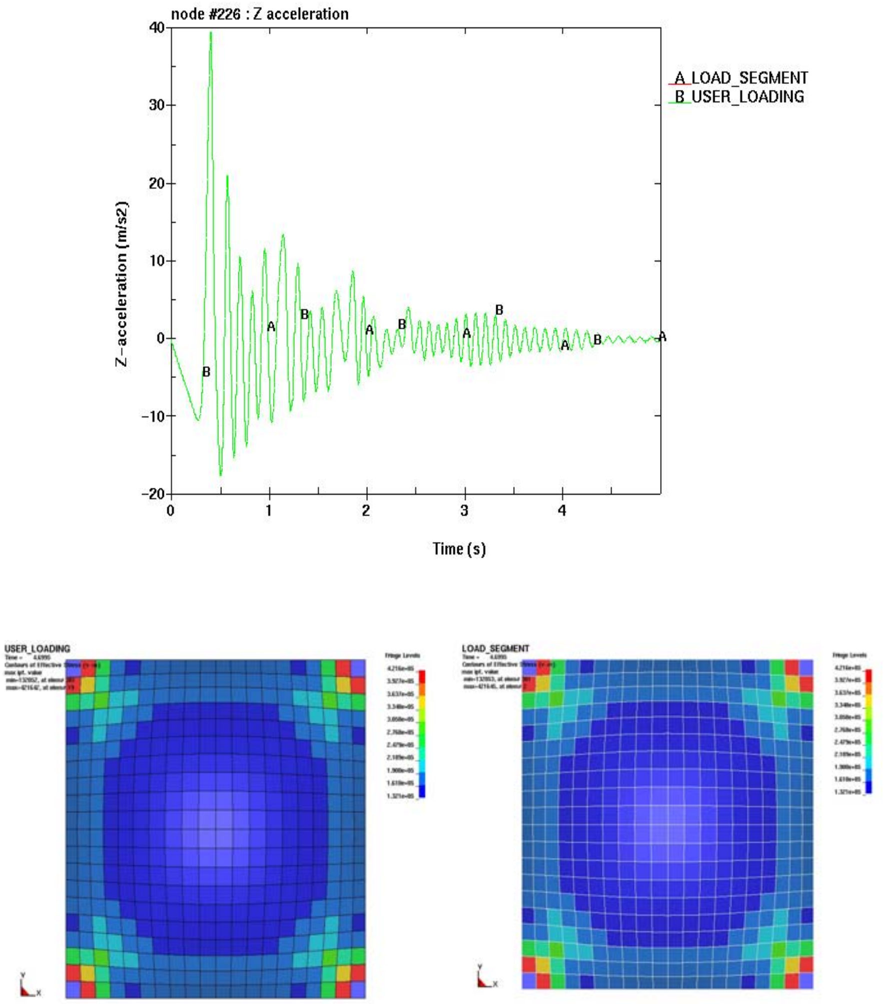


Figure 1 : Z-acceleration at node 226 & Von Mises stress at time t= 4,7s.

As shown in figures, both loading methods lead to the exact same nodal displacements and accelerations. The Von Mises stress is perfectly symmetric, as expected, and is identical in both cases.

4.3 Pressure as a function of displacement

The model of a square steel plate described previously is again used here.
The load is a pressure applied on each segment which is a function of the average Z_displacement of the four nodes of the segment and pressure is defined as :

$$p(t) = press0 \times (press1 + press2 \times displZ(t))$$

*USER_LOADING

In the subroutine *loadud* , pressure is a linear function of nodal average displacement, so user parameters are the coefficients of this function :

```
PRESS0=PARM(1)
PRESS1=PARM(2)
PRESS2=PARM(3)
```

The average nodal Z_displacement of element i at current time is computed using :

```
DEPLZ=(D(3,IXS(2,I))+D(3,IXS(3,I))+D(3,IXS(4,I))+D(3,IXS(5,I)))/4
PRESSURE=PRESS0*(PRESS1+PRESS2*DEPLZ)
```

Finally, we write current time and pressure for element #190 in unit 666. This is useful to compare pressure vs. time and nodal displacement vs. time :

```
IF (I.EQ.190) THEN
   WRITE (666,*) TIME, PRESSURE
ENDIF
```

In the keyword file, variables *press0, press1* and *press2* are defined by the user as follows :

```
*USER_LOADING
$     PARM1      PARM2     PARM3
        1.5   1.000+0  -4.000+2
```

4.4 Results

In order to check the results, we compare the average Z_displacement at nodes #226, 225, 205, 204 (ASCII file *nodout*) and the pressure applied on element #190 (ASCII file fort.666) (Figure 2)

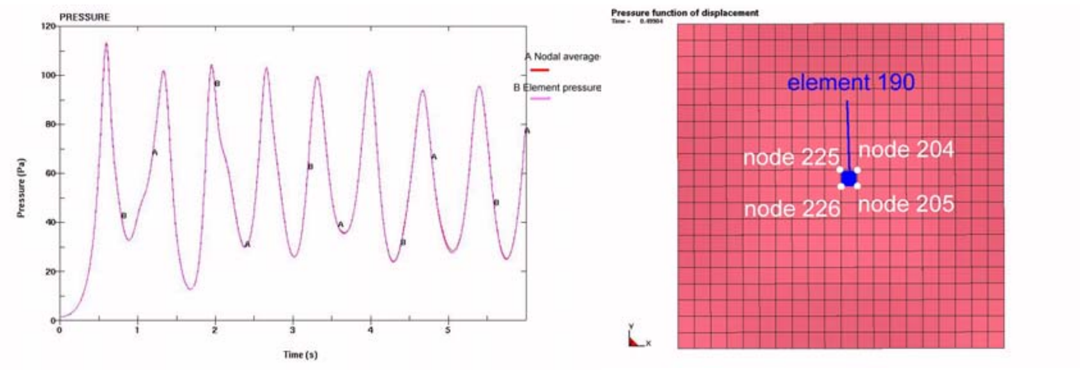


***Figure** 2 : Pressure comparison vs. time*

Comparison between pressure on element and pressure calculated using the nodal Z_displacement.

As shown in Figure 2 the pressure applied on element #190 corresponds perfectly to the pressure calculated using the average nodal displacement and the linear equation.

The usage of *USER_LOADING leads to exactly the same results than the usage of *LOAD_SEGMENT option. Both loading methods are identical.

5 CONWEP examples

Since version 960, LS-DYNA contains the *LOAD_BLAST option to model blast effects from explosions. The CONWEP option was implemented using the Rander-Pehrson & Bannister study (Réf.1).

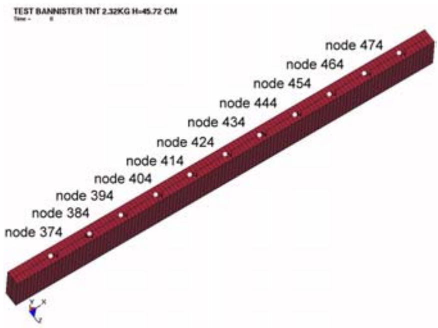
To call the CONWEP function, the input deck must contain the following options :

- *LOAD_BLAST : the user specifies the properties of the explosive here : mass, position, time, unit system and type of explosion (spherical or hemispherical).
- *LOAD_SEGMENT : here the user specifies the load curve ID = -2 to call the CONWEP function.

We used subroutine *loadud* to program this function as a user loading. The program uses the subroutines from Rander-Pehrson & Bannister as detailed in their report.

Rander-Pehrson & Bannister test

***Figure** 3 : Rander-Pehrson & Bannister beam test.*



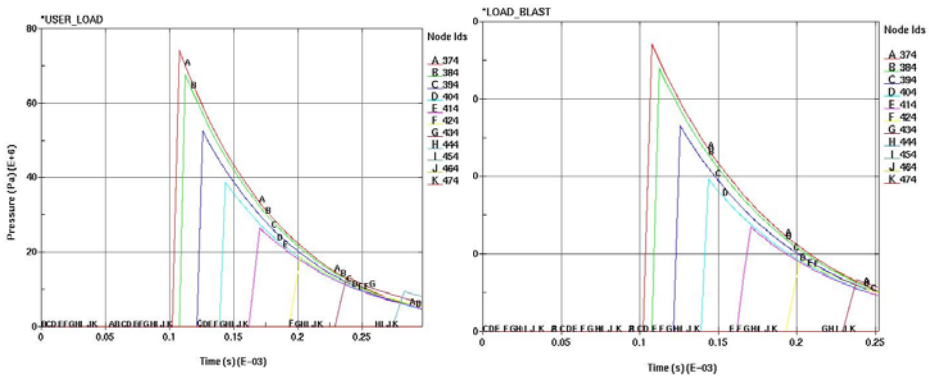
In order to test their implementation of the CONWEP function, Rander-Pehrson & Bannister modelled the explosion of 2.32 kg. of TNT above a rigid and reflecting surface. We reproduced this example.
To obtain the pressure profiles, the surface is modelled as a high-density fluid beam, whose dimensions are : LX = 1.2 m, LY = 0.1 m, LZ = 0.04 m.
The elastic material of the beam has a density of 20000 kg/m^3 , a Young's modulus equal to 1.e^{-4} Pa and a Poisson's ratio equal to zero. The translational degrees of freedom x and z are blocked. The pressure on one node can be evaluated using the relation : $p_g = \frac{1}{2}\rho e \ddot{y}_g = 1000 \times \ddot{y}_g$, where :

Pg : pressure at node (Pa)
ρ : density (20000 kg/m^3)
e : = LY, thickness of the beam (0.1 m)
$\ddot{y}_g$: Y_acceleration of node (m/s^2)

Then, the pressure profile can be obtained easily from the acceleration profile. We will consider 12 nodes on the beam (Figure 3) and we'll compare the pressure on these nodes for both loading methods : *LOAD_BLAST and *USER_LOADING.

5.1 Results

***Figure** 4 : Comparison of nodal pressure vs. time for both loadings*



As can be seen in Figure 4, both methods of load definition lead to identical results, as expected.

5.2 Explosion above a square steel plate

The square steel plate described previously is again used here. The explosion of 5 kg. of TNT takes place 1m above the centre of the plate. We compare the results obtained with *LOAD_BLAST and *USER_LOADING.

5.3 Results

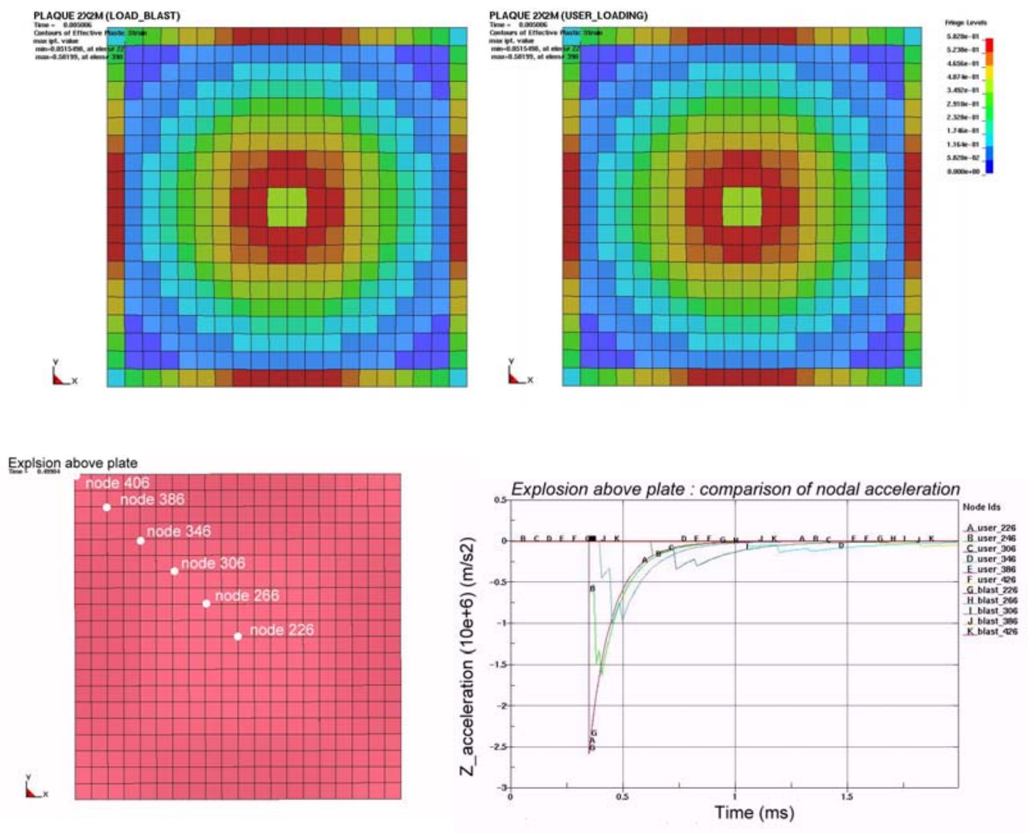


Figure 5 : Final plastic strain and comparison of nodal Z-acceleration

As the explosion takes place above the center of the plate, and as the plate is perfectly symmetric and isotropic, we will consider some nodes on the diagonal as shown in Figure 5.

In graphics representing plastic strain (Figure 5), we check, on one hand, the symmetry of deformation patterns and on the other hand, that both loading methods give similar values.

As shown in Figure 5 both loading methods lead to identical nodal acceleration .

6 CONCLUSION : User loading in 7 steps

The examples presented here intend to show how to use a user defined loading. To create a model with a user defined loading, follow these steps :

1. Choose the pressure/force function and choose user parameters *user1, …*

In the subroutine *loadud* (file *dyn21.f*) :

2. Declare variables *user1, user2, ...* in the declaration block

3. Initialise variables *user1 = parm(1), user2 = parm(2), ...*
4. Compute the pressure/force as a function of user variables and model variables : *pressure = f(user1, user2, … , nodal_displ, nodal_force, …).*
5. Compile *dyn21.f* with the LS-DYNA objects and create the executable file.
6. Create the input file containing the *USER_LOADING option and the values for chosen parameters.
7. Execute

7 REFERENCES

1. **Airblast Loading Model for DYNA2D and DYNA3D** – March 1997 – Glenn Randers-Pehrson & Kenneth A. Bannister – ARL
2. **CRIL TECHNOLOGY (DYNALIS) Internal Technical Report EQ 14**.3 – *Test du module CONWEP de LS-DYNA, version 960* – Gael Le Blanc – April 2000, CRIL Technology (Dynalis)
3. **Report D002-2000-E-R01 -** *Implementation d'un modele de souffle dans LS-DYNA* - – Gael Le Blanc – January 2000, CRIL Technology (Dynalis)

8 APPENDIX

```
      SUBROUTINE LOADUD(FNOD,DT1,TIME,IRES,X,D,V,A,IXS,
     . NUMELS,IXB,NUMELB,IDRFLG,TFAIL,ISF,P,NPC,FVAL,IOB,IADD)
C
C     INPUT ARRAYS
C
C     FNOD - GLOBAL NODAL FORCES
C
C     DT1 - CURRENT TIME STEP SIZE
C     TIME - CURRENT PROBLEM TIME
C     IRES - RESTART FLAG, ( 0=SOLUTION PHASE     )
C                          (-N=INPUT N PARAMETERS )
C                          ( 2=RESTART            )
C                          ( 3=READ DATA FROM RESTART FILE      )
C                          ( 4=WRITE DATA INTO DUMP FILE        )
C
C     WHEN DATA IS READ, DUMMY ARRAYS ARE PASSED IN THE CALL.
C     DATA SHOULD BE READ INTO A LOCAL COMMON BLOCK WHICH IS
C     WRITTEN INTO THE RESTART DATABASE.
C
C
C     D - DISPLACEMENTS
C     V - VELOCITIES
C     A - ACCELEATIONS
C
C     IXS - SHELL ELEMENT CONNECTIVITIES (IXS(1,*)=PART ID)
C                                        (IXS(2,*)=NODE 1)
C                                        (IXS(3,*)=NODE 2)
C                                        (IXS(4,*)=NODE 3)
C                                        (IXS(5,*)=NODE 4)
C
C     IXB - BEAM  ELEMENT CONNECTIVITIES (IXB(1,*)=PART ID)
C                                        (IXB(2,*)=NODE 1)
C                                        (IXB(3,*)=NODE 2)
C                                        (IXB(4,*)=ORIENTATION NODE)
C
C     NUMELS - NUMBER OF SHELL ELEMENTS
C     NUMELB - NUMBER OF BEAM  ELEMENTS
C     ISF    - SHELL ELEMENT FAILURE FLAG (1=ON)
C     TFAIL  - SHELL ELEMENT FAILURE TIME (EQ.0:OKAY)
C                                         (NE.0:FAILURE TIME)
C
C     IDRFLG - NONZERO IF DYNAMIC RELAXATION PHASE
```

```
C     P       - LOAD CURVE DATA PAIRS (ABSCISSA, ORDINATE)
C     NPC     - POINTER INTO P.  (P(NPC(LC)) POINTS TO THE BEGINNING
C                 OF LOAD CURVE ID LC.  NPOINTS=NPC(LC+1)-NPC(LC)=
C                 NUMBER OF POINTS IN THE LOAD CURVE.
C     FVAL    - FVAL(LC) IS THE VALUE OF LOAD CURVE LC AT T=TIME
C     IOB     - I/O BUFFER
C
      COMMON/USRLDV/PARM(160)
      CHARACTER*80 TXTS,MSSG                                         {1}
      DIMENSION A(3,*),V(3,*),D(3,*),FNOD(3,*),IXS(5,*),IXB(4,*),
     . X(3,*),TFAIL(*),P(*),NPC(*),FVAL(*),IOB(*)
C
      IF (IRES.LT.0) THEN
        N=ABS(IRES)
        WRITE (13,1030)
        MSSG='*** ERROR READING IN USER LOADING SUBROUTINE'
        DO 10 I=1,N,8
        CALL GTTXSG (TXTS,LCOUNT)
        READ (UNIT=TXTS,FMT=1020,ERR=400) (PARM(J),J=I,MIN(I+7,N))
        WRITE (13,1040)                   (PARM(J),J=I,MIN(I+7,N))
   10   CONTINUE
        WRITE (13,1050)
        RETURN
      ENDIF
C
      IF (IRES.EQ.3) THEN
        NLODDV=160
        CALL WRABSF(IOB,PARM,NLODDV,IADD)                          {2}
        IADD=IADD+NLODDV
        RETURN
      ENDIF
C
      IF (IRES.EQ.4) THEN
        NLODDV=160
        CALL RDABSF(IOB,PARM,NLODDV,IADD,IOERR)
        IADD=IADD+NLODDV
        RETURN
      ENDIF
C
      DO 20 I=1,NUMELS
C
      IXS2I=IXS(2,I)
      IXS3I=IXS(3,I)
      . . .
C
      XX11 =X(1,IXS2I)
      XX21 =X(2,IXS2I)                                           {3}
      XX31 =X(3,IXS2I)
C
      XX12 =X(1,IXS3I)
      XX22 =X(2,IXS3I)
      XX32 =X(3,IXS3I)
C
      XX13 =X(1,IXS4I)
      XX23 =X(2,IXS4I)
      XX33 =X(3,IXS4I)
C
      XX14 =X(1,IXS5I)
      XX24 =X(2,IXS5I)
      XX34 =X(3,IXS5I)
C
      FS1 =-XX11+XX12+XX13-XX14
      FS2 =-XX21+XX22+XX23-XX24
      FS3 =-XX31+XX32+XX33-XX34
C
      FT1 =-XX11-XX12+XX13+XX14
      FT2 =-XX21-XX22+XX23+XX24
```

```
      FT3 =-XX31-XX32+XX33+XX34
C
      E=FS1*FS1+FS2*FS2+FS3*FS3
      F=FS1*FT1+FS2*FT2+FS3*FT3
      G=FT1*FT1+FT2*FT2+FT3*FT3
C
      AREA=SQRT((E*G-F*F)/16.)
      TR1 =FS2*FT3-FS3*FT2
      TR2 =FS3*FT1-FS1*FT3
      TR3 =FS1*FT2-FS2*FT1
      XMG =.25/SQRT(TR1**2+TR2**2+TR3**2)
C
      PRESSURE= F(PARM(1), PARM(2),…)              {4}
C
      TR1 =XMG*TR1*AREA*PRESSURE
      TR2 =XMG*TR2*AREA*PRESSURE
      TR3 =XMG*TR3*AREA*PRESSURE
C
      FNOD(1,IXS2I)=FNOD(1,IXS2I)-TR1
      . . .
C
   20 CONTINUE
      RETURN
C
  400 CALL TERMIN (TXTS,MSSG,LCOUNT,1)
 1020 FORMAT(8E10.0)
 1030 FORMAT(////' U S E R   D E F I N E D   L O A D I N G',
     .           '   P A R A M E T E R S',/)
 1040 FORMAT(
     1 5X,'   PARAMETER NUMBER  ',I4,'=', E12.4)
 1050 FORMAT(////)
      END
```