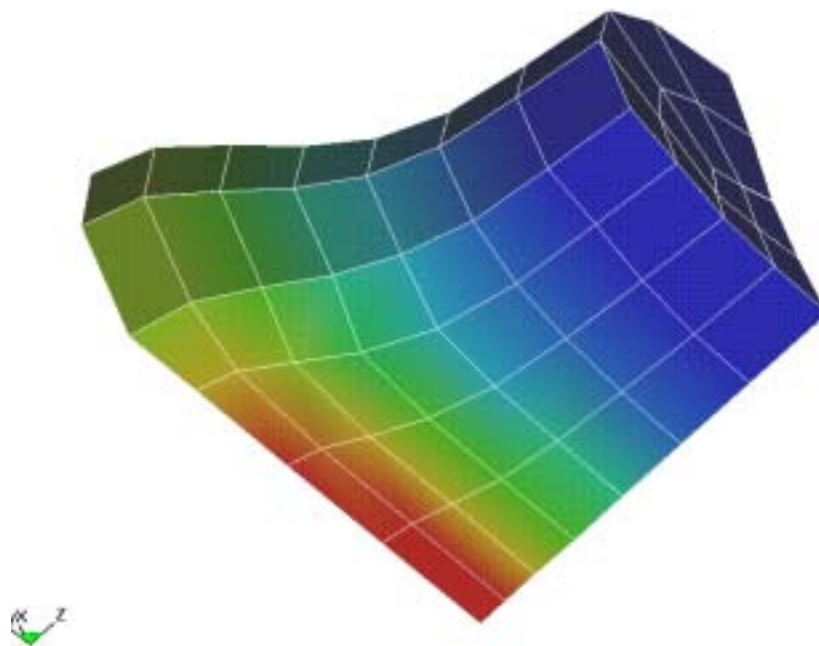


Mattias Unosson, Eric Buzaud

Scalar and Vectorized User Defined Material Routines in LS-DYNA



DEFENCE RESEARCH ESTABLISHMENT

Weapons and Protection Division

SE-147 25 TUMBA

FOA-R--00-01502-311--SE

April 2000

ISSN 1104-9154

Mattias Unosson, Eric Buzaud

Scalar and Vectorized User Defined Material Routines in LS-DYNA

Distribution: HKV, Dynalis, Engineering Research AB, LTU (Structural Mechanics), CTH (Concrete Structures), LiTH (Solid Mechanics)

FOA: Program, Utland, FOA 2, FOA 23

Issuing organization Defence Research Establishment Weapons and Protection Division SE-147 25 TUMBA SWEDEN	Document ref. No., ISRN FOA-R--00-01502-311--SE	
	Date of issue April 2000	Project No. E2011
	Project name (abbrev. if necessary) Structural protection for stationary/mobile tactical behaviour	
Author(s) Mattias Unosson Eric Buzaud	Initiator or sponsoring organization Swedish defence HQ	
	Project manager Håkan Hansson	
	Scientifically and technically responsible	
Document title Scalar and Vectorized User Defined Material Routines in LS-DYNA		
Abstract <p>In the report a description of how to write, implement and use your own material models in the finite element code LS-DYNA. Two material models (elastic and elastic-plastic) are implemented in the formats scalar and vectorized and verified with material models from the LS-DYNA standard library using a simple problem. Problems with programming and different systems are discussed and also how to handle options like equation of state, erosion etc.</p> <p>The report is a result from a cooperation with Dynalis, France.</p>		
Keywords LS-DYNA, user defined routine, material model, FE modelling		
Further bibliographic information		Language English
ISSN 1104-9154		ISBN
		Pages 47 p.
Distributor (if not issuing organization)		

Dokumentets utgivare Försvarets forskningsanstalt Avdelningen för Vapen och skydd SE-147 25 TUMBA	Dokumentbeteckning, ISRN FOA-R--00-01502-311--SE	
	Dokumentets datum April 2000	Uppdragsnummer E2011
	Projektamn (ev förkortat) Anläggningskydd för fast/rörligt uppträdande	
Upphovsman(män) Mattias Unosson Eric Buzaud	Uppdragsgivare FM	
	Projektansvarig Håkan Hansson	
	Fackansvarig	
Dokumentets titel Skalära och vektoriserade användardefinierade materialrutiner i LS-DYNA		
Sammanfattning I rapporten ges en beskrivning av hur man skriver, implementerar och använder egna material modeller i den finita element koden LS-DYNA. Två material modeller (elastisk och elastisk-plastisk) implementeras i de båda formaten skalär och vektoriserad samt verifieras mot material modeller ur LS-DYNAs standardbibliotek med hjälp av ett enkelt problem. Problem med programmering och olika datorsystem behandlas även, liksom hur man hanterar optioner som tillståndsekvationer, erosion etc. Arbetet är ett resultat från ett samarbete med Dynalis, Frankrike.		
Nyckelord LS-DYNA, användardefinierad rutin, materialmodell, FE modellering		
Övriga bibliografiska uppgifter	Språk Engelska	
ISSN 1104-9154	ISBN	
	Omfång 47 s.	

Distributör (om annan än ovan)

CONTENTS

- 1 INTRODUCTION 3
- 2 CODE CONTEXT 4
 - 2.1 Call for an user defined material routine..... 4
 - 2.2 Structure of a scalar user defined material routine 6
 - 2.3 Structure of a vectorized user defined material routine 8
 - 2.4 Compilation 9
 - 2.5 Input files..... 10
- 3 IMPLEMENTATION OF AN ELASTIC MODEL AS AN USER DEFINED MATERIAL MODEL 11
 - 3.1 LS-DYNA Standard material routine f3dm1 (type 1) 11
 - 3.2 Scalar userdefined material routine umat41..... 12
 - 3.3 Vectorized user defined material routine umat41v..... 13
 - 3.4 CPU cost..... 17
- 4 IMPLEMENTATION OF AN ELASTIC-PLASTIC MODEL WITH ISOTROPIC HARDENING AS AN USERDEFINED MATERIAL MODEL 18
 - 4.1 LS-DYNA Standard material routine f3dm3 (type 3)..... 18
 - 4.2 Scalar userdefined material routine umat42..... 18
 - 4.3 Vectorized user defined material routine umat42v 20
 - 4.4 CPU cost..... 22
- 5 MISCELLANEOUS OPTIONS 23
 - 5.1 Equation of state..... 23
 - 5.2 Erosion..... 24
 - 5.3 Load curves..... 24
- 6 CONCLUSIONS..... 25
- REFERENCES..... 25
- APPENDICES 26
 - Appendix A..... 26

Appendix B 31

Appendix C 33

Appendix D 35

Appendix E 36

Appendix F 37

Appendix G 38

Appendix H 39

Appendix I 40

Appendix J 41

Appendix K 43

Appendix L 44

Appendix M 46

Appendix N 48

1 INTRODUCTION

The explicit finite element code LS-DYNA, developed by Livermore Software Technology Corporation (<http://www.lstc.com/>), has given the possibility to use your own, user defined, material models since 1989. In December 1999 the Department of Protection and Materials at the Defence Research Establishment (<http://www.foa.se/>) and Dynalis (<http://www.dynalis.fr/>) established a cooperation aiming to learn how to use this possibility. During January and February 2000 the authors worked together at the Dynalis office in Gourdon, France.

The object of this report is to explain the context of a user defined material routine and with the help of two examples show step by step how to implement a user defined material in LS-DYNA. Finally a description of how to handle options as equation of state, erosion and strain rate effects is given.

2 CODE CONTEXT

2.1 Call for an user defined material routine

The software LS-DYNA gives the possibility to define up to 10 Fortran-routines containing user defined constitutive models. Thus an user can carry out computations with a maximum of 10 user defined material models.

LS-DYNA solves the fundamental conservation equations in continuum mechanics using an explicit time algorithm. In the time integration loop (external loop) the user routines are called after having calculated strains and strain rates. The user routine calculates the stress field using these inputs. The call for an user defined material is different from the call for a standard material, see figure 1 below.

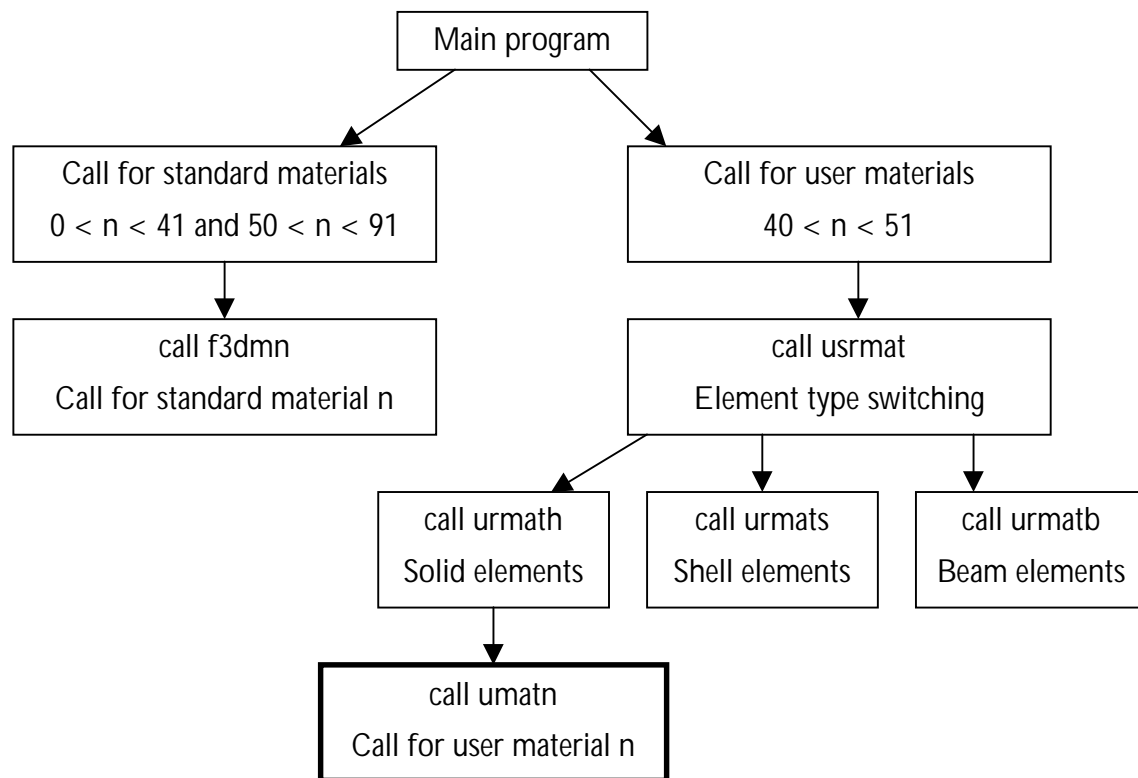


Figure 1. Call for an user defined material routine.

The routine urmata can be found in Appendix A.

The user routines can have one of two formats; scalar or vectorized. In the first case the user routine is sequentially called for each element. A vectorized routine is called with a block of elements and the size of this block depends on the type of machine. At the beginning of the urmata you can verify this block size (nlq) for different machines and change it if necessary. A vectorized routine has the advantage of increasing the performance on vector processors (CRAY, CONVEX etc).

The scalar routines are called umat41, umat42 ... umat50 and the standard structure of an user routine is found in table 1. The vectorized routines are called umat41v, umat42v ...umat50v and the standard structure of an user routine is found in table 2.

Table 1. Standard structure of an scalar user routine.

```

subroutine umatn (cm,eps,sig,hisv,dt1,capa,etype,tt)
dimension cm(*),eps(*),sig(*),hisv(*)
real dt1,capa,tt
character*(*) etype
(...)
return
end

```

Table 2. Standard structure of an vectorized user routine.

```

subroutine umatnv(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c
c - Parameter nlq
c For CRAY/T3E          use 24
c For CRAY/J90         use 256
c For CRAY/C90         use 256
c For CRAY/T90         use 256
c For CRAY/T90IEEEE    use 256
c For DEC/Alpha        use 89
c For EJ/VFF           use 1033
c For Hitachi          use 130
c For HP pa7x00        use 33
c For HP pa8x00        use 65
c For HP Exemplar      use 128
c For IBM              use 128
c For NEC/SX4          use 1024
c For SGI 4400         use 32
c For SGI R10000       use 87
c For SUN              use 128
parameter (nlq=87)
c
dimension cm(*),d1(nlq),d2(nlq),d3(nlq),d4(nlq),d5(nlq),d6(nlq),
. sig1(nlq),sig2(nlq),sig3(nlq),sig4(nlq),sig5(nlq),
. sig6(nlq),sigma(nlq,*),hist(nlq,*),
character*(*) etype
(...)
return
end

```

2.2 Structure of a scalar user defined material routine

All input and output variables are passed on as arguments in the call. But we will see later on that in some cases we will need to pass variables using so called common blocks.

The variables passed as arguments are found in the following table.

Table 3. Arguments for a scalar user defined material routine

cm(1)	Model parameter number 1
.	.
.	.
.	.
cm(48)	Model parameter number 48
eps(1)	Strain increment in local x-direction
eps(2)	Strain increment in local y-direction
eps(3)	Strain increment in local z-direction
eps(4)	Strain increment in local xy-direction
eps(5)	Strain increment in local yz-direction
eps(6)	Strain increment in local zx-direction
sig(1)	Stress in local x-direction
sig(2)	Stress in local y-direction
sig(3)	Stress in local z-direction
sig(4)	Stress in local xy-direction
sig(5)	Stress in local yz-direction
sig(6)	Stress in local zx-direction
hisv(1)	History variable number 1
hisv(2)	History variable number 2
.	.
.	.
.	.
hisv(n)	History variable number n
dt1	Current time step
capa	Shear reduction factor (Not used here)
etype	equals "brick" for solid elements equals "shell" for shell elements equals "beam" for beam elements
tt	Current accumulated time

The parameters must include the bulk modulus and the shear modulus. These are used by LS-DYNA to calculate penalty coefficients for contact algorithms and the time step.

The history variables are used only in the user routine and can for example be used to store the accumulated effective plastic strain and model damage variables. To visualize our history variables in a post-processor an additional control card containing the number of variables has to be defined.

All energy calculations are done outside the user routine which restricts us to material models that does not change the way of calculating the internal energy (i.e. no equations of state, no temperature dependency etc.). There are ways to circumvent this limitation and these are explained in the chapter on equations of state.

Rate objectivity is also handled outside the user routine and for this the Jaumann rate is used.

2.3 Structure of a vectorized user defined material routine

In the case of a vectorized user routine a group of elements is treated at a time by the user routine, whereas for a scalar routine only one element is passed through the routine. Apart from the arguments in the call, see table 4, the same is true as said for the scalar routine in chapter 2.2.

Note that the element numbering i in our routine is not the same as the global element numbering. The value i runs repeatedly from lft to llt ($llt-lft=nlq$) until all elements has been treated.

Table 4. Arguments for a vectorized user defined material routine

cm(1)	Model parameter number 1
.	.
cm(48)	Model parameter number 48
d1(i)	Strain rate in local x-direction for element number i
d2(i)	Strain rate in local y-direction for element number i
d3(i)	Strain rate in local z-direction for element number i
d4(i)	Strain rate in local xy-direction for element number i
d5(i)	Strain rate in local yz-direction for element number i
d6(i)	Strain rate in local zx-direction for element number i
sig1(i)	Stress in local x-direction for element number i
sig2(i)	Stress in local y-direction for element number i
sig3(i)	Stress in local z-direction for element number i
sig4(i)	Stress in local xy-direction for element number i
sig5(i)	Stress in local yz-direction for element number i
sig6(i)	Stress in local zx-direction for element number i
sigma(i,*)	Stresses and history variables (Normally not used)
hist(i,1)	History variable number 1 for element number i
hist(i,2)	History variable number 2 for element number i
.	.
hist(i,n)	History variable number n for element number i
lft	Lower value of element numbering i ($llt-lft=nlq$)
llt	Upper value of element numbering i ($llt-lft=nlq$)
dt1	Current time step
capa	Shear reduction factor (Not used here)
etype	equals "brick" for solid elements equals "shell" for shell elements equals "beam" for beam elements
tt	Current accumulated time

2.4 Compilation

In order to create our own LS-DYNA executable file containing the user routine we need the following:

- Fortran user routine
- Fortran 77 or Fortran 90 compiler
- Makefile (see Appendix B)
- The Fortran file dyn21.f (see Appendix C)
- Object code files (Files with extension .o in the Makefile)

The last three items are supplied by your LS-DYNA distributor and they will need the specifications for the system you will use. The systems used by the authors when working with this report are described below.

Table 5. System specifications

	<i>Dynalis system</i>	<i>FOA system</i>
Computer	Silicon Graphics Octane	Compaq Workstation XP1000
Processor	195 MHz MIPS 10000(IP30) processor with MIPS R10010 FPU, 256MB main memory.	667 MHz DEC/Alpha 21264A processor with 1024 MB main memory.
OS	IRIX64 release 6.5	Digital UNIX version 4.0d
Fortran	SGI Fortran 77 compiler	Digital Fortran 90 compiler version 5.2
Pre-processor	LS-INGRID version 3.5b	LS-INGRID version 3.5b
LS-DYNA version	950	950c
Post-processor	LS-POST version 1.0	LS-POST version 1.2

In the file dyn21.f we find all the available subroutine names for user material routines. There are two ways of linking our Fortran user routine to the compilation. Either we add the code in the dyn21.f file, but it is more convenient to comment out a call in the dyn21.f (as done in Appendix C) and place our user routine in a separate file, for example with the name umat41.f. When having done so and also having the Makefile and the object code files in the same directory we can execute the compilation and linking by typing "make" on a command line. Now we should have a new executable file and the name of the created file can be verified and changed in the Makefile at the line:

```
PROGRAMA = LS-DYNA_950c_compaq_40_pa_userdef
```

There could be problems with the compilation when using so called common blocs in your material routine, but this should your LS-DYNA distributor be able to help you with. For example on the Compaq Workstation described in table 5 the command;

```
c$omp thread private (/subtssloc/)
```

had to be added to our codes (see Appendices J-M) for a succesful linking and compilation. For the Dynalis system there was no need for such a command, but when trying to compile on a HP machine the line;

```
c$dir thread private (/subtssloc/)
```

had to be added.

2.5 Input files

The input files for an user defined material routine are easy to write and an example is given in table 6 below and the resulting LS-DYNA Keyword input file is shown in table 7.

Table 6. Example of an material definition as a LS-INGRID input file [1].

```

taurus int8 8;                                c To visualize history variables in the
                                              c post-processor (solid elements)

start
mat 1
type 42 vect                                  c Delete "vect" if scalar routine
ro 8.93
nump 7                                         c Size of cm array
                                              c (Number of material parameters)
numh 8                                         c Number of history variables
locb 5                                         c Bulk modulus position in cm array
locs 6                                         c Shear modulus position in cm array
c      cm(1)  cm(2)  cm(3)  cm(4)
para  1.170   0.350  0.004  0.001             c Material parameters (cm array)
c      cm(5)  cm(6)  cm(7)
      1.300   0.433   1.0
endmat

```

Table 7. Example of an material definition in LS-DYNA keyword format [2].

```

*DATABASE_EXTENT_BINARY
      8      0      0      0      0      0      0      0
      0      0      0      0      0      0
*MAT_USER_DEFINED_MATERIAL_MODELS
      1 8.9300000      42      7      8      0      5      6
      1      0
1.1700000 0.3500000 0.0040000 0.0010000 1.3000001 0.4330000 1.0000000

```

3 IMPLEMENTATION OF AN ELASTIC MODEL AS AN USER DEFINED MATERIAL MODEL

The problem used to verify our userdefined material models was the bar impact. The bar had a length of 0.6 cm and a diameter of 0.32 cm, see figure 2 for a schematic representation of the problem.

The bar was modeled using 84 constant stress 3-d solid elements. The boundary conditions consisted of no rotation around any of the three axis for both the upper and lower end surfaces and no displacement in the z-direction for the lower impacting surface. The initial conditions consisted of giving all nodes, except the nodes on the lower bottom surface which were fixed in z-direction, an initial velocity of 227 m/s. Double symmetry was used so that only a quarter of the crosssection was modelled.

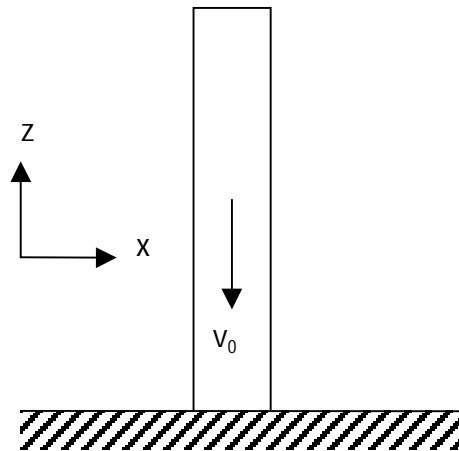


Figure 2. Schematic representation of the bar impact problem.

To validate the different versions of user material models comparisons were made for the acceleration history in the z-direction for the bar's top-center node, the pressure history for the bottom-center solid element and the internal energy of the projectile.

3.1 LS-DYNA Standard material routine f3dm1 (type 1)

This computation was used as reference case when validating the user defined material models. The LS-INGRID input file is found in Appendix D and in the following figure 3 you can see the bar in deformed configuration.

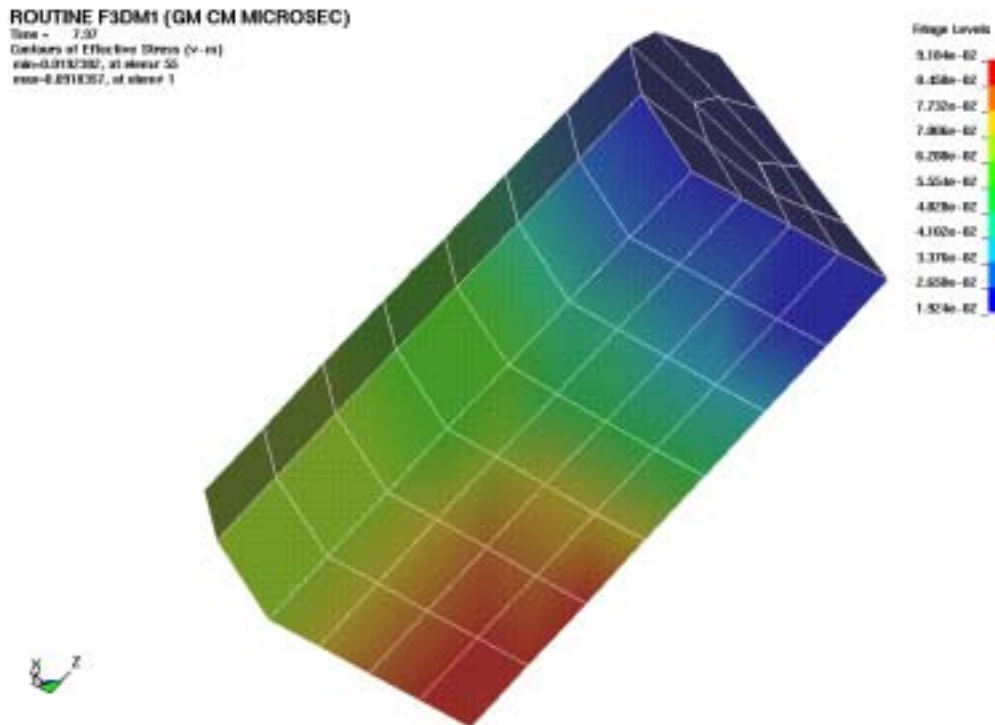


Figure 3. Effective stresses at time 8 micro seconds for standard material routine f3dm1.

3.2 Scalar userdefined material routine umat41

The source code is found in Appendix J and the LS-INGRID input file is found in Appendix E. In the following figures 4 to 6 comparisons are made with the standard material routine f3dm1. As can be seen from the graphs the two models gave exactly the same results.

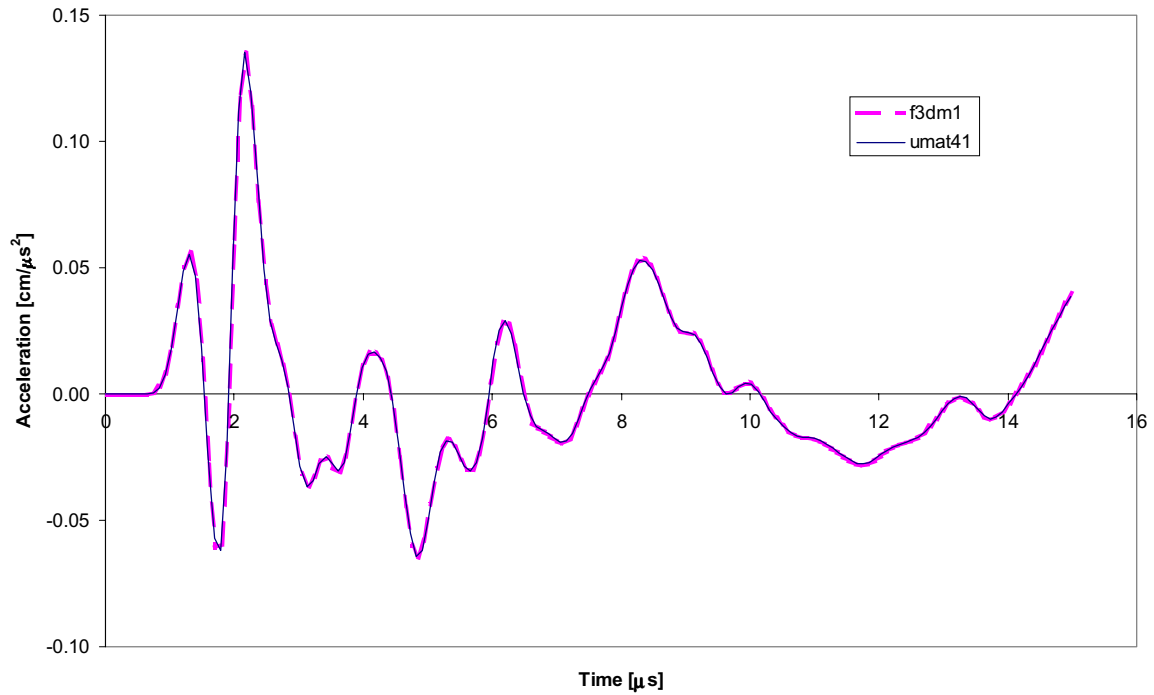


Figure 4. Reference acceleration for standard routine f3dm1 and user routine umat41.

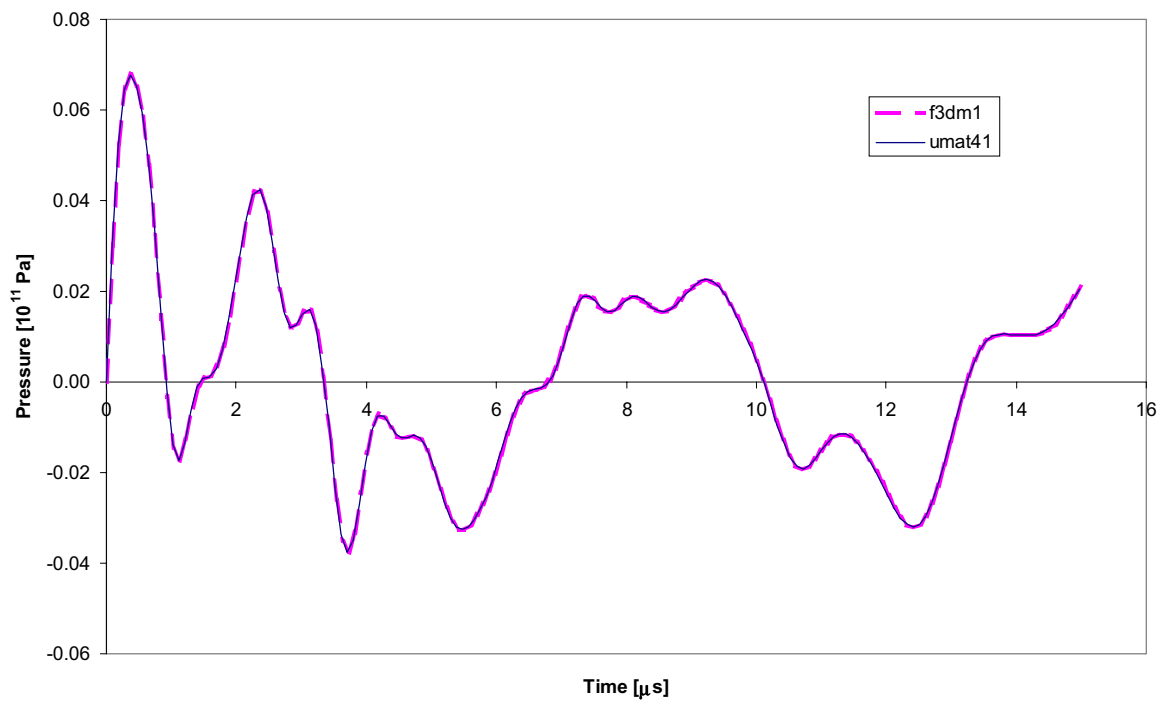


Figure 5. Reference pressure for standard routine f3dm1 and user routine umat41.

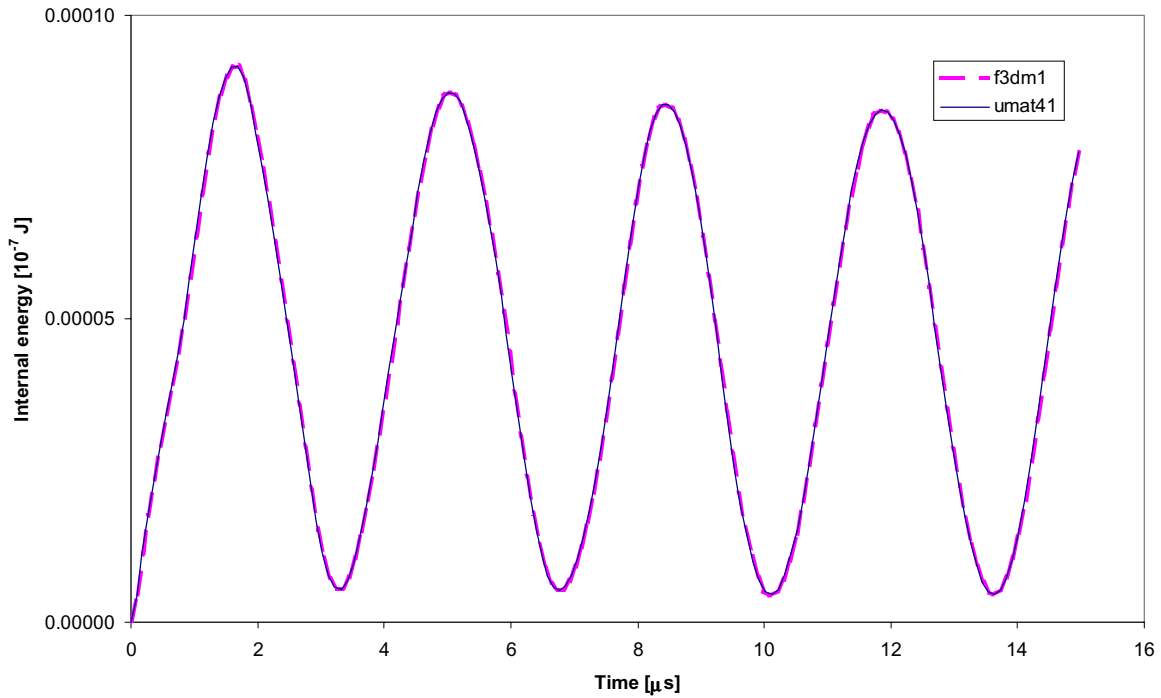


Figure 6. Reference internal energy for standard routine f3dm1 and user routine umat41.

3.3 Vectorized user defined material routine umat41v

The source code for this routine is found in Appendix K and the corresponding LS-INGRID input file in Appendix F. As you can see by comparing the routines the scalar version treats one element at a time (no loops), while the vectorized version treats a package containing a maximum of nlq elements at a time.

When working with a vectorized routine, it is essential to set the parameter nlq to the value corresponding to the machine we will use. At the top of the source code I have listed different types of machines and the corresponding value for nlq, and to underline the importance of this parameter I also give the values in the following table 8. As explained before, this is the maximum size of an element package treated at a time by our routine.

Table 8. Values for the parameter nlq

c For CRAY/T3E	use parameter (nlq=24)
c For CRAY/J90	use parameter (nlq=256)
c For CRAY/C90	use parameter (nlq=256)
c For CRAY/T90	use parameter (nlq=256)
c For CRAY/T90IEEE	use parameter (nlq=256)
c For DEC/Alpha/Compaq	use parameter (nlq=89)
c For EJ/VFF	use parameter (nlq=1033)
c For Hitachi	use parameter (nlq=130)
c For HP pa7x00	use parameter (nlq=33)
c For HP pa8x00	use parameter (nlq=65)
c For HP Exemplar	use parameter (nlq=128)
c For IBM	use parameter (nlq=128)
c For NEC/SX4	use parameter (nlq=1024)
c For SGI 4400	use parameter (nlq=32)
c For SGI R10000	use parameter (nlq=87)
c For SUN	use parameter (nlq=128)

More modifications needs to be done in our source code and this is due to lacking lines in the urmath-file. For the scalar format the file is correct, but not for the vectorized.

If you examin the lines in bold face in the urmath-file (Appendix A) you will see that the time step dt1 is not assigned a value in the case of a vectorized routine. We could do the appropriate changes in the urmath file, but adding the following line at the top of the vectorized routine solves also this problem and gives us acces to the timestep.

```
common /subtssloc/ dt1siz(nlq)
```

Further examination of the urmath file reveals also that the arrays sig1-sig6 and hist are not defined. To avoid problems with the memory allocation all the non-defined arrays should be given dummy names, for example as in Appendix K;

```
subroutine umat41v(cm,d1,d2,d3,d4,d5,d6,sig1dum,sig2dum,
. sig3dum,sig4dum,sig5dum,sig6dum,sigma,histdum,lft,llt,
. dt1dum,capa,etype,tt)
```

and in the user routine use instead;

```
c sigma(i,1) for sig1(i)
c sigma(i,2) for sig2(i)
c sigma(i,3) for sig3(i)
c sigma(i,4) for sig4(i)
c sigma(i,5) for sig5(i)
c sigma(i,6) for sig6(i)
c sigma(i,7) for hist(i,1)
c sigma(i,8) for hist(i,2)
c sigma(i,8) for hist(i,3) etc
```

The results from the vectorized routine is compared with the standard material routine f3dm1 in figures 7 to 9. As can be seen the results were as good as for the scalar version and the spreadsheet calculation gives the same magnitude in difference from the standard material computation.

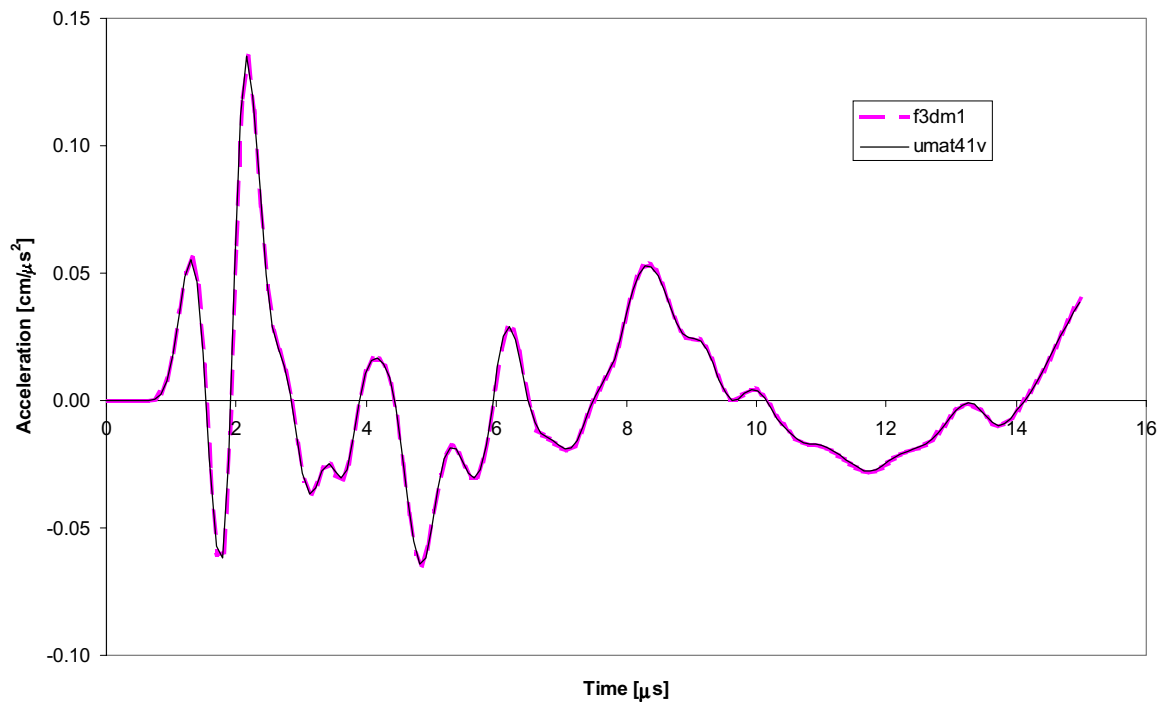


Figure 7. Reference acceleration for standard material routine f3dm1 and user routine umat41v.

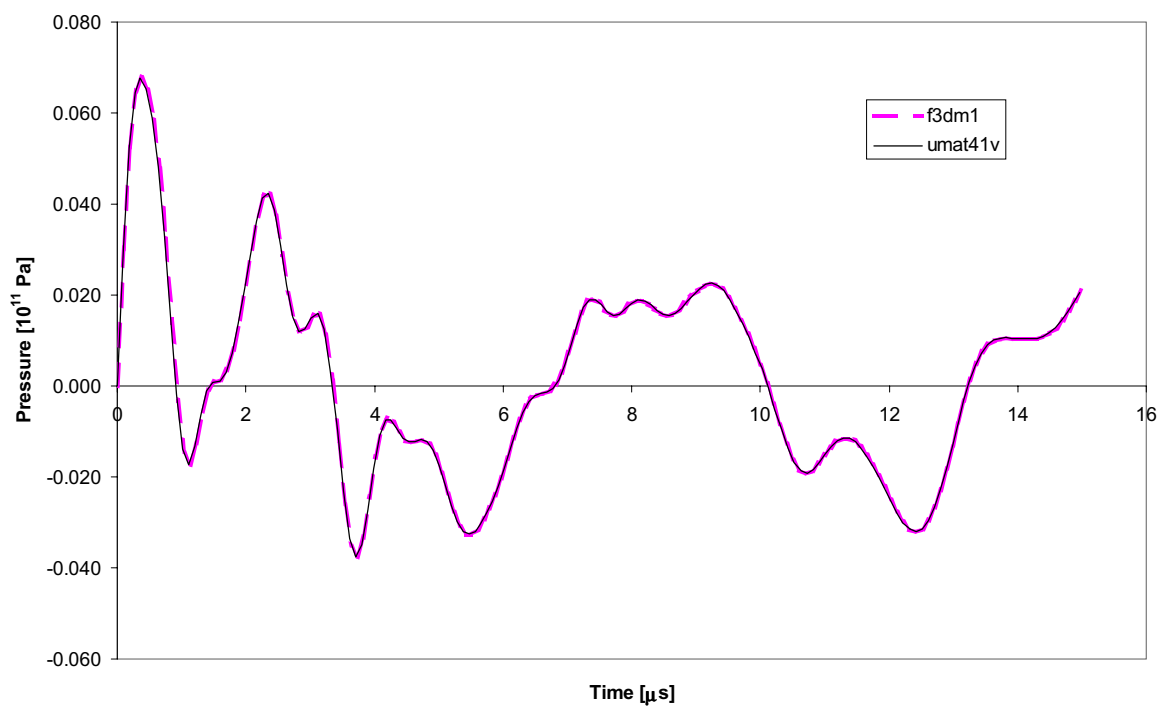


Figure 8. Reference pressure for standard material routine f3dm1 and user routine umat41v.

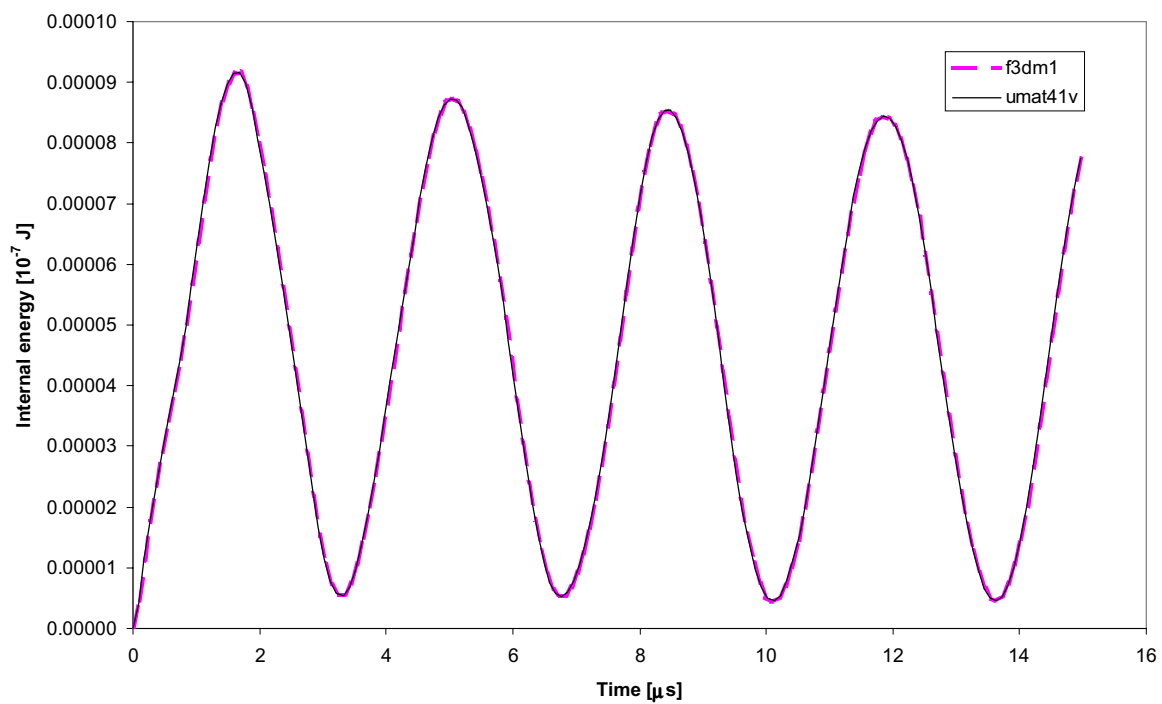


Figure 9. Reference pressure for standard and vectorized userdefined elastic material.

3.4 CPU cost

In the following figure comparisons are made for CPU element processing time for the different material routines. The same bar impact problem was used for these simulations but with a mesh consisting of 63 897 solid elements and the computations carried out to 50 μ s (see Appendix N).

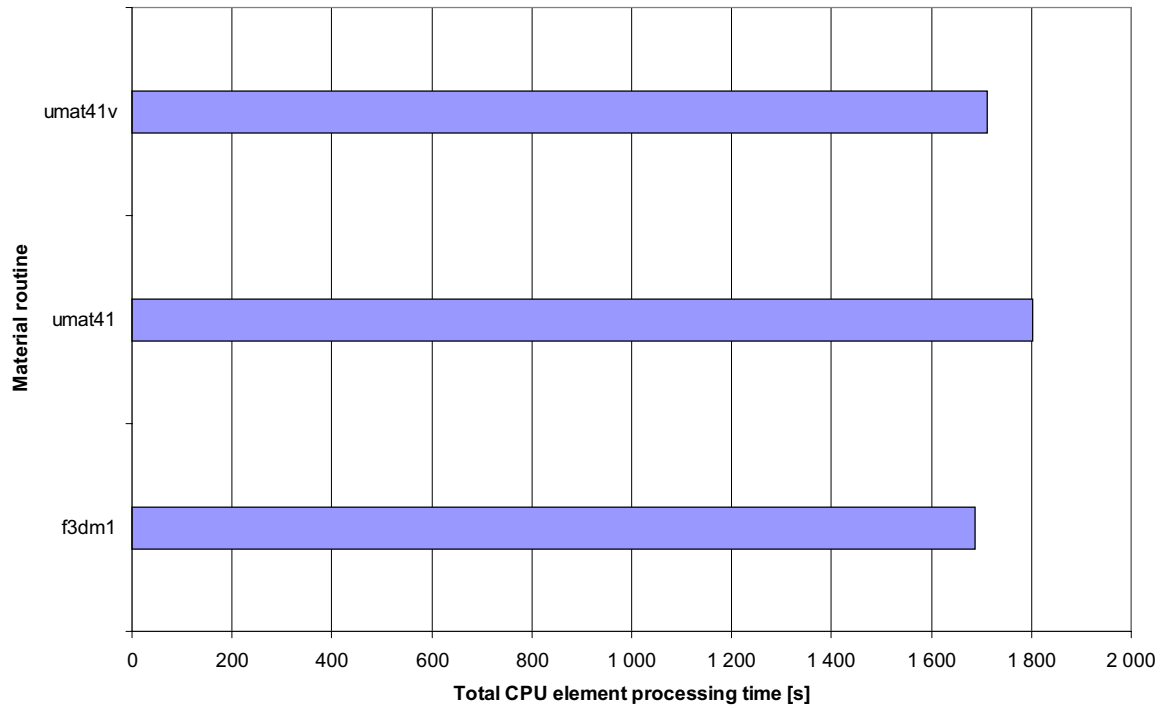


Figure 10. CPU element processing time for the three elastic material routines.

4 IMPLEMENTATION OF AN ELASTIC-PLASTIC MODEL WITH ISOTROPIC HARDENING AS AN USERDEFINED MATERIAL MODEL

To validate these user material routines comparisons were made for the acceleration history in the z-direction for the bar's top-center node, the effective plastic strain history for the bottom-center solid element and the internal energy

4.1 LS-DYNA Standard material routine f3dm3 (type 3)

This computation was also to be used as reference when validating the userdefined material models. The LS-INGRID input file is found in Appendix G and this model was used with the option of isotropic hardening and the final deformed configuration from the computation is shown in the following figure.

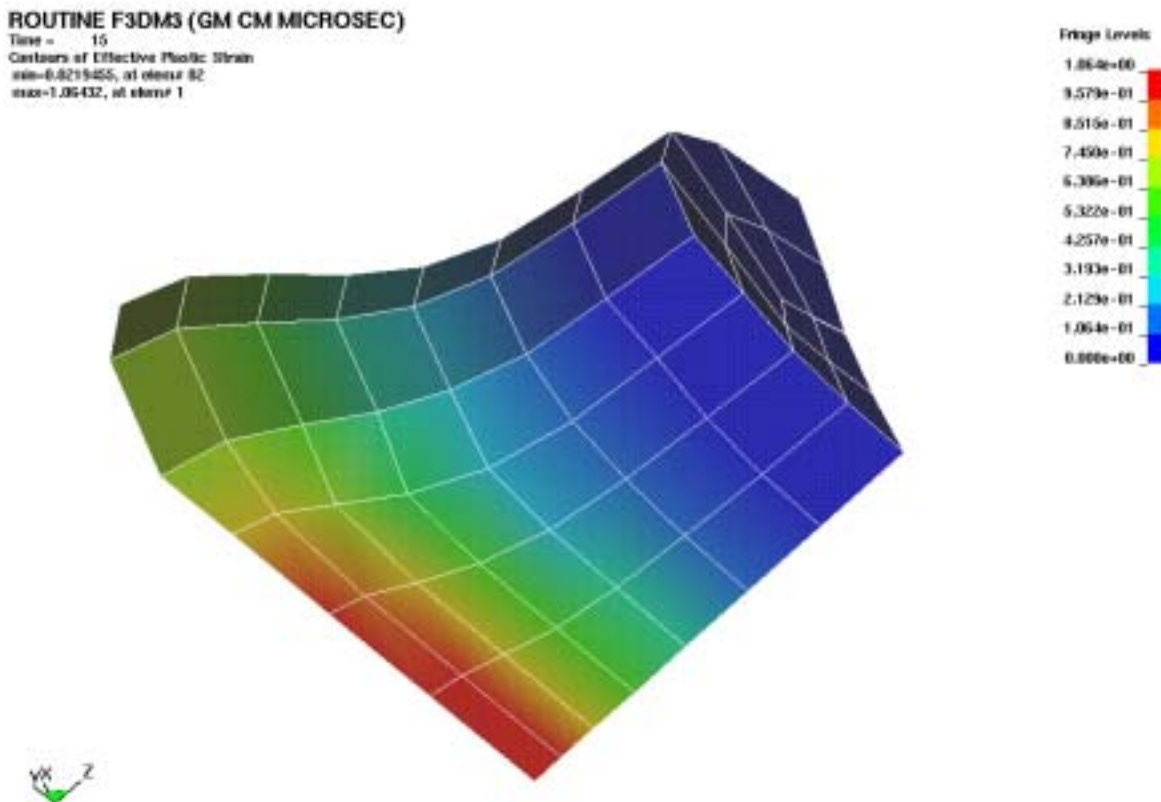


Figure 11. Effective stresses at time 15 microseconds for standard material routine f3dm3.

4.2 Scalar userdefined material routine umat42

The source code is found in Appendix L and the LS-INGRID input file is found in Appendix H. In the figures 12 to 14 comparisons are made with the standard material type 3. As can be seen from the graphs the two models gave exactly the same results.

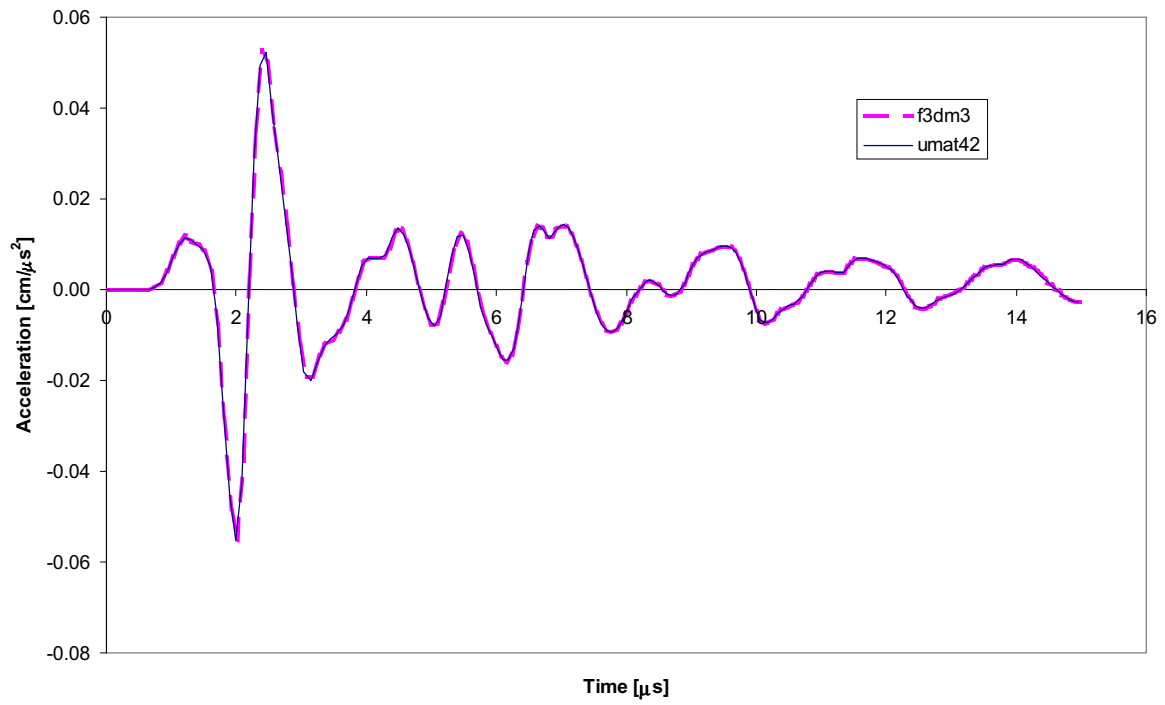


Figure 12. Reference acceleration for standard and scalar userdefined elastic-plastic material.

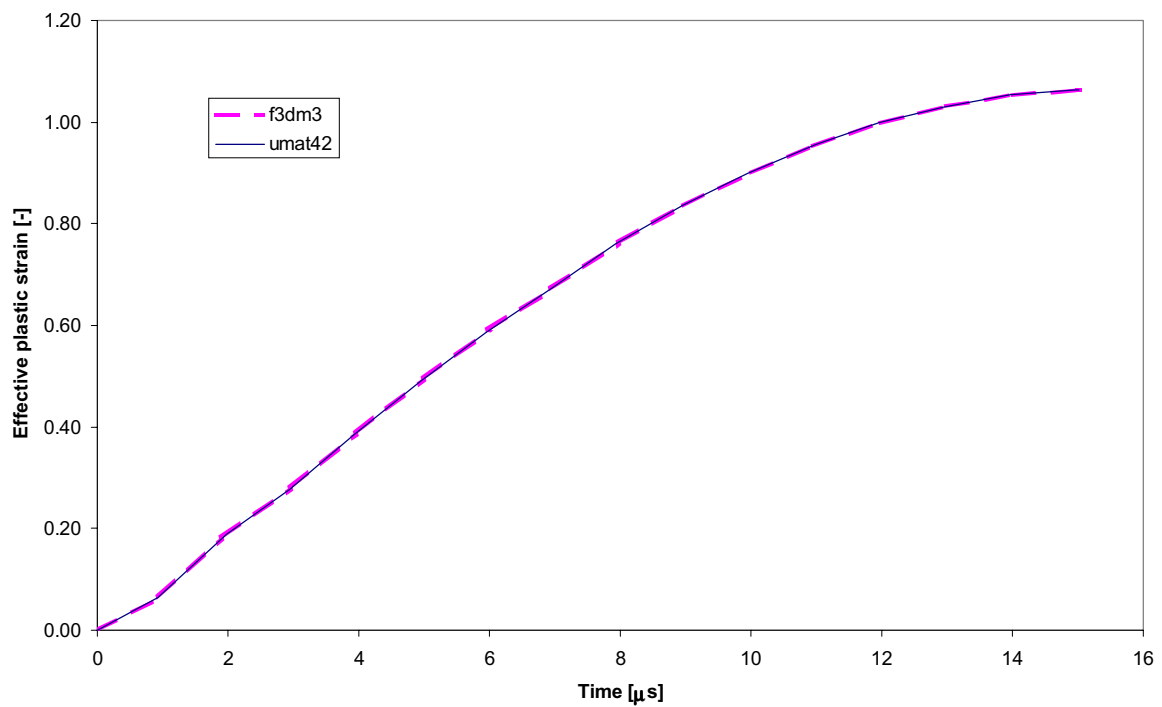


Figure 13. Reference effective plastic strain for standard and scalar userdefined elastic-plastic material.

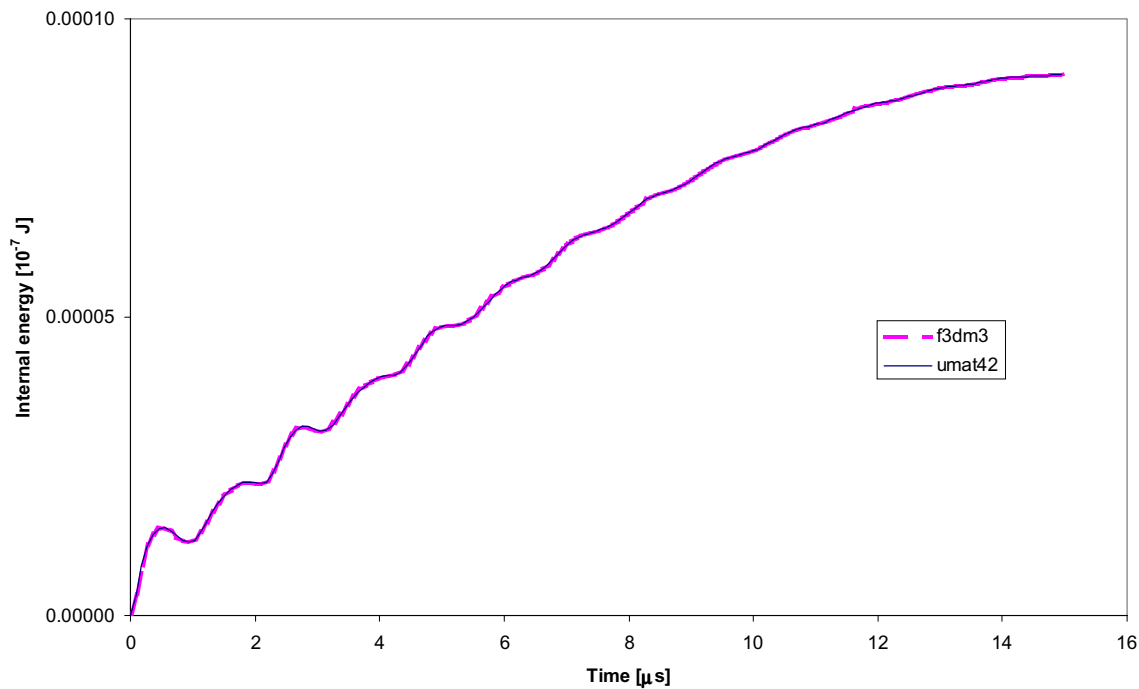


Figure 14. Reference internal energy for standard and scalar userdefined elastic-plastic material.

4.3 Vectorized user defined material routine umat42v

When using history variables in a vectorized routine another problem arises and this is also due to lacking lines in the urmath file. In the case of a vectorized routine the argument hist is never defined, but an examination of the urmath file reveals that the history variables are stored after the stresses in the sigma array in the following manner.

```
sigma(i,7)=hist(i,1)
sigma(i,8)=hist(i,2)
sigma(i,9)=hist(i,3)
etc
```

In the first history variable you always have to store the effective plastic strain, otherwise do not use it. The source code for this routine is found in Appendix M where you can verify these changes and the corresponding LS-INGRID input file is found in Appendix I.

Note: During the trial and error process with this problem we found that when using two user defined materials with same material type number (42 – 42v), the first material was chosen regardless of the material identification number for the part in the keyword file. Thus, do not use a scalar and a vectorized routine with the same number in a computation.

The results from this simulation are compared with the ones from the standard material type 3.in figures 15 to 17. As can be seen from the graphs the two models gave exactly the same results.

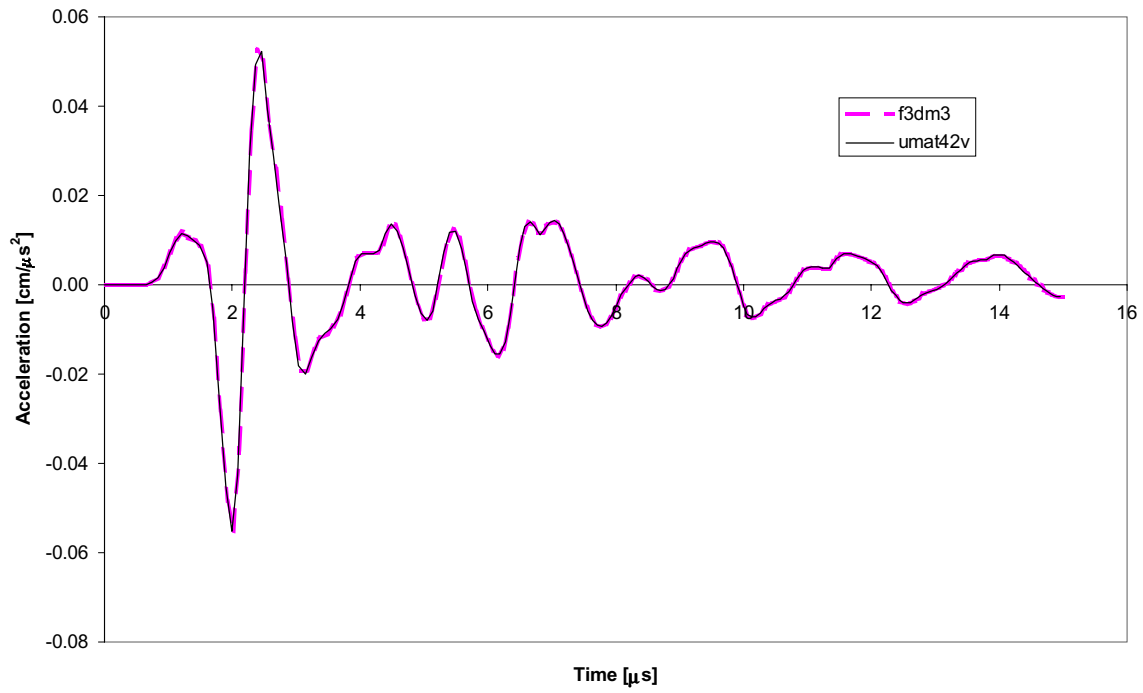


Figure 15. Reference acceleration for standard and vectorized userdefined elastic-plastic material.

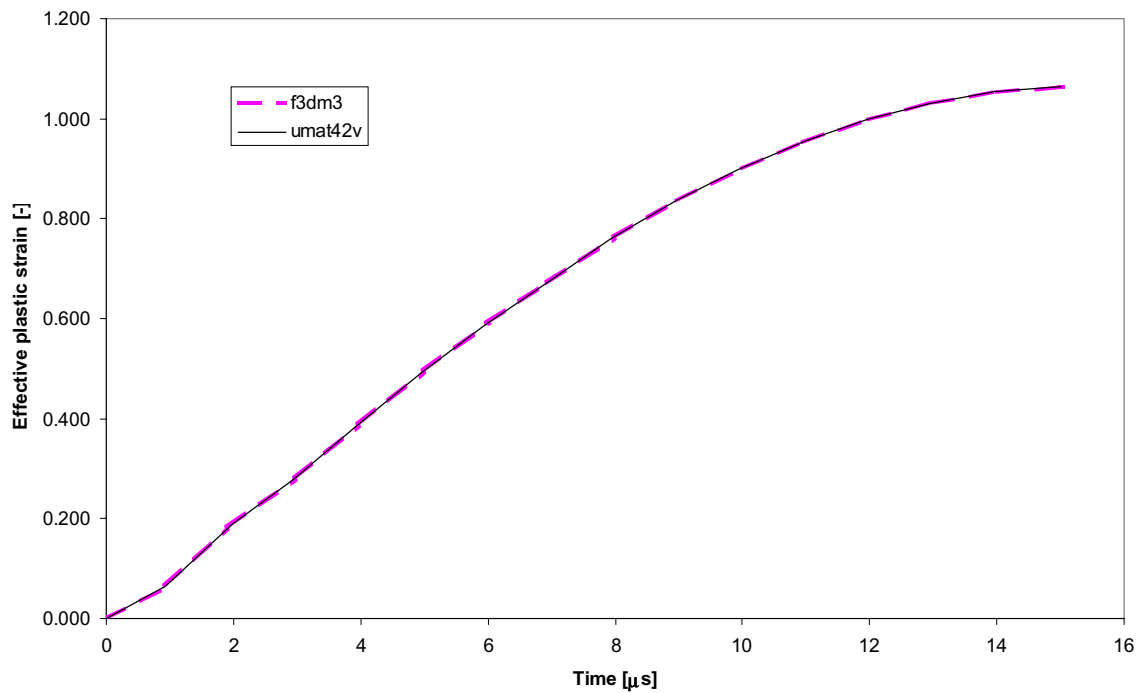


Figure 16. Reference effective plastic strain for standard and vectorized userdefined elastic-plastic material.

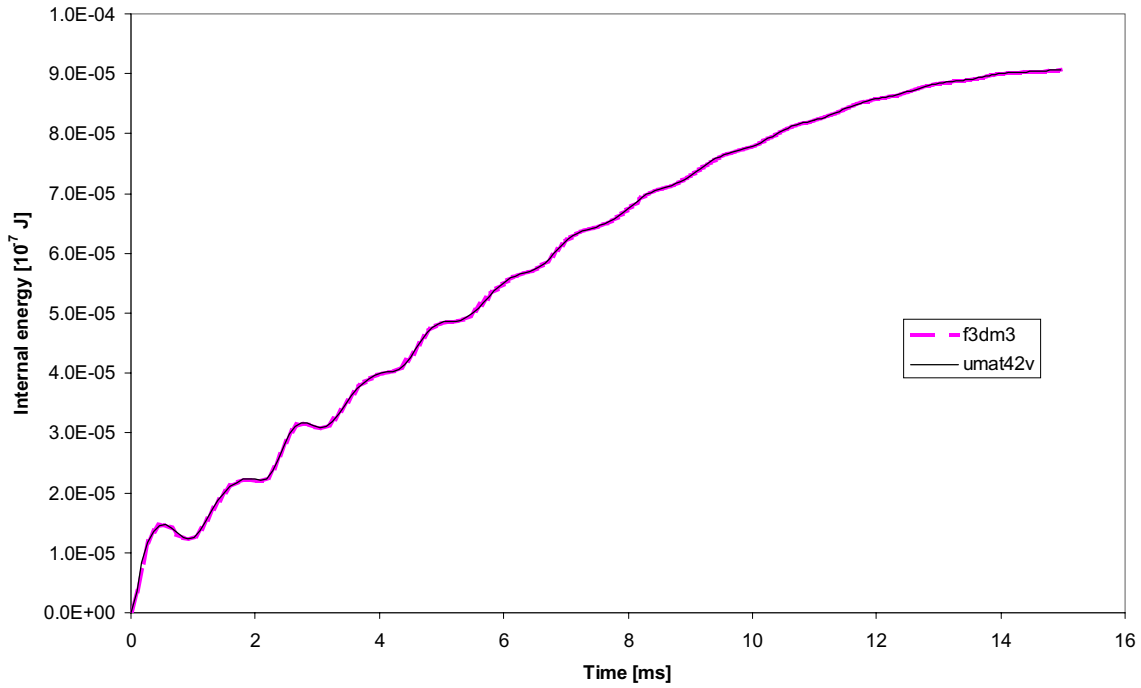


Figure 17. Reference internal energy for standard and vectorized userdefined elastic-plastic material.

4.4 CPU cost

In the following figure comparisons are made for CPU element processing time for the different material routines. The same bar impact problem was used for these simulations but with a mesh consisting of 63 897 solid elements and the computations carried out to 50 μ s (see Appendix N).

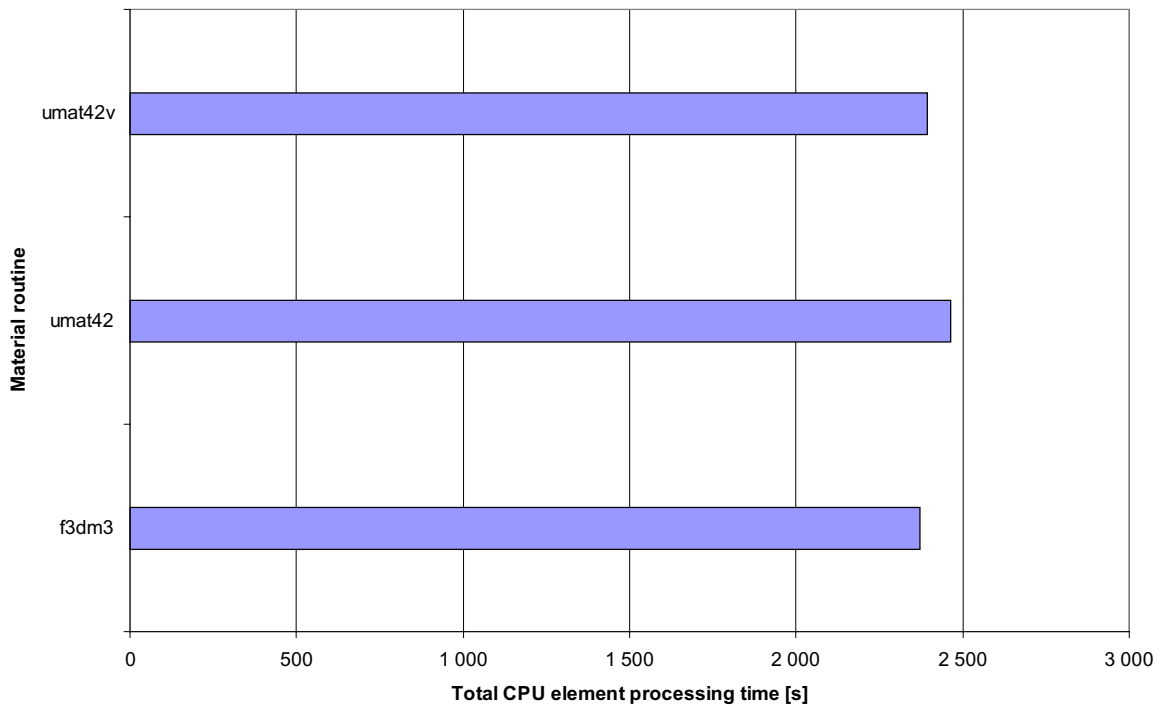


Figure 18. CPU element processing time for the three elastic-plastic material routines.

5 MISCELLANEOUS OPTIONS

5.1 Equation of state

For a user defined material the energy increment is calculated according to;

```

einc(i)=(d1(i)*sigma(i,1)+d2(i)*sigma(i,2)+d3(i)*sigma(i,3)
.      +d4(i)*sigma(i,4)+d5(i)*sigma(i,5)
.      +d6(i)*sigma(i,6)+dd(i)*bqs(i))*dtlsiz(i)
(...)
call umat43v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dtlsiz,capa,'brick',tt)
(...)
einc(i)=(d1(i)*sigma(i,1)+d2(i)*sigma(i,2)+d3(i)*sigma(i,3)
.      +d4(i)*sigma(i,4)+d5(i)*sigma(i,5)
.      +d6(i)*sigma(i,6))*dtlsiz(i)+einc(i)
(...)
end
(...)
call lieupd
(...)
ries(i)=ries(i)+.5*volnew(i)*(einc(i)+dtlsiz(i)*dde(i)*qp(i))

```

where the products containing the bqs and qp arrays are the bulk viscosity energies. The ries array is the element energy increment. If the material routine umat43v incorporates an equation of state the energy evolves in a different manner and the way of calculating the energy increment has to be changed. This could be done by changing the appropriate lines in the urmath and lieupd routines or try to trick LS-DYNA by adding the following lines (in bold face) to our routine:

```

subroutine umat43v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
(...)
common /aux9loc/ vlrhodummy(nlq),volnew(nlq)
(...)
sig1old(i)=sig1(i)
sig2old(i)=sig2(i)
sig3old(i)=sig3(i)
sig4old(i)=sig4(i)
sig5old(i)=sig5(i)
sig6old(i)=sig6(i)
c Subtract energy increment calculated in urmath except the bulk viscosity energy.
einc(i)=einc(i)-dt1*(
. d1(i)*sig1old(i)+d2(i)*sig2old(i)+d3(i)*sig3old(i)+
. d4(i)*sig4old(i)+d5(i)*sig5old(i)+d6(i)*sig6old(i))
c Calculate new pressure from equation of state and deviatoric stresses.
(...)
c Divide all energy by .5*volnew(i) according to lieupd routine.
c Add user deviatoric energy part.
einc(i)=einc(i)+.25*(hist(i,6)+volnew(i))*dt1*(
. d1(i)*(sig1old(i)+po(i)+sig1(i)+pnew(i))+
. d2(i)*(sig2old(i)+po(i)+sig2(i)+pnew(i))+
. d3(i)*(sig3old(i)+po(i)+sig3(i)+pnew(i))+
. d4(i)*(sig4old(i)+sig4(i))+
. d5(i)*(sig5old(i)+sig5(i))+
. d6(i)*(sig6old(i)+sig6(i)))/(.5*volnew(i))
c Add volumetric energy part from equation of state.
c (LS-DYNA theory manual, section 15.2)
einc(i)=einc(i)-.5*(volnew(i)-hist(i,6))*(po(i)+
. pnew(i))/(.5*volnew(i))
c Subtract energy increment that will be added in urmath.
einc(i)=einc(i)-dt1*(
. d1(i)*sig1(i)+d2(i)*sig2(i)+d3(i)*sig3(i)+
. d4(i)*sig4(i)+d5(i)*sig5(i)+d6(i)*sig6(i))
return
end

```

5.2 Erosion

The general erosion option *MAT_ADD_EROSION can be used together with an user defined material routine, but it is also possible to include an erosion algorithm within the user defined material subroutine. This allows us to define our own erosion criteria and to do this we first have to add the following declarations to the code:

```
logical failur, failgl  
common/failcm/failur, failgl, mtfail(1000)  
common/failcmloc/ifail(nlq)
```

In the following example of an erosion loop the variable hist(i,2) contains the accumulated effective plastic strain and fs is the limit for element deletion.

```
if (fs.ne.0.0.and.hist(i,2).gt.fs) then  
  sig1(i)=0  
  sig2(i)=0  
  sig3(i)=0  
  sig4(i)=0  
  sig5(i)=0  
  sig6(i)=0  
  hist(i,2) = fs  
  failur=.true.  
  ifail(inum)=1  
endif
```

This erosion algorithm works no matter the value for the erosion flag in the keyword format card *MAT_USER_DEFINED_MATERIAL_MODELS. How to use this flag has not been investigated here.

5.3 Load curves

At the time when this work was done it was not possible to use load curves with user defined routines. A solution is to include the load curve points in your material parameter array (cm) and to add an algorithm in your code accounting for strain rate effects etc. The maximum size of the material parameter array is for the time being restricted to 48.

6 CONCLUSIONS

For a scalar material routine the implementation is straight forward, but when using a vectorized routine the lacking of command lines in urmath file forces the user to get an insight in the user routine code context and do appropriate modifications in the user routine. It should however not be a problem for LSTC to modify the urmath file. The comparisons made for the CPU element processing time shows that the vectorized format is slightly faster than the scalar format.

The use of common blocs to pass arguments should be avoided if possible. The structure of these blocs changes often with new versions of LS-DYNA and often you will need to add machine specific command lines to your code for the compiling and linking to be succesful. The resulting code then works only for the current LS-DYNA version and the current machine you are using. If no common blocs are used and the urmath file is modified your material routine will work on any system.

Future work with user defined material routines would include investigations on how to use the erosion flag, the deformation gradient option and to implement a material routine with an equation of state and an user erosion algorithm. It would also be preferable if the LSTC changed the code so that standard options like equations of state and load curves could be used in the user routines, as in for example Autodyn, eliminating the need of including such options in your code.

REFERENCES

1. LSTC (1998). *LS-INGRID: A pre-processor and three dimensional mesh generator for the programs LS-DYNA, LS-NIKE3D and TOPAZ3D, version 3.5*. Livermore Software Technology Corporation, LSTC report.
2. LSTC (1999). *LS-DYNA keyword user's manual, nonlinear dynamic analysis of structures, version 950*. Livermore Software Technology Corporation, LSTC report.

APPENDICES

Appendix A

```
      subroutine urmath(cm,bqs,mt,lft,llt)
c include "double.inc"
c - Variable declarations
c
c For CRAY/T3E          use 24
c For CRAY/J90         use 256
c For CRAY/C90         use 256
c For CRAY/T90         use 256
c For CRAY/T90IEEE     use 256
c For DEC/Alpha        use 89
c For EJ/VFF           use 1033
c For Hitachi          use 130
c For HP pa7x00        use 33
c For HP pa8x00        use 65
c For HP Exemplar     use 128
c For IBM              use 128
c For NEC/SX4          use 1024
c For SGI 4400         use 32
c For SGI R10000       use 87
c For SUN              use 128
c
c      parameter (nlq=87)
c*****
c|  livermore software technology corporation  (lstc)
c|  -----
c|  copyright 1987,1988,1989 john o. hallquist, lstc
c|  all rights reserved
c*****
C_TASKCOMMON (aux2loc)
C_TASKCOMMON (aux14loc)
C_TASKCOMMON (aux18loc)
C_TASKCOMMON (aux33loc)
C_TASKCOMMON (aux35loc)
C_TASKCOMMON (bk36loc)
C_TASKCOMMON (aux40loc)
C_TASKCOMMON (aux41loc)
C_TASKCOMMON (soundloc)
C_TASKCOMMON (subtssloc)
      common/bk28/summss,xke,xpe,tt,xte0,erodeke,erodeie
      common/aux2loc/d1(nlq),d2(nlq),d3(nlq),d4(nlq),d5(nlq),d6(nlq),
1 wzzdt(nlq),wyydt(nlq),wxzdt(nlq),einc(nlq)
      common/bk07/n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,
1 n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,
2 n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,
3 n46,n47,n48,n49,n50,n51,n52,n53,n54,n55,n56,n57,n58,n59,n60,n61,
4 n62,n63,n64,n65,n66,n67,n68,n69,n70,n71,n72,n73,n74,n75,n76,n77,
5 n78,n79,locend,iname,idummy
      common/bk19/nconst(200),lenma,ncneos(20),mtecom(200)
      common/aux14loc/
&sigma(nlq,7),
&q11(nlq),q12(nlq),q13(nlq),q31(nlq),
&q32(nlq),q33(nlq),hist(nlq,54),q21(nlq),q22(nlq),q23(nlq)
      common/aux18loc/dd(nlq),def(nlq),ddq(nlq)
      common/aux33loc/
1 ix1(nlq),ix2(nlq),ix3(nlq),ix4(nlq),ixs(nlq,4),mxt(nlq)
      common/bk26/nintcy
      common/aux35loc/rhoa(nlq),cb(nlq),davg(nlq),p(nlq)
      common/aux40loc/
1 a11(nlq),a12(nlq),a13(nlq),a21(nlq),a22(nlq),a23(nlq),
2 a31(nlq),a32(nlq),a33(nlq),z11(nlq),z12(nlq),z13(nlq),
3 z21(nlq),z22(nlq),z23(nlq),z31(nlq),z32(nlq),z33(nlq)
      common/aux41loc/qq1(nlq),cbb(nlq),aj1(nlq)
      common/soundloc/ss(nlq),sndsp(nlq),diagm(nlq),sarea(nlq),
```

```

. dxl(nlq)
common/umatss/ibulkp(60),nusrcn(60),iorien(60),nusrmt,ishrmp(60),
.   ivectr(60),ihyper(60)
common/subtssloc/dt1siz(nlq)
common/bk36loc/index
dimension  cm(*),bqs(*),eps(6),sig(6),hsv(100)
REAL mlt1,mlt2
include 'mema.inc'

c
c   compute deformation gradients in undeformed configuration
c   to account for thermal effects, F must be integrated as in shl40s
c   second order accurate integration of Fdot=L*F
c
c   compute deformation gradients in undeformed configuration
cdjb to account for thermal effects, F must be integrated as in shl40s
cdjb second order accurate integration of Fdot=L*F
c
c   if (ihyper(mt).ne.0) then
c
c       if (tt.eq.0.0.and.nintcy.eq.0) then
c           if11=nconst(mt)-8
c           if21=nconst(mt)-7
c           if31=nconst(mt)-6
c           if12=nconst(mt)-5
c           if22=nconst(mt)-4
c           if32=nconst(mt)-3
c           if13=nconst(mt)-2
c           if23=nconst(mt)-1
c           if33=nconst(mt)
c           do 1 i=lft,llt
c               sigma(i,if11)=1.0
c               sigma(i,if21)=0.0
c               sigma(i,if31)=0.0
c               sigma(i,if12)=0.0
c               sigma(i,if22)=1.0
c               sigma(i,if32)=0.0
c               sigma(i,if13)=0.0
c               sigma(i,if23)=0.0
c               sigma(i,if33)=1.0
1 continue
c           endif
c
c
c       call integf(sigma(1,if11),sigma(1,if21),sigma(1,if31),
c   &               sigma(1,if12),sigma(1,if22),sigma(1,if32),
c   &               sigma(1,if13),sigma(1,if23),sigma(1,if33),lft,llt)
c
c   endif
c
c   global to local element axes transformation matrix
c
c   ivect=ivectr(mt)
c   if (iorien(mt).ne.0) then
c
c       call lcs22(lft,llt)
c
c   global to material axes transformation matrix
c   q is the transformation matrix from element axes to material axes
c
c   do 10 i=lft,llt
c       q21(i)=q32(i)*q13(i)-q12(i)*q33(i)
c       q22(i)=q33(i)*q11(i)-q13(i)*q31(i)
c       q23(i)=q31(i)*q12(i)-q11(i)*q32(i)
c       z11(i)=a11(i)*q11(i)+a21(i)*q12(i)+a31(i)*q13(i)
c       z12(i)=a12(i)*q11(i)+a22(i)*q12(i)+a32(i)*q13(i)
c       z13(i)=a13(i)*q11(i)+a23(i)*q12(i)+a33(i)*q13(i)
c       z21(i)=a11(i)*q21(i)+a21(i)*q22(i)+a31(i)*q23(i)
c       z22(i)=a12(i)*q21(i)+a22(i)*q22(i)+a32(i)*q23(i)

```

```

z23(i)=a13(i)*q21(i)+a23(i)*q22(i)+a33(i)*q23(i)
z31(i)=a11(i)*q31(i)+a21(i)*q32(i)+a31(i)*q33(i)
z32(i)=a12(i)*q31(i)+a22(i)*q32(i)+a32(i)*q33(i)
z33(i)=a13(i)*q31(i)+a23(i)*q32(i)+a33(i)*q33(i)
10 continue
c
c   transform stresses and strains to local system
c
   call ldsm22(lft,llt)
   call ldem22(lft,llt)
   endif
c
c   process elements
c
   mx=48*(mxt(lft)-1)
   gm=cm(mx+ishrmp(mt))
   bk=cm(mx+ibulkp(mt))
   nc=nconst(mt)+1
   ss(lft)=bk+4.*gm/3.
   do 20 i=lft,llt
   cb(i)=ss(lft)
   einc(i)=(d1(i)*sigma(i,1)+d2(i)*sigma(i,2)+d3(i)*sigma(i,3)
.         +d4(i)*sigma(i,4)+d5(i)*sigma(i,5)
.         +d6(i)*sigma(i,6)+dd(i)*bqs(i))*dt1siz(i)
20 continue
   mte40=mt-40
   if (ivect.eq.0) then
   do 90 i=lft,llt
   index =i
   dt1   =dt1siz(i)
   eps(1)=d1(i)*dt1
   eps(2)=d2(i)*dt1
   eps(3)=d3(i)*dt1
   eps(4)=d4(i)*dt1
   eps(5)=d5(i)*dt1
   eps(6)=d6(i)*dt1
   sig(1)=sigma(i,1)
   sig(2)=sigma(i,2)
   sig(3)=sigma(i,3)
   sig(4)=sigma(i,4)
   sig(5)=sigma(i,5)
   sig(6)=sigma(i,6)
   hsv(1)=sigma(i,7)
   if (nc.gt.1) then
   if (iorien(mt).ne.0) then
   do 30 j=2,nc
   hsv(j)=hist(i,j-1)
30 continue
   else
   do 40 j=2,nc
   hsv(j)=sigma(i,6+j)
40 continue
   endif
   endif
c
c   call user developed subroutines here
c
   go to (41,42,43,44,45,46,47,48,49,50), mte40
41 call umat41 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
c 41 call umat41 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt,
c   . hsv(nc-8))
   go to 60
42 call umat42 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt,
. crv)
   go to 60
43 call umat43 (cm(mx+1),sig,hsv,dt1,tt,i,d1,d2,d3,d4,d5,d6)
   go to 60
44 call umat44 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)

```

```

    go to 60
45 call umat45 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
    go to 60
46 call umat46 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
    go to 60
47 call umat47 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
    go to 60
48 call umat48 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
    go to 60
49 call umat49 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
    go to 60
50 call umat50 (cm(mx+1),eps,sig,hsv,dt1,capa,'brick',tt)
c
60 if (nc.gt.1) then
    if (iorien(mt).ne.0) then
        do 70 j=2,nc
            hist(i,j-1)=hsv(j)
        70 continue
    else
        do 80 j=2,nc
            sigma(i,6+j)=hsv(j)
        80 continue
    endif
    endif
    sigma(i,1)=sig(1)
    sigma(i,2)=sig(2)
    sigma(i,3)=sig(3)
    sigma(i,4)=sig(4)
    sigma(i,5)=sig(5)
    sigma(i,6)=sig(6)
    sigma(i,7)=hsv(1)
c
c
90 continue
c
    else
c
    if (mt.eq.41) then
        call umat41v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.42) then
        call umat42v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.43) then
        call umat43v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.44) then
        call umat44v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.45) then
        call umat45v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.46) then
        call umat46v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt)
    elseif (mt.eq.47) then
        call umat47v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt,
        . ipt)
    elseif (mt.eq.48) then
        call umat48v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt,
        . ipt,a(n8),a(n9))
    elseif (mt.eq.49) then
        call umat49v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,
        . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1siz,capa,'brick',tt,
        . ipt)
    elseif (mt.eq.50) then
        call umat50v(cm(mx+1),d1,d2,d3,d4,d5,d6,sig1,sig2,

```

```

. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dtlsiz,capa,'brick',tt,
. ipt)
endif
endif
c
do 100 i=lft,llt
einc(i)=(d1(i)*sigma(i,1)+d2(i)*sigma(i,2)+d3(i)*sigma(i,3)
.         +d4(i)*sigma(i,4)+d5(i)*sigma(i,5)
.         +d6(i)*sigma(i,6))*dtlsiz(i)+einc(i)
100 continue
c
if (iorien(mt).ne.0) then
c
c global stresses
c
call gblm22 (lft,llt)
c
endif
c
return
end

```

Appendix B

```
#
# Makefile for User Defined Options
# for DEC/ALPHA
#
# using f90 5.2 ft2
# had to still change define.inc
# added omp threadprivate and system() workaround to adios
# OMP vars:
# setenv MP_DEBUG 1
#
#
SHELL      = /bin/csh
LSDYNA     = ls950
CODESHOME  = /alpha4_5/codes
ARCH       = -tune ev5

PROGRAMA = LS-DYNA_950c_compaq_40_pa_userdef

# c compiler
CC = /bin/cc
CFLAGS = -c -O $(ARCH) -DALPHA

# cpp compiler
CPP = $(CC) -P -DALPHA -DOPENMP -DALPHAONLY
SED = mv

# f90 serial version
FC = f90 -omp -v -automatic -align records -align dcommons
FFLAGS = -convert big_endian -c -O4 $(ARCH)

# loader
LD = $(FC)
LDFLAGS = -O3 -convert big_endian -lX11 -lXmu -lc

# ANSYS
ANSYSLIB = ansys.a

# DIGLIB graphics library
DIGLIB = diglib.o gdxwin.o gdiris.o \
        gdsun2.o gdphigs.o gdcjet.o \
        gdhpdj.o gdhplj.o gdhppj.o gdPEX.o

# MADYMO 5.3 close coupling
NOMADY = adummy_madymo.o

# CAL3D coupling
NOCAL = adummy_cal3d.o

# USA coupling
NOUSA = adummy_usa.o

# LSTCIO
SECUR = sec.o

# Default compilation
.SUFFIXES :
.SUFFIXES :.o .c .F .f
.c.o      :
           $(CC) $(CFLAGS) $<
.f.o      :
           $(FC) $(FFLAGS) $<

# Main program
MAIN = dynm.o banner.o
```

```

OBJ1 = \
adummy_gl.o \
adummy_oasys.o \
adummy_user.o \
atemp.o \
dyn0.o \
dyn1.o dyn10.o dyn11.o dyn12.o dyn13.o dyn14.o dyn16.o \
dyn15.o dyn17.o dyn17a.o dyn17b.o dyn17s.o \
dyn18.o dyn18a.o dyn18b.o \
dyn2.o dyn20a.o dyn20b.o dyn20c.o dyn20d.o dyn20e.o \
dyn21.o dyn22t.o dyn22.o dyn23.o \
dyn24.o dyn25.o dyn26.o dyn27.o \
dyn28a.o dyn28b.o dyn28c.o dyn29.o \
dyn3.o dyn30.o dyn31.o dyn32.o dyn33.o dyn34.o dyn35.o \
dyn36.o dyn37.o dyn38.o dyn39.o \
dyn4.o dyn40.o dyn41.o dyn43.o dyn44.o dyn45.o \
dyn46.o dyn47.o dyn48.o dyn49.o \
dyn5a.o dyn5.o dyn50.o \
dyn51a.o dyn51b.o gebodc.o \
dyn6.o \
dyn7.o \
dyn8.o \
dyn9.o \
dynka.o dynkb.o dynkc.o dynkd.o \
dynpc.o dynpe.o dynu.o \
fem3d.o \
fullv.o \
ge.o \
gcfe.o \
maze9.o vda.o umat41.o umat41v.o umat42.o umat42v.o umat43v.o

```

```

OBJ2 = \
stencil.o timehist.o

```

```

OBJ3 = \
etime.o \
lalloc.o \
fqsrt.o \
jumpbug.o \
vda_parse.o

```

```

OBJECT = $(OBJ1) $(OBJ2) $(OBJ3) $(DIGLIB)

```

```

$(PROGRAMA): $(MAIN) $(NOCAL) $(SECUR) $(NOMADY) $(NOUSA) $(OBJECT)
$(LD) $(MAIN) $(NOCAL) $(SECUR) $(NOMADY) $(NOUSA) \
$(OBJECT) $(ANSYSLIB) lstcio.a $(LDFLAGS) -o $@

```

Appendix C

(Only Part of the file dyn21.f)

```
c      subroutine umat41 (cm,eps,sig,hisv,dt1,capa,etype,time)
c      implicit real*8 (a-h,o-z)                                double   c
c      isotropic elastic material
c
c      character*(*) etype
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      return
c      end
c      subroutine umat42 (cm,eps,sig,hisv,dt,capa,etype,time,crv)
c      implicit real*8 (a-h,o-z)                                double
c
c      elastic-plastic material with isotropic and kinematic hardening
c
c      dimension  cm(*),eps(*),sig(*),hisv(*),crv(101,2,*)
c      character*(*) etype
c      return
c      end
c      subroutine umat43 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat44 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat45 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat46 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat47 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat48 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
c      return
c      end
c      subroutine umat49 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c      character*(*) etype
c
```

```

return
end
subroutine umat50 (cm,eps,sig,hisv,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
dimension cm(*),eps(*),sig(*),hisv(*)
character*(*) etype
c
return
end
c subroutine umat41v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
c . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt) double
c implicit real*8 (a-h,o-z)
c character*(*) etype
c return
c end
c subroutine umat42v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
c . sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt) double
c implicit real*8 (a-h,o-z)
c character*(*) etype
c return
c end
subroutine umat43v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
subroutine umat44v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
subroutine umat45v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
subroutine umat46v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
subroutine umat47v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
subroutine umat50v(cm,d1,d2,d3,d4,d5,d6,sig1,sig2,
. sig3,sig4,sig5,sig6,sigma,hist,lft,llt,dt1,capa,etype,tt)
c implicit real*8 (a-h,o-z) double
character*(*) etype
return
end
.
.
.

```

Appendix D

Bar impact problem, material routine f3dm1 (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 3 5 ;
  1 3 5 ;
  1 8;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;
mate 1
end

c Define material properties
mat 1 1 ro 8.93 e 1.17 pr 0.35 endmat

end
tp 0.001
cont
```

Appendix E

Bar impact problem, material type umat41 (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 3 5 ;
  1 3 5 ;
  1 8;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;

mate 1
end

c Define material properties
mat 1
  type 41
  ro 8.93
  nump 4
  locb 3
  locs 4
c      E      ny      K      G
param 1.17 0.35 1.33 0.4333333
endmat

end
tp 0.001
cont
```

Appendix F

Bar impact problem material type umat41v (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 3 5 ;
  1 3 5 ;
  1 8;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;

mate 1
end

c Define material properties
mat 1
type 41 vect
ro 8.93
nump 4
locb 3
locs 4
c      E      ny      K      G
param 1.17 0.35 1.33 0.4333333
endmat

end
tp 0.001
cont
```

Appendix G

Bar impact problem, material routine f3dm3 (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 3 5 ;
  1 3 5 ;
  1 8;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;
mate 1
end

c Define material properties
mat 1 3 ro 8.93 sigy 0.004 e 1.17 etan 0.001 beta 1.0 pr 0.35 endmat

end
tp 0.001
cont
```

Appendix H

Bar impact problem, material type umat42 (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 3 5 ;
  1 3 5 ;
  1 8;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;
mate 1
end

c Define material properties
mat 1
type 42
ro 8.93
numh 7
nump 7
locb 5
locs 6
c      E      ny      sigy      ETAN      K      G      Beta
param 1.17 0.35 0.004 0.001 1.33 0.4333333 1.0
endmat

end
tp 0.001
cont
```

Appendix I

Bar impact problem, material type umat42v (gm cm microsec)

```
dn3d kw93

batch

term 15.0 plti 1.0 prti 1.0
gmprt nodout 0.001 elout 0.001 glstat 0.001;
c To visualize the history variables in LS-Taurus
taurus int8 7;

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
1 3 5 ;
1 3 5 ;
1 8;
c Define coordinate of index space
0 .16 .16
0 .16 .16
0 .6
c Delete corners for cylindrical mapping
di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
c Define node and element for history data
npb 1 1 2;
epb 1 1 1;
mate 1
end

c Define material properties
mat 1
type 42 vect
ro 8.93
numh 7
nump 7
locb 5
locs 6
c      E      ny      sigy      ETAN      K      G      Beta
param 1.17 0.35 0.004 0.001 1.33 0.4333333333 1.0
endmat

end
tp 0.001
cont
```

Appendix J

```
      subroutine umat41 (cm,eps,sig,hisv,dt1,capa,etype,time)
c      implicit real*8 (a-h,o-z)
c      double
c*****
c|   livermore software technology corporation   (lstc)
c|   -----
c|   copyright 1987,1988,1989 john o. hallquist, lstc
c|   all rights reserved
c*****
c
c      isotropic elastic material (sample user subroutine)
c
c variables
c
c      cm(1)=young's modulus
c      cm(2)=poisson's ratio
c
c      eps(1)=local x  strain
c      eps(2)=local y  strain
c      eps(3)=local z  strain
c      eps(4)=local xy strain
c      eps(5)=local yz strain
c      eps(6)=local zx strain
c
c      sig(1)=local x  stress
c      sig(2)=local y  stress
c      sig(3)=local z  stress
c      sig(4)=local xy stress
c      sig(5)=local yz stress
c      sig(6)=local zx stress
c
c      hisv(1)=1st history variable
c      hisv(2)=2nd history variable
c      .
c      .
c      .
c      hisv(n)=nth history variable
c
c      dt1=current time step size
c      capa=reduction factor for transverse shear
c      etype:
c      eq."brick" for solid elements
c      eq."shell" for all shell elements
c      eq."beam"  for all beam elements
c
c      time=current problem time.
c
c
c      all transformations into the element local system are
c      performed prior to entering this subroutine.  transformations
c      back to the global system are performed after exiting this
c      routine.
c
c      all history variables are initialized to zero in the input
c      phase.  initialization of history variables to nonzero values
c      may be done during the first call to this subroutine for each
c      element.
c
c      energy calculations for the dyna3d energy balance are done
c      outside this subroutine.
c
c      character*(*) etype
c      dimension  cm(*),eps(*),sig(*),hisv(*)
c
c      compute shear modulus, g
c
```

```

g2 =cm(1)/(1.+cm(2))
g  =.5*g2
c
if (etype.eq.'brick') then
davg=(-eps(1)-eps(2)-eps(3))/3.
p=-davg*cm(1)/((1.-2.*cm(2)))
sig(1)=sig(1)+p+g2*(eps(1)+davg)
sig(2)=sig(2)+p+g2*(eps(2)+davg)
sig(3)=sig(3)+p+g2*(eps(3)+davg)
sig(4)=sig(4)+g*eps(4)
sig(5)=sig(5)+g*eps(5)
sig(6)=sig(6)+g*eps(6)
c
elseif (etype.eq.'shell') then
c
gc    =capa*g
q1    =cm(1)*cm(2)/((1.0+cm(2))*(1.0-2.0*cm(2)))
q3    =1./(q1+g2)
eps(3)=-q1*(eps(1)+eps(2))*q3
davg  =(-eps(1)-eps(2)-eps(3))/3.
p     =-davg*cm(1)/((1.-2.*cm(2)))
sig(1)=sig(1)+p+g2*(eps(1)+davg)
sig(2)=sig(2)+p+g2*(eps(2)+davg)
sig(3)=0.0
sig(4)=sig(4)+g *eps(4)
sig(5)=sig(5)+gc*eps(5)
sig(6)=sig(6)+gc*eps(6)
c
elseif (etype.eq.'beam' ) then
q1    =cm(1)*cm(2)/((1.0+cm(2))*(1.0-2.0*cm(2)))
q3    =q1+2.0*g
gc    =capa*g
deti  =1./(q3*q3-q1*q1)
c22i  = q3*deti
c23i  =-q1*deti
fac   =(c22i+c23i)*q1
eps(2)=-eps(1)*fac-sig(2)*c22i-sig(3)*c23i
eps(3)=-eps(1)*fac-sig(2)*c23i-sig(3)*c22i
davg  =(-eps(1)-eps(2)-eps(3))/3.
p     =-davg*cm(1)/((1.-2.*cm(2)))
sig(1)=sig(1)+p+g2*(eps(1)+davg)
sig(2)=0.0
sig(3)=0.0
sig(4)=sig(4)+gc*eps(4)
sig(5)=0.0
sig(6)=sig(6)+gc*eps(6)
endif
c
return
end

```

Appendix K

```

      subroutine umat41v (cm,d1,d2,d3,d4,d5,d6,
      .sig1,sig2,sig3,sig4,sig5,sig6,sigma,hist,lft,llt,
      .dt1,capa,etype,tt)
c
c      implicit real*8 (a-h,o-z)
c
c      Elastic material (solid elements only)
c      Mattias Unosson (FOA) and Eric Buzaud (DYNALIS) 2000-01-27
c
c      Variables
c      cm(1)=Young's modulus
c      cm(2)=Poisson's ratio
c      cm(3)=Bulk modulus
c      cm(4)=Shear modulus
c
c This line is needed for succesful compilation on the DEC/Alpha
c$omp thread private (/subtssloc/)
c For CRAY/T3E          use parameter (nlq=24)
c For CRAY/J90          use parameter (nlq=256)
c For CRAY/C90          use parameter (nlq=256)
c For CRAY/T90          use parameter (nlq=256)
c For CRAY/T90IEEEE     use parameter (nlq=256)
c For DEC/Alpha         use parameter (nlq=89)
c For EJ/VFF           use parameter (nlq=1033)
c For Hitachi           use parameter (nlq=130)
c For HP pa7x00         use parameter (nlq=33)
c For HP pa8x00         use parameter (nlq=65)
c For HP Exemplar      use parameter (nlq=128)
c For IBM               use parameter (nlq=128)
c For NEC/SX4           use parameter (nlq=1024)
c For SGI 4400          use parameter (nlq=32)
c For SGI R10000        use parameter (nlq=87)
c For SUN               use parameter (nlq=128)
      parameter (nlq=89)
      character*(*) etype
      dimension cm(*),d1(nlq),d2(nlq),d3(nlq),d4(nlq),d5(nlq),d6(nlq)
      dimension sigma(nlq,*),hist(nlq,*)
      common /subtssloc/ dtlsiz(nlq)
c
c      compute shear modulus, g
      g2 =cm(1)/(1.+cm(2))
      g  =.5*g2
c
      if (etype.eq.'brick') then
      do i=lft,llt
c      d1 is strain rate for solids (but incremental strain for shells)
c      davg is volumetric strain rate
      hist(i,1)=hist(i,1)+(d1(i)+d2(i)+d3(i))*dtlsiz(i)
      hist(i,2)= 9999
      davg=(-d1(i)-d2(i)-d3(i))/3.
c      p is incremental pressure
      p=-davg*cm(1)/((1.-2.*cm(2)))*dtlsiz(i)
      p=-cm(3)*hist(i,1)-1./3.*(sigma(i,1)+sigma(i,2)+sigma(i,3))
c      sigma is total Cauchy stress
      sigma(i,1)=sigma(i,1)+p+g2*(d1(i)+davg)*dtlsiz(i)
      sigma(i,2)=sigma(i,2)+p+g2*(d2(i)+davg)*dtlsiz(i)
      sigma(i,3)=sigma(i,3)+p+g2*(d3(i)+davg)*dtlsiz(i)
      sigma(i,4)=sigma(i,4)+g*d4(i)*dtlsiz(i)
      sigma(i,5)=sigma(i,5)+g*d5(i)*dtlsiz(i)
      sigma(i,6)=sigma(i,6)+g*d6(i)*dtlsiz(i)
      end do
c
      endif
      return
      end

```

Appendix L

```
subroutine umat42 (cm,eps,sig,hisv,dt,capa,etype,time,crv)
c      implicit real*8 (a-h,o-z)
c
c      Elastic-plastic material with isotropic and kinematic hardening.
c      Niclas Strömberg (ERAB) 2000-01-25
c
c      Variables
c      cm(1)=Young's modulus          hisv(1)=Effective plastic strain
c      cm(2)=Poisson's ratio          hisv(2)=sxx
c      cm(3)=Yield strength           hisv(3)=syy
c      cm(4)=Hardening tangent modulus hisv(4)=szz
c      cm(5)=Bulk modulus             hisv(5)=sxy
c      cm(6)=Shear modulus            hisv(6)=syz
c      cm(7)=Beta (0=Kinem./1=isotrop.) hisv(7)=sxz
c
c      dimension cm(*),eps(*),sig(*),hisv(*),crv(101,2,*)
c
c      real sig_dev(1:6)
c      real gamma, gamma_1, gamma_2, fi, sig_y
c      real delta_eff, E_p, lambda_1, lambda_2, p_1
c
c      character*(*) etype
c
c      g2 =cm(1)/(1.+cm(2))
c      g  =.5*g2
c
c      davg=(-eps(1)-eps(2)-eps(3))/3.
c      p=-davg*cm(1)/((1.-2.*cm(2)))
c      sig(1)=sig(1)+p+g2*(eps(1)+davg)
c      sig(2)=sig(2)+p+g2*(eps(2)+davg)
c      sig(3)=sig(3)+p+g2*(eps(3)+davg)
c      sig(4)=sig(4)+g*eps(4)
c      sig(5)=sig(5)+g*eps(5)
c      sig(6)=sig(6)+g*eps(6)
c
c      p_1=(sig(1)+sig(2)+sig(3))/3.0
c      sig_dev(1)=sig(1)-p_1-hisv(2)
c      sig_dev(2)=sig(2)-p_1-hisv(3)
c      sig_dev(3)=sig(3)-p_1-hisv(4)
c      sig_dev(4)=sig(4)-hisv(5)
c      sig_dev(5)=sig(5)-hisv(6)
c      sig_dev(6)=sig(6)-hisv(7)
c
c      gamma_1=1.5*(sig_dev(1)**2+sig_dev(2)**2+sig_dev(3)**2)
c      gamma_2=3.0*(sig_dev(4)**2+sig_dev(5)**2+sig_dev(6)**2)
c      gamma=gamma_1+gamma_2
c
c      E_p=cm(1)*cm(4)/(cm(1)-cm(4))
c      sig_y=cm(3)+cm(7)*E_p*hisv(1)
c
c      fi=gamma-sig_y**2
c
c      if (fi.ge.0.0) then
c      delta_eff=(sqrt(gamma)-sig_y)/(3.0*cm(6)+E_p)
c      hisv(1)=hisv(1)+delta_eff
c      lambda_1=3.0*cm(6)*delta_eff/sqrt(gamma)
c      lambda_2=(1.0-cm(7))*E_p*delta_eff/sqrt(gamma)
c      sig(1)=sig(1)-lambda_1*sig_dev(1)
c      sig(2)=sig(2)-lambda_1*sig_dev(2)
c      sig(3)=sig(3)-lambda_1*sig_dev(3)
c      sig(4)=sig(4)-lambda_1*sig_dev(4)
c      sig(5)=sig(5)-lambda_1*sig_dev(5)
c      sig(6)=sig(6)-lambda_1*sig_dev(6)
c      hisv(2)=hisv(2)+lambda_2*sig_dev(1)
c      hisv(3)=hisv(3)+lambda_2*sig_dev(2)
c      hisv(4)=hisv(4)+lambda_2*sig_dev(3)

```

```
hisv(5)=hisv(5)+lambda_2*sig_dev(4)
hisv(6)=hisv(6)+lambda_2*sig_dev(5)
hisv(7)=hisv(7)+lambda_2*sig_dev(6)
endif
c
return
end
```

Appendix M

```

subroutine umat42v(cm,d1,d2,d3,d4,d5,d6,sig1dum,sig2dum,
. sig3dum,sig4dum,sig5dum,sig6dum,sigma,histdum,lft,llt,
. dt1dum,capa,etype,tt)
c      implicit real*8 (a-h,o-z)                                double
c
c      Isotropic Elastic-plastic material with isotropic and kinematic hardening.
c      (Brick elements only)
c      Niclas Stromberg (ERAB) 2000-01-25
c      Modified by Mattias Unosson (FOA) and Eric Buzaud (DYNALIS) 2000-01-26
c
c      Variables
c      cm(1)=Young's modulus          hist(1)=Effective plastic strain
c      cm(2)=Poisson's ratio          hist(2)=sxx
c      cm(3)=Yield strength           hist(3)=syy
c      cm(4)=Hardening tangent modulus hist(4)=szz
c      cm(5)=Bulk modulus             hist(5)=sxy
c      cm(6)=Shear modulus             hist(6)=syz
c      cm(7)=Beta (0=Kinem./1=isotrop.) hist(7)=sxz
c
c This line is needed for succesful compilation on the DEC/Alpha
c$omp thread private (/subtssloc/)
c For CRAY/T3E          use parameter (nlq=24)
c For CRAY/J90          use parameter (nlq=256)
c For CRAY/C90          use parameter (nlq=256)
c For CRAY/T90          use parameter (nlq=256)
c For CRAY/T90IEEEE     use parameter (nlq=256)
c For DEC/Alpha         use parameter (nlq=89)
c For EJ/VFF            use parameter (nlq=1033)
c For Hitachi           use parameter (nlq=130)
c For HP pa7x00         use parameter (nlq=33)
c For HP pa8x00         use parameter (nlq=65)
c For HP Exemplar       use parameter (nlq=128)
c For IBM               use parameter (nlq=128)
c For NEC/SX4           use parameter (nlq=1024)
c For SGI 4400          use parameter (nlq=32)
c For SGI R10000        use parameter (nlq=87)
c For SUN               use parameter (nlq=128)
      parameter (nlq=89)
c
c      common /subtssloc/ dt1siz(nlq)
c
c      dimension cm(*),d1(*),d2(*),d3(*),d4(*),d5(*),d6(*)
c      dimension sigma(nlq,*)
c
c      real sig_dev(1:6)
c      real gamma, gamma_1, gamma_2, fi, sig_y
c      real delta_eff, E_p, lambda_1, lambda_2, p_1
c
c      integer i,lft,llt
c
c      character*(*) etype
c
c      g2 =cm(1)/(1.+cm(2))
c      g  =.5*g2
c
c      do i=lft,llt
c         davg=(-d1(i)-d2(i)-d3(i))/3.
c         p=-davg*dt1siz(i)*cm(1)/((1.-2.*cm(2)))
c         sigma(i,1)=sigma(i,1)+p+g2*(d1(i)+davg)*dt1siz(i)
c         sigma(i,2)=sigma(i,2)+p+g2*(d2(i)+davg)*dt1siz(i)
c         sigma(i,3)=sigma(i,3)+p+g2*(d3(i)+davg)*dt1siz(i)
c         sigma(i,4)=sigma(i,4)+g*d4(i)*dt1siz(i)
c         sigma(i,5)=sigma(i,5)+g*d5(i)*dt1siz(i)
c         sigma(i,6)=sigma(i,6)+g*d6(i)*dt1siz(i)
c
c         p_1=(sigma(i,1)+sigma(i,2)+sigma(i,3))/3.

```

```

sig_dev(1)=sigma(i,1)-p_1-sigma(i,8)
sig_dev(2)=sigma(i,2)-p_1-sigma(i,9)
sig_dev(3)=sigma(i,3)-p_1-sigma(i,10)
sig_dev(4)=sigma(i,4)-sigma(i,11)
sig_dev(5)=sigma(i,5)-sigma(i,12)
sig_dev(6)=sigma(i,6)-sigma(i,13)
c
gamma_1=1.5*(sig_dev(1)**2+sig_dev(2)**2+sig_dev(3)**2)
gamma_2=3.0*(sig_dev(4)**2+sig_dev(5)**2+sig_dev(6)**2)
gamma=gamma_1+gamma_2
c
E_p=cm(1)*cm(4)/(cm(1)-cm(4))
sig_y=cm(3)+cm(7)*E_p*sigma(i,7)
c
fi=gamma-sig_y**2
c
if (fi.ge.0.0) then
  delta_eff=(sqrt(gamma)-sig_y)/(3.0*cm(6)+E_p)
  sigma(i,7)=sigma(i,7)+delta_eff
  lambda_1=3.0*cm(6)*delta_eff/sqrt(gamma)
  lambda_2=(1.0-cm(7))*E_p*delta_eff/sqrt(gamma)
  sigma(i,1)=sigma(i,1)-lambda_1*sig_dev(1)
  sigma(i,2)=sigma(i,2)-lambda_1*sig_dev(2)
  sigma(i,3)=sigma(i,3)-lambda_1*sig_dev(3)
  sigma(i,4)=sigma(i,4)-lambda_1*sig_dev(4)
  sigma(i,5)=sigma(i,5)-lambda_1*sig_dev(5)
  sigma(i,6)=sigma(i,6)-lambda_1*sig_dev(6)
c
  sigma(i,8)=sigma(i,8)+lambda_2*sig_dev(1)
  sigma(i,9)=sigma(i,9)+lambda_2*sig_dev(2)
  sigma(i,10)=sigma(i,10)+lambda_2*sig_dev(3)
  sigma(i,11)=sigma(i,11)+lambda_2*sig_dev(4)
  sigma(i,12)=sigma(i,12)+lambda_2*sig_dev(5)
  sigma(i,13)=sigma(i,13)+lambda_2*sig_dev(6)

  endif
end do
c
return
end

```

Appendix N

CPU_test (gm cm microsec)

```
dn3d kw93

batch

term 50.0 plti 1.0 prti 81.

velocity 0 0 -0.0227

c Define symmetry planes
plane 3 0 0 0 0 -1 0 .001 symm
      0 0 0 -1 0 0 .001 symm
      0 0 0 0 0 1 .001 symm

start
c Define index space
  1 20 39 ;
  1 20 39 ;
  1 60;
c Define coordinate of index space
  0 .16 .16
  0 .16 .16
  0 .6
c Delete corners for cylindrical mapping
  di 2 3 ; 2 3 ; ;
c Capture surfaces to be mapped and map in cyld. space
  sfi 1 -3;1 -3;; cy 0 0 0 0 0 1 .32
c mv rectangle towards center
  pb 2 2 0 2 2 0 xy [0.16/sqrt(2)] [0.16/sqrt(2)]
  mate 1
end

c Define material properties

c - f3dm1
mat 1 type 1 ro 8.93 e 1.17 pr 0.35 hgqt 5 endmat

c - umat41
c mat 1 type 41 ro 8.93 nump 4 locb 3 locs 4
c   param 1.17 0.35 1.33 0.4333333 hgqt 5 endmat

c - umat41v
c mat 1 type 41 vect ro 8.93 nump 4 locb 3 locs 4
c   param 1.17 0.35 1.33 0.4333333 hgqt 5 endmat

c - f3dm3
c mat 1 3 ro 8.93 sigy 0.004 e 1.17 etan 0.001 beta 1.0 pr 0.35 hgqt 5 endmat

c - umat42
c mat 1 type 42 ro 8.93 numh 7 nump 7 locb 5 locs 6
c   param 1.17 0.35 0.004 0.001 1.33 0.4333333 1.0 hgqt 5 endmat

c - umat42v
c mat 1 type 42 vect ro 8.93 numh 7 nump 7 locb 5 locs 6
c   param 1.17 0.35 0.004 0.001 1.33 0.4333333 1.0 hgqt 5 endmat

end
tp 0.001
cont
```