ABAQUS UMAT AND VUMAT HONEYCOMB MATERIAL MODEL

James G. Ratcliffe and Frank Leone

INTRODUCTION

A 1-dimensional material model was developed for simulating the transverse (thickness-direction) response of aluminum honeycomb structure. The model was implemented as a user-defined material subroutine (UMAT) in the commercial finite element analysis code ABAQUS®/Standard and was later rewritten as a VUMAT for use with ABAQUS®/Explicit. The UMAT/VUMAT has been applied to analyses for simulating quasi-static indentation tests on aluminum honeycomb-based sandwich plates. Comparison of analysis results with data from these experiments shows overall good agreement. Specifically, analyses of quasi-static indentation tests yielded accurate global specimen responses. Predicted residual indentation was also in reasonable agreement with measured values. The purpose of this white paper is to briefly outline the user subroutines, including a reprint of the UMAT and VUMAT for application by other users.

CORE DAMAGE MATERIAL MODEL

The purpose of the model described in this section is to represent the response of aluminum honeycomb structure to transverse loading. The model is based on a transverse stress-strain relationship measured using a flatwise compression/tension test. This stress-strain relationship is implemented into ABAQUS®/Standard as a user-defined, 1-dimensional material model. Details of the model, its implementation into ABAQUS®/Standard and evaluation of the implemented model follow in the remainder of this section.

Constitutive Law

The stress-strain relationship on which the current honeycomb material model is based is typically measured using the flatwise-compression specimen depicted in Figure 1. The specimen is based on that detailed in the standard test methods for conducting flatwise compression and tension tests (ASTM C365 [1] and ASTM C297 [2], respectively), and consists of a 50mm-square block of honeycomb bonded between two loading blocks. A typical stress-strain response from a flatwise compression test is illustrated in Figure 2. Compressive loading is depicted by the solid grey lines in Figure 2, while the solid orange lines indicate unloading. Specimens are loaded in compression until the honeycomb cell walls begin to collapse, which generally corresponds to the sudden decrease in stress as shown in Figure 2. Upon further compressive loading, the honeycomb specimen continues crushing, which is characterized either by a stress plateau or a slight increase in stress as indicated in Figure 2. The specimen is unloaded after the required amount of core crushing has been obtained. This initially results in an elastic response from the

specimen as the collapsed cell walls begin to deform. At some point during unloading, the specimen undergoes an overall reduction in stiffness, which is attributed to plastic deformation in aluminum honeycombs [3]. Complete unloading of the specimen either involves fracturing of the cell walls [3], or further plastic deformation, resulting in a residual tensile stress [4] as depicted in Figure 2. This constitutive relationship is specific to aluminum honeycombs and so may well adopt a different form in honeycombs made from other materials.

The current model idealizes the stress-strain relationship in Figure 2 as a series of linear functions depicted by the dashed lines in the figure. The parameters associated with this idealization are also depicted in the figure and act as the user input for the implementation of this model into ABAQUS®.

Implementation of Model into ABAQUS®

The constitutive model described in the previous section and illustrated in Figure 2 is implemented into ABAQUS®/Standard as a user material model (UMAT) [5]. The routine was later rewritten for use as a VUMAT in ABAQUS®/Explicit. The implemented model can be used with 2-node truss elements (ABAQUS®/Standard element type T3D2) in a finite element analysis in which the truss elements are used to represent a honeycomb structure. The UMAT is in the form of a user-defined subroutine, written in the computer language FORTRAN 77. The VUMAT is formatted in "free-form" to enable easier compilation of the subroutine alongside the opensource progressive damage analysis code CompDam [6]. ABAQUS® provides the user with access to a range of its own subroutines that may be called within the UMAT and VUMAT, enabling access to information such as analysis variables (state, field, etc.) and analysis time (step and increment number, time step, etc.). The parameters that define the constitutive model (Figure 2) are provided by the user in the analysis job input file (.inp) that contains all other user-defined information (node numbering, mesh connectivity, material, properties, load and boundary conditions, etc.) necessary for running an analysis. The UMAT or VUMAT is called at the time of executing an analysis at which point the UMAT or VUMAT is compiled and configured for operation within the analysis job.

Operation of the UMAT within a finite element analysis proceeds as follows (analyses using the VMAT follow a similar operation):

1. Analysis begins and user-input (Figure 2) is read by the UMAT.
2. The UMAT computes the tangent moduli, E_1, E_2, and E_3, (Figure 2).
3. At the beginning of each analysis increment, the current strain and the strain increment are provided to the UMAT by ABAQUS®. The UMAT checks these values against the strain values that define key stages of deformation in the constitutive model (e_1, e_2, and e_3 in Figure 2). The strain increment is used to determine the direction of loading in an element.
4. The UMAT checks the current compressive strain value against the maximum value determined in the previous iteration. If the current strain value exceeds the presently stored maximal value then the current strain is saved as the maximal value. This information is used to store the current damage state in an element.

5. A tangent modulus is selected based on the current strain and strain increment. This modulus is assigned to the stiffness matrix of the element. The stress in the element is then computed using the selected tangent modulus and the current strain increment.
6. The tangent stiffness and stress values are reported to ABAQUS®.
7. Steps 5 and 6 are repeated until the analysis converges corresponding to the end of the increment.
8. Steps 3-7 are repeated until completion of an analysis step(s).

The above operation is performed for each truss finite element. Additionally, a state of damage is defined in each element that corresponds to one of the three damage states depicted in Figure 2 (numbers in parenthesis in Figure 2). Damage state 1 corresponds to the region of the constitutive model before cell wall collapse has occurred (i.e. the initial elastic response). Damage state 2 corresponds to the region of the constitutive model during initial cell wall collapse. Damage state 3 corresponds to the region of the constitutive model during continued crushing of the honeycomb. This information is used by the UMAT to determine the correct tangent stiffness during unloading (as illustrated by the dashed arrows in Figure 3) and also to provide a record of the damage state of each element at any moment during an analysis. The UMAT and VUMAT subroutines are printed in Appendices A and B, respectively.

Evaluation of Material Model

SIMULATION OF QSI TESTS

The UMAT was evaluated by applying it to finite element analyses of a sandwich plate subjected to quasi-static indentation (QSI) loading. This test case was chosen because the specimen response does not involve flexure, which would not be captured by the current one-dimensional model. The sandwich plate consisted of an aluminum honeycomb core reinforced with quasi-isotropic graphite/epoxy composite facesheets. Complete details of the sandwich plate are given in Figure 3. QSI tests were previously conducted on this sandwich plate using a 25mm-diameter indenter and also a 76mm-diameter indenter [7]. Specimens were supported by a rigid back plate during each QSI test. A schematic of the test configuration is shown in Figure 3. The indentation/force response of each specimen was recorded in addition to the residual indentation in the sandwich plate at the end of each test [7]. Finite element analyses of these two test cases were conducted as part of an exercise to evaluate the developed UMAT. The 76mm indenter test case was deemed appropriate for evaluating the UMAT because the QSI response of the specimens was found to be largely attributed to core damage rather than facesheet damage [7], which is not accounted for in the following analyses. Even though significant facesheet damage (in the form of delamination) was observed in the 25mm indenter case, an analysis of this case was conducted nonetheless. In both cases, static, geometrically nonlinear analyses were conducted in order to account for indenter contact (as described below).

Finite element models used in the current evaluation of the UMAT consisted of 4-node shell elements (ABAQUS® type S4) to represent the upper facesheet. Each node of the shell elements was connected to 2-node truss elements, (ABAQUS® type

T3D2) used to represent the honeycomb core. The length of the truss elements was equal to the honeycomb core height (in this case 25mm). A typical finite element mesh is shown in Figure 4 (only small number of truss elements are illustrated in the figure for clarity). Two-axis symmetry was assumed in the analysis in order to reduce the overall model size. Therefore, the center of the sandwich specimen corresponds to the corner of the mesh labeled 'A' in Figure 4. A relatively fine mesh was used to represent the facesheet, with an element length of 0.5mm. All shell elements representing the facesheet had 1:1 aspect ratios. A similar meshing scheme was shown to yield a converged solution in a similar analysis conducted previously [8], and therefore a mesh convergence study was deemed to be unnecessary in the current case. The lower facesheet (Figure 3) was initially modeled with shell elements. However, this was found to be unnecessary and instead the boundary condition detailed in Figure 4 (w=0) was used to represent the supported specimen side. The stacking sequence of the upper facesheet was represented using a composite stack feature provided by ABAQUS®/Standard, where the ply orientation corresponded to the coordinate system shown in Figure 4. The nine engineering constants used to define the elastic response of the facesheet plies (facesheet damage was not considered in the current analyses) are presented in TABLE I, which also includes user input for the UMAT. The UMAT input data were obtained from previous flatwise compression/tension tests conducted on the sandwich structure [4]. The spherical indenters were represented using rigid elements (ABAQUS®/Standard type R3D3 and R3D4) and a frictionless contact algorithm provided by ABAQUS® was employed to facilitate contact between the rigid indenter elements and the shell elements representing the upper facesheet. The indenter mesh was positioned such that the node corresponding to the indenter tip was coincident with the plane of the facesheet, and the in-plane position of the indenter tip corresponded to the center of the sandwich specimen. Loading was applied to the rigid indenter mesh by prescribing a displacement along the global z-axis to a reference node positioned in the center of the indenter as shown in Figure 4. The nodes of the rigid indenter mesh were kinematically coupled with this dummy node such that they mimic any translation and rotation prescribed at this node. All prescribed boundary conditions are illustrated in Figure 4. Two analyses were performed, simulating the QSI tests using either a 25mm-diameter or a 76mm-diameter indenter. Each analysis was conducted in two steps. The first step involved translating the indenter in the z-axis towards the specimen up to a maximum prescribed indentation (denoted as z_{max} in Figure 4). In the second step, the prescribed indenter translation was reversed until the indenter returned to its original position. The maximum allowed time over which a solution is sought during an increment (referred to as 'time increment' in ABAQUS® terminology), $\Delta t_{inc\text{-}max}$, was limited to 0.1% of the total time step in each analysis (this limit on $\Delta t_{inc\text{-}max}$ was imposed as a result of the maximum analysis time increment study described later). The computed indentation/force response and residual indentation profiles from both analyses were compared with corresponding data measured previously [7].

EFFECT OF MAXIMUM ANALYSIS STEP TIME INCREMENT ON ANALYSIS SOLUTION

A static, geometrically nonlinear analysis in ABAQUS® comprises a series of steps such as those detailed above for translating the rigid indenter. In turn, each step

is partitioned into a number of increments for which ABAQUS® attempts to converge to a solution. Typically, the time step over which ABAQUS® attempts to converge to a solution, otherwise known as the time increment, is set to a relatively small value at the beginning of an analysis step. As the analysis proceeds, and if convergence is easily found, ABAQUS® automatically increases the time increment up to a maximum time increment that is either set by the user or ABAQUS® (see [9] for further details on analysis solution strategies utilized by ABAQUS®). It was found that the solution of an analysis using the currently developed UMAT was very sensitive to this maximum time increment, $\Delta t_{inc\text{-}max}$. This sensitivity was investigated using the analysis of the QSI test with the 76mm-diameter indenter. During this study, an analysis of this test configuration was repeated five times, each using a different value of $\Delta t_{inc\text{-}max}$, ranging from 2% of the total time step to 0.1% of the total time step. (In each case, the initial time increment was kept constant at 0.1% of the total time step.) The indentation/force responses and residual indentation profiles were computed and checked for convergence relative to $\Delta t_{inc\text{-}max}$.

RESULTS / DISCUSSION

Simulation of QSI Tests

The compressive force/indentation responses computed from simulations of the QSI tests with a 25mm-diameter indenter and 76mm-diameter indenter are presented in Figure 5. The force/indentation response from the actual tests is superimposed onto each plot. The overall force/indentation response compares very well to the measured responses. The beginning of each response is linear, corresponding to the initial linear response of the honeycomb. After some point, the honeycomb cell walls begin collapsing, which is characterized by an initial reduction in overall stiffness of the specimen (stiffness here is defined as the gradient to the tangent of the force/indentation curve). The amount of crushed honeycomb increases as indentation is increased. However, this is accompanied by an increase in contact area of the indenter with the neighboring facesheet. Consequently, continued indentation results in a gradual increase in specimen stiffness. After the maximum indentation is reached, the indenter is returned to its original position. The resulting unloading is nonlinear largely because the amount of contact area between the indenter and specimen reduces upon further unloading.

The computed residual indentation profiles (along Segment AB in Figure 4) from analyses of the 25mm-diameter and 76mm-diameter indenter cases are presented in Figure 6. The computed dent profiles are mirrored about the yz plane (Figure 4) in order to provide a meaningful comparison with the measured dent profiles. The measured residual indentation profiles [7] are superimposed onto both plots for comparison. The results show that the analysis of the 25mm indenter case captured the correct overall shape of the dent, although the maximum indentation was underestimated by 24%. The computed residual indentation from analysis of the 76mm indenter case was very close to the measured dent profile.

The peak applied force, residual indentation and total energy dissipation (area encompassed by the force/indentation responses) computed from both analyses are presented in TABLE II. The corresponding measured values [7] are also included in the table for comparison. In the 25mm-diameter indenter case, the computed peak

force is almost identical to the measured value (less than 2% difference). The computed residual indentation for this case, however, underestimates the measured value by almost 24%. The computed total energy dissipation compares more favorably and underestimated the measured value by 15%. This significant difference in dent profiles and energy dissipation is likely attributed to the fact that significant delamination was observed in the 25mm indenter case, which was not accounted for in the current analysis.

The peak force computed from the analysis of the 76mm-diameter indenter case was again in excellent agreement with the measured value (less than 1% difference). The computed residual indentation was in better agreement with the measured value (compared to the 25mm-diameter indenter case) and in this case underestimated the measured residual indentation by only 2%. The computed total energy dissipation underestimated the measured value by just under 10%. This relatively improved agreement in the 76mm indenter case is consistent with the fact that the specimen response was largely attributed to core damage [7].

In general, the overall computed specimen response compares very favorably with the experimental data in the 76mm indenter case, and suggests that the implemented one-dimensional material model is successful in capturing the response of honeycomb structure to transverse loading.

Maximum Analysis Time Increment

The maximum allowed time step during each increment, $\Delta t_{inc\text{-}max}$, in the above two analyses was limited to 0.1% of total time step. As mentioned previously, this limit was imposed as a result of a study conducted to determine the effect of maximum allowed time step on analysis results. The results of this study (based on analyses of the 76mm-diameter indenter case) are presented in Figure 7, where the computed peak force (Figure 7a) and maximum residual indentation (Figure 7b) are plotted as functions of the number of increments per analysis step. The number of increments per step is the reciprocal of $\Delta t_{inc\text{-}max}$, and the results are plotted in this manner to highlight the convergence of the solutions. Included in the plots are the measured peak force and residual indentation for this test case (denoted by dashed horizontal lines in each plot). Both plots show that the computed values tend to converge towards their measured counterparts as the number of increments per analysis step increases. Hence, from these results it was assumed that a converged solution was obtained when the number of increments per step was greater than or equal to 1000, i.e. when $\Delta t_{inc\text{-}max}$ 0.001.

SUMMARY AND CONCLUDING REMARKS

A material model for representing transverse loading of honeycomb structure was implemented as a material model into the commercial finite element analysis code ABAQUS®/Standard. The model is based on an empirically determined stress-strain relationship measured using a flatwise/compression specimen. The exercises conducted for evaluating the UMAT resulted in the following general observations:

- The results from analyses of the quasi-static indentation (QSI) tests showed overall good agreement with the corresponding test data. This favorable

comparison indicates that the one-dimensional material model can adequately capture the honeycomb core response, at least for QSI simulations.

- Results from analyses that employ the implemented material model would be sensitive to the time step taken during each analysis increment. However, this study indicated that a converged solution could be found when the maximum time increment in an analysis step was limited to 0.1% of the total time step.
- In general, this simple approach does not involve a significant computational burden, which should make it more tractable to simulate other damage mechanisms in the same analysis.

REFERENCES

1. ASTM C365/C365M-05 "Standard Test Method for Flatwise Compressive Properties of Sandwich Cores," 2008 Annual Book of ASTM Standards, Vol.15.03.
2. ASTM C3297/C297M-04 "Standard Test Method for Flatwise Tensile Strength of Sandwich Constructions," 2008 Annual Book of ASTM Standards, Vol.15.03.
3. Minakuchi, S., Y. Okabe, and N Takeda, "Segment-Wise Model for Theoretical Simulation of Barely Visible Indentation Damage in Composite Sandwich Beams: Part I - Formulation," *Composites Part A*, 39(1):133-144, 2008.
4. Czabaj, M. W., "Damage and Damage Tolerance of High Temperature Composites and Sandwich Composite Structures," Ph.D. dissertation, Cornell University, August 2010.
5. ABAQUS® 6.11 User Subroutines Reference Manual, Simulia, 2011.
6. CompDam Progressive Damage Analysis GitHub: https://github.com/nasa/CompDam_DGD
7. Singh, A. K., B. D. Davidson, D. P. Eisenberg, M. W. Czabaj, and A. T. Zehnder, "Damage Characterization of Quasi-Statically Indented Composite Sandwich Structures," *Journal of Composite Materials*, doi: 10.1177/0021998312446500, 2012.
8. Ratcliffe, J. and W. C Jackson, "A Finite Element Analysis for Predicting the Residual Compressive Strength of Impact-Damaged Sandwich Panels," NASA Technical Memorandum, NASA/TM-2008-215341, August 2008.
9. ABAQUS® 6.11 Analysis User's Manual, Simulia, 2011.

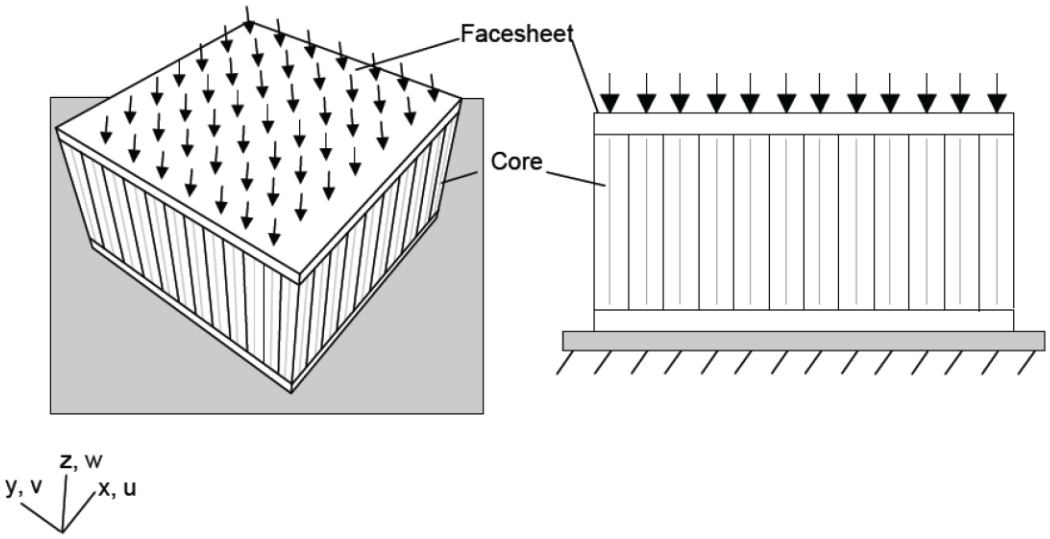


Figure 1. Schematic of a 50mm-square flatwise compression specimen.

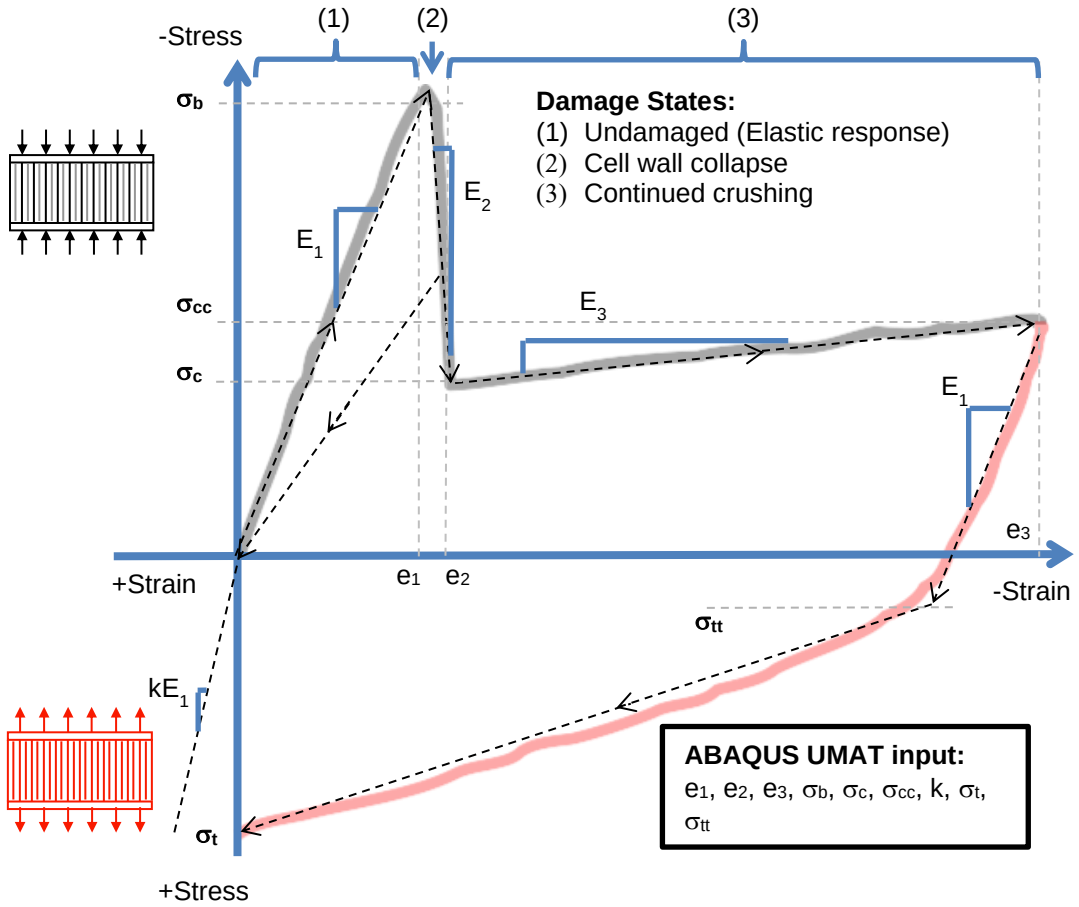


Figure 2. Idealized stress-strain relationship of honeycomb structure.

Sandwich plate details:

Facesheets:
8-ply graphite/epoxy tape
Layup $[45/0/-45/90]_2$
Thickness: 1.02mm
Core:
Aluminum (5052) honeycomb
Density: $50kg/m^3$
Cell size: 3.2mm
Cell wall thickness: 0.02mm
Height: 25.4mm
Sandwich plate dimensions:
Width: 152mm
Length: 178mm

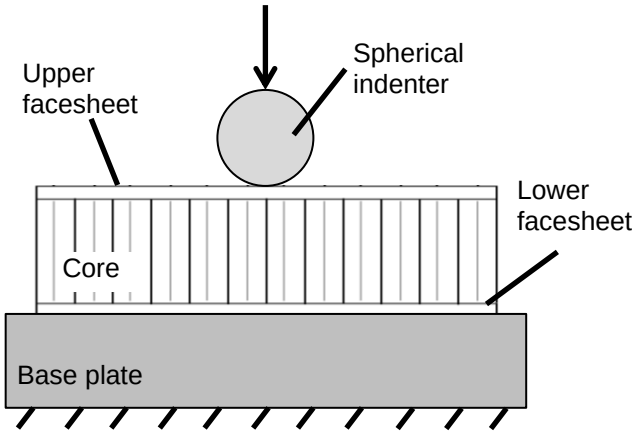
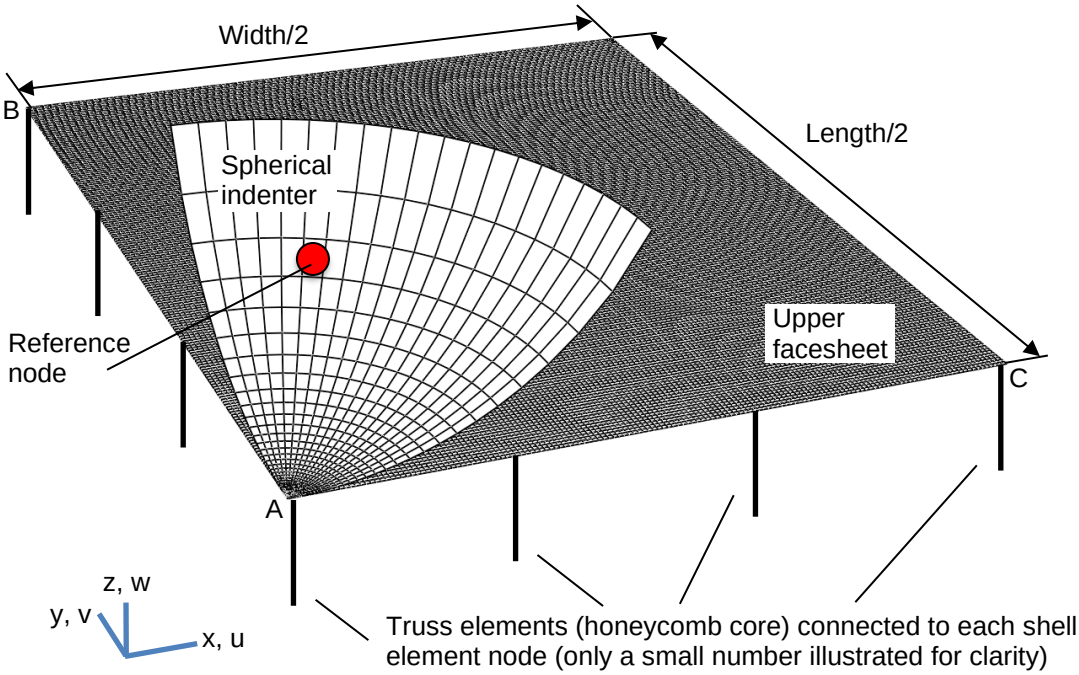


Figure 3. Details of quasi-static indentation test setup [20].



BOUNDARY CONDITIONS:

Nodes along Segment AB:
$u=0$; $\phi_y=0$*
Nodes along Segment AC:
$v=0$; $\phi_x=0$*
Truss nodes (side opposite facesheet):
$w=0$

Reference node:
Fixed except for $w=z_{max}$
z_{max}=1.44mm (25mm indenter case)
z_{max}=2.57mm (76mm indenter case)

* ϕ_i denotes rotation about the i^{th} axis

Figure 4. Finite element mesh and boundary conditions of QSI analyses (76mm indentor shown).

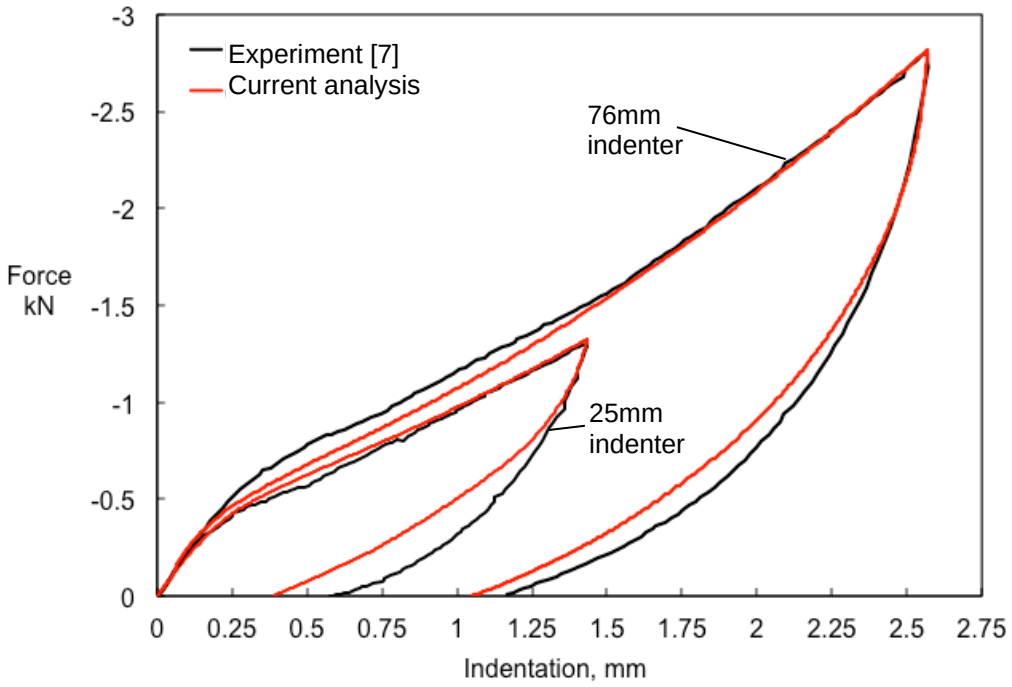


Figure 5. Computed versus measured force/indentation response of QSI specimens.

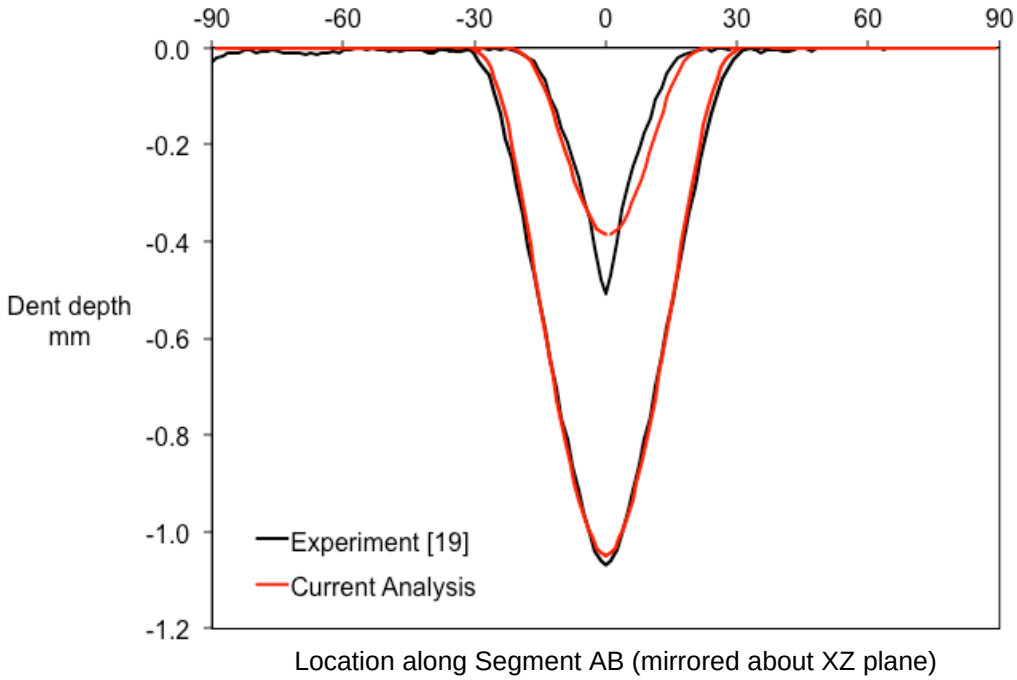


Figure 6. Computed versus measured residual indentation depths in QSI specimens.

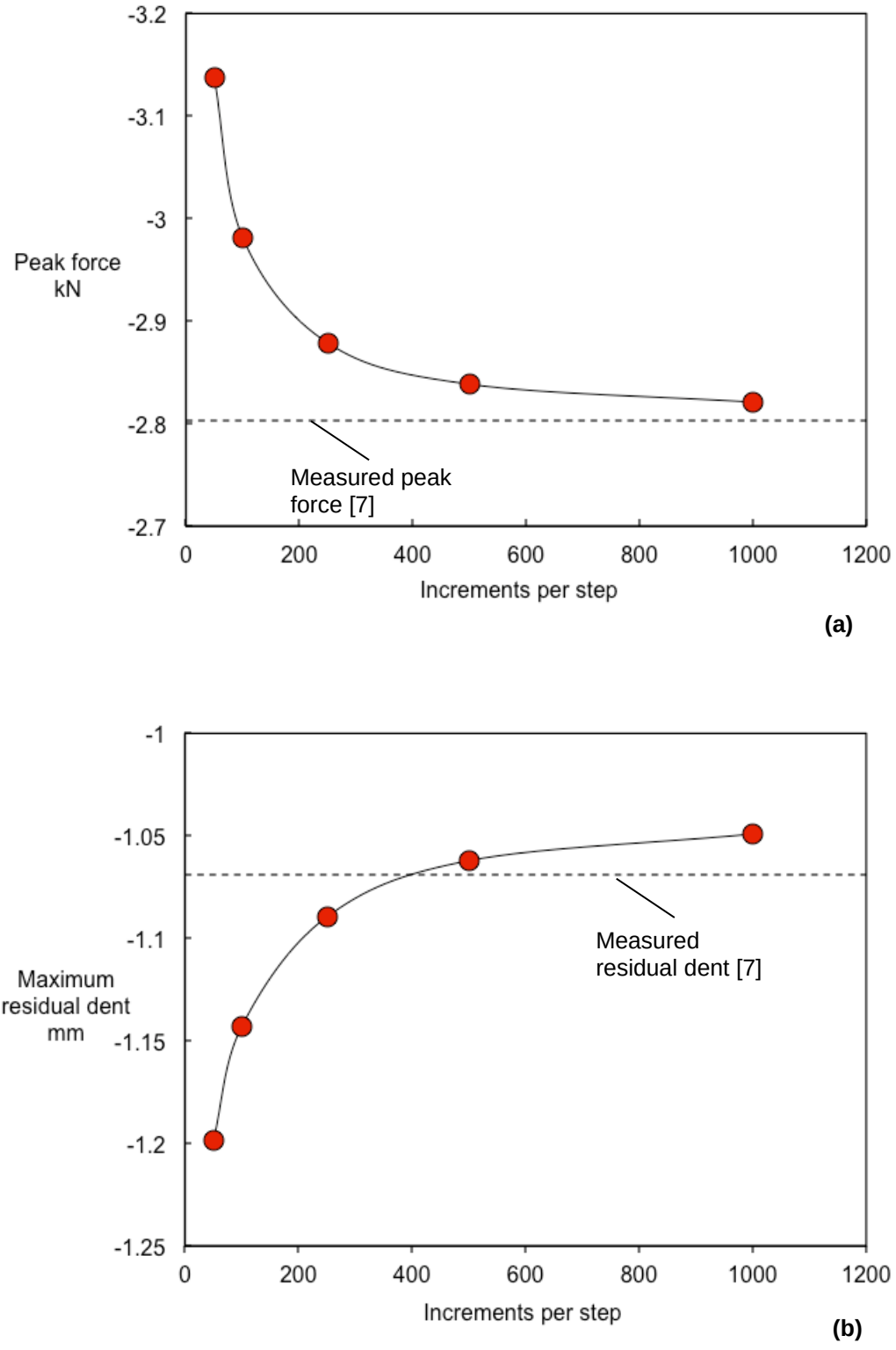


Figure 7. (a) Computed peak force versus increments per step, (b) Computed maximum resicual dent versus increments per step.

TABLE I. MATERIAL PROPERTIES USED IN ANALYSES.

Facesheet ply properties [4]		
E_{11}= 143.0 GPa	E_{22} = 12.9 GPa	E_{33} = 11.7 GPa
ν_{12} = 0.32	ν_{13} = 0.32	ν_{23} = 0.44
G_{12} = 4.1 GPa	G_{13} = 4.1 GPa	G_{23} = 3.98 GPa
UMAT input (Figure 2) [4]		
σ_b=2MPa	σ_c=0.84MPa	σ_{cc}=1.12MPa
σ_{tt}=0.01MPa	σ_t=1MPa	k=1.26
e_1=1880με	e_2=10660με	e_3=98800με

TABLE II. COMPUTED AND MEASURED QSI SPECIMEN RESPONSE.

25mm indentor QSI test			
	Computed	**Measured [7]**	**% difference**
Peak force (kN)	1327	1300	<2%
Max residual dent (mm)	0.39	0.51	-24%
Total energy dissipation (Nmm)	600	701	-15%
76mm indentor QSI test			
	Computed	**Measured [7]**	**% difference**
Peak force (kN)	2821	2800	<1%
Max residual dent (mm)	1.05	1.07	-2%
Total energy dissipation (Nmm)	2348	2599	-10%

APPENDIX 1: Core Damage Material Model UMAT

```
      SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*80 CMNAME
      CHARACTER*80 OUTPUT1
      CHARACTER*80 OUTPUT2
      CHARACTER*256 JOBNAME
      CHARACTER*256 OUTDIR
      CHARACTER*4 ext1
      CHARACTER*4 ext2
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
     3 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
      REAL:: ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3
C
      IF(KSTEP .EQ. 1 .AND. KINC .EQ. 1)THEN
        ext1='.sts';ext2='.sdv'
        CALL GETJOBNAME(JOBNAME,LENJOBNAME)
        CALL GETOUTDIR(OUTDIR,LENOUTDIR)
      OUTPUT1=OUTDIR(1:LENOUTDIR)//"/"
      OUTPUT1=OUTPUT1(1:(LENOUTDIR+1))//JOBNAME(1:LENJOBNAME)//ext1(1:4)
      OUTPUT2=OUTDIR(1:LENOUTDIR)//"/"
      OUTPUT2=OUTPUT2(1:(LENOUTDIR+1))//JOBNAME(1:LENJOBNAME)//ext2(1:4)
        OPEN(UNIT=16,FILE=OUTPUT1)
        OPEN(UNIT=17,FILE=OUTPUT2)
      ENDIF

      IF(STATEV(2) .LT. PROPS(1) .AND. STATEV(2) .GT. PROPS(2))THEN
      ep1=STATEV(2)
C      ENDIF
C      IF(STATEV(2) .EQ. 0 .AND. STATEV(2) .GE. PROPS(1))THEN
      ELSE
      ep1=PROPS(1)
      ENDIF
      ep2=PROPS(2)
      ep3=PROPS(3)
      sb=PROPS(4)
      sc=PROPS(5)
      scc=PROPS(6)
      k=PROPS(7)
      st=PROPS(8)
      stt=PROPS(9)
      e1=PROPS(4)/PROPS(1)
      e2=(PROPS(4)-PROPS(5))/(PROPS(1)-PROPS(2))
      e3=(PROPS(6)-PROPS(5))/(PROPS(3)-PROPS(2))

C    Solution-dependent state variables:
C    STATEV(1): Damage state, where:
C    (inactive)(-1)=Tensile loaded only (loading or unloading)
C               (1)=Elastically compressed (0>strain>=ep1)
C               (2)=Partially crushed (ep1>strain>=ep2)
C               (3)=Crushed (ep2>strain)
C    (inactive) (4)=Unloaded from crushed state with stress<stt
C    (inactive) (5)=Unloaded from crushed state with stress>stt
C    STATEV(2): Max applied -ve strain
C    STATEV(3): Stress at max applied -ve strain
```

```
C     STATEV(4): Strain when STRESS(1)=stt
C     STATEV(5): Strain when stress reaches stt during reloading
C     STATEV(6): Indicator for stress exceeding stt, where:
C                (0)=Stress didn't exceed stt
C                (1)=Stres exceeded stt


        CALL QSI_INTACT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
     5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3,I)


      RETURN
      END

      SUBROUTINE QSI_INTACT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
     5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3)

      INCLUDE 'ABA_PARAM.INC'

      CHARACTER*80 CMNAME
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
     3 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
      REAL:: ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3

C     Step 1: Elastic tensile loading and unloading:
      IF(STRAN(1) .GT. 0. .AND. STRESS(1) .GT. 0)THEN
        IF(DSTRAN(1) .GT. 0. .OR. DSTRAN(1) .LE. 0.)THEN
          DDSDDE(1,1)=k*e1
          STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
        ENDIF
        IF(DSTRAN(1) .GT. 0)THEN
        ENDIF
        IF(DSTRAN(1) .LE. 0)THEN
        ENDIF
      ENDIF

C     Step 2:Elastically compressed (0>strain>=ep1) loading and unloading
      IF(STRAN(1) .LE. 0. .AND. STRAN(1) .GE. ep1)THEN
C       Compressive loading
        IF(DSTRAN(1) .LE. 0.)THEN
C         Reloading from having been previously crushed
          IF(STRAN(1) .GT. STATEV(2))THEN
            IF(STATEV(2) .GE. ep2)THEN
            DDSDDE(1,1)=STATEV(3)/STATEV(2)
            ENDIF
            IF(STATEV(2) .LT. ep2)THEN
              IF(STRESS(1) .GE. stt)THEN
                DDSDDE(1,1)=e1
                STATEV(5)=STRAN(1)
              ELSE
```

```
123                 DDSDDE(1,1)=ABS((STATEV(3)-stt)/(STATEV(5)-STATEV(2)))
124               ENDIF
125             ENDIF
126           ELSE
127             DDSDDE(1,1)=e1
128             STATEV(1)=1
129           ENDIF
130           STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
131 C         This ensures that STATEV(2) contains the lowest strain reached
132 C         regardless of how many times the material has been reloaded.
133            IF(STRAN(1) .LT. STATEV(2))THEN
134              STATEV(2)=STRAN(1)
135              STATEV(3)=STRESS(1)
136            ENDIF
137          ENDIF
138 C       Tensile unloading
139           IF(DSTRAN(1) .GT. 0.)THEN
140             CALL UNLOADING(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
141      1 RPL,DDSDDT,DRPLDE,DRPLDT,
142      2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
143      3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
144      4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
145      5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3,I)
146           ENDIF
147         ENDIF
148
149
150 C       Step 3:Partially crushed (ep1>strain>=ep2) loading and unloading
151         IF(STRAN(1) .LT. ep1 .AND. STRAN(1) .GE. ep2)THEN
152 C         Compressive loading
153           IF(DSTRAN(1) .LE. 0.)THEN
154 C           Reloading from having been previously crushed
155             IF(STRAN(1) .GT. STATEV(2))THEN
156               IF(STATEV(2) .GE. ep2)THEN
157               DDSDDE(1,1)=STATEV(3)/STATEV(2)
158               ENDIF
159               IF(STATEV(2) .LT. ep2)THEN
160                 IF(STRESS(1) .GE. stt)THEN
161                   DDSDDE(1,1)=e1
162                   STATEV(5)=STRAN(1)
163                 ELSE
164                   DDSDDE(1,1)=ABS((STATEV(3)-stt)/(STATEV(5)-STATEV(2)))
165                 ENDIF
166               ENDIF
167             ELSE
168              DDSDDE(1,1)=e2
169              STATEV(1)=2
170             ENDIF
171             STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
172 C           This ensures that STATEV(2) contains the lowest strain reached
173 C           regardless of how many times the material has been reloaded.
174              IF(STRAN(1) .LT. STATEV(2))THEN
175                STATEV(2)=STRAN(1)
176                STATEV(3)=STRESS(1)
177              ENDIF
178           ENDIF
179 C         Tensile unloading
180           IF(DSTRAN(1) .GT. 0.)THEN
181             CALL UNLOADING(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
182      1 RPL,DDSDDT,DRPLDE,DRPLDT,
```

```
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
     5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3,I)
        ENDIF
       ENDIF

C      Step 4:Crushed (ep2>strain) loading and unloading
       IF(STRAN(1) .LT. ep2)THEN
C        Compressive loading
         IF(DSTRAN(1) .LE. 0.)THEN
C          Reloading from having been previously crushed
           IF(STRAN(1) .GT. STATEV(2))THEN
             IF(STATEV(2) .GE. ep2)THEN
             DDSDDE(1,1)=STATEV(3)/STATEV(2)
             ENDIF

C       IF stress exceeded stt during unloading:
            IF(STATEV(6) .EQ. 1)THEN
             IF(STATEV(2) .LT. ep2)THEN
               IF(STRESS(1) .GE. stt)THEN
                 DDSDDE(1,1)=e1
                 STATEV(5)=STRAN(1)
               ELSE
                 DDSDDE(1,1)=((stt-STATEV(3))/(STATEV(5)-STATEV(2)))
               ENDIF
             ENDIF
            ENDIF


C       IF stress hasn't exceeded stt during unloading:
            IF(STATEV(6) .EQ. 0)THEN
             IF(STATEV(2) .LT. ep2)THEN
               IF(STRESS(1) .LT. stt)THEN
                 DDSDDE(1,1)=e1
                 STATEV(5)=STRAN(1)
               ENDIF
             ENDIF
            ENDIF


C       Continue along plateau
           ELSE
            DDSDDE(1,1)=e3
            STATEV(1)=3
           ENDIF


           STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
C          This ensures that STATEV(2) contains the lowest strain reached
C          regardless of how many times the material has been reloaded.
            IF(STRAN(1) .LT. STATEV(2))THEN
              STATEV(2)=STRAN(1)
              STATEV(3)=STRESS(1)
            ENDIF
         ENDIF
C        Tensile unloading
         IF(DSTRAN(1) .GT. 0.)THEN
           CALL UNLOADING(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
```

```
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
     5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3,I)
        ENDIF
       ENDIF

        END SUBROUTINE QSI_INTACT


       SUBROUTINE UNLOADING(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
     5 ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3)

       INCLUDE 'ABA_PARAM.INC'

       CHARACTER*80 CMNAME
       DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
     3 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
       REAL:: ep1,ep2,ep3,sb,sc,scc,k,st,stt,kt,e1,e2,e3
C          Unloading when STATEV(2)>= ep1
           IF((STATEV(2)-ep1) .GE. 0.001)THEN
             DDSDDE(1,1)=e1
             STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
C         WRITE(*,'(4ES16.8,A8)')STATEV(1),STATEV(2),STATEV(3),
C       1 ep1,"Unload2"
           ENDIF
C          Unloading when ep1>STATEV(2)>=ep2
           IF((STATEV(2)-ep1) .LT. 0.001)THEN
             IF(STATEV(2) .GT. ep2)THEN
             DDSDDE(1,1)=STATEV(3)/STATEV(2)
             STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
C         WRITE(*,'(4ES12.4,A8)')STATEV(1),STATEV(2),STATEV(3),
C       1 ep1,"Unload3"
             ENDIF
           ENDIF
C          Unloading when ep2>STATEV(2): Two cases:
C          Unloading when STRESS(1)<=stt
           IF(STATEV(2) .LT. ep2 .AND. STRESS(1) .LE. stt)THEN
             DDSDDE(1,1)=e1
             STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
             IF(STRAN(1) .LT. ep2)THEN
               STATEV(4)=STRAN(1)
             ENDIF
             STATEV(6)=0
           ENDIF
C          Unloading when STRESS(1)>stt
           IF(STATEV(2) .LT. ep2 .AND. STRESS(1) .GT. stt)THEN
             DDSDDE(1,1)=((st-stt)/(0.-STATEV(4)))
             STRESS(1)=STRESS(1)+DDSDDE(1,1)*DSTRAN(1)
             STATEV(6)=1
           ENDIF
        END SUBROUTINE UNLOADING
```

APPENDIX 2: Core Damage Material Mode VUMAT

```
Subroutine VUMAT( &
  nblock,ndir,nshr,nstatev,nfieldv,nprops,lanneal,stepTime, &
  totalTime,dt,cmname,coordMp,charLength,props,density, &
  strainInc,relSpinInc,tempOld,stretchOld,defgradOld,fieldOld, &
  stressOld,stateOld,enerInternOld,enerInelasOld,tempNew, &
  stretchNew,defgradNew,fieldNew, &
  stressNew,stateNew,enerInternNew,enerInelasNew)

  Implicit Double Precision (a-h, o-z)
  Parameter (j_sys_Dimension = 2, n_vec_Length = 136, maxblk = n_vec_Length)

  Double Precision :: props(nprops), density_abq(nblock), coordMp(nblock,*), charLength(nblock,*),       &
    strainInc(nblock,ndir+nshr), relSpinInc(nblock,nshr), tempOld(nblock), tempNew(nblock),              &
    stretchOld(nblock,ndir+nshr), defgradOld(nblock,ndir+nshr+nshr), fieldOld(nblock,nfieldv),           &
    stressOld(nblock,ndir+nshr), stateOld(nblock,nstatev), enerInternOld(nblock), enerInelasOld(nblock), &
    stretchNew(nblock,ndir+nshr), defgradNew(nblock,ndir+nshr+nshr), fieldNew(nblock,nfieldv),           &
    stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev), enerInternNew(nblock), enerInelasNew(nblock)

  Double Precision :: stepTime, totalTime, dt

  Character(len=80) :: cmname

  Double Precision ep1, ep2, ep3, sb, sc, scc, k, st, stt, e1, e2, e3

  ! Read user-defined material properties
  ep1 = props(1)  ! compressive strain at damage initiation
  ep2 = props(2)  ! compressive strain after cell wall collapse
  ep3 = props(3)  ! compressive strain after continued crushing
  sb  = props(4)  ! compressive strength
  sc  = props(5)  ! compressive stress after cell wall collapse
  scc = props(6)  ! compressive stress after continued crushing
  k   = props(7)  ! ratio of tensile modulus to compressive modulus
  st  = props(8)  ! tensile strength
  stt = props(9)  ! tensile stress at which a crushed core's tangent stiffness changes from compressive modulus

  ! Calculate dependent properties
  e1 = sb/ep1  ! compressive modulus
  e2 = (sb - sc)/(ep1 - ep2)  ! tangent stiffness during cell wall collapse
  e3 = (scc - sc)/(ep3 - ep2)  ! tangent stiffness during continued crushing

  ! ------------------------------------------------------------------ !
  !    Solution-dependent variables (SDV):
  !    SDV1: Damage state, where:
  !              (-1)=Tensile loaded only (loading or unloading)
  !               (1)=Elastically compressed (0>strain>=ep1)
  !               (2)=Partially crushed (ep1>strain>=ep2)
```

```
47    !               (3)=Crushed (ep2>strain)
48    !    (inactive) (4)=Unloaded from crushed state with stress<stt
49    !    (inactive) (5)=Unloaded from crushed state with stress>stt
50    !    SDV2: Max applied -ve strain
51    !    SDV3: Stress at max applied -ve strain
52    !    SDV4: Strain when stressNew(1)=stt
53    !    SDV5: Strain when stress reaches stt during reloading
54    !    SDV6: Indicator for stress exceeding stt, where:
55    !               (0)=Stress didn't exceed stt
56    !               (1)=Stress exceeded stt
57    !    SDV7: Previous strain
58    ! ---------------------------------------------------------------------- !
59
60    Master: Do km = 1,nblock
61
62      Call QSI_INTACT(stressOld(km,1),stressNew(km,1),stateOld(km,:),stateNew(km,:),strainInc(km,1), &
63                      nstatev,ep1,ep2,ep3,sb,sc,scc,k,st,stt,e1,e2,e3)
64
65    end Do Master
66  ! ---------------------------------------------------------------------- !
67    Return
68  end Subroutine VUMAT
69
70
71  Subroutine CoreCrush(stressOld,stressNew,stateOld,stateNew,dstrain, &
72                       nstatev,ep1,ep2,ep3,sb,sc,scc,k,st,stt,e1,e2,e3)
73
74    Integer, intent(IN) :: nstatev
75    Double Precision, intent(IN) :: stressOld, stateOld(nstatev)
76    Double Precision, intent(OUT) :: stressNew, stateNew(nstatev)
77    Double Precision, intent(IN) :: dstrain
78    Double Precision, intent(IN) :: ep1,ep2,ep3,sb,sc,scc,k,st,stt,e1,e2,e3
79
80    Double Precision :: DDSDDE, strainOld, strain
81  ! ---------------------------------------------------------------------- !
82    stateNew(:) = stateOld(:)  ! Initialize state variables
83
84    strainOld = stateOld(7)  ! Previous strain
85    strain = strainOld + dstrain  ! Current strain
86    stateNew(7) = strain  ! Store updated strain
87
88    If (strain > 0 .AND. stateOld(2) >= ep1) Then  ! Tensile loading
89      DDSDDE = k*e1
90      stressNew = stressOld + DDSDDE*dstrain
91      stateNew(1) = -1
92
93    Else If (dstrain <= 0) Then  ! Compressive loading
```

```
94      If (strain <= stateOld(2)) Then  ! If strain is at a new high,
95        Loading: If (strain >= ep1) Then  ! Elastically compressed
96          DDSDDE = e1
97          stateNew(1) = 1
98        Else If (strain >= ep2) Then Loading  ! Partially crushed
99          DDSDDE = e2
100         stateNew(1) = 2
101       Else Loading  ! Fully crushed
102         DDSDDE = e3
103         stateNew(1) = 3
104       end If Loading
105
106       stressNew = stressOld + DDSDDE*dstrain
107       stateNew(2) = strain
108       stateNew(3) = stressNew
109       stateNew(6) = 0
110
111     Else  ! If the core is being reloaded,
112       Reloading: If (stateOld(2) >= ep1) Then  ! Elastically compressed
113         DDSDDE = e1
114         stressNew = stressOld + DDSDDE*dstrain
115         stateNew(1) = 1
116
117       Else If (stateOld(2) >= ep2) Then Reloading  ! Partially crushed
118         DDSDDE = stateOld(3)/stateOld(2)
119         stressNew = stressOld + DDSDDE*dstrain
120
121       Else Reloading  ! Fully crushed
122         If (stateOld(6) == 1) Then  ! Unloaded past stt
123           If (stressOld >= stt) Then
124             DDSDDE = e1
125             stressNew = stressOld + DDSDDE*dstrain
126             If (stressNew < stt) Then
127               stateNew(5) = (stt - stressOld)/e1 + strainOld
128               DDSDDE = ((stt - stateOld(3))/(stateNew(5) - stateOld(2)))
129               stressNew = stt + DDSDDE*(strain - stateNew(5))
130             end If
131           Else
132             DDSDDE = (stressOld - stateOld(3))/(strainOld - stateOld(2))
133             stressNew = stressOld + DDSDDE*dstrain
134           end If
135
136         Else  ! Not unloaded past stt
137           If (stressOld < stt) Then
138             DDSDDE = e1
139             stressNew = stressOld + DDSDDE*dstrain
140           end If
```

```
141         end If
142       end If Reloading
143     end If
144
145   Else  ! If the core is being unloaded,
146 !   Unloading when undamaged:
147     Unloading: If (stateOld(2) >= ep1) Then
148       DDSDDE = e1
149       stressNew = stressOld + DDSDDE*dstrain
150
151 !   Unloading when partially crushed:
152     Else If (stateOld(2) >= ep2) Then Unloading
153       DDSDDE = stateOld(3)/stateOld(2)
154       stressNew = stressOld + DDSDDE*dstrain
155       If (stressNew >= stt) stateNew(6) = 1
156
157 !   Unloading when fully crushed:
158     Else Unloading
159       If (stressOld <= stt .AND. stateOld(6) == 0) Then
160         DDSDDE = e1
161         stressNew = stressOld + DDSDDE*dstrain
162         If (stressNew > stt) Then
163           stateNew(4) = strainOld + (stt - stressOld)/e1
164           DDSDDE = -(st - stt)/stateNew(4)
165           stressNew = stt + DDSDDE*(strain - stateNew(4))
166           stateNew(6) = 1
167         end If
168
169       Else If (stressOld <= stt.AND.stateOld(6) == 1) Then
170         DDSDDE = ((stt - stateOld(3))/(stateOld(5) - stateOld(2)))
171         stressNew = stressOld + DDSDDE*dstrain
172         If (stressNew > stt) Then
173           DDSDDE = e1
174           stressNew = stt + DDSDDE*(strain - stateOld(5))
175         end If
176
177       Else
178         DDSDDE = (st - stressOld)/(-strainOld)
179         stressNew = stressOld + DDSDDE*dstrain
180         stateNew(6) = 1
181       end If
182
183     end If Unloading
184   end If
185
186   Return
187 end Subroutine CoreCrush
```