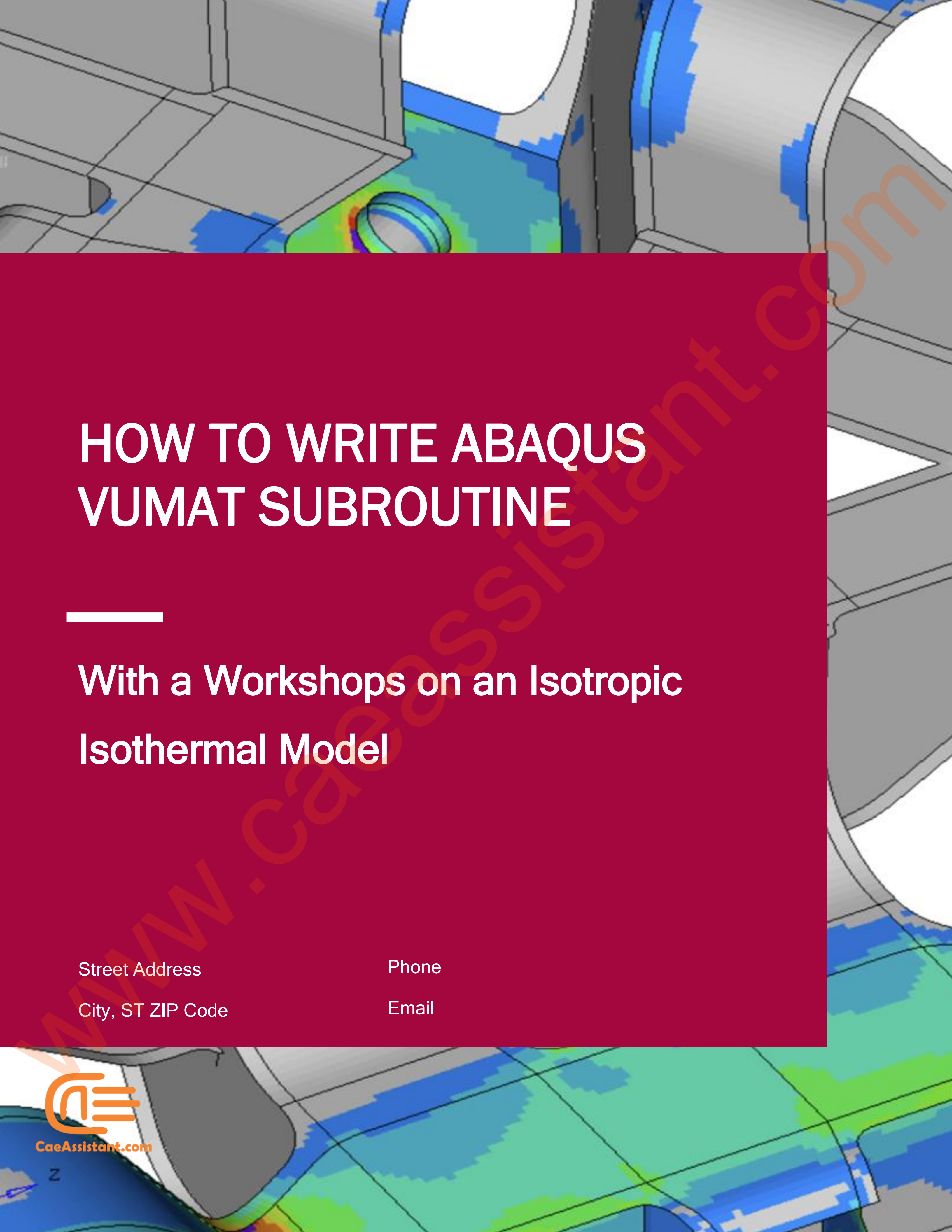


HOW TO WRITE ABAQUS VUMAT SUBROUTINE

With a Workshops on an Isotropic Isothermal Model

Street Address

City, ST ZIP Code

Phone

Email

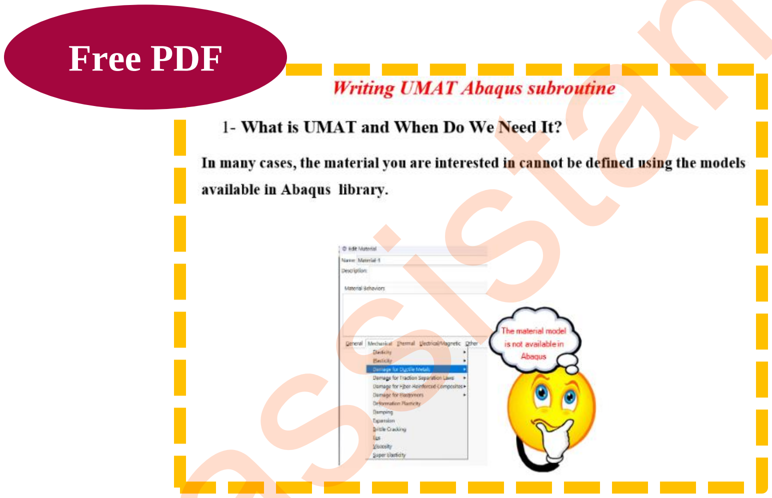
CaeAssistant.com

Table of Contents

Writing VUMAT Abaqus subroutine

Introduction

UMAT and **VUMAT** are two of the most widely used subroutines in Abaqus, enabling you to define custom materials within the software. In another FREE PDF, we covered this topic in detail, providing an in-depth discussion of the UMAT subroutine, along with two workshops.


Free PDF

Writing UMAT Abaqus subroutine

1- **What is UMAT and When Do We Need It?**

In many cases, the material you are interested in cannot be defined using the models available in Abaqus library.

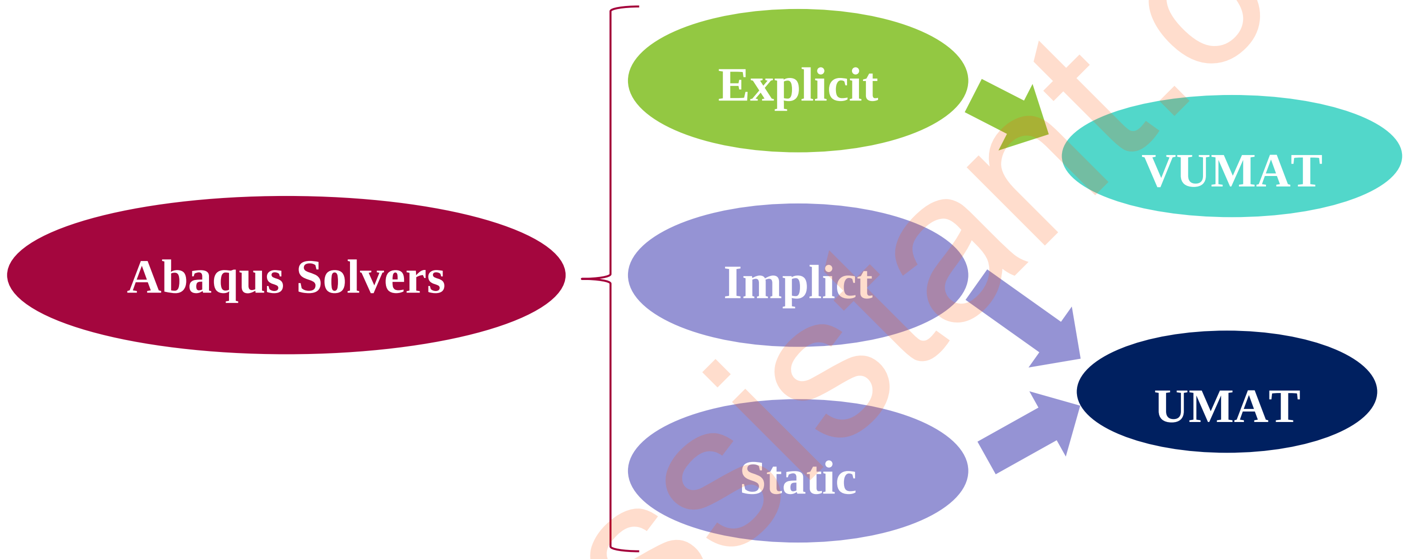
In the current PDF, our focus is on the VUMAT subroutine instead. For better understanding, we have prepared a workshop, where we explain how to model a material using VUMAT in Abaqus.

By reading this PDF, you will be able to write your first VUMAT subroutine.

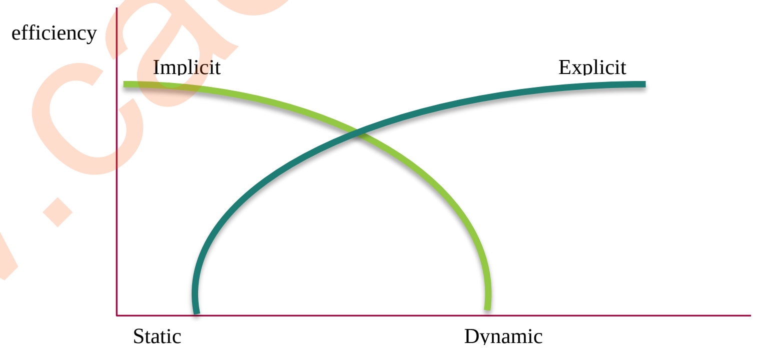
The input files and the VUMAT code related to the workshop are available on our GitHub page, and you can download them for free.

1- UMAT vs. VUMAT: How to choose the appropriate subroutine for your needs

In general, the difference between UMAT and VUMAT is that UMAT is developed for the Abaqus Standard solver, while VUMAT is developed for the Abaqus Explicit solver.



The advantage of the Explicit solver is that it does not require the formation of the stiffness matrix or convergence control, as is the case with the Implicit or static solvers. Therefore, the solution progresses incrementally without checking for convergence after each increment.



For this reason, the Explicit solver is typically used for dynamic problems and severe nonlinear problems, where convergence in Static solver can be challenging.

Notice that, when the mesh is very fine, the increments become small, and the Explicit solver can become time-consuming. In such cases, using the static solver may be more efficient.

Depending on your problem, you should first determine which solver to use and then select the appropriate subroutine, either UMAT or VUMAT. If you'd like to learn more about this topic, we have a blog post that explains it, and you can read it through this link.

2- UMAT vs VUMAT Subroutine: Differences

UMAT and VUMAT serve similar purposes in Abaqus. Their main difference is that UMAT is used for solving problems with the standard solver, while VUMAT is used for solving problems with the explicit solver. But do you think there are other differences as well?

- *The first difference is that in VUMAT, you don't need to define the Jacobian matrix; it's enough to define the stress components in the code.*

$$\sigma \overset{?}{\propto} \varepsilon$$

- *The second difference is that the order of the stress and strain components in* ***VUMAT*** *differs from that in* ***UMAT****. For symmetric 3D materials, the order is as follows:*

UMAT	VUMAT
$\begin{pmatrix}\sigma_{11}\\\sigma_{22}\\\sigma_{33}\\\sigma_{23}\\\sigma_{31}\\\sigma_{12}\end{pmatrix}$	$\begin{pmatrix}\sigma_{11}\\\sigma_{22}\\\sigma_{33}\\\sigma_{12}\\\sigma_{23}\\\sigma_{31}\end{pmatrix}$

- *Third Difference: In UMAT, the stran vector provides the engineering shear strain in the following form.*

$$\begin{pmatrix}\varepsilon_{11}\\\varepsilon_{22}\\\varepsilon_{33}\\2\varepsilon_{23}\\2\varepsilon_{31}\\2\varepsilon_{12}\end{pmatrix}=\begin{pmatrix}\varepsilon_{11}\\\varepsilon_{22}\\\varepsilon_{33}\\\gamma_{23}\\\gamma_{31}\\\gamma_{12}\end{pmatrix}$$

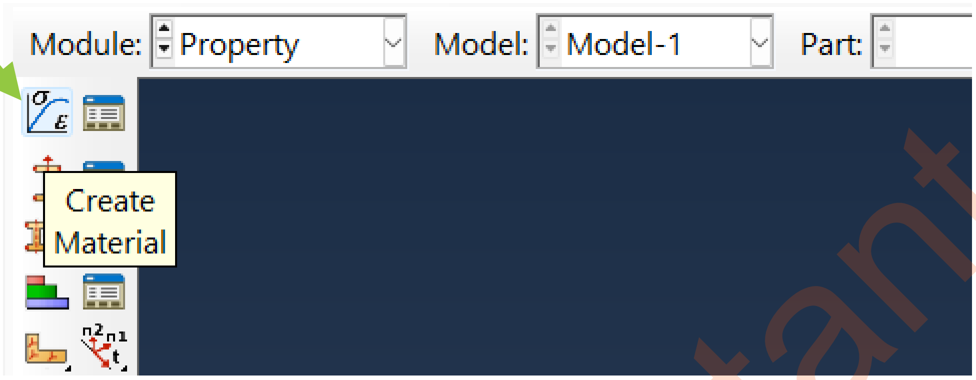
In contrast, its equivalent in VUMAT is stored as **strainIncrement**, which represents the strain increment and is expressed as follows.

$$\begin{pmatrix}\Delta\varepsilon_{11}\\\Delta\varepsilon_{22}\\\Delta\varepsilon_{33}\\\Delta\varepsilon_{12}\\\Delta\varepsilon_{23}\\\Delta\varepsilon_{31}\end{pmatrix}$$

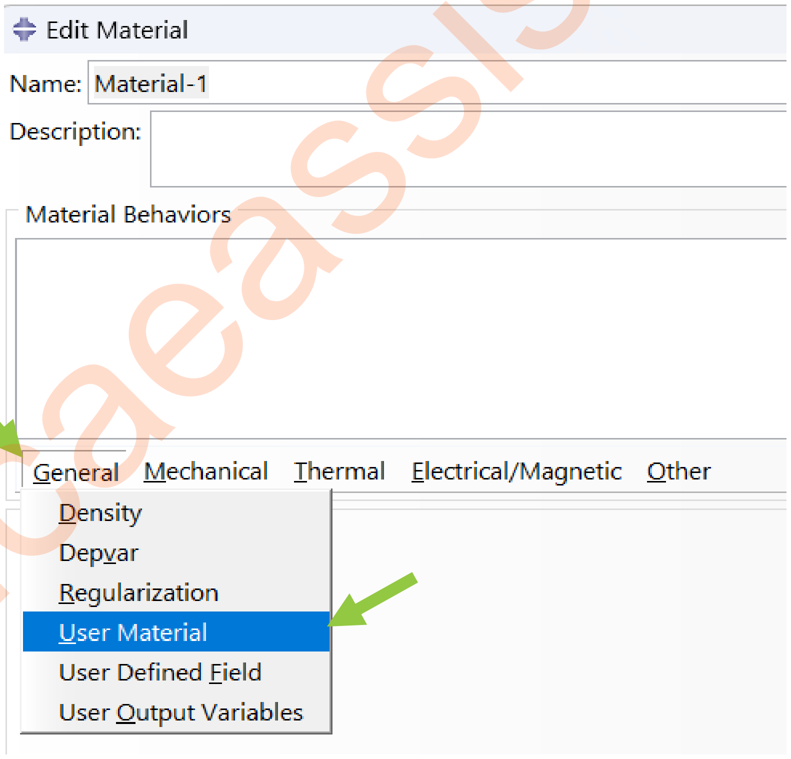
Therefore, to calculate the stress in VUMAT, we need to multiply the stiffness by twice the increment of shear strain values.

3- How to Implement VUMAT in Abaqus Model?

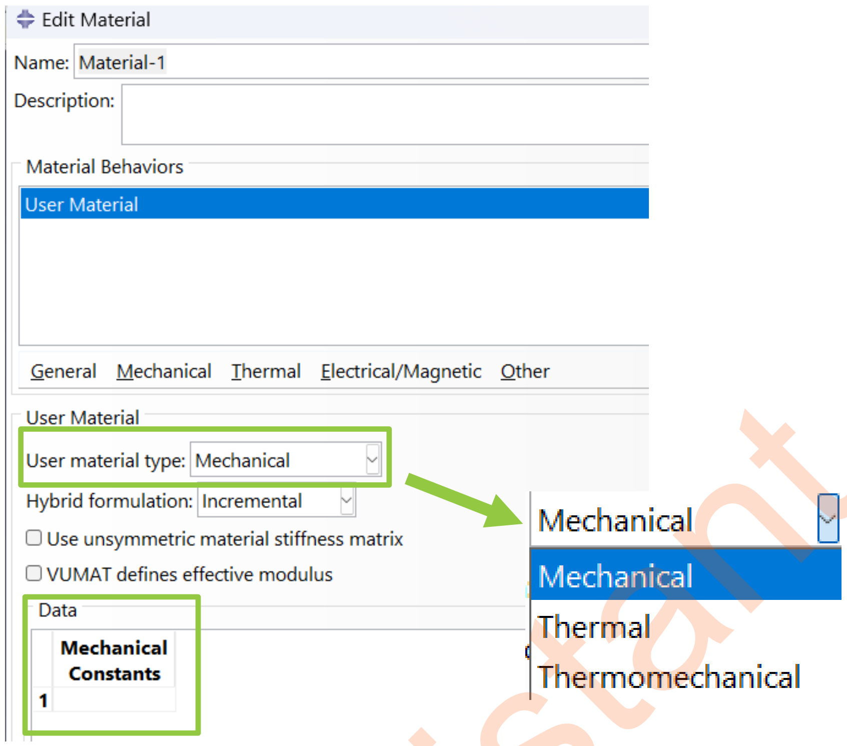
Implementing **VUMAT** in Abaqus is similar to implementing **UMAT**. You can define the material as shown in the figure below, with the only difference being that you need to define the step as explicit instead of general static or implicit.



Next, as shown in the next figure, click on General and select User Material.



When you select this option, a table similar to the one shown in next figure will appear.


Edit Material

Name: Material-1

Description:

Material Behaviors

User Material

General Mechanical Thermal Electrical/Magnetic Other

User Material

User material type: Mechanical

Hybrid formulation: Incremental

☐ Use unsymmetric material stiffness matrix

☐ VUMAT defines effective modulus

Data

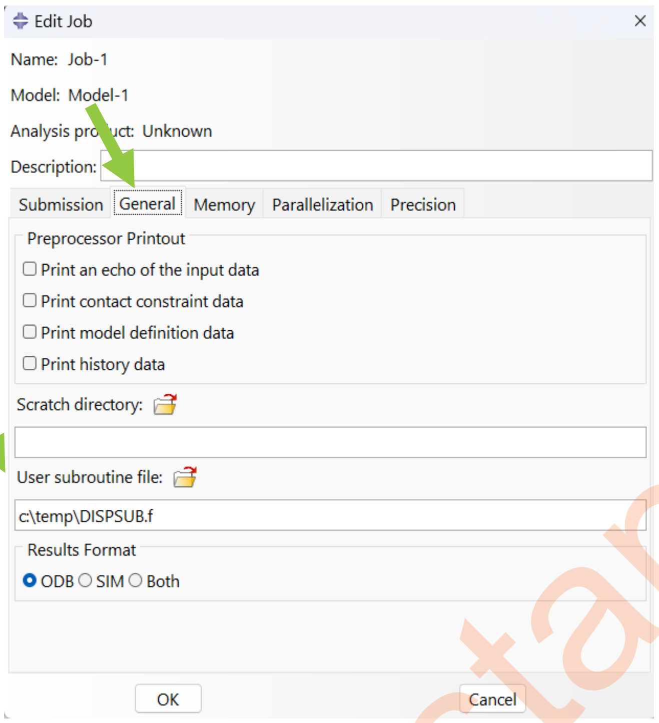
	Mechanical Constants
1	

As shown in figure, you can define a series of mechanical or thermal constants in this table, as we have discussed with details for the PDF on the implementation of UMAT subrputine.

- ***Notive that,*** *if your material is unsymmetric and you enable the respective checkbox, make sure to define the stress components in the VUMAT code according to the following format.*

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{21} \\ \sigma_{32} \\ \sigma_{13} \end{pmatrix}$$

Finally, you need to specify the path to the Fortran file containing the VUMAT subroutine, in your Abaqus model, for its implementation (according to the next figure).



Edit Job

Name: Job-1

Model: Model-1

Analysis product: Unknown

Description:

Preprocessor Printout

☐ Print an echo of the input data

☐ Print contact constraint data

☐ Print model definition data

☐ Print history data

Scratch directory:

User subroutine file:

c:\temp\DISPSUB.f

Results Format

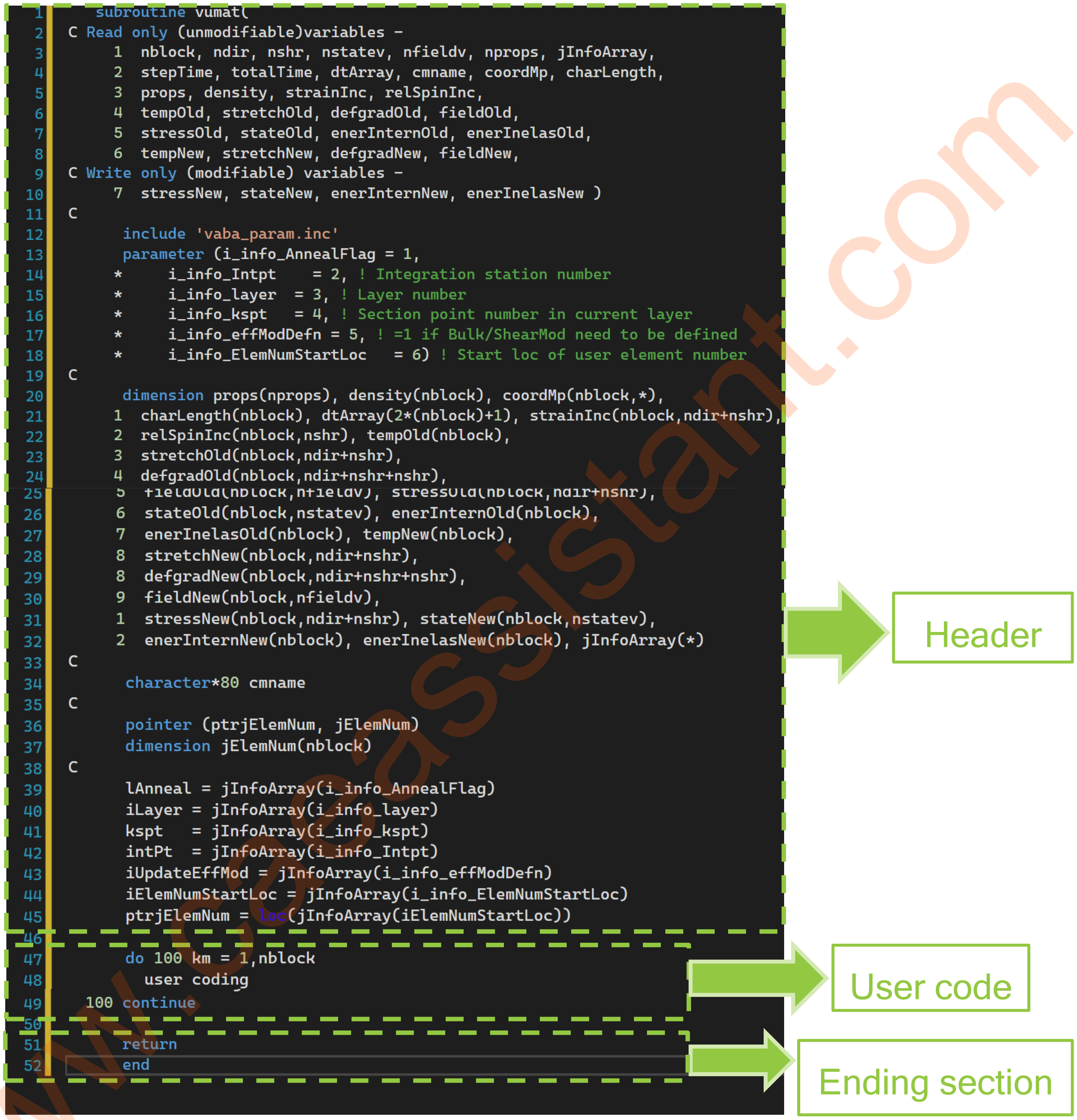
[✓ ODB] SIM Both

3-1- Subroutine's interface

- *The interface refers to the overall structure of a subroutine, in which our code needs to be written.*

- *The interface consists of a heading that includes the parameters required for the subroutine.*

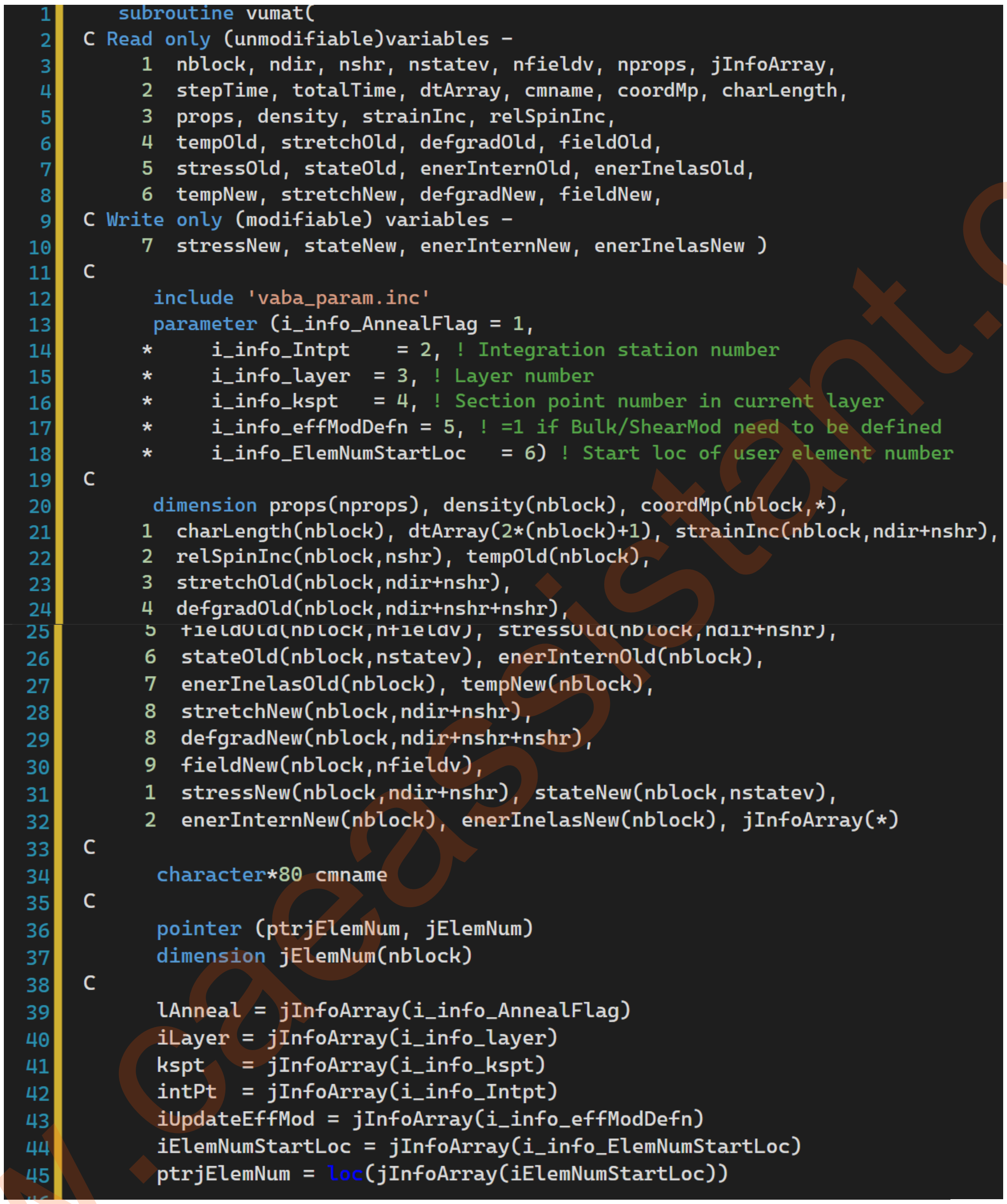
- *The interface of the VUMAT subroutine is shown in the figure below.*



```
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, jInfoArray,
     2  stepTime, totalTime, dtArray, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
      parameter (i_info_AnnealFlag = 1,
     *     i_info_Intpt    = 2, ! Integration station number
     *     i_info_layer  = 3, ! Layer number
     *     i_info_kspt   = 4, ! Section point number in current layer
     *     i_info_effModDefn = 5, ! =1 if Bulk/ShearMod need to be defined
     *     i_info_ElemNumStartLoc   = 6) ! Start loc of user element number
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), dtArray(2*(nblock)+1), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock), jInfoArray(*)
C
      character*80 cmname
C
      pointer (ptrjElemNum, jElemNum)
      dimension jElemNum(nblock)
C
      lAnneal = jInfoArray(i_info_AnnealFlag)
      iLayer = jInfoArray(i_info_layer)
      kspt   = jInfoArray(i_info_kspt)
      intPt  = jInfoArray(i_info_Intpt)
      iUpdateEffMod = jInfoArray(i_info_effModDefn)
      iElemNumStartLoc = jInfoArray(i_info_ElemNumStartLoc)
      ptrjElemNum = loc(jInfoArray(iElemNumStartLoc))

      do 100 km = 1,nblock
        user coding
  100 continue

      return
      end
```

3-1- HEADER

You can simply copy the header from the Abaqus documentation and paste it into your file.



```
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, jInfoArray,
     2  stepTime, totalTime, dtArray, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
      parameter (i_info_AnnealFlag = 1,
     *     i_info_Intpt    = 2, ! Integration station number
     *     i_info_layer  = 3, ! Layer number
     *     i_info_kspt   = 4, ! Section point number in current layer
     *     i_info_effModDefn = 5, ! =1 if Bulk/ShearMod need to be defined
     *     i_info_ElemNumStartLoc   = 6) ! Start loc of user element number
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), dtArray(2*(nblock)+1), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock), jInfoArray(*)
C
      character*80 cmname
C
      pointer (ptrjElemNum, jElemNum)
      dimension jElemNum(nblock)
C
      lAnneal = jInfoArray(i_info_AnnealFlag)
      iLayer = jInfoArray(i_info_layer)
      kspt   = jInfoArray(i_info_kspt)
      intPt  = jInfoArray(i_info_Intpt)
      iUpdateEffMod = jInfoArray(i_info_effModDefn)
      iElemNumStartLoc = jInfoArray(i_info_ElemNumStartLoc)
      ptrjElemNum = loc(jInfoArray(iElemNumStartLoc))
```

Some of the header's parameters must be defined in the user code, while others are provided for informational purposes. We will discuss the paramaters that you need to define, in following.

3-1-1- VARIABLES TO BE DEFINED

A list of the variables that you must update in the code, is provided in the table below.

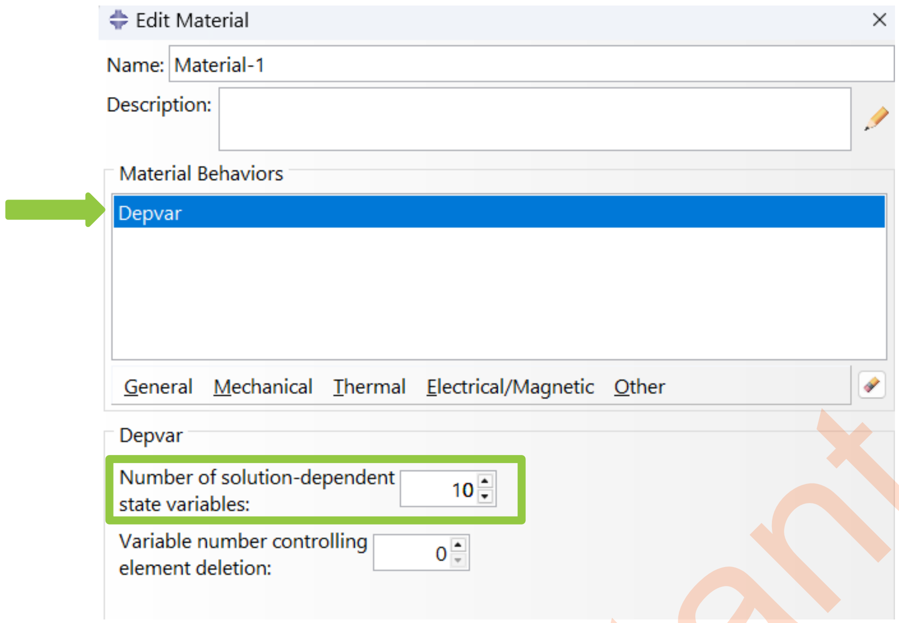
Variable		Definition
stressNew	In all situations	Stress tensor at each material point
stateNew	In all situations (state variables are used)	Stress tensor at each material point

According to the table, the first parameter to define in the VUMAT subroutine is stressNew. Here, stressNew represents the stress components at the end of increment, that you need to define in the subroutine to be send to Abaqus.

Interestingly, this subroutine includes a parameter called stressOld, which represents the stress value at the beginning of the increment and is provided for informational purposes only. This allows us to take its value, add the stress changes from the current increment, and define stressNew, according to the following figure.

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{pmatrix}$$

stateNew represents the state variables defined by the user, which must be explicitly specified for the subroutine to recognize them. To do this, you need to define the number of state variables in the material properties section within Abaqus, as shown in the figure below.


Edit Material

Name: Material-1

Description:

Material Behaviors

Depvar

Depvar

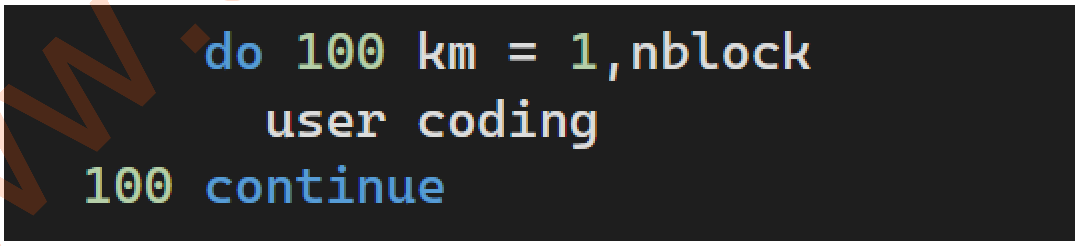
Number of solution-dependent state variables: 10

Variable number controlling element deletion: 0

It is important to note that you only need to update this variable if it is required for your implementation and has been defined in the software. Otherwise, updating it is unnecessary.

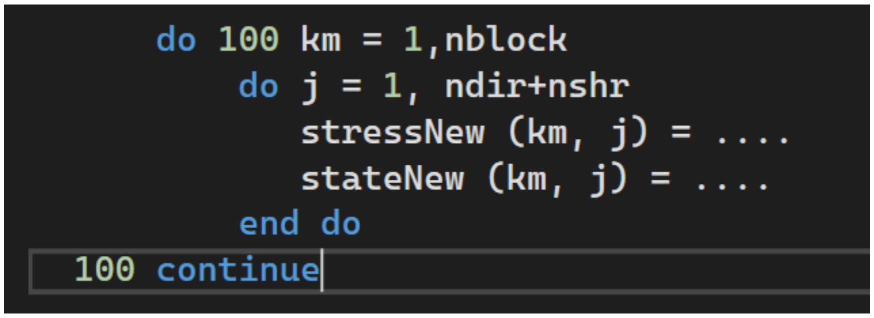
3-1-2- DIMENSIONS OF STRESSNEW AND STATENEW IN VUMAT SUBROUTINE

UMAT subroutine automatically iterates through all material points, so you don't need to define a loop. However, in the VUMAT subroutine, you must explicitly create a loop over the material points and update the stress components within it. The coding pattern for this process is illustrated in the figure below, and we have provided a more detailed explanation in the workshop section.



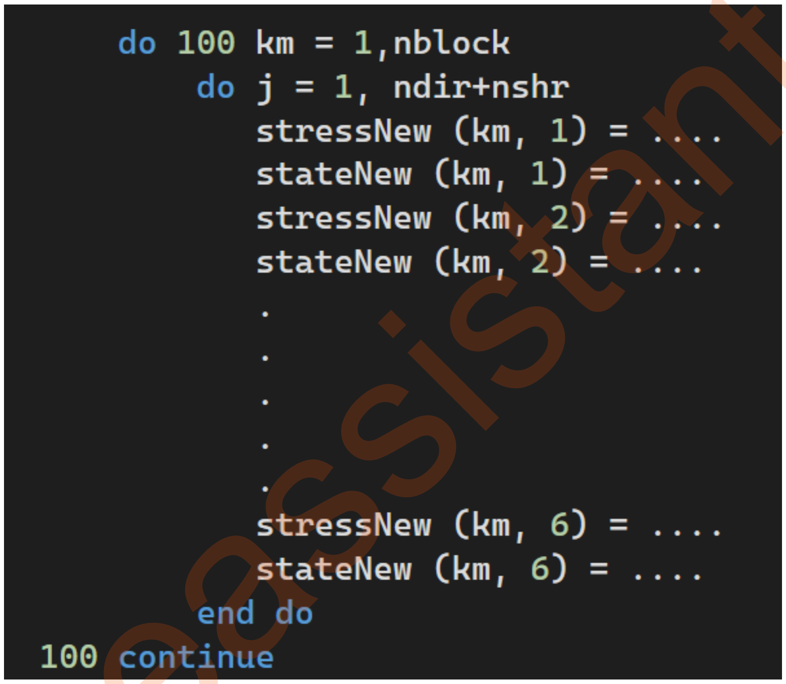
```
    do 100 km = 1,nblock
      user coding
100 continue
```

Accordingly, when you want to update stress or statev, you need define them in the following way.



```
      do 100 km = 1,nblock
          do j = 1, ndir+nshr
             stressNew (km, j) = ....
             stateNew (km, j) = ....
          end do
100   continue
```

Alternatively, you can define the stress components and state variables in the following way.



```
      do 100 km = 1,nblock
          do j = 1, ndir+nshr
             stressNew (km, 1) = ....
             stateNew (km, 1) = ....
             stressNew (km, 2) = ....
             stateNew (km, 2) = ....
             .
             .
             .
             .
             .
             stressNew (km, 6) = ....
             stateNew (km, 6) = ....
          end do
100   continue
```

As a conclusion, just note that stressNew and stateNew in VUMAT are represented as matrices with dimensions of nblock and ndir + nshr, where nblock is a variable passed in VUMAT for informational purposes, ndir represents the number of direct stress components, and nshr represents the number of shear stress components, respectively.

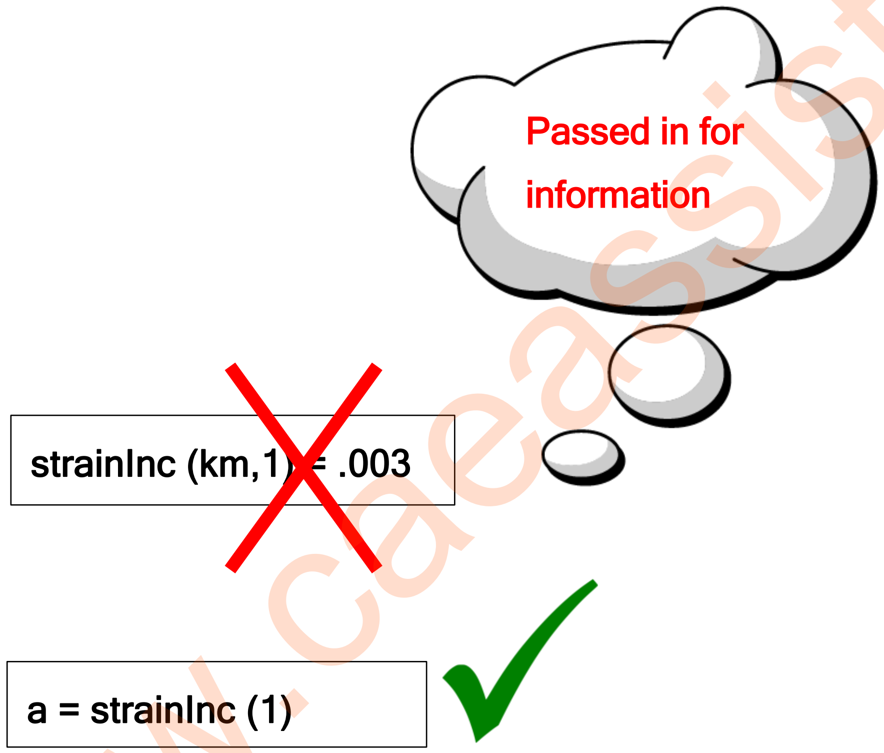
3-1-3- VARIABLES PASSED IN FOR INFORMATION

The material behavior at any given moment may depend on various parameters such as time, coordinate, current strain or stress.

Accordingly, there are certain variables within the VUMAT that provide you with the necessary information. For example, **strainInc** is a matrix with dimensions of (nblock, ndir + nshr), representing the variation of strain in the current increment.

Another example is stepTime, which represents the current time from the beginning of the step. Many other parameters for verification are included in the VUMAT subroutine, where you can check them and their roles in the Abaqus documentation.

Notice that, When a parameter inside the subroutine is provided only for informational purposes, you cannot assign a value to it, which leads to errors in your code.



```
strainInc (km,1) = .003
```

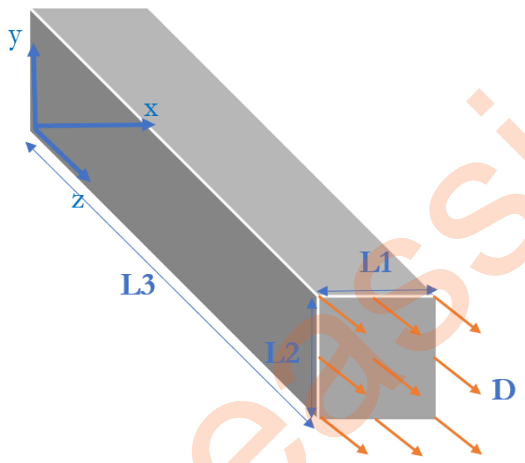
```
a = strainInc (1)
```

4- Workshop

In this section, we will conduct a workshop to teach you how to write VUMAT code for various problems and use it effectively.

Workshop 1: Isotropic Isothermal problem

- *Isotropic refers to a property of materials or systems that exhibit the same characteristics or behavior in all directions.*
- *Isothermal refers to a process or condition in which the temperature remains constant.*
- *The problem characteristics are shown in the next figure.*



Property

Elastic:
- E = 210 GPa
- ν = 0.3

Loads:
- D = 5 mm

Dimensions:
- L1 = 50 mm
- L2 = 50 mm
- L3= 600 mm

According to the figure, the beam is fixed in xy plan.

The generalized Hook's low in Matrix form for such an Isotropic problem is shown in the following figure.

$$\begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{31} \\ 2\varepsilon_{12} \end{pmatrix} = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{pmatrix} = \frac{1}{E}\begin{pmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(1+\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2(1+\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2(1+\nu) \end{pmatrix}\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{12} \end{pmatrix}$$

In which $\boldsymbol{\gamma}_{\mathbf{ij}=2\boldsymbol{\varepsilon}_{\mathbf{ij}}}$ is defined as the shear engineering strain.

Conversely, the stress in the material can be calculated from the strain using the following equation.

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{12} \end{pmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2 \end{pmatrix}\begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{31} \\ 2\varepsilon_{12} \end{pmatrix}$$

The above equation can be rewritten in the following form using the Lamé constants.

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{12} \end{pmatrix} = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}\begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{31} \\ 2\varepsilon_{12} \end{pmatrix}$$

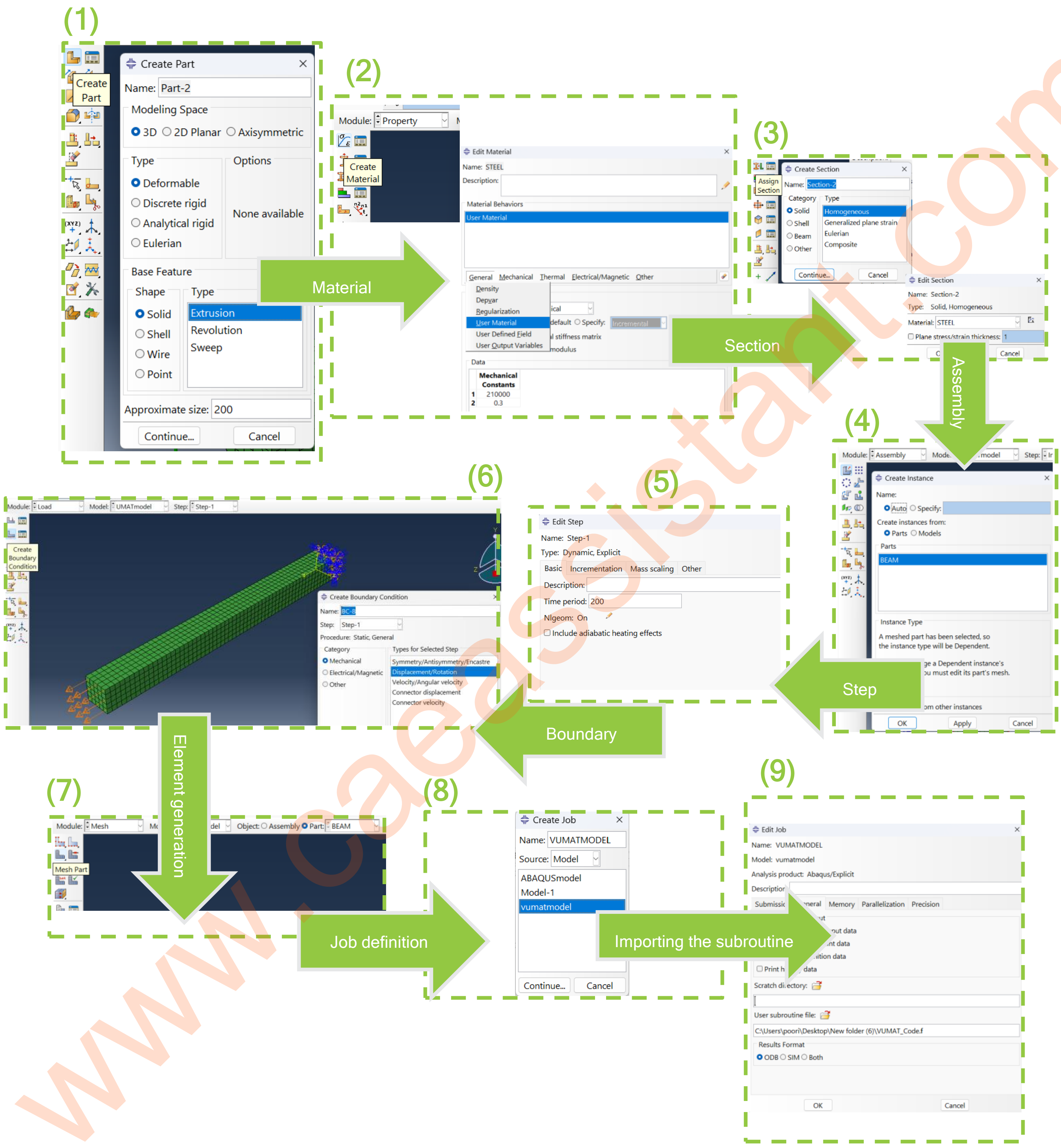
In which λ is the Lamé constant, and μ is the shear modulus, both defined as follows.

$$\lambda = \frac{E\nu}{(1-2\nu)(1+\nu)} \qquad \mu = \frac{E}{2(1+\nu)}$$

Developing the VUMAT code for this problem is the focus of this workshop.

Before we start writing the VUMAT code, let’s take a moment to review the modeling process in Abaqus together. This is illustrated in the next figure.

1- *According to the figure, first, you need to create 3D deformable part.*
2- *Then you need to go to the property module and define the material properties*
3- *Then you need to define a section and assign it to the specimen.*
4- *Next, you must create an instance of the model in assembly module*
5- *Afterward, create a static general step*
6- *Now, you can apply boundary conditions, including the fixed end and the applied displacement*
7- *Then define the elements*
8- *Next, create a job in the Job module*
9- *Finally, add the fortran subroutine file address to edit job general menu.*



(1) Create Part

Name: Part-2

Modeling Space: [✓ 3D] 2D Planar Axisymmetric

Type: [✓ Deformable] Discrete rigid Analytical rigid Eulerian

Options: None available

Base Feature

Shape: [✓ Solid] Shell Wire Point

Type: [✓ Extrusion] Revolution Sweep

Approximate size: 200

Continue... Cancel

(2) Module: Property

Edit Material

Name: STEEL

Description:

Material Behaviors: User Material

General Mechanical Thermal Electrical/Magnetic Other

Density
Depvar
Regularization
[✓ User Material]
User Defined Field
User Output Variables

Data

	Mechanical Constants
1	210000
2	0.3

(3) Create Section

Name: Section-2

Category: [✓ Solid] Shell Beam Other

Type: [✓ Homogeneous] Generalized plane strain Eulerian Composite

Continue... Cancel

Edit Section

Name: Section-2

Type: Solid, Homogeneous

Material: STEEL

Plane stress/strain thickness: 1

Cancel

(4) Module: Assembly

Create Instance

Name: [✓ Auto] Specify:

Create instances from: [✓ Parts] Models

Parts: BEAM

Instance Type

A meshed part has been selected, so the instance type will be Dependent.

ge a Dependent instance's
ou must edit its part's mesh.

om other instances

OK Apply Cancel

(5) Edit Step

Name: Step-1

Type: Dynamic, Explicit

Basic Incrementation Mass scaling Other

Description:

Time period: 200

Nlgeom: On

Include adiabatic heating effects

(6) Module: Load Model: UMATmodel Step: Step-1

Create Boundary Condition

Name: BC-8

Step: Step-1

Procedure: Static, General

Category: [✓ Mechanical] Electrical/Magnetic Other

Types for Selected Step: Symmetry/Antisymmetry/Encastre [✓ Displacement/Rotation] Velocity/Angular velocity Connector displacement Connector velocity

(7) Module: Mesh Object: Assembly [✓ Part:] BEAM

(8) Create Job

Name: VUMATMODEL

Source: Model

ABAQUSmodel
Model-1
[✓ vumatmodel]

Continue... Cancel

(9) Edit Job

Name: VUMATMODEL

Model: vumatmodel

Analysis product: Abaqus/Explicit

Description:

Submission General Memory Parallelization Precision

Print h data

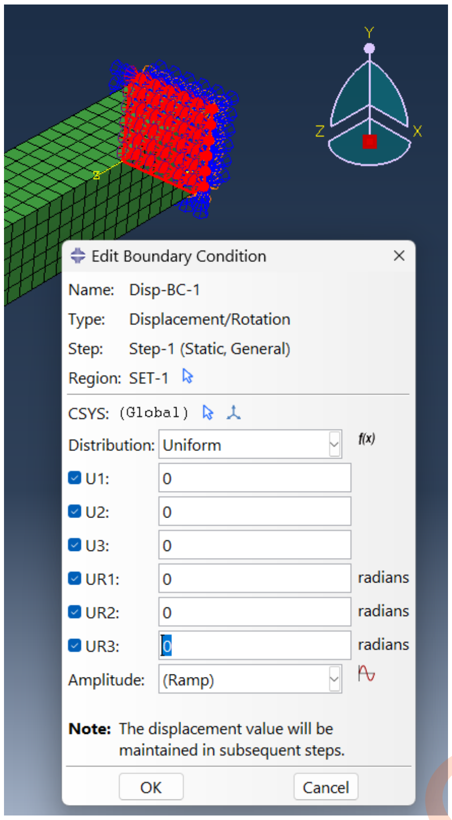
Scratch directory:

User subroutine file: C:\Users\poori\Desktop\New folder (6)\VUMAT_Code.f

Results Format: [✓ ODB] SIM Both

OK Cancel

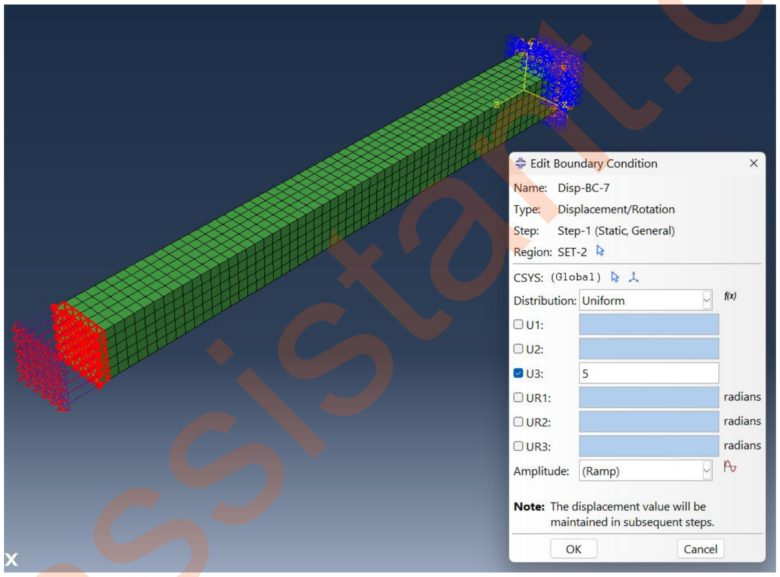
- *Notice that, for the boundary conditions, you need to define two displacement-type boundary conditions at the start and end of the beam, with the details shown in the figure below.*



Edit Boundary Condition

Name: Disp-BC-1
Type: Displacement/Rotation
Step: Step-1 (Static, General)
Region: SET-1
CSYS: (Global)
Distribution: Uniform
☑ U1: 0
☑ U2: 0
☑ U3: 0
☑ UR1: 0 radians
☑ UR2: 0 radians
☑ UR3: 0 radians
Amplitude: (Ramp)

Note: The displacement value will be maintained in subsequent steps.



Edit Boundary Condition

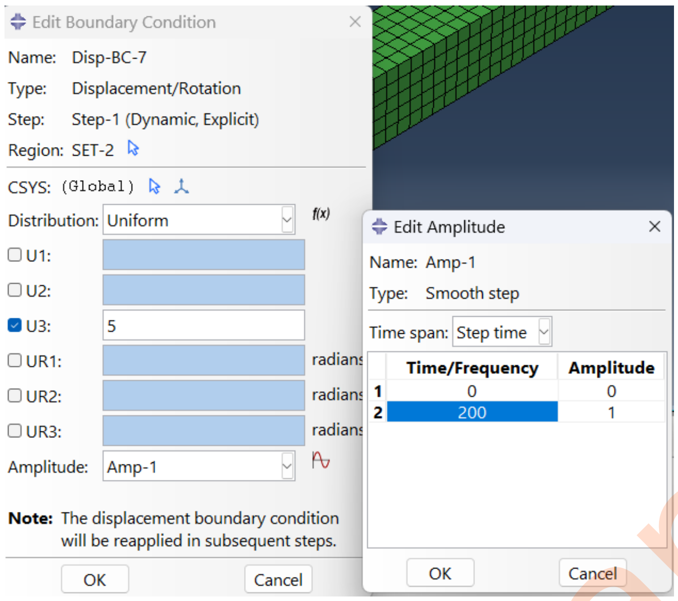
Name: Disp-BC-7
Type: Displacement/Rotation
Step: Step-1 (Static, General)
Region: SET-2
CSYS: (Global)
Distribution: Uniform
☐ U1:
☐ U2:
☑ U3: 5
☐ UR1: radians
☐ UR2: radians
☐ UR3: radians
Amplitude: (Ramp)

Note: The displacement value will be maintained in subsequent steps.

Applied displacement

Fixed end

In addition, since we want the loading to follow a quasi-static form, we need to define a smooth step for the boundary condition, ensuring that at the beginning of the solution, the loading rate is low and gradually increases. The method for applying smooth step loading is shown in figure below.


Edit Boundary Condition

Name: Disp-BC-7

Type: Displacement/Rotation

Step: Step-1 (Dynamic, Explicit)

Region: SET-2

CSYS: (Global)

Distribution: Uniform

U1:

U2:

U3: 5

UR1: radians

UR2: radians

UR3: radians

Amplitude: Amp-1

Note: The displacement boundary condition will be reapplied in subsequent steps.

Edit Amplitude

Name: Amp-1

Type: Smooth step

Time span: Step time

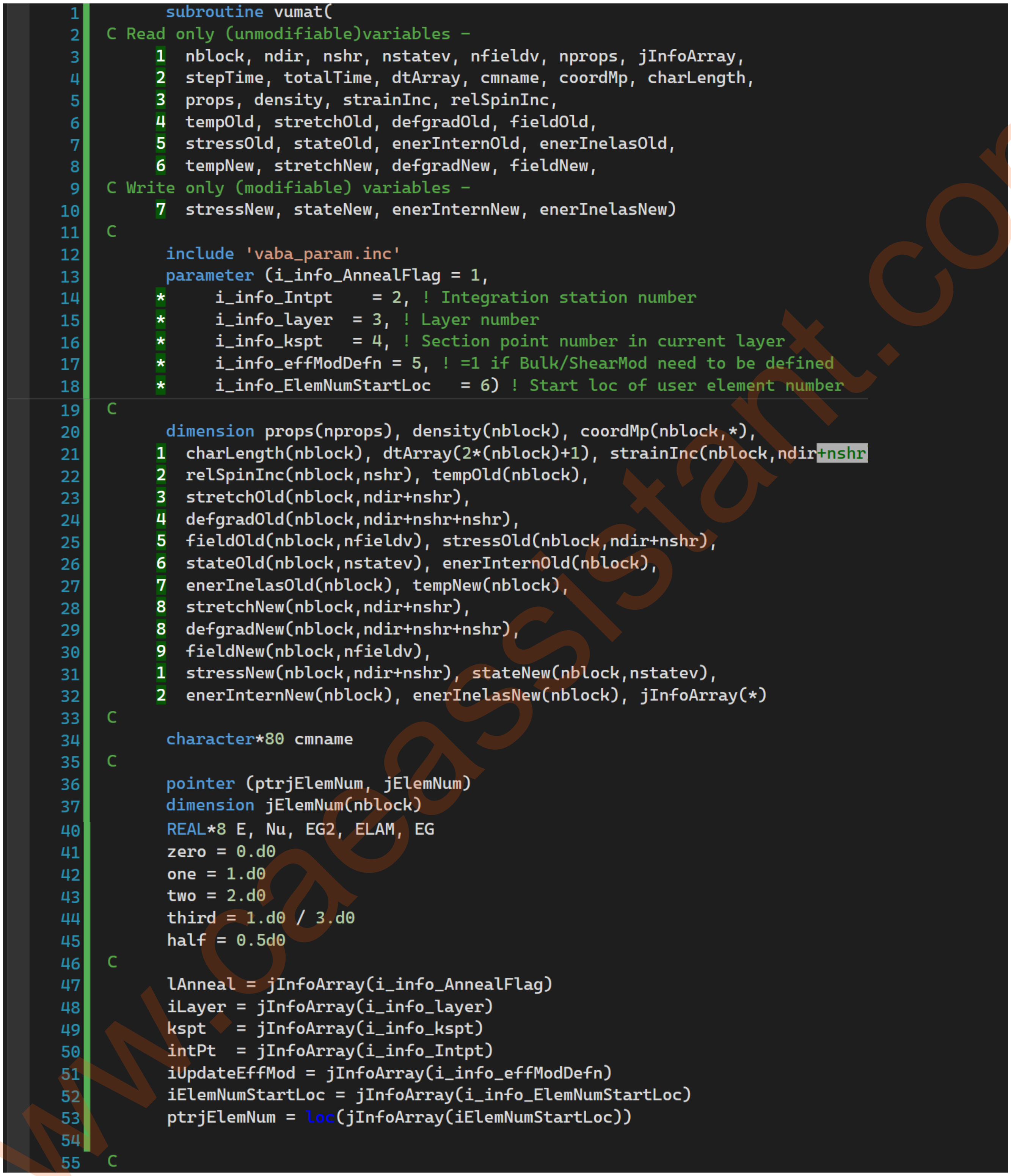
	Time/Frequency	Amplitude
1	0	0
2	200	1

As shown in this figure, we define the amplitude so that the load factor starts at zero at the beginning of the solution and gradually increases to one by the end of the step, which lasts 200 seconds. This factor is then multiplied by the defined displacement value of 5.

Let's see how the VUMAT code is written for such a problem.

As shown in the next figure, we first copied the subroutine header from the Abaqus documentation and pasted it into the Fortran code.

Then, we defined several parameters in the real(8) format, including the modulus of elasticity, Poisson’s ratio, a coefficient for the shear modulus, as well as the values for zero, one, and two.

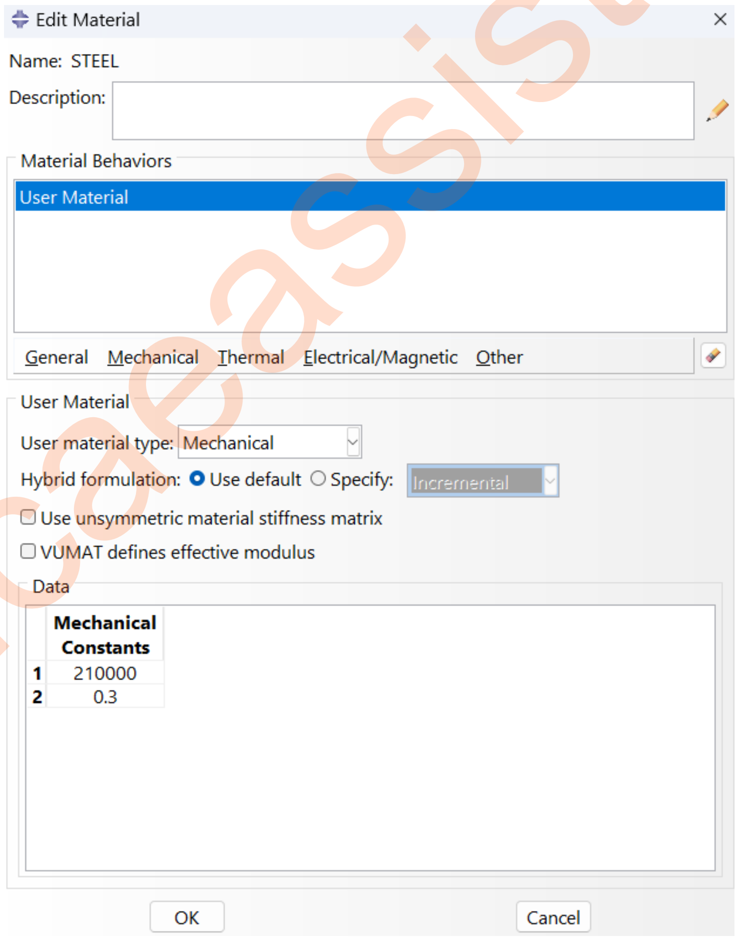


```fortran
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, jInfoArray,
     2  stepTime, totalTime, dtArray, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew)
C
      include 'vaba_param.inc'
      parameter (i_info_AnnealFlag = 1,
     *     i_info_Intpt    = 2, ! Integration station number
     *     i_info_layer  = 3, ! Layer number
     *     i_info_kspt   = 4, ! Section point number in current layer
     *     i_info_effModDefn = 5, ! =1 if Bulk/ShearMod need to be defined
     *     i_info_ElemNumStartLoc   = 6) ! Start loc of user element number
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), dtArray(2*(nblock)+1), strainInc(nblock,ndir+nshr
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock), jInfoArray(*)
C
      character*80 cmname
C
      pointer (ptrjElemNum, jElemNum)
      dimension jElemNum(nblock)
      REAL*8 E, Nu, EG2, ELAM, EG
      zero = 0.d0
      one = 1.d0
      two = 2.d0
      third = 1.d0 / 3.d0
      half = 0.5d0
C
      lAnneal = jInfoArray(i_info_AnnealFlag)
      iLayer = jInfoArray(i_info_layer)
      kspt   = jInfoArray(i_info_kspt)
      intPt  = jInfoArray(i_info_Intpt)
      iUpdateEffMod = jInfoArray(i_info_effModDefn)
      iElemNumStartLoc = jInfoArray(i_info_ElemNumStartLoc)
      ptrjElemNum = loc(jInfoArray(iElemNumStartLoc))

C
```

In the next step, we define the physical constants for determining the stiffness matrix (presented in the next figure).

$$\begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}$$

Notice that, as shown below, we stored the values of the Young's modulus and Poisson's ratio as mechanical constants. Whenever we need these values, we can call them using props(1) and props(2), respectively.



Name: STEEL
Description:
Material Behaviors
User Material

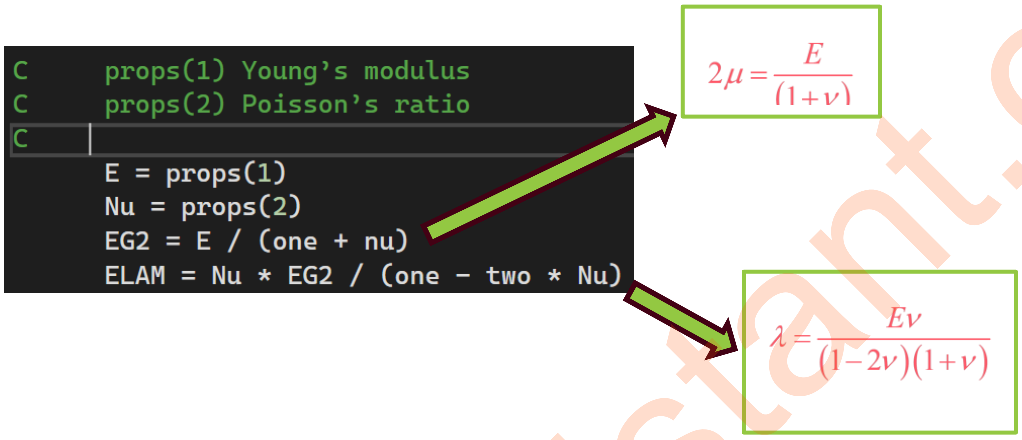
User Material
User material type: Mechanical
Hybrid formulation: [✓ Use default] Specify: Incremental
☐ Use unsymmetric material stiffness matrix
☐ VUMAT defines effective modulus
Data

	Mechanical Constants
1	210000
2	0.3

- *In the next figure, E is the Young's modulus, and NU is the Poisson's ratio.*

- *EG2 is the shear modulus multiplied by 2.*

- *ELAM is the Lamé constant.*

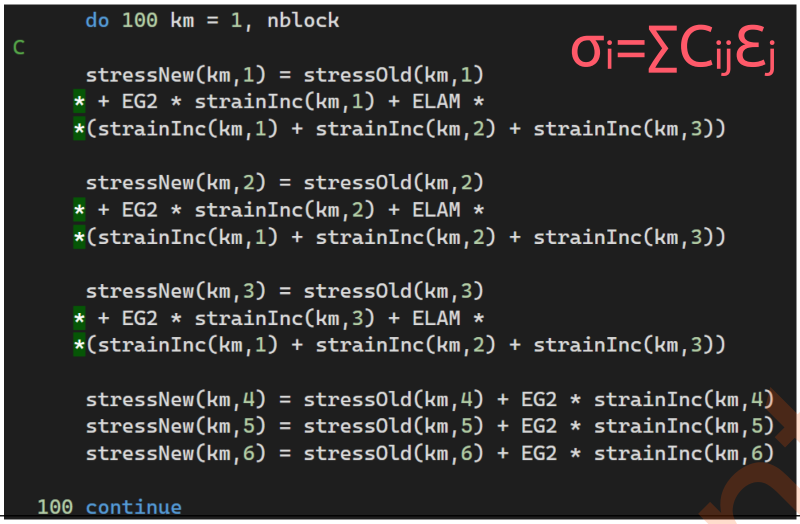


```
C      props(1) Young's modulus
C      props(2) Poisson's ratio
C
       E = props(1)
       Nu = props(2)
       EG2 = E / (one + nu)
       ELAM = Nu * EG2 / (one - two * Nu)
```

$$2\mu = \frac{E}{(1+\nu)}$$

$$\lambda = \frac{E\nu}{(1-2\nu)(1+\nu)}$$

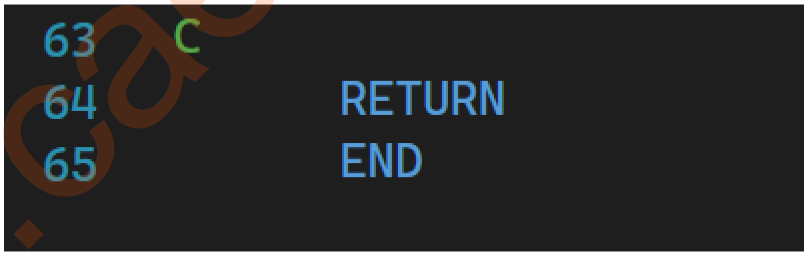
Now, it's time to calculate the stress. Using the formula below, we calculated the stress components within a for loop.



```
      do 100 km = 1, nblock
C
      stressNew(km,1) = stressOld(km,1)
     * + EG2 * strainInc(km,1) + ELAM *
     *(strainInc(km,1) + strainInc(km,2) + strainInc(km,3))

      stressNew(km,2) = stressOld(km,2)
     * + EG2 * strainInc(km,2) + ELAM *
     *(strainInc(km,1) + strainInc(km,2) + strainInc(km,3))

      stressNew(km,3) = stressOld(km,3)
     * + EG2 * strainInc(km,3) + ELAM *
     *(strainInc(km,1) + strainInc(km,2) + strainInc(km,3))

      stressNew(km,4) = stressOld(km,4) + EG2 * strainInc(km,4)
      stressNew(km,5) = stressOld(km,5) + EG2 * strainInc(km,5)
      stressNew(km,6) = stressOld(km,6) + EG2 * strainInc(km,6)

  100 continue
```

- $$\Delta\sigma = c\Delta\varepsilon^e = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix} \begin{pmatrix} \Delta\varepsilon_{11} \\ \Delta\varepsilon_{22} \\ \Delta\varepsilon_{33} \\ 2\Delta\varepsilon_{12} \\ 2\Delta\varepsilon_{23} \\ 2\Delta\varepsilon_{31} \end{pmatrix}$$

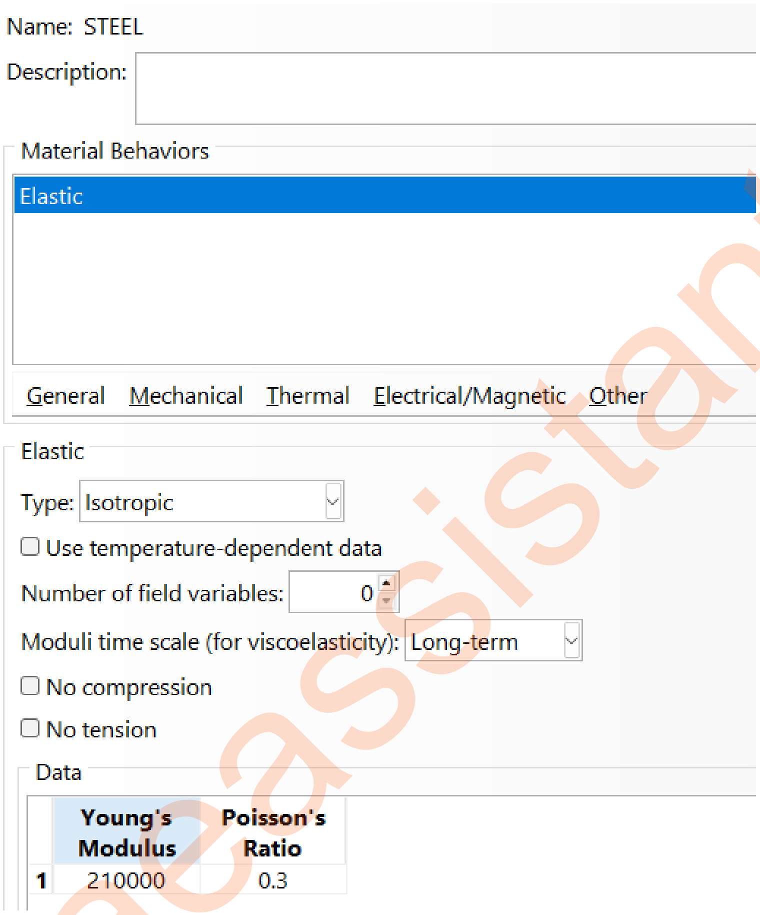
Finally, as shown in the figure below, the subroutine concludes and returns the values to Abaqus.



```
C
      RETURN
      END
```

RESULTS

In this section, we solved the problem once using the VUMAT subroutine and once using Abaqus's built-in material model, with the same properties. The process for defining the materials in Abaqus is illustrated in the figure below.


Name: STEEL

Description:

Material Behaviors

Elastic

Elastic

Type: Isotropic

☐ Use temperature-dependent data

Number of field variables: 0

Moduli time scale (for viscoelasticity): Long-term

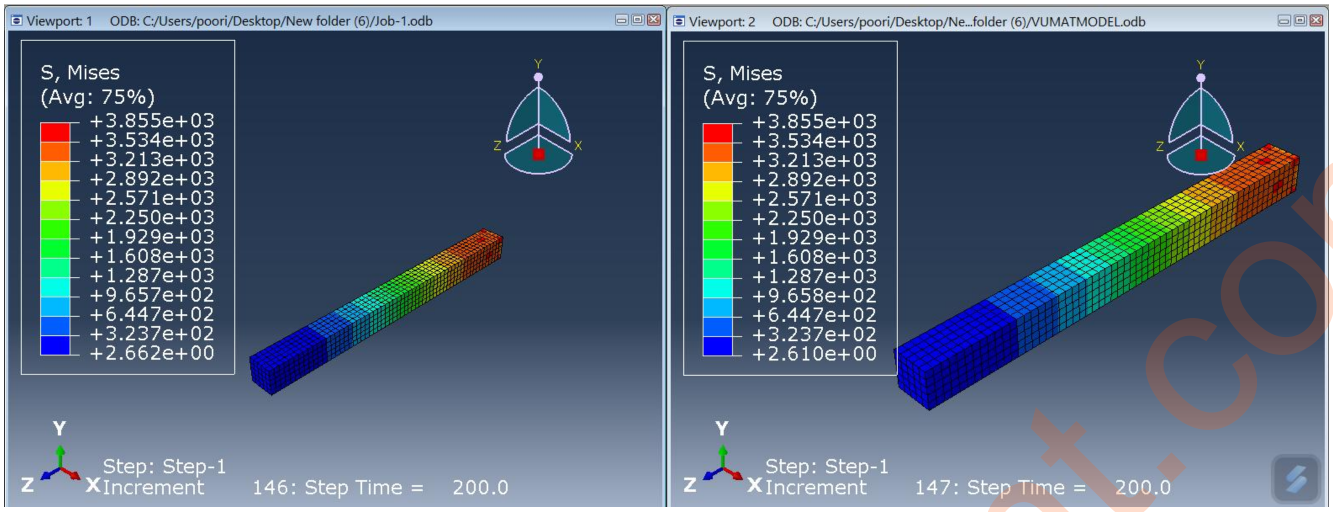
☐ No compression

☐ No tension

Data

	Young's Modulus	Poisson's Ratio
1	210000	0.3

We plotted the Mises stress in the job output. As shown in the figure below, the pattern obtained using Abaqus's material model and the VUMAT subroutine is identical, verifying the accuracy of the VUMAT code.



We have provided the Abaqus modeling files and Fortran code for this example at this GitHub link**.**