**Progressive Release eXplicit VCCT**

**(user guide)**

Table of Contents

[1. Introduction 3](#_Toc117760234)

[1.1. Algorithm 3](#_Toc117760235)

[1.2. Element description 3](#_Toc117760236)

[1.3. Initial delamination 3](#_Toc117760237)

[2. System snapshot and installation 4](#_Toc117760238)

[3. Input definition 4](#_Toc117760239)

[3.1. Abaqus input file 4](#_Toc117760240)

[3.1.1. Defining a PRX-VCCT mesh 4](#_Toc117760241)

[3.1.2. Assigning an PRX-VCCT element 5](#_Toc117760242)

[3.1.3. Assigning material properties 5](#_Toc117760243)

[3.2. Python input file 8](#_Toc117760244)

[3.3. Hardcoded variables – NVCCT.py: 9](#_Toc117760245)

[4. Running a model 9](#_Toc117760246)

[5. Limitations 10](#_Toc117760247)

[5.1. Mesh 10](#_Toc117760248)

[5.2. Initial Crack 10](#_Toc117760249)

[5.3. Geometry 10](#_Toc117760250)

[Appendix A – List of files 11](#_Toc117760251)

[References 11](#_Toc117760252)

[6. Revision list 12](#_Toc117760253)

# Introduction

This guide accompanies the PRX-VCCT software and intends to provide a basic overview that can enable the user to setup and run the code. For questions/issues/clarification add a discussion/issue to the git repository: <https://github.com/UCL-UofU/PRX-VCCT>. The PRX-VCCT formulation follows generally what described in [1-3]. Reference [1-3] as appropriate as well as <https://github.com/UCL-UofU/PRX-VCCT> when publishing results obtained/derived from using this code.

## Algorithm

Figure 1 provides an overview of the algorithm, highlighting the main components. As the job is submitted, a set of python scripts is called which load and store the mesh information, and assign any pre-existent cracks. This information is passed to Abaqus via “*UEXTERNAL\_DB.f*” to the *UEL.f* as required. Both *UEXTERNAL\_DB.f* and *UEL.f* are given in the “*UEL3D\_INTERFACE\_NVCCT.f*” . These files contain all abaqus user subroutines required to run the code. As the simulation progresses, at the end of each increment, the python scripts are called to update nodal stiffnesses as dictated by the PRX\_VCCT algorithm (both fatigue and static) and this new information is again passed to Abaqus via *UEXTERNAL\_DB.f*” to the “*UEL.f”.*

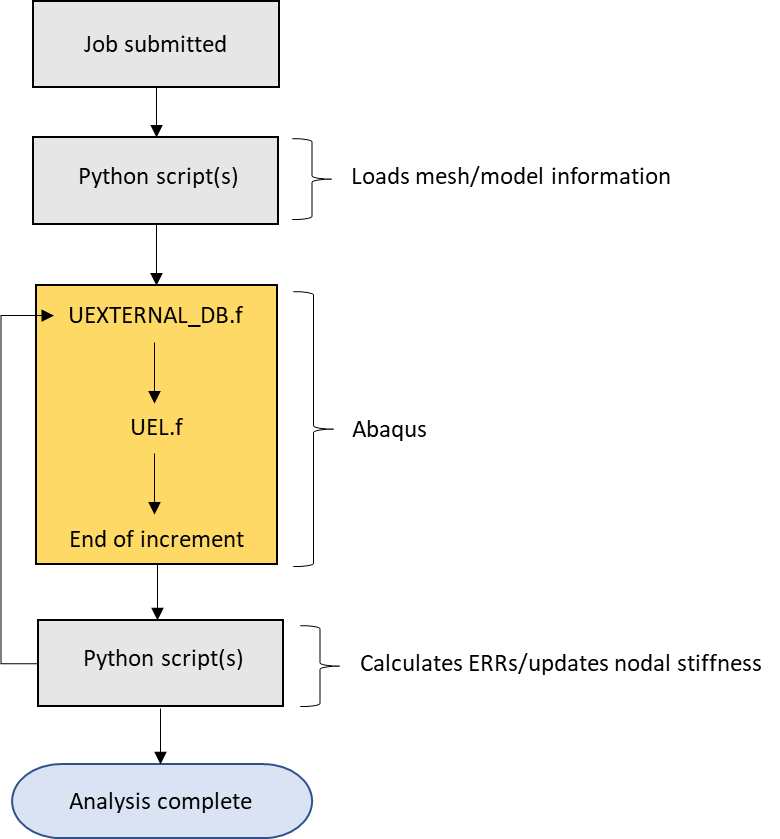


Figure 1 Algorithm overview.

## Element description

The VCCT algorithm interfaces with Abaqus via an interface element (8 nodes). The nodal connectivity of the interface element should be defined consistently with the definition of interface elements in Abaqus (COH3D8 – see Abaqus manual for additional details).

## Initial delamination

An initial delamination can be specified via an external file ‘node\_map\_info.ndmg’. In this file the failed nodes that will form the initial delamination should be listed (one per line).

Alternatively, an initial delamination can be defined via the definition of an element set and assignment via \*uel of damage state “DSTAT” equal to “1”. This will result in setting all damage state variables of the nodes to DSTAT = 1. The delamination specified via element set result in a maximum dimension that is one node ligament larger than the area defined by the element set, see Figure 2.

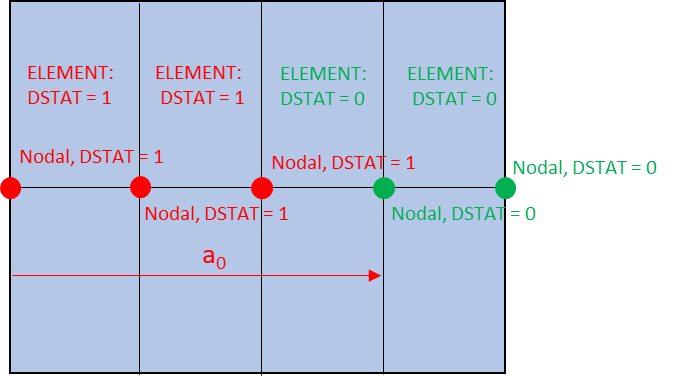


Figure 2 Initial delamination definition via element set and \*uel assignement.

# System snapshot and installation

The code has been developed and verified in the following environment:

* Operating System -  Open-SuSE 64bit  Linux (version 42.1)
* tcsh
* Abaqus 6.25-1
* Intel Fortran version 2021.4
* Python 3.13.9
* Numpy:
  + python-numpy-2.3.3

Listing 1 Current environment in use to run the extended interface finite element code.

Departures from this environment need to be tested. Due to lack of resources to support different versions/environments the **users are advised to try to match the system above as closely as possible** when setting up and running the code. In addition, the users should ensure that the Abaqus installation used is set up to run with user subroutines.

**To install the software**, besides guaranteeing a **system environment** like the described above, the users should copy **the files in “*CODE\_MASTER*” to a local directory.** **This directory, subsequently named “CODE DIRECTORY” should be different than the working directory**.

# Input definition

The code expects **two input files**: an **abaqus input file** with termination \*.inp and a **python input file** with termination .py. Both files need to be named the same (with exception of the file termination), e.g. *input\_file\_name.inp* and *input\_file\_name.py*.

## Abaqus input file

The Abaqus *\**.inp should be setup according to Abaqus syntax. The keywords and structure specific to the PRX\_VCCT code are highlighted next.

### Defining a PRX-VCCT mesh

The interface(s) specified via PRX-VCCT are expected to be defined within a single part. Currently the code does not support PRX-VCCT definitions across multiple parts. The name of the part in which a PRX-VCCT is defined should start with “\_VCCT”, example given bellow.

\*Part, name=\_VCCT\_example

Listing 2 PRX\_VCCT part definition.

If the user does not want to define a part the following syntax can be used, where an additional ‘\*’ is added to the beginning of the line. This will “comment” the line for Abaqus, but still signal the python script to read pertinent information to be used in the PRX-VCCT algorithm. It is important to highlight that despite the **absence of a part specification via the Abaqus keyword, the** **user should include the syntax provided in the listing below.**

\*\*Part, name=\_VCCT\_example

Listing 3 PRX\_VCCT virtual part definition.

### Assigning an PRX-VCCT element

The PRX-VCCT element definition should follow the syntax given in the listing below via Abaqus’ keyword \*user element. As mentioned, each element consists of 8 nodes. The connectivities of the nodes should be established following the convention used to specify Abaqus native cohesive elements. The listings below show the definition of the \*user element card.

\*\*

\*USER ELEMENT, TYPE=U308, NODES=8,COORDINATES=3,PROPERTIES=20,VARIABLES=22

1,2,3

\*Element, type=U308, elset=PRX\_VCCT\_interfaces

element\_number, node 1, …, node 8

.

.

.

Listing 4 User element definition

### Assigning material properties

The material property assignment is performed via the Abaqus command \*uel property, as exemplified in the listing below.

\*UEL PROPERTY, ELSET=elset\_name

dstat, ORI\_IT, ORI\_IB, fg\_x, fg\_y, fg\_z, Y\_N\_I, s\_n\_I,

Y\_S\_I, s\_s\_I, GIC\_IB, GIIC\_IB, eta\_IB, GIC\_IT, GIIC\_IT, eta\_IT,

dadn\_IB, r\_curve\_IB, dadn\_IT, r\_curve\_IT

Listing 5 Assigning properties to the user elements. Note that abaqus only reads 8 properties per line, so the positions (comma separate) of the variables given above in-line as well as between lines should be preserved.

The value “*elset\_name*” should be replaced by the name of the element set to which the properties are going to be assigned. **The code expects this element set to be defined explicitly as a list**. **The elset\_name should start with ‘VCCT’**. Additionally, it cannot be defined using Abaqus “generate” commands or implicitly during the element generation. Not all material properties are required to perform an analysis, however dummy properties need to be specified for completion. For example, fatigue growth rates, will not be used during a quasi-static simulation, however dummy fatigue growth rates are required to be specified for completion of the input deck. The description of the input properties input via the command \*uel property are given in Table 1.

The labels in curved brackets “*IT*” and “*IB*” indicate that variable is assigned to the interface fracture process when the top ply is activated and when the bottom ply is activated, respectively.

Table 1 Description of the input properties to be provided via \*uel command common to both fatigue and quasi-static versions.

|  |  |
| --- | --- |
| Variable | Variable description |
| dstat | State variable of associated with the element set; can be either 1 or 0 |
| ori\_{IT/IB} | orientation of the plies above (IT) and below (IB) in degrees. This information can be used to incorporate directional fracture. In this case variation of critical ERR with relative orientation to the plies (or other substract) above (IT) and below (IB) the interface |
| (fz,fy,fz) | versor of reference direction in global coordinates (verified only for 1,0,0). used to define orientation of plies in a reference coordinate system |
| Y\_N\_{I} | normal strength of interface |
| s\_n\_{I} | coefficient of the S-N curve associated with strength normal to the interface – reserved (not active) |
| Y\_S\_{I} | shear strength of interface (cohesive formulation) of sub-element 1/sub-element 2 |
| s\_s\_{I} | coefficient of the S-N curve associated with the shear strength of the interface - reserved (not active) |
| GIC\_{IT/IB} | Mode I fracture toughness of interface if top ply activated (IT)/or if bottom ply activated (IB); prescribe same value if interface is not directionally dependent, i.e. not dependent on whether top/bottom ply is activated. |
| GIIC\_{IT/IB} | Mode II fracture toughness of interface if top ply activated (IT) and bottom ply activated (IB); prescribe same value if interface is not directionally dependent, i.e. not dependent on whether top/bottom ply is activated. |
| eta\_{IT/IB} | BK criterion exponent of interface if top ply activated (IT)/or if bottom ply activated (IB); prescribe same value if interface is not directionally dependent, i.e. not dependent on whether top/bottom ply is activated. |
| dadn\_{IT/IB} | Number of the table specifying the mixed-mode Paris Law for the interface when the top (IT) or bottom (IB) plies are activated |
| r\_curve\_{IT/IB} | Number of the table specifying the mixed-mode r-curve for the interface when the top (IT) or bottom (IB) plies are activated |

For reference the BK fracture criterion assumes fracture occurs when:

|  |  |
| --- | --- |
|  | Eq. 1 |

with given by:

|  |  |
| --- | --- |
|  | Eq. 2 |

In the present implementation , with .

Damage onset via a stress criterion is enabled in the current version. This feature has not been fully assessed and should be exercised with care. The goal is to provide PRX-VCCT with a convenience feature for cases where multiple damage instances are expected to occur at weak singularities throughout the simulation, and in which the expected response (or desired solution) is fracture driven rather than strength driven. The onset criterion implemented is currently of the form:

|  |  |
| --- | --- |
|  | Eq. 3 |

Enabling S-N curves for onset damage in fatigue is still under development. To provide context to the input variables s\_n\_{I} and s\_s\_{I} reserved, the form of the assumed S-N curves is provided below:

|  |  |
| --- | --- |
|  | Eq. 4 |

where is the static strength, coefficient determined experimentally and the number of cycles.

In addition to the \*uel property definition, two other tables should be prescribed defining the mixed mode Paris Law and R-Curve behavior. **These are PRX-VCCT tables** that are only pre-processed by the PRX-VCCT code and not by ABAQUS. Hence, **'\*\*' are required for before each table line**, such that the table is ignored by the Abaqus’ pre-processor. **To associate a Paris Law** or R-Curve given in tabular form **with the material definition**, **the parameters dadn\_{IT/IB} and r\_curve\_{IT/IB}** **should equal corresponding Paris Law table number "dadn\_no" or R-curve table number "r\_curve\_no".** The mixed-mode Paris Law can be defined tabular form as follows:

\*\*VCCT\_DADN, DADN=dadn\_no

\*\*C\_1, beta\_1, mu\_1, gamma\_1, rho\_1, mm\_1

\*\*C\_2, beta\_2, mu\_2, gamma\_2, rho\_2, mm\_2

...

\*\*C\_n, beta\_n, mu\_n, gamma\_n, rho\_n, mm\_n

Listing Mixed-Mode Paris Law definition using a tabular form.

This table assumes a Paris-law equation of the type:

|  |  |
| --- | --- |
|  | Eq. 5 |

The table below provides correspondence between the parameters in the Listing 6 and the Paris Law variables in Eq. 5.

Table 2 Correspondence between the variables given in the VCCT\_DADN table and the Paris Law definition.

|  |  |
| --- | --- |
| Variable | Variable description |
| C\_{i} | in Eq. 5 |
| beta\_{i} | in Eq. 5 |
| mu\_{i} | in Eq. 5 |
| gamma\_{i} | in Eq. 5 |
| rho\_{i} | in Eq. 5 |
| mm\_{i} | mode-mixity associated with each line {i} |

The parameters r\_curve\_{IT/IB} are integer numbers corresponding to R-Curve definitions. The R-curve is defined in tabular form as a function of mode-mixity as follows:

\*\*VCCT\_R\_CURVE, R\_CURVE = r\_curve\_no

\*\*f\_1, a\_1, mm\_1

\*\*f\_2, a\_2, mm\_2

...

\*\*f\_n, a\_n, mm\_n

Listing Mixed-Mode Paris Law definition using a tabular form.

where the relationship:

|  |  |
| --- | --- |
|  | Eq. 6 |

is assumed. The table below provides a correspondence between the parameters in the Listing 7 and the R\_curve variables.

Table 3 Correspondence between the variables given in the r\_curve table and the R\_curve definition.

|  |  |
| --- | --- |
| Variable | Variable description |
| f\_{i} | in Eq. 6 |
| a\_{i} | in Eq. 6 |
| mm\_{i} | mode-mixity associated with each line {i} |

## Python input file

The python input file provides a vehicle to pass additional information to the python script(s). The variables accepted are given below (with default values prescribed in case they are omitted):

PR = True

max\_normalized\_inc = 0.2

cut\_back\_factor = 0.1

static\_scale\_f = 10.0

stress\_crit\_tol = 0.05

unst\_gth\_tol = 0.05

nlgeom = False

R\_ratio = 0.1

contact = True

k\_c\_user = 1.0

steps\_cycles = None

no\_fail\_zone\_char\_length = None

no\_of\_buckets = 4

Listing Variables input via the python file.

A description of the variables is provided in the table below:

Table 4 Description of input variables available via the python input file

|  |  |
| --- | --- |
| Variable | Variable description |
| PR | enable progressive release algorithm vs. instantaneous release |
| max\_normalized\_inc | controls the maximum fraction of a ligament between node pairs that a delamination can grow in each increment. Coarser meshes may require decreasing “max\_normalized\_inc” |
| cut\_back\_factor | Factor by which the next crack increment will be decreased. For example time\_increment\_new = time\_increment\_current \* cut\_back\_factor |
| static\_scale\_f | Static growth rate scale factor. The larger the value the better the discrete fracture criterion will be approximated at the expense of more solution increments and a less smooth crack front. |
| stress\_crit\_tol | Knockdown to the such that failure occurs if that > 0.95 |
| unst\_gth\_tol | Fraction of the fracture criterion above which a cut back will be triggered. For example, if f\_cr > 1.0 + unst\_gth\_tol, a cut back will be triggered in the next increment. |
| nlgeom | controls whether a nonlinear geometric formulation is used. If 'nlgeom = True' the abaqus step definition should also be set to use nonlinear geometry formulation. |
| R\_ratio | sets the ratio between the minimum load and the maximum load applied via the step definition (Fatigue). |
| contact | selects whether contact should be enforced across a failed interface. |
| K\_c\_user | factor multiplying the default penalty stiffness used to enforce contact across failed crack (1 is the default value) |
| steps\_cycles | cycles to be performed in each analysis step, provided as a list steps\_cycles = [0,10000,…]. If not specified all steps assumed to be quasi-static. The number of steps in the list needs to correspond to the number of steps specified in Abaqus. |
| no\_fail\_zone\_char\_length | length of a square region around the crack tip, within which onset will not be activated |
| no\_of\_buckets | no of buckets that will be used to search for no-fail regions (based on “no\_fail\_zone\_char\_length”). For large problems this number may need to be increased. |

# Running a model

Once the FE model with required material properties, boundary conditions and applied traction or displacement is ready for a given specimen/coupon/panel, the following shell script is executed to run the model, linking abaqus and the python scripts. An example shell script is provided with the code “*input\_stat.sh*” and its contents are detailed below. The shell script reads:

#!/bin/tcsh

*rm* -f \*.txt

*rm* -f \*.npy

*rm* -f \*.npyc

*cp* /*source code directory*/\*.f /working directory

*cp* /*source code directory*/\*.py /working directory

*cp* /*source code directory*/\*.sh /working directory

*python* dir\_name.py

*chmod* 777 \*

*abaqus*  j=*jobname* input=*filename* double user=UEL3D\_DIR\_CZ.f cpus=no\_cpus

Listing 10 Example of shell script used to link and run the code – provided as “*input\_stat.sh*” together with the example input files.

These are the set commands on Linux/Unix OS. For running it on w*indows* environment or with other shell (that not tcsh), the script needs to be modified accordingly. Additionally, user will need to, before each new run, guarantee the “*source code directory*” and the “*working directory*” are adequately provided, as well as the Abaqus call (last line), including input file name, “*filename*” and number of cpus “*no\_cpus*”. The maximum number of ‘cpus‘ depends upon the availability of number of Abaqus tokens.

To execute the shell script type in file *‘input\_stat.sh’* in the command prompt.

# Visualization

Abaqus does not allow the visualization of user elements. This can be circumvented by specifying dummy or ghost elements that are superposed to the user elements, and transferring pertinent information, such as damage state etc, via a common block. The basic building blocks of this approach are in-place but the user will need to establish the links.

Meanwhile, energy release rate ERR output, actual and virtual crack positions, etc, can be viewed via GT\_i.txt files. These files are produced at the end of each increment “i”. Their production can be suppressed by setting “verbose\_vcct\_el\_inc = False” in NVCCT.py line: 5098. Currently they will produce GT\_i.txt files with the following columns:

Table 5 Columns in the GT\_i.txt files

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vcct node # |  |  | *d* |  |  |  |  | next node # |  |  |  |

Note that , , and have three columns each. Additional details below:

Table 6 Definition of the columns in the GT\_i.txt files

|  |  |
| --- | --- |
| vcct node # | number of the node being evaluated |
|  | Node position - three columns |
|  | Crack virtual position – three columns |
| *d* | Damage variable |
|  | Energy Release Rate – shear direction 1 |
|  | Energy Release Rate – shear direction 2 |
|  | Energy Release Rate – Mode I |
|  | fracture criterion |
| next node # | number of the node towards which the crack is growing |
|  | crack front vector 1 – one side |
|  | crack front vector 2 – other side |
|  | accumulated crack growth |

# Limitations

The known limitations below reflect implementation choices rather than fundamental limitations of the approach and can be revisited in future versions. The approach may have other limitations which are at present unknown or not documented.

## Mesh

One of the limitations of the present implementation is that the element ligaments along orthogonal (or quasi-orthogonal) mesh lines should not vary. In practice this means that transitioning from coarse to fine meshes is not accommodated. Indeed, results from meshes that departure from regular, orthogonal meshes should be scrutinized.

# Appendix A – List of files

The following files comprise of **user subroutines and auxiliary shell scripts and python scripts that constitute the PRX-VCCT code:**

* **dir\_name.py**
  + updates directory names as needed
* **Initialize\_cze2vcct.py**
  + Functions used to gather mesh information and establish the link between the 8 noded cohesive-like element used to interface with abaqus and the PRX-VCCT non-local nodal implementation.
* **NVCCT.py**
  + Updates the elements as required at the end of each converged increment
* **Update\_elements\_nvcct.sh**
  + Script that calls update\_elements\_cz.py, updating the environment variables such that the python libraries required by update\_elements\_cz are loaded
* **UEL3D\_INTERFACE\_NVCCT.f**
  + User element defined subroutine. Contains the subroutines needed to interface with abaqus including UEL.f, and UEXTERNAL\_DB.f

# References

[1] N. V. De Carvalho, G. E. Mabson, R. Krueger and L. R. Deobald. A new approach to model delamination growth in fatigue using the Virtual Crack Closure Technique without re-meshing. *Engineering Fracture Mechanics*. 222, 106614, 2019.

[2] De Carvalho NV, Ramnath M, Mabson GE, Krueger R. An explicit delamination propagation algorithm to simulate delamination growth under quasi-static and fatigue loading without re-meshing using virtual crack closure technique and progressive nodal release. *Journal of Composite Materials*. 2022;56(13):2063-2081.

[3] R. Krueger, N. V. De Carvalho. Development of a C-ELS Specimen-Based Numerical Benchmark for Mode II Delamination and Assessment of Two VCCT-based propagation strategies. In: 37th American Society for Composites Technical Conference, Tucson, AZ, USA, 2022.

[4] Andrew Westerhoff, N. V. De Carvalho. PRX-VCCT. GitHub. <https://github.com/UCL-UofU/PRX-VCCT>.