A METHODOLOGY TO DEFINE DIFFUSE INTERFACE IN ABAQUS

A. Quintanas-Corominas, E. Martínez-Pañeda

1 INTRODUCTION

This report proposes a methodology to define diffuse interfaces in Abaqus. The core idea is to split the simulation into two. The first simulation solves a phase-field equation to obtain a scalar field referred inhere as the interface indicator, which defines the transition zone between the interfaces and the bulk. The second simulation, which is aimed at solving the physics of the problem, loads the interface indicator from the former simulation to interpolate the material properties between the bulk and interface regions.

2 THEORETICAL ASPECTS

The generation of the diffuse interface is done by solving the well-known interface phase field model, which in his strong form reads:

$$\begin{cases} d - \ell \nabla^2 d = 0 & \text{in} \quad \Omega \\ d(0) = 1 & \text{on} \quad \Gamma \\ \nabla d \cdot \mathrm{n} = 0 & \text{on} \quad \partial\Omega \end{cases}$$

where η is the interface indicator parameter, which is 0 at the bulk region and 1 at the interface, and ℓ is a scale parameter that controls the width of the transition between both regions, see Figure 1.

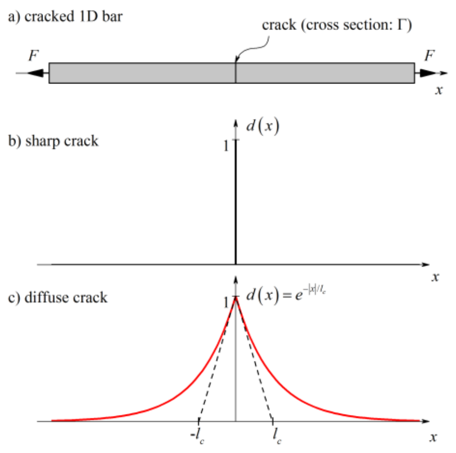
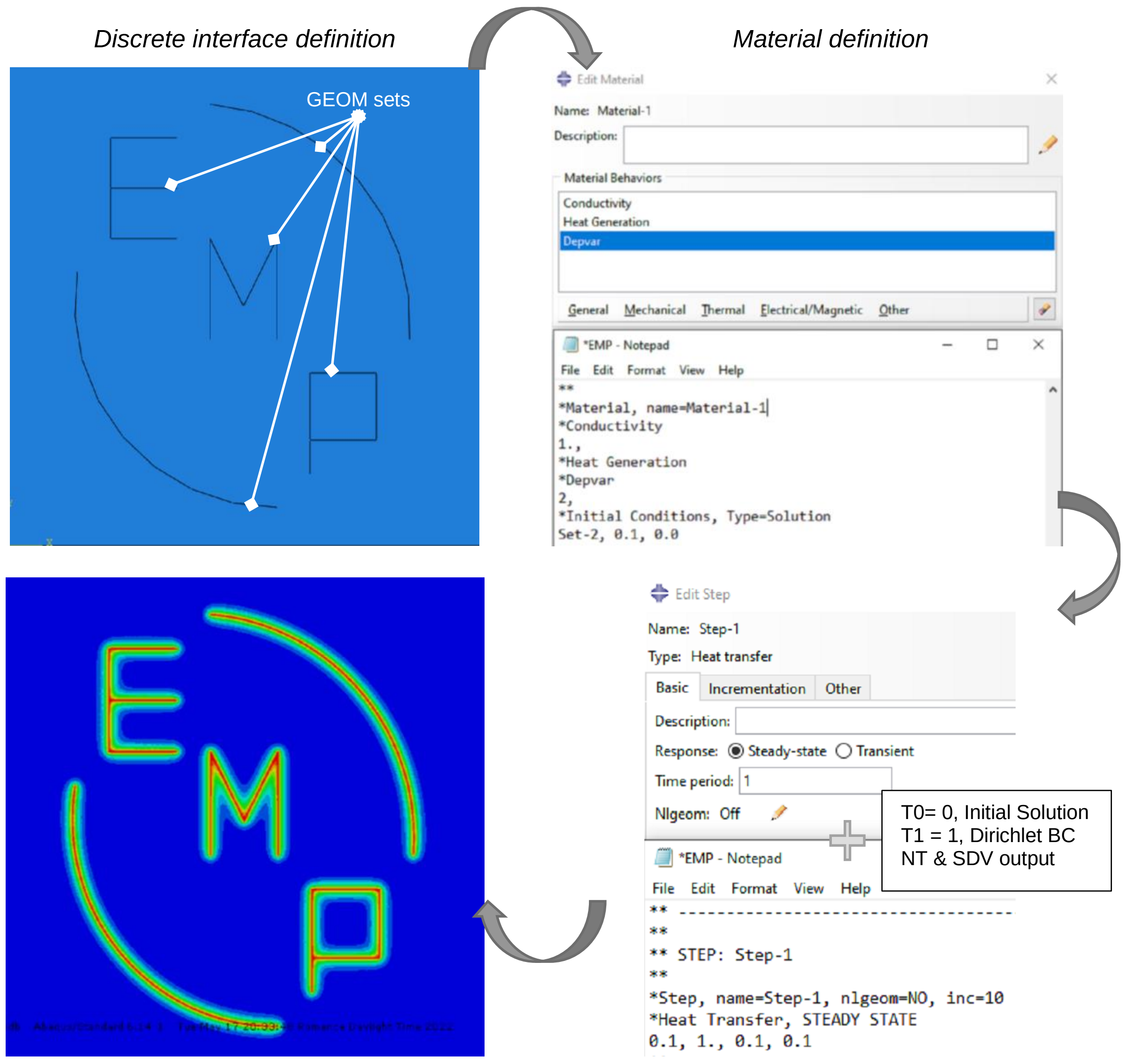


Figure 1: Diffuse crack representation using the phase-field equation [Reproduced from www.molnar-research.com].

3 ABAQUS MODELLING STRATEGY

3.1 DIFFUSE INTERFACE GENERATION SIMULATION

The generation of the diffuse interface in Abaqus is done through a HEAT TRANSFER ANALYSIS (HTA) using a HETVAL user subroutine following the procedure depicted in Figure 2.


Discrete interface definition
GEOM sets
Material definition
Edit Material
General Mechanical Thermal Electrical/Magnetic Other
*EMP - Notepad
File Edit Format View Help
Edit Step
Basic Incrementation Other
T0= 0, Initial Solution
T1 = 1, Dirichlet BC
NT & SDV output
Diffuse interface representation
BC & Step Definition

Name: Material-1
Description:
Material Behaviors
Conductivity
Heat Generation
Depvar

```
**
*Material, name=Material-1
*Conductivity
1.,
*Heat Generation
*Depvar
2,
*Initial Conditions, Type=Solution
Set-2, 0.1, 0.0
```

Name: Step-1
Type: Heat transfer
Description:
Response: [✓ Steady-state] Transient
Time period: 1
Nlgeom: Off

```
** ----------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1, nlgeom=NO, inc=10
*Heat Transfer, STEADY STATE
0.1, 1., 0.1, 0.1
```

Figure 2: Schematic representation of the Diffuse Interface Generation simulation

The steps are:

1) Create a model in the CAE, which must contain the interfaces as NODAL or GEOMETRICAL sets. In this example, the sets are defined in the Part, but they can also be defined in the Assembly.
2) Create a material model with the following characteristics and assign it to the whole model:
    - Conductivity = 1
    - Heat Generation
    - Depvar = 2
3) Mesh the model by setting the element type corresponding to an HTA, e.g. DC2D4 for quads.
4) Create the *ASSEMBLY*.
5) Create a new step setting, a *HEAT TRANSFER ANALYSIS* with a *STEADY STATE RESPONSE* with a total simulation/physical time of 1s.
6) Set the boundary conditions of the problem by setting that at t=1s, the temperature is equal to 1 on the node sets defined in (1).
7) Create a PREDEFINED FIELD which must set the INITIAL TEMPERATURE of 0 in the whole model.
8) Ask the temperature (NT) and state variables (SVD) as nodal FIELD OUTPUT REQUEST.
9) Modify the input file (inside or outside the CAE) to add:

```
*Initial Conditions, Type=Solution
<a>, <b>, 0.0
```

   where <a> is the name of a nodal set defining all the nodes of the model and <b> is a real number defining the regularization length scale.

10) Run the simulation.

3.2 PHYSICAL SIMULATION

The methodology for reading the diffuse interface consists of setting an initial condition of the temperature that is read from an Abaqus output database (ODB). This strategy is done by setting in the input file the following keywords before the first step section:

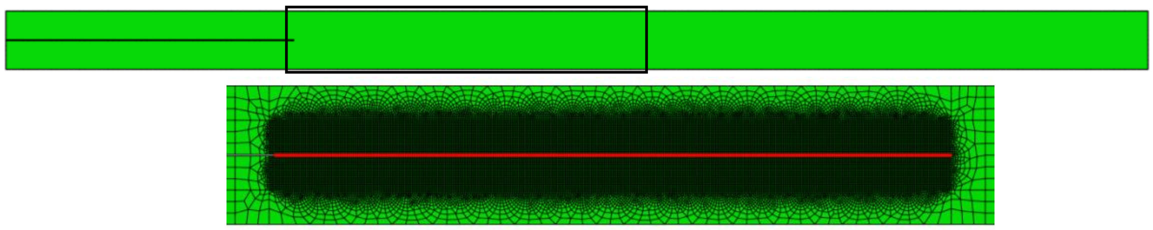
```
*Initial Conditions, Type=field, file=<a>, output variable=NT,
```

where <a> is the name of the ODB resulting from the Diffuse interface generation simulation. Thanks to this keyword in the input file, the interface topology previously diffused can be recovered through the variables related to the PREDEFINED FIELDS. Note that the usage of the diffuse interface inside the user subroutines depends on the modelling strategy employed.

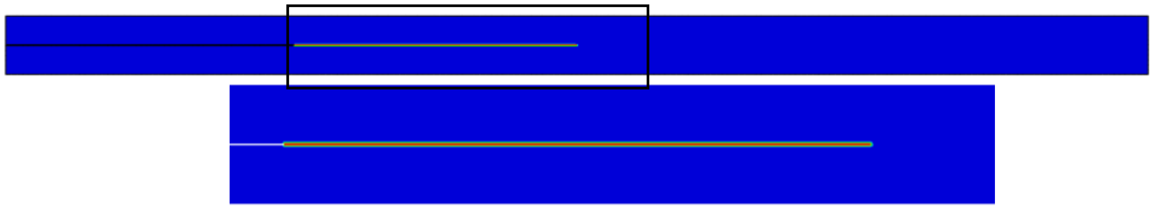
4 REPRESENTATIVE EXAMPLE: UMAT/HETVAL

This example illustrates how the diffuse interface methodology can be employed with the UMAT/HEATVAL phase-field fracture presented by Navidtehrani et al.[1][2]. For this purpose, a Double Cantilever Beam (DCB) specimen with a single interface between the top and bottom arms is considered.

The pre- and post-process of the Diffuse Interface Generation simulation are shown in Figure 3a and b, respectively. The former illustrates the CAE model of de DCB with the node sets where the interface wants to be generated, while the latter depicts the interface topology indicator after the simulation.



(a)



(b)

Figure 3: Diffuse Interface Generation simulation for Double Cantilever Beam.

In this example, the indicator of the diffuse interface topology is used to interpolate the fracture toughness between the interface and the bulk region. This interpolation must be a monotonous transition between the bulk and the interface regions. For the sake of simplicity, the well-known quadratic form, which is widely used in the PFA problems, is employed for the interface indicator function:

$$g(d) = (1 - d)^2$$

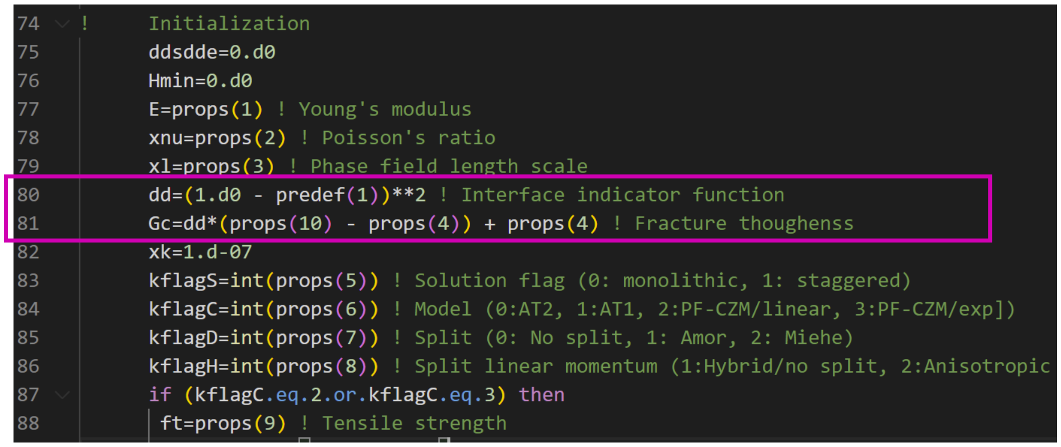
where ϕ is 0 on the bulk region and 1 on the interface. Note that this function satisfies the necessary properties for a smooth transition, i.e. $g(d = 0) = 1$, $g(d = 1) = 1$, and $g'(d) < 0$. Then, the following expression can be employed to interpolate a property G_c between the value of the bulk G_{cB} and the interface G_{cI}:

$$G_c = g(d)(G_{cB} - G_{cI}) + G_{cI}$$

Another option could be to employ a linear interpolation: $G_c = (1 - d)G_{cB} + dG_{cI}$.

[1] Y. Navidtehrani, C. Betegón, E. Martínez-Pañeda. *Materials* 14(8) (2021).
[2] Y. Navidtehrani, C. Betegón, E. Martínez-Pañeda. *Applications in Engineering Science* 6 (2021).

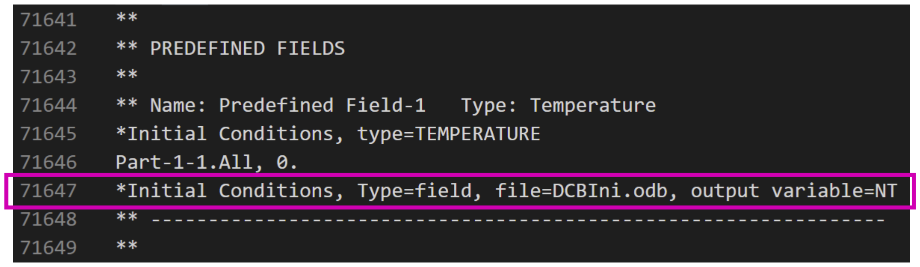
Figure 4 shows two snapshots of the modifications introduced in the UMAT(s) phase-field fracture codes, which can be found at www.empaneda.com. As can be appreciated, two new lines of code have been introduced: (l.80) recovers the interface topology indicator from the initial simulation and (l.81) interpolates the fracture toughness using the quadratic form aforementioned.



```
74  !     Initialization
75        ddsdde=0.d0
76        Hmin=0.d0
77        E=props(1) ! Young's modulus
78        xnu=props(2) ! Poisson's ratio
79        xl=props(3) ! Phase field length scale
80        dd=(1.d0 - predef(1))**2 ! Interface indicator function
81        Gc=dd*(props(10) - props(4)) + props(4) ! Fracture thoughenss
82        xk=1.d-07
83        kflagS=int(props(5)) ! Solution flag (0: monolithic, 1: staggered)
84        kflagC=int(props(6)) ! Model (0:AT2, 1:AT1, 2:PF-CZM/linear, 3:PF-CZM/exp])
85        kflagD=int(props(7)) ! Split (0: No split, 1: Amor, 2: Miehe)
86        kflagH=int(props(8)) ! Split linear momentum (1:Hybrid/no split, 2:Anisotropic
87        if (kflagC.eq.2.or.kflagC.eq.3) then
88         ft=props(9) ! Tensile strength
```

Figure 4: UMAT(s) modifications to interpolate Gc according to the interface indicator.

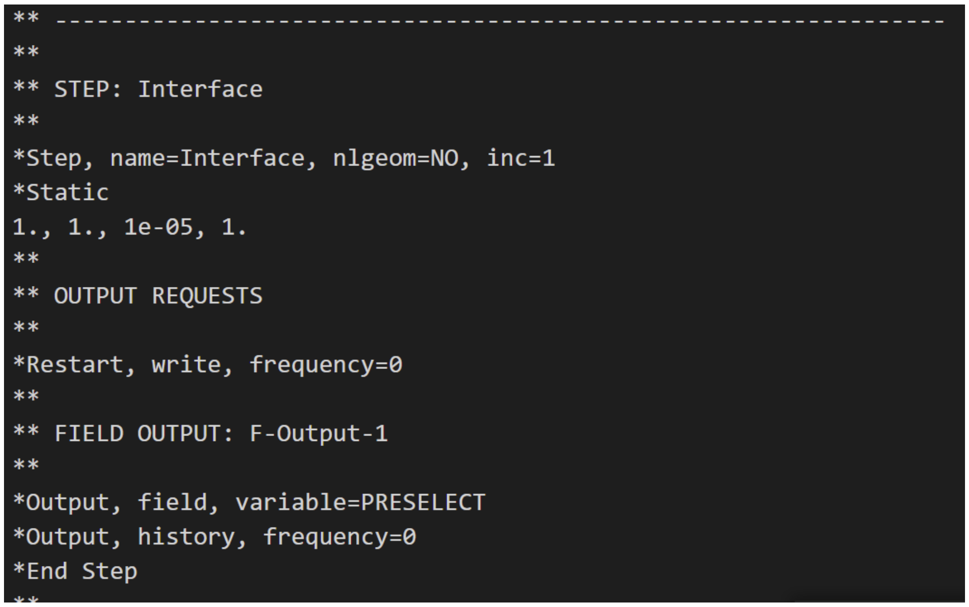
Figure 5 illustrates a snapshot of the INITIAL CONDITION definition in the input file of the Physical simulation. *NOTE: The input file of the Diffuse Interface Generation simulation is called DCBIni.*



```
**
** PREDEFINED FIELDS
**
** Name: Predefined Field-1   Type: Temperature
*Initial Conditions, type=TEMPERATURE
Part-1-1.All, 0.
*Initial Conditions, Type=field, file=DCBIni.odb, output variable=NT
** ----------------------------------------------------------------
**
```

Figure 5: DCB Physical simulation input file.

Although it is not mandatory, an initial step is defined before starting the mechanical loading. This step aims to make the HETVAL subroutine aware of the interpolated Gc before starting the loading to avoid undesired behaviours. In this regard, no boundary conditions are applied. Figure 6 shows the definition of this initial step, while Figure 7 depicts the results at the end of this step. *NOTE: In this example, the fracture toughness of the bulk region is 10 N/mm, and the interface one is 0.2347 N/mm. The SDV4 is the state variable used in the original subroutine to share the Gc between the UMAT and HETVAL.*



```
** ------------------------------------------------------------------
**
** STEP: Interface
**
*Step, name=Interface, nlgeom=NO, inc=1
*Static
1., 1., 1e-05, 1.
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
*Output, history, frequency=0
*End Step
```

Figure 6: DCB Interface step definition.

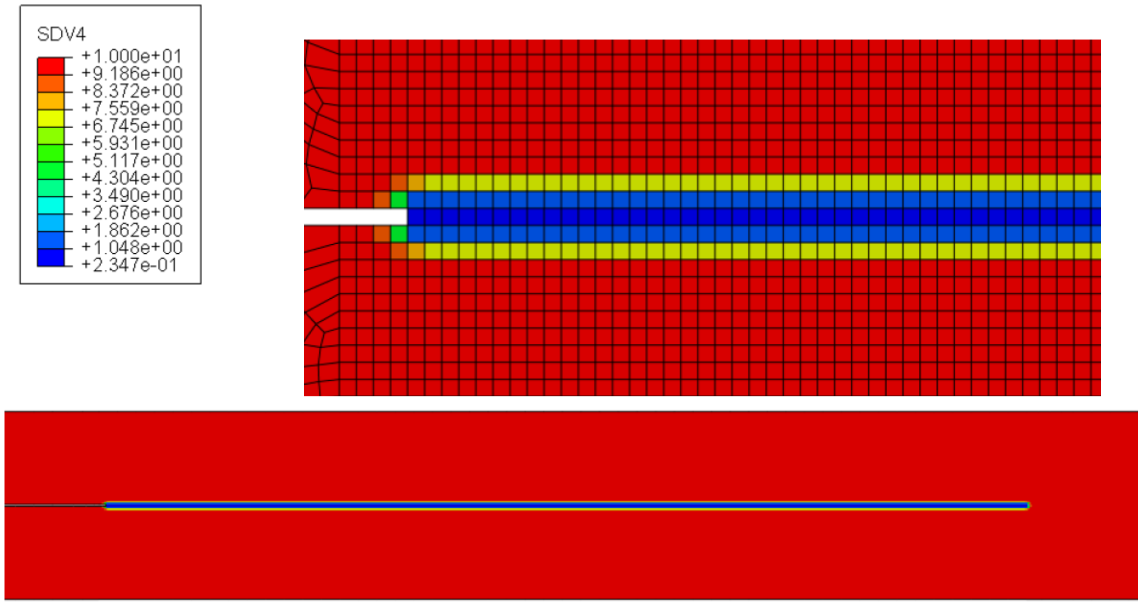


Figure 7: Gc at the elemental level at the end of the initial step.

Figure 8 illustrates the results after the initial step for different length scales. As is expected, the larger values of ℓ induce a larger transition between both regions.

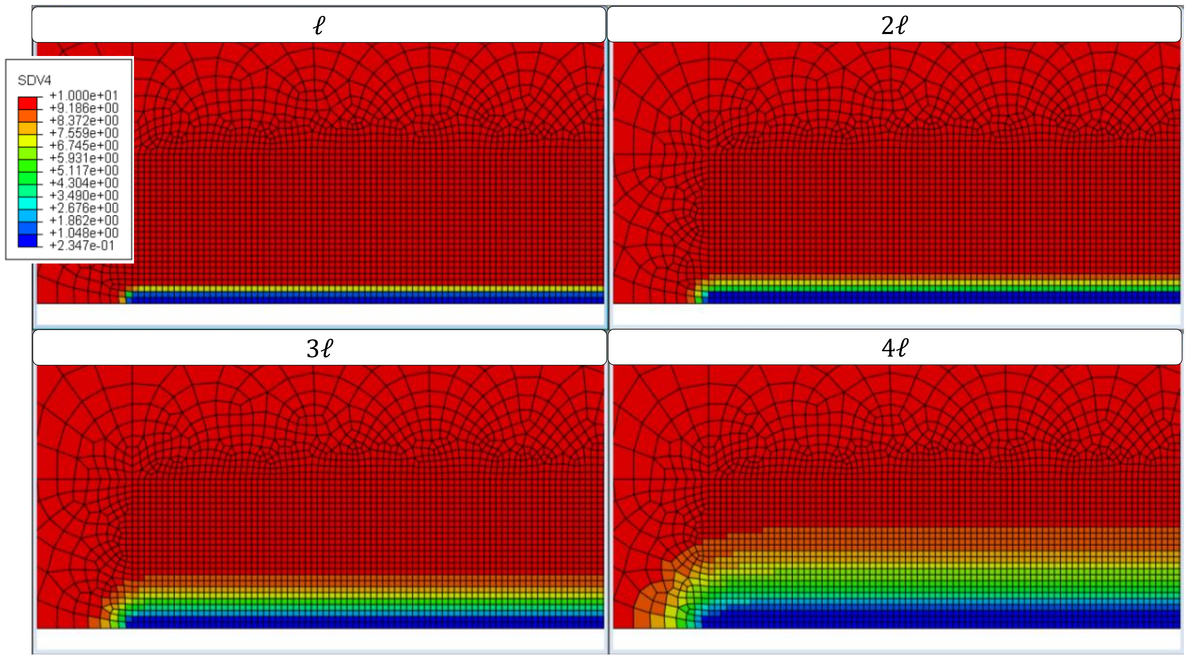


Figure 8: Gc at the elemental level at the end of the initial step for different transition lengths.

5 KNOWN ISSUES

One issue identified hinders this approach's potential and must be considered. Although the simulation to create the interface indicators provides a correct solution (i.e. 0 at the bulk, 1 at the interface, and smooth transition between them), the actual interface vanishes when only one node has the indicator value at 1. It vanishes because, during the physical simulation, no one of the integration points will be inside the interface. This problem can be easily overcome by defining a larger ℓ or defining the elements instead of nodes.

6 FILES

The files to reproduce the results of the representative example can be found with this report, and they are:

1) DCBIni.inp: Abaqus input file for the Diffuse Interface Generation simulation.
2) DCBRun.inp: Abaqus input filer for the Physical simulation (it only contains the initial step).
3) DIFFINTGEN.for: Abaqus user subroutine for the Diffuse Interface Generation simulation.
4) HETVALg.for: Modified version of the Abaqus user subroutine for the Physical simulation.

The sequence and Abaqus commands to execute the codes are:

1) Diffuse Interface Generation simulation: "*abaqus job=DCBIni user=DIFFINTGEN.for cpus=4*"
2) Physical simulation: "*abaqus job=DCBRun user=HETVALg.for cpus=4*"