SANDIA REPORT

SAND2012-8720

Unlimited Release

Printed October 2012

Verification of J -integral capability in Sierra Mechanics

Yuki Ohashi, James W. Foulk III, Alex J. Lindblad

Prepared by

Sandia National Laboratories

Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>

Verification of J -integral capability in Sierra Mechanics

Yuki Ohashi, James W. Foulk III, Alex J. Lindblad

Abstract

A finite-deformation, massively parallel, domain J -integral has been developed in Sierra Mechanics with specific emphasis on pressurized loadings. We review the development of the domain integral and the inclusion of terms to correct for boundary tractions. We note the syntax employed in Sierra and illustrate the choice of weight function. Details are provided regarding domain selection, crack direction, and the construction of the test function. Because quality meshes are required to obtain accurate measures of the driving force, methods were developed in Cubit for creating element boundaries orthogonal to the crack front. We briefly review the new capabilities and provide guidance for mesh construction. A comprehensive set of examples illustrate the current implementation in Sierra. Examples include straight crack fronts, curved crack fronts, and non-planar crack faces. We investigate elastic and elastic-plastic cases. The inclusion of boundary tractions is verified through a pressurized ellipsoidal flaw. When possible, the calculated driving forces are compared to analytical solutions. Although the J -integral is available in many commercial codes, the finite deformation implementation in Sierra will enable analysts to examine detailed component level models and include fracture mechanics as a tool in the Sandia design process.

Acknowledgment

We would like to acknowledge the patience of Mike Chiesa. He continued support for this work through Physics & Engineering Models (P&EM) was instrumental. The Sierra code team was also both patient and supportive. Nathan Crane and Ben Spencer worked on early stories. Although the bulk of the code was implemented by author Alex Lindblad, Mike Veilleux also contributed through excellent ideas and bug fixes in the pressure correction. Martin Heinsteins and Joe Jung have supported this work and we hope to continue this collaboration through future improvements. We would be remiss if we did not also thank our customers. Brian Somerday, Chris San Marchi, Dorian Balch, and Paul Spence have not only been effective advocates, but have also provided continued funding for follow-on efforts. We would also like to acknowledge the important contributions of reviewer Shawn English. A more polished and honest document results from his good work.

Contents

1	Introduction	11
2	Background	13
2.1	Technique for Computing J	14
3	Sierra implementation	19
3.1	Input Commands	19
3.2	Output	21
3.3	Sierra methodology for calculating $\mathbf{q}$	23
3.3.1	Computing Necessary Crack Tip Information During Initialization . . .	23
3.3.2	Computing J During an Output Request	26
	Specification of $\mathbf{q}^I$ for J^K	26
	Special treatment at the A_0^+ and A_0^- boundaries	27
	Specification of a Global $\mathbf{q}$	27
	Computing J	28
4	Required discretization in CUBIT	31
4.1	Ellipsoidal flaw	31
5	Case Studies	35
5.1	Infinite Crack	36
5.2	Penny Flaw	37
5.2.1	Penny Flaw Results: Radial Convergence in J	38
5.2.2	Penny Flaw Results: J versus Load	39

5.3	Cone Crack	41
5.3.1	Cone Crack Results: Bulk Behavior	43
5.3.2	Cone Crack Results: Radial Convergence in J	43
5.3.3	Cone Crack Results: J versus Load	43
5.4	Flat Plate with Embedded Elliptical Flaw	44
5.4.1	Embedded Ellipse Results: Radial Convergence in J	45
5.4.2	Embedded Ellipse Results: J Along Crack Front	46
5.5	C-shaped Compact Tension Specimen	48
5.5.1	C-specimen Results: Bulk Behavior	49
5.5.2	C-specimen Results: Radial Convergence in J	50
5.5.3	C-specimen Results: J Along Crack Front	50
6	Conclusion	57
	References	58

List of Figures

2.1	A domain integral over Ω_0 having boundary Γ_0 that surrounds a thumbnail defect. The crack front tangent, crack front normal, and crack face normal are $\mathbf{S}_{CF}$, $\mathbf{N}_{CF}$, and $\mathbf{T}_{CF}$, respectively.	17
3.1	Example weight functions for a J -integral integration domain. Weight functions shown for domain 5.	21
3.2	How the crack front direction is determined, pictorially.	24
3.3	Scheme to more accurately calculate the driving force at the first interior node. This methodology will mitigate spikes in the driving force associated with an erroneous pressure correction.	28
3.4	The weight function $\mathbf{q}$ corresponding to J_{avg} for all nodes along the crack front. The weight function is automatically calculated to be normal to the crack front.	29
3.5	The magnitude of the weight function $\mathbf{q}^K$ corresponding to J^K for node K along the crack front. Note that node K is the first node interior from the boundary and the weight function is a plateau “tent” function that is 1 along the normal emanating from node K . The weight function is automatically calculated for each node K along the crack front. The entire body is shown to clearly illustrate that $\ \mathbf{q}\ = 0$ for nodes outside the search width sw_K	30
4.1	Screenshots of Cubit that illustrate the new snap_to_normal command. Through two snap_to_normal operations, the mesh is adjusted to be orthogonal to curve 57.	33
5.1	Discretization employed for a crack in an infinite body. A K-field displacement boundary condition is applied to the exterior of the body.	37
5.2	Penny flaw graphic and finite element mesh.	39
5.3	Elements in red are included in the contour over which J is calculated for the penny flaw model: (a) first contour increasing to (e) fifth contour.	39
5.4	Normalized J versus contour for penny flaw model at various loads. Values approaching an asymptote indicate that radial convergence been achieved at this load.	40

5.5	J versus load for pennyflaw model with original mesh refinement using three Sierra weight functions and the analytical solution.	41
5.6	Percent error of J for the 30° pennyflaw model with the original and the refined mesh for three Sierra weight functions. The results indicate that we have attained mesh convergence.	42
5.7	Graphical representation of cone crack problem. Quarter symmetry is used for the finite element model. The conical crack is embedded in a 30 inch cube.	43
5.8	Domain 10 of cone crack model. Elements in red are included in the J -integral calculation.	44
5.9	J versus contour of cone crack specimen at various loads.	45
5.10	J (in-lb/in ²) versus load P calculated for cone crack model.	46
5.11	Embedded elliptical crack in flat plate with pressure P applied on the crack faces.	47
5.12	Mesh of embedded elliptical crack in flat plate.	48
5.13	L_2 error in J versus contour for the embedded ellipse model with applied crack face pressure loading.	49
5.14	J calculated from domain 9 at each node along the crack front for the embedded elliptical flaw problem with applied crack face pressure. Symmetry surface values have unresolved spikes in value. $\phi = 0$ aligns with the major axis of the ellipse.	51
5.15	C-specimen (a) dimensions and (b) mesh. Displacement is applied at the nodeset in the center of the plug. The crack front nodes are highlighted along the bottom surface of the mesh.	52
5.16	Domain 10 for C-specimen model as calculated by (a) Abaqus and (b) Sierra for an identical mesh. Elements in red are included in the domain.	52
5.17	Load versus load line opening results from Abaqus and Sierra for C-specimen.	53
5.18	J at the center of the C-specimen plotted against the contour for Abaqus calculation and three weight functions in Sierra. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown.	53
5.19	J at the surface of the C-specimen plotted against contour for Abaqus calculation and three weight functions in Sierra. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown.	54

5.20	Domain region and equivalent plastic strain at free surface of C-specimen at various loads calculated in Sierra. We correlate the fact that the calculated J is not converged with the observation that the plastic zone is not encompassed within the contour regions (a). Large radii are necessary to investigate the radial convergence of J	55
5.21	J -integral through the thickness of the C-specimen. The driving forces for the Abaqus and Sierra are comparable. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown. The J -integral is sampled from the contour 10.	56

List of Tables

3.1	Global Variables for J -integral	22
3.2	Nodal Variables for J -integral	22
3.3	Element Variables for J -integral	22
5.1	Summary of Case Studies for J -integral	35
5.2	Percent error in the J -integral with respect to domain, weight function, sub-domain, and mesh size.	38

Chapter 1

Introduction

A J -integral capability for Sandia codes previously consisted of a multi-step, highly user-input dependent and error-prone process that calculated the J -integral values as a post-processing step on the output exodus database. An effort within the Hydrogen Failure project (under Physics & Engineering Models) was initiated to add a domain integral and streamline the procedure. As a result, all calculations now occur within the Sierra codes and the analyst need only specify the crack front and crack face and algorithms within Sierra mechanics will determine the regions, weight functions, and the J -integral values automatically. Multiple weight functions have been implemented. The user has the choice to calculate J using a linear, plateau, or hybrid plateau-ramp function. Results from all three of these methods will be discussed.

Abaqus calculations were run using version 6.7.1 on the server Spurr using one or two processors. Unless otherwise noted, Sierra calculations were run on Sandia's compute server Redsky using the version of the day (VOTD) executable. Sierra 4.26 will be the first release version to contain a bug-free pressure correction. Any example shown in this document can be replicated using Sierra 4.26.

This report is broken into sections that make the material more accessible to the reader. Chapter 2 provides the reader with information needed to calculate the J -integral via a domain (volume) integral. Chapter 3 details the implementation and syntax employed in Sierra Mechanics. Methodologies for selecting domains and the weight function are covered. The next chapter, Chapter 4, clearly illustrates the additional steps in Cubit that are required to ensure orthogonal meshes. Details are also given with regard to mesh construction. The bulk of the document resides in Chapter 5. Cases for straight crack fronts, curved crack fronts, and non-planar crack faces are presented. An elastic-plastic case is also provided. Finally, we conclude and motivate future work in Chapter 6.

It is our hope that a robust methodology for calculating the J -integral for innumerable geometries and loadings can enable Sandia analysts to employ fracture mechanics without undue burden. We will continue to strive to generalize the methodology. In addition, we will seek to include both pre-processing and post-processing tools to ensure a rapid turnaround in analysis time. Finally, we stress that this document is *not* a primer on how to apply the J -integral to elastic-plastic fracture mechanics. The successful application of the J -integral as a criteria for crack extension in inelastic bodies is beyond the scope of the current document.

Chapter 2

Background

J is analogous to the energy release rate G from linear elastic fracture mechanics. For a body with potential energy π having a crack of length a , Irwin [6], Eshelby [3], and Rice [12] define the applied energy release rate or driving force to be $G = J = -\delta\pi/\delta a$. We term J the driving force because J is a configurational force. In the seminal work of [3], Eshelby

The negative gradient of the total energy with respect to the position of an imperfection may conveniently be called the force on it. This force, in a sense fictitious, is introduced to give a picturesque description of energy changes, and must not be confused with the ordinary surface and body forces acting on the material.

Crack propagation occurs when $J(a) = R(a)$, where $R(a)$ is the material resistance. For constant R , the resistance is often referred to as the critical energy release rate J_c . In mode I, the critical energy release rate is termed J_{Ic} . For a review of the J -integral and its application for infinitesimal deformations, please refer to oft-referenced texts on fracture mechanics [2, 7, 1].

In the reference configuration, the vectorial form of the J -integral in finite deformation [11] is

$$\mathbf{J} = \int_{\Gamma_0} \boldsymbol{\Sigma} \mathbf{N} dA \quad (2.1)$$

where $\boldsymbol{\Sigma} = W\mathbf{I} - \mathbf{F}^T \mathbf{P}$ is referred to as the Eshelby energy-momentum tensor [4] and $\mathbf{N}$ is the normal to surface Γ_0 in the reference configuration. W is the stored energy density in the reference configuration and $\mathbf{F}$ and $\mathbf{P}$ are the deformation gradient and first Piola-Kirchhoff stress, respectively. Rice [12] realized that because $\boldsymbol{\Sigma}$ is divergence-free in the absence of body forces, one can examine $\mathbf{J}$ in the direction of the defect $\mathbf{L}$ (unit vector) and obtain a path-independent integral for traction-free crack faces. J can be written as

$$J = \int_{\Gamma_0} \mathbf{L} \cdot \boldsymbol{\Sigma} \mathbf{N} dA \quad (2.2)$$

and interpreted as a path-independent driving force in the direction of the defect. We note that one can also express $\boldsymbol{\Sigma}$ in terms of $\bar{\boldsymbol{\Sigma}}$, where $\bar{\boldsymbol{\Sigma}} = W\mathbf{I} - \mathbf{H}^T \mathbf{P}$ and $\mathbf{H} = \text{Grad } \mathbf{u}$. Although $\boldsymbol{\Sigma}$ is symmetric and $\bar{\boldsymbol{\Sigma}}$ is not symmetric, they are equivalent when integrated over the body ($\text{Div } \mathbf{P} = \mathbf{0}$). In fact, differences in the energy-momentum tensor stem from

the functional dependence of the stored energy function W . Σ and $\bar{\Sigma}$ derive from $W(\mathbf{F})$ and $W(\mathbf{H})$, respectively. When integrated, both collapse to the familiar 2-D relation for infinitesimal deformations.

$$J = \int_{\Gamma} \mathbf{e}_1 \cdot \Sigma \mathbf{n} ds = \int_{\Gamma} (W n_1 - u_{i,1} \sigma_{ij} n_j) ds \quad (2.3)$$

The above formulation for the J -integral does not account for surface tractions or temperature gradients. If the issue at hand involves both pressure loadings (on the crack faces) and/or gradients in temperature, the J -integral will no longer be path independent without modification. This will be discussed briefly in the next section, Section 2.1.

2.1 Technique for Computing J

J is often expressed as a line (2D) or surface (3D) integral surrounding the crack tip. Defining a surface over which to compute this integral and performing projections of the required field values onto that surface presents a number of difficulties in the context of a finite element code.

To compute the J -integral in a finite element code, it is more convenient to perform a volume integral over a domain surrounding the crack tip. We can then leverage the information at integration points rather than rely on less accurate projections. To do this, we follow the method described in [10]. We replace the crack direction $\mathbf{L}$ with a smooth function $\mathbf{q}$. On the inner contour of the domain Γ_0 , $\mathbf{q} = \mathbf{L}$. On the outer contour of the domain C_0 , $\mathbf{q} = \mathbf{0}$. Because the outer normal of the domain $\mathbf{M}$ is equal and opposite of the normal $\mathbf{N}$ on Γ_0 , there is a change of sign. For traction-free surfaces, we can apply the divergence theorem, enforce $\text{Div} \Sigma = \mathbf{0}$, and find that the energy per unit length $\bar{J}$ is

$$\bar{J} = - \int_{\Omega_0} \Sigma : \text{Grad } \mathbf{q} dV. \quad (2.4)$$

We note that all the field quantities are given via simulation and we choose to define $\mathbf{q}$ on the nodes of the domain $\mathbf{q}^I$ and employ the standard finite element shape functions to calculate the gradient. We can specify the crack direction $\mathbf{L}$ or assume that the crack will propagate in the direction normal to the crack front $\mathbf{N}_{CF}$. Figure 2.1 attempts to illustrate a domain Ω_0 having a boundary Γ_0 with crack front normal $\mathbf{N}_{CF}$. If $\mathbf{S}_{CF}$ is tangent to the crack front and $\mathbf{T}_{CF}$ is normal to the lower crack face, $\mathbf{N}_{CF} \times \mathbf{T}_{CF} = \mathbf{S}_{CF}$. We note that for non-planar, curving cracks, $\mathbf{N}_{CF}$, $\mathbf{T}_{CF}$, and $\mathbf{S}_{CF}$ are functions of the arc length S . For a crack front S_0 , we can define the average driving force in terms of energy per unit area as

$$J_{avg} = \frac{\bar{J}}{\int_{S_0} \mathbf{L} \cdot \mathbf{N}_{CF} dS}. \quad (2.5)$$

While the average driving force is useful for interpreting experimental findings and obtaining a macroscopic representation of the driving force, we also seek to examine the local driving

force $J(S)$. Using the finite element interpolation functions to discretize $\mathbf{L}$ through the smooth function $\mathbf{q}$, we find $\mathbf{q} = \lambda^I \mathbf{q}^I$. For a specific node K , we can define $|\mathbf{q}^K| = 1$ and $\mathbf{q}^I = \mathbf{0}$ for all other $I \neq K$ on S_0 . Note that we still need to specify the function $\mathbf{q}$ in the $\mathbf{N}_{CF} - \mathbf{T}_{CF}$ plane from the inner contour Γ_0 to the outer contour C_0 . The resulting expression for the approximate, pointwise driving force at node K on the crack front is

$$J^K = \frac{\bar{J}}{\int_{S_0} \lambda^K \mathbf{q}^K \cdot \mathbf{N}_{CF} dS}. \quad (2.6)$$

Again, we note that if the direction of propagation $\mathbf{L}$ is taken in the direction of the normal to the crack front $\mathbf{N}_{CF}$, the denominator is $\int_{S_0} \lambda^K dS$. More information regarding the pointwise approximation of J^K can be found in [13, 5].

One can effectively deal with surface tractions and temperature gradients in the construction of the J -integral. If the tractions on the crack faces, $F_0^+ + F_0^-$ are non-zero (eg., pressure loading), they will contribute to the J -integral and render the methodology path dependent. This is also true for the “caps” on the domain volume $A_0^+ + A_0^-$. We can, however, “subtract off” their contributions by construction. Referring to Equation 2.2, we employ the weight function $\mathbf{q} = \mathbf{L}$ and subtract contributions from the crack faces and end caps to find

$$J = \int_{\Gamma_0} \mathbf{q} \cdot \boldsymbol{\Sigma} \mathbf{N} dA - \int_{F_0^+ + F_0^- + A_0^+ + A_0^-} \mathbf{q} \cdot \mathbf{F}^T \mathbf{P} \mathbf{N} dA. \quad (2.7)$$

We would then apply the divergence theorem and convert the surface integral into a volume integral (with stipulations on the weight function). In the prior derivation, we employed $\text{Div} \boldsymbol{\Sigma} = \mathbf{0}$. If we would like to include body forces or the dependence of the stored energy function on temperature, we must include additional terms. For $W = W(\mathbf{F}, \theta)$, we find the domain integral $\bar{J}$ to be

$$\bar{J} = - \int_{\Omega_0} \boldsymbol{\Sigma} : \text{Grad} \mathbf{q} dV - \int_{\Omega_0} \frac{\partial W}{\partial \theta} \text{Grad} \boldsymbol{\theta} \cdot \mathbf{q} dV - \int_{F_0^+ + F_0^- + A_0^+ + A_0^-} \mathbf{q} \cdot \mathbf{F}^T \mathbf{P} \mathbf{M} dA. \quad (2.8)$$

Moreover, one can also add other terms that affect the free energy in the same manner. For example, if the concentration of a species γ affects the free energy $W = W(\mathbf{F}, \theta, \gamma)$, one should also include those terms in the domain integral. To be clear, we can also express $\mathbf{P} \mathbf{N}$ as the traction in the reference configuration $\mathbf{T}$. We note that this traction will most likely stem from an applied traction such as a pressure boundary condition and we are required to “push-back” the applied traction in the current configuration to an applied traction in the reference configuration. To note this change in notation, we re-express the volume integral as

$$\bar{J} = - \int_{\Omega_0} \boldsymbol{\Sigma} : \text{Grad} \mathbf{q} dV - \int_{\Omega_0} \frac{\partial W}{\partial \theta} \text{Grad} \boldsymbol{\theta} \cdot \mathbf{q} dV - \int_{F_0^+ + F_0^- + A_0^+ + A_0^-} \mathbf{q} \cdot \mathbf{F}^T \mathbf{T} dA. \quad (2.9)$$

To be consistent, we can also view $\bar{J}$ with regard to $\bar{\boldsymbol{\Sigma}}$ and express the volume integral as

$$\bar{J} = - \int_{\Omega_0} \bar{\boldsymbol{\Sigma}} : \text{Grad} \mathbf{q} dV - \int_{\Omega_0} \frac{\partial W}{\partial \theta} \text{Grad} \boldsymbol{\theta} \cdot \mathbf{q} dV - \int_{F_0^+ + F_0^- + A_0^+ + A_0^-} \mathbf{q} \cdot \mathbf{H}^T \mathbf{T} dA. \quad (2.10)$$

Currently, the pressure correction is included within Sierra. We have not, however, included thermal gradients in the current formulation. We note that the pressure correction is applied to both the crack faces and the caps of the domain volume.

Figure 2.1. A domain integral over Ω_0 having boundary Γ_0 that surrounds a thumbnail defect. The crack front tangent, crack front normal, and crack face normal are $\mathbf{S}_{CF}$, $\mathbf{N}_{CF}$, and $\mathbf{T}_{CF}$, respectively.

Chapter 3

Sierra implementation

The J -integral can be calculated in Sierra with recently implemented capabilities. One must only specify the crack front nodes, crack plane, and the number of domains within a crack front radius. In addition, one can choose between linear, plateau, and plateau-ramp weight functions to calculate the domain integral.

3.1 Input Commands

A user can request that J -integrals be computed during the analysis by including one or more `J INTEGRAL` command blocks in the `REGION` scope. This block can contain the following commands:

```
BEGIN J INTEGRAL <jint_name>
#
# integral parameter specification commands
CRACK DIRECTION = <real>dir_x <real>dir_y <real>dir_z
CRACK PLANE SIDE SET = <string list>side_sets
CRACK TIP NODE SET = <string list>node_sets
INTEGRATION RADIUS = <real>int_radius
NUMBER OF DOMAINS = <integer>num_domains
FUNCTION = PLATEAU|PLATEAU_RAMP|LINEAR(PLATEAU)
USE SURFACE FOR EDGE DIRECTION = ON|OFF(ON)
SYMMETRY = OFF|ON(OFF)
DEBUG OUTPUT = OFF|ON(OFF) WITH <integer>num_nodes NODES ON THE CRACK FRONT
#
# time period selection commands
ACTIVE PERIODS = <string list>period_names
INACTIVE PERIODS = <string list>period_names
END J INTEGRAL <jint_name>
```

A set of parameters must be provided to define the crack geometry and the integration domains used in the calculation of the J -integral. The model must be set up so that there is a sideset on one surface of the crack plane behind the crack tip and a nodeset containing the

nodes on the crack tip. Both the **CRACK PLANE SIDE SET** and **CRACK TIP NODE SET** commands must be used to specify the names of the sideset behind the crack tip and the nodeset on the crack tip, respectively.

By default, the direction of crack propagation is computed from the geometry of the crack plane and tip as provided in the crack plane sideset and crack tip nodeset ($\mathbf{L} = \mathbf{N}_{CF}$).

The **CRACK DIRECTION** command can optionally be used to override the direction of crack propagation $\mathbf{L}$ computed from the geometry. This command takes three real numbers that define the three components of the crack direction vector as arguments.

To fully define the domains used for the domain integrals, the radius of the domains and the number of domains must also be specified. A series of disc-shaped integration domains are formed with varying radii going out from the crack tip. The **INTEGRATION RADIUS** command is used to specify the radius r_{outer} of the outermost domain. The number of integration domains is specified using the **NUMBER OF DOMAINS** command. The radii of the domains increase linearly going from the innermost to the outermost domain.

The weight function $\mathbf{q}$ used to calculate the J -integral is specified by use of the option **FUNCTION** command line. The **LINEAR** function set the weight function to 1.0 on the crack front Γ_0 and 0.0 at the edge of the domain C_0 , r_{outer} away from the crack tip. The **PLATEAU** function, which is the default behavior, sets all values of the weight function to 1.0 that lie within the domain of integration and all values outside of the domain are set to 0.0. This allows for integration over a single ring of elements at the edge of the domain. The third option for the **FUNCTION** command is **PLATEAU_RAMP**, which for a single domain will take on the same values as the **LINEAR** function. However, when there are multiple domains over the radius r_{outer} , the n^{th} domain will have weight function values of 1.0 over the inner $(n - 1)$ domains and will vary from 1.0 to 0.0 over the outer n^{th} ring of the domain. These functions can be seen graphically in Figure 3.1.

We note that in employing both the **PLATEAU** and the **PLATEAU_RAMP** functions, one is effectively taking a line integral at finite radius (albeit different radii). In contrast, the **LINEAR** option can be viewed as taking the $\lim \Gamma_0 \rightarrow 0^+$.

Future sections will describe the calculation of the weight function $\mathbf{q}$. The command **USE SURFACE FOR EDGE DIRECTION** indicates whether the element edge or the adjacent geometric surface is employed to find the crack direction $\mathbf{L}$.

If the model is a half symmetry model with the symmetry plane on the plane of the crack, the optional **SYMMETRY** command can be used to specify that the symmetry conditions be accounted for in the formation of the integration domains and in the evaluation of the integral. The default behavior is for symmetry to not be used.

The user may optionally specify the time periods during which the J -integral is computed. The **ACTIVE PERIODS** and **INACTIVE PERIODS** command lines are used for this purpose.

Figure 3.1. Example weight functions for a J -integral integration domain. Weight functions shown for domain 5.

3.2 Output

A number of variables are generated for output when the computation of the J -integral is requested. The average value of J for each integration domain is available as a global variable, as described in Table 3.1. The pointwise value of J at nodes along the crack for each integration domain is available as a nodal variable, as shown in Table 3.2. Element variables such as the Eshelby energy-momentum tensor and fields defining the integration domains are also available, as listed in Table 3.3.

If the `DEBUG OUTPUT` line command is set to `ON`, the weight functions, q , will be output for each node based j value that is calculated. The user must specify the number of nodes on the crack front via the remainder of the command line, `WITH <integer>num_nodes ON CRACK FRONT`. This will result in an extensively large output file due to the fact that every node in the integration domain will now compute and carry $(NumNodeOnCrackFront + 1) * NumDomains$ weight function vectors around. This will potentially result in memory errors if there is not a significant number of processors to better distribute the memory requirements.

Table 3.1. Global Variables for J -integral

Variable Name	Type	Comments
j_average_<jint_name>	Real[]	Average value of the J -integral over the crack. Array sized to number of integration domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

Table 3.2. Nodal Variables for J -integral

Variable Name	Type	Comments
j_<jint_name>	Real[]	Pointwise value of J -integral along crack. Array sized to number of integration domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

Table 3.3. Element Variables for J -integral

Variable Name	Type	Comments
energy_momentum_tensor	FullTen36	Energy momentum tensor
integration_domains_<jint_name>	Integer[]	Flag indicating elements in integration domains. Set to 1 if in domain, 0 otherwise. Array sized to number of domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

3.3 Sierra methodology for calculating q

Computing the J -integral within Sierra is a two-phase process which includes initialization and calculation. We note that the J -integral computation only occurs when application output is written. Because the J -integral is computed in the reference configuration, Sierra computes crack front quantities only during initialization. This primarily includes the crack front direction at each node along the crack front, which is needed to compute the weight function q . Other quantities computed during initialization include: a per node line length, a per node search width, neighboring nodes along the crack front, and the vector normal to the crack front direction. All of these quantities are used during the domain integration to compute J and are described in more detail in the following sections.

3.3.1 Computing Necessary Crack Tip Information During Initialization

During program initialization the J -integral routines compute static quantities that are used during program execution when the J -integral values are calculated. To compute the nodal q vector for all nodes in the integration domain, a number of crack front node specific data must be computed. This information is best described via the data structure that is used within Sierra to store this information:

```
struct CrackFrontNodeData {
    Vec3d m_location;
    Vec3d m_direction;
    Vec3d m_dirNormal;
    Vec3d m_tempDirection;
    Real m_searchWidth;
    Real m_lineLength;
    Int m_neighborOne;
    Int m_neighborTwo;
    Int m_numNeighbors;
}
```

To clarify, Vec3d is a class that represents three dimensional vectors, a Real is a double precision and Int is an integer. m_location is the global position of the crack front node, m_direction is the direction of the crack front normal at a particular node, m_dirNormal is the normal to the crack front face, m_tempDirection is a temporary direction stored to handle edge conditions, m_searchWidth is the width of the slice used to find nodes within an integration domain for computing J at a node along the crack front, m_lineLength is the distance along the crack front for computing J at a node, and m_neighborOne, m_neighborTwo and m_numNeighbors are used to store IDs of node neighbors on the crack front. Using this data structure allows the code to concisely store information about the crack front globally across all processors. The following steps are used to fill the above data structure in order to

facilitate the computation of $\mathbf{q}$, the integration weight vector, at each node in the integration domain.

The first step is to compile a parallel consistent list of crack front node locations and global IDs. This is accomplished by each processor looping over its locally owned¹ crack front nodes and saving the global coordinates of each node and then performing a parallel gather operation to make this list parallel consistent.

Figure 3.2. How the crack front direction is determined, pictorially.

From here the code computes the crack front directions based on the surface defined by the line command `CRACK PLANE SIDE SET`, which is done over several steps. First, the code loops over each face in the side set and then over each node of each face. If a face in the side set contains two nodes that lie on the crack front, which is determined by checking the face node's global ID against the parallel consistent list of crack front node IDs, then a crack front edge normal is computed for that face. The edge normal, $\mathbf{d}_i$ as seen in Figure 3.2, is the normalized vector that points from the centroid of the face to the closest projected point on the crack front. The face centroid is computed by averaging the four nodal locations of the face, $\mathbf{P}_c = \sum_{i=1}^n \mathbf{P}_i$. If we define the following two vectors: $\mathbf{v}_{32} = \mathbf{P}_2 - \mathbf{P}_3$ and $\mathbf{v}_{3c} = \mathbf{P}_c - \mathbf{P}_3$ where $\mathbf{v}_{32}$ is a vector along the crack front and $\mathbf{v}_{3c}$ is the vector that points from a node on the crack front to the centroid of the face, then we can compute the closest projected points as

$$\mathbf{P}_{cpp1} = \mathbf{P}_3 + \left(\mathbf{v}_{3c} \cdot \frac{\mathbf{v}_{32}}{\|\mathbf{v}_{32}\|} \right) \frac{\mathbf{v}_{32}}{\|\mathbf{v}_{32}\|} \quad (3.1)$$

Now, the edge direction is computed as $\mathbf{d}_1 = \mathbf{P}_{cpp1} - \mathbf{P}_{c1}$ with an edge length of $l_1 = \|\mathbf{v}_{32}\|$. If the node is attached to another face as shown in Figure 3.2, we can find $\mathbf{d}_2 = \mathbf{P}_{cpp2} - \mathbf{P}_{c2}$ with an edge length of l_2 . We could construct the crack front normal through

¹Locally owned implies the node object was created and lives on that processor.

a mere average of adjacent vectors ($\mathbf{d}_1$, $\mathbf{d}_2$), or we can construct a more accurate normal $\mathbf{N}_{CF}$ at the node given l_1 , and l_2 . First, we assume a convex combination of the normals and constrain that combination to be parallel to the desired normal $\mathbf{N}_{CF}$ through

$$(\alpha \mathbf{d}_1 + (1 - \alpha) \mathbf{d}_2) \times \mathbf{N}_{CF} = \mathbf{0} \quad (3.2)$$

and solve for the coefficient α analytically for the simplified case of a circular crack with a sufficient radius to invoke the small angle assumption. The solution for alpha is

$$\alpha = \frac{l_2}{l_1 + l_2} \quad (3.3)$$

and the desired normal $\mathbf{N}_{CF}$ can be defined to be

$$\mathbf{N}_{CF} = \frac{l_2 \mathbf{d}_1 + l_1 \mathbf{d}_2}{\|l_2 \mathbf{d}_1 + l_1 \mathbf{d}_2\|}. \quad (3.4)$$

In addition to looking at circular defects, the algorithm was employed for elliptical crack fronts in Mathematica. The results were very encouraging. The error was orders of magnitude lower than an simple average of $\mathbf{d}_1$ and $\mathbf{d}_2$.

We note that the current algorithm relies on the element boundaries to be orthogonal to the crack front. An alternative approach would find the normal to the plane of one facet $\mathbf{T}_1$

$$\mathbf{T}_1 = \frac{(\mathbf{P}_2 - \mathbf{P}_{c1}) \times (\mathbf{P}_3 - \mathbf{P}_{c1})}{\|(\mathbf{P}_2 - \mathbf{P}_{c1}) \times (\mathbf{P}_3 - \mathbf{P}_{c1})\|} \quad (3.5)$$

and then find the normal to the crack front

$$\mathbf{d}_1 = \frac{\mathbf{T}_1 \times \mathbf{v}_{32}}{\|\mathbf{T}_1 \times \mathbf{v}_{32}\|}. \quad (3.6)$$

One can then employ the same process for weighting the normals at the faces, $\mathbf{d}_1$ and $\mathbf{d}_2$, to find the crack front normal at the node $\mathbf{N}_{CF}$. Although the more accurate representation of the normal through a cross product has not yet been implemented, the methodology is straightforward and may be included in a future release.

During the process for finding the crack front normal $\mathbf{N}_{CF}$, we can also compute the line length l_i for each node. The line length is the average of the two adjacent crack front segment lengths. The search width is the minimum of the two adjacent crack front segment lengths. The tangent vector $\mathbf{S}_{CF}$ can be found by averaging the two line segment vectors $\mathbf{v}_{32}$ for each node on the crack front. A future release may calculate a more accurate tangent along the crack front through a cross product. This can be accomplished by using a weighted value of the facet normal $\mathbf{T}_{CF}$

$$\mathbf{T}_{CF} = \frac{l_2 \mathbf{T}_1 + l_1 \mathbf{T}_2}{\|l_2 \mathbf{T}_1 + l_1 \mathbf{T}_2\|} \quad (3.7)$$

and then the cross product to find the tangent vector $\mathbf{S}_{CF}$

$$\mathbf{S}_{CF} = \mathbf{N}_{CF} \times \mathbf{T}_{CF}. \quad (3.8)$$

3.3.2 Computing J During an Output Request

J -integral values are only computed when Sierra processes output requests. It firsts computes the discrete nodal J values along the length of the crack front and then computes the global, or average, J value. It is done in this order so if the user is requesting the integration domain or weight function $\mathbf{q}$ as part of the output, they will receive the domain and/or weight function for the global $\mathbf{q}$ not for an arbitrary node along the crack front. However, as noted earlier, if the user turns on the debug output via the `DEBUG OUTPUT = ON WITH <integer>num_nodes NODES ON THE CRACK FRONT` command line, they will get weight functions and integration domains for every node along the crack front in addition to the global domains/weight functions. We do not recommend this as the default output.

There are five steps for computing J within Sierra: compute the weight function $\mathbf{q}$, compute the integration domain, compute the gradient of the weight function, compute the length of the crack front to integrate over (line length) and finally compute the J value itself. The calculation of the weight function is detailed enough to warrant its own sections below with a discussion of the other steps following. Keeping with the order of execution, the following section will explain the computation of $\mathbf{q}^I$ for J^K followed by a section describing how the global $\mathbf{q}(S)$ is computed.

Specification of $\mathbf{q}^I$ for J^K

For a given node K along the crack front, the $\mathbf{q}$ values for each node I in the integration domain are computed using the following equation

$$\mathbf{q}^I = w_p w_f \mathbf{N}_{CF}^K, \quad (3.9)$$

where w_p is a weight factor dependent on the search width sw_K at node K along the crack front and the distance d_p from node I to the closest projected point on a plane defined by the crack front normal $\mathbf{N}_{CF}^K$ and the crack face normal $\mathbf{T}_{CF}^K$. One is essentially searching with a plane aligned with the direction of propagation that runs perpendicular to the crack faces. The “radial” weight factor w_f is a weight factor dependent on which type of weight function we are using, as defined by the `FUNCTION` command line, and the distance from node I from to the closest nodal point on the crack front, d_f . Based on the definition of d_p and the search width for node K , sw_K , the weight function w_p is defined as

$$w_p = \begin{cases} 0 & \text{if } d_p \geq sw_K \\ sw_K - d_p & \text{if } d_p < sw_K. \end{cases} \quad (3.10)$$

Since the weight function w_f is based on the function the user specified in the input deck, it can take on three forms which are dependent on the distance d_f .

For the `LINEAR` function w_f is defined as

$$w_f = \begin{cases} 0 & \text{if } d_f > r_d \\ 1 - \frac{d_f}{r_{outer}} & \text{if } d_f < r_{outer}. \end{cases} \quad (3.11)$$

For the PLATEAU function w_f is defined as

$$w_f = \begin{cases} 0 & \text{if } d_f > r_{outer} \\ 1 & \text{otherwise.} \end{cases} \quad (3.12)$$

For the PLATEAU RAMP function w_f is dependent on the number of domains requested, n_d . If the current domain of interest is d , the current domain radius is r_d and the radius of the next smallest domain is r_{d-1} , then the w_f is defined as

$$w_f = \begin{cases} 1 & \text{if } d_f < r_d \\ 1 - \frac{d_f - r_{d-1}}{r_d - r_{d-1}} & \text{otherwise.} \end{cases} \quad (3.13)$$

Special treatment at the A_0^+ and A_0^- boundaries

Initially the search width sw_K was set by the maximum of the two adjacent crack front line segments. This can induce errors at the first interior node if the line segments are not equivalent. This stems from the fact that $\mathbf{q}$ will be non-zero at the boundary and one cannot integrate past the boundary. Even if one chooses the minimum of the segment lengths, errors can still occur at the first interior node along the crack front.

For a curved crack front and the current methodology for assigning the $\mathbf{q}$ vectors, the $\mathbf{q}$ vectors on nodes behind the crack front on the surface A_0^+ , A_0^- will be non-zero for the first interior node. Note that the crack faces are F_0^- and F_0^+ . While this does incur some error it can also yield additional errors if tractions are applied on A_0^+ , A_0^- or F_0^+ , F_0^- . These cases often cause “spikes” in the driving force at the first interior node. A methodology to correct this issue for the first interior node is to set $\mathbf{q}^I = \mathbf{0}$ for all surface nodes A_0^+ or A_0^- . This will increase the accuracy of the domain integral and ensure that the pressure correction given in Equation 2.10 is zero.

This methodology is illustrated in Figure 3.3. In setting the $\mathbf{q}$ vectors for node K , we need to determine if node J lives on the boundary A_0^+ . This can be accomplished by determining which crack front node is closest to node J . For this case, the crack front node closest to node J would be node I . Because we know with certainty that node I is on an edge (it only has one neighbor on the crack front), we conclude that node J is also on a boundary and we set $\mathbf{q}^J = \mathbf{0}$. Because $\mathbf{q} = \mathbf{0}$ on the entire surface A_0^+ , there will not be a contribution from the pressure correction term at node K .

Specification of a Global $\mathbf{q}$

The weight function $\mathbf{q}^I$ for node I in the integration domain for computing the average value J_{avg} is defined as

$$\mathbf{q}^I = w_f \mathbf{N}_{CF,c} \quad (3.14)$$

Figure 3.3. Scheme to more accurately calculate the driving force at the first interior node. This methodology will mitigate spikes in the driving force associated with an erroneous pressure correction.

where w_f was previously defined for LINEAR, PLATEAU, and PLATEAU RAMP and $\mathbf{N}_{CF,c}$ is the crack front normal of the closest node on the crack front from node I in the integration domain.

Computing J

As noted above, once the weight function $\mathbf{q}$ is computed, the steps necessary to complete the calculation of J are: find all elements within the integration domain, compute the gradient of $\mathbf{q}$, compute the integrated line length along the crack front and compute J . These steps are fairly straight forward and described below.

Because the domain integral is expressed as a summation over elements, and the gradient of $\mathbf{q}$ is part of that expression, an element is considered to be within the integration domain if any of its nodal $\mathbf{q}$ vector components have a nonzero magnitude. The code loops over all elements that can potentially be in the domain, checks the magnitude of each node's $\mathbf{q}$ vector and marks the nodes as in the domain or out of the domain appropriately.

The gradient of $\mathbf{q}$ is computed via a set of standard gradient operators for each element type. However, the current implementation of the J -integral calculation only supports uniform gradient hex elements, and as such, this call is hard coded for these elements. Development is underway to generalize this method to make all elements available to the J -integral calculation.

Assuming that the crack propagates in the direction normal to the crack front, we can find the integrated distance along the crack front l though a sum of discretized line lengths

l_K for each node K along the crack front. The average driving force calculated along the crack front as J_{avg} is

$$J_{avg} = \frac{1}{l} \sum_{i=0}^n \Sigma_i : \text{Grad } \mathbf{q}_i \quad (3.15)$$

where n is the number of elements in the integration domain. Σ_i and $\text{Grad } \mathbf{q}_i$ are the energy momentum tensor and gradient of the weight function for element i in the integration domain, respectively. The weight function corresponding to J_{avg} on the crack front is illustrated in Figure 3.4. For ease of illustration, the weight function is only shown for the nodes along the crack front. The weight function will decay in the direction of the normal depending on the choice of weight function (linear, plateau, plateau_ramp) and the domain radius.

Figure 3.4. The weight function $\mathbf{q}$ corresponding to J_{avg} for all nodes along the crack front. The weight function is automatically calculated to be normal to the crack front.

Although finding J_{avg} can be useful, our ultimate goal is to calculate a spatially-varying driving force at a node K with a corresponding a set of weight functions $\mathbf{q}^K$. To find the driving force at a particular crack front node, we employ

$$J^K = \frac{1}{l_K} \sum_{i=0}^n \Sigma_i : \text{Grad } \mathbf{q}_i^K \quad (3.16)$$

where $\mathbf{q}_i^K$ now corresponds to a weight function specific to node K . The weight function for node K is illustrated in Figure 3.5. Note that magnitude of the weight function $\|\mathbf{q}\| = 1$

is displayed. The entire integration domain is included to illustrate the local nature of the weight function for node K .

Figure 3.5. The magnitude of the weight function q^K corresponding to J^K for node K along the crack front. Note that node K is the first node interior from the boundary and the weight function is a plateau “tent” function that is 1 along the normal emanating from node K . The weight function is automatically calculated for each node K along the crack front. The entire body is shown to clearly illustrate that $\|q\| = 0$ for nodes outside the search width sw_K .

Chapter 4

Required discretization in CUBIT

In order to enable mesh quality, the closest point projection to a crack-front node, and the correct prescription of the test function $\mathbf{q}^I$, the hexahedral mesh should be orthogonal to the crack front. An orthogonal mesh will ensure the elements are not skewed along the crack front. Because these elements will experience large deformation during crack-tip blunting, well-formed elements increase the accuracy of the solution. We note that this capability is not specific to crack front nodes. Any ellipsoidal surface with a constant bias will generate skewed elements.

In addition, an orthogonal mesh will ensure that the location of a point-wise surface integral will be a closest point projection from the crack-tip node. Consequently, any surface integral via a domain integral at a node along the crack front will be most accurate if the specified radius is a minimum. In addition to increasing the accuracy of point-wise evaluations of the J -integral, an orthogonal mesh will also ease the search algorithm for point-wise evaluations. The current algorithm is discussed in Section 3.3.2. A search is performed along the normal to the crack front. If the mesh is aligned with the normal, the specification of $\mathbf{q}$ is straightforward. Misalignment can result in a “checkboarding” of the integration domains and there exists the possibility that $\mathbf{q}^I$ will always be one and the J -integral will be zero.

Because the methodologies employed in Section 3.3.2 require element edges that are normal to the crack front, we need to ensure that the crack tip elements are indeed orthogonal. Future work may generalize the calculation of J^K but we are currently limited to hexes. Given these requirements, we collaborated with the Cubit team to add the capability to generate meshes that are orthogonal to a surface. The Cubit team implemented the command

```
adjust boundary surface XX snap_to_normal curve XX
```

which enables the generated elements along a curve to be “snapped” normal to the curve. A typical journal file and the resulting mesh is illustrated in the next section.

4.1 Ellipsoidal flaw

We begin with a simple example of an ellipsoidal flaw. A journal file is noted below.

```

#{tn_rad_min = 0.4}
#{tn_rad_maj = 1.5 * tn_rad_min}
#{r_contour= 0.085}
#{cyl_height = tn_rad_maj}

create surface ellipse major radius {tn_rad_maj + r_contour} minor radius {tn_rad_min + r_contour} zplane
#{ellipse_surf = Id("Surface")}
sweep surface {ellipse_surf} perpendicular distance {cyl_height}
#{ellipse_vol = Id("Volume")}

create surface ellipse major radius {tn_rad_maj} minor radius {tn_rad_min} zplane
#{ellipse_surf2= Id("Surface")}
sweep surface {ellipse_surf2} perpendicular distance {cyl_height}
#{ellipse_vol2 = Id("Volume")}

create surface ellipse major radius {tn_rad_maj - r_contour} minor radius {tn_rad_min - r_contour} zplane
#{ellipse_surf3 = Id("Surface")}
sweep surface {ellipse_surf3} perpendicular distance {cyl_height}
#{ellipse_vol3 = Id("Volume")}

webcut volume 1 with sheet extended from surface 5
webcut volume 2 with sheet extended from surface 8
delete volume 3 to 5

webcut volume all with plane xplane noimprint nomerge
webcut volume all with plane yplane noimprint nomerge

delete volume with y_coord < 0 or x_coord < 0

imprint all
merge all

curve 57 interval 32
mesh curve 57

surface 55 63 scheme map
mesh surface 55 63

adjust boundary surface 55 snap_to_normal curve 57
adjust boundary surface 63 snap_to_normal curve 57 fixed curve 57

```

The results of the included journal file are illustrated in Figure 4.1. Figure 4.1(a) illustrates the mapped mesh along an ellipsoidal flaw, curve 57. First, surface 55 is adjusted through the `snap_to_normal` command. This will enable the element edges in surface 55 to be orthogonal to curve 57. The next step involves adjusting surface 63 while fixing the node positions along curve 57. This will enable continuity in the normal through curve 57. The resulting mesh is illustrated in Figure 4.1(b). The process of mapping and snapping has been very effective for the ellipsoidal flaws investigated in this work and is now a supported feature in Cubit thus ensuring that analysts will be able to obtain accurate assessments of the driving force with little difficulty in the meshing process.

We note that one can also ensure that the nodes on curve 57 remain unchanged through the `snap_to_normal` command by issuing

```
node in curve 57 fixed
```

Figure 4.1. Screenshots of Cubit that illustrate the new `snap_to_normal` command. Through two `snap_to_normal` operations, the mesh is adjusted to be orthogonal to curve 57.

prior to adjusting the boundary on surface 63. We note that the accuracy of the normal does depend on a sufficient discretization along crack front. This can be investigated through studies in mesh refinement.

Chapter 5

Case Studies

We analyzed five geometries with increasing complexity to fully exercise the J -integral implementation in Sierra. The case studies, summarized in table 5.1, vary in material characteristics, crack front shape, and crack plane shape. Comparisons are made against an analytical solution, when available, or using code comparison against Abaqus, when an analytical solution does not exist. Unless otherwise noted, Sierra calculations were completed in Sierra version 4.17.

Table 5.1. Summary of Case Studies for J -integral

	Material	Crack Front	Crack Surface	Solution
Infinite Crack	Linear Elastic	Straight	Planar	Analytical
Penny Flaw	Linear Elastic	Curved	Planar	Analytical
Cone Crack	Linear Elastic	Curved	Conical	Code Comparison
Embedded Ellipse	Linear Elastic	Curved	Planar	Analytical
C-Specimen	Elastic-plastic	Straight	Planar	Code Comparison

The infinite crack case employs linear elastic material properties and has a straight crack front. The penny flaw and cone crack models also utilize elastic material properties and have curved crack fronts. While the penny flaw geometry has a planar crack face, the cone crack model has a non-planar crack face. However, both geometries are axisymmetric in nature and the calculated J -integrals should be identical at all crack front nodes. The embedded elliptical flaw also has a curved, planar crack front, but it is not axisymmetric; thus, the J values should vary along the crack front. The C-shaped compact tension specimen has a planar, straight crack front, yielding a constant $\mathbf{q}$ at all crack front nodes, but the geometry and loading condition should yield varying J 's through the specimen thickness. This model also exercises the J -integral capability using an elastic-plastic material model. To be clear, elastic in the context of this report is hypoelastic. For small strains, this assumption is consistent with linear elastic fracture mechanics (LEFM).

For each case, we compare the linear, plateau, and plateau-ramp weight functions. In all cases, the linear weight function tends to yield lower and more inaccurate driving forces. This is due to the fact that the linear weight function samples the elements at the crack

tip. As noted in Chapter 3, the linear weight function represents the true nature of the J , $\lim \Gamma_0 \rightarrow 0^+$. Regrettably, the chosen element type struggles to capture the singularity and the crack tip fields lead to inaccuracies in the driving force. To remedy this issue, the plateau and plateau-ramp weight functions were implemented. One is effectively evaluating the J -integral at increasing radii using the plateau weight function. Given the aforementioned issues at the crack tip and the fact that the J -integral is path independent, larger radii yield more accurate and convergent representations of the driving force. The default weight function in Sierra is the plateau weight function. We note that the weight function employed in Abaqus is aligned with the plateau weight function. Code comparisons with Abaqus should employ the plateau weight function.

5.1 Infinite Crack

Probably the most straightforward case is the infinite crack. For this particular geometry, a plane-strain K -field displacement boundary condition is applied at the far-field. Given a crack tip of location (x_0, y_0) we can construct both a radius r and an angle θ for each node on the boundary node set. For an applied stress intensity (K_I, K_{II}) , we can employ LEFM solutions [9] for the boundary displacement $\mathbf{u}$ where $\mathbf{u} = \mathbf{u}(K_I, K_{II}, r, \theta)$. Provided that we satisfy “small-scale yielding”, we are effectively applying the driving force. The sample discretization is illustrated in Figure 5.1. Note the the radius r of the of the body is 150 mm and the applied K_I is quite small ($\sim 1 \text{ MPa}\sqrt{\text{m}}$) to guarantee that the region of interest is small compared to the boundary. For instance, we assume a characteristic strength σ_c to be on the order of $E/10,000$. Given a Young’s modulus E of 207 GPa and a Poisson’s ratio ν of 0.3, we can find a characteristic radius r_c to be $\sim 40 \text{ }\mu\text{m}$ through the relation

$$r_c = \frac{1}{2\pi} \frac{K_I^2}{\sigma_c^2}. \quad (5.1)$$

We note that because $r_c \ll r$, we can consider the applied driving force to be the analytic solution. Although symmetry is illustrated in Figure 5.1, we only employ the upper-half of the disk and invoke the symmetry option in the J -integral. Elements at the crack tip for the the two discretizations are $h_1 = 2.5 \text{ }\mu\text{m}$ and $h_2 = 1.25 \text{ }\mu\text{m}$. Note that the ratio of boundary radius to the crack-tip element size is 60,000 and 120,000 for $h_1 = 2.5 \text{ }\mu\text{m}$ and $h_2 = 1.25 \text{ }\mu\text{m}$, respectively.

For an applied K_I of $0.89700 \text{ MPa}\sqrt{\text{m}}$, we find the driving force via Irwin [6], $J = G = K_I^2(1 - \nu^2)/E$, to be 3.5372 J/mm^2 in plane strain. For each domain, we evaluate the driving force (via the J -integral) for the three weight functions (linear, plateau, plateau-ramp) over 5 sub-domains. Table 5.2 illustrates the percent error for the different weight functions for varying domain, subdomain, and mesh size.

We note that for the smaller domain radius, $r = 25 \text{ }\mu\text{m}$, sub-domain 5 represents 10 elements in h_1 and 20 elements in h_2 . Given the analytical solution, the computational solution is reasonable with errors less than a percent. Generally, we conclude that the

Figure 5.1. Discretization employed for a crack in an infinite body. A K-field displacement boundary condition is applied to the exterior of the body.

plateau function is preferable but it was not the most accurate for all cases tested. Most codes that evaluate the J -integral employ the plateau function as it represents a surface integral at a greater radius from the crack tip. We note that because we are currently forced to employ under-integrated elements with hourglass control at the crack-tip, more detailed studies would convolute hourglass control with convergence in the driving force. When this capability is extended to all element types we will again employ this test case.

5.2 Penny Flaw

Our next case study involves a cylindrical specimen with a penny shaped flaw orthogonal to the direction of loading. Figure 5.2(a) shows a graphical representation of the specimen. For our analysis, we use a specimen with outer radius, $r = 1.0$ in, height, $h = 1.5$ in, and crack radius, $a = 0.1$ in. This gives crack ratio, a/w , of 10%. This geometry was chosen due to its simplicity and existence of an analytical solution. The values of J should be identical at all crack front nodes for this axisymmetric case.

We take advantage of the symmetric nature of the specimen by modeling half the height and a 30° wedge. Figure 5.2(b) is the mesh used in the analysis. A planar mesh is swept in the circumferential direction, except for a small plug at the centerline, which is meshed and swept in the axial direction. This allows us to create a mesh without wedge elements at

Table 5.2. Percent error in the J -integral with respect to domain, weight function, sub-domain, and mesh size.

domain radius	weight function	sub-domain	% Error, h_1	% Error, h_2
10 mm	linear	1	0.554	0.134
		5	0.379	0.0938
	plateau	1	0.436	0.108
		5	0.254	0.0652
	plateau ramp	1	0.554	0.134
		5	0.279	0.0697
25 μ m	linear	5	0.379	0.427
	plateau	5	0.968	0.240
	plateau ramp	5	0.279	0.240

the center. The mesh is refined at the crack tip to an element size of approximately 0.005 inches.

We also analyzed a 360° mesh to fully exercise the ability of Sierra to correctly calculate the direction of the q-vectors used in the J -integral calculation. In addition, we completed a uniform mesh refinement of the 30° wedge to confirm mesh convergence. The original (unrefined) 30° wedge has 174,896 elements and took approximately 1 hour to run on 16 processors on the computational cluster Redsky. We requested 5 contours within a contour radius of 0.075 inches. This radius keeps the domains in the area where the mesh is planar at each node along the crack front. Results presented in the following sections are for the 30° wedge.

5.2.1 Penny Flaw Results: Radial Convergence in J

Contour regions are calculated using a radius from the crack tip, as defined by the user in the input deck. The contour regions used for the penny flaw analysis can be seen in Figure 5.3. At larger radii, the contour area includes elements outside the regular shaped mesh. Note that the fourth contour expands the region to a highly irregular mesh because the elements are in a transitional region between the refined mesh and the coarser mesh. This may explain the slight, but marked difference in calculated J -values from the expected smooth J versus contour curve for the plateau and plateau-ramp function results seen in Figure 5.4. Ignoring this anomalous fourth contour value, the plateau and plateau-ramp functions calculate J results that slightly overshoot the J value and decrease slightly with increasing contours. This is similar to the results noted (but not illustrated) in the elastic K -field calculations detailed in Section 5.1.

We claim radial convergence if J asymptotes to a value as the radius over which J is

Figure 5.2. Penny flaw graphic and finite element mesh.

Figure 5.3. Elements in red are included in the contour over which J is calculated for the penny flaw model: (a) first contour increasing to (e) fifth contour.

calculated increases. Figure 5.4 shows the normalized J values over increasing contours at one load. Ignoring the anomalous fourth contour for the plateau and plateau-ramp functions in Sierra, the plots support the argument that radial convergence in J has been achieved at this load.

5.2.2 Penny Flaw Results: J versus Load

We compare the expected results from the analytical solution against the values calculated using the three different weight functions. The analytical solution [14] for a penny flaw cylinder under far-field uniform tension is summarized below. Results can be seen in Figure 5.5.

Figure 5.4. Normalized J versus contour for penny flaw model at various loads. Values approaching an asymptote indicate that radial convergence been achieved at this load.

$$\begin{aligned}
 K_I &= \frac{2}{\pi} \sigma \sqrt{\pi a} \\
 \text{where } K_I &= \text{stress intensity,} \\
 \sigma &= \text{far field stress, and} \\
 a &= \text{crack radius.}
 \end{aligned} \tag{5.2}$$

$$\begin{aligned}
 J &= \frac{\kappa_{11}^2}{E'}, \\
 \text{where } E' &= \frac{E}{1-\nu^2}.
 \end{aligned} \tag{5.3}$$

A uniform refinement of the 30° wedge was made, resulting in a model with 8 times as many elements as the original mesh. A full mesh refinement study with multiple refinements and different meshes is needed to claim numerical convergence. However, the results between

Figure 5.5. J versus load for pennyflaw model with original mesh refinement using three Sierra weight functions and the analytical solution.

the two refinements are comparable, as seen in Figure 5.6, lending credibility to the claim that we have numerical convergence in our model. The deviation from the analytical solution for both the original mesh and the refined mesh are below 1% error, with the plateau and plateau-ramp functions yielding more accurate results than the linear weight function.

5.3 Cone Crack

Our next case study, the cone crack problem, includes a curved crack front, but this case also includes a non-planar crack face. We used an example from the Abaqus v6.7 Example Problems manual [5] to compare against results in Sierra. We use elastic material properties, apply a uniform pressure loading, and compare against the results from Abaqus. We used the full three dimensional model for Abaqus and ran the mesh and input deck given in the example problems manual. Abaqus problems are dimensionless; we assume English units for clarity. A quarter symmetry model is used for both Abaqus and Sierra analyses. The model is a cube with 30 in. sides, has a conical crack with inner radius of 10 in. and length of 15 in. at a 45° angle from the free surface, and has a 10 psi distributed load on the cracked free face. For each analysis, 10 contour regions are requested within a radius of 5.0 in from the crack tip. Figure 5.7 shows the graphical representation of the cracked region.

The Abaqus model, *SymmConeCrackOrphan.inp*, and its associated mesh files were pro-

Figure 5.6. Percent error of J for the 30° pennyflaw model with the original and the refined mesh for three Sierra weight functions. The results indicate that we have attained mesh convergence.

vided in the Example Problems Manual. Abaqus requires that the normal to the crack plane be defined at every crack front node. As the crack in this problem is curved, this requires a rather lengthy and complicated input deck involving the definition of the normal vector at every crack front node. The Abaqus mesh of the cone crack model, shown in Figure 5.8(a), has 9,517 elements, and the input deck, excluding the node and element definitions, is approximately 1,500 lines of code. The Abaqus model takes approximately 10 minutes to run on 2 processors on Spurr.

Due to the inability to convert the Abaqus mesh to an exodus format usable by Sierra, the cone crack model was re-created and meshed using Cubit for the Sierra calculation. The Sierra mesh is more refined than the Abaqus mesh. The mesh size in Sierra does not stem from a convergence study but was chosen to be ample for the current analysis. The mesh created for Sierra, shown in Figure 5.8(b), has 431,720 elements, and the input deck has approximately 150 lines of code. The simplicity of the Sierra input deck is due to the automated fashion in which the normal to the crack plane at each crack tip node are defined. These normals must be defined explicitly for Abaqus models, causing an order of magnitude increase in the length of the input. The Sierra model took approximately 75 hours to run on 32 processors on Redsky.

Figure 5.7. Graphical representation of cone crack problem. Quarter symmetry is used for the finite element model. The conical crack is embedded in a 30 inch cube.

5.3.1 Cone Crack Results: Bulk Behavior

We compare the overall reaction load to the displacement at the center of the pressure loading. The Abaqus and Sierra results are nearly identical, lending confidence to the assumption that the geometry and materials are being interpreted similarly by both analysis codes.

5.3.2 Cone Crack Results: Radial Convergence in J

We claim radial convergence in J if the J values asymptote with increasing contour radius values. As seen in Figure 5.9, the results from Abaqus and from the three weight functions in Sierra are almost identical at lower loads; at higher loads, the results begin to deviate from each other. The Abaqus results seem to align with the plateau and plateau-ramp weight function results. We note that for the Sierra implementation, the driving force initially increases during the first few contours. With increasing contours, J slightly decreases. Infinitesimal decreases in the driving force at extended contours are consistent with the K -field and penny flaw calculations. We note that one cannot observe those decreases in Figure 5.9. We claim that J has converged at the loads of interest.

5.3.3 Cone Crack Results: J versus Load

The J versus reaction load for Abaqus and Sierra runs are shown in Figure 5.10. The calculated results are nearly identical for Abaqus and the values of J calculated from all

Figure 5.8. Domain 10 of cone crack model. Elements in red are included in the J -integral calculation.

three weight functions in Sierra. This further verifies the capability within Sierra to calculate the J -integral of a crack for specimens with linear elastic material properties.

5.4 Flat Plate with Embedded Elliptical Flaw

Our next case, the fully embedded elliptical flaw in a flat plate, was chosen to validate the case where both the weight function and the driving force varies along the crack front. The embedded elliptical flaw also exercised Sierra's capability to calculate J for a geometry with pressure applied to the crack face. This geometry also highlighted the necessity of having the ability to create mesh lines that are orthogonal to the crack front, and a separate effort to ensure that Cubit has this orthogonal meshing capability was launched. A discussion of this effort can be found in Chapter 4.

A fully embedded elliptical flaw that is orthogonal to the applied stress direction is inserted into a flat plate. Figure 5.11(a) shows a graphical representation of the specimen. For our analysis, we use a specimen with total height, $2h = 20.0$ in, total width, $2W = 20.0$ in, and total thickness, $t = 2.0$ in. The elliptical crack has a minor radius, $a = 0.1$ in that aligns with the thickness direction of the plate, and a major radius, $c = 0.2$ in that aligns with the width of the plate. Figure 5.11(b) shows the dimensions of the elliptical crack, as well as the convention for defining the angle, ϕ , as a measure of location along the crack front. $\phi = 0$ aligns with the width direction of the plate, or the major axis of the elliptical crack.

The finite element model includes half the height, half the thickness, and half the width of the plate (1/4 elliptical crack, or 1/8 of the specimen). The mesh is refined at the crack tip to an element size of approximately 0.0015 inches. Due to the meshing algorithm used to ensure mesh lines orthogonal to the crack front, the elements normal to the crack front

Figure 5.9. J versus contour of cone crack specimen at various loads.

are not exactly identical in size. Figure 5.12(a) shows the mesh used in the analysis.

The embedded elliptical flaw mesh has 1 million elements and took approximately 6 hours to run on 64 processors on the computational cluster Redsky. We requested 10 contours within a contour radius of 0.035 inches for each model. This model was analyzed in Sierra version 4.25.

5.4.1 Embedded Ellipse Results: Radial Convergence in J

Contour regions are calculated by Sierra using the user defined radius from the crack tip. The ninth contour region is used for the following results of the embedded flaw analysis can be seen in Figure 5.12(b). The results shown in the following section use this contour. We show results from the ninth contour to utilize the largest possible contour while ensuring that the elements within the domain remained planar at each location along the crack front.

We claim radial convergence in J if, at a given load, J asymptotes to a value as the radius over which J is calculated increases. Figure 5.13 illustrates the L_2 error in J with increasing contours. The error remains fairly stable after the third contour for the plateau and plateau-ramp functions. The error for the entire crack front is based on the analytical solution given in Equation 5.7. The plots support the argument that radial convergence has been achieved at this load. The linear weight function yields J values that are still slightly increasing with increasing numbers of contours. The linear weight function may require more

Figure 5.10. J (in-lb/in²) versus load P calculated for cone crack model.

domains to claim radial convergence; however, due to the geometry of the specimen, this is not possible.

5.4.2 Embedded Ellipse Results: J Along Crack Front

Due to the non-uniform nature of the flaw geometry, the driving force of the crack will vary along the crack front. The accepted solution [1] uses a combination of geometry factors and the crack-face pressure to calculate K_I , the mode I stress intensity factor.

$$K_I = PG_0 \sqrt{\frac{\pi a}{Q}}$$

where K_I = stress intensity,

P = applied uniform crack face pressure,

a = crack radius in thickness direction,

G_0 = non-dimensional geometry factor, and

Q = flaw shape parameter.

(5.4)

The non-dimensional geometry factor, G_0 , and the flaw shape parameter, Q , are defined:

Figure 5.11. Embedded elliptical crack in flat plate with pressure P applied on the crack faces.

$$\begin{aligned}
 G_0 &= [\sin^2 \phi + (\frac{a}{c})^2 \cos^2 \phi]^{1/4} \\
 Q &= 1 + 1.464(\frac{a}{c})^{1.65}
 \end{aligned}
 \tag{5.5}$$

For mode I crack opening, we also know that the stress intensity can be described in terms of the J -integral with the following relationship:

$$\begin{aligned}
 K_I &= \sqrt{\frac{JE}{1-\nu^2}} \\
 \text{where } J &= J\text{-integral,} \\
 E &= \text{Young's modulus, and} \\
 \nu &= \text{Poisson's ratio.}
 \end{aligned}
 \tag{5.6}$$

Setting the two equal to one another, we can calculate the analytical solution for J .

$$J = P^2 G_0^2 \frac{\pi a}{Q} \frac{1-\nu^2}{E}
 \tag{5.7}$$

Figure 5.14 shows the varying values of J from contour 9 at each node along the crack front for all three weight functions in Sierra. Because we have shown radial convergence in J at this load, the value at contour 9 (or any other contour greater than 3) can be used to represent the value of J at each node along the crack front.

The nodes at the edges of the mesh yield non-smooth values of the driving force, as seen by the error plot in Figure 5.14(b). However, the error at the edge nodes is still within 1%.

Figure 5.12. Mesh of embedded elliptical crack in flat plate.

Element performance (we are currently limited to a uniform-gradient formulation) might be a contributing factor. Ignoring these boundary values for J , the driving force calculated along the crack front at this load is within 0.2% of the analytical solution for the plateau and plateau-ramp case and 0.7% of the analytical solution for the linear weight function. There is a slight trend towards increasing errors at $\phi = 0$, or along the major axis of the ellipse.

5.5 C-shaped Compact Tension Specimen

Work done by Kim et. al [8] was used as a model for our final case study. The C-shaped compact tension specimen, or C-specimen, has a straight crack front and flat crack plane, resulting in a fairly simple calculation of the crack direction. However, the elastic-plastic material model and varying J -integral values across the specimen thickness adds complexity and interest to the problem.

We modeled a straight sided, i.e. not side grooved, C-specimen with crack length $a = 4.94$ mm, and ligament length, $w = 9.14$ mm, Figure 5.15(a). A quarter symmetry model, with half the height, and half the thickness, was created, and the elements were biased towards the free surface of the model, Figure 5.15(b). To avoid localization of plastic strains near the loading regions, a stiff, elastic pin was inserted into the loading hole of the C-specimen mesh. The mesh is contiguous between the pin and specimen to avoid computational complexity associated with contact surfaces. The baseline model consisted of 28,968 elements and took approximately 1 hour on 32 processors to run to a point just past yielding.

Figure 5.13. L_2 error in J versus contour for the embedded ellipse model with applied crack face pressure loading.

As no analytical solution exists, we compared the Sierra results with results obtained by analyzing the same mesh in Abaqus. A multi-linear elastic-plastic material model was used for both analyses, and we analyzed a uniaxial tension specimen to confirm that the input material behaved as expected for both analysis codes. The output stress versus strain relationship was used to confirm that the material model does not contribute to any differences that may appear in the C-specimen results.

We use identical meshes for the Abaqus and Sierra analyses; however, the two codes determine contour regions differently. Abaqus automatically creates contours by following a path of elements around the crack tip. Because the mesh at the crack tip is rectangular, the contours created are rectangular. Sierra creates contours within a circular region by including elements within each user specified radius, resulting in a pixellated circular contour region regardless of the mesh shape. Figure 5.16 illustrates the differences in contour region created by the two codes for the same mesh. We also should note that Abaqus employs a small-strain implementation of the J -integral while Sierra employs a finite-deformation implementation of the J -integral.

5.5.1 C-specimen Results: Bulk Behavior

A comparison of the load versus load-line opening for the displacement controlled C-specimen analysis in Figure 5.17 shows that there are slight deviations between the Abaqus results and the Sierra results. These differences are small at lower loads but are magnified at higher loads. As we have confirmed that the elastic-plastic material model is interpreted

correctly by each code, and we used identical meshes for both codes, we attribute some of the differences to the element formulations. The C3D8R element in Abaqus does not mirror the uniform-gradient element in Sierra. That fact that both are using hourglass stabilization complicates any comparison at large deformations. Without a more general implementation in Sierra, code comparisons only have utility at small deformations.

5.5.2 C-specimen Results: Radial Convergence in J

We claim that radial convergence in J is achieved if J converges as the contour radius increases. Because value of J varies through the thickness, as we will discuss in the next section, we use the center and the surface of the C-specimen at each contour to test for radial convergence. Figure 5.18 and Figure 5.19 show the J versus contour plots at three different loads for the center and the surface of the C-specimen, respectively. At the highest load shown on the plot, the Abaqus and Sierra results cannot be compared directly because the output interval yielded slightly different load values. This inability to control output load values is inherent in analyses that are displacement controlled.

Within the elastic region, we can comfortably assert that radial convergence in J is reached within the 10 contours that we used for loading. The results from the two smaller loads shown in figure 5.20 show that the contour region encompasses regions of plastic strain above 1%. At the higher load, the contour region does not enclose the plastic strain region, and calculated J has not converged. A larger contour radius is required to ensure radial convergence at the higher loads. The maximum contour radius is based on the specimen geometry and crack length and is not unlimited. There will be a load at which it will no longer be possible to achieve radial convergence in J .

5.5.3 C-specimen Results: J Along Crack Front

The straight sided specimen causes a “tunneling” effect - the J -integral value is lower at the free surfaces and peaks at the center-plane. In order to better capture the gradient in the driving force at the free surfaces of the specimen, the element density is higher at the free surfaces and lower at the center plane.

Figure 5.21 shows J at contour 10 at each node through specimen thickness at three loads for Abaqus results and all three weight functions for Sierra. Although we have failed to illustrate radial convergence in J for the given outer radius, the outer contour is representative of the gradient in J along the crack front. Again, note that at the highest load shown, Abaqus and Sierra results cannot be directly compared as the loads at which the curves are plotted are not identical.

(a) J along crack front for domain 9

(b) Percent error of J

Figure 5.14. J calculated from domain 9 at each node along the crack front for the embedded elliptical flaw problem with applied crack face pressure. Symmetry surface values have unresolved spikes in value. $\phi = 0$ aligns with the major axis of the ellipse.

Figure 5.15. C-specimen (a) dimensions and (b) mesh. Displacement is applied at the nodeset in the center of the plug. The crack front nodes are highlighted along the bottom surface of the mesh.

Figure 5.16. Domain 10 for C-specimen model as calculated by (a) Abaqus and (b) Sierra for an identical mesh. Elements in red are included in the domain.

Figure 5.17. Load versus load line opening results from Abaqus and Sierra for C-specimen.

Figure 5.18. J at the center of the C-specimen plotted against the contour for Abaqus calculation and three weight functions in Sierra. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown.

Figure 5.19. J at the surface of the C-specimen plotted against contour for Abaqus calculation and three weight functions in Sierra. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown.

(a) Domain 10 elements in red.

(b) Plastic Strain at $P = 1712$ N

(c) Plastic Strain at $P = 2444$ N

(d) Plastic Strain at $P = 3357$ N

Figure 5.20. Domain region and equivalent plastic strain at free surface of C-specimen at various loads calculated in Sierra. We correlate the fact that the calculated J is not converged with the observation that the plastic zone is not encompassed within the contour regions (a). Large radii are necessary to investigate the radial convergence of J .

Figure 5.21. J -integral through the thickness of the C-specimen. The driving forces for the Abaqus and Sierra are comparable. The weight function in Abaqus mirrors the plateau weight function. Note that the results from the two codes cannot be directly compared at highest load shown. The J -integral is sampled from the contour 10.

Chapter 6

Conclusion

This document has attempted to provide analysts with sufficient detail on the theory, implementation, discretization, application, and verification of a finite-deformation J -integral capability in Sierra. Fundamentally, the surface integral that embodies the driving force has been converted to a domain integral for increased accuracy and ease of computation. We have also added a pressure correction to the domain integral that enables path-independent results for tractions on the body and/or crack faces. Temperature gradients are not currently “corrected” and future work can remedy this issue. Working with Cubit developers, we were able to create the `snap_to_normal` option that enables orthogonal meshes along the crack front. While we believe this option is helpful for numerous situations, an orthogonal mesh greatly increases the robustness and accuracy of the current J -integral implementation.

A set of examples with analytical solutions (via linear elastic fracture mechanics) provides verification for straight, circular, and elliptical crack fronts. We also explored the increasing complexity of curved crack surfaces and inelasticity through a cone-crack and a C-specimen geometry, respectively. For these cases, we compared to a commercial code to ensure that the solutions were smooth and representative. We note that the weight function in Abaqus mirrors the plateau weight function. For small strains, both Abaqus and Sierra (using the plateau weight function) yield comparable driving forces. Inaccuracies in the J -integral for the linear weight function stem from the fact that the domain integral is sampling both inaccurate and nonproportional fields. Greater accuracy is achieved through the plateau and plateau-ramp weight functions. The goal of this report is not to make recommendations for applying the J -integral to elastic-plastic fracture mechanics. Rather, we seek to examine the implemented infrastructure for numerous cases to ensure that the domain integral is ready for the production environment. We must aid analysts and implement additional element types beyond the uniform-gradient hexahedral element if the current J -integral implementation will be widely used for J_2 plasticity and the accompanying isochoric motions.

Finally, we note that we have applied the current methodology to more complicated geometries not represented in this report. Although these varying geometries have aided our debugging efforts, included all the relevant cases is beyond the scope of this document. The team is not currently aware of any issues with the implementation. This document attempts to illustrate that verification. Generalizing and improving the capability will be the subject of future reports.

References

- [1] T.L. Anderson. *Fracture Mechanics, Fundamentals and Applications*. CRC Press, Boston, 1991.
- [2] Broek D. *Elementary Engineering Fracture Mechanics*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1986.
- [3] J.D. Eshelby. The force on an elastic singularity. *Philosophical Transactions of the Royal Society of London*, A244:87–112, 1951.
- [4] J.D. Eshelby. *Inelastic behavior of solids*. McGraw-Hill, New York, 1970.
- [5] HKS. *ABAQUS Version 6.7, Theory Manual*. Hibbitt, Karlsson and Sorensen, Providence, RI, 2007.
- [6] G.R. Irwin. Analysis of stresses and strains near the end of a crack traversing a plate. *Journal of Applied Mechanics*, 24:361–364, 1957.
- [7] M.F. Kanninen and C.H. Popelar. *Advanced Fracture Mechanics*. Oxford University Press, New York, 1985.
- [8] Y. Kim, Y.J. Chao, M.J. Pechersky, and M.J. Morgan. C-specimen fracture toughness testing: Effect of side grooves and η factor. *Journal of Pressure Vessel Technology*, 126:293–299, 2004.
- [9] B.R. Lawn. *Fracture of Brittle Solids*. Cambridge University Press, Cambridge, 1986.
- [10] F.Z. Li, C.F. Shih, and A. Needleman. A comparison of methods for calculating energy release rates. *Engineering Fracture Mechanics*, 21:405–421, 1985.
- [11] G.A. Maugin. *Material Inhomogeneities in Elasticity*. Chapman & Hall/CRC, New York, 1993.
- [12] J.R. Rice. A path independent integral and the approximate analysis of stress concentration by notches and cracks. *Journal of Applied Mechanics*, 35:379–386, 1968.
- [13] C.F. Shih, B. Moran, and T. Nakamura. Energy release rate along a three-dimensional crack front in a thermally stressed body. *International Journal of Fracture*, 30:79–102, 1986.
- [14] H. Tada, P.C. Paris, and G.R. Irwin. *The Stress Analysis of Cracks Handbook*. Del Research Corporation, Hellertown, PA, 1973.

DISTRIBUTION:

1 MS 0899 Technical Library, 8944 (electronic copy)