Automatic 3D Crack Placement using the Python API in Abaqus CAE

James C. Sobotka and R. Craig McClung

Southwest Research Institute

***Abstract**: This paper illustrates automated capabilities using the Python API in Abaqus/CAE to build, execute, and process complicated geometries with 3D semi-elliptical crack fronts suitable for linear-elastic analyses. These Python scripts adopt the same workflow used in Abaqus/CAE, which enables rapid debugging and post-analysis modifications through the GUI if necessary. These scripts are fully parameterized and support control over a crack's size, its shape, its location, and its discretization, e.g., the number of elements along its front. Due to the complexity of the required meshes, these scripts produce highly detailed crack-front regions with hundreds of edges, faces, and cells. Furthermore, these scripts support post-processing of stress-intensity factors along the crack front and other quantities of interest.*

These scripts increase the efficiency of an analyst to develop a large series of analyses without sacrificing solution quality. This improved efficiency enables improved solution quality during model development: increased mesh refinement, better mesh control, and precise partitioning of the near-tip region. For example, high quality analyses may be generated in seconds, not hours. The efficiency gains from this approach are especially noticeable when using the scripting process for large batches of jobs.

In this paper, we demonstrate the fidelity of these scripts against various benchmark solutions – e.g., exact analytical solutions, Newman-Raju solutions, and weight function methods in NASGRO®. Furthermore, this paper showcases some of the recent applications of this technique: cracks in lugs, part-through elliptical cracks, and large-scale verification efforts.

***Keywords**: Aircraft, Fatigue, Fracture, LEFM, Scripting, Verification.*

1. Introduction

Over the past fifty years, the aerospace industry has adopted the damage tolerant approach to ensure safe operations of airframes and engines. The damage tolerant approach models crack growth under cyclic loads using linear elastic fracture mechanics (LEFM). LEFM links the applied loading to an applied stress intensity factor (SIF) via SIF solutions for an assumed crack geometry that's embedded in a structural component. While there are many SIF solutions for through-thickness cracks in idealized 2D geometries, there are fewer SIF solutions for 3D cracks in realistic 3D geometries. Consequently, the aerospace industry often applies conservative safety

factors to these SIF solutions, and these conservative safety factors unnecessarily reduce the lives of aerospace components, *i.e.,* the aerospace component is pulled from service even though there exists some non-trivial but non-quantifiable remaining life.

Consequently, the production of new SIF solutions remains an active area of research, and due to the complexity of these geometries, most of these solutions are calibrated by results from finite element analyses. For example, Abaqus supports several methods to compute SIF solutions for stationary and growing cracks, including the extended finite element method (Moës 1999) and the virtual crack closure technique (Shivakumar 1988). Abaqus also features many outputs that can be used to compute SIF's, including strain energy and crack front displacements.

In the present work, we focus on SIF's computed by domain integral methods (Nikishkov 1987) that are supported directly in Abaqus/Standard. The domain integral method requires a specialized mesh around the crack front as illustrated in Figure 1. In the remainder of this work, we will refer to this region as the "crack tube". In the crack tube, the crack front has a semi-elliptical shape that is consistent with physical observations during fatigue crack growth processes. Every segment along this semi-elliptical crack front has been surrounded by 20-node hex elements with one collapsed face and mid-side nodes placed at the quarter-point location to produce the appropriate LEFM singularity (Barsoum 1976). These near-front elements are then surrounded by ring layers of hex elements arranged in a semi-circular or circular pattern to support domain integral calculations. Within the crack tube, each ring of elements should be set up to provide a virtual crack extension, and elements edges should be normal to elliptical crack front, *i.e.,* the crack-advance direction. Subject to these constraints, the element should be as well-formed as possible to limit numerical issues.

Let us now try to construct the crack tube using Abaqus/CAE. First, sweep the cross-section (two quartered circles with coincident centers) around an elliptical path to form a crack tube part. This crack tube can then be used to divide the larger component into cells. In Abaqus/CAE, the near-front region needs to be meshed as a wedge region, and the surrounding elements can either be meshed using the HEX or the SWEEP commands. (Note: Abaqus/CAE automatically converts the wedge elements into brick elements with the appropriate collapsed face if the crack front has been defined in the interaction menu.) Abaqus/CAE constructs an appropriate mesh if the region is not too elliptical, but there are problems as the ellipse becomes more elongated. Figure 2 shows meshes produced for a 3:1 ellipse using the HEX and the SWEEP commands. Both meshes contain badly formed elements that trigger errors with Abaqus/Standard. Furthermore, element edges in both meshes are not normal to the direction of crack advance. We suspect that these errors arise since the meshing commands attempt to produce good quality elements in the surrounding regions that are not good quality elements for a crack front calculation. In any case, these meshes are inappropriate for SIF calculations.

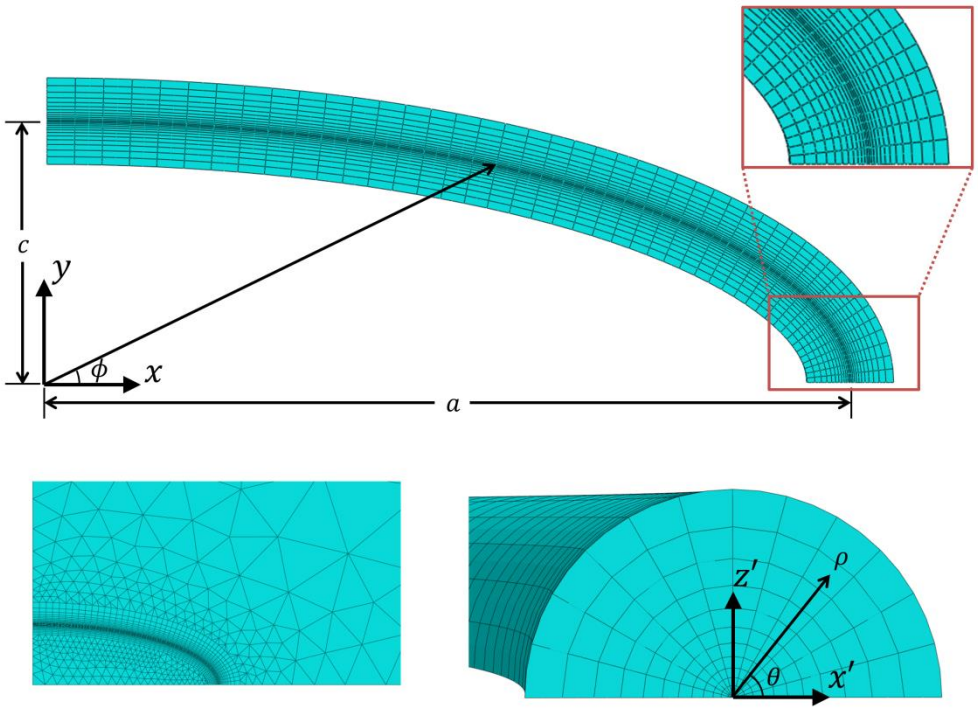

Figure 1. Crack tube region with dimensions and its location within the finite element mesh. Note lines of crack advance remain straight in this figure. The crack tube region is represented as a semi-circle for clarity.

The issues outlined in the previous paragraph prevent good quality meshes suitable for SIF calculations from being easily created with Abaqus/CAE. As a result several groups have produced software packages to create meshes suitable for crack front calculations, including FEACrack™, ZenCrack, FRANC3D, and Beasy. These software packages create an orphan mesh in the fracture critical region outside of the Abaqus/CAE framework. Consequently, they cannot take full advantage of the capabilities embedded in Abaqus/CAE.

In this work, we describe a new Python scripting capability to produce 3D crack front meshes directly using the Abaqus/CAE Python API. These commands enable post-processing of the domain integral results to external ASCII text files for storage and further processing. These scripts are robust and suitable for parametric studies of hundreds-to-thousands of 3D crack front fronts. Results from these scripts have been verified against exact analytical solutions, legacy aerospace solutions, and solutions found in NASGRO® and DARWIN®. Furthermore, these scripts support new SIF solutions for 3D cracks in lugs and part-through elliptical crack fronts.

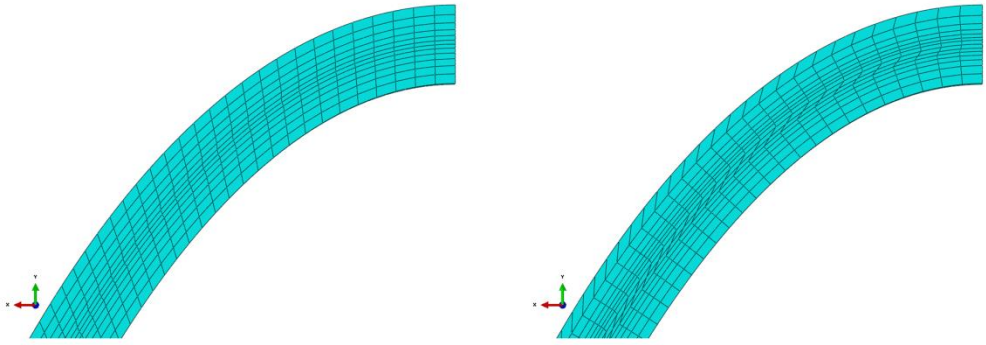

Figure 2. Crack tube regions produced using Abaqus commands on a swept elliptical cross-section using the HEX (left) and SWEEP (right) commands.

2. Scripting Details

This section describes key details of scripts that embed crack tube regions into 3D continuum bodies. These scripts have been developed as a series of Python libraries that are imported into other Python scripts that coordinate different aspects of the analysis, *i.e.*, these coordinating scripts create 3D geometric components, assign material properties, populate assemblies, set interactions, define boundary conditions, mesh the model, and output a *.inp file for Abaqus/Standard. Issues with these coordinating scripts will not be discussed since they are agnostic to the crack front creation. Furthermore, the crack front scripts discussed below could be integrated into Abaqus/CAE via a plug-in that would support direct user interaction through the GUI without the need for any coordinating scripts.

Consider a crack modeled as an ellipse that is arbitrarily located at the center of the $x-y$ plane as shown in Figure 1. A full ellipse represents an embedded crack front, and a truncated ellipse represents a crack with surface tips, *e.g.*, a corner crack or a surface crack. Any point on the ellipse can defined using the parameterization:

$$x = a\cos(\phi),$$

$$y = c\sin(\phi).$$

Here, a and c are dimensional terms that define the size and shape of the ellipse, and $\phi \in (-\pi, \pi]$ is a non-dimensional parameter that describes the location along the ellipse. It should not be treated as a physical angle.

Any point in the crack tube positioned on this ellipse may be defined in 3D Euclidian space by the following mapping along the crack front:

$$x = a\cos(\phi) + \rho\cos(\theta),$$

$$y = c\sin(\phi) + \rho\cos(\theta),$$

$$z = \rho\sin(\theta).$$

Here, (ρ, θ) define a circle centered on the elliptical crack front with $\theta \in (-\pi, \pi]$. (Alternatively, we might define a coordinate system $x' - z'$ centered at any point on the ellipse on shown in Figure 1. The radial coordinates of $x' - z'$ would then be (ρ, θ).)

This mapping is not unique at $\phi = arctan(-c/a)$ since the Jacobian determinant of the mapping equals $-\rho \times (c \times cos(\phi) + a \times sin(\phi))$. Instead, we construct a second mapping at discrete points, $\phi = \phi_i$, and linearly interpolate between these points to produce a unique mapping. These discrete points represent nodal locations along the crack front, and increasing the number of discrete points ϕ_i increases the mesh resolution along the crack front as a result.

The piecewise linear mapping remains unique for $0 < \rho < \rho_{\max}$. At $\rho = \rho_{\max}$, one or more points in Euclidian space becomes undefined for $|\theta| \geq \pi/2$. That is, multiple values of (θ, ϕ) converge to the same (x, y, z) location despite the piecewise linear mapping. Furthermore, finite precision for the computational geometry leads to a minimum value for the crack tube of, $\rho_{\min}$, where $\rho_{\min} = \rho_{\max}$ for most cases. These considerations limit the crack tube's radius to ρ_{mesh} with $\rho_{min} \leq \rho_{mesh} \leq \rho_{max}$.

Our script defines the value for ρ_{mesh} by starting with a large initial estimate on the order of the ellipse dimensions. It then reduces ρ_{mesh} to prevent self-intersection and to satisfy criteria that balance the element size defined along ϕ with the element size defined at $\rho \to \rho_{mesh}$. The script also throws an error if the size of ρ_{mesh} is below some fraction of the overall assembly's dimensions. This last criteria prevents elements from being too small to compute smooth and accurate SIF's across the crack front. Its value reflects prior experience.

Once ρ_{mesh} is defined, the script proceeds to build the 3D part that represents the crack tube region. The points $\phi = \phi_i$ define a piecewise linear curve that sets a sweep path. The script also defines coincident circular cross-sections that bound the near-front region (at some fraction of ρ_{mesh}) and the surrounding region at ρ_{mesh}. The circular cross-sections are further divided by a perpendicular cross that has one arm aligned with the crack face opening. The cross and circles are swept along the linearized ellipse to produce the crack tube.

Early versions of the script created a 3D solid part with this sweep operation, but it lead to random discontinuities at some values of ϕ_i in the swept part. These discontinuities prevented further meshing operations. Later, it was determined that sweeping a 3D shell did not lead to these discontinuities. If a 3D continuum region is needed, it can be created by capping the crack tube and converting the shell region to a solid region.

After the sweep operation, the script has produced a crack tube with the appropriate divisions for elements aligned with (ρ,θ), but elements aligned with ϕ require further partitions to enforce the piecewise linear mapping between ϕ_i. Consequently, the script partitions the crack tube at every ϕ_i to produce edges normal to the crack front. During this process, the script assigns datum planes by three vertices that are approximately located at a value for ϕ_i. These datum planes align normal vectors generated by the crack front that is defined by the computational geometry. These crack front vectors includes the finite precision effects. The script then partitions the crack tube with these datum planes. These partitions produce to the final crack tube geometry.

The crack tube geometry is a 3D shell part that may be imported into an assembly, aligned with a larger component, and then used to partition to the larger 3D solid component. Prior to the partitioning operation, the crack tube should be aligned with the component and completely constrained by the appropriate geometric constraint options. Otherwise, the partitioning operation may lead to a catastrophic error that causes script failure.

Once the crack tube has been positioned, scripting commands use the Python API to define the crack and assign output to it. These commands are covered in the Abaqus Python scripting reference manual (Abaqus 6.14-3), and the interested reader should refer to it. It should be noted that defining the crack involves defining the entire crack front from the linearized ellipse. This is most easily accomplished by finding one edge of the ellipse and incrementally joining edges to it that have the lowest angle.

Before meshing, it is beneficial to sort all edges of the crack tube into one of three categories that correspond to edges aligned with one of the crack tube coordinates: (ρ,θ,ϕ). These edges need to be seeded prior to meshing. For example, all of the edges connected by a single vertex to a vertex on the crack front are edges aligned with ρ. In the current scripts, these connectivity relationships define all crack tube edges without reference to their geometric locations as this approach reduces errors and increases efficiency. Cells should also be defined as either near-front cells or cells surrounding the near-front region that support domain integral calculations.

The scripts now perform several meshing operations, including:

- Assign wedge elements to the near-front cells and wedge elements to cells surrounding the near-front region. These elements should be quadratic elements with full integration.
- Set a constant mesh seed of one element on all edges aligned with ϕ.This divides each sector into a single slice of elements with dimensions set by the ϕ_i partitions.
- Set a constant mesh seed of some predefined number of elements on all edges aligned with θ. Typically, we set this number equal to 6, leading to a total of 24 elements around a fully circular crack tube.

- Set a constant mesh seed of one element on all edges aligned with ρ, if these edges are in the defined in the near-front cells. These elements will feature the desired square root singularity in the near-front region.
- Set a graded mesh seed on all other edges aligned with ρ. This mesh seed should increase towards the near-front cells. The number of elements is predefined and should equal or exceed the number of domains. The biasing accounts for the size the near-front elements and the average element along ϕ.

These scripts utilize a transition region that links the crack tube to the remainder of the structure. It would extremely difficult to mesh this transition region using hex elements. Instead, our script employs quadratic tetrahedral elements that are tied to the quadratic hex elements. The tie capability in Abaqus demonstrates good continuity of displacements, strains, and stresses if the elements are approximately the same size and have reasonable aspect ratios. However, the value for ρ_{mesh} may lead to elements where the element size along θ at ρ_{mesh} are smaller than the element size along ϕ. In this case, the element size mismatch may lead to oscillatory stresses along the crack front. When needed, our scripts place additional guide cells ahead of the crack front ($|\theta| \le \pi / 2$) where the mapping is always one-to-one and where there exist high stress gradients that might otherwise cause numerical difficulties. Regions behind the crack front have less impact on SIF results since they do not have high stress gradients.

The above text describes scripting features that build the finite element analysis. Once an analysis has been solved in Abaqus/Standard, we reload into Abaqus/CAE to extract the domain integral from the *.odb file. Abaqus stores this information as a time history dependent series divided by crack front nodes and domain integrals. This data storage method is not ideal for visualization of SIF's across the crack front. Consequently, the scripts interrogate the history information in the *.odb file to extract critical pieces of information at each time and each domain integral location (that corresponds to values of ϕ_i), including:

- Locations of the crack front in Euclidian and crack tube coordinates;
- SIF values for each domain integral along the crack front; and
- SIF values for each domain integral at important locations along the crack front, *e.g.*, the deepest crack tip.

This information is important to verify the results from the scripting capability. SIF's at each point along the crack front from each domain integral should converge to the same value (excluding the first domain integral that tends to accumulate numerical errors). For the same domain integral, SIF's along the crack front should not exhibit any non-physical oscillations from one ϕ_i to another. (This definition excludes mid-side nodes.) Jumps in the stress intensity factor values should be limited to surface regions where the assumed crack front shape may be incorrect due to corner point singularities (Pook 1994). The processed information is stored in ASCII text files that are suitable for further processing and long-term storage.

3. Discussion

The scripting capabilities described in the earlier sections have been evolving for more than three years. In their current form, they support the efficient modeling of crack fronts for LEFM analyses. They output via Abaqus input files that contain the following information:

- Crack tube geometries suitable for LEFM. This crack tube geometry can be easily inserted into rectangular plates and at circular holes.
- Crack definitions to compute domain integral quantities in Abaqus/Standard, including J-integrals and SIF's for Mode I, II, and III loading.
- History output definitions of the crack front results at discrete times corresponding to user-defined load steps.
- Sets and surfaces of all critical regions in the crack tube, *e.g.,* the near-front cells.
- Meshed crack front regions that are suitable for finite element analysis using Abaqus/Standard.
- Post-processed text files of SIF solutions.

The geometric operations described in the previous section represent a complex process to form a suitable crack tube region. The final crack tube geometry often has several hundred vertices, edges, faces, and cells to ensure a suitable crack tube mesh. The creation of these geometric entities may be traced through Abaqus/CAE, and they are tracked in the various sets and surfaces defined in the scripting process. Furthermore, users may interact with these geometric entities using Abaqus/CAE, not a third-party GUI.

There are several advantages to the scripting capabilities detailed in this work, including:

- *Efficiency*: Once these scripts are set up, complex crack front geometries can be built in only a few seconds as opposed to minutes-to-hours. Furthermore, error handling in Python reduces the amount of time that a user needs to interact with these scripts. In many cases, a full set of models may be built while the user performs other work.
- *Accuracy*: These scripts produce finite element geometries that represent our best-approximation of the crack front. Consequently, they support high-fidelity SIF solutions for damage tolerance analyses. As shown in the following section, they have excellent accuracy relative to analytical, legacy, and modern SIF solutions.
- *Completeness*: Previously, only a few crack front geometries were examined (*e.g.,* $a/c = 0.1, 0.5, 1, 2, 10$) due to high costs associated with building the geometries. Due to efficiency gains, these scripts support verification efforts over the complete SIF solution space, *e.g.,* any value of a/c between $0.1 \le a/c \le 10$. These gains support new verification efforts aimed to build credibility for the solutions.

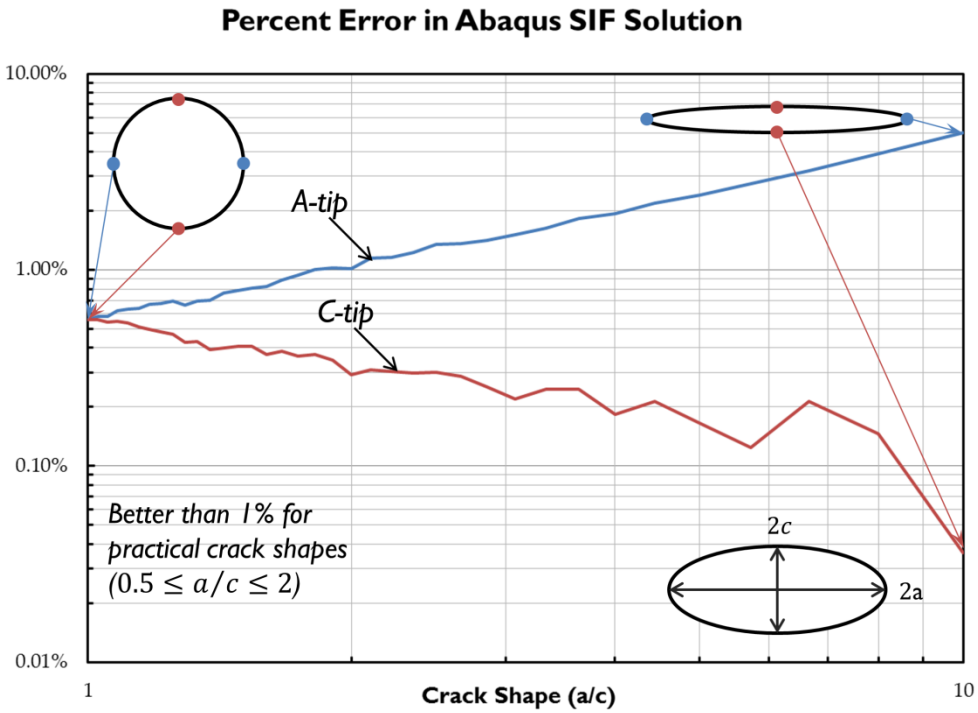

Figure 3. Verification of the scripting procedure against results from Irwin's exact solution for an elliptical crack front in an infinite body under uniform tension.

4. Verification

Figure 3 shows the percent error of these scripts vs. the exact analytical solution by Irwin (Tada 1973). His solution places an embedded elliptical crack in an infinite plate under tension loading. Our Abaqus solution fixes the number of ϕ aligned elements at 60 and sets the far boundary equal to 100 times the crack dimensions. We allow the crack shape to vary from a circle at $a/c=1$ to an elongated ellipse at $a/c=10$. (The inverse crack shapes (c/a) are covered due to symmetry in an infinite body.) At $a/c=1$, the a-tip and c-tip have the same levels of error since the crack is circular and the solution is constant. As a/c increases, the percent error increases at the sharp a-tip and decreases at the increasingly flat c-tip. This variation may arise from poor element aspect ratios at the elongated crack tip, and the error may be reduced by increasing the number of elements along the crack front. However, the magnitude of the SIF decreases at the elongated tip relative to the shallow tip. Consequently, large a/c ratios represent an unstable shape that the crack will grow out of with increased cyclic loading. The error remains less than 1% for the practical crack shapes $0.5 \le a/c \le 2$ normally encountered in life assessments.

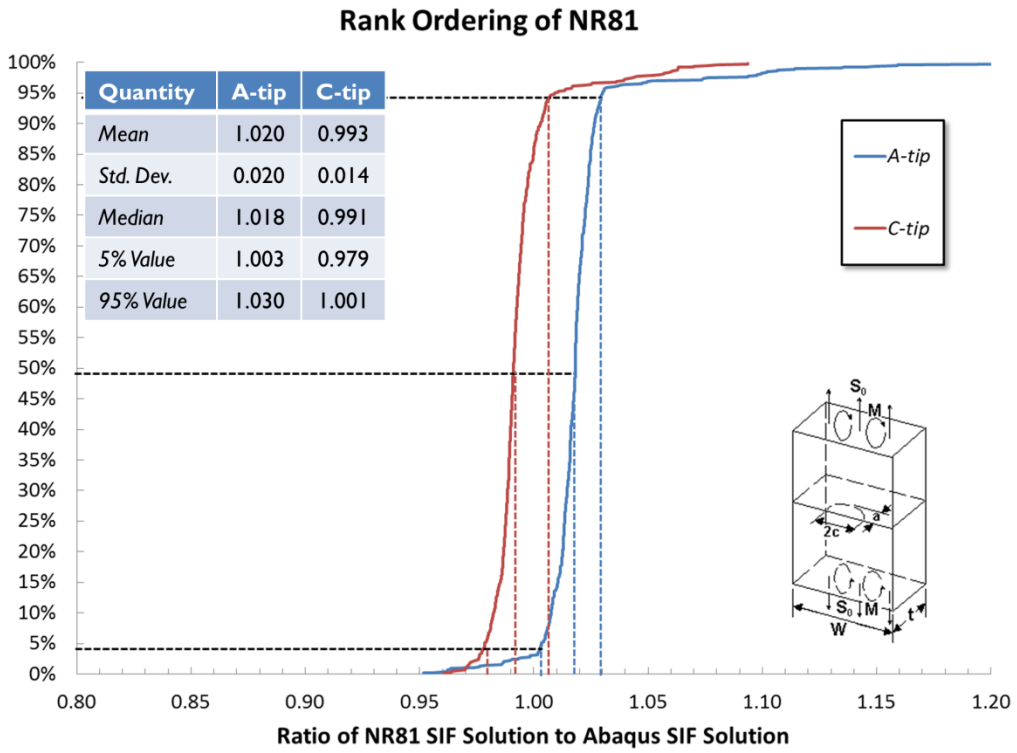

Figure 4. Verification of the scripting procedure against results from the legacy solutions of Newman and Raju.

Figure 4 presents a rank ordering of error, with error expressed as the ratio of the Newman-Raju solution (Newman 1981) to the Abaqus SIF calculation. The Newman-Raju solution provides SIF's for a surface crack centered in a plate under tension and bending loading. The Newman-Raju solution represents a legacy solution based on finite element analyses with limited degrees of freedom due to computational costs. Here, we selected 213 geometries from the solution space of the Newman-Raju solutions by Latin Hypercube Sampling, modeled them using the scripting process, and computed SIF's with Abaqus under remote tension and bending. More than 90% of the Abaqus SIF solutions are within 3% of the legacy SIF solution. Errors tend to accumulate for very short or very long crack tips; the Abaqus solution may be highlighting problems in the limiting cases of the Newman-Raju solutions.

These scripts have also been tested against SIF solutions in the current fatigue crack growth codes NASGRO and DARWIN. Specially, these codes support weight function solutions where the far-field loadings are input as univariant or bivariant stresses along the crack plane. Consequently,

they enable demonstrations of the scripting capabilities for loading cases beyond tension and bending, *e.g.*, quadratic stress gradients (McClung 2013). Furthermore, the weight function solutions were calibrated using independent numerical analyses (Lee 2008). For six major SIF solutions (CC09, CC11, SC30, SC31, EC04, and EC05), over 10,000 finite element analyses were constructed using the above scripting capabilities for a range of geometries and loading. This study (Sobotka 2018) reveals differences between Abaqus SIF calculations and the independent weight function SIF solutions usually less than 5%. Larger differences may develop for small cracks loaded by oscillatory stress fields that produce low magnitude SIFs. This result provides significant credibility to the SIF solutions from Abaqus, NASGRO, and DARWIN.

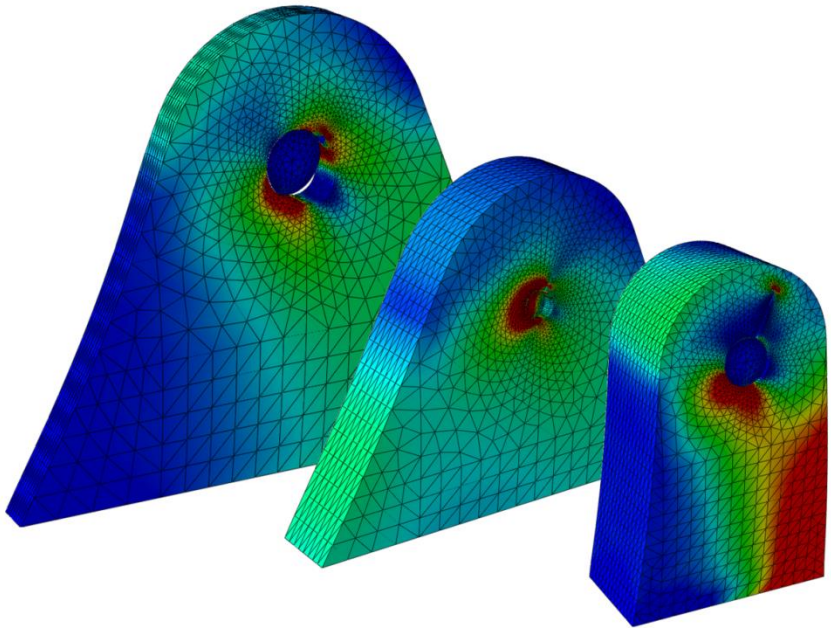

Figure 5. Corner cracks in obliquely loaded and tapered lugs under pin-loading.

5. Applications

Figure 5 shows a series of obliquely loaded and tapered lugs with corner cracks. In these solutions, the geometry and loading angle combine to define the crack initiation angle based on the maximum opening stress at the surface of the lug-to-pin interface. These solutions support crack growth perpendicular to this angle. Consequently, the scripting capabilities have been enhanced to orient the corner crack at any angle and to insert it at the cylindrical hole. This process required a remapping of the crack plane geometry to match the crack orientation plane defined by the maximum opening stress. As illustrated in Figure 5, finite element analyses employ general

contact between the deformable lug and deformable pin. Results from these analyses have been used to support new SIF's for tapered and obliquely loaded holes (Sobotka et al. 2016).

Figure 6 illustrates another series of new solutions derived using the scripting capability. These solutions focus on a part-through crack where the crack front is semi-elliptical, but it intersects the free surface at a non-normal angle. Such crack fronts may arise in thin sheets under an out-of-plane bending moment. To model these geometries, the scripts terminate the crack tube at some distance from the free surface; otherwise the crack tube over-extends itself beyond the free surface of the plate. Figure 6 shows the Mises stress variation from these analyses for a crack front. Stresses increase from point A to point B where the crack breaks through at a non-normal angle. These results show increased SIF's at point B rather than point A and indicate that the crack will try to straighten itself under the current loading. Results from these analyses support new SIF's using a weight function methodology.

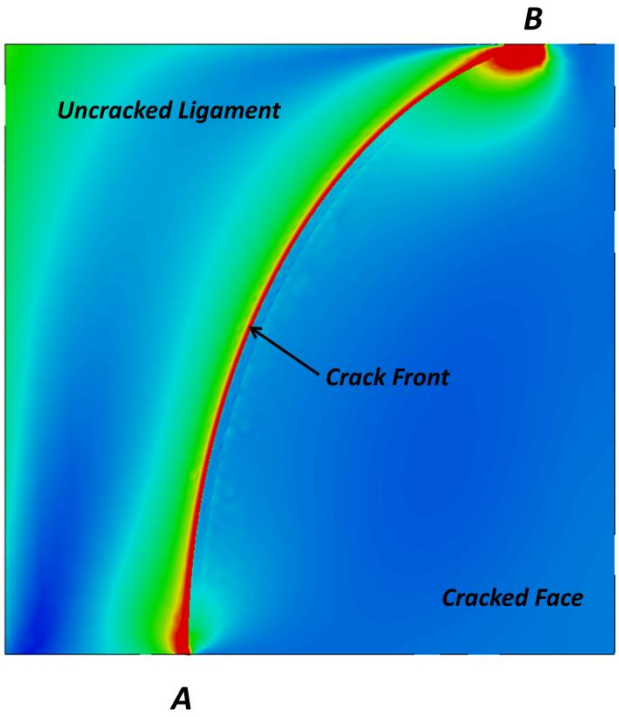

Figure 6. Part-through crack with a semi-elliptical crack front that does not intersect the free surface at a normal angle.

6. Summary

This paper describes new scripting capabilities that support the construction of finite element meshes with elliptical crack fronts. These scripts employ the Python API functions supported by Abaqus/CAE to build geometric models, to insert cracks within assemblies, to define crack fronts, to discretize regions, and to post-process results as ASCII text files. In these scripts, the basic geometry is represented as a piecewise linear ellipse surrounded by a circular crack tube that is suitable for domain integral calculations. This geometry produces a regular mesh region with increased refinement as the crack tip sharpens. Consequently, these scripts enable elliptical crack fronts with 10:1 ratios with a limited loss in accuracy. Verification studies indicate that crack fronts built using these scripts deviate from analytical, legacy, and current stress intensity factor solutions by no more than 5% for most solutions. Furthermore, these scripts enable the development of new stress intensity factor solutions for unique geometries, including cracks in obliquely loaded and tapered lugs and part-through cracks where the crack front intersects the surface at a non-normal angle.

7. References

1. Abaqus Users Manual, Version 6.14-3, Dassault Systémes Simulia Corp., Providence, RI.
2. Barsoum, R. S., "On the Use of Isoparametric Finite Elements in Linear Fracture Mechanics," International Journal for Numerical Methods in Engineering, vol. 10, 1976, pp. 25-37.
3. Lee, Y.-D., McClung, R. C., and Chell, G. G., "An Efficient Stress Intensity Factor Solution Scheme for Corner Cracks at Holes under Bivariant Stressing," Fatigue and Fracture of Engineering Materials and Structures, vol. 31, 2008, pp. 1004-1016.
4. McClung, R. C., Lee, Y.-D., Cardinal, J., and Guo, Y., "The Pursuit of K: Reflections on the Current State of the Art in Stress Intensity Factor Solutions for Practical Aerospace Applications," Proceedings of the 27th Symposium of the International Committee on Aeronautical Fatigue and Structural Integrity, Jerusalem, June 2013, pp. 361-375.
5. Moës, N., Dolbow, J., and Belytschko, T., "A Finite Element Method for Crack Growth without Remeshing," International Journal for Numerical Methods in Engineering, vol. 10, 1999, pp. 131-150.
6. Newman, J. C., and Raju, I.S., "An Empirical Stress-Intensity Factor Equation for the Surface Crack," Engineering Fracture Mechanics, vol. 16, 1981, pp. 185-192.
7. Nikishkov, G.P. and Atluri, S.N., "Calculation of Fracture Mechanics Parameters for an Arbitrary Three-Dimensional Crack, by the 'Equivalent Domain Integral' Method," International Journal for Numerical Methods in Engineering, vol. 24, 1987, pp. 1801-1821.
8. Pook, L.P., "Some Implications of Corner Point Singularities," Engineering Fracture Mechanics, vol. 48, 1994, pp. 367-378.

9. Shivakumar, K. N., Tan, P.W., and Newman, J. C., "A Virtual Crack-Closure Technique for Calculating Stress Intensity Factors for Cracked Three Dimensional Bodies," International Journal of Fracture, vol. 36, 1988, pp. R43-R50.

10. Sobotka, J.C. and McClung, R. C., "Uncertainty Quantification of Stress Intensity Factor Solutions," Submitted to Journal of Verification, Validation and Uncertainty Quantification, 2018.

11. Sobotka, J.C., Lee, Y.D., McClung, R. C., and Cardinal, J.C., "Modeling Cracks at Pin-Loaded Holes in Plates and Lugs: Assumptions, Verification, and Validation," Aircraft Airworthiness & Sustainment Conference, Grapevine, TX, USA, March 21-24, 2016.

12. Sobotka, J.C., Lee, Y.-D., McClung, R. C., and Cardinal, J.C., "Modeling Cracks in Obliquely Loaded and Tapered Lugs," Aircraft Structural Integrity Program (ASIP) Conference, San Antonio, TX, USA, November 28 – December 1, 2016.

13. Tada, H., Paris, P. C., and Irwin, G. R., "The Stress Analysis of Cracks Handbook," Del Research Corp., Hellertown, Pennsylvania, 1973.