ARL-TR-9733 ● AUG 2023

Correspondence of Measured and Simulated Melt Pool Geometries in Laser-Processed High-Strength Steel

by Stephen Cluff, Clara Mock, Brandon McWilliams, and Manoj Kumar Reddy Rangapuram

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9733 ● AUG 2023

Correspondence of Measured and Simulated Melt Pool Geometries in Laser-Processed High-Strength Steel

Stephen Cluff, Clara Mock, Brandon McWilliams
DEVCOM Army Research Laboratory

Manoj Kumar Reddy Rangapuram
Department of Mechanical and Aerospace Engineering,
Missouri University of Science and Technology

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
August 2023	Technical Report	September 2021	May 2023

4. TITLE AND SUBTITLE
Correspondence of Measured and Simulated Melt Pool Geometries in Laser-Processed High-Strength Steel

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
W911NF-20-2-0251		
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Stephen Cluff, Clara Mock, Brandon McWilliams, and Manoj Kumar Reddy Rangapuram

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-MD Aberdeen Proving Ground, MD 21005	ARL-TR-9733

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID IDs: Stephen Cluff, 0000-0001-8666-8244; Clara Mock, 0000-0001-8055-2371; Brandon McWilliams, 0000-0002-0494-3140

14. ABSTRACT
The ability to accurately simulate the thermal histories exhibited by directed energy deposition additive manufacturing processes is essential in the development of agile additive platforms that seek to serve Army needs. The current work presents a mesoscale finite element model that captures the distribution of temperatures formed from the laser processing of a high-strength steel. The model is used to simulate the thermal gradients surrounding the melt pool formed by the laser and is validated by comparison to experimental measurements of melt pool geometries. Comparisons made to cross-sectional and top-surface melt pool geometries captured by an in-situ welding camera show good agreement between measured and simulated melt pool geometries for various sets of process parameters. It is concluded that the model presented is capable of accurately capturing the thermal gradients immediately outside melt pools formed in high-strength steel, and for the calculation of part-scale thermal histories.

15. SUBJECT TERMS
additive manufacturing, finite element analysis, thermal history, melt pool, high-strength steel, directed energy deposition, Sciences of Extreme Materials

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	41

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Stephen Cluff	(410) 278-9913

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

To address the need for agile and cost-effective manufacturing platforms for Army materiel, additive manufacturing (AM) methods have been explored to build parts quickly and cheaply near the point of application. Additive methods demonstrate extreme flexibility in the geometries that can be created, with minimum dedicated tooling required to finish the part. Additionally, AM is a unique way to process engineering materials to create parts with unique structures and mechanical properties. These benefits can be leveraged to create parts with properties tailored for their specific end use. To take full advantage of these potential benefits, modeling and simulation can be used to gain insight and minimize the experimental costs of characterizing AM processes.

AM methods, such as directed energy deposition (DED) and powder-bed fusion (PBF) are techniques in which an energy source, most commonly a laser, provides heat for the melting and deposition of metal onto a substrate.[1–8] The laser forms a melt pool as the feedstock material is deposited, often in either powder or wire form. Material is added to the substrate layerwise to create a part in near net shape with minimal need for postprocess machining. As a part is built, material is often remelted or reheated as new layers are deposited. This results in a complex evolution of the temperature over time for each location in the part, referred to as the thermal history. This thermal history governs nearly every aspect of an additively manufactured part's formation and performance, including the development of microstructure, residual stresses, porosity, and various aspects of the part's final geometry.[1,3,4,6] As such, it is essential to have a means of both predicting and analyzing the thermal history as a function of input parameters (laser power, scan speed, etc.). Modeling and simulation provide such a means.

Many thermal history models exist in the literature. A full review of these is not presented here, but excellent review articles have been recently published.[1–10] Many different modeling approaches have been taken, each focusing on different physical phenomena or length scales. Models that consider part-scale phenomena usually neglect to consider smaller-scale phenomena, such as Marangoni flow within the melt pool or material vaporization.[11–14] In contrast, highly detailed models are severely limited in their scope due to their high computational complexity. A significant challenge in the simulation of AM is in capturing the effect of small-scale phenomena on larger-scale results, such as how the evolution of the size and shape of the melt pool affects overall part geometry. Mesoscale models seek to bridge this gap, incorporating smaller-scale phenomena into a scalable framework that can simulate larger material volumes over longer times.

The current work presents a custom mesoscale finite element (FE) thermal history model that seeks to capture the size and shape of melt pools formed in a high-strength steel. This steel, with the composition given in Table 1, is readily printable and capable of excellent static properties in the as-printed condition.[15–18] This material's microstructure is highly dependent on the thermal gradient, typically forming martensite under the extreme cooling rates.[18] As such, a melt pool scale thermal history model is essential to study the microstructural development of this material in the regions that immediately surround the melt pool. To that end, the present model is used to simulate the geometries and surrounding thermal gradients of melt pools formed in this steel by laser heating, demonstrating their dependence on processing parameters. The model is tuned and validated by comparison to experimentally observed melt pool geometries. Ultimately, the mesoscale nature of this model allows calculation of thermal histories at scales that drive the performance of additively manufactured high-strength steel parts.

Table 1 High-strength steel composition[15]

Element	C	Cr	Mo	Mn	V	Si	Ni	Fe
Wt.%	0.24–0.32	2–3	0.5–1.5	≤1	0.05–0.35	≤0.25	≤3	Balance

2. Methods

2.1 Experimental Procedure

For the current study, four different melt pool geometries were generated experimentally and subsequently modeled. All experiments were conducted in an RPM Innovations 222 DED system under an inert argon atmosphere. Each experiment consisted of a single line melted into a solid build plate by rastering the laser over its surface. A single build plate was used, with the same composition given in Table 1. The laser was used to form single scan lines 1.5 inches (38.1 mm) in length using the laser powers and scan speeds shown in Table 2. Each experiment was performed with no material deposition. Additionally, each scan was recorded by a Xiris XVC-1000 complementary metal oxide sensor (CMOS) camera mounted in the DED head. Raw camera output was written to file by Xiris WeldStudio software for later image analysis and comparison to simulated melt pool geometries.

After printing, the single laser track experiments were cut from the build plate, sectioned, polished, and etched to reveal the melt pool cross-sectional dimensions. Each cross section was polished using silicon carbide paper and 9-, 3-, and 1-μm diamond abrasive. Following polishing, each sample was etched using a 5% Nital solution (in methanol) for 30 s. Each cross section was imaged and measured using

a Keyence VHX-7000 optical microscope. The measured dimensions of each cross section were subsequently compared to simulated melt pool geometries.

Table 2 Experimental configurations of laser power and laser scan speed

Designation	Laser power W	Contour speed inch/min (mm/s)
450–30	450	30 (12.7)
450–40	450	40 (16.9)
650–30	650	30 (12.7)
650–40	650	40 (16.9)

2.2 Modeling Approach

2.2.1 Model Formulation and Boundary Conditions

The model presented here captures the flow of heat in conductive, convective, and radiative modes. The FE method is used to calculate the distribution of temperature throughout the build plate as the laser rasters over the surface. The FE model is implemented in the Abaqus 2020 solver, with custom-written user subroutines to govern heat input from the laser and boundary conditions. The conduction of heat is governed by the standard heat equation given in Eq. 1.

$$k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)+Q=\rho C\frac{\partial T}{\partial t} \tag{1}$$

The values of t, Q, and T are time, internal heat, and temperature, respectively. The material properties of k, ρ, and C are the thermal conductivity, density, and heat capacity, respectively; each are functions of temperature. Both convective and radiative heat transfer occur at the surface of the build plate and so are incorporated into the model via boundary conditions according to Eqs. 2 and 3, respectively.

$$q_c=-h_c(T_S-T_0) \tag{2}$$

$$q_r=-\varepsilon\sigma(T_S^4-T_0^4) \tag{3}$$

These are both functions of the surface temperature, T_S, and ambient temperature, T_0. The values of h_c, ε, and σ are the convection coefficient, material emissivity, and Stefan–Boltzmann constant. In the current model, the emissivity value is assumed to be $\varepsilon = 0.5$.

The FE geometry used for each simulation is shown in Fig. 1. A portion of the build plate is simulated with dimensions of 50 mm in length, 20 mm in width, and 6 mm in depth. Room-temperature boundary conditions are imposed on the outside and bottom surfaces, and convective and radiative boundary conditions are applied to

the top surface. The model's size was tested to ensure that predicted melt pool sizes were unaffected by proximity to the model's boundaries. Similarly, the mesh density was tested for convergence.

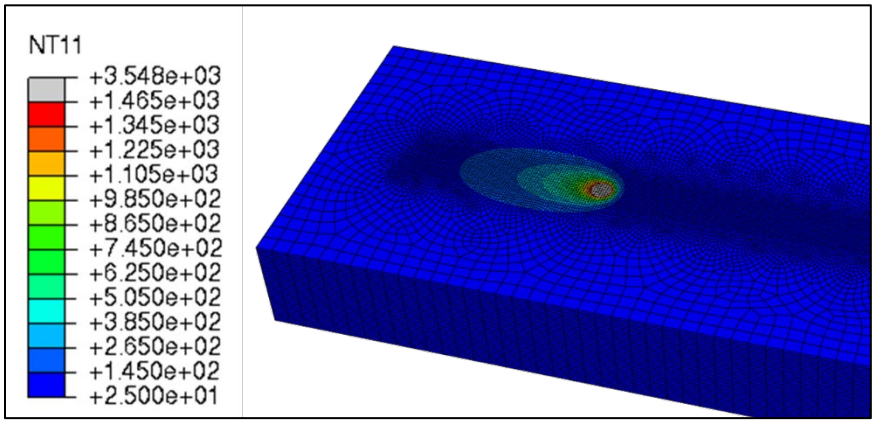

Fig. 1 The FE analysis (FEA) geometry used to calculate the melt pool shape. The temperature contour used to define the edge of the simulated melt pool is the average of the liquidus and solidus temperatures (1738 K).

Heat input from the laser is simulated as a 2-D Gaussian moving heat source and is implemented in a custom-written user subroutine (see Appendixes A and B for more information). The function used to define the distribution of heat applied to the surface of the model is given by Eq. 4, where the coordinates x and y define the local position relative to the laser's center. The laser power P and material absorptivity α define the total energy entering the part in units of power. The standard deviations σ_x and σ_y define the distribution of that energy over the beam's cross section. The values of both standard deviations were measured to be $\sigma_x = \sigma_y = 0.2058$ mm, using calibration data provided for the laser used. The absorptivity is used as a tuning parameter for this model and is adjusted to bring simulated and experimental melt pool dimensions into agreement.

$$F(x,y) = \frac{\alpha P}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} \tag{4}$$

Powder-fed DED requires the use of an inert carrier gas to deposit metal powder into the melt pool as well as a purge gas flow to keep the laser column clean of metal powder. This purge flow contributes to significant convective cooling that must be accounted for. Though no material was deposited in the single-track experiments, the purge flow was still used during each print. This flow can be treated as a single round nozzle jet stream impinging on a flat surface. The convection coefficient is estimated using (a) the known purge flow velocity of 19 m/s from a D = 6.86-mm-diameter nozzle at a height of H = 18.54 mm, (b) the

calculated Reynolds number of R_e = 5239, and (c) a Prandtl number of P_r = 0.7 for the Argon carrier gas. The experimental correlation proposed by Martin[19] for a single round nozzle impinging jet is used to calculate the average convection coefficient over a circular flat surface with an effective radius of r = 17.3 mm, resulting in a predicted value of $\overline{h_c}$ = 108 W/m^{2}K. The experimental correlation used to perform this calculation is given in Eq. 5, where the effective area ratio, A_r, is given by the relation $A_r = D^2/4r^2$.

$$\frac{\overline{h_c}D}{k_{Ar}P_r^{0.42}} = G[2R_e^{0.5}(1 + 0.005R_e^{0.55})^{0.5}]$$

where

$$G = 2A_r^{0.5}\frac{1-2.2A_r^{0.5}}{1+0.2\left(\frac{H}{D}-6\right)A_r^{0.5}} \quad (5)$$

2.2.2 Material Properties

The thermal material properties of high-strength steel must be considered functions of temperature to be accurate, given the large range of temperatures that every volume of material experiences during the model's run. These properties were generated by the JMatPro software package using the composition given in Table 1. The generated density, thermal conductivity, and heat capacity used by the current model are shown in Fig. 2. Latent heat of melting/solidification was included in this model but neglected for the face-centered cubic–body-centered cubic transition.

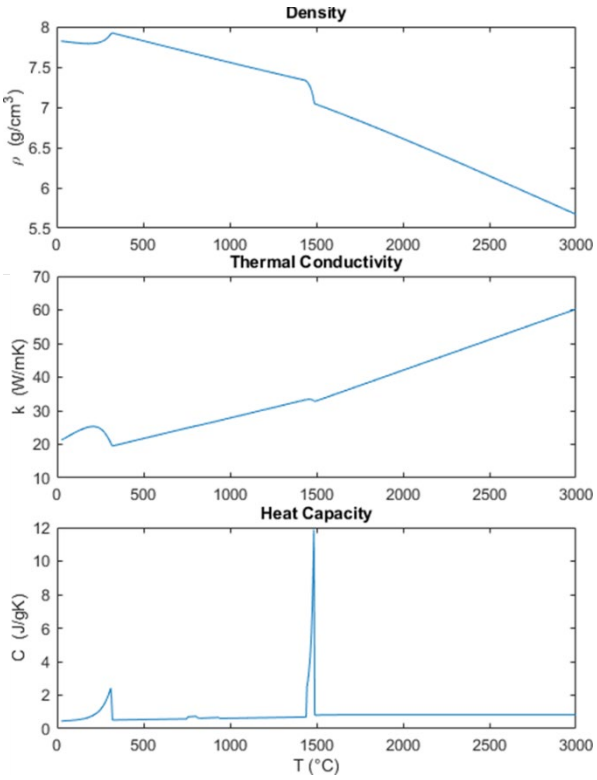

Fig. 2 Temperature-dependent properties as simulated by JMatPro

The extreme thermal gradients in the melt pool give rise to surface-tension-induced material flow, referred to as Marangoni flow.[2,11–14] A full simulation of that flow is not included in this model but is instead approximated using the anisotropic thermal conductivity method.[13,14] In this method, the increased rate of heat transfer due to the fluid motion within the melt pool is approximated by the linear scaling of the material's thermal conductivity for all regions of the model that are above the material's liquidus temperature. This allows the shape of the melt pool to account for some measure of material advection in the melt pool. For the current work, the adjustment of liquid thermal conductivities is done in an orthotropic manner according to Eq. 6, where the z direction is parallel to the laser and the x direction lies in the direction of the laser's travel. The correction factors N are taken here to be $N_x = N_z = 1$, while N_y (adjustment of the melt pool's width) is used as a tuning parameter. The final values of input parameters, and all other model inputs, are summarized in Table 3.

$$\begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} = k(T) \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} \qquad For\ all\ T > T_L \tag{6}$$

Table 3 Model input parameters

Model input parameters		
Laser power	P	450, 650 W
Laser scan speed	v	30, 40 inch/min
Absorptivity	α	0.3
Emissivity	ε	0.5
Convection coefficient	h_c	108 W/m^{2}K
Laser standard deviation	σ_x,σ_y	0.2058 mm
Conductivity correction factor	N_y	1.5
Thermal conductivity	$k(T)$	(see Fig. 2)
Material density	$\rho(T)$	(see Fig. 2)
Heat capacity	$C(T)$	(see Fig. 2)
Liquidus	T_L	1763 K
Solidus	T_S	1713 K
Ambient temperature	T_0	298 K

3. Results

3.1 Cross-sectional Geometry

Figure 3 shows the polished and etched melt pool cross sections formed during the single-track experiments, exposing the boundaries between melt pool (the innermost region), heat-affected zone (the brighter intermediate region), and unaffected base plate regions (the darker external region). Intuitively, the melt pool width and depth are shown to be proportional to laser power and inversely proportional to scan speed. These melt pool widths and depths are tabulated in Table 4.

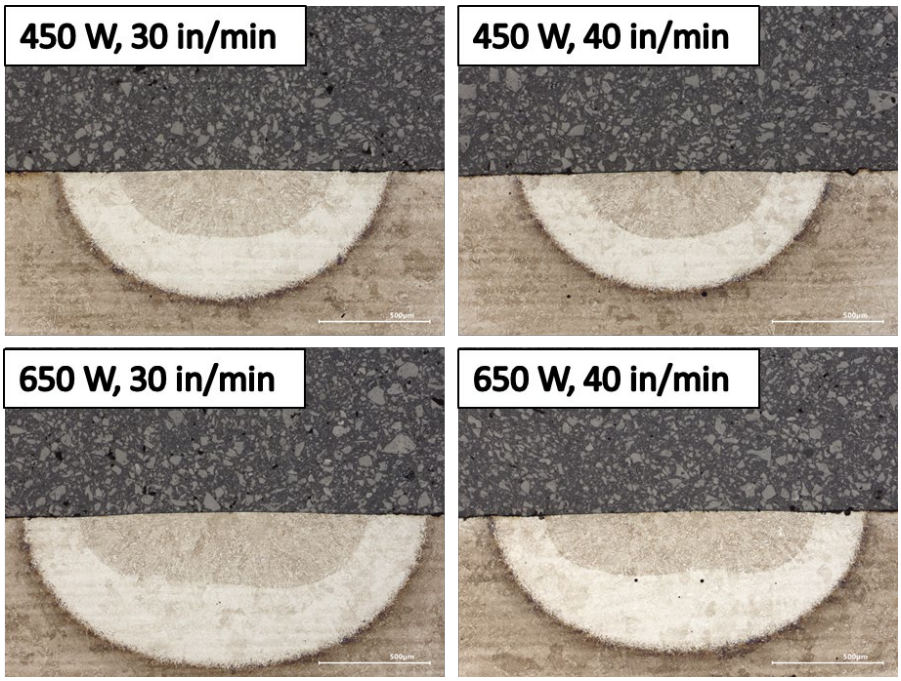

Fig. 3 Polished and etched cross sections of the single-track experiments, exposing the dimensions of the melt pool

Table 4 Measured dimensions of single-track melt pool cross sections

Single track (power–speed)	Melt pool width (µm)	Melt pool depth (µm)
450–30	965	300
450–40	880	295
650–30	1300	328
650–40	1218	273

3.2 CMOS Camera Imaging

Raw data from the CMOS camera was used to capture images such as that shown in Fig. 4. It is important to use the raw data from the camera as adjustments to image contrast, brightness, and gamma can obscure the underlying meaning of the image data. The camera's response is quantified by the output of a unitless number that is directly tied to the intensity of light absorbed by the camera's sensor. For this camera, the values reported lie within the range of 1800–3000. It is expected that for a given material system the camera's response would be consistent enough for a comparison of melt pool top-surface geometries formed under different conditions.

Single frames of camera data were captured at times that correspond to the output of a single time step in the FEA simulation. By using the known melt pool width from cross-sectional measurements, an intensity value can be chosen from the

image that corresponds with the melt pool's edge and its contour can be plotted. This was done for each experiment. All four experiments had a camera value at the expected melt pool width that was within 2% of 2400. The contour line corresponding with this value shows the top-surface geometry of the experimental melt pool at a single point in time.

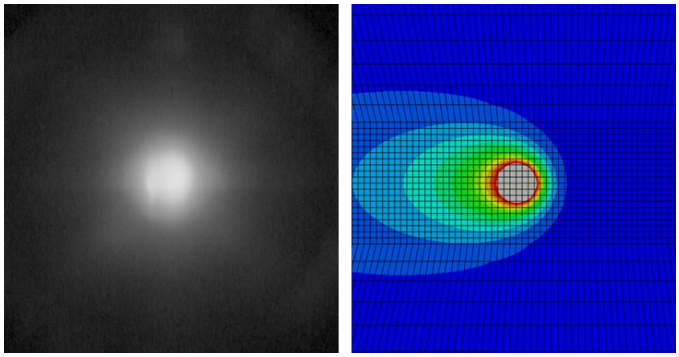

Fig. 4 Side-by-side comparison of experimental and simulated melt pool top surfaces for a single set of process parameters. The image gathered by in-situ CMOS camera (left) can be compared to the simulated melt pool (right) via the known melt pool width.

3.3 Comparison of Experiment and Simulation

By manually adjusting the values of the absorptivity α and the correction factor N_y, the simulated melt pools were brought into good agreement with experimental melt pool width and depth as measured by cross sectioning. Figure 5 shows how the simulated melt pool widths and depths compare to experimentally measured values. For all comparisons the boundary of the melt pool in the simulation is assumed to be at the temperature contour that corresponds to the average of the liquidus and solidus temperatures.

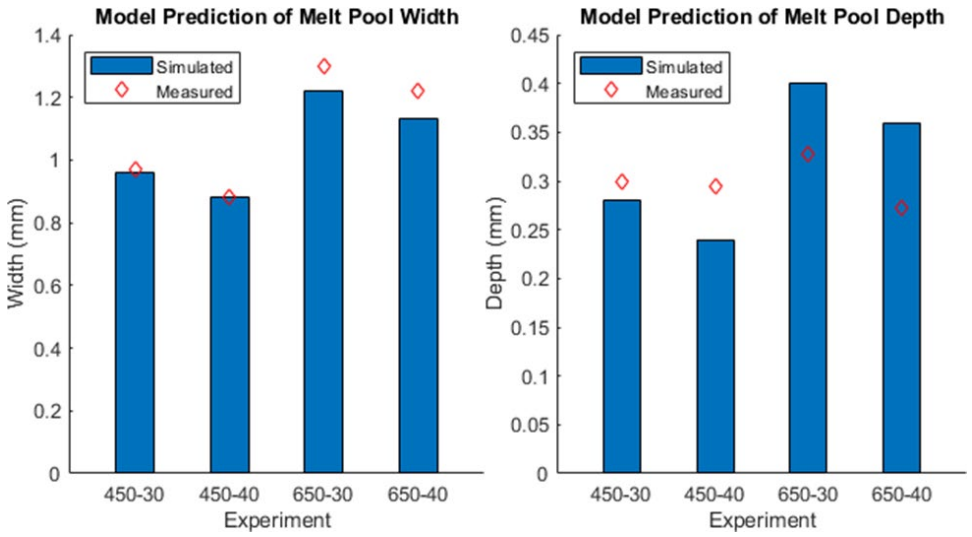

Fig. 5 Comparison of simulated and measured values for melt pool width and depth for the single-track experiments

There is excellent agreement between simulation and measurement for melt pool width, with more error seen in the comparison of melt pool depths. The simulations at the lower laser powers underpredicted the melt pool depths while the predicted depths were overestimated for the higher laser powers. The comparison of melt pool lengths is facilitated by the images gathered in-situ by the CMOS camera. Figure 6 shows the comparison of melt pool top-surface geometries as predicted by the model and as determined experimentally. The top-surface geometries show excellent agreement between simulation and experiment across all sets of process parameters. As expected, all results show a proportional relationship with laser power as well as a narrowing and lengthening of the melt pool with increasing contour speed.

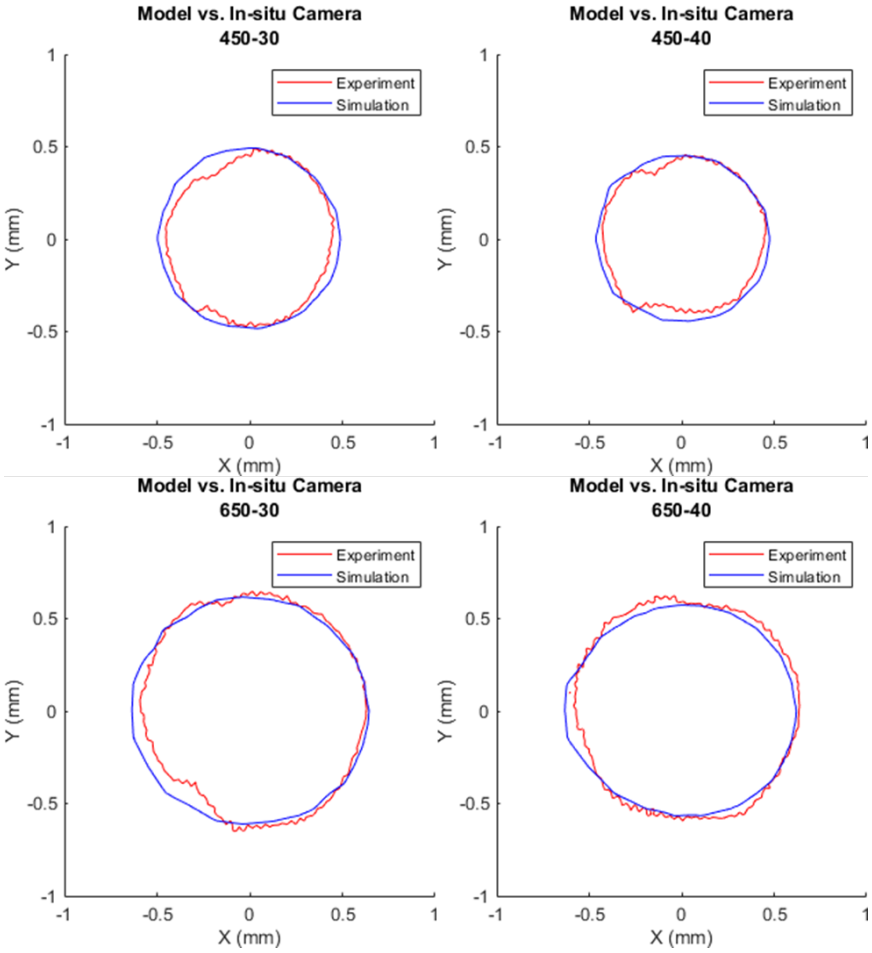

Fig. 6 Single-track melt pool top-surface geometry as predicted by simulation vs. experimental observation via in-situ camera measurement

4. Discussion and Conclusions

The excellent agreement between simulated and measured melt pool geometries suggests that the mesoscale model presented here successfully captures their dependence on the given process parameters for this steel. Capturing melt pool geometries with high fidelity is a necessary precursor to many other subsequent simulations at various length scales. Part scale simulations of thermal history, spanning larger domains in space and time, will naturally be sensitive to inaccuracies in the amount of heat applied to the model's surface. A small discrepancy can add up significantly over time, causing the unrealistic buildup of heat. Because the edge of the melt pool is treated as an isotherm, verifying the melt pool's geometry is accurately captured in simulation ensures that realistic amounts of heat are flowing from the melt pool's boundary into the solid material surrounding it. Even if the subsequent simulation does not seek to capture melt pool scale phenomena, these simulations provide valuable validation for the choice of material absorptivity and constants used to define radiative/convective boundary conditions. Similarly, a realistic representation of the melt pool's geometry also provides a validation of the thermal gradients in the surrounding solid material. These thermal gradients are essential for the subsequent simulation of microstructural development and evolution.

The discrepancies seen in the prediction of melt pool depth, though not severe, should be examined. These discrepancies are likely related to the simplifying assumptions that this model requires, specifically the single-value choices for material absorptivity and emissivity. These values are themselves functions of temperature, as well as several other factors (i.e., levels of surface oxidation, laser power, surface geometry, and so on). However, these values are very difficult to characterize and it is common practice to treat them as single values. Discrepancies in melt pool depth may also be related to the low number of measured melt pool cross sections. Since only a single set of cross sections were measured, there may be a statistical component not accounted for. Additionally, the cross-sectional images gathered for the higher power samples seemed to exhibit some instability at the melt pool's bottom—possibly due to the formation of porosity resulting from the higher energy density. Accounting for this may bring the predicted depths into better agreement with measured depths at the higher laser power.

The present results also demonstrate the value in using CMOS camera output to characterize melt pool top-surface geometries. For each experiment, the camera intensity value corresponding with the melt pool's edge was independently determined by comparison to cross-sectional measurements. Since all determined values were near each other, this suggests that it could be used as a consistent means

of determining the melt pool's edge under a variety of processing conditions. Since the chosen intensity is tied to a physically measured value, this would only be useful for extracting a single contour from camera images. Regardless, this could be a useful tool for observing the evolution of the melt pool's shape during a build.

In summary, the following conclusions can be drawn from the present results:

- The present model is capable of accurately simulating melt pool geometries in high-strength steel.
- The current model can be used to assess the effect of process parameters (laser power and contour speed) on melt pool size and shape.
- Top-surface melt pool geometry can be tracked in-situ using a CMOS welding camera, when properly calibrated for a given material system.
- By accurately representing the shape of the melt pool, the calculated thermal gradients surrounding the melt pool in the solid material can be used to characterize the evolution of the material's microstructure.

Future efforts to model AM processes will extend the current model into the calculation of part-scale thermal histories and the prediction of microstructural formation. Efforts will also extend applications of the current model into different AM methods, including the addition of material. As the thermal history drives nearly all aspects of an additively manufactured part's mechanical properties, this model will lay the groundwork for many future modeling and simulation efforts.

5. References

1. Bartlett JL, Li XD. An overview of residual stresses in metal powder bed fusion. Addit Manuf. 2019;27:131–149.

2. Cook PS, Murphy AB. Simulation of melt pool behaviour during additive manufacturing: underlying physics and progress. Addit Manuf. 2020;31:100909.

3. Francois MM, Sun A, King WE, Henson NJ, Tourret D, Bronkhorst CA, Carlson NN, Newman CK, Haut T, Bakosi J, et al. Modeling of additive manufacturing processes for metals: challenges and opportunities. Curr Opin Solid State Mater Sci. 2017;21(4):198–206.

4. Guan XY, Zhao YF. Modeling of the laser powder-based directed energy deposition process for additive manufacturing: a review. Int J Adv Manuf Technol. 2020;107(5–6):1959–1982.

5. Heigel JC, Michaleris P, Reutzel EW. Thermo-mechanical model development and validation of directed energy deposition additive manufacturing of Ti-6Al-4V. Addit Manuf. 2015;5:9–19.

6. Sun Z, Guo W, Li L. Numerical modelling of heat transfer, mass transport and microstructure formation in a high deposition rate laser directed energy deposition process. Addit Manuf. 2020;33:101175.

7. Yang QC, Zhang P, Cheng L, Min Z, Chyu MK, To AC. Finite element modeling and validation of thermomechanical behavior of Ti-6Al-4V in directed energy deposition additive manufacturing. Addit Manuf. 2016;12:169–177.

8. Zhang ZD, Huang Y, Kasinathan AR, Shahabad SI, Ali U, Mahmoodkhani Y, Toyserkani E. 3-Dimensional heat transfer modeling for laser powder-bed fusion additive manufacturing with volumetric heat sources based on varied thermal conductivity and absorptivity. Opt Laser Technol. 2019;109:297–312.

9. Goldak JA, Akhlaghi M. Computational welding mechanics. Springer Science & Business Media; 2005.

10. Rodgers TM, Moser D, Abdeljawad F, Underwood Jackson OK, Carroll JD, Jared BH, Bolintineanu DS, Mitchell JA, Madison JD . Simulation of powder bed metal additive manufacturing microstructures with coupled finite difference-Monte Carlo method. Addit Manuf. 2021;41:101953.

11. Heigel JC, Michaleris P, Palmer TA. Measurement of forced surface convection in directed energy deposition additive manufacturing. Proc Inst Mech Eng Part B–J Eng Manuf. 2016;230(7):1295–1308.

12. Ladani L, Romano J, Brindley W, Burlatsky S. Effective liquid conductivity for improved simulation of thermal transport in laser beam melting powder bed technology. Addit Manuf. 2017;14:13–23.

13. Nikam S, Wu H, Harkin R, Quinn J, Lupoi R, Yin S, McFadden S. On the application of the anisotropic enhanced thermal conductivity approach to thermal modelling of laser-based powder bed fusion processes. Addit Manuf. 2022;55:102870.

14. Nikam SH, Quinn J, McFadden S. A simplified thermal approximation method to include the effects of Marangoni convection in the melt pools of processes that involve moving point heat sources. Numer Heat Transf; A-Appl. 2021;79(7):537–552.

15. Agrawal P, Shukla S, Thapliyah S, Agrawal P, Nene SS, Mishra RS, McWilliams BA, Cho KC. Microstructure-property correlation in a laser powder bed fusion processed high-strength AF-9628 steel. Adv Eng Mater. 2021;23(1):2000845.

16. Kudzal AD, McWilliams BA, Taggart-Scarff J, Knezevic M. Fabrication of a low alloy ultra-high strength (> 1500 MPa yield) steel using powder bed fusion additive manufacturing. Mater Sci Eng; A-Struct Mater Properties Microst. 2020;770:138512.

17. Seede R, Shoukr D, Zhang B, Whitt A, Gibbons S, Flater P, Elwany A, Arroyave R, Karaman I. An ultra-high strength martensitic steel fabricated using selective laser melting additive manufacturing: densification, microstructure, and mechanical properties. Acta Materialia. 2020;186:199–214.

18. Seede R, Zhang B, Whitt A, Picak S, Gibbons S, Flater P, Elwany A, Arroyave R, Karaman I. Effect of heat treatments on the microstructure and mechanical properties of an ultra-high strength martensitic steel fabricated via laser powder bed fusion additive manufacturing. Addit Manuf. 2021;47:102255.

19. Martin H. Heat and mass transfer between impinging gas jets and solid surfaces. In: Hartnett JP, Irvine TF, editors. Advances in heat transfer. vol. 13. Elsevier; c1977. p. 1–60.

Appendix A. Custom Heat Source Definition

A.1 Implementation in Abaqus

The custom heat source presented here was implemented in the Abaqus 2020 finite element (FE) solver. The description of this model and its implementation assumes that the reader is familiar with the basic use of Abaqus, input file structure and syntax, user subroutines, table collections, and element/surface load types. The custom heat source uses the toolpath-mesh intersection utility module provided by Abaqus, which contains functions that can be called from user subroutines to define and track where a moving toolpath intersects model elements. Several aspects of this module are discussed here; for a complete description of the toolpath-mesh intersection module and any of the keywords/labels used here, see the product documentation for Abaqus.

The custom moving heat source is applied in Abaqus using the "*DFLUX" keyword in the model input file, with the "MBFNU" distributed flux type label. This signals Abaqus to call the user subroutines "UMDFLUXSETUP" at the beginning of each step/increment and "UMDFLUX" for each element during element calculations. Within these subroutines, the toolpath-mesh intersection module is used to query a given event series to calculate what elements are intersected by a moving virtual "tool." This tool has a prismatic shape, with the user providing the length, width, and height of the box. This box is also subdivided into prismatic segments. The centroid of each segment is a point for which element interactions are calculated; when the centroid of a segment passes through an element, that is treated as an event where heat is to be applied to that element. In short, each segment centroid is treated as a moving point heat source. The user provides the distribution of heat flux applied through the region of the moving tool via the UMDFLUX subroutine, and so each segment must apply at its center an amount of heat that equals the integral of heat flux over its own volume. The sum of heat flux applied by all segments then equals the total flux to be absorbed from the laser.

A general 3-D Gaussian heat source is implemented in the current model, defined by Eq. A-1.

$$f(x,y,z) = \frac{2\alpha P}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+\frac{z^2}{\sigma_z^2}\right)} \tag{A-1}$$

The integral of this function over the range spanned by a single segment $\{(x,y,z)|x_0 \le x \le x_1, y_0 \le y \le y_1, z_0 \le z \le z_1\}$ gives the total heat flux F to be applied at the centroid of that segment. The maximum value of z is expected to be

the top surface of the model where z is defined to be zero. The integral is then computed according to Eq. A-2.

$$F = \int_{z_0}^{z_1}\int_{y_0}^{y_1}\int_{x_0}^{x_1} f(x,y,z)\,dxdydz = \frac{\alpha P}{4}\cdot A\cdot B\cdot C$$

where

$$\begin{aligned} A &= ERF\left(\frac{x_1}{\sqrt{2}\sigma_x}\right) - ERF\left(\frac{x_0}{\sqrt{2}\sigma_x}\right) \\ B &= ERF\left(\frac{y_1}{\sqrt{2}\sigma_y}\right) - ERF\left(\frac{y_0}{\sqrt{2}\sigma_y}\right) \\ C &= ERF\left(\frac{z_1}{\sqrt{2}\sigma_z}\right) - ERF\left(\frac{z_0}{\sqrt{2}\sigma_z}\right) \end{aligned} \quad \text{(A-2)}$$

Depending on the density of the FE mesh, the number of subdivisions in the x, y, and z dimensions of the tool can be used to scale the resolution of the applied flux distribution. For example, if the tool were not subdivided at all, there would be only one segment and thus a single moving point heat source; the integral of the Gaussian function over its entire domain would be equal to one, and the total flux of the laser as defined by the power and absorptivity would be applied at that one point. This can be used to reduce the dimensionality of the Gaussian function from three to two—by not subdividing the box in the z direction, the distribution is reduced to a 2-D source applied to the top surface of the model, as represented in Eq. 4 in Section 2.2.1 of the report.

A.2 Input Files and Parameters

An example set of input files to run a simple model is provided in Appendix B. These files provide the main directives to run an Abaqus model via the hub input file (single_track.inp), and include the definition of material data, FE geometry, laser movement data, and so on. A description of each file and its format is given in Table A-1. The set of parameters that define the distribution of heat flux applied from the simulated laser are described in Table A-2.

Table A-1 Example model file contents

moving_source.inp	This is the main hub input file, from which each other input file is called. This file consists of the definitions of parameters, table collections, event series, and solution steps. For descriptions of parameters defining the moving heat source, see Table A-2.
custom_types.inp	This file contains the prototypes for the parameter tables defined in moving_source.inp.
material_data.inp	This file contains the material definition and thermal properties needed for a transient thermal FE analysis.
FE_geometry.inp	This file contains node, element, and set information. For 3-D transient thermal analysis, the element type should be DC3D8. This file also contains the section definition.
laser_event_series.inp	This file contains the event series that defines the motion of the laser. Each line of the file is a single event, consisting of five comma-separated values: time, *x* location, *y* location, *z* location, laser on/off (nonzero value defines laser as "on" from that time stamp to the next).
custom_laser_source.f	This file contains the custom code for the UMDFLUXSETUP and UMDFLUX user subroutines, which implement the heat source as described in Section A.1.

Table A-2 Parameters defining laser heat flux

BoxSubDiv1	Number of subdivisions in the toolbox local X direction (the local coordinate system is defined as having the *x* direction parallel to the tool's direction of motion, and the *z* direction parallel to the direction the laser is pointing).
BoxSubDiv2	Number of subdivisions in the toolbox local Y direction.
BoxSubDiv3	Number of subdivisions in the toolbox local Z direction.
Offset1	Offset in toolbox local X direction from toolpath position (the toolbox can be defined as having a translated position relative to the given event series). It is recommended to leave this at a value of 0.
Offset2	Offset in toolbox local Y direction from toolpath position. It is recommended to leave this at a value of 0.
Offset3	Offset in toolbox local Z direction from toolpath position. It is recommended to leave this at a value of 0.
BoxLength1	Size of tool in box local X direction.
BoxLength2	Size of tool in box local Y direction.
BoxLength3	Size of tool in box local Z direction.
LaserDirX	X component of vector defining laser direction, in global coordinate system.
LaserDirY	Y component of vector defining laser direction, in global coordinate system.
LaserDirZ	Z component of vector defining laser direction, in global coordinate system.
LaserPower	Laser power, in milliwatts.
sigmaX	Standard deviation of Gaussian heat source in X direction.
sigmaY	Standard deviation of Gaussian heat source in Y direction.
sigmaZ	Standard deviation of Gaussian heat source in Z direction.
emissivity	Material emissivity.
absorption	Material absorptivity.
convectionCoeff	Convection coefficient on all free surfaces for which no other boundary condition is applied, in units of mW/mm^2K.

When setting the size of the toolbox and number of subdivisions, be aware that if any point source (e.g., segment centroid) lies on an element boundary, Abaqus will apply that heat source to both elements, doubling the heat input. If possible, be sure that no point sources remain on any element boundaries.

To ensure a correct input of heat into the model, it is recommended to define the tool center position to be directly on the model's surface, with an odd number of box divisions in the *z* direction.

Appendix B. Example Input Source Code

B.1 Main Input File: moving_source.inp

```
*HEADING
** Melt pool scale simulation of single track to simulate melt pool dimensions
** under different process conditions. Used to tune input parameters
** (absorptivity, emissivity, and anisotropic thermal conductivity in melt pool)
** Process parameters:
** Laser power - 450 W, Contour speed - 30 in/min
**Units: <mm,N,tonne,s>
**-----------------------------------------------
**PARAMETERS
**-----------------------------------------------
*parameter
 AmbientTemperature        = 25
 absoluteZeroTemperature    = -273.15
 StefanBoltzmann           = 5.67e-11
** Unit: mW/(mm^2.C^4)
** -------------------------
**General laser parameters
** -------------------------
 BoxSubDiv1 = 10
 BoxSubDiv2 = 10
 BoxSubDiv3 = 1
 Offset1 = 0
 Offset2 = 0
 Offset3 = 0
 BoxLength1 = 1.49
 BoxLength2 = 1.49
 BoxLength3 = 1.5
 LaserDirX = 0
 LaserDirY = 0
 LaserDirZ = -1
 LaserPower = 450000
**-----------
**Gaussian laser parameters
**-----------
 sigmaX = 0.2058
 sigmaY = 0.2058
 sigmaZ = 0.2058
**
 emissivity = 0.5
 absorption = 0.3
 convectionCoeff = 108e-3
```

```
**-----------------------------------------------
*PHYSICAL CONSTANTS, ABSOLUTE ZERO=<absoluteZeroTemperature>,
 STEFAN BOLTZMANN=<StefanBoltzmann>
**-----------------------------------------------
*include, input=custom_types.inp
*include, input=material_data.inp
*include, input=FE_geometry.inp
**-----------------------------------------------
*TABLE COLLECTION, NAME="LaserParameters"
*PARAMETER TABLE,  TYPE="GeneralLaserParameters"
 "LaserPath_series","SimpleGaussian",<BoxSubDiv1>,<BoxSubDiv2>,
 <BoxSubDiv3>,<Offset1>,<Offset2>,<Offset3>,<BoxLength1>,
 <BoxLength2>,<BoxLength3>,<LaserDirX>,<LaserDirY>,<LaserDirZ>,
 <LaserPower>
*PARAMETER TABLE,  TYPE="SimpleGaussian"
 <sigmaX>,<sigmaY>,<sigmaZ>
*PARAMETER TABLE, TYPE="HeatTrans"
 <absorption>
**-----------------------------------------------
*Event Series, NAME="LaserPath_series",
           TYPE="LaserPath",
           TIME=TOTAL TIME,
           INPUT=laser_event_series.inp
**-----------------------------------------------
*Initial Conditions, Type=Temperature
 ALLNODES, 25.
**-----------------------------------------------
*Step, inc=1000000, name = Step-1
*Heat Transfer, Deltmx=10000
.05, 3.0, .01, .05
*DFLUX
 ALLELEM,MBFNU,,"LaserParameters"
*FILM
 ALLELEM,FFS,25.0,<convectionCoeff>
*RADIATE
 ALLELEM,RFS,25.0,<emissivity>
**
*BOUNDARY
 BOTTOMNODES, 11, 11, 25.
*OUTPUT,FIELD
*NODE OUTPUT
 NT
*END STEP
**
```

B.2 Table Types: custom_types.inp

```
**-----------------------------------------------
*EVENT SERIES TYPE, Fields=1, NAME="LaserPath"
 "Laser on or off"
**-----------------------------------------------
*PARAMETER TABLE TYPE,  NAME="GeneralLaserParameters",  PARAMETERS=15
 STRING, "Laser Event Series Name"
 STRING, "Laser Source Type Tophat or SimpleGaussian"
 INTEGER  , "BoxSubDiv1"
 INTEGER  , "BoxSubDiv2"
 INTEGER  , "BoxSubDiv3"
 FLOAT    , "Offset1"
 FLOAT    , "Offset2"
 FLOAT    , "Offset3"
 FLOAT    , "BoxLength1"
 FLOAT    , "BoxLength2"
 FLOAT    , "BoxLength3"
 FLOAT,  "LaserDirX"
 FLOAT,  "LaserDirY"
 FLOAT,  "LaserDirZ"
 FLOAT,  "LaserPower"
**
*PARAMETER TABLE TYPE, NAME="SimpleGaussian", PARAMETERS=3
 FLOAT,   "sigmaX"
 FLOAT,   "sigmaY"
 FLOAT,   "sigmaZ"
**
*PARAMETER TABLE TYPE, NAME="Tophat", PARAMETERS=1
 FLOAT,   "Radius"
**
*PARAMETER TABLE TYPE,  NAME="HeatTrans", PARAMETERS = 1
 FLOAT,  "Absorption Coefficient"
```

B.3 Material Input Data: material_data.inp

```
*Material,name=steel
*Conductivity, Type=ORTHO
21.2755,21.2755,21.2755,25
25.1234,25.1234,25.1234,180
19.7540,19.7540,19.7540,340
21.6789,21.6789,21.6789,495
23.6047,23.6047,23.6047,650
25.5354,25.5354,25.5354,810
```

27.4045,27.4045,27.4045,965
29.3295,29.3295,29.3295,1120
31.3166,31.3166,31.3166,1280
33.2416,33.2416,33.2416,1435
32.7618,32.7618,32.7618,1490
34.5796,51.8694,34.5796,1590
37.3972,56.0958,37.3972,1745
40.3058,60.4586,40.3058,1905
43.1234,64.6851,43.1234,2060
45.9410,68.9115,45.9410,2215
48.8495,73.2743,48.8495,2375
51.6672,77.5008,51.6672,2530
54.4848,81.7272,54.4848,2685
57.3933,86.0900,57.3933,2845
60.2110,90.3164,60.2110,3000
*Density
7.8220E-09,25
7.7907E-09,180
7.9125E-09,340
7.8279E-09,495
7.7444E-09,650
7.6578E-09,810
7.5748E-09,965
7.4949E-09,1120
7.4131E-09,1280
7.3346E-09,1435
6.9647E-09,1590
6.8351E-09,1745
6.6961E-09,1905
6.5573E-09,2060
6.4152E-09,2215
6.2660E-09,2375
6.1196E-09,2530
5.9722E-09,2685
5.8197E-09,2845
5.6721E-09,3000
*Specific Heat
4.54910E+08,25
8.25680E+08,3000
*Latent Heat
2.1799E+11,1440,1490

B.4 Model Finite Element Input Data: FE_geometry.inp

NOTE: This file contains the definition of all nodes, elements, and sets defined in the fine element analysis (FEA) geometry. It is too long to include in its entirety. For the purposes of this example, the FEA geometry assumed is a block of elements with side length 0.15 mm, and X,Y,Z dimensions of 39, 6, and 6 mm, respectively. All elements must be of type DC3D8 or DC3D6. The global origin is assumed to be at the center of the top surface of the block, with the X direction oriented along the long dimension of the block. The file must define the sets ALLNODES (all nodes in the model), ALLELEM (all elements in the model), and BOTTOMNODES (all nodes on the bottom and side surfaces of the model, excluding the top surface). The end of the file contains the following section assignments:

```
**
*Orientation, name=orthoCond
 1, 0, 0, 0, 1, 0
 1, 0
**
*Solid Section, elset=ALLELEM, orientation=orthoCond, material=steel
**
```

B.5 Laser Event Series Data: laser_event_series.inp

```
0,-19.05,0,0,1
3,19.05,0,0,0
```

B.6 Custom Heat Source Code: custom_laser_source.f

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    UMDFLUXSETUP
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      subroutine UMDfluxSetup(
     *                  lop,
     *                  cMBFnuElsetName,
     *                  jFluxActiveElems, nFluxActiveElems,
     *                  kStep, kInc, time, dt)
!
      include 'aba_param.inc'
      include 'aba_ptk_enums.inc'
      include 'aba_evs_param.inc'
#include <SMAAspUserSubroutines.hdr>
#include <PtkUtilitySubs.hdr>
C
C  ---------------------
C  VARIABLE DECLARATIONS
C  ---------------------

      parameter( j_lop_InitStep = 1, j_lop_InitIncrement = 2)
      parameter ( zero = 0.d0, one = 1.d0, two = 2.d0)
      parameter ( maxParams=50 )
```

```
      !UMDFLUXSETUP input parameters
      dimension time(2), dt(2), jFluxActiveElems(nFluxActiveElems)
      character*80 cMBFnuElsetName

      !Event series, laser source type, and distribution file names
      character*80 cLaserEventSeries, cLaserSourceType, cDistFilename

      !getParameterTable() interface
      character*80 cParameterTableTypeName, cParams(maxParams)
      dimension iParamDataTypes(maxParams), rParams(maxParams), iParams(maxParams)

      !Table array position parameters
      parameter(iLaserEventSeries = 1, iLaserSourceType = 2)
      parameter(iLaserBoxSubDivX = 3, iLaserBoxSubDivY = 4, iLaserBoxSubDivZ = 5)
      parameter(iLaserBoxOffSetX = 6, iLaserBoxOffSetY = 7, iLaserBoxOffSetZ = 8)
      parameter(iLaserBoxLengthX = 9, iLaserBoxLengthY = 10, iLaserBoxLengthZ = 11)
      parameter(iLaserVectorX = 12, iLaserVectorY = 13, iLaserVectorZ = 14)
      parameter(iRadius = 1)

      !PtkSetEventSeriesProperties() interface
      dimension jPtkEvtOps  (nPtkEventOpt_ArraySize)
      dimension jPtkEvtProps(nPtkIntEventProperties_ArraySize)
      dimension rPtkEvtProps(nPtkRealEventProperties_ArraySize)

      data jPtkEvtOps /nPtkEventOpt_ArraySize*zero/
      data rPtkEvtProps /nPtkRealEventProperties_ArraySize*zero/
      data jPtkEvtProps /nPtkIntEventProperties_ArraySize*zero/

      !getEventSeriesSliceProperties() interface
      character*80 cPROPSevs(n_evsC_size)
      dimension    iProps(n_evsI_size), rProps(n_evsR_size)

      !PtkGetNumIntersectedElements() and PtkGetIntersectedElements() interface:
      pointer(ptr_elemListHandler,elemListHandler)

C     ---------------
C     EXECUTABLE CODE
C     ---------------

      !Extract general parameters from Laser parameter table
      cParameterTableTypeName = 'GeneralLaserParameters'
      call getParameterTable(cParameterTableTypeName,
     *                  numParams, iParamDataTypes, iParams,
     *                  rParams, cParams, jError)

      cLaserEventSeries = cParams(iLaserEventSeries)
      cLaserSourceType = cParams(iLaserSourceType)

      iBoxSubDivX =  iParams(iLaserBoxSubDivX)
      iBoxSubDivY =  iParams(iLaserBoxSubDivY)
      iBoxSubDivZ =  iParams(iLaserBoxSubDivZ)

      rLaserBoxOffSetX =  rParams(iLaserBoxOffSetX)
      rLaserBoxOffSetY =  rParams(iLaserBoxOffSetY)
      rLaserBoxOffSetZ =  rParams(iLaserBoxOffSetZ)

      rLaserBoxLengthX =  rParams(iLaserBoxLengthX)
      rLaserBoxLengthY =  rParams(iLaserBoxLengthY)
      rLaserBoxLengthZ =  rParams(iLaserBoxLengthZ)

      LaserVecX = rParams(iLaserVectorX)
      LaserVecY = rParams(iLaserVectorY)
      LaserVecZ = rParams(iLaserVectorZ)

      rNumSubDiv = iBoxSubDivX * iBoxSubDivY * iBoxSubDivZ
```

```
      !If beginning of step, set laser distribution and event series properties
      !If beginning of increment, compute toolpath intersections
      if ( lop .eq. j_lop_InitStep) then

        !Setup laser distribution
        if (cLaserSourceType(1:14) .eq. 'SimpleGaussian') then

          !No setup needed here currently

        else if (cLaserSourceType(1:6) .eq. 'Tophat') then

          cParameterTableTypeName = 'Tophat'
          call getParameterTable(cParameterTableTypeName,
     *                  numParams, iParamDataTypes, iParams,
     *                  rParams, cParams, jError)

          rRadius = rParams(iRadius)
          rDiameter = two*rRadius

          if (iBoxSubDivZ.gt.one) then
            CALL STDB_ABQERR(-3,"Only 1 subdivision allowed in toolpath box Z direction for tophat distribution",0,0,0)
          end if

          if (rLaserBoxLengthX.lt.rDiameter .or. rLaserBoxLengthY.lt.rDiameter) then
            CALL STDB_ABQERR(-3,"Toolpath box X and Y dimensions must be at least the top hat beam diameter",0,0,0)
          end if

        end if

        !Setup heat source event series
        !Initialize toolpath-mesh intersection object
        call PtkSetMeshAndEventSeries(cMBFnuElsetName,cLaserEventSeries)

        !Set the option and property arrays to be passed into PtkSetEventSeriesProperties()
        iPtkEventType = iPtkEventSeriesShape_Box

        jPtkEvtOps(iPtkEventOpt_Algorithm) = iPtkEventSeriesAlgorithm_SubSegment

        rPtkEvtProps(iPtkRealEventProperties_BoxLength1)=rLaserBoxLengthX
        rPtkEvtProps(iPtkRealEventProperties_BoxLength2)=rLaserBoxLengthY
        rPtkEvtProps(iPtkRealEventProperties_BoxLength3)=rLaserBoxLengthZ

        rPtkEvtProps(iPtkRealEventProperties_Offset1)=rLaserBoxOffSetX
        rPtkEvtProps(iPtkRealEventProperties_Offset2)=rLaserBoxOffSetY
        rPtkEvtProps(iPtkRealEventProperties_Offset3)=rLaserBoxOffSetZ

        rPtkEvtProps(iPtkRealEventProperties_NormalVector1)=LaserVecX
        rPtkEvtProps(iPtkRealEventProperties_NormalVector2)=LaserVecY
        rPtkEvtProps(iPtkRealEventProperties_NormalVector3)=LaserVecZ

        jPtkEvtProps(iPtkIntEventProperties_BoxSubDiv1) = iBoxSubDivX
        jPtkEvtProps(iPtkIntEventProperties_BoxSubDiv2) = iBoxSubDivY
        jPtkEvtProps(iPtkIntEventProperties_BoxSubDiv3) = iBoxSubDivZ

        !Set event series properties by passing properties and options arrays into PtkSetEventSeriesProperties()
        call PtkSetEventSeriesProperties(cLaserEventSeries,
     *                  cLaserEventSeries,
     *                  iPtkEventType, jPtkEvtOps,
     *                  jPtkEvtProps, rPtkEvtProps)

      else if (lop .eq. j_lop_InitIncrement) then

        !Get event series slice properties
        call getEventSeriesSliceProperties(cLaserEventSeries, iProps, rProps, cPROPSevs)

        !Get the event start and end times from event series slice properties
```

```
        if(iProps(i_evsI_time) .eq. 1) then
          !step time
          tEnd = time(1)
        else
          !total time
          tEnd = time(2)
        end if

        tStart   = tEnd-dt(1)

        !Compute toolpath intersection
        call PtkCompute(cLaserEventSeries,tStart,tEnd)

        !Update nFluxActiveElems and jFluxActiveElems
        nFluxActiveElems  = 0
        ptr_elemListHandler = 0
        ptr_elemListHandler = PtkGetNumIntersectedElements(cLaserEventSeries,nFluxActiveElems)

        call PtkGetIntersectedElements(elemListHandler,jFluxActiveElems)

      end if

    return
    end

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    UMDFLUX
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    subroutine UMDflux(
   *      jFlags,
   *      amplitude,
   *      noel, nElemNodes, ielemNodes,
   *      mcrd, coordNodes, uNodes,
   *      kstep, kinc, time, dt, jltyp,
   *      Temp, nPredef, Predef,
   *      nsvars, svars, sol, dsol,
   *      nIntp,volElm,volInt,
   *      nHeatEvents, flux, dFluxdT,
   *      csiStart, csiEnd )
!
      include 'aba_param.inc'
      include 'aba_evs_param.inc'
      include 'aba_ptk_enums.inc'
#include <SMAAspUserSubroutines.hdr>
#include <PtkUtilitySubs.hdr>
C
C  ---------------------
C  VARIABLE DECLARATIONS
C  ---------------------

      parameter ( j_uti_True = 1, j_uti_False = 0 )
      parameter ( zero = 0.d0, one = 1.d0, two = 2.d0 )
      parameter ( maxParams=50, maxProps=10, maxEventNum=1000)

      !UMDFLUX input parameters
      dimension
   *  jFlags(2), ielemNodes(nElemNodes),
   *  coordNodes(mcrd,nElemNodes), uNodes(mcrd,nElemNodes),
   *  volInt(nintp),
   *  time(2), dt(2),
   *  sol(nElemNodes), dsol(nElemNodes),
   *  svars(2, nsvars,nElemNodes),
   *  Temp(2, nElemNodes), Predef(2,nPredef,nElemNodes)

      dimension
   *  flux(maxEventNum),
```

```
     *  dFluxdT(maxEventNum),
     *  csiStart(3,maxEventNum),
     *  csiEnd(3,maxEventNum)

        !Event series, laser source type, and distribution file names
        character*80 cLaserEventSeries, cLaserSourceType, cDistFilename

        !getParameterTable() interface
        character*80 cParameterTableTypeName, cParams(maxParams)
        dimension iParamDataTypes(maxParams), rParams(maxParams),
     *          iParams(maxParams)

        !Table array position parameters
        parameter(iLaserEventSeries = 1, iLaserSourceType = 2)
        parameter(iLaserBoxSubDivX = 3, iLaserBoxSubDivY = 4, iLaserBoxSubDivZ = 5)
        parameter(iLaserBoxOffSetX = 6, iLaserBoxOffSetY = 7, iLaserBoxOffSetZ = 8)
        parameter(iLaserBoxLengthX = 9, iLaserBoxLengthY = 10, iLaserBoxLengthZ = 11)
        parameter(iLaserPower = 15)
        parameter(iAbsorptionCoeff = 1)
        parameter(iSigmaX = 1, iSigmaY = 2, iSigmaZ = 3)
        parameter(iRadius = 1)

        !ptkGetDataAccess() interface
        pointer ( ptr_iEventRecordHandler,iEventRecordHandler)
        pointer ( ptr_eventRecord, eventRecord )
        pointer ( ptr_field, field )
        real*8 eventRecord(*), field(*)

C  ---------------
C  EXECUTABLE CODE
C  ---------------

        !Extract parameters from general Laser parameter table
        cParameterTableTypeName = 'GeneralLaserParameters'
        call getParameterTable(cParameterTableTypeName,
     *                  numParams, iParamDataTypes, iParams,
     *                  rParams, cParams, jError)

        cLaserEventSeries = cParams(iLaserEventSeries)
        cLaserSourceType = cParams(iLaserSourceType)

        iBoxSubDivX =  iParams(iLaserBoxSubDivX)
        iBoxSubDivY =  iParams(iLaserBoxSubDivY)
        iBoxSubDivZ =  iParams(iLaserBoxSubDivZ)

        rLaserBoxLengthX =  rParams(iLaserBoxLengthX)
        rLaserBoxLengthY =  rParams(iLaserBoxLengthY)
        rLaserBoxLengthZ =  rParams(iLaserBoxLengthZ)

        rSubBoxX = rLaserBoxLengthX/iBoxSubDivX
        rSubBoxY = rLaserBoxLengthY/iBoxSubDivY
        rSubBoxZ = rLaserBoxLengthZ/iBoxSubDivZ

        rLaserPower = rParams(iLaserPower)

        rNumSubDiv = iBoxSubDivX * iBoxSubDivY * iBoxSubDivZ

        !Check box divisions
        if((rLaserBoxLengthX.ne.zero .and. iBoxSubDivX.eq.zero).or.
     *    (rLaserBoxLengthY.ne.zero .and. iBoxSubDivY.eq.zero).or.
     *    (rLaserBoxLengthZ.ne.zero .and. iBoxSubDivZ.eq.zero)) then

          CALL STDB_ABQERR(-3,"Box cannot have zero subdivisions in any dimension",0,0,0)

        end if
```

```
      !Get event data for this element
      ptr_iEventRecordHandler = 0
      numRecords = 0
      nRecordLength = 0
      nFieldLength = 0
      jPtkEvtType = iPtkEventSeriesShape_Undefined
      jPtkAlgo = iPtkEventSeriesAlgorithm_Undefined
      ptr_iEventRecordHandler=PtkGetDataAccess(cLaserEventSeries,
     *                                         noel,
     *                                         numRecords,
     *                                         nRecordLength,
     *                                         nFieldLength,
     *                                         jPtkEvtType,
     *                                         jPtkAlgo)

      !If the number of events for this element exceed the maximum, raise error
      if(numRecords .gt. maxEventNum) then
        CALL STDB_ABQERR(-3,"Too many events for this element",0,0,0)
      end if

      !If there is an event for this element, continue
      if( numRecords.gt.0 ) then

        !Check Ptk geometry and algorithm
        if(jPtkEvtType.ne.iPtkEventSeriesShape_Box .or.
     *     jPtkAlgo.ne.iPtkEventSeriesAlgorithm_SubSegment) then

          call STDB_ABQERR(-3,"Incorrect Ptk shape or algorithm",0,0,0)

        end if

        !Extract absorptivity from parameter table
        absorption = 0
        cParameterTableTypeName = 'HeatTrans'
        call getParameterTable(cParameterTableTypeName,
     *                numParams, iParamDataTypes, iParams,
     *                rParams, cParams, jError)

        absorption = rParams(iAbsorptionCoeff)

        !Extract laser parameters from the appropriate table or dynamic array
        if (cLaserSourceType(1:14) .eq. 'SimpleGaussian') then

          rSigmaX = 0
          rSigmaY = 0
          rSigmaZ = 0
          cParameterTableTypeName = 'SimpleGaussian'
          call getParameterTable(cParameterTableTypeName,
     *                numParams, iParamDataTypes, iParams,
     *                rParams, cParams, jError)

          rSigmaX = rParams(iSigmaX)
          rSigmaY = rParams(iSigmaY)
          rSigmaZ = rParams(iSigmaZ)

        else if (cLaserSourceType(1:6) .eq. 'Tophat') then

          cParameterTableTypeName = 'Tophat'
          call getParameterTable(cParameterTableTypeName,
     *                numParams, iParamDataTypes, iParams,
     *                rParams, cParams, jError)

          rRadius = rParams(iRadius)

          numBoxes = 0.d0
          do ii = 1, iBoxSubDivX
```

```
      centerX = ii*(rSubBoxX/2) - (rLaserBoxLengthX/2.d0)
      do jj = 1, iBoxSubDivY
        centerY = jj*(rSubBoxY/2) - (rLaserBoxLengthY/2.d0)
        rDist = sqrt(centerX*centerX + centerY*centerY)
        if (rDist.le.rRadius) then
          numBoxes = numBoxes + 1.d0
        end if
      end do
    end do

end if

!Set up dynamic arrays and extract event data using PtkGetData()
id_eventRecord = 21
ptr_eventRecord = SMALocalFloatArrayCreate(id_eventRecord,nRecordLength,zero)

id_Field = 22
ptr_Field = SMALocalFloatArrayCreate(id_Field,nFieldLength,zero)

call PtkGetData(iEventRecordHandler, eventRecord, Field)

!For each event, calculate the amount of heat flux into the element
oneOverDt = 1.0/dt(1)
nHeatEvents = numRecords
do kEvent = 1, nHeatEvents

   !Set index for starting location within eventRecord array for this event
   iOffset   = (kEvent-1)*nPtkBoxSubSegmentEventRecord_Size

   !Get coordinates of current segment in local box coordinate system
   iXLocal = iOffset + iPtkBoxSubSegmentEventRecord_LocalSegCoord1
   xLocal = eventRecord(iXLocal)
   yLocal = eventRecord(iXLocal + 1)
   zLocal = eventRecord(iXLocal + 2)

   !Get event start and end times from event record
   startTime = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_timeStart)
   endTime = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_timeEnd)
   duration = endTime - startTime

   !Set starting and ending coordinates for this event, as required by UMDFLUX
   csiStart(1, kEvent) = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordStart1)
   csiStart(2, kEvent) = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordStart2)
   csiStart(3, kEvent) = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordStart3)
   csiEnd(1,kEvent)   = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordEnd1)
   csiEnd(2,kEvent)   = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordEnd2)
   csiEnd(3,kEvent)   = eventRecord(iOffset + iPtkBoxSubSegmentEventRecord_CoordEnd3)

   !Compute energy distribution factor
   distributionFactor = 0;
   if (zLocal.ge.0) then

      !Define bounds of integration for the current segment of the box
      rLoX = (xLocal - (rSubBoxX/2))
      rHiX = (xLocal + (rSubBoxX/2))
      rLoY = (yLocal - (rSubBoxY/2))
      rHiY = (yLocal + (rSubBoxY/2))
      rLoZ = (zLocal - (rSubBoxZ/2))
      rHiz = (zLocal + (rSubBoxZ/2))

      if (rLoZ.lt.0.0) then
        rLoZ = 0.0
      end if

      if( cLaserSourceType(1:14) .eq. 'SimpleGaussian' ) then
```

```
            !Integrate outside of the box in case a significant amount of the gaussian lies outside
            rBoxLoBoundX = (-1.0*(rLaserBoxLengthX/2.0))/(rSigmaX * sqrt(2.0))
            rBoxHiBoundX = ((rLaserBoxLengthX/2.0))/(rSigmaX * sqrt(2.0))
            rBoxLoBoundY = (-1.0*(rLaserBoxLengthY/2.0))/(rSigmaY * sqrt(2.0))
            rBoxHiBoundY = ((rLaserBoxLengthY/2.0))/(rSigmaY * sqrt(2.0))
            rBoxLoBoundZ = (-1.0*(rLaserBoxLengthZ/2.0))/(rSigmaZ * sqrt(2.0))
            rBoxHiBoundZ = ((rLaserBoxLengthZ/2.0))/(rSigmaZ * sqrt(2.0))

            rBoxErfX = 0.5*(ERF(rBoxHiBoundX) - ERF(rBoxLoBoundX))
            rBoxErfY = 0.5*(ERF(rBoxHiBoundY) - ERF(rBoxLoBoundY))
            rBoxErfZ = 0.5*(ERF(rBoxHiBoundZ) - ERF(rBoxLoBoundZ))

            rOutOfBox = 1.0 - (rBoxErfX*rBoxErfY*rBoxErfZ)
            rToAdd = rOutOfBox/rNumSubDiv

            !Closed-form integration of a 3D gaussian function over a rectangular domain
            rLoBoundX = rLoX/(rSigmaX * sqrt(2.0))
            rHiBoundX = rHiX/(rSigmaX * sqrt(2.0))
            rLoBoundY = rLoY/(rSigmaY * sqrt(2.0))
            rHiBoundY = rHiY/(rSigmaY * sqrt(2.0))
            rLoBoundZ = rLoZ/(rSigmaZ * sqrt(2.0))
            rHiBoundZ = rHiZ/(rSigmaZ * sqrt(2.0))

            rErfX = 0.5*(ERF(rHiBoundX) - ERF(rLoBoundX))
            rErfY = 0.5*(ERF(rHiBoundY) - ERF(rLoBoundY))
            rErfZ = 0.5*(ERF(rHiBoundZ) - ERF(rLoBoundZ))

            distributionFactor = 2.0*(rErfX*rErfY*rErfZ) + rToAdd

          else if (cLaserSourceType(1:6) .eq. 'Tophat') then

            rDist = sqrt(xLocal*xLocal + yLocal*yLocal)

            if (rDist.le.rRadius) then
              distributionFactor = 1.d0/numBoxes
            else
              distributionFactor = 0
            end if

          end if

        end if

        if (Field(kEvent).eq.zero) then
          laserOn = zero
        else if (Field(kEvent).ne.zero) then
          laserOn = one
        end if

        !Set total heat flux for this event
        totalPower = laserOn*absorption*rLaserPower*distributionFactor*duration*oneOverDt

        flux(kEvent) = totalPower
        dfluxdT(kEvent) = 0

      end do
    end if

  return
  end
```

List of Symbols, Abbreviations, and Acronyms

2-D	two-dimensional
3-D	three-dimensional
AM	additive manufacturing
CMOS	complementary metal oxide sensor
DED	directed energy deposition
FE	finite element
FEA	finite element analysis
PBF	powder-bed fusion

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

1 (PDF) DEVCOM ARL FCDD RLB CI TECH LIB

3 (PDF) DEVCOM ARL FCDD RLA MD S CLUFF B MCWILLIAMS C MOCK

1 (PDF) MISSOURI UNIV SCI TECHNOL MKR RANGAPURAM