The Pennsylvania State University

The Graduate School

Department of Mechanical and Nuclear Engineering

COST MODELING AND DESIGN TOOLS FOR ADDITIVE MANUFACTURING WITH LASER POWDER BED FUSION

A Thesis in

Mechanical Engineering

by

Michael W. Barclift

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2018

The thesis of Michael W. Barclift was reviewed and approved* by the following:

Timothy W. Simpson
Paul Morrow Professor of Engineering Design and Manufacturing
Thesis Advisor

Nicholas Meisel
Assistant Professor of Engineering Design

Karen A. Thole
Professor of Mechanical Engineering
Head of the Department of Mechanical and Nuclear Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Additive Manufacturing (AM) is a novel process that uses 3D model data to create complex geometry and functional structures by joining materials layer-by-layer. Despite growing interest, industry's adoption of AM has been limited due to challenges in cost-effectiveness, lack of subject matter expertise, and disparities amongst commercial software programs that capture the full breadth of design inputs and process considerations for AM. Designers have limited tools within the 3D modeling (e.g., CAD) environment that inform them on critical design parameters and tradeoffs that can impact the overall cost and feasibility of producing a part in AM.

In this thesis, cost modeling and design tools are examined for Laser-Powder Bed Fusion (LPBF). Traditional cost models have estimated that the material cost can range up to 46% of the total cost; however, these models have not accounted for the reuse (i.e., recycling) of the un-melted powder feedstock in LPBF. To capture susceptibilities to chemical contamination, diminished powder size distributions, and inconsistent mechanical performance, financial depreciation models using Sum-of-the-Years digits and Straight Line are implemented to define the value of a powder feedstock as function of each build cycle reuse in LPBF. A case-study is presented for an automotive upright designed for production and analyzed using a generic LPBF activity-based cost model. Sensitivity analysis revealed that traditional cost models assuming infinite material reuse undervalued the cost of build jobs with virgin powder by 3-11% or 13-75% depending on the material, feedstock price, and maximum permitted reuses in LPBF.

Cost modeling is iterative and estimates will vary as updates are made to the 3D model. To aid in informing designers on costs of their parts, a software plug-in is presented using the SolidWorks Application Programming Interface (API) that integrates the proposed LPBF cost model within the 3D CAD environment. The tool enables designers to generate support structures and distinguish from internal and external supports on their part. In addition to querying volume and surface data from the 3D model, the manipulation of the part's build orientation allows designers to concurrently estimate build time, feedstock requirements, and optimize parts for AM production while they are being designed in CAD. A case study is presented for an automotive upright where results found that varying the support angle by 15 degrees, underpredicted support structure volume by 34% and build time by 20%. Furthermore, poor packing of geometries on the build platform led to powder depreciation costs being nearly twice the material costs. Based on this two-part study, recommendations are made for additional research on LPBF cost modeling, post-processing cost modeling, powder feedstock reusability metrics, and CAD-integrated design tools with greater inputs, support structure libraries, and considerations for AM processes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Dr. Timothy Simpson for serving on my committee and for being my advisor throughout my time at Penn State. His advising style forced me to seek out answers for myself, explore new ways of thinking, and critically analyze conclusions. I appreciate his knowledge, encouragement, expertise, and deep care for his students. He is a true evangelist for the Additive Manufacturing and Design field, and has created a rich culture of exploring new ideas, challenging old conventions, and learning together with fellow students. Thank you, Dr. Nicholas Meisel for serving on my committee and mentoring me from our earliest days back at Virginia Tech. Even though I've known Dr. Meisel for many years, I always learn something new every time I see him. Thank you, Dr. Richard Martukanitz and Dr. Sanjay Joshi for your wide breadth of knowledge and profound insight on laser processing and additive manufacturing technologies. Additionally, I appreciate the staff at CIMP-3D for their openness, suggestions, and critiques which helped me develop my research. Specifically, I would like to recognize Corey Dickman, Ken Meinert, Abdalla Nassar, and Wes Mitchell. I would also like to acknowledge Eduard Romaguera and Andrew Armstrong for their contributions and support for this research. I am grateful for the members of EDOG, Made by Design Lab, Brite Lab, PSU 3D Printing Club, MEGA, and BGSA for the many great friendships formed during my time at the university. Lastly, I also want to thank my family for their endless love and support. Most importantly, I want to recognize my father, Wayne E. Barclift, who unfortunately passed away during my time in the program. He was one of the most selfless and whole-hearted people I knew, and he is one of the many reasons why I became an engineer. I hope to continue to grow as a person and live a life of character and principles just like he did. You will be missed. We will always love you.

Chapter 1

Additive Manufacturing Overview and Research Motivation

Additive Manufacturing (AM) is a process of “joining materials to make objects from 3D model data, usually layer upon layer” [1]. Beginning in 1984 with the patent of Vat Photopolymerization [2], AM has grown to seven modalities for producing a 3D object with each varying based on their feedstocks, energy sources, and processing techniques for joining each layer. Vat Photopolymerization (VP), Material Extrusion (ME), and Powder Bed Fusion (PBF) were first applied in “rapid prototyping” for the purpose of examining the “form and fit” of a product to aid in design communication and accelerate development cycles. Increased machine accuracy and reliability, along with the introduction of Sheet Lamination (SL), Binder Jetting (BJ), and Material Jetting (MJ), allowed AM to expand to “rapid tooling” for the production of expendable jigs and fixtures, along with the creation of low-cost patterns for casting and injection molding. Advancements in laser power density and repeatability in processing metals using PBF and Directed Energy Deposition (DED) have now led to AM becoming a promising technology for the “direct digital manufacturing” of functional engineering components. The customizability of the 3D model and the layer-by-layer fabrication of AM enables designers to explore complex geometries, multiple materials, mesostructures, functional features, and consolidated assemblies. AM provides a broader solution space and greater design freedom than traditional manufacturing. However, modern advancements in AM machines have outpaced the development of software, design guidelines, and regulatory standards resulting in limited expertise on the full capabilities, limitations, and appropriate applications of AM technology.

1.1 Brief History of Metal Additive Manufacturing

AM for direct metal components was pioneered in 1995 through DED at Sandia National Laboratories and later commercialized by Optomec as Laser-Engineering Shape (LENS). Their process consisted of propelling metal feedstock, via inert gas, into a melt pool generated by a fiber laser focused on a given location on a build substrate [2]. In the same era, BJ was being introduced by ExOne [2], where an adhesive agent was selectively deposited to bond layers of metal powder and form a “green part” before undergoing a post-build de-binding and sintering process to achieve full densification. Early studies in Laser PBF (LPBF) were conducted at General Electric by Carter and Jones [3] where they used a Nd:YAG laser to directly sinter a component made from an iron feedstock. However, they only achieved “35%” density for the as-built structure and required a post-build Hot-Isostatic Pressing (HIP) operation to achieve full density. Improvements in the LPBF process were achieved by Das *et al.* [4] where they achieved “98.5%” density in processing Ti-6Al-4V alloy and “99.5%” density after HIP. LPBF was most widely innovated through the efforts of the Fraunhofer Institute of Laser Technology and their developments in “Selective Laser Powder Remelting” (SLPR) shown in Figure 1-1 [5].

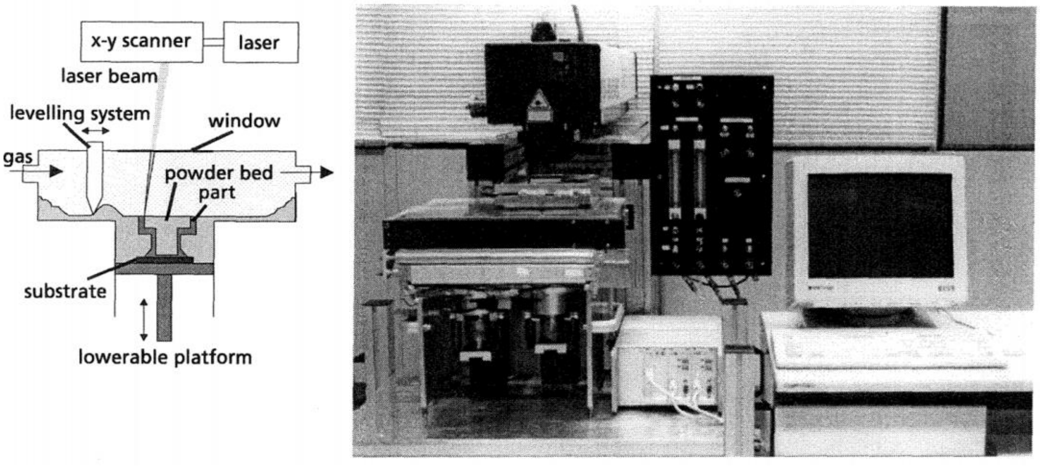


Figure 1-1: SLPR Prototype at Fraunhofer Institute of Laser Technology [5]

SLPR contributed to fundamental advancements in laser hatch/contour strategies, matching laser wavelengths to the absorptivity of the metal powder, minimizing beam distortion with f-theta lens, mitigating material oxidation through inert gas in the build chamber, and anchoring parts to a build substrate to minimize distortion in the AM process [2]. By the early 2000's, SLPR was commercialized and marketed by European manufacturers: MTT Technologies (now Renishaw), Concept Laser, Phenix (now 3D Systems), and EOS Gmbh. In 2001, Electron Beam Melting was also commercialized by Arcam [2], providing an alternative to LPBF as a technology that used propelled electrons, magnetic lens, and beam splitting to process metal parts.

1.2 Laser-Powder Bed Fusion

LPBF is an AM process where "thermal energy selectively fuses regions of a powder bed" to produce parts [2]. As shown in Figure 1-2, LPBF consists of a powder delivery system, where a feed bed (i.e., hopper, dispenser) supplies a layer of powder onto a part bed containing the build job geometries. Thermal energy from a laser scans the part bed surface and fully melts a region of powder particles to form a solidified cross-section. The powder bed lowers and a coating mechanism (i.e., blade, rake, roller) spreads additional powder from the feed bed on top of the scanned layer. Energy is applied to the newly recoated surface to solidify the next layer of the part, and the AM process repeats for each layer until all geometries have been fabricated. At the completion of the build job, parts are removed from the machine, while surrounding un-melted powder is recovered from the part bed and overflow bin for reuse in later builds. LPBF is commonly executed in a build chamber filled with inert gas (e.g., Argon, Nitrogen) or under vacuum to mitigate the reactivity of the powder feedstock during the melting of each cross-section. Technologies that offer LPBF are Direct Metal Laser Sintering (DMLS), Selective Laser

Melting (SLM), Direct Metal Laser Melting (DMLM), Direct Metal Printing (DMP), Laser Melting (LM), and LaserCUSING [6].

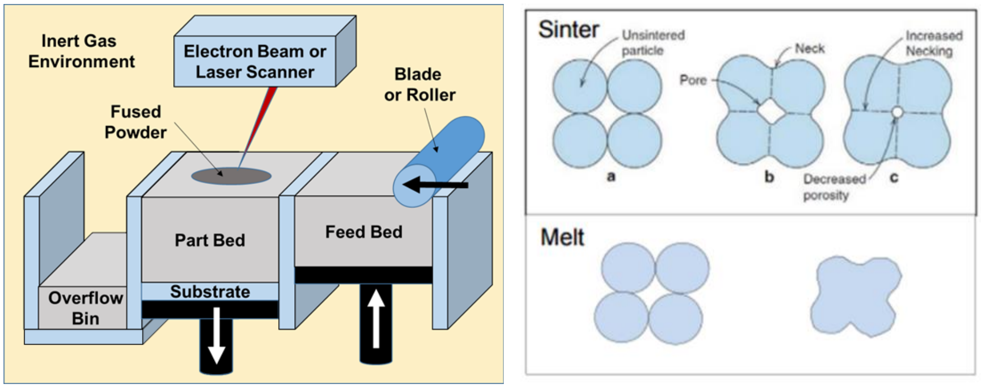


Figure 1-2: Powder Bed Fusion Schematic (left) and Fusing of Powder Particles (right) [2]

1.3 Design Considerations for Laser-Powder Bed Fusion

In comparison to polymeric AM processes, LPBF with metal powder feedstocks requires extensive operations in order to generate a fully-functional component. In LPBF, build orientation and support structure generation are non-trivial tasks with the goal of anchoring [7] the part to the build substrate. This design process is critical due to the risk of thermal distortion and mechanical delamination during the build which can lead to a part feature colliding with the recoater and subsequently failing the build [8]. In contrast, the heated powder bed in EBM mitigates thermal distortion; however, the design of support structures is aimed at dissipating heat and minimizing curling for fine features [9]. Depending on the application, LPBF parts may have internal voids (i.e., porosity) and residual stress, which require thermally post-processing via stress relief, heat treatment, annealing, and HIP, in order to meet microstructure and performance requirements [10]. Parts may also undergo mechanical post-processing via electrical discharge machining, shot-peening, milling, and manual support structure removal. Lastly, end-use parts may require non-destructive testing such as dye-penetrant, computed tomography, white light metrology, digital radiography, and CNC probing to validate the surface finish, external dimensions, and internal structure.

1.4 Motivation

Industry has shown a growing demand for LPBF and metal AM modalities because of the capabilities of producing components with complex shapes, functional features, lower part count, and minimal labor compared to traditional manufacturing [11]. The 2018 Wohler's Report found that "1768" metal AM machines were sold in 2017, marking a "80 %" increase in sales compared to 2016 as shown in Figure 1-3 [12].

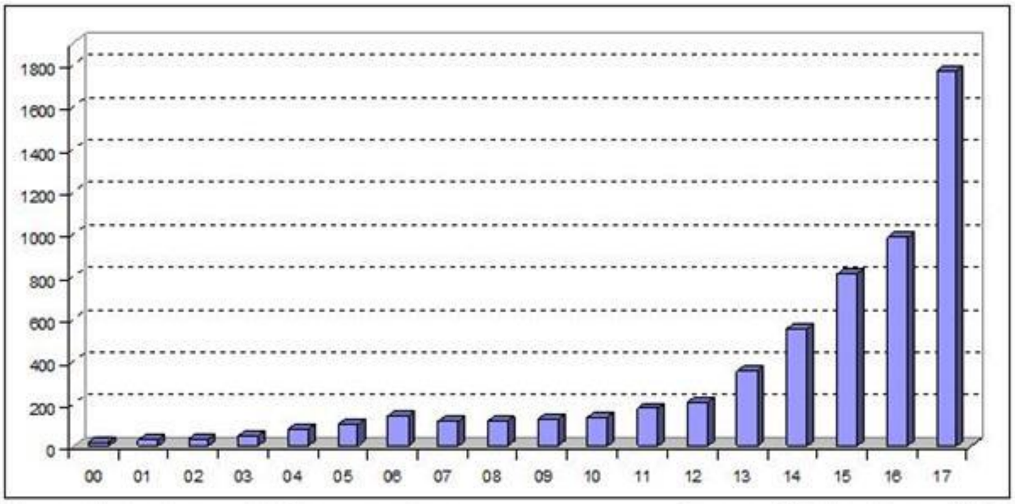
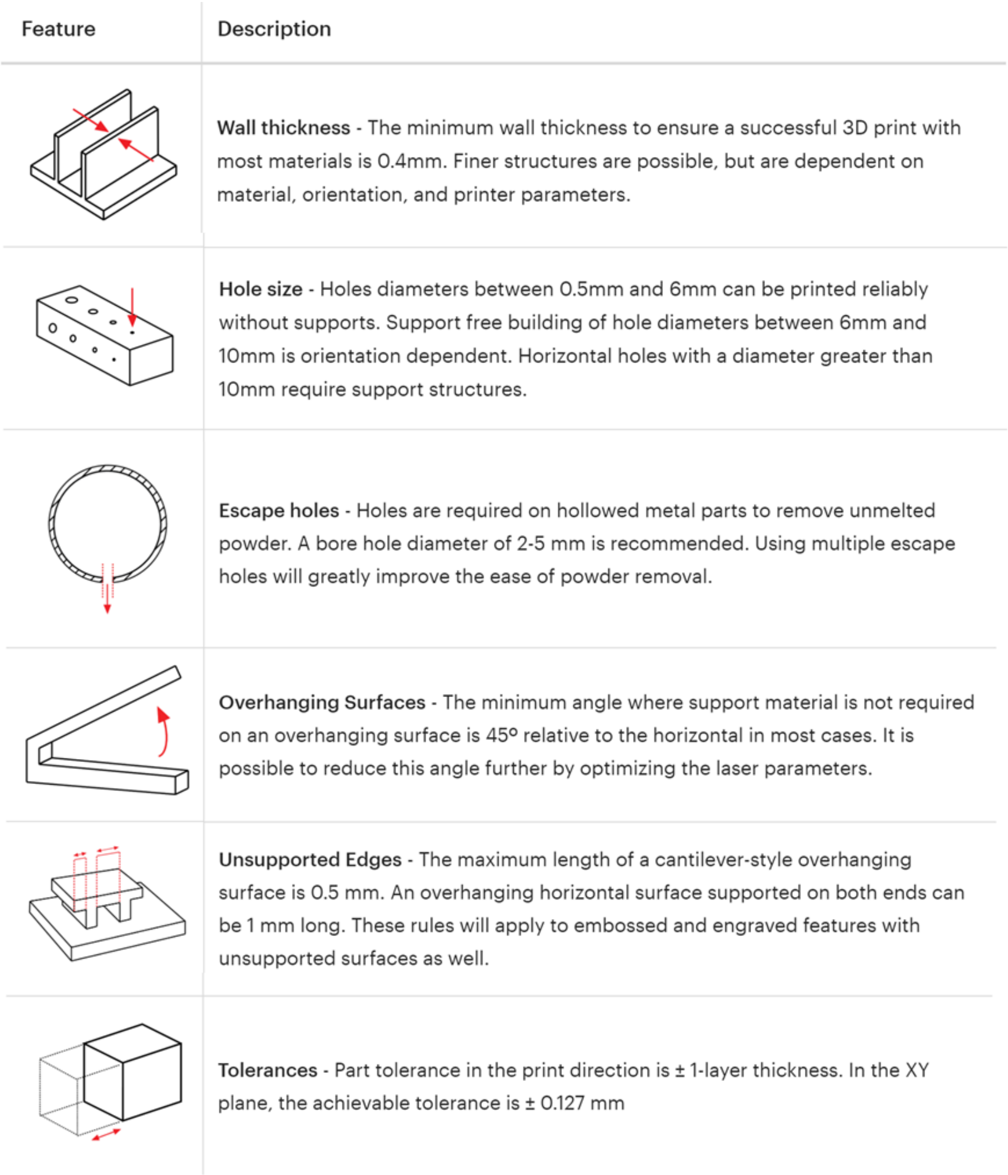


Figure 1-3: Sales of metal AM machines from 2000-2017 [12]

While interest in metal AM has seen investments, acquisitions, and partnerships to expand the capability of the technology [13-15], mass adoption of AM for the production of metal components has been limited with few successful examples in industry. In addition to uncertain material properties and limited qualification standards [16], AM technologies have been hindered by cost effectiveness [17]. An industry report of 900 global companies by Ernst & Young [18] reported that 40% of companies cannot afford the acquisition costs for a generic AM machine, ranging from \$500-\$999k [19], while 20% cannot afford the operating costs and materials. In LPBF, the most prevalent feedstock is metal powder, a material which can range in price from "\$260 to \$450 per kg" [20]. Due to this expense, un-melted feedstock is often reused (i.e., recycled) in subsequent build jobs to save costs over purchasing additional virgin powder [21]. Despite these savings, reused powder endures partial sintering with each build job due to latent heat from the melt-pool, leading to subsequent changes in the powder size distribution [22]. This phenomenon creates non-spherical particles and satellites leading to porosity and rough surfaces [23]. In parallel, the formation of oxides, soot, and exposure to ambient atmosphere can disseminate chemical impurities into the feedstock [10, 24]. Contrasting to this inherent variability in the process, LPBF cost models have implicitly assumed the powder has infinite reusability. Consequently, *there are limited studies that have examined how the cost effectiveness of LPBF is impacted when accounting for a powder feedstock with a finite quantity of reuses before it is no longer permissible in the AM process.*

Additionally, design is another critical activity that influences cost in LPBF. Poorly designed components are at risk of recoater collision, thermal distortion, and lengthy build hours (e.g., 200-300 h [10]). With machine time being one of the largest cost drivers in LPBF, designers can choose to reduce build height, minimize support structures, efficiently pack the build volume, or modify design features based on the build orientation [25-27]. However, designers must also consider trade-offs between cost and manufacturability. One example being: support structures can mitigate the propagation of residual stress during the build, but they may have limited line-of-sight access during post-processing and removal. Furthermore, designers must ensure that their geometry embodies proper Design for AM (DFAM) principles, examples shown in Figure 1-4, to ensure that the component is produced in an effective and feasible manner.



Feature	Description
	Wall thickness - The minimum wall thickness to ensure a successful 3D print with most materials is 0.4mm. Finer structures are possible, but are dependent on material, orientation, and printer parameters.
	Hole size - Holes diameters between 0.5mm and 6mm can be printed reliably without supports. Support free building of hole diameters between 6mm and 10mm is orientation dependent. Horizontal holes with a diameter greater than 10mm require support structures.
	Escape holes - Holes are required on hollowed metal parts to remove unmelted powder. A bore hole diameter of 2-5 mm is recommended. Using multiple escape holes will greatly improve the ease of powder removal.
	Overhanging Surfaces - The minimum angle where support material is not required on an overhanging surface is 45° relative to the horizontal in most cases. It is possible to reduce this angle further by optimizing the laser parameters.
	Unsupported Edges - The maximum length of a cantilever-style overhanging surface is 0.5 mm. An overhanging horizontal surface supported on both ends can be 1 mm long. These rules will apply to embossed and engraved features with unsupported surfaces as well.
	Tolerances - Part tolerance in the print direction is ± 1-layer thickness. In the XY plane, the achievable tolerance is ± 0.127 mm

Figure 1-4: DFAM guidelines for LPBF [28]

In modern practice, designers apply DFAM and iterate on the geometry through a software workflow of multiple programs (i.e., Solid Modeling, Surface Modeling, Finite Element, File Repair, Build Preprocessing, etc.) and multiple file format conversions before outputting an STL file. Despite the STL serving as the input to the slicer for an AM machine, this file lacks specification data (e.g., units, coordinate system, material data, modeling features, etc.) [29], which can significantly hinder the ability to perform future design modification and updating. Although the STL's build orientation and support structures can be adjusted with print preparation software, the STL's dearth of specifications pose a risk for lost design intent which can lead to defective parts, variability, and unattended costs later in production [30]. Alternative file formats such as 3MF and AMF store more manufacturing metadata (i.e., color, curvature, functionally-graded materials, etc.) and have seen recent integration into commercial 3D CAD programs [31, 32]. However, these files types are exported as boundary-representations that do not possess the underlying solid modeling and feature data for later modification by the designer. Beyond 3D-scanning and reverse engineering applications, the 3D CAD model is the fundamental starting point for design and development of a component in AM [2]. Therefore, *design tools must be introduced and integrated with 3D CAD modeling to aid, inform, and guide designers on manufacturability, LPBF cost estimates, and DFAM considerations at the earliest stage in the development process.*

1.5 Research Objectives

The goal in this thesis is to answer the following research questions: 1) *"How does the reusability of a metal powder feedstock impact the total costs in LPBF?"* and 2) *"What variables should designers consider when assessing their geometries with CAD-integrated DFAM tools*?" To answer the first research question, an existing LPBF cost model is modified by introducing a financial depreciation model. Through this model, the value of a powder feedstock is a function of the quantity of build cycles reused in LPBF before reaching a terminal value where it is a scrap powder and no longer permitted in the AM process. The second research question is explored by introducing a software plug-in using a 3D CAD program's application programming interface (API) to query metadata from a given geometry and provide feedback to the designer on cost, build volume constraints, and support structure generation for a given build layout.

1.6 Thesis Overview

Beginning with a literature review in Chapter 2, previous work on AM cost modeling and powder reusability is summarized along with current research on design tools to support DFAM and optimization for LPBF. Chapter 3 introduces the expanded LPBF cost model with powder reusability along with the proposed formulas and variables. A case study is presented for an automotive upright with sensitivity studies on low-to-high unit price feedstocks, geometry replicates in the build chamber, blending of virgin and reused feedstock, and low-to-high unit production. Chapter 4 expands the work in Chapter 3 by integrating the proposed cost model with a SolidWorks API plug-in that allows designers to examine the impact of powder dosage, material selection, support angle, and build layout on total cost. Additionally, ray-trace projection is implemented in a macro to automate internal and external support structure generation on the geometry followed by particle swarm optimization to determine an optimal build orientation that facilitates manufacturability and post-processing. Chapter 5 summarizes the contributions of this thesis, limitations in the reported findings, and recommendations for future work.

Chapter 2

Literature Review

Chapter 2 is presented in four sub-sections. Section 2.1 provides an overview of various cost models used in AM and fundamental observations on the cost effectiveness of the AM process against traditional manufacturing. Section 2.2 delves into related work on metal powder reuse in LPBF and notes what researchers have observed in regards to changing feedstock quality and properties. Shifting emphasis to design, Section 2.3 reviews previous DFAM tools and their common inputs and functions provided in their programs. Lastly, Section 2.4 examines optimization practices used in DFAM and is organized based on build orientation optimization for component producibility and cost reduction in the AM process.

2.1 Cost Modeling in Additive Manufacturing

The earliest cost models for AM were created by Alexander [33], who studied the impact of build orientation on the costs of parts produced through VP and ME. Using activity-based costing (ABC), Alexander grouped costs into three activities: (1) Prebuild preparation, (2) Build, and (3) Post-processing. Alexander's work determined an interdependence between costs and build orientation due to build height, build time, support materials, and post-processing. Further work by Hopkinson and Dickens [34] compared Selective Laser Sintering (SLS), VP, and ME against Injection Molding. Hopkinson and Dickens demonstrated that these AM technologies are most economically competitive in low-volume production due to the near constant cost for AM and injection molding's progressive cost reduction due to distributing tooling cost over a high quantity of units. Ruffo *et al*. [35] expanded Hopkinson and Dicken's cost model by splitting total costs into two categories. (1) Direct costs, a fixed value based on the quantity of materials and part volume, and (2) Indirect costs which are variable and treated as a function of time. Their study highlighted that build time in PBF is a summation of the “layer exposure time”, “recoating time”, and “time to heat/cool” the bed chamber to a given temperature. They also presented a “waste factor” to allocate for feedstock that could not be reclaimed from the build bed. As a follow-up, Ruffo *et al*. [36] studied cost allocation for build jobs with mixed part geometries in SLS comparing strategies that allocated cost based on a part’s volume relative to the total volume of built parts along with allocation based on a part’s theoretical cost at near-infinite production.

In cost modeling for LPBF with metal feedstock, Atzeni *et al*. [37] compared the costs of producing end-usable metal parts through DMLS against traditional Die-Casting, and reported a similar trend as Hopkinson and Dickinson where AM was most competitive at low-volume production. Atzeni *et al.* assumed that manufacturing took place in Western Europe with an operator hourly rate of “20.00 - 30.00 €/hr” and that each part would undergo “heat treatment” after the build. Material cost estimates were calculated with an assumed “10% increase in the part’s volume” to empirically account for support and waste materials. Baumers *et al*. [38] furthered Atzeni *et al*. efforts by studying the relationship between build platform utilization against energy consumption and build productivity in EBM and DMLS processes. Baumers *et al*. showed that specific energy consumption in DMLS can be reduced from “337 MJ/kg” to “240 MJ/kg” when conducting a build job with parts fully utilizing the build platform area over individually produced parts. Lindemann *et al*. [39] performed a lifecycle study for production in metal AM using Time-driven Activity-Based Costing (ABC). Costs for a generic metal AM machine were discretized into an activity workflow of CAD Preparation, Machine Preparation, Build Process, Support Removal, and Surface Treatment. Considering a single part geometry for production at 4500 h/year, their cost model found that 74% of total costs can be attributed to machine costs, followed by material costs at 12%. Sensitivity analysis revealed that the material costs could vary between 5% and 46% of the total cost based on the value of the feedstock.

Rickenbacher *et al*. [40] expanded the work of Lindemann *et al*. by proposing a generic cost model for SLM. Rickenbacher *et al*. used ABC while accounting for multiple geometries and part quantities in the same build job. Costs associated with the build time for a multiple geometry build job were evenly divided among all the parts in a layer-wise manner, based on their respective build heights. In a case study for three geometries, Rickenbacher *et al*. found that a total cost savings of 41% can be achieved by optimizing the quantity of parts on the build platform (i.e., packing density). A recent study by Fera *et al*. [41] expanded upon the Rickenbacher *et al*. cost model by allocating for energy consumption and introducing an Overall Equipment Effectiveness (OEE) index to account for “planned downtime, breakdowns, minor stops and production rejects” on the AM machine in a production environment.

Despite these developments in cost modeling literature for PBF, reviews by Thomas and Gilbert [17] and Costabile *et al*. [42] have noted that limited cost models capture the effect of powder reuse. Chan *et al*. [43] highlighted the assumption of “indefinite (powder) reuse” in their AM life cycle assessment but remarked that reuse should be further studied to examine the sustainability of metal AM processes.

In summary, cost modeling in AM is conducted primarily through engineering-based cost methods that sum the material, machine, and labor costs while also considering indirect expenses such as overhead and consumables. Relatively high machine and material costs limit AM’s cost-effectiveness to low-volume production. With material costs ranging between 5-46% [39] of the total costs in metal AM processes, current cost models are assuming unlimited reusability and infinite reuses for the powder feedstock. Consequently, traditional cost models are financially valuing reused powder as virgin powder. These models lack an analytical method for determining the value of the reused material and allocating costs as it undergoes physical and chemical changes in subsequent reuses in LPBF.

2.2 Metal Feedstocks in LPBF

Having identified that previous cost models assumed infinite reuse for the powder feedstock, this section delved into literature that studied metal powder and the impact of reused powder on the quality and performance of LPBF applications. To begin, metal powder is defined as a substance containing particles of elemental metals or alloys, normally less than 1000 microns in size [44]. This substance can be produced through various electro-chemical and thermo-mechanical processes such as Gas Atomization, Water Atomization, Centrifugal Disintegration [45-47]. In metallurgical applications, the powder can be characterized by the Density, Powder Size Distribution (PSD), Chemical Composition, Surface Chemistry, Morphology, Crystalline Phases, Flowability, and Thermal Properties [24]. Variability in powder properties can occur at numerous stages throughout the lifecycle of the material. Axelsson [48] conducted a study on Ti-6Al-4V powder produced from three independent manufacturers for EBM. Analysis of the chemical compositions found Nitrogen, Chlorine, and Yttrium outside ASTM F2924 limits, indicating contamination during the feedstock production process. Powder is also sensitive to oxidation, a naturally occurring chemical reaction where oxygen atoms undergo diffusion and exchange electrons with a metallic element to form oxides [49]. Oxides form surface films that can alter the absorptivity and melting of the material [50]. In addition to oxidation, the build environment of PBF machines can promote the formation of carbides and nitrides due to prolonged durations and reactivity at elevated temperatures [51].

Table 2-1 provides a list of AM literature that has studied the implications of reusing powder feedstock in PBF. The most common procedure in these studies was to load the AM machine with a quantity of virgin powder, complete a build job consisting of test coupons (e.g., cubes, tensile bars, etc.), remove all un-melted powder from the machine (e.g., build chamber, dispenser, overflow bin), sieve the powders together, reload the machine with the sieved feedstock and then iterate. Studies by Tang *et al*. [52] and Grainger [53] showed that Ti-6Al-4V powder lots can progressively gain oxygen content and exceed chemistry limits with successive reuses in the AM process. Despite operating in inert or vacuum environments, the oxygen pick-up was attributed to “exposure time” during the melting process, “powder handling”, and “sieving”. Tang *et al*. recommended no more than 4 build cycles in EBM to maintain compliance with ASTM F3001 (Grade 23). Grainger’s results showed that the oxygen content could potentially exceed Grade 23 limits between 15-35 build cycles; however, the study concluded there was “no requirement” to dispose of un-melted powder after it had been reused in a number of build cycles due to their study representing a “worst case” production scenario in LPBF. Mechanical properties for machined specimens built from reused Ti-6Al-4V ranged from a UTS of 910-1039 MPa (14-18% elongation) in EBM and 1012-1095 MPa (7-17% elongation) in LPBF. Seyda *et al*. [38] also reported that reuse led to changes in particle morphology which caused surface roughness to increase from 92 to 123 microns on the as-built surface.

Table 2-1: Literature on Powder Reuse in PBF

Ref.	Authors	Material	Machine	Reuse Metric	Reuses
[52]	Tang *et al*	Ti-6Al-4V	Arcam A2	Build Cycles	21
[53]	Grainger	Ti-6Al-4V ELI	Renishaw AM250	Build Cycles	38
[54]	O’Leary *et. al.*	Ti-6Al-4V ELI	Renishaw AM250	Build Cycles	5
[55]	Seyda *et al.*	Ti-6Al-4V	EOSINT M 270	Build Cycles	12
[24]	Slotwinski *et al.*	CoCr MP1	EOSINT M 270	Build Cycles	8
[24]	Slotwinski *et al.*	17-4 SS GP1	EOSINT M 270	Build Cycles	8
[56]	Jacob *et al.*	17-4 SS PH1	EOSINT M 270	Build Cycles	11
[57]	Aboulkhair *et al.*	AlSi10Mg	Realizer SLM50	-	-
[58]	Asgari *et al.*	AlSi10Mg	EOSINT M 290	-	-
[59]	Ardila *et al.*	IN718	Realizer SLM250	Build Cycles	14
[60]	Samant and Lewis	IN718	EOSINT M 280	Build Cycles	13

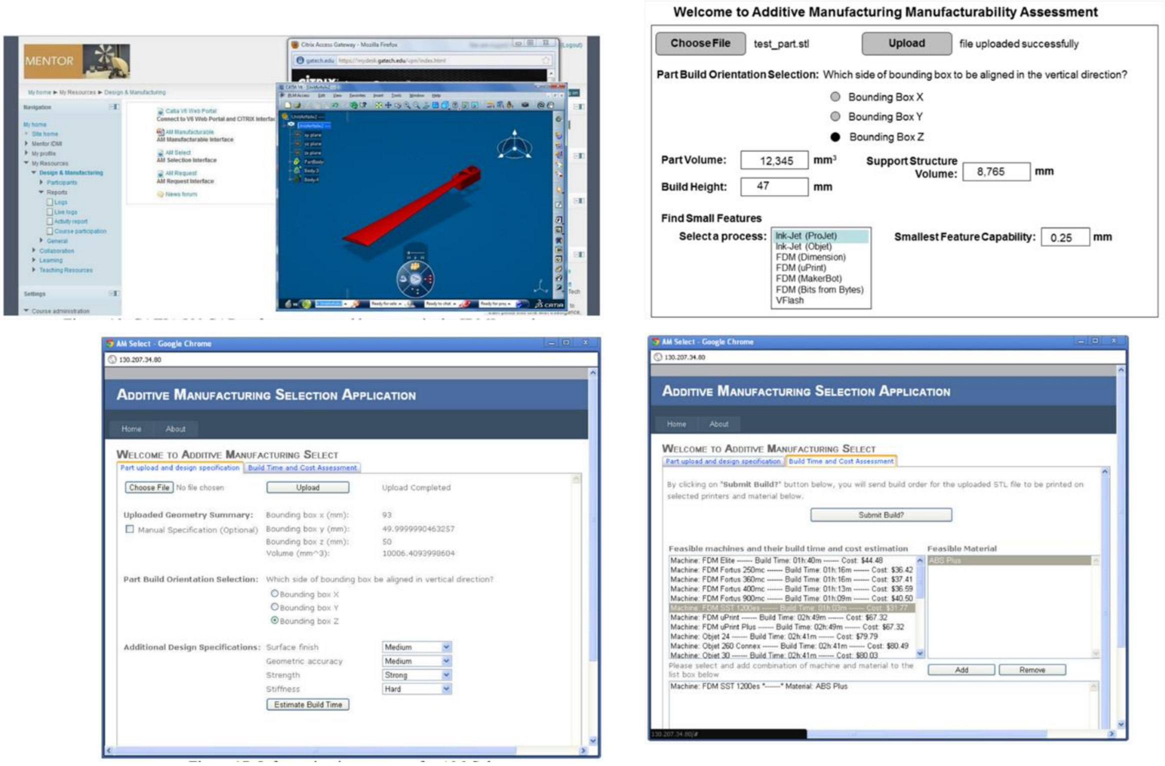
When examining additional alloys, the implications of powder reuse differ compared to those seen in Ti-6Al-4V. A study by Slotwinski *et al*. [24] on 17-4 SS GP1 and CoCr MP1 observed a similar trend of enlargement of the powder size distribution due to agglomerates and non-spherical particles with reuse. Their analysis determined that there was no significant difference in the chemical composition of the feedstocks after 8 build cycle reuses but did identify excess oxidation on the surface chemistry. A follow-up study by Jacob *et al*. [56] found that 17-4 SS PH1 showed no significant change in the chemical composition, powder morphology, and microstructure after 11 build cycles. Mechanical properties ranged from a UTS of 1325-1380 MPa (23-27% elongation) which exceeded the minimum values quoted by the PBF vendor; however, all of the specimens in their study did undergo machining and heat treatment prior to testing. For AlSi10Mg, Aboulkhair *et al*. [57] reported no significant change in the chemical composition while Asgari *et al*. [58] found that the mechanical properties, particle size, microstructure, and morphology were comparable to virgin powder. Despite their conclusions, both Aboulkhair *et al*. and Asgari *et al*. did not define a reuse metric and appeared to only report results after a reuse of one build cycle. Since AlSi10Mg is reactive in the presence of oxygen and moisture, their findings may not be applicable to feedstocks that may undergo a high quantity of reuses in PBF. Research on IN718 [59, 60] reported stable mechanical, chemical composition, and microstructure properties with reuses in LPBF. Given that IN718 is a high-temperature, age-hardened superalloy, additional studies would have to be conducted to validate these findings for reused feedstock in fatigue and high-temperature creep applications.

Standards by ASTM and NASA [61-67] have provided initial guidance on powder reuse in PBF. Both standards recommend sieving the feedstock after each build job along with the definition of a reuse metric, agreed upon by the manufacturer and the customer, for tracking a powder lot throughout production. ASTM suggests the metric of “times processed in the build chamber” (i.e., build cycles). NASA suggests that manufacturers start with the metrics of “1000 hours of machine operation, 60 days (in the machine), or 30 build operations” for non-reactive feedstocks and “500 hours of machine operation, 30 days (in the machine), or 10 build operations” for reactive alloys (e.g. Titanium, Aluminum). One difference is that ASTM standards permit the blending of virgin powder and reused powder if they already meet “chemical composition” requirements. NASA standards prohibit the “additions (of powder) to a post-production powder lot for control of PSD or chemistry”. This may be due to the fact that while reused powder can be physically sieved by particle diameter, the presence of chemical impurities, non-spherical agglomerates, and oxides may not be explicitly removed from the powder lot. Thus, under NASA standards, a reused powder could only be blended with other lots if it already met the same PSD and chemistry as virgin powder.

Upon reviewing powder reuse literature, it is difficult to generalize findings across a wide variety of PBF technologies. The reusability of a powder feedstock is both alloy and AM machine dependent due to the production process of the powder, varied energy-material interaction, powder sieving/handling, and build chamber environments. While reactive metals such as Ti-6Al-4V and AlSi10Mg may attain chemical impurities with subsequent build jobs, other alloys such as IN718 and 17-4 SS GP1 may exhibit virtually identical properties compared to virgin powder. While some studies have found little to no variation in the tensile properties, the literature is limited on the impact of reused powder on fatigue properties and correlating them to mixing virgin/reused feedstocks, testing with as-built vs. machined surfaces, heat treated specimens, along with parts built with standard or customized process parameter sets (e.g., power, scan velocity, offset, hatch spacing, skywriting, tool path).

2.3 CAD-Integrated Tools for DFAM

In order to account for the considerations of powder reuse outlined in the previous sections and to properly quote a part, CAD-integrated tools must be introduced in order to provide designers with early feedback during geometry definition and to aid in iterating towards an acceptable design. Despite this need, AM literature has limited examples of CAD-integrated tools that support DFAM with many works deferring to stand-alone analytical cost models. Perhaps the most relevant study was an Integrated Design and Manufacturing Infrastructure (IDMI) system developed by Rosen *et al.* [68] to support high school students in collaborative design and distributive manufacturing. Using an online web portal, shown in Figure 2-1, the workflow began with designing parts in CATIA and exporting an STL into a manufacturing assessment program. The AM-Manufacturable module allowed students to vary build orientation and observe variations in their part's build height and support structure volume. Students could select an AM process and the program would provide feedback on minimum feature size and scaling of the 3D model in the build volume. Next, an AM-Select program directed students to select machines and materials from a database. They could specify surface finish, strength, accuracy, stiffness requirements, and then compare build time estimates and costs. The build time estimates were based on a generalized build time equation for material jetting, material extrusion, and PBF processes [69]. Finally, the AM-Request module provided a queue of available machines where students scheduled and digitally submitted their files for printing.



Welcome to Additive Manufacturing Manufacturability Assessment

Choose File test_part.stl Upload file uploaded successfully

Part Build Orientation Selection: Which side of bounding box to be aligned in the vertical direction?

○ Bounding Box X
○ Bounding Box Y
● Bounding Box Z

Part Volume: 12,345 mm^3
Support Structure Volume: 8,765 mm
Build Height: 47 mm

Find Small Features
Select a process: Ink-Jet (ProJet)
Ink-Jet (Objet)
FDM (Dimension)
FDM (uPrint)
FDM (MakerBot)
FDM (Bits from Bytes)
VFlash
Smallest Feature Capability: 0.25 mm

AM Select - Google Chrome
130.207.34.80

ADDITIVE MANUFACTURING SELECTION APPLICATION

Home About

WELCOME TO ADDITIVE MANUFACTURING SELECT

Part upload and design specification | Build Time and Cost Assessment

Choose File No file chosen Upload Upload Completed

Uploaded Geometry Summary:
☐ Manual Specification (Optional)
Bounding box x (mm): 93
Bounding box y (mm): 49.9999990463257
Bounding box z (mm): 50
Volume (mm^3): 10006.4093998604

Part Build Orientation Selection: Which side of bounding box be aligned in vertical direction?
○ Bounding box X
○ Bounding box Y
● Bounding box Z

Additional Design Specifications:
Surface finish: Medium
Geometric accuracy: Medium
Strength: Strong
Stiffness: Hard

Estimate Build Time

AM Select - Google Chrome
130.207.34.80

ADDITIVE MANUFACTURING SELECTION APPLICATION

Home About

WELCOME TO ADDITIVE MANUFACTURING SELECT

Part upload and design specification | Build Time and Cost Assessment

By clicking on **"Submit Build?"** button below, you will send build order for the uploaded STL file to be printed on selected printers and material below.

Submit Build?

Feasible machines and their build time and cost estimation
Machine: FDM Elite ------- Build Time: 01h:40m ------- Cost: $44.48
Machine: FDM Fortus 250mc ------- Build Time: 01h:16m ------- Cost: $36.42
Machine: FDM Fortus 360mc ------- Build Time: 01h:16m ------- Cost: $37.41
Machine: FDM Fortus 400mc ------- Build Time: 01h:13m ------- Cost: $36.59
Machine: FDM Fortus 900mc ------- Build Time: 01h:09m ------- Cost: $40.50
Machine: FDM SST 1200es ------- Build Time: 01h:03m ------- Cost: $31.77
Machine: FDM uPrint ------- Build Time: 02h:49m ------- Cost: $67.32
Machine: FDM uPrint Plus ------- Build Time: 02h:49m ------- Cost: $67.32
Machine: Objet 24 ------- Build Time: 02h:41m ------- Cost: $79.79
Machine: Objet 260 Connex ------- Build Time: 02h:41m ------- Cost: $80.49
Machine: Objet 30 ------- Build Time: 02h:41m ------- Cost: $80.03

Feasible Material
ABS Plus

Please select and add combination of machine and material to the list box below

Add Remove

Machine: FDM SST 1200es *-------* Material: ABS Plus

Figure 2-1: IDMI with AM-Manufacturable, AM-Select, and AM-Request modules [68]

Contemporary CAD programs (e.g., Creo, Solidworks, Netfabb, 3DExperience) have begun offering basic AM preparation tools such as positioning 3D models in the build volume, visualizing support structures, detecting minimum feature size, and interfacing with an AM machine’s slicer [70, 71]. However, these programs have limited machine configuration, build time estimation, and cost modeling tools that capture the full fidelity of LPBF. Xometry, an AM service provider, launched a SolidWorks plug-in enabling designers to directly quote their models using Xometry’s online cost estimation platform [72]. Xometry’s platform allows designers to select between a variety of polymer and metal AM technologies, along with traditional milling and CNC. Given their focus on prototyping, the Xometry platform provides limited options for design modification, build orientation, and support structures. Because their cost model is proprietary, they provide minimal information on the parameters and considerations they use when costing parts in AM. Similarly, 3DExperience launched “Marketplace Make” as a software plug-in allowing designers to upload 3D CAD models and receive automated quotes from service bureaus. Again, the costing and feasibility assessment algorithms are proprietary and lack configurability options which may be available to designers that have direct access to AM equipment at their facility.

Overall, CAD-integrated design tools have limited studies conducted for metal AM. Rosen *et al.* [68] presented a web-based co-design platform that aided designers in exploring the manufacturability and generic costs for AM parts. Commercial CAD programs have begun offering similar tools that enable designers to examine the support structure, part placement, and slice file generation. Yet, the functionality of these programs is catered to prototyping and does not provide sophisticated cost estimation and manufacturing feasibility modules for assessing whether a component is appropriate for AM.

2.4 Orientation Optimization in DFAM

Because of the costliness of LPBF, optimization provides a valuable method for achieving objectives such as minimizing mass for a given geometry, reducing support structures, or minimizing cost. Although techniques such as topology and shape optimization are widely used in DFAM, build orientation shall be the scope of this section due to its direct influence on support structure accessibility and compensation modeling for thermal distortion.

Allen and Dutta [73] conducted early research on build orientation in VP and described three categories of faces requiring support structures as: (1) overhangs, (2) floating faces, and (3) unstable bases. They evaluated orientations based on minimal support contact area and low center of mass. Alexander *et al.* [33] further examined the impact of build orientation in ME and VP with respect to cost and noted a relationship between build time, accuracy, and surface roughness due to staircasing. Hur and Lee [74] suggested optimizing a multi-objective function based on build height, support volume, and accuracy, based on the ratio between the cusp area between layers and the STL's facet. Most studies optimizing multi-objective functions have used heuristic techniques and evolutionary algorithms to explore a broader solution space and multiple candidate orientations [75-81].

Morgan *et al*. [82] studied optimizing the orientation of metal components in DMLS and highlighted differences in optimization criteria compared to polymer AM processes due to post-processing, support structure removal, and thermal distortion. Using multiple-starting orientations, their model consisted of an unconstrained optimization algorithm that minimized total support volume. The computation time for determining a global minimum was correlated to the numbers of faces on the STL file, varying from a run-time of 2000 s at 5000 facets, up to nearly 12000 s at 20000 facets. Verma *et al.* [83] presented a framework for optimizing orientation in DMLS. STL files were evaluated using a Build Time Index and a Surface Inaccuracy Index based on the number of facets perpendicular to the build direction. Optimal orientation was determined using sequential quadratic programming and then analyzed in subsequent algorithms for adaptive and uniform slicing methods. With the build orientation capable of dropping ductility in Ti-6Al-4V from 12% to 7% [84, 85], surrogate-based optimizers have been implemented to maximize factor of safety and mechanical properties [86].

To review, this section found that build orientation impacts build time, support volume, surface roughness, and mechanical properties. Multi-objective and heuristic-based optimization can be used to determine a build orientation; however, computation time varies based on the resolution of the input 3D model. Another gap is the role of orientation on support structure removal in metal AM. Vaidya and Anand [87] demonstrated that the accessibility of generic AM support structures can be determined along 6 orthogonal and 12 diagonal directions through a segmented slice image processing algorithm on the CAD model. Accessing support structures after a metal AM build for removal can demand lengthy post-processing time and should be considered when optimizing a geometry or planning the build layout.

In the next chapter, the implications of powder reuse shall be furthered studied by examining a generic cost model for LPBF and introducing a costing methodology to capture the variation in the financial value of feedstock as it is reused in subsequent LPBF build jobs.

Chapter 3

Cost Modeling for Reused Powder Feedstocks in LPBF

As discussed in the previous chapter, the reusability of a powder feedstock is not explicitly captured in traditional AM cost models. To better account for the limited lifecycle for a powder feedstock, two models are presented for determining the financial value of a feedstock as it undergoes processing and reuse in LPBF. To evaluate each powder reuse cost model, a case study is presented for an automotive upright and assessed using the proposed models along with a generic LPBF cost model [40]. Following the case study, a sensitivity analysis is conducted examining different costing scenarios for multiple geometry replicates, alternative materials, and mass production.

3.1 Financial Depreciation Model for Reused Powder Feedstock

Literature [52-60] has shown that the implications of powder reuse varies based on the material alloy and the AM technology. Current standards [61-65] state that powder reuse is permitted; however, feedstock usage and limitations are subjective and based on agreement between the customer and the manufacturer. With no overarching guidance, manufacturers face uncertainty as to how one should financially value the feedstock as it is transitions from virgin, to reused, to scrap powder. With no definitive costing methodology, manufacturers face the risk of overvaluing reused powder as virgin powder, along with the risk of net capital loss due to excess scrap powder at the end of production. To correlate the reuses of the input material in LPBF to the monetary price of the feedstock, we propose that the powder be valued through a financial depreciation model.

In accounting practice, depreciation is defined as the "gradual decline in the financial value of property due to increasing age and eventual obsolescence" [87]. With the risk of porosity, non-uniform powder morphologies, and chemical contamination, the quality of the powder feedstock can diminish as it is continually reused in PBF. Using a depreciation model, the systematic loss in a feedstock's financial value is proportional to the powder's degraded properties and quality over a given duration of time.

Although depreciation is traditionally used in business accounting for the United States' Internal Revenue System [88], depreciation is proposed strictly in the context of a costing method for LPBF. Depreciation is a function of the maximum allowable duration for the feedstock and its salvage value, estimated market value, when it has reached the end of its useful life. As discussed previously in Section 2.2, with each powder feedstock having a unique elemental composition and LPBF technology, the corresponding duration for reuse will vary based on factors related to the build chamber environment, energy-material interaction, and powder handling.

Three common depreciation models are: (1) Straight-Line (SLN), (2) Double Declining Balance (DDB), and (3) Sum-of-the-Year's Digits (SOYD) [53]. SLN assumes a uniform reduction in value with each increment in time; however, this linear depreciation is fixed and assumes that the powder feedstock loses uniform amounts of value regardless of being at the beginning or end of its useful life. DDB presents a more accelerated model where the feedstock rapidly loses value at early stages of its useful life and then gradually less; however, since DDB applies a constant multiplier for depreciation, the salvage value is not explicitly designated and therefore unadaptable to whichever value the user designates for the end-use scrap, unless manually corrected. Serving as a median between SLN and DDB, SOYD exhibits a moderate drop in value at early life and less as the material is increasingly reused. One advantage over DDB is that the salvage value of SOYD can be specified by the user, making it a more adaptable model for a wide range of materials, with one drawback being that SOYD does not depreciate as rapidly as the DDB.

Given their customizable inputs and moderate depreciation rates, SOYD and SLN were considered for this study. Key differences being that SOYD can capture the scenario where virgin powder, being at most risk of chemical contamination and oxides, is modeled with a large rate of decline in value after initial uses in a LPBF process and then a slower rate of decline as it becomes a scrap powder. Whereas SLN can capture the scenario where a feedstock loses value at a steady and constant rate with each build, such as the lot being regularly replenished with virgin powder. The two models are presented in Figure 3-1.

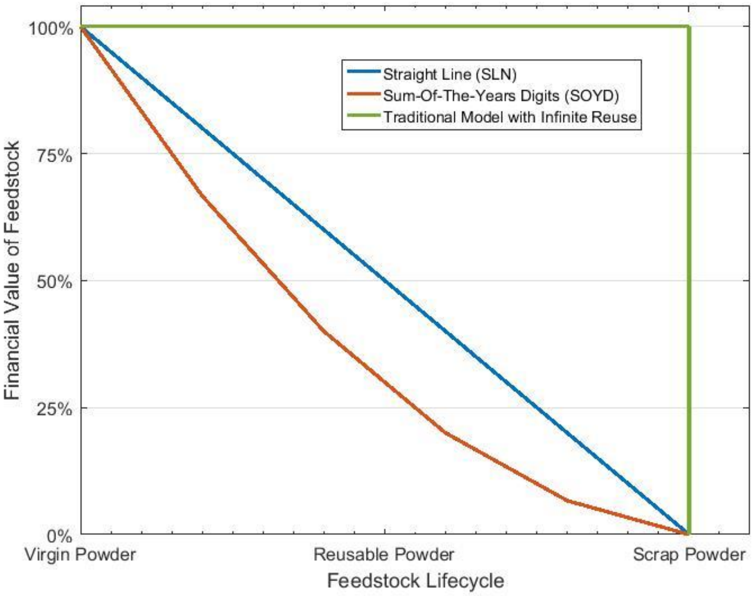


Figure 3-1: Comparison of Depreciation Methods for Powder Feedstock

With the two depreciation models, units need to be specified for assessing the duration in which the powder feedstock is reused. Build cycles are widely cited in the research [52-60]; however, build cycles are an imprecise measure due to variations in the underlying build time, build height, part orientation, and quantity of parts. While an AM build job is typically measured in units of hours [2], this value can be convoluted with the idle time and recoating time of the AM machine. Time duration metrics do not provide insight on the explicit process parameters and build chamber conditions in which a powder feedstock was processed. Additionally, time duration does not provide comparability between builds which have short versus tall build heights, demanding differing powder volumes to conduct the build. Jacob *et al.* [56] proposed one powder reuse metric as the “ratio of (laser) exposure hours to the total powder volume in the build”. They argued that such a metric can be better generalized, independent of process parameters, and allow a comparison of between different LPBF machines. Despite these promising aspects, the powder reuse metric from Jacob *et al.* hasn’t been widely studied to validate their claims.

In spite of the aforementioned limitations, for this cost model, build cycles was selected as the unit for measuring the reuse duration of a powder feedstock due to available documentation in literature and limited alternatives. Equation 1 is our proposed method for valuing the powder feedstock as function of build cycles using SOYD, with Equation 2 for SLN. For this equation, it is assumed that a powder lot is reused on a single AM machine, where the overflow, part bed, and feed bed powder are mixed and sieved after each build. Builds are conducted using consistent process parameters, atmosphere, and material handling conditions.

$$C_{m_{u+1}} = Cm_u - \left(Cm_0 - S\right) \cdot \left(\frac{U_{max} - u + 1}{\frac{U_{max}(U_{max} + 1)}{2}}\right) \tag{1}$$

where:
Cm_u is the cost of the powder feedstock that has been used *u* times ($/kg),
Cm_0 is the cost of a virgin powder feedstock ($/kg),
S is the salvage value of the powder at the end of its depreciable life ($/kg),
U_{max} is the maximum quantity of build cycles a powder can be used in LPBF (-),
u is the number of build cycles a powder has underwent in LPBF (-).

$$C_{m_u} = Cm_0 - u \cdot \frac{(Cm_0 - S)}{U_{max}} \tag{2}$$

where:
Cm_u is the cost of the powder feedstock that has been used u times (\$/kg),
Cm_0 is the cost of a virgin powder feedstock (\$/kg),
S is the salvage value of the powder at the end of its depreciable life (\$/kg),
U_{max} is the maximum quantity of build cycles a powder can be used in LPBF (-),
u is the number of build cycles a powder has underwent in LPBF (-).

3.2. Activity-Based Cost Modeling for LPBF

The proposed depreciation models are implemented by expanding upon the work of Rickenbacher *et al.* [40] and following a similar workflow to that shown in Figure 3-2. A built-up part, P_i, shall consist of geometry, G_i with N_i quantity in a build job for LPBF. Due to overlap with the previous model by Rickenbacher *et al.*, the only presented equations are those that have been modified or are unique to this thesis.

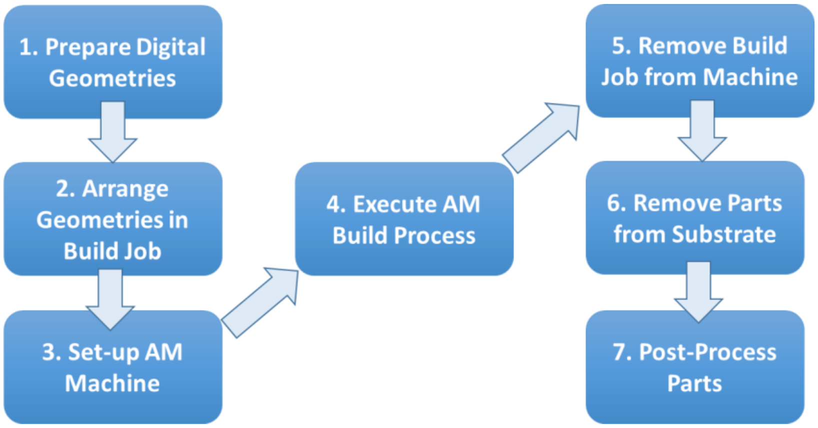

1. Prepare Digital Geometries
2. Arrange Geometries in Build Job
3. Set-up AM Machine
4. Execute AM Build Process
5. Remove Build Job from Machine
6. Remove Parts from Substrate
7. Post-Process Parts

Figure 3-2: Rickenbacher et al.'s Workflow for LPBF [40]

The first labor activity in this cost model is the preparation of the digital geometry data [40]. Upon receiving the customer’s digital model, it is assumed that the geometry meets general DFAM rules (e.g., process selection, fully enclosed surfaces, wall thickness, tolerances) [91-93].

The tasks in this activity are selecting a build orientation and generating support structures. Software packages such as NetFabb and Materialise Magics [94, 95] can automate these processes; however, these programs are not robust for metal AM and may produce designs that satisfy manufacturing requirements but fail to meet product specifications [96]. Thus, this activity is an iterative process that relies on experiential knowledge of the AM designer and must be tailored to the given LPBF technology and geometry.

Once all of the parts have been successfully prepared, the digital geometries are read into a build layout program. The manipulation and placement of geometries on the digital build tray can have repercussions regarding the likelihood of build failure (e.g., collision with recoating mechanism, curling, surface roughness) [97]. With limited literature and standards for this activity, it is also conducted in an iterative manner relying on previous knowledge and past experience. The time required for arranging the geometries is a function of the total part geometries and replicates present in the AM build job.

Machine set-up consists of uploading the digital build tray files, selecting process parameters, initializing inert gas (or vacuum depending on LPBF technology), and readying system hardware. With metal powders having explosive and physiological hazards [98-100], material handling is dangerous, requiring timely and duteous tasks for safe activity. Additional time can occur if the build calls for a different material than the one currently loaded in the machine. The total time for changing materials includes the tasks of unloading the current powder, cleaning the build chamber, replacing consumables (e.g., filters, inert gas), loading the new feedstock, and cleaning all ancillary equipment (i.e., vacuum). While Rickenbacher *et al.* used empirical factors for extra effort under inert environment and material change frequency, this has been removed since they are captured in the machine set-up and material change time, shown in Equation 3.

$$C_{setup}(P_i) = \left(C_{op} + C_{Mach}\right) \cdot \frac{\left(T_{setup} + T_{mat.change}\right)}{\sum_i N_i} \tag{3}$$

where:
C_{setup} is the cost per part for setting up the AM machine ($),
P_i is the built-up AM part corresponding to *i*th geometry (-),
C_{Mach} is the AM machine's hourly rate ($/h),
C_{oper} is the operator's hourly rate ($/hour),
T_{setup} is the time for setting up the machine (h),
$T_{mat.change}$ is total time for changing and re-loading powder in AM machine (h),
N_i is the quantity of parts with *i*th geometry (-).

The derivation of a high-fidelity build-time estimator is outside the scope of this paper. With commercial solutions available through software and the AM machine's preprocessors [101], we instead defer to a generic formula [102] for calculating individual build times in lieu of the previous regression model specific to SLM. For build jobs with multiple parts, this formula assumes that the build rate for exposing each voxel is constant and that all latency due to positioning the laser between melted powder regions is negligible. Using the algorithm proposed by Rickenbacher *et al.* [40], the recoating time for build jobs with multiple build heights is calculated in a time fraction manner for each part:

$$T_{build}(P_i) = \frac{T_{idle}}{\sum_i N_i} + T_{build\,speed} \cdot \sum_i \left(N_i \cdot V_{total_i}\right) + T_{recoat}(P_i) \tag{4}$$

where:
T_{build} is the total time required for building up a single part in a given build job (h),
P_i is the built-up AM part corresponding to *i*th geometry (-),
T_{idle} is the time when the AM machine is inactive (e.g., heating, cooling) (h),
$T_{build\,speed}$ is the average time for AM machine to melt a voxel of powder (h/cm^3),
N_i is the quantity of parts with *i*th geometry (-),
V_{total_i} is the total volume of the part and support structures for *i*th geometry (cm^3),
T_{recoat} is the total recoating time allocated to a single part (h).

After estimating the build time, the required amount of feedstock in order to execute the build job must be determined. Unique to LPBF is that the part bed lowers by one layer thickness and the feed bed platform rises between "two or three times" [10] the layer thickness to account for changes in powder leveling during the melting of each layer. This ratio of the vertical rise of the feed bed to the lowering of the part bed is referred to as dosage (DS) [103], also known as "charge". If DS is set too low, then the build can be prone to powder shorting and insufficient coverage of the build plate [104]. With limited equations available in the present literature, Equation 5 is proposed as a means for estimating the mass of the powder loaded in the feed bed for a generic AM machine using LPBF.

$$M_{FB} = DS \cdot D_x \cdot D_y \cdot Bh(P_i) \cdot \rho_t \tag{5}$$

where:
M_{FB} is the total mass of the powder loaded into the AM machine's feed bed (kg),
DS is the vertical rise of the feed bed per layer thickness in the build (-),
D_x is the length of the dispenser platform in the feed bed (mm),
D_y is the width of the dispenser platform in the feed bed (mm),
Bh is the build height of the tallest part in the build job (mm),
ρ_t is the powder tap density (kg/cm^3).

Once the operator has completed all hardware and software set-up, then the build job commences, and the AM machine proceeds to fabricate the designated part(s). To cost the activity of the AM machine throughout this duration, costs are grouped into four categories as shown in Equation 6. Machine costs includes costs pertaining to utilization and inert gas, multiplied by the build time allocated to a given part (see Equation 4). Material costs, as presented in Equation 7, are related to the mass of the powder feedstock melted by the AM machine to produce the part. In Equation 8, the powder depreciation model from Section 3.1 is introduced, where the material costs are valued as a function of the build cycles endured by the powder feedstock loaded in the AM machine. For the mass of the part, in Equation 9, empirical factors are included to compensate for powder losses due to "particles trapped in the filters" [30] during processing and loose powder trapped in hollow support structures.

$$C_{build}(P_i) = C_{Machine}(P_i) + C_{Material}(P_i) + C_{Powder\ Depreciation}(P_i) + C_{Mixing\ Depreciation}(P_i) \quad (6)$$

where:
C_{build} is the cost per part for building up a part using the AM machine ($),
P_i is the built-up AM part corresponding to *i*th geometry (-),
$C_{machine}$ is the cost per part for operating the AM machine during a build job ($),
$C_{material}$ is cost per part for the melted powder feedstock in AM process ($),
$C_{Powder\ Depreciation}$ is cost per part for un-melted feedstock in AM process ($),
$C_{Mixing\ Depreciation}$ is cost per part for blending feedstocks into the powder lot ($).

$$C_{machine}(P_i) = T_{build} \cdot \left(C_{mach} + C_{gas}\right) \quad (7)$$

where:
$C_{machine}$ is the cost per part for producing a build job in the AM process ($),
P_i is the built-up AM part corresponding to *i*th geometry (-),
T_{build} is the time for building up the entire job in the AM process (h),
C_{mach} is the AM machine's hourly operating cost ($/h),
C_{gas} is the cost for inert gas consumption during the build ($/h),
N_i is the quantity of parts with *i*th geometry (-).

$$C_{material}(P_i) = M_i \cdot Cm_u \quad (8)$$

where:
$C_{material}$ is cost per part for the powder feedstock melted in the AM process ($),
P_i is the built-up AM part corresponding to *i*th geometry (-),
M_i is the mass of a part with *i*th geometry (kg),
Cm_u is the cost of the powder feedstock that has been used in *u* build cycles ($/kg).

$$M_i = (1+\alpha) \cdot \rho_w \cdot (V_{part_i} + V_{supports_i}) + \gamma \cdot \rho_t \cdot V_{supports_i} \quad (9)$$

where:
M_i is the mass of a part with *i*th geometry (kg),
α is the percentage of powder loss due to process inefficiency (%),
γ is the percentage of powder loss due to being trapped within support structures (%),
V_{part_i} is the volume of the part body for the *i*th geometry (cm^3),
$V_{supports_i}$ is the volume of the support structures for the *i*th geometry (cm^3),
ρ_w is the powder wrought density (kg/cm^3),
ρ_t is the powder tap density (kg/cm^3).

The third cost category in Equation 6 is explained by reviewing the fundamentals of the LPBF process. A build job requires an excess of powder feedstock to fill the powder bed and support built-up geometries throughout processing. Any un-melted powder at the end of a build can become degraded by agglomerates, soot, and oxides, which are inherent to the AM process. These byproducts diminish the financial value of the feedstock, regardless of subsequent sieving, because the un-melted powder becomes populated with impurities that can propagate into future layers or builds. Thus, any un-melted powder in the part bed loses the opportunity to be implemented as a virgin powder and produce parts with minimal deviation from the base material. To allocate cost for this phenomenon, we propose that the financial value lost by the surrounding un-melted powder be charged to all parts produced within the build job. For this third category, the proposed cost is defined as “Powder Depreciation” and cost is calculated using Equation 10:

$$C_{Powder\ Depreciation}(P_i) = \frac{M_i}{\sum_i (N_i \cdot M_i)} \cdot \left(M_{FB} - \sum_i (N_i \cdot M_i)\right) \cdot (Cm_u - Cm_{u+1}) \qquad (10)$$

where:
$C_{Powder\ Depreciation}$ is the cost per part for un-melted feedstock in the AM process ($),
P_i is the built-up AM part corresponding to ith geometry (-),
M_i is the mass of a part with ith geometry (kg),
N_i is the quantity of parts with ith geometry (-),
M_{FB} is the total mass of the powder loaded into the AM machine’s feed bed (kg),
Cm_u is the cost of the powder feedstock that has been used in u build cycles ($/kg).

In Equation 10, the powder depreciation cost is calculated by taking the mass of the feedstock loaded in the feed bed and subtracting the total mass of all built-up parts, including their corresponding powder losses. This is multiplied by the difference in financial value of the feedstock, at its present amount of build cycles, to the diminished value after one additional build cycle. Parts within the build job are allocated the depreciation cost as a function of their mass fractions relative to the total mass of all built-up parts. Through depreciation, this costing method accounts for the melting and, in-parallel, the lost value of un-melted powder feedstock when building up a part in PBF. As previously mentioned in Equation 5, the calculation of the powder mass loaded in the feed bed is non-trivial and must be sufficient to fill the part bed completely and build-up the geometries. The amount of loaded powder feedstock is influenced by the total volume of geometries in the part bed, their location on the substrate, the material type, and how the LPBF technology accounts for changes in the levelling of a layer as powder regions melt and re-solidify.

Lastly, when there is insufficient mass to continue production, a mixture of powders (e.g., 80% reused, 20% virgin) may be blended together in order to refill the feed bed. This scenario is captured in Equation 11. Under present ASTM standards [61-65], a powder mixture containing any used powder, is classified as a "used powder". Similar to the observations in Equation 10, when a small quantity of virgin powder is added to a reused powder lot, that virgin powder loses the opportunity to be blended into a lot with other virgin powder and is thus diminished in value. Therefore, the value of the blended feedstock, regardless of any additional virgin powder, is penalized to the value of the most reused powder within the lot. Equation 11 captures this cost by assuming that the majority of the blended powder lot shall consist of reused powder and that any newly added powder has less build cycles and is a relatively smaller portion of the total mass in the blended lot.

$$C_{Mixing\ Depreciation}(P_i) = \frac{M_i}{\sum_i (N_i \cdot M_i)} \cdot \beta_k \cdot \left(M_{FB} - \sum_i (N_i \cdot M_i)\right) \cdot (Cm_w - Cm_{u+1}) \quad (11)$$

where:
$C_{Mixing\ Depreciation}$ is cost per part for blending feedstocks into the powder lot ($).
P_i is the built-up AM part corresponding to *i*th geometry (-),
M_i is the mass of a part with *i*th geometry (kg),
N_i is the quantity of parts with *i*th geometry (-),
β_k is the mass percentage of a powder k added to the powder lot (%),
M_{FB} is the total mass of the powder loaded into the AM machine's feed bed (kg),
Cm_w is the cost of the powder feedstock that has been used in *w* build cycles ($/kg).
Cm_u is the cost of the powder feedstock that has been used in *u* build cycles ($/kg).

Once the AM process has finished building, the build job and all subsequent parts are physically removed from the AM machine. Equation 12 [40] takes the time required for this task and evenly divides it among the total number of parts created in the build job. Activities at this stage pertain to removing the build substrate from the machine, collecting all loose un-melted powder from the part bed, cleaning the machine, removing all powder from the feed bed and overflow bins, sieving the used powder, storage, and documentation. Empirical factors for extra effort under inert environment have been removed from the original model.

$$C_{Removal}(P_i) = (C_{op} + C_{mach}) \cdot \frac{T_{Rem}}{\sum_i N_i} \tag{12}$$

where:
$C_{Removal}$ is the cost per part for removing the substrate/parts from the AM machine ($),
P_i is the built-up AM part corresponding to ith geometry (-),
T_{Rem} is the time required to remove parts, clean machine, perform all ancillary tasks (h),
C_{oper} is the operator's hourly rate ($/hour),
C_{Mach} is the AM machine's hourly operating cost ($/h),
N_i is the quantity of parts with ith geometry (-).

The next step involves separation of parts from the build substrate. Modifying the original formula [40], parts produced in a build job using LPBF may have built-up residual stress and thus undergo a stress-relief [105] to reduce geometric distortion upon separation. Once completed, wire electrical discharge machining (EDM) is used to physically detach all parts from the substrate. The costs for wire EDM are allocated based on the contact area occupied by a part, and their support structures, on the substrate as follows in Equation 13.

$$C_{Substrate}(P_i) = \frac{C_{stress}}{\sum_i N_i} + C_{EDM} \cdot \frac{A_{con}(G_i)}{\sum_i N_i \cdot A_{con}(G_i)} \tag{13}$$

where:
$C_{Substrate}$ is the cost per part for separating a part from the substrate ($),
P_i is the built-up AM part corresponding to ith geometry (-),
C_{stress} is the cost for stress-relieving a build plate($),

C_{EDM} is the total cost for separating a part via EDM ($),
A_{con} is the connected area of a part to the substrate (cm^2),
N_i is the quantity of parts with *i*th geometry (-).

Once all parts have been separated from the build substrate, these components can undergo additional post-processing to meet customer requirements. Due to parts having individually-tailored functions and applications, the required operations and sequence of their events will vary due to the specifications ordered by the costumer. For Equation 14 [40], cost is calculated for post-processing based on support structure removal for an individual part. The time for post-processing is a function of the part's geometric complexity and can increase if additional time or equipment is needed. For estimation purposes, the equation is broadly defined and can be extended to additional post-processing operations as designated by the user.

$$C_{postp}(P_i) = \sum\nolimits_i \left(T_{postp}(G_i) \cdot \left(C_{op} + C_{tools}\right)\right) \tag{14}$$

where:
C_{postp} is the total cost for post-processing ($),
P_i is the built-up AM part corresponding to *i*th geometry (-),
T_{postp} is the time required to post-process a part geometry ($),
G_i is the ith geometry (-),
C_{op} is the operator's hourly rate ($/hour),
C_{tools} is the hourly rate of tools and machines for post-processing ($).

In summary, this cost model consists of 7 activities in the AM workflow (see Figure 7). Costs are allocated based on an AM operator's labor for part preparation, arranging geometries on the build tray, setting-up the AM machine, executing the build job, and removing the substrate and decommissioning the machine. Afterwards, post-processing of the build job begins with stress-relief, wire EDM, followed last by individual post-processing to produce a fully-functional component. These 7 activities are added together in Equation 15 to produce the total cost for a part made using PBF:

$$C_{Total}(P_i) = C_{prep}(P_i) + C_{buildjob}(P_i) + C_{Setup}(P_i) + C_{Build}(P_i) + C_{Removal}(P_i) + C_{Substrate}(P_i) + C_{Postp}(P_i) \qquad (15)$$

where:
C_{total} is the total manufacturing costs ($),
C_{prep} is the cost per part for preparing the digital geometry data ($),
$C_{buildjob}$ is the cost per part for the build tray assembly ($),
C_{setup} is the cost for setting up the machine ($),
C_{build} is the cost for building up a part in the AM process ($),
$C_{Removal}$ is the cost for removing the substrate/parts from the machine ($),
$C_{Substrate}$ is the cost for separating a part from the substrate ($),
C_{postp} is the total cost for post-processing ($).

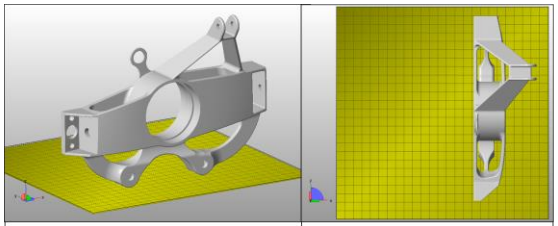
Lastly, Equation 16 is introduced for calculating the build volume utilization (i.e., capacity, packing) [25] to quantify the extent at which all the geometries are occupying space in the print bed during a build operation.

$$BVU_i = \frac{\sum_i \left(N_i \cdot V_{total_i} \right)}{Sb_x \cdot Sb_y \cdot Bh(P_i)} \qquad (16)$$

where:
BVU is the capacity at which a build job occupies the volume within the print bed (%),
V_{total_i} is the total volume of the part and support structures for ith geometry (cm^3),
N_i is the quantity of parts with ith geometry (-),
Sb_x is the length of the build substrate in the part bed (cm),
Sb_y is the width of the build substrate in the part bed (cm),
Bh is the build height of the tallest part in the build job (cm),

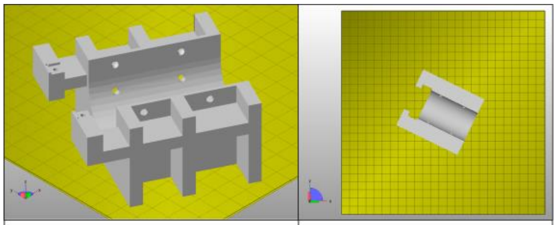
3.3. Implementation and Case Studies

To demonstrate our approach against a traditional LPBF cost model, costs were studied for two parts designed for and produced by LPBF. Figures 3-3 and 3-4 show the geometric data and build orientation for each of the parts. These parts were selected due to their differences in geometry, volumes, and build materials. All parts were manufactured on an EOSINT M280 DMLS machine at Penn State's Center for Innovative Materials Processing through Direct Digital Deposition (CIMP-3D). Following the workflow of Rickenbacher *et al.*, STL files were first imported into Materialise Magics for digital preparation and support structure generation. Geometries were then entered into EOS RP Tools and sliced to form the build job file. Next, the files were entered into EOS PSW, where process parameters were selected, and then the AM process commenced. Once completed, the build substrate was removed from the machine, the machine was cleaned, powder was sieved, and build documentation was completed. Finally, the parts and substrate were shipped to a local manufacturer for stress-relief and wire EDM, before returning to CIMP-3D for support removal.



$Geometry$	G_1
N_i	1
$V_{part}\ (cm^3)$	175
$V_{supports}\ (cm^3)$	412
$V_{total} (cm^3)$	587
$A_{con} (cm^2)$	69
$Bounding\ Box\ (mm)$	74 x 218 x 172

Figure 3-3: Automotive Upright Geometry G_1



$Geometry$	G_2
N_i	1
$V_{part}\ (cm^3)$	95
$V_{supports}\ (cm^3)$	97
$V_{total} (cm^3)$	192
$A_{con} (cm^2)$	59
$Bounding\ Box\ (mm)$	76 x 84 x 47

Figure 3-4: Testing Apparatus Geometry G_2

All equations from Section 3.2 were written in a MATLAB program and used for costing each of the builds. For this case study, constants for powder losses due to process inefficiency (α) and powder trapped in support structures (γ) were chosen to be 40% and 25%, respectively, based on CIMP-3D staff's experience from five years of producing metal AM parts at the facility [48]. Additionally, a DS of 2.25 was assumed to be sufficient for all build jobs in the case study.

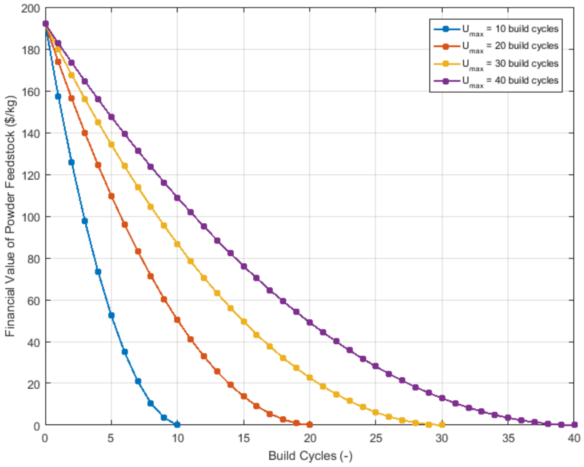


Figure 3-5: Powder Feedstock Value vs. Build Cycles vs. Maximum Build Cycles

The graph in Figure 3-5 shows an example of the cost for a powder feedstock at each of the maximum build cycles. Each point represents the financial value of a powder feedstock with the given number of accumulated reuses. The salvage value, or estimated resale value, of a powder that exceeded the maximum amount of build cycles was assumed to be zero. Since the feedstocks have a wide range of permissible reuses, each feedstock was modeled as having maximum reuses for up to 10 or 30 build cycles to correspond with recommendations from NASA [67] and allow comparison between conservative and moderate reuse limits. Table 3-1 and Table 3-2 contains all the constants used in this model, with material data collected from datasheets available by the manufacturer [106-109]. Additionally, two prices are considered for the virgin powder to highlight scenarios where a manufacturer may conduct a one-time purchase of powder lots, generally a more expensive case. In contrast, the second scenario being manufacturers in large-scale production purchasing large allotments of powder which may receive a discounted price.

Table 3-1: Cost Model Constants

Variable	**Description**	**Value**	**Units**
C_{oper}	Operator's hourly rate	110	\$/h
C_{pc}	Cost for computer workstation with all software and licenses	100	\$/h
C_{Mach}	AM machine's hourly rate	60	\$/h
C_{gas}	Cost for inert gas consumption during the build	10	\$/h
$T_{buildjob}$	Time required for arranging all geometries in the build job	1	h
T_{setup}	Time required to set up the AM machine	2	h
$T_{mat.change}$	Time for changing and loading new powder into AM machine	3	h
T_{idle}	Time when AM machine is inactive in build (heating, cooling)	-	h
$T_{recoat\ rate}$	Average time for AM machine to spread one layer of powder	9	sec
T_{rem}	Time required to remove substrate, clean machine after build	3	h
α	Percentage of powder loss due to process inefficiency	40	%
γ	Percentage of powder loss due to entrapment in supports	25	%
T_{Rem}	Time required to remove parts, clean machine after build	3	h
C_{stress}	Total cost for thermally processing all parts on build substrate	350	\$
C_{EDM}	Total cost for separating all parts on substrate via EDM	200	\$
C_{tools}	Cost for work area with tools and machines for post-processing	50	\$
DS	Vertical rise of the feed bed per layer thickness in the build	2.25	-
D_x	Length of the dispenser platform in the feed bed	228	cm
D_y	Width of the dispenser platform in the feed bed	250	cm
Sb_x	Length of the build substrate platform in the part bed	250	cm
Sb_y	Width of the build substrate platform in the part bed	250	cm

Table 3-2: Material Constants

$\boldsymbol{Material}$	$\boldsymbol{\rho_t}$ $\boldsymbol{(g/cm^3)}$	$\boldsymbol{\rho_w}$ $\boldsymbol{(g/cm^3)}$	$\boldsymbol{L_T}$ $\boldsymbol{(\mu m)}$	$\boldsymbol{V_{build\ speed}}$ $\boldsymbol{(cm^3/h)}$	$\boldsymbol{Cm_0}$ $\boldsymbol{(\$/kg)}$	$\boldsymbol{S}$ **(\$)**	$\boldsymbol{U_{max}}$ $\boldsymbol{(-)}$
Ti64	2.74	4.41	30	13.5	{272, 680}	0	{10, 30}
GP1	5.3	7.8	20	7.2	{40, 100}	0	{10, 30}
AlSi10Mg	1.5	2.67	30	26.6	{60, 150}	0	{10, 30}
IN718	5.1	8.15	40	14.4	{76, 190}	0	{10, 30}

Table 3-3 lists the resulting build data for the two example parts. Geometry G_1 had a build time of 55 hours, and Geometry G_2 had a build time of 31 hours. The estimated build time was over-predicted for both parts by 6%. This discrepancy is attributed to the assumption that T_{idle} is zero for preheating, machine cool-down, and laser positioning between hatches during the build process. G_1 required 61 kg of powder to perform the build, whereas G_2 required 32 kg. The next sections highlight costing scenarios that were generated for all material reuses ranging from virgin powder, reused powder, and powders that reached their maximum allowable build cycles.

Table 3-3: Labor Time and Build Results

$Part$	$Material$	N_i $(-)$	V_{total} (cm^3)	Bh (mm)	T_{prep} (h)	$T_{build\ estimate}$ (h)	$T_{build\ actual}$ (h)	T_{postp} (h)	M_i (kg)	M_{FB} (kg)
G_1	Ti64	1	587	172	3	58	55	3	3.9	61
G_2	GP1	1	192	47	2	33	31	2	2.2	32

3.3.1 Comparison of Depreciation Models

The first case study was conducted to conceptually highlight differences between a traditional LPBF cost model and the proposed powder reuse model using SOYD and SLN. Build costs for $C_{material}$, $C_{powder\ depreciation}$, and $C_{mixing\ depreciation}$ were estimated for production of 10 units of G_1, with Ti-6Al-4V, valued at \$272/kg, with a maximum reuse duration of 10 build cycles. Production assumed one part per build job. Feedstock blending was assumed to take place after every build job with 6.39 % of the feed bed being mixed with virgin powder in order to replenish powder that was consumed by the build geometry. Additionally, it was assumed that all required feedstock for production, including mixing, was purchased prior to the build, amounting to an initial investment of \$26,139. The results of the comparison study are shown in Table 3-4 and Table 3-5.

Table 3-4: Costing for SOYD vs. Infinite Reuse

Sum-of-the-Years Digits Depreciation						**Traditional**		
Build Cycle	Cm_u **($/kg)**	$C_{material}$ **($)**	$C_{powder\ depreciation}$ **($)**	$C_{mixing\ depreciation}$ **($)**	$Total$ **($)**	Cm_u **($/kg)**	$C_{material}$ **($)**	$Total$ **($)**
1	272	1061	2824	0	3885	272	1061	1061
2	223	868	2541	193	3602	272	1061	1061
3	178	694	2259	366	3320	272	1061	1061
4	138	540	1977	521	3037	272	1061	1061
5	104	405	1694	656	2755	272	1061	1061
6	74	289	1412	771	2473	272	1061	1061
7	49	193	1130	868	2190	272	1061	1061
8	30	116	847	945	1908	272	1061	1061
9	15	58	565	1003	1626	272	1061	1061
10	5	19	282	1042	1343	272	1061	1061
Total Sum					26139			10610
Total Net					0			-15529

The results in Table 3-4 show that for the first build, the $C_{material}$ was $1061 and the $C_{powder\ depreciation}$ was $2824, nearly 2.6 times the cost of the material. As the number of build cycles increased, $C_{material}$ decreased from $2824 to $282, while the powder depreciation rose from $0 to $1042 due to the un-melted powder approaching the salvage value of zero as the powder was continually reused. In comparison to a traditional LPBF model which assumed indefinite reuse, the material cost was $1061 and constant over the production of 10 build cycles, but was undervaluing SOYD's sum of $C_{material}$, $C_{powder\ depreciation}$, and $C_{mixing\ depreciation}$ resulting in a total that was 1.2 to 3.6x smaller. When looking at the net total in comparison to the initial investment of $26,139 for all the feedstock, SOYD was able to recuperate all of costs at the end of production, whereas the infinite reuse costing approach had a net loss of -$15529.

Table 3-5: SLN vs. Traditional Infinite Reuse

Straight Line Depreciation						Traditional		
Build Cycle	Cm_u **($/kg)**	$C_{material}$ **($)**	$C_{powder\ depreciation}$ **($)**	$C_{mixing\ depreciation}$ **($)**	$Total$ **($)**	Cm_u **($/kg)**	$C_{material}$ **($)**	$Total$ **($)**
1	272	1061	1553	0	2614	272	1061	1061
2	245	955	1553	106	2614	272	1061	1061
3	218	849	1553	212	2614	272	1061	1061
4	190	743	1553	318	2614	272	1061	1061
5	163	636	1553	424	2614	272	1061	1061
6	136	530	1553	530	2614	272	1061	1061
7	109	424	1553	636	2614	272	1061	1061
8	82	318	1553	743	2614	272	1061	1061
9	54	212	1553	849	2614	272	1061	1061
10	27	106	1553	955	2614	272	1061	1061
Total Sum					26139			10610
Total Net					0			-15529

In SLN, the material cost for the first build was $1061, with a feedstock depreciation of $1553. As the quantity of builds increased, the sum of the of $C_{material}$, $C_{powder\ depreciation}$, and $C_{mixing\ depreciation}$ was constant throughout the build at $2614. Because SLN is a linear function, all changes for the of $C_{material}$, and $C_{mixing\ depreciation}$ were in equal increments of $106 with each build. Unlike SOYD, $C_{powder\ depreciation}$ was the same value for every build job regardless of consisting of a virgin or scrap powder. When compared to traditional costing in LPBF, SLN was 2.4x larger but was also constant in value regardless of the whether the build was virgin powder or near the end of permitted reuses. Like SOYD, SLN was able to break even and recuperate the capital investment of all of the powder in the builds. Based on these two tables, it can be determined that *SLN and SOYD provide a means of recuperating capital loss for feedstock which may have limited reuses in LPBF whereas, a traditional model assuming indefinite reuse may risk a net loss due to undervaluing the overall costs for the build with respect to un-melted feedstock and any blending prior to the build.*

3.3.2 Total Costing for Example Parts

Having completed the comparisons between SLN and SOYD against a traditional LPBF cost model, the next case study aimed to understand how the powder reuse cost model would impact total costs. This case study examined the total costs for a single build job built with a feedstock that had accumulated various amounts of reuses. No feedstock blending was considered in these cases. Depreciation was modeled using SOYD. Figure 3-6 shows the range of total costs, including all labor activities, for Geometry G_1 as a function of the build cycles endured by the feedstock loaded for the build job. The points along the graph represent the cost for a build job loaded with a powder feedstock with the given number of reuses. Powders having a U_{max} of 10 build cycles were the most expensive due to having the shortest allowable reuses and therefore the most rapid decline in value. Meanwhile, powders with a 30 build cycle limit had a longer reuse duration and thus a slower rate of decline.

The total cost for manufacturing the part with virgin powder, represented as zero build cycle feedstocks, ranged between \$10,000 and \$16,500 when the feedstock was valued at \$680/kg. Whereas, total build costs ranged between \$8800 and \$12,000 when virgin powder was valued at \$272/kg. The lowest cost scenario was the use of a powder that exceeded the maximum amount of permissible build cycles (i.e., powder that is chemically out-of-specification, diminished flowability, etc.), and total costs were \$6800. This is because as a powder is increasingly reused, $C_{material}$ and $C_{powder\ depreciation}$ decrease in proportion to the diminishing financial value of the powder feedstock with each build cycle. At this lower limit, the feedstock's value has been reduced to the salvage value, zero for this case study. Hence, the material cost has diminished and the depreciation cost has become zero, because of zero difference in the financial value of a feedstock that has become a scrap material.

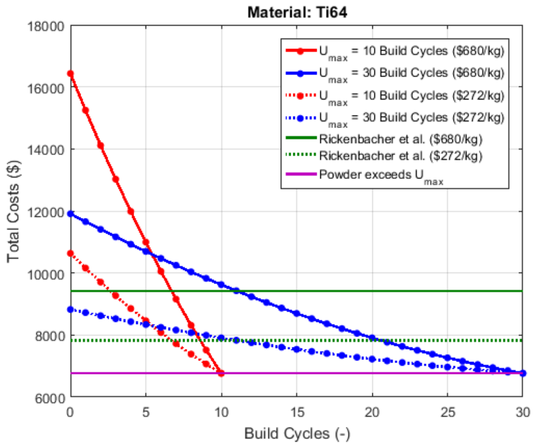


Figure 3-6: Total Costs vs. Build Cycles for Powder Reuse for Geometry *G_1*

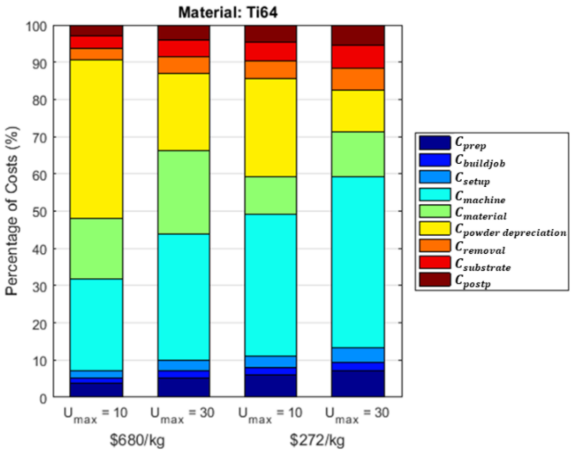


Figure 3-7: Cost for Workflow Activities for Geometry *G_2*

The previous Rickenbacher *et al.* model, which assumed unlimited powder reuse, had a constant value of $7840 and/or $9400, regardless of the reuses accumulated by the input feedstock. In comparison to these calculated cost scenarios, *traditional models with unlimited reuse cost undervalued the total cost of build jobs with virgin Ti64 powder between 26% and 75%, when the feedstock was valued at $680/kg, or from 12% to 35%, when valued at $272/kg.* After 7 and/or 13 build cycles, the Rickenbacher *et al.* cost model started to overvalue total costs and thus, build jobs using a powder that had surpassed these cycles could achieve a cost savings. The largest cost savings was a 38% reduction compared to the Rickenbacher *et al*, specifically when using a powder that exceeds its useful life (U_{max}).

Figure 3-7 shows a cost breakdown using virgin powder and each of their maximum reuses. The top three costs were $C_{powder\ depreciation}$, $C_{machine}$, and $C_{material}$. The depreciation cost was the largest cost for a powder with a 10 build cycle limit and amounted to 42% of the total cost. When U_{max} was equal to 30 build cycles, the depreciation cost was overtaken by the machine cost, and subsequently minimized to 20% of costs. In comparison, the depreciation costs were more than 2.6 times the value of the material costs.

Similar analysis was conducted on Geometry G_2 and displayed in Figures 3-8 and 3-9. Builds using virgin GP1 powder had a range between \$5000 and \$5700. The lower limit for builds using powders that exceeded U_{max} was \$4600. Using the fixed material cost model as the reference, which valued the builds at approximately \$4900, *virgin GP1 powder builds were undervalued between 3% and 11%, when the initial feedstock was valued at \$100/kg, but was only undervalued at 1% and 4%, with virgin powder valued at \$40/kg*. Upon surpassing 6 and/or 11 build cycles, the totals costs became overvalued, and thus a cost savings of 5% could be achieved by building with a powder outside the allowable for U_{max}. The largest cost for Geometry G_2 was $C_{machine}$ at 42%, $C_{powder\ depreciation}$ at 11%, and C_{postp} at 10% of the total costs. Similar to Geometry G_1, when using a 10 build cycle maximum virgin powder, the depreciation cost was twice that of the material costs.

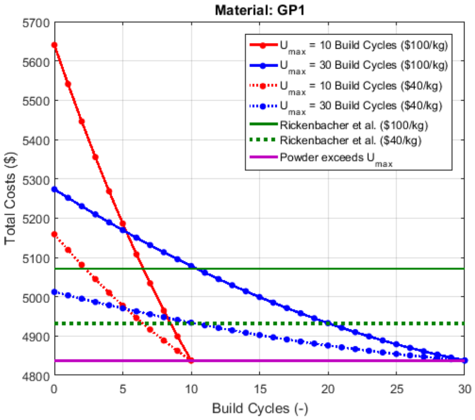


Figure 3-8: Total Costs vs. Build Cycles for Powder Reuse for Geometry G_1

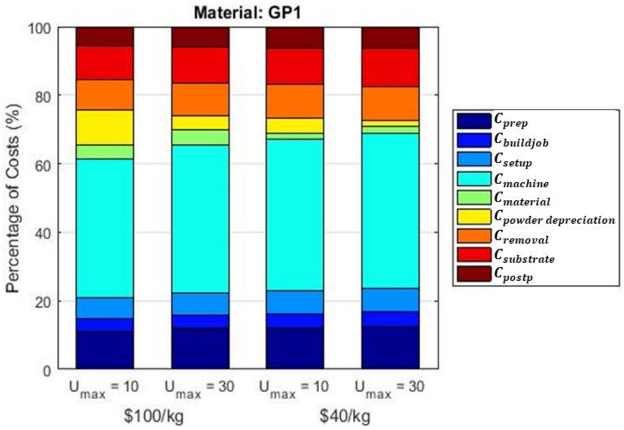


Figure 3-9: Cost for Workflow Activities for Geometry G_2

Based on these observations, $C_{powder\ depreciation}$ was one of the largest costs for Geometries G_1 and G_2. It was at most 42% of the total costs for G_1 but only 11% in G_2. When looking specifically at G_1 and its use of Ti-6Al-4V powder, the relatively larger depreciation cost in comparison to G_2 is due to G_1 requiring nearly twice as much powder to fill the feed bed due to differences in tap densities and part build heights, along with the Ti64 powder being nearly seven times more expensive than GP1. The machine cost had a relatively higher percentage in GP1 because of its 20 micron layer thickness and 7.2 $\mathrm{h/cm^3}$ build speed, which was half the speed when using Ti64. In both examples, when using a powder with a U_{max} of 10 build cycles, the depreciation cost was greater than twice the cost of the melted material. One interpretation for this result is that the utilized build envelope (i.e., volume packing) is uneconomical for the given build since the surrounding un-melted powder is being put at risk of contamination and degradation, and thus more costly than the built-up parts produced in the AM process.

Overall, this section found that $C_{machine}$ was the most pervasive cost for both builds of the example parts. This is due to the volume of the geometries and support structures being consolidated during processing, their corresponding build heights, and the build speeds at which the DMLS machine can melt the given material. The labor activities pertaining to $C_{buildjob}, C_{setup}, C_{removal}$, and $C_{substrate}$ showed no significant cost fluctuation among the example parts. This is because these activities are standardized procedures with average completion times based on the skill of the AM operator and independent of the geometries in the build. While C_{prep} and C_{postp} can vary for complex geometries requiring support structures, these costs ranged between 3-11% of the costs in all of the builds, due to larger costs being attributed to machine time, material, and depreciation.

3.3.3 Sensitivity to Build Volume Utilization and Material Selection

Since both example parts were printed as single component build jobs, this scenario may not be indicative of industrial applications in mass production. Consequently, single component build jobs create a scenario where the depreciation cost of the un-melted powder may exceed the cost of melted powder, implying that more cost is being allocated to the un-melted material than the material that is being processed. Based on this observation, a sensitivity analysis was conducted by varying the build volume utilization to determine how the material and powder depreciation cost change as more part replicates are added to the build plate.

For this section, $G_1$was varied from one to three replicates, while G_2 from one to five; the quantity of replicates differed due to build volume constraints for the respective geometry. Both material and depreciation costs were normalized as percentages of the total cost of their respective build jobs to account for subsequent increases in machine time and post-processing due to the additional geometries. Secondly, sensitivities pertaining to the build material of the parts was explored by modeling the builds with feedstock of Ti-6Al-4V, GP1, IN718 and AlSi10Mg to highlight differences due to the build rates and processing of the designated material. Build time estimates were produced for these revised build job assemblies generated using Equation 5. The generated costs all assume the same value of dosage along with the mass in the feed bed for each build job. Each model assumes the builds consisted of a virgin powder feedstock with maximum build cycle limits between 10 and 30 build cycles, with no blending. Results are shown in Tables 3-6, 3-7, 3-8 and 3-9. Rows were highlighted to identify the minimum BVU at which $C_{material}$ was larger than $C_{powder\ depreciation}$.

Table 3-6: Ti-6Al-4V Sensitivity for Material vs. Powder Depreciation Costs

			U_{max} = 10 reuses				**U_{max} = 30 reuses**			
			$680/kg		**$272/kg**		**$680/kg**		**$272/kg**	
	N (-)	**BVU (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**
G_1	1	5	16	43	10	26	**22**	**21**	**6**	**5**
	2	11	24	30	14	17	30	13	7	3
	3	16	**29**	**22**	**16**	**12**	33	9	8	2
G_2	1	6	13	30	7	16	**17**	**13**	**4**	**3**
	2	13	20	21	11	11	24	9	5	2
	3	19	**25**	**15**	**13**	**8**	27	6	6	1
	4	26	28	12	14	6	30	4	7	1
	5	32	30	9	15	4	31	3	7	1

Table 3-7: GP1 Sensitivity for Material vs. Powder Depreciation Costs

			U_{max} = 10 reuses				**U_{max} = 30 reuses**			
			\$100/kg		**\$40/kg**		**\$100/kg**		**\$40/kg**	
	N (-)	**BVU (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**
G_1	1	5	6	17	3	8	**6**	**6**	**1**	**1**
	2	11	8	10	3	5	8	4	1	1
	3	16	**8**	**7**	**4**	**3**	9	3	2	1
G_2	1	6	4	11	2	5	**5**	**4**	**1**	**1**
	2	13	6	7	3	3	6	3	1	0
	3	19	**7**	**5**	**3**	**2**	7	2	1	0
	4	26	7	4	3	2	8	1	1	0
	5	32	8	3	3	1	8	1	1	0

Table 3-8: AlSi10Mg Sensitivity for Material vs. Powder Depreciation Costs

			U_{max} = 10 reuses				**U_{max} = 30 reuses**			
			\$152/kg		**\$60/kg**		**\$152/kg**		**\$60/kg**	
	N (-)	**BVU (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**
G_1	1	5	6	13	3	6	**6**	**5**	**1**	**1**
	2	11	8	9	4	4	9	3	2	1
	3	16	**10**	**7**	**4**	**3**	10	2	2	0
G_2	1	6	3	7	1	3	**3**	**3**	**1**	**0**
	2	13	5	5	2	2	5	2	1	0
	3	19	**7**	**4**	**3**	**2**	7	1	1	0
	4	26	7	3	3	1	8	1	1	0
	5	32	8	2	3	1	8	1	1	0

Table 3-9: IN718 Sensitivity for Material vs. Powder Depreciation Costs

			U_{max} = 10 reuses				**U_{max} = 30 reuses**			
			$192/kg		**$76/kg**		**$192/kg**		**$76/kg**	
	N (-)	**BVU (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**	**$C_{mat.}$ (%)**	**$C_{powder\ dep.}$ (%)**
G_1	1	5	12	32	7	18	**15**	**15**	**3**	**3**
	2	11	18	22	9	11	20	9	4	2
	3	16	**21**	**16**	**10**	**8**	23	6	5	1
G_2	1	6	9	20	4	10	**10**	**8**	**2**	**2**
	2	13	14	14	7	7	15	5	4	1
	3	19	**16**	**10**	**8**	**5**	18	4	3	1
	4	26	18	8	9	4	19	3	4	1
	5	32	20	6	9	3	20	2	4	0

For Ti-6Al-4V and a $680/kg feedstock with a maximum of 10 reuses, powder depreciation ranged from 8 to 42% of the total costs. At a build volume utilization between 16 and 19%, the material cost began to be greater than the powder depreciation costs indicating a build job that was more cost efficient than those built with single components. The effects of varying feedstock costs from $680/kg to $272/kg, reduced both the material and powder depreciation costs by nearly half. For feedstocks with a maximum of 30 reuses, powder depreciation ranged from 3 to 20% of the total costs and only a utilization of 5-6% of the build volume was needed for the material cost to be greater than depreciation. Consequently, for a lower cost feedstock, both material and depreciation costs were reduced by a factor of four.

GP1 showed a similar trend where a build volume utilization between 16 and 19% saw material costs being greater than powder depreciation for a 10 reuse feedstock and 5-6% for 30 reuse. However, the powder depreciation costs were relatively smaller and only ranged from 2-16% and 1-6%. This trend was continued in AlSi10Mg. Despite being a more expensive and less dense material than GP1, AlSi10Mg ranged in depreciation costs between 2-13% and 1-5%. Results for IN718 showed that powder depreciation costs ranged from 5-32% and 2-14%.

Additionally, the results exhibited the same behavior as Ti-6Al-4V where a feedstock at a reduced price of $76/kg nearly halved or reduced the material and powder depreciation cost by four times the value of a full priced feedstock.

To summarize, this sensitivity analysis explored the effect of increasing build volume utilization and changing the build material. In all models with a limit of 10 reuses, a build volume utilization larger than 15%, resulted in the material costs being greater than powder depreciation costs, regardless of the material type. Whereas, powders with a higher limit, 30 build cycles, only required at most 6%. Given that the powder depreciation cost is calculated as the difference between the mass of the un-melted powder and the mass of powder melted for the parts, a highly packed build tray with a high quantity and large mass of produced parts reduces the overall powder depreciation since more of the surrounding powder is melted and consumed during the build job. Less un-melted powder remains after the build, and thus, the material costs are greater than the powder depreciation; however, post-processing costs may increase as a result of the additional part quantities. Ti-6Al-4V and IN718 both showed depreciation costs ranging near maximums of 32% and 42%. However, GP1 and AlSi10Mg only had maximums of 13-16%. In the case of GP1, this material has a build rate that is the half the speed of Ti-6Al-4V and IN718; thus, it will have double the machine time costs. Coupled with a low feedstock price, GP1 shows a smaller value of powder depreciation relative to the other materials. For AlSi10Mg, its light tap and wrought density is also nearly half compared to the other materials, and therefore it will have nearly half the powder depreciation costs. Ti-6Al-4V and IN718 both exhibit high values of powder depreciation due to nearly similar build speeds, moderate powder densities, and high unit price. The largest effect of the total costs was the interaction between price of powder, going from the expensive to the economical price, and increasing reuse limit from 10 to 30 build cycles which reduces total costs by nearly a quarter. Given that each alloys has unique chemistry, reactivity, and functional requirements, the savings when a feedstock is permitted for up to 30 reuses may only be applicable to non-reactive alloys such as IN718 and GP1, as initially suggested by NASA.

3.3.4 Costing at Mass Production

The last sensitivity study was to determine how the cost model for powder feedstock depreciation would impact total costs for builds in mass production. Using geometries G_1 and G_2, costs were modeled for a production range of 1 to 1000 units with G_1 being limited to three parts per build platform and G_2 at five. All production was assumed to take place in series on one AM machine with no parallel production. For production requiring multiple build jobs, additional time for removing the platform, sieving the powders, and starting the next build was allocated to C_{setup} . Additionally, all $C_{substrate}$ charges for stress relief and wire-EDM was assumed to take place in series on a per build platform basis, with no parallel processing. For productions requiring multiple build jobs, powder mixing consisted of adding virgin powder to the feed bed with β_k equal to the mass percentage of the total amount of geometries in the build, relative to the mass of the loaded feed bed and selected material densities. When any powder exceeded the designated U_{max}, the lot was discarded and the next build was initialized with virgin powder. Similar to the study in Section 3.3.3, each production scenario was modeled by varying material selection and the cost of the feedstock at a baseline and discounted price. To aid in comparison, costs were simplified and modeled as the “normalized cost per part”, where the cost per part for a given production quantity and depreciation model was divided by the cost per part for a build with the same quantity when calculated in a traditional cost model assuming infinite powder reuse. The resulting costs are presented in Figures 3-10 and 3-11.

Normalized cost per part was largest for builds with one component of production, whereas the costs was lowest at a mass production of 1000 parts which is expected given that additional components aid in sharing costs for build file preparation along with labor for machine set-up and platform removal. The low-end “valleys” of the plots are associated with production requiring full build volume utilization for all the builds and thus minimal powder depreciation costs while, the rising “peaks” pertain to production requiring partial utilization which may have one unit printing in build job and therefore incurring excess powder depreciation costs. Between the range of 100 to 1000 units, the SOYD and SLN cost models began to converge to a constant value indicating a limit for how much cost could lower during high unit production. Depreciation models that had equal limits for U_{max}, converged to the same normalized cost per part but differed when the cost of the feedstock was either a low or high value. Regardless of the number of parts, all depreciation models converged to a cost that was larger than the builds with an infinitely reusable feedstock indicating that in mass production, builds assuming infinite reuse would always be undervalued by a given amount.

In Figure 3-10, builds for G_1 with Ti-6Al-4V and IN718 had the largest costs over builds with infinite powder, while GP1 and AlSi10Mg were the smallest. Production with Ti-6Al-4V priced at $680/kg had the largest range in normalized cost per part valued from “1.72x” to “1.15x” for builds with U_{max} = 10 build cycles, and “1.25x” to “1.05x” for builds with U_{max} = 30 build cycles, indicating that builds at high unit production were undervalued from 5-15%. Using a lower priced Ti-6Al-4V powder at $272/kg, the costs halved to ranges of 1.35x to 1.07x and 1.12x to 1.02x, respectively. Builds with the lowest normalized cost per part occurred with AlSi10Mg between 1.15x and 1.04x for powder with a reuse limit of 10 builds along with costs of 1.05x to 1.01x for powders with U_{max} = 30 build cycles.

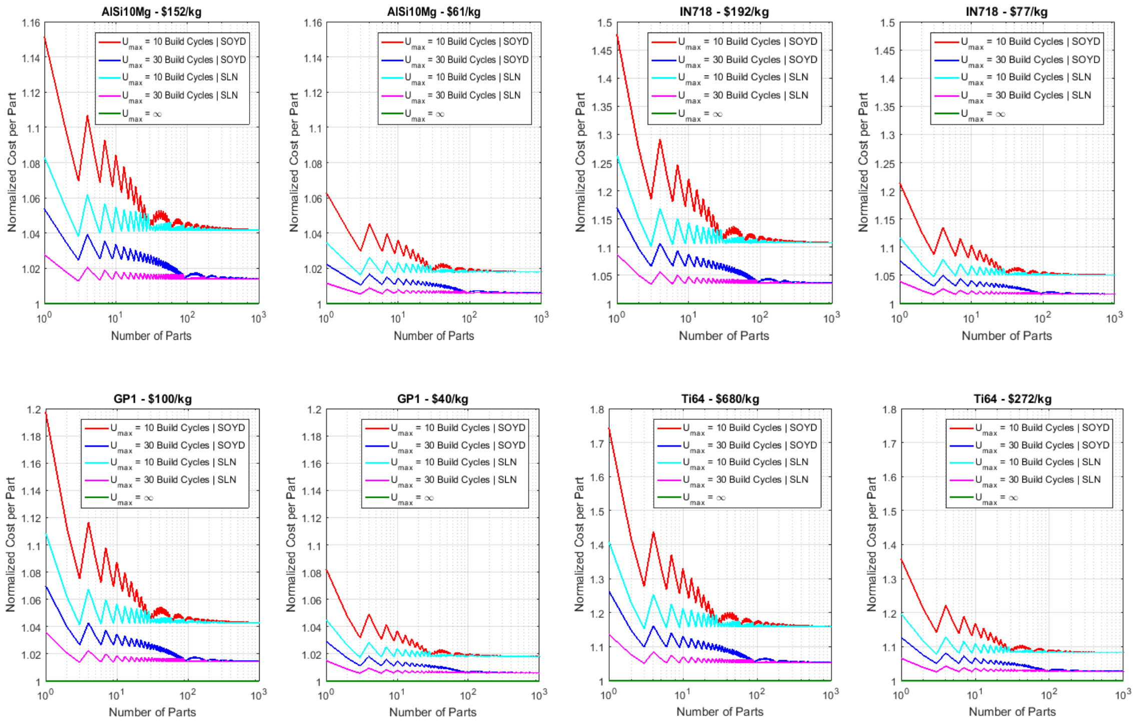


Figure 3-10: Normalized cost per part for production of G_{1}, Automotive Upright

Similar observations were found in G_2, where builds using Ti-6Al-4V and IN718 had the largest costs as opposed to those with AlSi10Mg and GP1. Builds with Ti-6Al-4V and IN718 ranged from normalized costs of “1.40x” to “1.05x” and “1.24x” to “1.04x” with powders limited to a reuse of 10 build cycles. However, lower priced feedstocks and builds of AlSi10Mg and GP1, at either full or a discounted price, all converged to normalized cost per parts that were less than or equal to “1.02x” indicating only a 2% undervaluing compared to builds assuming infinite powder reuse.

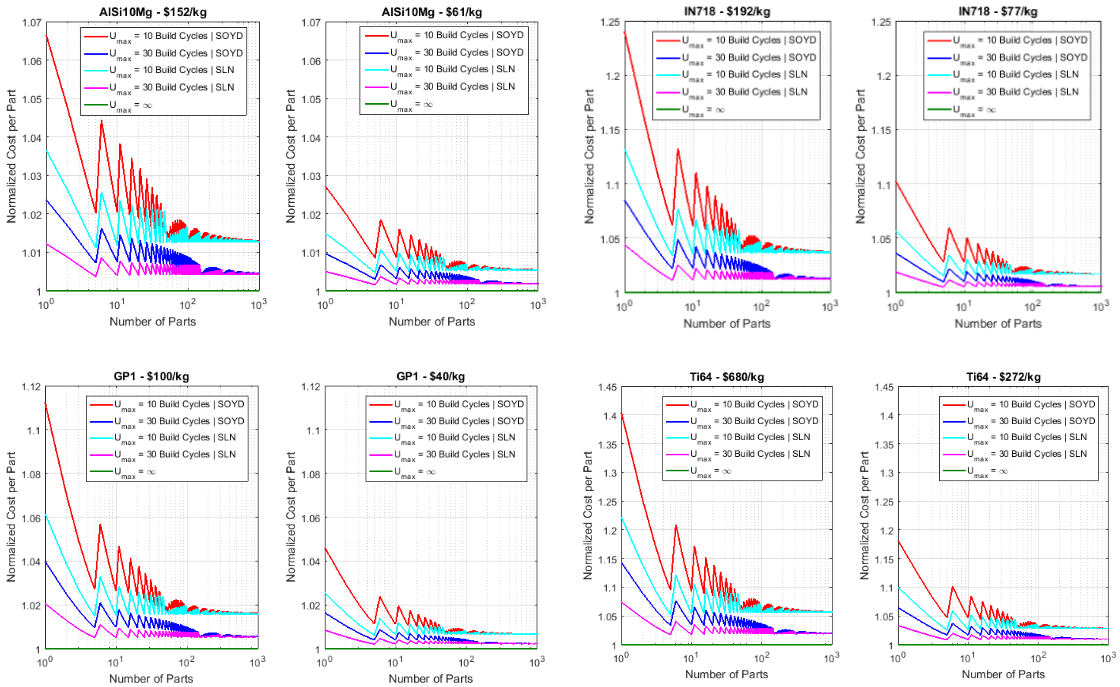


Figure 3-11: Normalized cost per part for production of G_{2}, Test Apparatus

The findings for this section are that builds with production less than 100 parts (33 and 20 build jobs for G_1 and G_2 respectively) had the largest normalized costs and were at most “1.75x” the cost of the builds with infinite reusable powder, due to high $C_{powder\ depreciation}$ when initializing a build with virgin powder. As production quantities increased, non-recurring expenses such as C_{prep} and $C_{build\ job}$ decreased over time due to a consistent geometry, build file, and a high number of replicates. Additionally, cost for $C_{material}$ decreased due to average price for the material depreciating over time based on the reuse limit for the feedstock. Repeating expenses that were shared for a build job such as C_{setup}, $C_{removal}$, $C_{powder\ depreciation}$, and $C_{mixing\ depreciation}$ caused cyclical increases in the cost due to additional activities of mixing virgin powder to the feed bed lot, reloading powder, and re-prepping the machine for the next build in production. While total C_{postp} increased with higher quantities of parts, on a per part basis the cost was constant since the time allotted to remove supports was consistent for each replicate.

Increasing the quantity of parts past 100, material selection, and decreasing the price of the powder had the largest impact in reducing the normalized cost per part. Ti-6Al-4V and IN718 had the largest normalized cost per part due to high price of their feedstock and their subsequently large $C_{powder\ depreciation}$ and $C_{mixing\ depreciation}$ expenses. Because G_1 and G_2 have differing build volume utilization, both converge to dissimilar normalized cost per parts, with G_1 reaching a value of “1.17x” compared to “1.06x” for G_2 in Ti-6Al-4V limited to 10 reuses. GP1 and AlSi10Mg had costs less than “1.02” due to, respectively, low powder cost and low tap densities leading to low powder depreciation costs. Based on these observations, the results suggest that cost modeling with powder reuse estimates an additional 10% or more in costs for productions below 100 units with high priced Ti-6Al-4V and IN718, and marginal amounts of additional costs for AlSi10Mg, GP1, or lower priced feedstocks in production with over 100 units.

3.5 Model Limitations

Uncertainties in the model are the exact number of reuses permitted for each material alloy, an open and active area of research in the metal AM community. The parametric analysis accounted for two different maximum build cycle limits, but literature and standards are limited that provide recommendations on the exact extent for which a reused metal powder can be used to produce AM parts in functional engineering applications (e.g., aerospace, biomedical, etc.). With the introduction of new LPBF technologies that enable the feedstock to be loaded one-time and never removed from the machine, this model may be limited in its applicability towards measuring the reuse and depreciation of the powder feedstock as it is recirculated in a LPBF machine.

The build time estimator was specific to an EOS machine and assumed a constant build rate which can fail to capture the dynamic nature of the LPBF process when it is performed on single and multiple geometries. Higher fidelity methods could consider the lasing system architecture (e.g., gantry, F-theta lens, multiple lasers), scan speeds for a given material parameter sets, hatching surfaces, upskin/downskin contours, and the estimated temperature for each voxel being melted. Despite the model considering the use of a lower price feedstock, their accessibility in industry is often limited to large manufacturing firms that purchase multiple tons of feedstock for serial production. Thus, their highly reduced prices may not be accessible to small service bureaus. While alternative feedstocks may be ordered from different providers, it also creates the risk of voiding machine warranty if improperly used.

The selection of a depreciation model is highly empirical and is material, product, industry, and business model dependent for a manufacturer. SOYD and SLN provide two different means of valuing parts that use a depreciating powder feedstock however, they add cost which impact overall cost effectiveness in low volume and serial production. Manufacturers focused on non-functional prototyping or those requiring less stringent properties may be better served assuming infinitely reusable powder. Meanwhile, manufacturers that have limited access to virgin powder or firms in highly regulated industries may be able use a cost model with finite powder reuse to allocate additional costs to the builds and better recuperate capital for discarding scrap powders at their reuse limit.

Post-processing accounted for the removal of support structures; however, functional components may call for more sophisticated processes such as annealing, shot peening, or CT (computed tomography) scanning in order to validate the integrity of the part. While these examples considered DMLS, costing for EBM will have differences due to variations in PBF technology and required labor activities. In addition, the empirical waste factors in calculating the part mass were specific to an EOSINT M280 and may not be applicable to other PBF processes. Finally, costs for the re-design and engineering of a pre-existing component in order to be made using AM were not captured in this model.

Having observed that LPBF costs can be undervalued when not accounting for reused powder, the next chapter shall expand upon these implications by examining how these cost estimates can be integrated with a DFAM tool and also highlight how the build geometry, orientation, and support structure specifications by the designer can impact cost.

Chapter 4
CAD-Integrated Cost Estimation and Build Orientation Optimization

The findings from Chapter 3 showed that total cost and powder depreciation vary from component to component based on various parameters related to the design of the geometry, powder feedstock, maximum permitted reuses, and quantity of replicates for the build. However, geometry definition and build orientation selection is an iterative process that can yield numerous costing scenarios. To support designers, a CAD-integrated cost estimator was developed as a software plug-in for a commercial CAD program to enable early geometry evaluation for LPBF and to better capture the influence of machine and design parameters on the overall costs for a given design. This chapter provides an overview of the system architecture, coding approach, and graphical-user interface (GUI) used in preparing the tool. The plug-in was programmed in SolidWorks and demonstrated on an automotive upright. Sensitivities studies are presented along with comparison to a comparable commercial tool, 3DXpert.

4.1 Program Workflow

As discussed in Section 2.2.3, few scholarly works have focused on CAD-integrated DFAM tools and cost modeling for end-use metal AM components. To fill this gap, we propose the method shown in Figure 4-1 for estimating costs for part production in a generic metal AM system, while modeling in a generic CAD program.

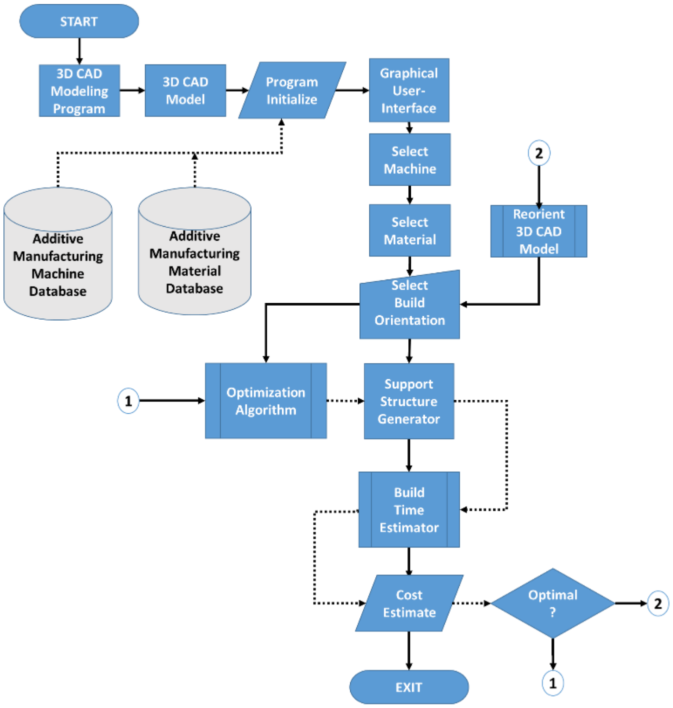


Figure 4-1: CAD-Integrated Cost Estimation Framework

As seen in the figure, the workflow begins with the designer creating a 3D CAD model and then activating the cost-estimation program directly within the CAD software. After initialization, a graphical user-interface (GUI) is opened, consisting of multiple tabs for users to explore various AM processes. In this work, the AM process is assumed to be a nonspecific LPBF machine. Similar to the IMDI from Rosen et al. [68], upon selecting a process, users are presented with information regarding their part’s volume, bounding box, build height, and dimensions relative to the build volume of the AM machine. The coordinate system is aligned with the CAD system and assumes the 3D model’s build height is along the Z-axis. Users select an AM machine and are then prompted to select a material for producing their part. The specific machine and material data are loaded from an external database connected to the cost estimation program. The material’s sub-section in the user-interface provides information on the powder feedstock cost, tap density, wrought density, layer thickness, and build rate for the material. Upon selecting material, the user-interface updates to provide users with the number of layers to produce their component and the estimated part mass after printing. For simplification, we consider all layers to be of uniform thickness.

After a material is selected, the user navigates to the build parameters subsection in the GUI and inputs a value for the charge amount, (i.e., dosage). The program outputs an estimate for feed bed mass, M_{FB}, calculated using Equation 5 from Chapter 3. Next, the user inputs waste factors for process inefficiency (α) and powder trapped in support structures (γ) leading to an output for the component mass, M_i, calculated via Equation 9 from Chapter 3. Finally, in the build parameters subsection, the user specifies the hourly machine rate, C_{mach}, for their AM machine, the cost for inert gas usage is material-dependent and assumed to be included in the machine rate.

4.2 Support Structure Generation

After specifying the build parameters, users specify the orientation for building their part. Each orientation selected prompts the GUI to update, indicating changes to build height, total bounding box, and subsequent changes to mass of the feed bed required to execute the build job. Upon reaching a satisfactory orientation, the user can then start the support structure generation module by specifying the minimum support angle. This angle is the critical angle at which a face can be built at an incline to the build surface and not require support material. The value can vary from between 30-45° and depends on the AM process [92, 110]. To generate support structures, our framework uses a ray tracing method, similar to the one proposed by Allen and Dutta. Despite the availability of more sophisticated techniques such as voxelization, octree-representation, and Direct CAD slicing [87, 111, 112], ray tracing was selected due to its efficiency to integrate into CAD.

The ray tracing is conducted on a uniformly spaced grid based on the dimensions of the selected AM machine's build plate. To initialize the ray tracing, the user specifies a number of grid partitions (see Figure 4-2). The points along the partitions are selected as the starting point, and a ray is transmitted in the direction of the body. The normal vector is returned from the body's surface at all intersecting points. The difference between the surface normal angle and the build direction is calculated to determine if the vector is above or below the user's designated support angle. If the angle is less than the minimum support angle, then the coordinates for the point of intersection are stored in an array, and the ray trace continues until the algorithm has traversed the bounding dimensions of the part.

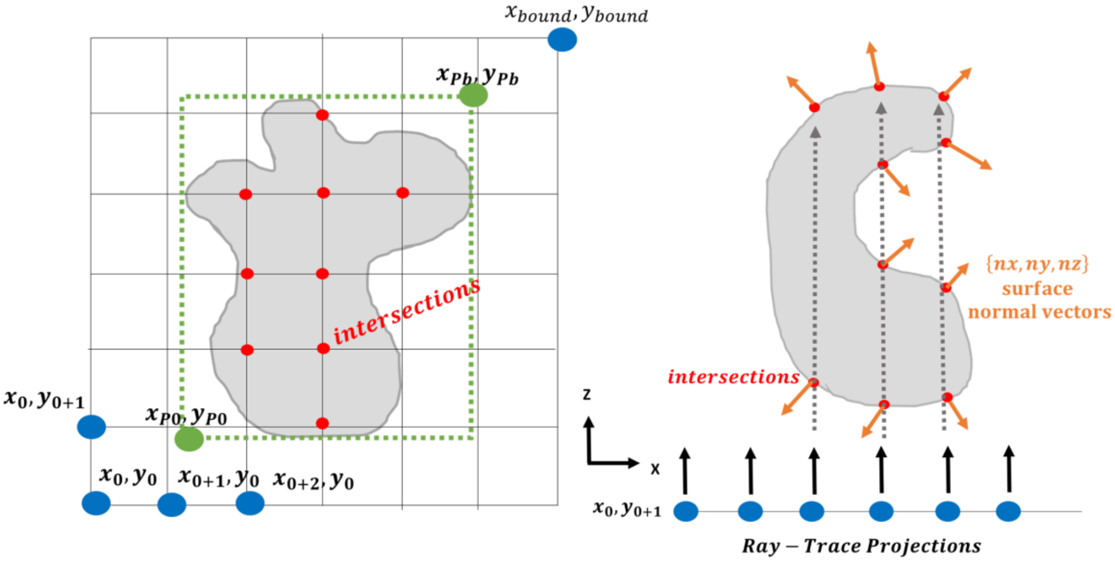


Figure 4-2: Top-view of component undergoing ray trace with 6 grid partitions (left) geometry intersections (right)

Upon completion of the ray trace algorithm, the coordinates from the arrays are used to form geometric extrusions for all identified overhanging surfaces. Extrusions that connect from the body's surface to the build plate are labeled as external support structures, and extrusions that connect the body's surface to another surface on the body are referred to as internal support structures. To prevent overlapping bodies, all extrusions are generated with standard geometric cross-section shapes that fit within the grid (partition) spacing of the ray traces. The part now has all of the required support structures from which a volume is determined for estimation purposes.

After generating the support structures, the program calculates and returns an estimated cost for producing the component. Since Chapter 3 demonstrated that machine cost, material cost, and feedstock depreciation account for over 60% of the total cost in LPBF, these were the only costs calculated in the program due to their inherent variation with the CAD geometry, build orientation, and support structures. C_{mach}, $C_{material}$, $C_{powder\ depreciation}$, $C_{mixing\ depreciation}$, and T_{build} were all calculated using their respective equations from Chapter 3.

4.3 SolidWorks API Programming

To execute the program framework, a software plug-in was programmed in SolidWorks 2015. SolidWorks was chosen because of easy access to their Application Programming Interface (API) and online reference library of commands and functions [113]. The SolidWorks API consisted of a Visual Basic for Applications (VBA) development environment where the tool was programmed and executed. A screenshot for the GUI is shown in Figure 4-3 along with a terminology for the CAD geometry. Upon initialization, a macro loads the AM machine and materials data from a local directory on the host computer. The CAD model’s bounding box coordinates are gathered using the *GetPartBox* function. The volume is queried from an assigned ModelDoc2 object using the *CreateMassProperty* function.

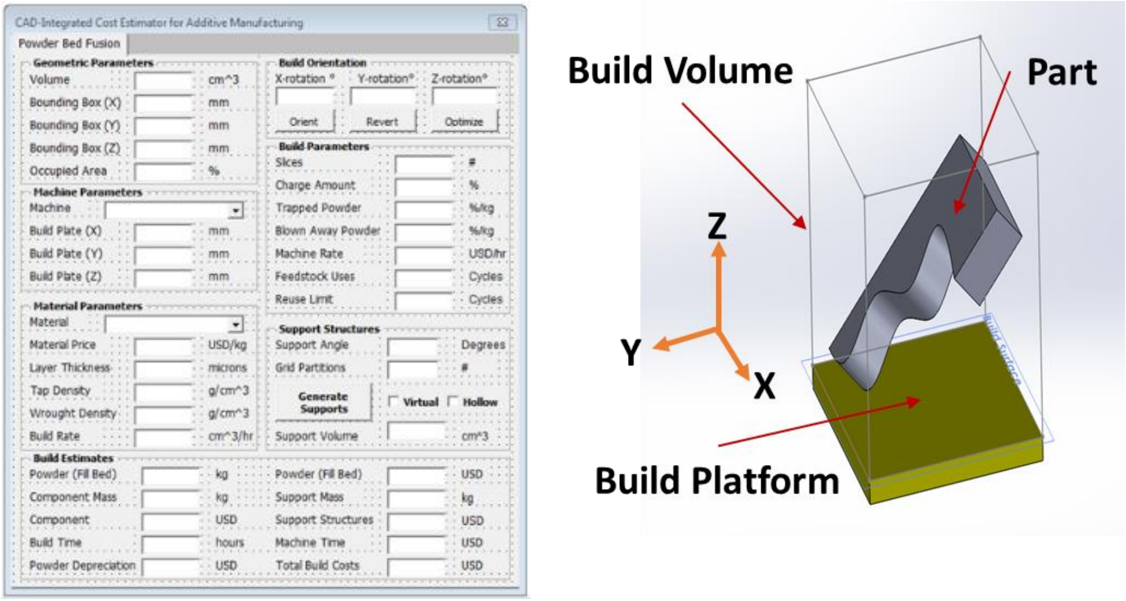


Figure 4-3: Cost Estimator GUI (left) and CAD Geometry Visualization (right)

Upon selecting a machine, a macro constructs the build bed volume using the previously acquired perimeter coordinates with the *CreateLine2* command. It then produces a reference plane for the build surface through the *CreatePlaneThru3Points3* corresponding to the lowest Z-coordinates of the part bounding box. For visualization purposes, the build platform is generated as a solid-body through a *FeatureManager* object and the *FeatureExtrusion2* command. The build orientation is controlled through the *InsertMoveCopyBody2* function where the user specifies a relative rotation about one of the coordinate axes.

During support structure generation, the *RayIntersections* command takes a base point and projects it along a vector (i.e., build direction). The ray trace returns a value corresponding to the number of entries and exits of the target geometry. If an intersection occurs, then the outward normal vector is queried from the CAD model's surface at the point through *GetRayIntersectionsPoints*. The angle between the surface normal and the build direction is calculated using the arccosine relation. Support structures are then created using an *IModeler* object by forming geometric primitives on the build surface or body below an overhang, through the *CreateFeatureFromBody3* command. In this case-study, primitives were chosen to be solid-body cylinders due to their low memory requirements, simple programming, and minimal input definitions.

As shown in Figure 4-4, support structures are color-coded with teal and purple corresponding to internal and external support structures, respectively. Computational performance is improved by turning off the SolidWorks graphics update, removing dynamic highlighting, and running support structure generation in the background.

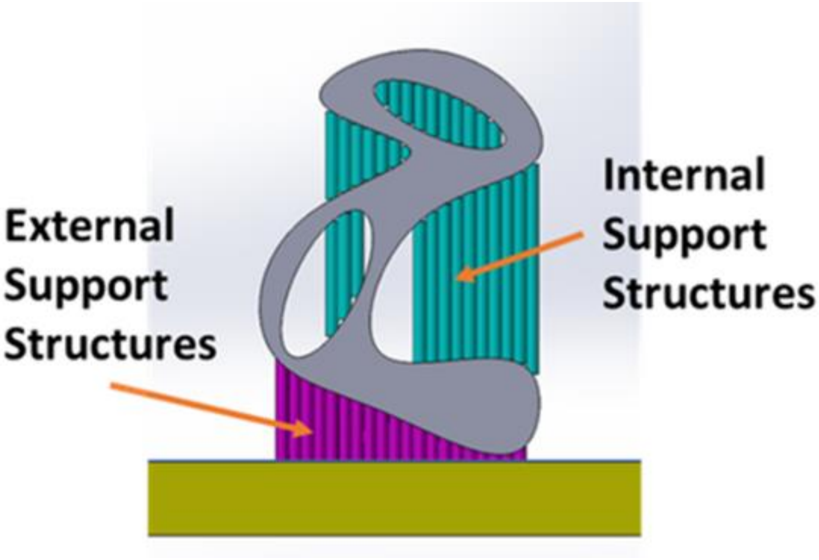


Figure 4-4: Geometry with overhanging surfaces

4.4 Build Orientation Optimization

Particle Swarm Optimization (PSO) was used in searching for an optimal build orientation and was selected due to its convergence speed and computational efficiency compared to genetic algorithms and robustness in finding global minima compared to gradient descent methods [114]. The design variables were the relative rotations of the CAD model about the X and Y axes. The constraints ensured that the CAD model did not rotate into an orientation where the part geometry extends outside the build volume.

To begin, the user specifies an objective function to minimize (i.e. internal support volume, external support volume, build time). Upon randomly generating a starting population of candidate orientations, particles change their position and trajectories to align with the particle with the best objective value. Each particle varies position through a velocity function based on their current motion, memory influence, and swarm influence. The algorithm converges upon reaching a tolerance specified by the user. More details on the PSO algorithm can be found in [114].

4.5 Macro Implementation and Examples

To demonstrate the plug-in's capabilities, costs were evaluated for the automotive upright, G_1. This geometry was selected because of its complex geometry with numerous overhanging surfaces and its function as an end-use metal AM component. Previous work by Maranan *et al.* [115] found a tradeoff between manufacturability and costs when printing this design. They decided to select the orientation shown in Figure 4-5 because it was easier for accessing and removing support structures even though it required a considerably longer build time than lying the part flat on the build plate. Consequently, the resulting support volume was 202.4% greater than the build material. To match their study, production is modeled for an EOSINT M280 DMLS machine with a machine rate of $72/hr, assumed to be inclusive of machine depreciation, overhead, electricity, and inert gas. The build material is assumed to be EOS Ti-6Al-4V powder processed at 30 micron layer thickness, with a build rate of 13.5 cm^3/hr and a value of $680/kg for virgin powder. The results from applying the macro to this part are discussed next.

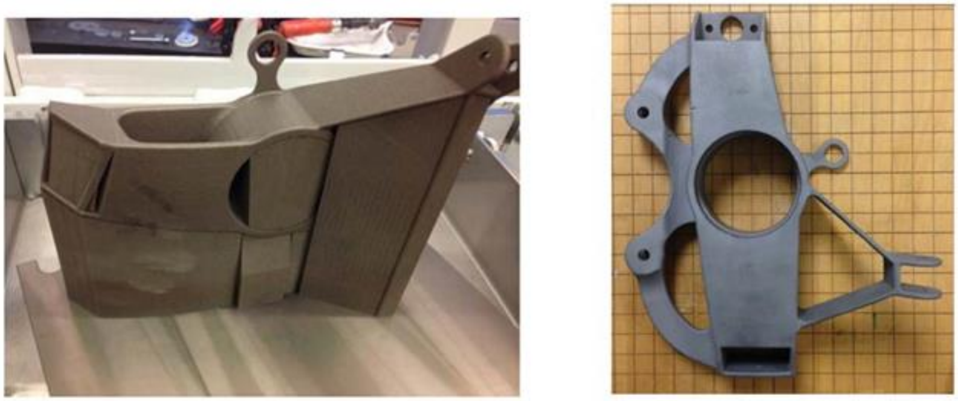


Figure 4-5: Automotive Upright, Magics model with supports (Top), Printed DMLS component (bottom) [115]

4.5.1 Parameter Sensitivity on Cost Estimate

In order to properly implement the plug-in, a sensitivity analysis was first conducted to determine how much error could result from incorrectly defining build parameters during cost estimation. The support angle, the maximum angle from the horizontal where support structures are required for a surface, was selected due to its wide variation among manufacturers [116]. The support angle was parameterized and varied between 60, 45, 30 degrees. Steep support angles at 60 degrees and above are conservative and are selected to help ensure the production of component, whereas shallow angles near 30 degrees are aggressive and are used to save costs (material and machine time). The diameter of the support structures was varied from a coarse diameter (10mm) to fine (0.625 mm) to determine the effects of inaccuracy and poor resolution. Figure 4-6 shows an example of the CAD model for the upright with the generated supports; the results are summarized in Table 4-1.

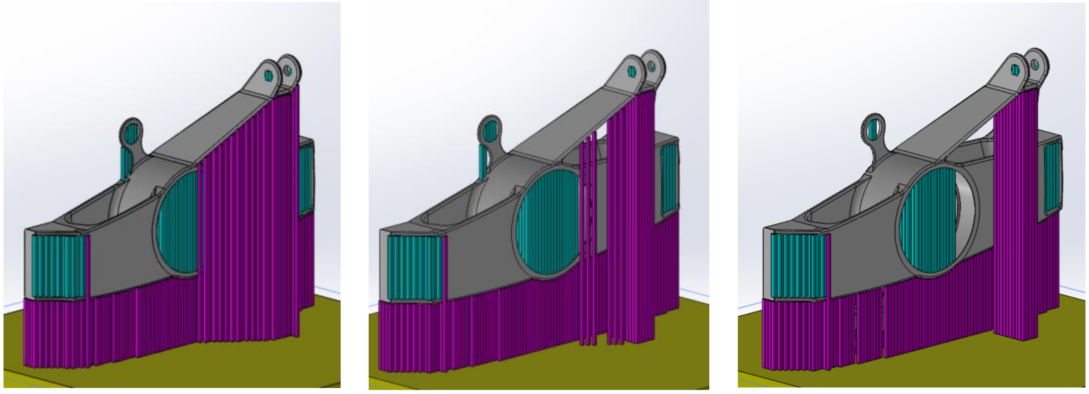


Figure 4-6: Automotive Upright with Support Angle: 60° (Left), 45° (Middle), 30° (Right)

Table 4-1: Support Diameter vs. Support Angle

Support Diameter mm	Support Angle: 60°		Support Angle: 45°		Support Angle: 30°	
	Support Volume cm^3	Build Time H	Support Volume cm^3	Build Time h	Support Volume cm^3	Build Time H
10.00	398	56	230	45	191	42
5.00	411	57	268	48	222	45
2.50	449	60	298	50	238	46
1.25	447	60	276	49	230	45
0.63	439	59	281	49	230	45
Mean (SD)	429 (20)	58 (2)	271 (23)	48 (2)	222 (16)	45 (1)
Cost (SD)	\$1290 (60)	\$4180 (140)	\$810 (70)	\$3460 (140)	\$670 (30)	\$3240 (70)

*Results from Maranan *et. al*: Build Time: 54 hours, Support Volume: 410 cm^3

Maranan *et al*.'s automotive upright cost \$3890 in machine time, \$520 for the direct part, and \$1480 for the support structures [115]. Costs for powder and mixing depreciation were not reported in their study. The analysis revealed that the cost estimate was most sensitive to the support angle. *These results indicate that: underestimating the support angle by 15 or more degrees can under-predict support material costs by 34% and build time by 20%.* This is because as the support angle becomes steeper, more surfaces on the geometry are treated as overhangs and thus, require a larger volume of support structures, which uses more material and build time. The coarse and fine support diameter size showed marginal variation as the resolution increased. With metal AM showing promise towards geometrically complex designs, a conservative support angle is likely to be appropriate in accounting for thermal deformation and curling concerns.

This first case study found that the accuracy of the macro is influenced by the support angle, which dictates the volume of support structures generated for the 3D CAD model. The resolution of the support diameter showed little significant variation with the estimated build time and support volume. The lack of variation is due to over-estimation by the coarse-sized support structures. Given the complex geometry of this component, the results will change when considering multiple orientations where over-estimation at low-resolution may not be as precise as higher resolution estimates.

4.5.2 Consideration for Multiple Replicates

Although the upright was built alone in the AM process, this practice is not common in industry. Often, multiple components are built on the same platform to save costs as demonstrated in Section 3.3.3. The manner at which to arrange components to minimize feedstock depreciation and post-processing costs is not clear from literature and is geometry dependent. To further investigate this relationship, the flat and vertical oriented uprights were modeled in the macro as shown in Figure 4-7. The ray trace was executed with a support structure diameter of 2.5 mm and a support angle of 60 degrees. Results are given in Table 4-2. The dosage is set to 2.25 per layer thickness [55]. Recyclability for the Ti64 powder is presumed for a maximum of 15 build cycles [68], with a salvage value of zero. Waste factor for process inefficiency, α, is assumed to be 40% and trapped powder is set to 25% based on reported results [24].

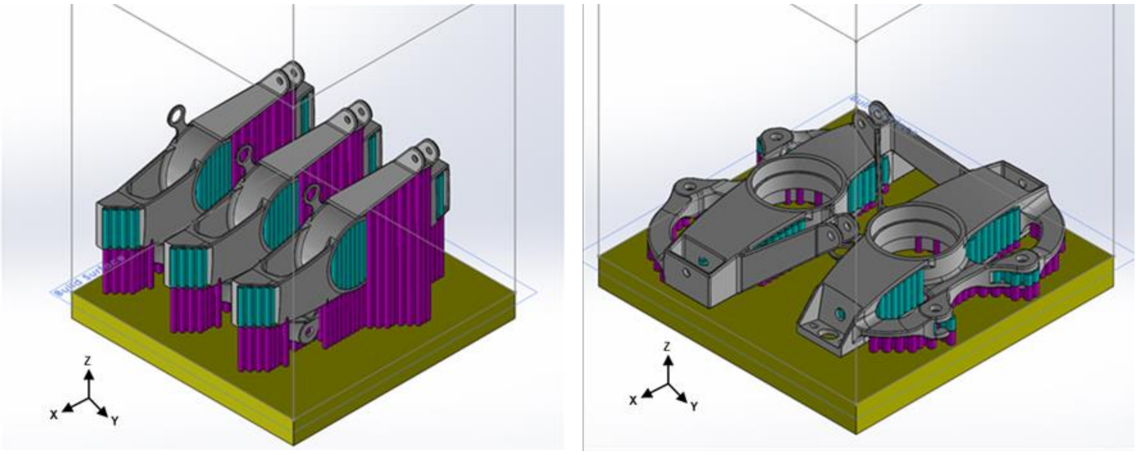


Figure 4-7: Vertical (left) and Flat (right) Orientation

Table 4-2: Orientation vs. Part Count vs. Cost

Variables		Vertical Orientation		Flat Orientation	
Part Quantity	-	1	3	1	2
Build Time	H	60	151	29	52
$C_{machine}$	$	4320	10870	2050	3670
$C_{material}$	$	2660	7980	1380	2750
$C_{powder\ depreciation}$	$	4850	4250	2040	1870
Cost per part	$	11,830	7700	5580	4150

Table 2 shows that the longest build time was for the vertical orientation with 3 uprights at 132 hours. The shortest build time was for the flat orientation with a single component at 29 hours. The flat orientations had the lowest amount of support structures and therefore had lower material costs, compared to the vertical orientations. Powder feedstock depreciation costs were higher in the vertical builds due to their tall build heights and more powder required to the fill the powder bed. *Consequently, for builds with a single component, the depreciation of the un-melted powder was 1.4x to 1.8x the material cost.* These qualities made the vertical build with one component as the most expensive build at $11,830 per part. The lowest cost component was the two flat uprights due to their relatively low machine and material costs. However, as noted by Maranan *et al.*, the flat orientation is not feasible due to the presence of internal support structures that are difficult to remove during post-processing.

Having examined the influence and costing implications for multiple components when including depreciation costs for recycled powder, this section revealed that a flat orientation with two components was the most economical; however, the manufacturability of the orientation was not feasible, as supports could not be accessed for removal. The vertical orientation was feasible for producing 3 components but at a cost increase of 66% compared to the flat orientation. Both orientations showed that builds with a single component were uneconomical due to high powder depreciation costs being nearly twice the cost of the build material consumed in the part. The findings suggest that an optimal build will encompass a higher utilization (efficient packing) of the build volume with more components, which will increase build time a new design tradeoff found in metal AM against powder depreciation costs.

4.5.3 Build Orientation Optimization

To further investigate the role of orientation on costs, the upright was optimized in PSO. The objective function was to minimize internal supports to reduce the amount of inaccessible supports during post-processing. Each PSO model was conducted with a population of 15 particles and ran for a total of 40 iterations. Convergence was determined when the deviation in the objective function value was equal to a tolerance of zero after 20 iterations. To improve run-time, the support structure volume was calculated analytically for each orientation through the numerical data provided from the ray-trace algorithm. Upon convergence, the geometric primitives for the supports were generated for the output orientation and then validated with the queried data from the CAD model. Figure 4-8 shows the optimal two orientations found by PSO with the full results in Table 4-3.

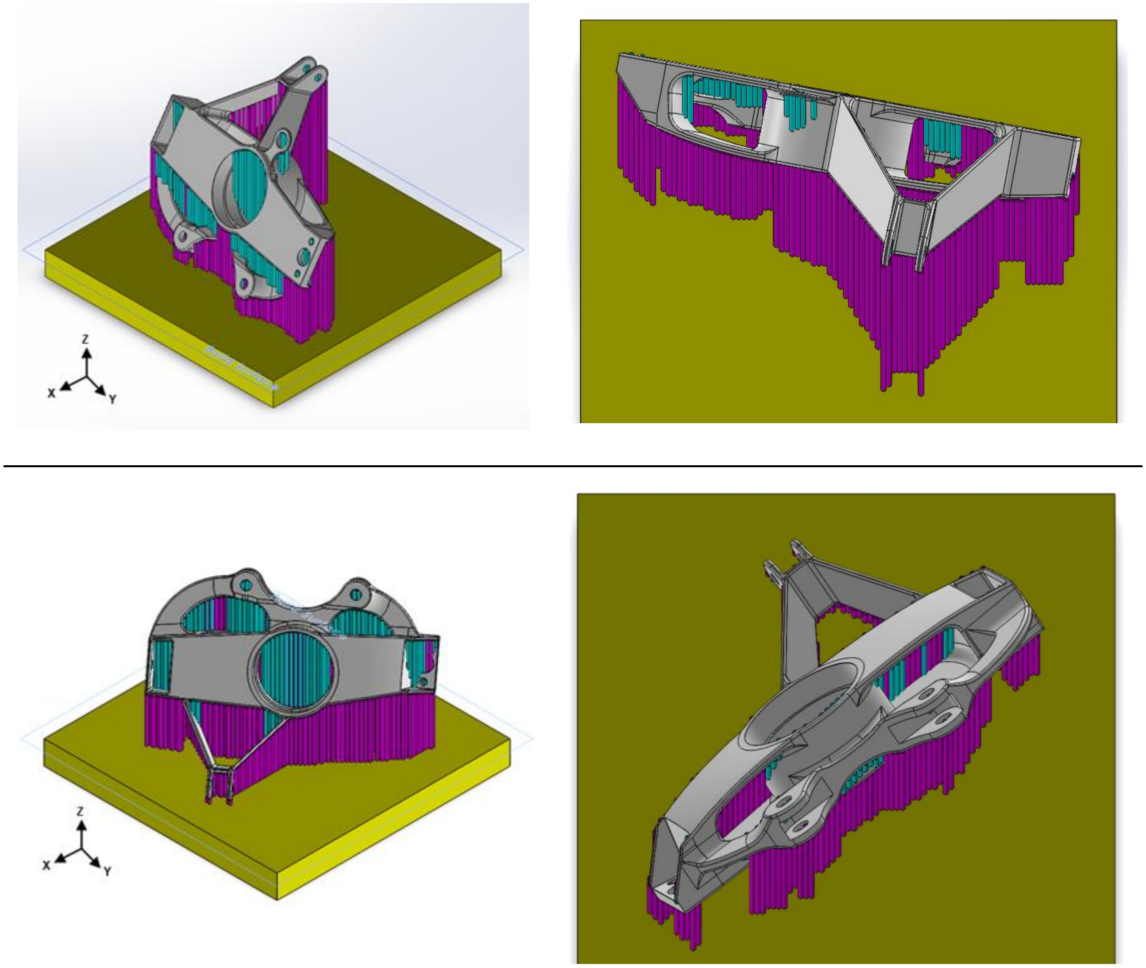


Figure 4-8: Optimal orientations to reduce internal supports. Orientation A (top) and Orientation B (bottom) with isometric and top views – rotated to show internal supports

Table 4-3: Optimization Results

Orientation	Build Time H	Internal Support Volume cm^3	Total Support Volume cm^3	Depreciated Powder kg	Cost per Part $
Flat	29	106	165	24	5580
A	54	43	422	45	9560
B	57	34	443	41	9940
Vertical	60	116	423	56	11,830

The most economical orientation was the flat with a cost of \$5580 per part. However, in this orientation 64% of all support structures were internal. Using the two orientations A and B from the PSO, the internal support volume decreased to 7-10% of the total supports. Subsequently, the total costs increased due to changes in build time and depreciated powder. In comparison to the vertical orientation selected by Maranan *et al.*, the PSO generated orientations reduced total cost by 12-16% due to lower powder depreciation costs. The results suggest a tradeoff between total cost and internal support structure volume.

4.5.3 Software Benchmarking

The last study compares the results estimated by the macro with estimates from 3DXpert, a commercial 3D CAD software offering similar capabilities for geometry assessment and build planning for AM. Both programs allow users to position and orient geometries on the build platform for a selected AM machine. Both provide tools for orientation optimization of the geometry based on given objective criteria and support structure generation at various overhang angles. Lastly, both provide estimates of build time, material, and machine cost based on user-defined parameters. One distinct feature of 3DXpert is its library of support structures primitives and AM machine build spaces, offering additional options for customization to aid the designer. To compare the predictive capability of the CAD-integrated cost estimator's algorithm with 3DXpert, the studies from Section 4.5.2 and 4.5.3 were re-run and compared between the programs using the same geometry and same build parameters as before with virgin Ti-6Al-4V powder, \$680/kg, 9 seconds of recoating time, 60 degree support angle, and 13.5 cm^3/h build rate. To aid in comparison, $C_{material}$ was the direct material cost and factors for lost powder during the process and from support structures were not calculated. Both programs optimized the build orientation by using the objective of minimizing internal support structures. Images and tables comparing the two are shown in Figure 4-9 along with Tables 4-4 and 4-5.

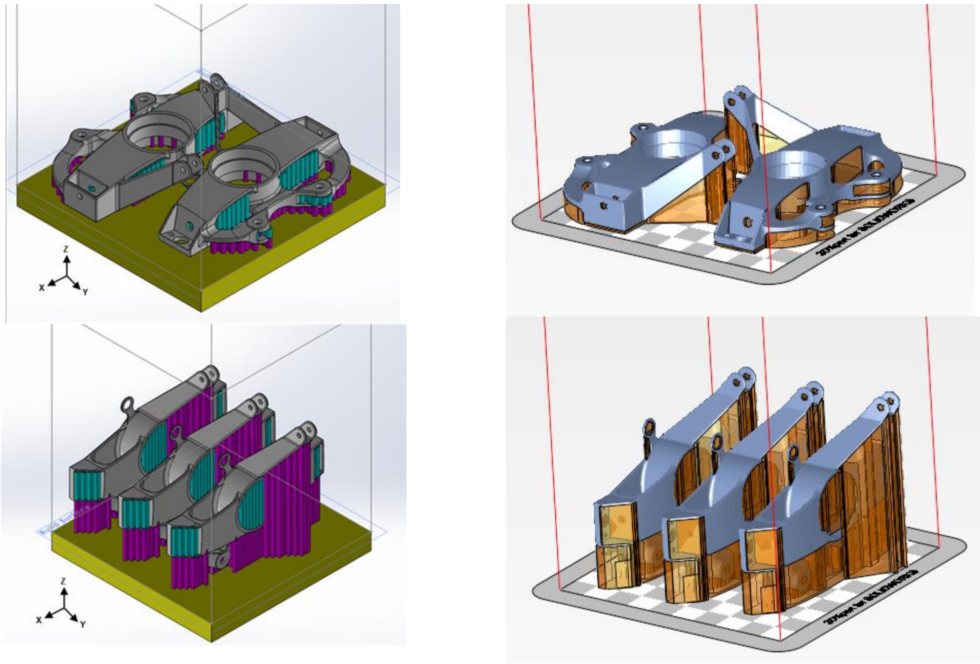


Figure 4-9:Support Structures Generated, CAD-Integrated Cost Estimator (left), 3DXpert (right)

Table 4-4: Support Volume vs. Orientation vs. Costs Comparison

	Orientation	Internal Support Volume	Total Support Volume	Build Time	$C_{machine}$	$C_{material}$*
		cm^3	cm^3	H	\$	\$
CAD-Integrated Cost Estimator	Flat	106	165	29	2090	1000
	Optimal	34	443	57	4100	1830
	Vertical	116	423	60	4320	1770
3DXpert	Flat	125	206	32	2320	1120
	Optimal	40	432	55	4020	1793
	Vertical	113	519	65	4700	2040

*Not accounting for powder losses due to filters or supports

Table 4-5: Replicates vs. Support Volume vs. Orientation vs. Costs Comparison

	Orientation	Replicates	Total Support Volume	Build Time	$C_{machine}$	$C_{material}$ *
			cm^3	H	\$	\$
CAD-Integrated Cost Estimator	Flat	1	165	29	2090	1500
	Flat	2	330	52	3670	3000
	Vertical	1	423	60	4320	1770
	Vertical	3	1269	151	10870	5310
3DXpert	Flat	1	206	32	2320	1120
	Flat	2	413	62	4450	2224
	Vertical	1	519	65	4700	2040
	Vertical	3	1556	167	12000	6170

*Not accounting for powder losses due to filters or supports

Results from Table 4-4 indicate that for the flat orientation, the cost estimator calculated 165 cm^3 of supports with 3DXpert at 206 cm^3 amounting to 20% discrepancy between the two estimates. These differences are attributed to 3DXpert using solid supports for the full volume under an overhang region versus the cost estimation tool using 2.5 mm diameter cylindrical supports to approximate the support volume. The build time was estimated at 29 hours versus 32 hours resulting in a 10% discrepancy, also attributed to differing estimates in the support structure volume. Overall, the material cost and machine time estimates between the two programs varied by a most 11% showing a moderate level of agreement.

Despite differing support generation methods, when tasked with independently finding the optimal build orientation for reducing the number of internal supports, both programs converged to similar solutions as shown in Figure 4-10. The plug-in estimated 34 cm^3 of internal supports which was smaller than the 40 cm^3 determined by 3DXpert; however, for the overall support volume, both showed strong agreement with only a 3% discrepancy due to slight variation in the part orientation and the resolution of the support structures.

Similar trends from the flat orientation estimates were seen in the vertical orientation and also the replicates builds in Table 4-5, where discrepancies in the estimated support volume were 20% and 10% for the build time. However, for the vertical orientation with one component, Maranan et al. reported that the build time was 54 hours and the support volume: 410 cm^3 thus, both programs were overpredicting the support volume and build time by 3-26% and 11-20% respectively. Based on the observations from Section 4.5.1, the support volume error is a result of differences in the support angle which was not reported by Maranan *et al.* along with build time being overgeneralized through the use of an average build rate.

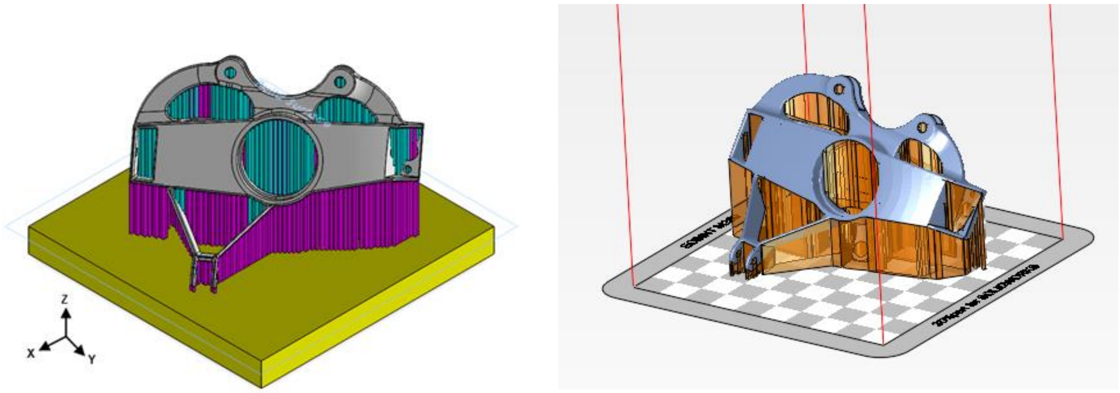


Figure 4-10: Optimized Solutions for Reducing Internal Support Structures

4.5.4 Limitations

One limitation to the proposed macro is the computational efficiency of generating the support structures. With run times near 1300 seconds for a high resolution of supports with 0.5 mm diameter, the flexibility to iterate from different oriented designs and generate the feature body is limited. The support structures were extruded from points to points, and therefore they do not directly intersect the surface, leading to gaps and approximation errors. The ray trace grid was also susceptible to aliasing due to a fixed and non-rotating coordinate system aligned with the build platform and not the 3D model. The methods used in the macro were selected due to ease of implementation and may not be as efficient and robust as those demonstrated in literature [117].

This discrepancy was apparent when comparing the CAD-integrated cost estimator to 3DXpert, where the cost estimation tool underpredicted support volume by about 20% highlighting the imprecision of the ray trace projection method and limitations of using cylinders as a support structure primitive. Both tools used an average volumetric build rate and a recoating time to estimate build time, but both showed an error of 11-20% compared to the reported build results. As mentioned in Section 3.4.1, improvements can be found through higher fidelity build estimators accounting for the geometry, machine parameters, and processing physics or potentially through empirical models based on regression analysis and statistical studies similar to those conducted by Rickenbacher *et. al*.

For a single component, costs for machine time, material, and feedstock depreciation were predicted to be $11,830 and $5580 based on orientation. However, an online quote from Sculpteo, an AM service provider, estimated total costs as “$4260” [118]. It was unclear whether their quote included overhead, labor, margin or whether they were using highly reused powder or assuming steep support angles.

In all, this chapter contextually emphasized metal AM; however, the implemented macro was only metal AM specific in regards to the cost model. The demonstrated CAD-integrated cost estimation approach can be generalized to any AM process requiring support structures. Additional tailoring for metal AM would have to account for surface roughness and thermal distortion inherent to the given material, build orientation, along with the post-processing required for finishing the part.

Chapter 5
Conclusions and Closing Remarks

5.1 Thesis Summary

This thesis examined the cost effectiveness of AM by studying how powder feedstock reusability can impact total costs in LPBF while integrating the proposed model within a CAD-integrated design tool. Sum-of-the-Years Digits and Straight Line depreciation were introduced to model the financial value of powder feedstock as it was reused in LPBF. Equations for $C_{powder\ depreciation}$ and $C_{mixing\ depreciation}$ were presented to allocate costs for the un-melted powder in the LPBF process and costs for blending reused and virgin powder lots. Cost was modeled for an automotive upright and test apparatus component. The feedstock was modeled as having 10 and 30 build cycles of reuse and compared against a traditional cost model that assumed infinite powder reuse. A sensitivity study was conducted to examine how build volume utilization can decrease $C_{powder\ depreciation}$ and increase $C_{material}$. Costs was also modeled for the mass production of 1000 units and compared for the depreciation models using a "normalized cost per part" against a traditional cost model.

Following the demonstration of the powder depreciation cost model, a CAD-integrated design tool was developed to apply the proposed cost model into a software plug-in embedded in a 3D CAD program. The plug-in modeled the build volume and position of the component on the substrate for a LPBF machine. The tool's GUI provided options for querying geometric data, specifying build parameters, generating support structures, optimizing build orientation, and conducting a simplified cost estimate. Ray-trace projection and grid partitions were used to identify intersections on the build geometry and generate internal and external support structures. A case study was presented using the automotive upright to demonstrate the capability of the plug-in. Sensitivity analysis was conducted by varying the support angle, orientation of parts on the build platform, and optimizing the build orientation by searching for the orientation that reduced internal support structures through particle swarm optimization.

5.2 Contributions

The key contributions of this work include the SOYD and SLN depreciation cost models for valuing a powder feedstock as it is reused in LPBF, the equations for $C_{powder\ depreciation}$ and $C_{mixing\ depreciation}$, along with the results from the case studies. The most prominent results were that traditional cost models with unlimited powder reuse undervalued build jobs with virgin powder by 13-75% in Ti-6Al-4V and 3-11% for GP1. Upon exceeding 13 build cycles in Ti-6Al-4V, 11 for GP1, the total costs for build jobs achieved a cost savings of 38% or 5% when using a highly reusable powder or powder feedstock exceeding U_{max}. Regardless of material type and geometry, it was found that 16-19% of the build volume utilization was the minimal quantity needed in order for $C_{material}$ to exceed $C_{powder\ depreciation}$. In mass production, high priced materials such as IN718 and Ti-6Al-4V can produced “normalized cost per part” that are 10% or more than traditional cost for productions with less than 100 parts (i.e., 33 and 20 build jobs respectively). In constrast, builds with lower priced feedstock such as AlSi10Mg and GP1 converge to costs that were only “1.04x” or “1.02x” traditional costs after 1000 parts.

Additional contributions include the framework and customizable code for a CAD-integrated cost estimator in SolidWorks 2015. Key findings were that the cost estimate was sensitive to the support angle, which could over-underestimate support material costs by 34% and build time by 20%. The resolution and diameter of the support structures showed no notable impact on the accuracy and precision of the cost estimate. A build with two parts oriented flat on the build plate had the lowest cost at $4150 per part. The orientation was infeasible, but the production of vertical components provided an alternative at $7700 per part. The optimal build orientation found from particle swarm optimization reduced the internal support volume from 116 cm^3 to 34 cm^3. Benchmarking with 3DXpert revealed 20% discrepancy in the support volume estimation and 10% discrepancy in the build time estimate but only 3% discrepancy for the support volume and build time for the optimized orientation that reduced internal supports.

5.3 Highlighted Limitations

For cost modeling with reused powder feedstock, it is uncertain what is the proper reuse limit and metric for measuring the duration for which a powder can be reused in LPBF. Although 10 and 30 build cycles were used in the model, feedstocks may be post-processed to remove chemical impurities, improve powder size distribution, and subsequently be recertified for future production use. Additionally, the rate of decline and salvage value designated for the powder feedstock will vary from manufacturer based on the component’s industry of use and regulatory restrictions. A 2.25 dosage amount is very aggressive and is typically larger for builds consisting of higher build volume utilizations which can lead to higher feedstock depreciation costs. Mass production with LPBF faces lengthy build times which may be susceptible to power outages and build restarts which can add further build setup and labor costs. Additionally, mass production can demand large quantities of powder feedstock and require the feed bed to be refilled with extra powder during the build in order to produce all parts and fully complete the build process.

In the plug-in, accuracy of the estimated support structure volume was heavily dependent on the specified support angle and geometry of the supports. The ray-trace projection and grid partitioning algorithm were susceptible to aliasing as the parts was being rotated due to alignment with the build platform. Cylindrical supports were applied at each intersection point and were not organized by overhang regions which resulted in over 100 individual solid bodies in the part history and run times in excess of 1300 seconds for high resolution supports hindering the speed and flexibility of analyzing multiple part configurations.

5.3 Future Work

Powder feedstock reusability can be better studied by identifying which properties impact fatigue and functional performance in as-built, machined, and heat-treated components. Generalized metrics for powder reuse and duration limits can be studied to better compare how parts differ than those produced with virgin powder and with differing LPBF machines. Improvements to the LPBF cost model could be found through a regression-based build time estimator that examines similar LPBF technologies (e.g. power, layer thickness, recoat speed, quantity of lasers, etc.) across multiple build jobs with varying geometries, replicates, and build heights. Mass production cost studies in LPBF could be performed to better highlight the rate of build failure, power outages, and build restarts that occur in a production environment.

For the CAD-integrated cost estimator, the macro's ray-trace projection can be revised with high precision techniques such as octree and voxelization. Speed can improved by querying surface tessellation data and coordinates from the graphics body displayed by the CAD program. The support structure primitives can be expanded from cylinders to a library of different shapes including hollow, solid, tree, and scaffolding supports. Additional functions can be added to the macro including analyzing and positioning multiple bodies along with having the capability to generate slice files directly from the CAD geometry.

References

[1] ASTM F2792 - 12a “Standard Terminology for Additive Manufacturing Technologies.” *ASTM International*, Web. Oct. 14, 2015. <https://www.astm.org/Standards/F2792.htm>

[2] Gibson, I., Rosen, D. and Stucker, B., 2015. *Additive Manufacturing Technologies.* Springer, New York, NY.

[3] Carter, W.T. and Jones, M.G., 1993, August. “Direct laser sintering of metals.” *Proceedings of Solid Freeform Fabrication Symposium,* pp. 51-59.

[4] Das, S., Wohlert, M., Beaman, J.J. and Bourell, D.L., 1998. “Producing metal parts with selective laser sintering/hot isostatic pressing.” *JoM*, 50(12), pp.17-20.

[5] Meiners, W., Over, C., Wissenbach, K. and Poprawe, R., 1999. “Direct generation of metal parts and tools by selective laser powder remelting (SLPR).” *Proceedings of Solid Freeform Fabrication Symposium,* pp. 655-662.

[6] Frazier, W.E., 2014. “Metal additive manufacturing: a review.” *Journal of Materials Engineering and Performance*, 23(6), pp.1917-1928.

[7]] Hussein, A., Hao, L., Yan, C., Everson, R. and Young, P., 2013. “Advanced lattice support structures for metal additive manufacturing.” *Journal of Materials Processing Technology*, 213(7), pp.1019-1026.

[8] Sochalski-Kolbus, L.M., Payzant, E.A., Cornwell, P.A., Watkins, T.R., Babu, S.S., Dehoff, R.R., Lorenz, M., Ovchinnikova, O. and Duty, C., 2015. “Comparison of residual stresses in Inconel 718 simple parts made by electron beam melting and direct laser metal sintering.” *Metallurgical and Materials Transactions* A, 46(3), pp.1419-1432.

[9] Béraud, N., Vignat, F., Villeneuve, F. and Dendievel, R., 2014. “New trajectories in Electron Beam Melting manufacturing to reduce curling effect.” *Procedia CIRP*, 17, pp.738-743.

[10] Moylan, S.P., Slotwinski, J.A., Cooke, A.C., Jurrens, K. and Donmez, M.A., 2013. “Lessons Learned in Establishing the NIST Metal Additive Manufacturing Laboratory.” *NIST*. Web. Aug. 13, 2018. <https://nvlpubs.nist.gov/nistpubs/technicalnotes/nist.tn.1801.pdf>

[11] Kinsella, M. E., 2008, "Additive Manufacturing of Superalloys for Aerospace Applications," Metals Branch, Ceramics and NDE Division, Editor.

[12] Wohlers, T., 2018. “Wohlers Report 2018.” *Wohlers Associates, Inc*. Web. Aug. 13, 2018. <https://wohlersassociates.com/2018report.htm>

[13] "GE Plans to Invest $1.4B to Acquire Additive Manufacturing Companies Arcam and SLM”. *GE Aviation*. Web. Sep. 6, 2016. <https://www.geaviation.com/press-release/other-news-information/ge-plans-invest-14b-acquire-additive-manufacturing-companies>

[14] "Audi and EOS: Development Partnership Focuses on Holistic Approach for Metal-Based Additive Manufacturing," *EOS Gmbh*. Web. Jan. 24, 2017. <https://www.eos.info/press/development-partnershi-between-audi-and-eos>

[15] Mahon, L., 2016. "Alcoa Opens 3D Printing Metal Powder Plant," *3D Printing Industry*. <https://3dprinting industry.com/news/alcoa-opens-3d-printing-metal-powder-plant-84522/>

[16] Seifi, M., Salem, A., Beuth, J., 2016, "Overview of Materials Qualification Needs for Metal Additive Manufacturing," *JoM,* 68(3) pp. 747-764.

[17] Thomas, D.S. and Gilbert, S.W., 2014. “Costs and cost effectiveness of additive manufacturing”. *NIST Special Publication*, 1176, p.12. Web. Aug. 13, 2018. < https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1176.pdf>

[18] Müller, A., and Karevska, S., 2016, "How will 3D printing make your company the strongest link in the value chain?". Ernst & Young, Web. Aug. 13, 2018. < https://www.ey.com/Publication/vwLUAssets/ey-global-3d-printing-report-2016-full-report/$FILE/ey-global-3d-printing-report-2016-full-report.pdf>

[19] “Senvol Database”. Senvol, Web. Jan. 24, 2017. <http://senvol.com/database>.

[20] Medina, F., 2013. “Reducing metal alloy powder costs for use in powder bed fusion additive manufacturing: Improving the economics for production.” Department of Metallurgical, Materials, and Biomedical Engineering. Ph.D. Dissertation, *University of Texas, El Paso*, El Paso, TX.

[21] Dawes, J., Bowerman, R. and Trepleton, R., 2015. “Introduction to the additive manufacturing powder metallurgy supply chain.” *Johnson Matthey Technology Review*, 59(3), pp.243-256.

[22] Slotwinski, J.A., Garboczi, E.J., Stutzman, P.E., Ferraris, C.F., Watson, S.S. and Peltz, M.A., 2014. “Characterization of metal powders used for additive manufacturing.” *Journal of Research of the National Institute of Standards and Technology*, 119, pp. 460-493.

[23] Slotwinski, J.A. and Garboczi, E.J., 2015. “Metrology needs for metal additive manufacturing powders.” *JOM*, 67(3), pp.538-543.

[24] Slotwinski, J.A., Garboczi, E.J. and Hebenstreit, K.M., 2014. “Porosity measurements and analysis for metal additive manufacturing process control.” *Journal of Research of the National Institute of Standards and Technology,* 119, pp. 494-528.

[25] Baumers, M., Tuck, C., Wildman, R., "Energy Inputs to Additive Manufacturing: Does Capacity Utilization Matter?" *Proceedings of Solid Freeform Fabrication Symposium*, pp. 30-40.

[26] Cheng, W., Fuh, J., Nee, A., 1995, "Multi-Objective Optimization of Part-Building Orientation in Stereolithography," *Rapid Prototyping Journal*, 1(4), pp. 12-23.

[27] Byun, H. S., and Lee, K. H., 2005, "Determination of the Optimal Part Orientation in Layered Manufacturing using a Genetic Algorithm," *International Journal of Production Research*, 43(13), pp. 2709-2724.

[28] Redwood, B., Schöffer, F., and Gerret, B., 2017, “The 3D Printing Handbook: Technologies, Design and Applications,” *3D Hubs B.V.* New York, NY.

[29] Hiller, J. D., and Lipson, H., 2009, "STL 2.0: a proposal for a universal multi-material Additive Manufacturing File format," *Proceedings of Solid Freeform Fabrication Symposium*, pp. 266-278.

[30] Zelinski, P., 2015, "Additive's Idiosyncrasies," Additive Manufacturing, Web. Aug. 15, 2018. <http://www.Additivemanufacturinginsight.Com/Articles/Additives-Idiosyncrasies>

[31] "3MF Consortium Signs New Members 3D Systems, Materialise, Siemens PLM Software and Stratasys," Computer Business Week, pp. 147.

[32] "Standard Specification for Additive Manufacturing File Format (AMF) Version 1.2,". Web. Aug. 13, 2018. < https://www.astm.org/Standards/ISOASTM52915.htm>

[33] Alexander, P., Allen, S. and Dutta, D., 1998. Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design*. Vol 30 No. 5, pp- 343-356.

[34] Hopkinson, N., and P. Dickens. 2003. "Analysis of Rapid Manufacturing—using Layer Manufacturing Processes for Production." *Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*. 217 (1). pp. 31–39.

[35] Ruffo, M., Tuck, C. and Hague, R. (2006). Cost estimation for rapid manufacturing - laser sintering production for low to medium volumes. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 220, pp. 1417–1427.

[36] Ruffo, M., and R. Hague, 2007. "Cost estimation for rapid manufacturing simultaneous production of mixed components using laser sintering." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 221, pp.1585-1591.

[37] Atzeni, E., and A. Salmi, 2012. "Economics of additive manufacturing for end-usable metal parts." *The International Journal of Advanced Manufacturing Technology,* 62, pp. 1147-1155.

[38] Baumers, M., 2012. "Economic Aspects of Additive Manufacturing: Benefits, Costs, and Energy Consumption." Doctoral Thesis. *Loughborough University*.

[39] Lindemann, C., Jahnke, U., Moi, M. and Koch, R., 2012. “Analyzing product lifecycle costs for a better understanding of cost drivers in additive manufacturing.” *Proceedings of Solid Freeform Fabrication Symposium*, pp. 177-188.

[40] Rickenbacher, L., Spierings, A. and Wegener, K., 2013. “An integrated cost-model for selective laser melting (SLM).” *Rapid Prototyping Journal*, 19 (3), pp. 208-214.

[41] Fera, M., Macchiaroli, R., Fruggiero, F. and Lambiase, A., 2018. “A new perspective for production process analysis using additive manufacturing—complexity vs production volume.” *The International Journal of Advanced Manufacturing Technology*, 95 (1), pp. 673-685.

[42] Costabile, G., Fera, M., Fruggiero, F., Lambiase, A. and Pham, D., 2017. Cost models of additive manufacturing: A literature review. *International Journal of Industrial Engineering Computations*, 8 (2), pp. 263-283.

[43] Chan, R., Manoharan, S. and Haapala, K.R., 2017. “Comparing the Sustainability Performance of Metal-Based Additive Manufacturing Processes.” *ASME 2017 International Design Engineering Technical Conference.* American Society of Mechanical Engineers, 2017. pp. V004T05A039-V004T05A039

[44] ASTM B243 - 13 Standard Terminology for Powder Metallurgy. *ASTM International*, 7 Jan. 2015. Web. Aug. 13, 2018. <https://www.astm.org/DATABASE.CART/HISTORICAL/B243-13.htm>

[45] Maringer, R. and Patel, A., "Patent WO1989000471A1 - Centrifugal Disintegration." *World Intellectual Property Organization*, 26 Jan. 1989. Web. 4 Jan. 2016.

[46] Howells, R., Stoner, G. and Stockumas, J., "Patent: US4880162 - Gas Atomization Nozzle for Metal Powder Production." *USPTO Patent Database*. 14 Nov. 1989.

[47] Sidney, I., Clark, R., and Baltrukovicz, B., "Patent US4080126 - Water Atomizer for Low Oxygen Metal Powders." *USPTO Patent Database*. 21 Mar. 1978.

[48] Axelsson, S., 2012. “Surface Characterization of Titanium Powders with X-ray Photoelectron Spectroscopy.” Doctoral dissertation, Chalmers University of Technology, Gothenburg, Sweden.

[49] Silberberg, M.S., Duran, R., Haas, C.G. and Norman, A.D., 2006. *Chemistry: The molecular nature of matter and change* (Vol. 4). New York: McGraw-Hill.

[50] Watkins, K.G., Steen, W. and Mazumder, J., 2010. *Laser Material Processing*. Springer, New York, NY.

[51] Jamshidinia, M., Sadek, A., Wang, W. and Kelly, S., 2015. “Additive manufacturing of steel alloys using laser powder-bed fusion.” *Advanced Materials & Processes*, 173 (1), pp. 20-24.

[52] Tang, H.P., Qian, M., Liu, N., Zhang, X.Z., Yang, G.Y. and Wang, J., 2015. “Effect of powder reuse times on additive manufacturing of Ti-6Al-4V by selective electron beam melting.” *JOM*, 67 (3), pp. 555-563.

[53] Grainger, L. “Investigating the effects of multiple re-use of Ti6Al4V powder in additive manufacturing (AM),” *Renishaw plc*, UK. Web. Accessed: 26 June 2016. <http://www.renishaw.com/en/blog-post-how-much-can-you-recycle-metal-additive-manufacturing-powder--38882>

[54] O'Leary, R., Setchi, R. and Prickett, P.W., 2015. “An investigation into the recycling of Ti-6Al-4V powder used within SLM to improve sustainability.” *International Conference on Sustainable Design and Manufacturing*, pp. 377-388.

[55] Seyda, V., Kaufmann, N. and Emmelmann, C., 2012. “Investigation of aging processes of Ti-6Al-4 V powder material in laser melting.” *Physics Procedia*, 39, pp.425-431.

[56] Jacob, G., Brown, C., Donmez, A., Watson, S. and Slotwinski, J., 2017. “Effects of powder recycling on stainless steel powder and built material properties in metal powder bed fusion processes.” *NIST*. Web. Aug. 13, 2018. <https://nvlpubs.nist.gov/nistpubs/ams/NIST.AMS.100-6.pdf>

[57] Aboulkhair, N.T., Everitt, N.M., Ashcroft, I. and Tuck, C., 2014. “Reducing porosity in AlSi10Mg parts processed by selective laser melting.” *Additive Manufacturing*, 1, pp.77-86.

[58] Asgari, H., Baxter, C., Hosseinkhani, K. and Mohammadi, M., 2017. On microstructure and mechanical properties of additively manufactured AlSi10Mg_200C using recycled powder. *Materials Science and Engineering*, 707, pp. 148-158.

[59] Ardila, L.C., Garciandia, F., González-Díaz, J.B., Álvarez, P., Echeverria, A., Petite, M.M., Deffley, R. and Ochoa, J., 2014. “Effect of IN718 recycled powder reuse on properties of parts manufactured by means of selective laser melting.” *Physics Procedia*, 56, pp. 99-107.

[60] Samant, R., Lewis, B., “Metal Powder Recycling and Reconditioning in Additive Manufacturing.” Web Accessed. 1 Dec. 2017. < https://ewi.org/paper-metal-powder-recycling-and-reconditioning-in-additive-manufacturing/>

[61] ASTM F2924 – 14 “Standard Specification for Additive Manufacturing Titanium-6 Aluminum-4 Vanadium with Powder Bed Fusion." *ASTM International*. Web. Dec. 1, 2017. < https://www.astm.org/Standards/F2924.htm>

[62] ASTM F3001 – 14 “Standard Specification for Additive Manufacturing Titanium-6 Aluminum-4 Vanadium ELI with Powder Bed Fusion." *ASTM International*. Web. Dec. 1, 2017. <https://www.astm.org/Standards/F3001.htm>

[63] ASTM F3055-14a “Standard Specification for Additive Manufacturing Nickel Alloy (UNS N07718) with Powder Bed Fusion." *ASTM International*. Web. Dec. 1, 2017. <https://www.astm.org/Standards/F3055.htm>

[64] ASTM F3056 – 14e1 “Standard Specification for Additive Manufacturing Nickel Alloy (UNS N06625) with Powder Bed Fusion." *ASTM International*. Web. Dec. 1, 2017. <https://www.astm.org/Standards/F3056.htm>

[65] ASTM F3184-16 “Standard Specification for Additive Manufacturing Stainless Steel Alloy (UNS S31603) with Powder Bed Fusion” *ASTM International*. Web. Dec. 1, 2017. <https://www.astm.org/Standards/F3184.htm>

[66] MSFC-STD-3716 “Standard for Additively Manufactured Spaceflight Hardware By Laser Powder Bed Fusion In Metals” *National Aeronautics and Space Administration.* Web. Dec. 1, 2017. <https://www.nasa.gov/sites/default/files/atoms/files/msfcstd3716baseline.pdf>

[67] MSFC-STD-3717 “Specification for Control and Qualification of Laser Powder Bed Fusion Metallurgical Processes” *National Aeronautics and Space Administration.* Web. Dec. 1, 2017. <https://www.nasa.gov/sites/default/files/atoms/files/msfcspec3717baseline.pdf>

[68] Rosen, D. W., Schaefer, D., and Schrage, D., 2012, "GT MENTOR: A high school education program in systems engineering and additive manufacturing," *Proceedings of Solid Freeform Fabrication Symposium,* pp. 62-80.

[69] Yim, S., and Rosen, D., 2012, "Build time and cost models for additive manufacturing process selection," *ASME 2012 International Design Engineering Technical Conferences and Computers and Information In Engineering Conference*, DETC2012-70940, pp. 375-382.

[70] Guignard, V. "SolidWorks Print3D Tool," Web. Apr. 27, 2017.<http://www.javelin-tech.com/blog/2017/04 /solidworks- print-3d-tool/>

[71] "PTC and Stratasys Collaborate to Define and Deliver Design for Additive Manufacturing," Web. Jun. 8, 2015. PTC. <https://www.ptc.com/en/news/2015/ptc-stratasys-design-for-additive-manufacturing>

[72] "Xometry - the Manufacturing on-Demand Platform - Launches Dassault Systemes SOLIDWORKS(R) Integration," Web. Jul. 14, 2016 < http://www.prnewswire .com/news-releases/xometry---the-manufacturing-on-demand-platform---launches-dassault-systemes-solidworks-integration-300298668.html>

[73] Allen, S., and Dutta, D., 1994, "On the computation of part orientation using support structures in layered manufacturing," *Proceedings of Solid Freeform Fabrication Symposium*, pp. 259-269.

[74] Hur, J., and Lee, K., 1998, "The Development of a CAD Environment to Determine the Preferred Build-Up Direction for Layered Manufacturing," *The International Journal of Advanced Manufacturing Technology,* 14 (4) pp. 247-254.

[75] Li, Y., and Zhang, J., 2013, "Multi-Criteria GA-Based Pareto Optimization of Building Direction for Rapid Prototyping," *The International Journal of Advanced Manufacturing Technology*, 69(5-8) pp. 1819-1831.

[76] Phatak, A. M., and Pande, S. S., 2012, "Optimum Part Orientation in Rapid Prototyping using Genetic Algorithm," *Journal of Manufacturing Systems*, 31(4) pp. 395-402.

[77] Canellidis, V., Giannatsis, J., and Dedoussis, V., 2009, "Genetic-Algorithm-Based Multi-Objective Optimization of the Build Orientation in Stereolithography," *The International Journal of Advanced Manufacturing Technology*, 45 (7-8), pp. 714-730.

[78] Ghorpade, A., Karunakaran, K. P., and Tiwari, M. K., 2007, "Selection of Optimal Part Orientation in Fused Deposition Modelling using Swarm Intelligence," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 221 (7), pp. 1209-1219.

[79] Thrimurthulu, K., Pandey, P. M., and Reddy, N. V., 2004, "Optimum Part Deposition Orientation in Fused Deposition Modeling," *International Journal of Machine Tools and Manufacture*, 44 (6), pp. 585-594.

[80] Liu, Y., Yang, Y., and Wang, D., 2016, "A Study on the Residual Stress during Selective Laser Melting (SLM) of Metallic Powder," *The International Journal of Advanced Manufacturing Technology*, 87 (1-4), pp. 647-656.

[81] Kruth, J., Badrossamay, M., Yasa, E., 2010, "Part and material properties in selective laser melting of metals," *Proceedings of The 16th International Symposium on Electromachining*.

[82] Morgan, H. D., Cherry, J. A., Jonnalagadda, S., 2016, "Part Orientation Optimisation for the Additive Layer Manufacture of Metal Components," *The International Journal of Advanced Manufacturing Technology*, 86 (5-8), pp. 1679-1687.

[83] Verma, A., Tyagi, S., and Yang, K., 2015, "Modeling and Optimization of Direct Metal Laser Sintering Process," *The International Journal of Advanced Manufacturing Technology*, 77 (5-8), pp. 847-860.

[84] Simonelli, M., Tse, Y. Y., and Tuck, C., 2014, "Effect of the Build Orientation on the Mechanical Properties and Fracture Modes of SLM Ti–6Al–4V," *Materials Science and Engineering*, 616, pp. 1-11.

[85] Wauthle, R., Vrancken, B., Beynaerts, B., 2015, "Effects of Build Orientation and Heat Treatment on the Microstructure and Mechanical Properties of Selective Laser Melted Ti6Al4V Lattice Structures," *Additive Manufacturing*, 5, pp. 77-84.

[86] Ulu, E., Korkmaz, E., Yay, K., 2015, "Enhancing the Structural Performance of Additively Manufactured Objects through Build Orientation Optimization," *Journal of Mechanical Design*, 137 (11), pp. 111410.

[87] Vaidya, R., and Anand, S., 2016, "Image Processing Assisted Tools for Pre-and Post-Processing Operations in Additive Manufacturing," *Procedia Manufacturing*, 5, pp. 958-973.

[88] Gale Encyclopedia of American Law. 2010. Depreciation. Donna Batten, Ed. 3rd ed. Vol. 3. Detroit, MI: Gale. 427-428. *Gale Virtual Reference Library*. Web. Oct. 17, 2015. <https://www.cengage.com/search/productOverview.do?Ntt=encyclopedia+of+american+law%7C%7C14596495191662317855214033770727553087 4&N=197&Ntk=APG%7C%7CP_EPI&Ntx=mode%2Bmatchallpartial>

[89] Internal Revenue System, “Overview of Depreciation," U.S. Department of Treasury. Web. Oct. 17, 2015. <https://www.irs.gov/publications/p946/ch01.html#en_US_2013_publink1000107337>

[90] Newnan, D.G., Eschenbach, T. and Lavelle, J.P., 2004. “Engineering economic analysis.” *Oxford University Press*, Oxford, UK.

[91] Samperi, M., “Development of Design Guidelines for Metal Additive Manufacturing and Process Selection.” M.S. Thesis, Mechanical Engineering, *The Pennsylvania State University*. June 2014.

[92] “EOS GmbH, Basic Design Rules for Additive Manufacturing.” *Electro Optical Systems*, Germany. Web. Oct. 15, 2015. <https://cdn0.scrvt.com/eos/public/ab4f0542d66453fc/5f889ab7e3f72bd3d44b22205ba8b68b/EOS-Basic-Design-Rules_Additive-Manufacturing_EN.pdf>

[93] Kannan, T., “Design for Additive Manufacturing.” *Geometric Global*. Web. Oct. 15, 2015. <https://dfmpro.geometricglobal.com/files/2017/05/Whitepaper-Design-for-Additive-Manufacturing.pdf>

[94] “netfabb Support Structures.” netfabb Gmbh. Web. Jan. 3, 2016. <http://www.netfabb.com/support_structures.php>.

[95] “Magics SG+ Module” Materialise NV. Web. Jan. 3, 2016. <http://software.materialise.com/magics-sg-plus-module>.

[96] Wright, S. 2015, "A Story of Failure and Success in Metal AM: The Reality of Developing a Titanium Bike Part." *Metal Additive Manufacturing Magazine*. 1 (3). pp. 41-50.

[97] Vayre, B., Vignat, F. and Villeneuve, F., 2013. “Identification on some design key parameters for additive manufacturing: application on electron beam melting.” *Procedia CIRP*, 7, pp. 264-269.

[98] Office of Environmental Health and Safety, Waste Management Log Book, *The Pennsylvania State University*. Web. Aug. 30, 2016. < http://abe.psu.edu/safety/lab-safety-plan/chemical-waste/chemical-waste-log-book>.

[99] Oller, A.R., Kirkpatrick, D.T., Radovsky, A. and Bates, H.K., 2008. “Inhalation carcinogenicity study with nickel metal powder in Wistar rats.” *Toxicology and applied pharmacology*, 233(2), pp.262-275.

[100] Jacobson, M., Cooper, A.R. and Nagy, J., 1964. *Explosibility of metal powders*. U.S. Bureau of Mines, Washington, D.C.

[101] Wright, Spencer. "Notes on Magics." 18 Aug. 2015. Web. Accessed: 15 Jan. 2016. <http://pencerw.com/feed/2015/8/18/notes-on-magics>.

[102] Baumers, M., C. Tuck, R. Wildman, I. Ashcroft, E. Rosamond, and R. Hague., 2012. "Combined Build-Time, Energy Consumption and Cost Estimation for Direct Metal Laser Sintering." *Proceedings of Solid Freeform Fabrication Symposiu,* pp. 932-944.

[103] Mindt, H.W., Desmaison, O., Megahed, M., Peralta, A. and Neumann, J., 2018. “Modeling of powder bed manufacturing defects.” *Journal of Materials Engineering and Performance*, 27(1), pp. 32-43.

[104] Foster, B.K., Reutzel, E.W., Nassar, A.R., Hall, B.T., Brown, S.W. and Dickman, C.J., 2015. “Optical, layerwise monitoring of powder bed fusion.” *Proceedings of Solid Freeform Fabrication Symposium*. pp. 295-307.

[105] Thöne, M., Leuders, S., Riemer, A., Tröster, T. and Richard, H.A., 2012. “Influence of heat-treatment on selective laser melting products–eg Ti6Al4V.” *Proceedings of Solid Freeform Fabrication Symposium*, pp. 492-498.

[106] “EOS Stainless Steel GP1”, Material Data Sheet, Germany. Web. Oct. 15, 2015. <https://cdn0.scrvt.com/eos/public/5f84f5d2c88ac900/05fb1582834a38c85ef6dd859733a230/EOS_StainlessSteel-GP1_en.pdf>

[107] “EOS Titanium Ti64”, Material Data Sheet, EOS GmbH – Electro Optical Systems, Germany. Web. Oct. 15, 2015. <https://cdn.eos.info/d27ce4e4388315f2/7bed25543c76/Ti64ELI_M290_Material_data_sheet_06-16_en.pdf>

[108] “EOS Nickel Alloy IN718”, Material Data Sheet, *EOS GmbH – Electro Optical Systems*, Germany. Web. Oct. 15, 2015. < http://ip-saas-eos-cms.s3.amazonaws.com/public/4528b4a1bf688496/ff974161c2057e6df56db5b67f0f5595/EOS_NickelAlloy_IN718_en.pdf>

[109] “EOS Aluminum AlSi10Mg”, Material Data Sheet, *EOS GmbH – Electro Optical Systems*, Germany. Web. Oct. 15, 2015. <https://cdn0.scrvt.com/eos/public/8837de942d78d3b3/4e099c3a857fdddca4be9d59fbb1cd74/EOS_Aluminium_AlSi10Mg_en.pdf>

[110] Poyraz, Yasa, E., Akbulut, G., 2015. "Investigation of Support Structures for Direct Metal Laser Sintering (DMLS) Of In625 Parts,", *Proceedings of Solid Freeform Fabrication Symposium*, pp. 560-574.

[111] Marschallinger, R., Jandrisevits, C., and Zobl, F., 2015, "A Visual LISP Program for Voxelizing AutoCAD Solid Models," *Computers & Geosciences*, 74, pp. 110-120.

[112] Kerbrat, O., Mognol, P., and Hascot, J., 2011, "A New DFM Approach to Combine Machining and Additive Manufacturing," *Computers in Industry*, 62 (7), pp. 684-692.

[113] "2015 SolidWorks API Help,". Web. Jan. 22, 2017. *Dassault Systemes*. <http://help.solidworks.com/2016/English/api/sldworksapiprogguide/Welcome.htm>

[114] Hassan, R., Cohanim, B., De Weck, O. and Venter, G., 2005, April. “A comparison of particle swarm optimization and the genetic algorithm.” *In 46th AIAA / ASME / ASCE / AHS / ASC Structures, Structural Dynamics and Materials Conference*, pp. 1897 - 1909.

[115] Maranan, V., Simpson, T. W., Palmer, T., 2016, "Application of Topology Optimization and Design for Additive Manufacturing Guidelines on an Automotive Component," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V02AT03A030.

[116] Brackett, D., Ashcroft, I. and Hague, R., 2011. “Topology optimization for additive manufacturing.” *Proceedings of Solid Freeform Fabrication Symposium*, pp. 348-362.

[117] Huang, P., Wang, C.C. and Chen, Y., 2014. “Algorithms for Layered Manufacturing in Image Space.” *Advances in Computers and Information in Engineering Research, Volume 1*. ASME Press, New York, NY.

[118] “Sculpteo” Web. Aug. 15, 2018. < https://www.sculpteo.com/en/>

Appendix A

Cost Model for Powder Feedstock Depreciation MATLAB Code I

```
%% Cost Modeling for Reused Powder Feedstocks in Laser Powder Bed Fusion
% Michael W. Barclift - Penn State University - University Park, PA 16801
% mzb5747@psu.edu | mwb81@vt.edu |
% SAE Upright Individual Cost Modeling Code
%%

clc
clear all
close all
format long g

%% Initilization
xmax = 250.8; %Buildplate width on EOS M280 (mm)
ymax = 250.8; %Buildplate length on EOS M280 (mm)

dxmax = 250.8; %Dispenser width on EOS M280 (mm)
dymax = 227.80625; %Dispenser length on EOS M280 (mm)

Mat.wden = [7.8,8.15,2.67,4.41]; %Density of solidified powder feedstock (g/cm^3)
Mat.tapden = [5.3, 5.1, 1.5,2.74]; %Density of un-melted powder feedstock (g/cm^3)
Mat.buildrate = [7.2,14.4,26.6,13.5]; %Average time needed by AM machine to solidfy a voxel (hr/cm^3)
Mat.layer = [20,40,30,30]; %Layer Thickness (microns)
Mat.vprice = [105, 192,152,680]; %Price of virgin powder feedstock ($/kg)
Mat.salvage = 0; %Estimated value of powder feedstock at end of useful life (hr)
Mat.life = 0; %Estimated useful life of powder feedstock in build cycles (-)
Mat.use1 = 0; %Selected cycle in life of powder feedstock
Mat.waste = 0.4; %Percentage of powder lost during AM build process (%)
Mat.trapped = 0.25; %Percentage of powder trapped in Support Structures
Mat.charge = 2.25; %Amount of excess powder added per layer

Cost.oper = 110; %AM machine operator's cost ($/hr)
Cost.pc = 100; %Cost of the computer workstation ($/hr)
Cost.mach = 60; %Cost of the AM machine during build operation ($/hr)
Cost.stress = 350; %Cost to stress relief components on build substrate ($)
Cost.EDM = 200; %Cost to wire-EDM components on AM build substrate ($)
Cost.gas = 10; %Cost to use inert gas during AM build process ($/hr)
Cost.tools = 50; %Cost to use post-processing tools/equipment ($/hr)

Time.prep = [3]; %Time to generate support structures for digital models (hr)
Time.buildjob = 1; %Time to compile and arrange geometries on the build tray
Time.setup = 2; %Time to setup AM machine, gas, software, and pre-processes (hr)
Time.change = 0; %Time to change loaded powder, clean machine, filters, and reload (hr)
Time.recoat = 9; %Time AM machine needs to spread a new layer of powder (sec)
Time.buildrate = [7.2,14.4,26.6,13.5]; %Average time needed by AM machine to solidfy a voxel (hr/cm^3)
Time.removal = 3; %Time to remove build substrate from AM machine, sieve powder, clean, and documentation (hr)
Time.postp = 3; %Time required to post-process individual parts (hr)

%% User Inputs
Part.zheight = [172]; %Build height of parts (mm)
Part.vol = [175]; %Part geometric volume (cm^3)
Part.supports = [412]; %Part goemetric volume for supports (cm^3)
Part.totalvol = Part.vol + Part.supports; %Total volume (cm^3)
%Part.dispvol = Mat.charge*[2484.375,2484.375,2506.88]; %Volume to fill build chamber (cm^3)
%Part.dispvol = Mat.charge*[Part.zheight.*dxmax*dymax]/10^3; %Volume to fill build chamber (cm^3)
Part.dispvol = Mat.charge*[Part.zheight.*dxmax*dymax]/10^3; %Volume to fill build chamber (cm^3)
```

```
%------------------EDIT HERE TO CHANGE NUMBER OF PARTS--------------------
%  for mchoice = 1:4 %Material choice
  pcc_count = 0;
%     for pcc = 1:3 %Part count cases selector
    Part.count.cases = [1,2,4,6,8]; %Scenarios of part count cases

%        Part.N = Part.count.cases(pcc)*[3,3,3]; %Number of replicates for a geometry (-)
    Part.N = [1];

    Part.postp = [1]; %Parts needing to be post-processing (True (1) False (0))
    i = 1; %Selected Part Volume
    p = 1; %Selected Part Geometries to Include in Cost Analysis
    j = 1; %Selected AM Process

    %------------------EDIT HERE TO CHANGE MATERIAL--------------------
      mchoice = 4;
    k = mchoice; %Selected Material [GPI - 1, IN718 - 2, AlSi10Mg -3, Ti64 -4]

    %% Build Time Estimate for Powder Be Fusion
    %1. Complie build tray - part data
    PartData = [Part.zheight;Part.totalvol;Part.dispvol;Part.N];

    %2. Sort all part data by increasing z-height
    PartData = sortrows(PartData',1)';

    %3. Convert z-height to layers
    PartData(1,:) = round(PartData(1,:)*1000/Mat.layer(k));

    %4. Calculate layerwise recoating time allocation
    layer = PartData(1,1:p);
    Time.rc = zeros(1,p);

    for u = 0:(p-1)
      if u == 0
        Time.rc(1,u+1) = 1/60*1/60*Time.recoat*(layer(u+1)-0)./sum(PartData(4,(u+1):p));
      elseif u > 0
        Time.rc(1,u+1) = Time.rc(1,u) + 1/60*1/60*Time.recoat*(layer(u+1)-layer(u))./sum(PartData(4,(u+1):p));
      end
    end

    Time.trc = Time.recoat*layer(u+1)*1/60*1/60; %Total recoating time for the buildjob (hr)

    %-------Editing Build Time Estimate to be for whole build--------
    %Time.exp = Part.N*PartData(2,1:p).*1/Mat.buildrate(k); %Total time for solidfying each part volume (hr)
    Time.exp = PartData(2,1:p).*1/Mat.buildrate(k);

    Time.delay = 0/sum(PartData(4,1:p)); %Total time for heat, cooling, inactive (hr)

     Time.buildt = (Time.rc + Time.exp);%CHANGE(Time.rc.*Pmax)+(Time.exp.*Pmax);
    %% Part Mass Calculation
    Part.vol = PartData(2,1:p);
    Part.bed = PartData(3,1:p);

    %% Iterative Loop Structure
    Mat.lifecases = [1,1,1,1];

    for index = 1:length(Mat.lifecases)
      Mat.lifecases = [10,30,10,30]-0*[1,1,1,1]; %Scenarios of Estimated Life for Powder Feedstock
```

```
        count = 1;
        Mat.life = Mat.lifecases(index);

        %% Depreciation Cost Calculation
        % SLN
        %    Dep = @(cmo,S,U,u) (cmo)-(u).*cmo/(U);
        %    Mat.value = Mat.vprice(k);
        %    for u = 2:Mat.life
        %        Mat.value(u) = Dep(Mat.value(1),Mat.salvage,Mat.life,(u-1));
        %    end
        %    Mat.value(u+1) = 0;

        % SOYD DEPRECIATION METHOD
        Dep = @(cmu,cmo,S,U,u) cmu-(cmo-S)*(U-u+1)./(U*(U+1)/2);
        Mat.value = Mat.vprice(k);

        if index>2
           Mat.value=Mat.value*0.4;
        end

        for u = 2:Mat.life
           Mat.value(u) = Dep(Mat.value(u-1),Mat.value(1),Mat.salvage,Mat.life,(u-1));
        end
        Mat.value(u+1) = 0;

        % Double Declining Balance Method
        Dep = @(cmu,U,u) cmu - 2/U.*(cmu);
        Mat.value3 = Mat.vprice(k);
        for u = 2:Mat.life
           Mat.value3(u) = Dep(Mat.value3(u-1),Mat.life,(u-1));
        end
        Mat.value3(u+1)=0;


        for index2 = 1:Mat.life
           Mat.use1 = index2;

           %% Calculation of Costing
           Part.bedmass = Part.bed(1:p).*Mat.tapden(k)/1000;
           Part.bedmass = unique(Part.bedmass)';
           %Cost.dep = (Part.bedmass(p)-sum(Part.mass.*Part.N(1:p))).*(Mat.value(1)-Mat.value(2));

           %% Calculate Part Volumes Partitions for Layerwise Depreciation

           Part.volp = length(Part.bedmass); %Minimum of number volume partitions
           Part.volpi = zeros(Part.volp,p); %Intialize Part Volume Partitions Input

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
           % USER MUST INPUT DATA FOR VOLUME AT PARTITIONS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

           % DATA FOR PENCIL THRUSTERS
           %  Part.volpi = [15.615,15.698,9.40;0,0,0.05]; %User Input Partition Data
           %  %OLD DATA WITH SUPPORTS STILL IN VOLUME
%             Part.volpi = [14.91, 15.39,9.40;0,0,0.05];
%             Part.supports = [0.705,0.308,0;0,0,0];
```

```
         Part.volpi = Part.totalvol;
         Part.volpi = Part.vol-Part.supports;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

        Part.N = PartData(4,1:p);
        Part.mass = 1/1000*((1+Mat.waste)*(Part.volpi + Part.supports))*Mat.wden(k) +
1/1000*(Mat.trapped*Part.supports)*Mat.tapden(k);
        Part.N = logical(Part.volpi).*repmat(Part.N,Part.volp,1);

        %------THIS PART CALCULATES DEPRECIATION FOR POWDER IN THE BED AND PARTS------
        %------Calculated via Conversation of Mass and Equal Depreciation
        %------Method is volume agnostic with no penalty for large/small parts as of 5/25/2016
        % %      ====OLD METHOD===== 5/26/2016
        %      for s = 0:(Part.volp-1)
        %      if s == 0
        %          Part.depmass(s+1,:) = (Part.bedmass(s+1)-0) - sum(Part.mass(s+1,:).*Part.N(s+1,:)); %Mass of
Depreciated Powder in the Bed
        %          Part.depma(s+1,:) = Part.depmass(s+1,:)./sum(Part.N(s+1,:)); %Mass of Depreciated Powder
Allocated to each parts
        %      elseif s >0
        %          Part.depmass(s+1,:) = (Part.bedmass(s+1)-Part.bedmass(s)) - sum(Part.mass(s+1,:).*Part.N(s+1,:));
%Mass of Depreciated Powder in the Bed
        %          Part.depma(s+1,:) = Part.depmass(s+1,:)./sum(Part.N(s+1,:)); %Mass of Depreciated Powder
Allocated to each parts
        %      end
        %      end
        %
        %      if Mat.use1 == 0
        %      Part.dep = Part.depmass.*(Mat.value(0+1)-Mat.value(1+1));
        %      Part.dep = logical(Part.volpi).*repmat(Part.dep,1,p);
        %      Part.depma = Part.depma.*(Mat.value(0+1)-Mat.value(1+1));
        %      Part.depma = logical(Part.volpi).*repmat(Part.depma,1,p);
        %
        %      else
        %      Part.dep = Part.depmass.*(Mat.value(Mat.use1)-Mat.value(Mat.use1+1));
        %      Part.dep = logical(Part.volpi).*repmat(Part.dep,1,p);
        %      Part.depma = Part.depma.*(Mat.value(Mat.use1)-Mat.value(Mat.use1+1));
        %      Part.depma = logical(Part.volpi).*repmat(Part.depma,1,p);
        %
        %      end
        %
        %      Part.depma = sum(Part.depma); %Depreciation Cost Allocated to Each Individual Part
        %      %=====END OF OLD METHOD===== 5/26/2016

        %% NEW METHOD
        %=====NEW METHOD BASED ON MASS (volumes assume air voids which add error to model)====
5/26/2016
        for s = 0:(Part.volp-1)
          if s == 0
              Part.depmass(s+1,:) = (Part.bedmass(s+1)-0) - sum(Part.mass(s+1,:).*Part.N(s+1,:)); %Mass of
Depreciated Powder in the Bed
          elseif s >0
              Part.depmass(s+1,:) = (Part.bedmass(s+1)-Part.bedmass(s)) - sum(Part.mass(s+1,:).*Part.N(s+1,:));
%Mass of Depreciated Powder in the Bed
          end
        end
        Part.depma = (Part.N.*Part.mass)./sum(Part.N.*Part.mass); %This is the percentage of depreciated mass
allocated to each part based on mass
%         Part.depma = repmat(Part.depmass,1,p).*Part.depma; %This is the actual depreciated mass allocated

        if Mat.use1 == 0
           Part.dep = Part.depmass.*(Mat.value(0+1)-Mat.value(1+1));
           % Part.dep = logical(Part.volpi).*repmat(Part.dep,1,p);
%             Part.depma = Part.depma.*(Mat.value(0+1)-Mat.value(1+1));
           % Part.depma = logical(Part.volpi).*repmat(Part.depma,1,p);

        else

           if length(Mat.value)==1
              Mat.value = [Mat.value,0];
           end

           Part.dep = Part.depmass.*(Mat.value(Mat.use1)-Mat.value(Mat.use1+1));
           % Part.dep = logical(Part.volpi).*repmat(Part.dep,1,p);
%             Part.depma = Part.depma.*(Mat.value(Mat.use1)-Mat.value(Mat.use1+1));
           % Part.depma = logical(Part.volpi).*repmat(Part.depma,1,p);
        end

        Part.dep = (Part.dep.*Part.depma)./Part.N;

%         Part.depma = sum(Part.depma); %Depreciation Cost Allocated to Each Individual Part

        %% Activity-Cost Calculations
        Pmax = Part.N(1,:);
        Pmax_v = sum(Pmax)*ones(1,length(Pmax));
        Cost.prep = (Cost.oper + Cost.pc)*Time.prep./Pmax;
        Cost.buildjob = (Cost.oper + Cost.pc)*Time.buildjob./Pmax_v;
        Cost.setup = (Cost.oper + Cost.mach)*(Time.setup + Time.change)./Pmax_v;



        Cost.build.mach = (Cost.mach+Cost.gas)*Time.buildt;

        if Mat.use1 == 0
           Cost.build.mat = Part.mass*Mat.value(Mat.use1);
%             Cost.build.mat = sum(Part.mass)*Mat.value(Mat.use1+1);
        else
           Cost.build.mat = Part.mass*Mat.value(Mat.use1);
%             Cost.build.mat = sum(Part.mass)*Mat.value(Mat.use1);
        end

        %CHANGE sum(Part.mass)*Mat.value(1).*Pmax;
        Cost.build.dep = Part.dep; %CHANGE Part.depma.*Pmax;
        Cost.build.total = Cost.build.mach+Cost.build.mat+Cost.build.dep;
        Cost.removal = (Cost.oper + Cost.mach)*(Time.removal)./Pmax_v;
        Cost.substrate = Cost.stress./Pmax_v + Cost.EDM./Pmax_v;

        % FIX POST PROCESSING COSTS FOR EACH MODEL
        Cost.postp = Time.postp*(Cost.oper+Cost.tools).*Part.postp;

        Cost.total = [Cost.prep; Cost.buildjob; Cost.setup; Cost.build.total; Cost.removal; Cost.substrate; Cost.postp]
        Cost.total2 = [Cost.prep; Cost.buildjob; Cost.setup; Cost.build.mach; Cost.build.mat; Cost.build.dep;
Cost.removal; Cost.substrate; Cost.postp]

        %Individual Part Costing stored for each use case [5,10,15,20]
        if index ==1
```

```
          Ind_5(:,:,Mat.use1) = Cost.total2;
        elseif index == 2
          Ind_10(:,:,Mat.use1) = Cost.total2;
        elseif index == 3
          Ind_15(:,:,Mat.use1) = Cost.total2;
        elseif index == 4
          Ind_20(:,:,Mat.use1) = Cost.total2;
        end

        %Total Build Costing stored for each use case [5,10,15,20]
        if index ==1
          tb_5(:,:,Mat.use1) = sum(round(Cost.total2,2)*Pmax',2);
        elseif index == 2
          tb_10(:,:,Mat.use1) = sum(round(Cost.total2,2)*Pmax',2);
        elseif index == 3
          tb_15(:,:,Mat.use1) = sum(round(Cost.total2,2)*Pmax',2);
        elseif index == 4
          tb_20(:,:,Mat.use1) = sum(round(Cost.total2,2)*Pmax',2);
        end

        Cost.total3 = sum(Cost.total2')';

        if index ==1
        Cost.totalzero = sum(sum([Cost.prep; Cost.buildjob; Cost.setup; Cost.build.mach; Cost.removal;
Cost.substrate; Cost.postp])*Pmax');
        end

        datatotal(index,Mat.use1) = sum(sum(round(Cost.total2,2)*Pmax'));

        %Percentage based Material and Depreciation Costing stored for each use case [5,10,15,20]
        if index ==1
          % 6/17/16 - This method is better and less manual ---> mvd5_test(Mat.use1,:) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
          mvd_5(Mat.use1,:) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];

          %    mvd_5(:,:,Mat.use1) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
        elseif index == 2
          mvd_10(Mat.use1,:) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];

          %   mvd_10(:,:,Mat.use1) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
        elseif index == 3
          mvd_15(Mat.use1,:) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
          %  mvd_15(:,:,Mat.use1) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
        elseif index == 4
          mvd_20(Mat.use1,:) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
```

```
          %  mvd_20(:,:,Mat.use1) =
100*[sum(sum(round(Cost.build.mat,2)*Pmax'))./datatotal(index,Mat.use1),sum(sum(round(Cost.build.dep,2)*Pmax'))
./datatotal(index,Mat.use1)];
        end

        mass_all =
[sum(Part.mass).*Pmax,sum(sum(Part.mass).*Pmax),max(Part.bedmass),sum(sum(Part.mass).*Pmax)./max(Part.bedm
ass)];
        vol_all =
[Part.vol.*Pmax,sum(Part.vol.*Pmax),max(Part.bed/Mat.charge),sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge)];

        Cost.build.mat = Part.mass*Mat.value(1);

         if index ==1
        Cost.inf = sum(sum([Cost.prep; Cost.buildjob; Cost.setup; Cost.build.mach; Cost.build.mat; Cost.removal;
Cost.substrate; Cost.postp])*Pmax');
         end

        if index ==3
        Cost.inf2 = sum(sum([Cost.prep; Cost.buildjob; Cost.setup; Cost.build.mach; Cost.build.mat; Cost.removal;
Cost.substrate; Cost.postp])*Pmax');
         end

        if k == 1
%           mvd_all_1(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
      bmu_1(pcc_count+1,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass of
parts over mass of powder used.
      bvu_1(pcc_count+1,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization - volume
of part over volume of bed.
    elseif k == 2
%           mvd_all_2(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
       bmu_2(pcc_count+1,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass of
parts over mass of powder used.
      bvu_2(pcc_count+1,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization - volume
of part over volume of bed.
    elseif k == 3
%           mvd_all_3(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
       bmu_3(pcc_count+1,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass of
parts over mass of powder used.
      bvu_3(pcc_count+1,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization - volume
of part over volume of bed.
    elseif k == 4
%           mvd_all_4(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
       bmu_4(pcc_count+1,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass of
parts over mass of powder used.
      bvu_4(pcc_count+1,:) = sum(Part.vol.*Pmax)/(xmax*ymax*max(Part.zheight)/(10^3)); %Build Volume
Utilization - volume of part over volume of bed.
    end

        %% Clearing of Data
        Part.depmass = []; %Clearing variables for re-calculation
        Part.depma = []; %Clearing variables for re-calculation
        Part.dep = []; %Clearing variables for re-calculation
        s = []; %Clearing variables for re-calculation
        Part.N = []; %Clearing variables for re-calculation
        Part.mass = []; %Clearing variables for re-calculation
      end
      count = count + 1;
    end
    uses = [10,30,10,30];
     for d = 1:4
    plot(0:uses(d),[datatotal(d,1:uses(d)),Cost.totalzero])
    hold on
     end


      plot([0,Mat.life],[Cost.inf,Cost.inf])
     hold on
      plot([0,Mat.life],[Cost.inf2,Cost.inf2])
    plot([0,Mat.life],[Cost.totalzero, Cost.totalzero])


    if mchoice == 1
    title('Material: GP1')
    elseif mchoice == 2
    title('Material: IN718')
    elseif mchoice == 3
    title('Material: AlSi10Mg')
    elseif mchoice == 4
    title('Material: Ti64')
    end

  legend('U_m_a_x = 10 Build Cycles ($680/kg)','U_m_a_x = 30 Build Cycles ($680/kg)','U_m_a_x = 10 Build Cycles ($272/kg)','U_m_a_x = 30 Build Cycles ($272/kg)','Rickenbacher et al. ($680/kg)','Rickenbacher et al. ($272/kg)','Powder exceeds U_m_a_x')
    xlabel('Build Cycles (-)')
    ylabel('Total Costs ($)')





    % Cost Break down code-----------
    figure
    colormap jet

    tb_5(:,:,1) = tb_5(:,:,1)./sum(tb_5(:,:,1))*100
    tb_10(:,:,1) = tb_10(:,:,1)./sum(tb_10(:,:,1))*100
    tb_15(:,:,1) = tb_15(:,:,1)./sum(tb_15(:,:,1))*100
    tb_20(:,:,1) = tb_20(:,:,1)./sum(tb_20(:,:,1))*100

    bar([tb_5(:,:,1)';tb_10(:,:,1)';tb_15(:,:,1)';tb_20(:,:,1)'],'stacked')
    legend('C-prep','C-buildjob','C-setup','C-build-machine','C-build-material','C-build-depreciation','C-removal','C-substrate','C-postp')
%     legend('C_p_r_e_p','C_b_u_i_l_d_j_o_b','C_s_e_t_u_p','C_b_u_i_l_d_-_m_a_c_h_i_n_e','C_b_u_i_l_d_-_m_a_t_e_r_i_a_l','C_b_u_i_l_d_-_d_e_p_r_e_c_i_a_t_i_o_n','C_r_e_m_o_v_a_l','C_s_u_b_s_t_r_a_t_e','C_p_o_s_t_p')
    ylabel('Percentage of Costs (%)')
    ylim([0 100])

    ax = gca;
    ax.XTickLabel = {'U_m_a_x = 10','U_m_a_x = 30','U_m_a_x = 10','U_m_a_x = 30'};
    xlabel('$680/kg                    $272/kg')

     if mchoice == 1
    title('Material: GP1')
    elseif mchoice == 2
    title('Material: IN718')
    elseif mchoice == 3
    title('Material: AlSi10Mg')
    elseif mchoice == 4
    title('Material: Ti64')
    end
%       ---------------



%
%       if k == 1
%           mvd_all_1(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
% %         bmu_1(pcc_count,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass
of parts over mass of powder used.
% %         bvu_1(pcc_count,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization -
volume of part over volume of bed.
%       elseif k == 2
%           mvd_all_2(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
% %           bmu_2(pcc_count,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass
of parts over mass of powder used.
% %         bvu_2(pcc_count,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization -
volume of part over volume of bed.
%       elseif k == 3
%           mvd_all_3(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
% %           bmu_3(pcc_count,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass
of parts over mass of powder used.
% %         bvu_3(pcc_count,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization -
volume of part over volume of bed.
%       elseif k == 4
%           mvd_all_4(pcc_count,:) = [mvd_5(1,:),mvd_10(1,:),mvd_15(1,:),mvd_20(1,:)];
% %           bmu_4(pcc_count,:) = sum(sum(Part.mass).*Pmax)./max(Part.bedmass); %Build Mass Utilization - mass
of parts over mass of powder used.
% %         bvu_4(pcc_count,:) = sum(Part.vol.*Pmax)/max(Part.bed/Mat.charge); %Build Volume Utilization -
volume of part over volume of bed.
%       end
%
%
% figure
% hold on
% j = 1;
%
%
% C = [1 .5 0];
%
%
% for i = 1:2:length(mvd_all_4)
% %   if mod(j/2,1)>0
%   if i == 1;
%   plot(100*[bvu_4],mvd_all_4(:,i),':d','color',[1 0 0],'linewidth',1.5)
%    plot(100*[bvu_4],mvd_all_4(:,i+1),'--x','color',[1 0 0],'linewidth',1.25)
%   elseif i == 3;
%
%         plot(100*[bvu_4],mvd_all_4(:,i),':d','color',[0.9 0.75 0],'linewidth',1.5)
%    plot(100*[bvu_4],mvd_all_4(:,i+1),'--x','color',[0.9 0.75 0],'linewidth',1.25)
%
%      elseif i == 5;
%
%       plot(100*[bvu_4],mvd_all_4(:,i),':d','color',[0 0.5 0],'linewidth',1.5)
%    plot(100*[bvu_4],mvd_all_4(:,i+1),'--x','color',[0 0.5 0],'linewidth',1.25)
%      elseif i == 7;
%
```

```
%         plot(100*[bvu_4],mvd_all_4(:,i),':d','color',[0.25 0.25 1],'linewidth',1.5)
%      plot(100*[bvu_4],mvd_all_4(:,i+1),'--x','color',[0.25 0.25 1],'linewidth',1.25)
%     end
%     j = j +1;
% end

% title('Material: Ti64');
% legend('U_m_a_x = 10 Build Cycles | Material Cost','U_m_a_x = 10 Build Cycles | Depreciation Cost' , 'U_m_a_x = 30 Build Cycles | Material Cost','U_m_a_x = 20 Build Cycles | Depreciation Cost', 'U_m_a_x = 30 Build Cycles | Material Cost','U_m_a_x = 30 Build Cycles | Depreciation Cost','U_m_a_x = 40 Build Cycles | Material Cost','U_m_a_x = 40 Build Cycles | Depreciation Cost')

% grid on
% xlabel('Build Volume Utilization (%)');
% ylabel('Percentage of Total Costs (%)');
```

Appendix B
Cost Model for Powder Feedstock Depreciation MATLAB Code II

```
%% Cost Modeling for Reused Powder Feedstocks in Laser Powder Bed Fusion
% Michael W. Barclift - Penn State University - University Park, PA 16801
% mzb5747@psu.edu | mwb81@vt.edu |
% Mass Production of SAE Upright from 1-1000 Units Cost Modeling Code

%%
clc
clear all
close all
format long g

%% Initilization
xmax = 250.8; %Buildplate width on EOS M280 (mm)
ymax = 250.8; %Buildplate length on EOS M280 (mm)

dxmax = 250.8; %Dispenser width on EOS M280 (mm)
dymax = 227.80625; %Dispenser length on EOS M280 (mm)

Mat.wden = [7.8,8.15,2.67,4.41]; %Density of solidified powder feedstock (g/cm^3)
Mat.tapden = [5.3, 5.1, 1.5,2.74]; %Density of un-melted powder feedstock (g/cm^3)
Mat.buildrate = [7.2,14.4,26.6,13.5]; %Average time needed by AM machine to solidfy a voxel (hr/cm^3)
Mat.layer = [20,40,30,30]; %Layer Thickness (microns)
Mat.vprice = [105, 192,152,680]; %Price of virgin powder feedstock ($/kg)
Mat.salvage = 0; %Estimated value of powder feedstock at end of useful life (hr)
Mat.life = 0; %Estimated useful life of powder feedstock in build cycles (-)
Mat.use1 = 0; %Selected cycle in life of powder feedstock
Mat.waste = 0.4; %Percentage of powder lost during AM build process (%)
Mat.trapped = 0.25; %Percentage of powder trapped in Support Structures
Mat.charge = 2.25; %Amount of excess powder added per layer

Cost.oper = 110; %AM machine operator's cost ($/hr)
Cost.pc = 100; %Cost of the computer workstation ($/hr)
Cost.mach = 60; %Cost of the AM machine during build operation ($/hr)
Cost.stress = 350; %Cost to stress relief components on build substrate ($)
Cost.EDM = 200; %Cost to wire-EDM components on AM build substrate ($)
Cost.gas = 10; %Cost to use inert gas during AM build process ($/hr)
Cost.tools = 50; %Cost to use post-processing tools/equipment ($/hr)

Time.prep = [3]; %Time to generate support structures for digital models (hr)
Time.buildjob = 1; %Time to compile and arrange geometries on the build tray
Time.setup = 2; %Time to setup AM machine, gas, software, and pre-processes (hr)
Time.change = 0; %Time to change loaded powder, clean machine, filters, and reload (hr)
Time.recoat = 9; %Time AM machine needs to spread a new layer of powder (sec)
Time.buildrate = [7.2,14.4,26.6,13.5]; %Average time needed by AM machine to solidfy a voxel (hr/cm^3)
Time.removal = 3; %Time to remove build substrate from AM machine, sieve powder, clean, and documentation (hr)
Time.postp = 3; %Time required to post-process individual parts (hr)

%% User Inputs
Part.zheight = [172]; %Build height of parts (mm)
Part.vol = [175]; %Part geometric volume (cm^3)
Part.supports = [412]; %Part goemetric volume for supports (cm^3)
Part.totalvol = Part.vol + Part.supports; %Total volume (cm^3)
%Part.dispvol = Mat.charge*[2484.375,2484.375,2506.88]; %Volume to fill build chamber (cm^3)
%Part.dispvol = Mat.charge*[Part.zheight.*dxmax*dymax]/10^3; %Volume to fill build chamber (cm^3)
Part.dispvol = Mat.charge*[Part.zheight.*dxmax*dymax]/10^3; %Volume to fill build chamber (cm^3)
```

```
%------------------EDIT HERE TO CHANGE NUMBER OF PARTS--------------------
%  for mchoice = 1:4 %Material choice
   pcc_count = 0;
%     for pcc = 1:3 %Part count cases selector
      Part.count.cases = [1,2,4,6,8]; %Scenarios of part count cases

%        Part.N = Part.count.cases(pcc)*[3,3,3]; %Number of replicates for a geometry (-)
      Part.N = [1000];
%        blendrate = 0.2;

      Part.postp = [1]; %Parts needing to be post-processing (True (1) False (0))
      i = 1; %Selected Part Volume
      p = 1; %Selected Part Geometries to Include in Cost Analysis
      j = 1; %Selected AM Process

      %------------------EDIT HERE TO CHANGE MATERIAL--------------------
        mchoice = 4;
      k = mchoice; %Selected Material [GPI - 1, IN718 - 2, AlSi10Mg -3, Ti64 -4]

      %% Build Time Estimate for Powder Be Fusion
      %1. Complie build tray - part data
      PartData = [Part.zheight;Part.totalvol;Part.dispvol;Part.N];

      %2. Sort all part data by increasing z-height
      PartData = sortrows(PartData',1)';

      %3. Convert z-height to layers
      PartData(1,:) = round(PartData(1,:)*1000/Mat.layer(k));

      %4. Calculate layerwise recoating time allocation
      layer = PartData(1,1:p);
      Time.rc = zeros(1,p);

      for u = 0:(p-1)
         if u == 0
            Time.rc(1,u+1) = 1/60*1/60*Time.recoat*(layer(u+1)-0)./sum(PartData(4,(u+1):p));
         elseif u > 0
            Time.rc(1,u+1) = Time.rc(1,u) + 1/60*1/60*Time.recoat*(layer(u+1)-layer(u))./sum(PartData(4,(u+1):p));
         end
      end

      Time.trc = Time.recoat*layer(u+1)*1/60*1/60; %Total recoating time for the buildjob (hr)

      %-------Editing Build Time Estimate to be for whole build--------
      %Time.exp = Part.N*PartData(2,1:p).*1/Mat.buildrate(k); %Total time for solidfying each part volume (hr)
      Time.exp = PartData(2,1:p).*1/Mat.buildrate(k);

      Time.delay = 0/sum(PartData(4,1:p)); %Total time for heat, cooling, inactive (hr)

        Time.buildt = (Time.rc + Time.exp);%CHANGE(Time.rc.*Pmax)+(Time.exp.*Pmax);
      %% Part Mass Calculation
      Part.vol = PartData(2,1:p);
      Part.bed = PartData(3,1:p);

      %% Iterative Loop Structure
%        Mat.lifecases = [1,1,1,1];
       Mat.lifecases = [10,10,30,30,10,10,30,30];
```

```
    for index = 1:length(Mat.lifecases)

% Mat.lifecases = [10,30,10,30]-0*[1,1,1,1]; %Scenarios of Estimated Life for Powder Feedstock
        count = 1;
        Mat.life = Mat.lifecases(index);

        %% Depreciation Cost Calculation
        % SLN
        %    Dep = @(cmo,S,U,u) (cmo)-(u).*cmo/(U);
        %    Mat.value = Mat.vprice(k);
        %    for u = 2:Mat.life
        %        Mat.value(u) = Dep(Mat.value(1),Mat.salvage,Mat.life,(u-1));
        %    end
        %    Mat.value(u+1) = 0;

        % SOYD DEPRECIATION METHOD
        Dep = @(cmu,cmo,S,U,u) cmu-(cmo-S)*(U-u+1)./(U*(U+1)/2);
        Mat.value = Mat.vprice(k);

        if rem(index,2)==0
            Mat.value=Mat.value*0.4;
        end

        for u = 2:Mat.life
            Mat.value(u) = Dep(Mat.value(u-1),Mat.value(1),Mat.salvage,Mat.life,(u-1));
        end
        Mat.value(u+1) = 0;

        if index>4
           Mat.value = fliplr(0:Mat.value/Mat.lifecases(index):Mat.value);
        end

%           % Double Declining Balance Method
%           Dep = @(cmu,U,u) cmu - 2/U.*(cmu);
%           Mat.value3 = Mat.vprice(k);
%           for u = 2:Mat.life
%               Mat.value3(u) = Dep(Mat.value3(u-1),Mat.life,(u-1));
%           end
%           Mat.value3(u+1)=0;


%           for index2 = 1:length(Mat.lifecases)
             Mat.use1 = 1;

            %% Calculation of Costing
            Part.bedmass = Part.bed(1:p).*Mat.tapden(k)/1000;
            Part.bedmass = unique(Part.bedmass)';
            %Cost.dep = (Part.bedmass(p)-sum(Part.mass.*Part.N(1:p))).*(Mat.value(1)-Mat.value(2));

            %% Calculate Part Volumes Partitions for Layerwise Depreciation

            Part.volp = length(Part.bedmass); %Minimum of number volume partitions
            Part.volpi = zeros(Part.volp,p); %Intialize Part Volume Partitions Input

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % USER MUST INPUT DATA FOR VOLUME AT PARTITIONS
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

            % DATA FOR PENCIL THRUSTERS
            %  Part.volpi = [15.615,15.698,9.40;0,0,0.05]; %User Input Partition Data
            %  %OLD DATA WITH SUPPORTS STILL IN VOLUME
%              Part.volpi = [14.91, 15.39,9.40;0,0,0.05];
%              Part.supports = [0.705,0.308,0;0,0,0];

             Part.volpi = Part.totalvol;
             Part.volpi = Part.vol-Part.supports;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

            Part.N = PartData(4,1:p);
            Part.mass = 1/1000*((1+Mat.waste)*(Part.volpi + Part.supports))*Mat.wden(k) +
1/1000*(Mat.trapped*Part.supports)*Mat.tapden(k);
            Part.N = logical(Part.volpi).*repmat(Part.N,Part.volp,1);

         Cost.total = zeros(Part.N,10);
         totalN = Part.N;
         blendrate = (3*Part.mass)./Part.bedmass;

         for zz=1:totalN

             builds = ceil(zz/3);

              if zz<=3
                Part.N = zz;
                Part.depmass = (Part.bedmass-Part.N*Part.mass); %Mass of Depreciated Powder in the Bed
                Part.depma = (Part.N.*Part.mass)./sum(Part.N.*Part.mass); %This is the percentage of depreciated mass
allocated to each part based on mass
                Part.dep = Part.depmass.*(Mat.value(0+1)-Mat.value(1+1));
                Cost.build.mat = Part.N*Part.mass*Mat.value(Mat.use1);
                Cost.infmat = Part.N*Part.mass*Mat.value(1);
                Time.exp = Part.N*Part.vol.*1/Mat.buildrate(k);
                Time.buildt = (Time.trc + Time.exp);

                Cost.mix = blendrate*Part.bedmass*(Mat.value(1)-Mat.value(Mat.use1));

                Cost.build.mach = (Cost.mach+Cost.gas)*Time.buildt;
              else
                  Cost.build.mach = 0;
                  Cost.build.mat = 0;
                  Cost.infmat = 0;
                  Cost.mix=0;
                  Part.dep=0;

                  Mat.use1 = 1;

                  for bb=1:builds

%                         if (Mat.use1-1)==Mat.lifecases(index)
%                         Mat.use1 = 1;
                      if bb>1
                          if rem((bb-1),Mat.lifecases(index))==0
                          Mat.use1=1;
                          else
```

```
               Mat.use1=Mat.use1+1;
               end
             end

             if bb==builds
             Part.N=rem(zz,3);

              if Part.N==0
               Part.N=3;
              end

             else
             Part.N = 3;
             end

             Part.depmass = (Part.bedmass-Part.N*Part.mass); %Mass of Depreciated Powder in the Bed
             Part.dep = Part.dep + Part.depmass.*(Mat.value(Mat.use1)-Mat.value(Mat.use1+1));
             Cost.build.mat = Cost.build.mat + Part.N*Part.mass*Mat.value(Mat.use1);
             Cost.infmat = Cost.infmat + Part.N*Part.mass*Mat.value(1);
             Time.exp = Part.N*Part.vol.*1/Mat.buildrate(k);
             Time.buildt = (Time.trc + Time.exp);
             Cost.mix = Cost.mix + blendrate*Part.bedmass*(Mat.value(1)-Mat.value(Mat.use1));
             Cost.build.mach = Cost.build.mach + (Cost.mach+Cost.gas)*Time.buildt;
           end
         end

        Cost.prep(zz) = (Cost.oper + Cost.pc)*Time.prep;
        Cost.buildjob(zz) = (Cost.oper + Cost.pc)*Time.buildjob;
        Cost.setup(zz) = (Cost.oper + Cost.mach)*(Time.setup + Time.change)*(ceil(zz/3));
        Cost.deptotal(zz) = Part.dep;
        Cost.buildmat(zz) = Cost.build.mat;
        Cost.mixtotal(zz) = Cost.mix;
        Cost.infm(zz) = Cost.infmat;
        Cost.buildmach(zz) = Cost.build.mach;
        Cost.removal(zz) = (Cost.oper + Cost.mach)*(Time.removal)*(ceil(zz/3));
        Cost.substrate(zz) = (Cost.stress+ Cost.EDM)*(ceil(zz/3));

        % FIX POST PROCESSING COSTS FOR EACH MODEL
        Cost.postp(zz) = Time.postp*(Cost.oper+Cost.tools).*zz;

        Cost.total(zz,:) =
[Cost.prep(zz),Cost.buildjob(zz),Cost.setup(zz),Cost.buildmat(zz),Cost.deptotal(zz),Cost.mixtotal(zz),Cost.buildmach(
zz),Cost.removal(zz),Cost.substrate(zz),Cost.postp(zz)];
        Cost.inf(zz,:) =
[Cost.prep(zz),Cost.buildjob(zz),Cost.setup(zz),Cost.infm(zz),Cost.buildmach(zz),Cost.removal(zz),Cost.substrate(zz),
Cost.postp(zz)];

      end

        c1(:,:,index)=Cost.total;
        c2(:,:,index)=Cost.inf;

        %% Clearing of Data
        Part.depmass = []; %Clearing variables for re-calculation
        Part.depma = []; %Clearing variables for re-calculation
        Part.dep = []; %Clearing variables for re-calculation
        s = []; %Clearing variables for re-calculation
        Part.N = []; %Clearing variables for re-calculation
        Part.mass = []; %Clearing variables for re-calculation
%       end
```

```
            count = count + 1;
        end


        t=sum(Cost.total,2)';
        semilogx(t./(1:totalN),'r');
        xlabel('Number of Units')
        ylabel('Cost per Part($)')

        hold on

        p=sum(Cost.inf,2)';
        semilogx(p./(1:totalN),'b');

        grid on

        data = [Cost.total(1,:)',Cost.total(10,:)'/10,Cost.total(100,:)'/100,Cost.total(totalN,:)'/totalN];
        g = sum(data);
        data2 = [data(:,1)./g(1),data(:,2)./g(2),data(:,3)./g(3),data(:,4)./g(4)]*100;
        figure

        colormap('jet')
        bar(flip(data2)','stacked')

        figure
        semilogx(sum(c1(:,:,1),2)'./(1:totalN),'r')
        hold on
        semilogx(sum(c1(:,:,2),2)'./(1:totalN),'r--')
        semilogx(sum(c1(:,:,3),2)'./(1:totalN),'b')
        semilogx(sum(c1(:,:,4),2)'./(1:totalN),'b--')

        semilogx(sum(c1(:,:,5),2)'./(1:totalN),'c')
        semilogx(sum(c1(:,:,6),2)'./(1:totalN),'c--')
        semilogx(sum(c1(:,:,7),2)'./(1:totalN),'m')
        semilogx(sum(c1(:,:,8),2)'./(1:totalN),'m--')


        semilogx(sum(c2(:,:,1),2)'./(1:totalN),'g')
        semilogx(sum(c2(:,:,2),2)'./(1:totalN),'g--')
        grid on
        xlabel('Number of Units')
        ylabel('Cost per Part($)')

        grid on

        colormap('jet')

figure

subplot(1,2,1,'XScale','log')
box on
xlim([1,totalN]);
hold on

ratio1(1,:) = (sum(c1(:,:,1),2)'./(1:totalN))./(sum(c2(:,:,1),2)'./(1:totalN));
ratio1(2,:) = (sum(c1(:,:,3),2)'./(1:totalN))./(sum(c2(:,:,1),2)'./(1:totalN));
ratio1(3,:) = (sum(c1(:,:,5),2)'./(1:totalN))./(sum(c2(:,:,1),2)'./(1:totalN));
ratio1(4,:) = (sum(c1(:,:,7),2)'./(1:totalN))./(sum(c2(:,:,1),2)'./(1:totalN));
ratio1(5,:) = (sum(c2(:,:,1),2)'./(1:totalN))./(sum(c2(:,:,1),2)'./(1:totalN));
```

```
semilogx(ratio1(1,:),'r','LineWidth',1.5)
semilogx(ratio1(2,:),'b','LineWidth',1.5)
semilogx(ratio1(3,:),'c','LineWidth',1.5)
semilogx(ratio1(4,:),'m','LineWidth',1.5)
semilogx(ratio1(5,:),'color',[0 0.5 0],'LineWidth',1.5)

yscale = ylim;

legend('U_m_a_x = 10 Build Cycles | SOYD',...
    'U_m_a_x = 30 Build Cycles | SOYD',...
    'U_m_a_x = 10 Build Cycles | SLN',...
    'U_m_a_x = 30 Build Cycles | SLN',...
    'U_m_a_x = \infty')

grid on
title('Ti64 - $680/kg')
xlabel('Number of Parts')
ylabel('Normalized Cost per Part')

% figure
subplot(1,2,2,'XScale', 'log')
box on
hold on

ratio1(1,:) = (sum(c1(:,:,2),2)'./(1:totalN))./(sum(c2(:,:,2),2)'./(1:totalN));
ratio1(2,:) = (sum(c1(:,:,4),2)'./(1:totalN))./(sum(c2(:,:,2),2)'./(1:totalN));
ratio1(3,:) = (sum(c1(:,:,6),2)'./(1:totalN))./(sum(c2(:,:,2),2)'./(1:totalN));
ratio1(4,:) = (sum(c1(:,:,8),2)'./(1:totalN))./(sum(c2(:,:,2),2)'./(1:totalN));
ratio1(5,:) = (sum(c2(:,:,2),2)'./(1:totalN))./(sum(c2(:,:,2),2)'./(1:totalN));

% semilogx(ratio1')
semilogx(ratio1(1,:),'r','LineWidth',1.5)
semilogx(ratio1(2,:),'b','LineWidth',1.5)
semilogx(ratio1(3,:),'c','LineWidth',1.5)
semilogx(ratio1(4,:),'m','LineWidth',1.5)
semilogx(ratio1(5,:),'color',[0 0.5 0],'LineWidth',1.5)
ylim(yscale)

legend('U_m_a_x = 10 Build Cycles | SOYD',...
    'U_m_a_x = 30 Build Cycles | SOYD',...
    'U_m_a_x = 10 Build Cycles | SLN',...
    'U_m_a_x = 30 Build Cycles | SLN',...
    'U_m_a_x = \infty')

xlim([1,totalN]);
grid on
title('Ti64 - $272/kg')
xlabel('Number of Parts')
ylabel('Normalized Cost per Part')
```

Appendix C
CAD-Integrated Cost Estimator Macro Initialize Code

```
Dim swApp As Object
Sub main()

Set swApp = Application.SldWorks
UserForm2.Show vbModeless

End Sub
```

Appendix D
CAD-Integrated Cost Estimator Userform Code

```
Option Explicit
'==============================================================================
'CAD-INTEGRATED COST ESTIMATOR FOR ADDITIVE MANUFACTURING
' Michael Barclift
' Pennsylvania State University
' July 16, 2018
' mzb5747@psu.edu

'CONTRIBUTORS
'Andrew Armstrong

'REVIEWERS
'Timothy Simpson
'Nicholas Meisel
'Sanjay Joshi
'===============================================================================


' --------------Variable Definitions-----------------

   Dim swApp As SldWorks.SldWorks
   Dim swModel As SldWorks.ModelDoc2
   Dim swModel_part As SldWorks.ModelDoc2
   Dim swModDocExt As SldWorks.ModelDocExtension
   Dim swMass As SldWorks.MassProperty
   Dim swSupports As SldWorks.MassProperty
   Dim swSelMgr As SldWorks.SelectionMgr
   Dim swSelData As SldWorks.SelectData
   Dim swPlane As SldWorks.RefPlane
   Dim swSketch As SldWorks.Sketch
   Dim bv_corners(6) As Double
   Dim swModelView As ModelView
   Dim support_volume As Double
   Dim support_volume_raw As Double
   Dim support_volume_actual As Double
   Dim Body As Variant
   Dim optim As Boolean
   Dim k As Integer
   Dim supportsurfaces As Integer

   Dim cboxnum As Double
   Dim partvolume As Double
```

```
Public matdensity As Double
Public matcost As Double

Dim FilePath As String
Dim uuni As Boolean
Dim massi As Boolean
Dim LineFromFile As String
Dim LineItems As Variant
Dim am_machine As String
Dim am_material As String
Dim num As Integer
Dim corners(6) As Double
Dim swSketchPt(8) As SldWorks.SketchPoint
Dim swSketchSeg(12) As SldWorks.SketchSegment

Dim boolstatus As Boolean
Dim bRet As Boolean
Dim parea As Double
Dim xmax As Double
Dim ymax As Double
Dim zmax As Double
Dim partvol As Double
Dim slice As Integer
Dim layert As Double
Dim pratio As Double

Dim count As Integer
Dim xrotv As Double
Dim yrotv As Double
Dim zrotv As Double
Dim numnum As Integer
Dim SpptCount As Integer
Dim startval As Integer

Dim X_max            As Double
Dim X_min            As Double
Dim Y_max            As Double
Dim Y_min            As Double
Dim Z_max            As Double
Dim Z_min            As Double

'=================================
'Variables for Support Generation
'=================================
  Dim Y_gap As Double
Dim Support_Radius As Double
Dim Support_Rad_Y As Double
Dim Support_Rad_X As Double
Dim swSketchSegment As SldWorks.SketchSegment
Dim swSketchMgr As SldWorks.SketchManager
Dim Min As String
Dim T As Single
Dim Y_inc As Double
Dim swBodySelect As Variant
Dim num_in_scan As Integer
Dim Rp As Integer

'Definition of dynamic arrays
Dim xc_array() As Double
Dim yc_array() As Double
Dim xc_array_up() As Double
Dim yc_array_up() As Double
Dim zc_array_up() As Double
Dim zc_array_h() As Double
Dim SpptCount2 As Integer
Dim X_inc As Double
Dim X_gap As Double
Dim Zray As Double
Dim res As Double
Dim Build_Direction(2) As Double
Dim itr As Integer

'Ray Trace Projectors
Dim shoot_center(2) As Double

  Const hitRadius As Double = 0.0000095
  Const offset As Double = 0.0000001
  Dim center_vPts As Variant

  'Center points
  Dim xc As Double
  Dim yc As Double
  Dim zc As Double

  'Vector Z values
  Dim hit_center As Integer
  Dim oangle As Double
  Dim oangle2 As Double
  Dim p As Integer

  Dim myFeature As Object
  Dim color As Variant
```

```
    Dim swFeat2 As Object
    Dim skSegment As Object
    Dim support As Object
    Dim swFeat As SldWorks.Feature
    Dim calc_sv As Double
    Dim totalheight()
    Dim totalheight2()
    Dim tempvar(8) As Double

    Dim swModeler As SldWorks.Modeler
    Dim swBody As SldWorks.Body2
      Dim dblData(8) As Double



Private Sub Label543_Click()

End Sub

  Private Sub UserForm_initialize()

  Set swApp = Application.SldWorks
  Set swModel = swApp.ActiveDoc
  Set swModelView = swModel.GetFirstModelView
  Set swModDocExt = swModel.Extension
  Set swMass = swModDocExt.CreateMassProperty
  Set swModeler = swApp.GetModeler

  swModelView.EnableGraphicsUpdate = False
  boolstatus = swModel.Extension.HideFeatureManager(True)

  eos_rr.Value = 10

' --------------Load Material Data----------------
  'FilePath = "X:\Downloads\AM_Material_Data.csv"
  FilePath = "C:\AM_Costing_Tool\AM_Material_Data.csv"
  Open FilePath For Input As #1
  num = 0
  numnum = 0

  Do Until EOF(1)
  Line Input #1, LineFromFile
  LineItems = Split(LineFromFile, ",")

' --------------Populate Data Fields----------------
  am_machine = LineItems(0)
```

```
am_material = LineItems(4)
count = 0

If num = 0 Then
ElseIf num > 0 Then

If am_machine = "EOSINT M280" Then
eos_mat.AddItem (CStr(am_material))

eos_bpx.Value = CStr(LineItems(1))
eos_bpy.Value = CStr(LineItems(2))
eos_bpz.Value = CStr(LineItems(3))

eos_mp.AddItem (CStr(LineItems(5)))
eos_br.AddItem (CStr(LineItems(8)))
eos_lt.AddItem (CStr(LineItems(6)))
eos_td.AddItem (CStr(LineItems(7)))
eos_wd.AddItem (CStr(LineItems(9)))

'ElseIf am_machine = "Arcam S12" Then
'arcam_mat.AddItem (CStr(am_material))

'arcam_bpx.Value = CStr(LineItems(1))
'arcam_bpy.Value = CStr(LineItems(2))
'arcam_bpz.Value = CStr(LineItems(3))

'arcam_mp.AddItem (CStr(LineItems(5)))
'arcam_br.AddItem (CStr(LineItems(8)))
'arcam_lt.AddItem (CStr(LineItems(6)))
'arcam_td.AddItem (CStr(LineItems(7)))

'ElseIf am_machine = "Optomec MR-7" Then
'opto_mat.AddItem (CStr(am_material))

'opto_bpx.Value = CStr(LineItems(1))
'opto_bpy.Value = CStr(LineItems(2))
'opto_bpz.Value = CStr(LineItems(3))

'opto_mp.AddItem (CStr(LineItems(5)))
'opto_mfr.AddItem (CStr(LineItems(8)))
'opto_td.AddItem (CStr(LineItems(7)))

End If

End If
```

```
   'Old code to remove redundancy
   'If num > 2 Then
   'If wordsold <> words Then
   'mbox.AddItem (CStr(words))
   'matbox.AddItem (CStr(words2))
   'Else
   'matbox.AddItem (CStr(words2))
   'End If
   'End If

   num = num + 1
   Loop

Close #1

   deleteall

   startval = 0
  swModel.ClearSelection
  boolstatus = swModel.Extension.SelectByID2("Build Volume", "SKETCH", 0, 0, 0, False, 0,
Nothing, 0)
   If boolstatus Then
   startval = startval + 1
   End If

   swModel.ClearSelection
   boolstatus = swModel.Extension.SelectByID2("Build Platform", "BODYFEATURE", 0, 0, 0, False,
0, Nothing, 0)
   If boolstatus Then
   startval = startval + 1
   End If

   boolstatus = swModel.Extension.SelectByID2("Support Structures", "BODYFEATURE", 0, 0, 0,
False, 0, Nothing, 0)
   If boolstatus Then
   startval = startval + 1

   End If
  swModel.ClearSelection

  'Global Definition of Part Volume
   partvolume = swMass.Volume
  partvolume = (partvolume * 100 * 100 * 100) '1615.91 volume of substrate
  partvolume = Format(partvolume, "0.00")
  eos_compv.Value = partvolume
  If startval > 2 Then
  QuickDataInitialize
  Else
  ModelInitialize
   DataInitialize
  End If

  SpptCount = 0
  swModelView.EnableGraphicsUpdate = True
  boolstatus = swModel.Extension.HideFeatureManager(False)

 End Sub
   Function ModelInitialize()

   swModel.SetDisplayWhenAdded False
   BoundingBoxCode
   buildplate
   swModel.SetDisplayWhenAdded True

   End Function

   Function DataInitialize()

' -----------Read Geometry Bounding Box----------------------
  xmax = (corners(3) - corners(0)) * 1000
  ymax = (corners(4) - corners(1)) * 1000
  zmax = (corners(5) - corners(2)) * 1000

  xmax = Format(xmax, "0.00")
  ymax = Format(ymax, "0.00")
  zmax = Format(zmax, "0.00")

  eos_compx.Value = xmax
  eos_compy.Value = ymax
  eos_compz.Value = zmax

  parea = (xmax * eos_compy.Value) / (eos_bpx.Value * eos_bpy.Value)
  parea = Format(parea, "0.00")
  eos_compa.Value = parea * 100

  ' -----------Read Geometry Data----------------------
  'Stop using default units
  'swMass.UseSystemUnits = False
  'uuni =
swModDocExt.SetUserPreferenceInteger(swUserPreferenceIntegerValue_e.swUnitSystem,
swUserPreferenceOption_e.swDetailingNoOptionSpecified,
swUnitSystem_e.swUnitSystem_Custom)
  'massi =
swModDocExt.SetUserPreferenceInteger(swUserPreferenceIntegerValue_e.swUnitsMassPropLe
ngth, swUserPreferenceOption_e.swDetailingNoOptionSpecified, swLengthUnit_e.swMETER)

  pratio = 100 * partvolume / (xmax * ymax * zmax / 10 / 10 / 10)
  pratio = Format(pratio, "0.00")
  eos_pack.Value = pratio

   End Function
   Function QuickDataInitialize()

   Const MaxDouble        As Double = 1.79769313486231E+308
   Const MinDouble        As Double = -1.79769313486231E+308

   X_max = MinDouble
   X_min = MaxDouble
   Y_max = MinDouble
   Y_min = MaxDouble
   Z_max = MinDouble
   Z_min = MaxDouble

   ' Solid body

   Dim vBodies            As Variant

   vBodies = swModel.GetBodies2(swSolidBody, False)

   Dim i                  As Long

  ProcessBodies vBodies, X_max, X_min, Y_max, Y_min, Z_max, Z_min

   'Actual Corners of Part Bounding Box
   corners(0) = X_min
   corners(1) = Y_min
   corners(2) = Z_min
   corners(3) = X_max
   corners(4) = Y_max
   corners(5) = Z_max

   DataInitialize

   End Function

Function BoundingBoxCode()
```

```
'==========================
'Perfect Bounding Box Code
'==========================

   boolstatus = swModel.Extension.SelectByID2("Build Volume", "SKETCH", 0, 0, 0, False, 0,
Nothing, 0)
   If boolstatus = True Then
   swModel.EditDelete
   End If

   'Get point data for bounding box along XYZ axes
   'corners = swModel.GetPartBox(True) - Old Method 1/9/2017
   swModel.Insert3DSketch2 True
   swModel.SetAddToDB True

   Const MaxDouble        As Double = 1.79769313486231E+308
   Const MinDouble        As Double = -1.79769313486231E+308

   X_max = MinDouble
   X_min = MaxDouble
   Y_max = MinDouble
   Y_min = MaxDouble
   Z_max = MinDouble
   Z_min = MaxDouble

   ' Solid body

   Dim vBodies            As Variant

   vBodies = swModel.GetBodies2(swSolidBody, False)

   Dim i                  As Long

  ProcessBodies vBodies, X_max, X_min, Y_max, Y_min, Z_max, Z_min

   'Corners for Build Volume
   corners(0) = (X_min + X_max) / 2 - 0.125
   corners(1) = (Y_min + Y_max) / 2 - 0.125
   corners(2) = (Z_min - 0.002)
   corners(3) = (X_min + X_max) / 2 + 0.125
   corners(4) = (Y_min + Y_max) / 2 + 0.125
   corners(5) = Z_min + 0.304

   'Build Volume Corners for Global Reference
   bv_corners(0) = corners(0)
   bv_corners(1) = corners(1)
```

```
    bv_corners(2) = corners(2)
    bv_corners(3) = corners(3)
    bv_corners(4) = corners(4)
    bv_corners(5) = corners(5)


' --------------Draw Bounding Box Code-----------------
' REFERENCE: Bounding Box Code originally coded by Wayne Tiffany, Oct 12, 2004 - Updated
10/15/04
' Accessed and Modified by Michael Barclift, on Dec 7, 2015 at
www.soldworks.com/forums/APIHelp
' START OF BOUNDING BOX-DRAW CODE HERE

  'Draw points at each corner of bounding box
  Set swSketchPt(0) = swModel.CreatePoint2(corners(3), corners(1), corners(5))
  Set swSketchPt(1) = swModel.CreatePoint2(corners(0), corners(1), corners(5))
  Set swSketchPt(2) = swModel.CreatePoint2(corners(0), corners(1), corners(2))
  Set swSketchPt(3) = swModel.CreatePoint2(corners(3), corners(1), corners(2))
  Set swSketchPt(4) = swModel.CreatePoint2(corners(3), corners(4), corners(5))
  Set swSketchPt(5) = swModel.CreatePoint2(corners(0), corners(4), corners(5))
  Set swSketchPt(6) = swModel.CreatePoint2(corners(0), corners(4), corners(2))
  Set swSketchPt(7) = swModel.CreatePoint2(corners(3), corners(4), corners(2))

  ' Now draw bounding box
  Set swSketchSeg(0) = swModel.CreateLine2(swSketchPt(0).x, swSketchPt(0).Y, swSketchPt(0).Z,
swSketchPt(1).x, swSketchPt(1).Y, swSketchPt(1).Z)
  Set swSketchSeg(1) = swModel.CreateLine2(swSketchPt(1).x, swSketchPt(1).Y, swSketchPt(1).Z,
swSketchPt(2).x, swSketchPt(2).Y, swSketchPt(2).Z)
  Set swSketchSeg(2) = swModel.CreateLine2(swSketchPt(2).x, swSketchPt(2).Y, swSketchPt(2).Z,
swSketchPt(3).x, swSketchPt(3).Y, swSketchPt(3).Z)
  Set swSketchSeg(3) = swModel.CreateLine2(swSketchPt(3).x, swSketchPt(3).Y, swSketchPt(3).Z,
swSketchPt(0).x, swSketchPt(0).Y, swSketchPt(0).Z)
  Set swSketchSeg(4) = swModel.CreateLine2(swSketchPt(0).x, swSketchPt(0).Y, swSketchPt(0).Z,
swSketchPt(4).x, swSketchPt(4).Y, swSketchPt(4).Z)
  Set swSketchSeg(5) = swModel.CreateLine2(swSketchPt(1).x, swSketchPt(1).Y, swSketchPt(1).Z,
swSketchPt(5).x, swSketchPt(5).Y, swSketchPt(5).Z)
  Set swSketchSeg(6) = swModel.CreateLine2(swSketchPt(2).x, swSketchPt(2).Y, swSketchPt(2).Z,
swSketchPt(6).x, swSketchPt(6).Y, swSketchPt(6).Z)
  Set swSketchSeg(7) = swModel.CreateLine2(swSketchPt(3).x, swSketchPt(3).Y, swSketchPt(3).Z,
swSketchPt(7).x, swSketchPt(7).Y, swSketchPt(7).Z)
  Set swSketchSeg(8) = swModel.CreateLine2(swSketchPt(4).x, swSketchPt(4).Y, swSketchPt(4).Z,
swSketchPt(5).x, swSketchPt(5).Y, swSketchPt(5).Z)
  Set swSketchSeg(9) = swModel.CreateLine2(swSketchPt(5).x, swSketchPt(5).Y, swSketchPt(5).Z,
swSketchPt(6).x, swSketchPt(6).Y, swSketchPt(6).Z)
  Set swSketchSeg(10) = swModel.CreateLine2(swSketchPt(6).x, swSketchPt(6).Y,
swSketchPt(6).Z, swSketchPt(7).x, swSketchPt(7).Y, swSketchPt(7).Z)
```

```
 Set swSketchSeg(11) = swModel.CreateLine2(swSketchPt(7).x, swSketchPt(7).Y,
swSketchPt(7).Z, swSketchPt(4).x, swSketchPt(4).Y, swSketchPt(4).Z)

Set swSketch = swModel.GetActiveSketch2
 swSketch.Name = "Build Volume"
 swModel.Insert3DSketch2 False

   'Actual Corners of Part Bounding Box
   corners(0) = X_min
   corners(1) = Y_min
   corners(2) = Z_min
   corners(3) = X_max
   corners(4) = Y_max
   corners(5) = Z_max

   bRet = swSketchPt(3).Select4(True, swSelData): Debug.Assert bRet
   bRet = swSketchPt(6).Select4(True, swSelData): Debug.Assert bRet
   bRet = swSketchPt(2).Select4(True, swSelData): Debug.Assert bRet

   Set swPlane = swModel.CreatePlaneThru3Points3(True)

   swPlane.Name = "Build Surface"

 swModel.SetAddToDB False

' END OF BOUNDING BOX-DRAW CODE HERE -----------------------

End Function

Private Sub Label540_Click()
   Dim swApp As SldWorks.SldWorks
   Dim swModel As SldWorks.ModelDoc2
   Dim swPart As SldWorks.PartDoc
   Dim vBodies As Variant
   Set swApp = Application.SldWorks
   Set swModel = swApp.ActiveDoc

   Set swPart = swModel
   vBodies = swPart.GetBodies2(swSolidBody, False)
   If IsEmpty(vBodies) Then Exit Sub
   Debug.Print UBound(vBodies) + 1
End Sub

Function buildplate()
'==============================
' Creates EOS M280 Build Plate
```

```
'==============================

Dim swFeatureManager As Object
Dim swModelDocExt As Object
Dim boolstatus As Boolean

Set swApp = Application.SldWorks
Set swModel = swApp.ActiveDoc
Set swFeatureManager = swModel.FeatureManager
Set swModelDocExt = swModel.Extension

'Delete the Old Material
swModel.ClearSelection
boolstatus = swModel.Extension.SelectByID2("Build Platform", "BODYFEATURE", 0, 0, 0, False, 0,
Nothing, 0)
   If boolstatus = True Then
   swModel.EditDelete
   End If

boolstatus = swModel.Extension.SelectByID2("Build Plate", "SKETCH", 0, 0, 0, False, 0, Nothing,
0)
   If boolstatus = True Then
   swModel.EditDelete
   End If

boolstatus = swModel.Extension.SelectByID2("Build Surface", "PLANE", 0, 0, 0, False, 0, Nothing,
0)
swModel.SketchManager.InsertSketch True
swModel.ClearSelection2 True

'Dim xmax As Double
'Dim ymax As Double
'Dim zmax As Double

'boolstatus =
swModel.Extension.SetUserPreferenceToggle(swUserPreferenceToggle_e.swSketchAddConstTo
RectEntity, swUserPreferenceOption_e.swDetailingNoOptionSpecified, True)
'boolstatus =
swModel.Extension.SetUserPreferenceToggle(swUserPreferenceToggle_e.swSketchAddConstLin
eDiagonalType, swUserPreferenceOption_e.swDetailingNoOptionSpecified, True)
'Dim vSkLines As Variant
'vSkLines = swModel.SketchManager.CreateCenterRectangle(c1, c2, c3, c1 + 0.125, c2 + 0.125,
c3)

Dim skPoint As Object
```

```
Set skPoint = swModel.SketchManager.CreatePoint(bv_corners(0), bv_corners(1), 0#)
Set skPoint = swModel.SketchManager.CreatePoint(bv_corners(0), bv_corners(4), 0#)
Set skPoint = swModel.SketchManager.CreatePoint(bv_corners(3), bv_corners(1), 0#)
Set skPoint = swModel.SketchManager.CreatePoint(bv_corners(3), bv_corners(4), 0#)

swModel.SetPickMode
swModel.ClearSelection2 True
Dim skSegment As Object
Set skSegment = swModel.SketchManager.CreateLine(bv_corners(0), bv_corners(1), 0#,
bv_corners(3), bv_corners(1), 0#)
Set skSegment = swModel.SketchManager.CreateLine(bv_corners(3), bv_corners(1), 0#,
bv_corners(3), bv_corners(4), 0#)
Set skSegment = swModel.SketchManager.CreateLine(bv_corners(3), bv_corners(4), 0#,
bv_corners(0), bv_corners(4), 0#)
Set skSegment = swModel.SketchManager.CreateLine(bv_corners(0), bv_corners(4), 0#,
bv_corners(0), bv_corners(1), 0#)

Dim swSketch As SldWorks.Sketch
Set swSketch = swModel.GetActiveSketch2
swSketch.Name = "Build_Plate"

'swModel.EditRebuild3

'swModel.SketchManager.InsertSketch False
'swModel.ClearSelection2 True
swModel.SketchManager.InsertSketch False
swModel.ClearSelection2 True

boolstatus = swModel.Extension.SelectByID2("Build_Plate", "SKETCH", 0, 0, 0, False, 0, Nothing,
0)
swModel.ClearSelection2 True
boolstatus = swModel.Extension.SelectByID2("Build_Plate", "SKETCH", 0, 0, 0, False, 4, Nothing,
0)
Dim swFeat As SldWorks.Feature
Set swFeat = swModel.FeatureManager.FeatureExtrusion2(True, False, True, 0, 0, 0.0254,
0.00254, False, False, False, False, 1.74532925199433E-02, 1.74532925199433E-02, False, False,
False, False, False, True, True, 0, 0, False)

'boolstatus = swModel.Extension.SelectByID2("Build Plate", "SKETCH", 0, 0, 0, False, 4, Nothing,
0)
'Dim name As Object
'Set myFeature = swModel.FeatureManager.FeatureExtrusion2(True, False, True, 0, 0, 0.0254,
0.00254, False, False, False, False, 1.74532925199433E-02, 1.74532925199433E-02, False, False,
False, False, False, True, True, 0, 0, False)
```

```
'Set myFeature = swModel.FeatureManager.FeatureExtrusion2(True, False, False, 2, 0, 0.01,
0.01, False, False, False, False, 1.74532925199433E-02, 1.74532925199433E-02, False, False,
False, False, False, True, True, 0, 0, False)
'name = myFeature.GetID

Dim swSelMgr As SldWorks.SelectionMgr
'Dim featName As String, featType As String
'Set swFeat = swModel.FeatureManager.FeatureExtrusion2(True, False, True, 0, 0, 0.0254,
0.00254, False, False, False, False, 1.74532925199433E-02, 1.74532925199433E-02, False, False,
False, False, False, True, True, 0, 0, False)
'swModel.EditRebuild3

If swSelMgr Is Nothing Then

Else
swFeat.Name = "Build Platform"

swModel.ClearSelection
swModel.SelectionManager.EnableContourSelection = False

Set swSelMgr = swModel.SelectionManager

boolstatus = swModel.Extension.SelectByID2("Build Platform", "BODYFEATURE", 0, 0, 0, False, 0,
Nothing, 0)

Set swFeat = swSelMgr.GetSelectedObject6(1, -1)

Dim color As Variant

'color = swFeat.GetMaterialPropertyValues2(swInConfigurationOpts_e.swThisConfiguration, "")
color = swFeat.GetMaterialPropertyValues2(1, Empty)

                color(0) = 1
                color(1) = 1
                color(2) = 0
                color(3) = 1
                color(4) = 1
                color(5) = 0.8
                color(6) = 0.3215
                color(7) = 0
                color(8) = 0

swFeat.SetMaterialPropertyValues2 color, swAllConfiguration, Empty

swModel.ClearSelection2 True
```

```
End If

'Set swSelMgr = swModel.SelectionManager

'Set swFeat = swSelMgr.GetSelectedObject6(1, -1)
'swFeat.name = "Build Platform"

'1/16/2017

swModel.ClearSelection
swModel.EditRebuild3

End Function


Private Sub CommandButton7_Click()

If eos_pe.Value = vbNullString Then

totalcost.Value = vbNullString

'ElseIf eos_pw.Value = vbNullString Then

totalcost.Value = vbNullString

ElseIf eos_mc.Value = vbNullString Then

totalcost.Value = vbNullString

Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = (eos_compmass.value) * eos_mp.value + (eos_mw.value / 100) *
eos_compmass.value * eos_mp.value + (eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")

End If

End Sub

Private Sub arcam_mat_Change()
count = arcam_mat.ListIndex

arcam_mp.ListIndex = count
arcam_lt.ListIndex = count
```

```
arcam_td.ListIndex = count
arcam_br.ListIndex = count
End Sub

Private Sub ComboBox1_Change()

'eos_compmass.value = eos_wrotd.value * eos_compv.value / 1000
'eos_compmass.value = Format(eos_compmass.value, "0.00")

End Sub

Function GetMax(Val1, Val2, Val3, Val4)

' Finds maximum of four values

  GetMax = Val1



  If Val2 > GetMax Then

    GetMax = Val2

  End If

  If Val3 > GetMax Then

    GetMax = Val3

  End If

  If Val4 > GetMax Then

    GetMax = Val4

  End If

End Function

Function GetMin(Val1, Val2, Val3, Val4)


' Finds minimum of four values

  GetMin = Val1
```

```
   If Val2 < GetMin Then
      GetMin = Val2
   End If
   If Val3 < GetMin Then
      GetMin = Val3
   End If
   If Val4 < GetMin Then
      GetMin = Val4
   End If
End Function
Function ProcessTessTriangles(vTessTriangles, X_max, X_min, Y_max, Y_min, Z_max, Z_min)
Dim i                 As Long
   For i = 0 To UBound(vTessTriangles) / (1 * 9) - 1
'      ' Debugging output only
'      Debug.Print "Pt(" + Str(i) + ") = "
'      Debug.Print " (" + _
'         Str(vTessTriangles(9 * i + 0)) + "," + _
'         Str(vTessTriangles(9 * i + 1)) + "," + _
'         Str(vTessTriangles(9 * i + 2)) + ")"
'      Debug.Print " (" + _
'         Str(vTessTriangles(9 * i + 3)) + "," + _
'         Str(vTessTriangles(9 * i + 4)) + "," + _
'          Str(vTessTriangles(9 * i + 5)) + ")"
'       Debug.Print " (" + _
'          Str(vTessTriangles(9 * i + 6)) + "," + _
'          Str(vTessTriangles(9 * i + 7)) + "," + _
'          Str(vTessTriangles(9 * i + 8)) + ")"


      X_max = GetMax((vTessTriangles(9 * i + 0)), (vTessTriangles(9 * i + 3)), (vTessTriangles(9 * i + 6)), X_max)
      X_min = GetMin((vTessTriangles(9 * i + 0)), (vTessTriangles(9 * i + 3)), (vTessTriangles(9 * i + 6)), X_min)


      Y_max = GetMax((vTessTriangles(9 * i + 1)), (vTessTriangles(9 * i + 4)), (vTessTriangles(9 * i + 7)), Y_max)
      Y_min = GetMin((vTessTriangles(9 * i + 1)), (vTessTriangles(9 * i + 4)), (vTessTriangles(9 * i + 7)), Y_min)


      Z_max = GetMax((vTessTriangles(9 * i + 2)), (vTessTriangles(9 * i + 5)), (vTessTriangles(9 * i + 8)), Z_max)
      Z_min = GetMin((vTessTriangles(9 * i + 2)), (vTessTriangles(9 * i + 5)), (vTessTriangles(9 * i + 8)), Z_min)
   Next i
'Finished with iterations
End Function
Sub ProcessBodies(vBodies, X_max, X_min, Y_max, Y_min, Z_max, Z_min)
Dim i                 As Long
Dim swBodyZ           As SldWorks.Body2
Dim swFace            As SldWorks.Face2
Dim vTessTriangles    As Variant
```

```
   ' Probably empty if no reference surfaces

   If IsEmpty(vBodies) Then Exit Sub

   For i = 0 To UBound(vBodies)

      Set swBodyZ = vBodies(i)

      Set swFace = swBodyZ.GetFirstFace

      While Not swFace Is Nothing

         vTessTriangles = swFace.GetTessTriangles(True)

         ProcessTessTriangles vTessTriangles, X_max, X_min, Y_max, Y_min, Z_max, Z_min

         Set swFace = swFace.GetNextFace

      Wend

   Next i

End Sub

Private Sub CommandButton10_Click()

boolstatus = generate_supports(eos_sangle.Value, eos_sres.Value)

 End Sub
Function deletesup()

'-----------------------------------------------------------
'Delete Previous Support Structures (Internal and External)
'-----------------------------------------------------------

Dim longstatus As Long
Dim DeleteOption As Long

swApp.CommandInProgress = True

boolstatus = swModel.Extension.SelectByID2("Internal Support Structures", "FTRFOLDER", 0, 0,
0, True, 0, Nothing, 0)
boolstatus = swModel.Extension.SelectByID2("External Support Structures", "FTRFOLDER", 0, 0,
0, True, 0, Nothing, 0)
```

```
DeleteOption = SwConst.swDelete_Absorbed
longstatus = swModel.Extension.DeleteSelection2(DeleteOption)

'swModel.EditDelete
swModel.ClearSelection


Dim itr As Integer
itr = 0
boolstatus = swModel.Extension.SelectByID2("Internal_Support_" + CStr(itr), "BODYFEATURE",
0, 0, 0, False, 0, Nothing, 0)
'boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
 'boolstatus = swModel.Extension.SelectByID2("Inner_", "SKETCH", 0, 0, 0, False, 4, Nothing, 0)

While boolstatus
longstatus = swModel.Extension.DeleteSelection2(DeleteOption)
swModel.EditDelete
itr = itr + 1
boolstatus = swModel.Extension.SelectByID2("Internal_Support_" + CStr(itr), "BODYFEATURE",
0, 0, 0, False, 0, Nothing, 0)
'boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
Wend
'swModel.EditDelete

itr = 0
boolstatus = swModel.Extension.SelectByID2("External_Support_" + CStr(itr), "BODYFEATURE",
0, 0, 0, False, 0, Nothing, 0)
'boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
 'boolstatus = swModel.Extension.SelectByID2("Inner_", "SKETCH", 0, 0, 0, False, 4, Nothing, 0)

While boolstatus
longstatus = swModel.Extension.DeleteSelection2(DeleteOption)
swModel.EditDelete
itr = itr + 1
boolstatus = swModel.Extension.SelectByID2("External_Support_" + CStr(itr), "BODYFEATURE",
0, 0, 0, False, 0, Nothing, 0)
'boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
Wend

swApp.CommandInProgress = False
'swModel.EditDelete
```

```
End Function

Function generate_supports(sangle, sres)

'==============================================================================
'DIRECT SUPPORT STRUCTURE GENERATION CODE FOR CAD-INTEGRATED COST ESTIMATOR
'
' Michael Barclift
' Andrew Armstrong
'
' Version 1.0 - Date: 9/29/2016 - Ray-Trace Projection - Height Method
' Version 2.0 - Date: 10/19/2016 - Ray-Trace Projection - Next Body
' Version 3.0 - Date: 1/28/2017 - Ray-Trace Normals - Internal Supports
' Version 4.0 - Date: 2/14/2017 - Temporary Bodies
'
'===============================================================================

    'Performance Tune-Up
    UserForm2.Hide
 '   DoEvents

    'Combination 1 Works
    swApp.UserControl = False
    swModel.Visible = False
    'swApp.Visible = False
    boolstatus = swModel.Extension.HideFeatureManager(True)
    swModelView.EnableGraphicsUpdate = False
    'swApp.Frame.KeepInvisible = True
    'swModel.Visible = False

    'Combination 2
    'swModelView.EnableGraphicsUpdate = False
    'boolstatus = swModel.Extension.HideFeatureManager(True)


If sangle = vbNullString Then
    MsgBox ("Define Support Angle")
    ElseIf sres = vbNullString Then
    MsgBox ("Define Support Resolution")
    Else

T = Timer

    sangle = Abs(180 - sangle)

    Set swApp = CreateObject("SldWorks.Application")
    Set swModel = swApp.ActiveDoc
    Set swSelMgr = swModel.SelectionManager
    Set swSelData = swSelMgr.CreateSelectData

    swModel.ClearSelection

    'Delete the Old Material
    If SpptCount > 0 Then
       deletesup
       End If

    'X_max = corners(3)
    'X_min = corners(0)
    'Y_max = corners(4)
    'Y_min = corners(1)
    'Z_max = corners(5)
    'Z_min = corners(2)

    X_max = bv_corners(3)
    X_min = bv_corners(0)
    Y_max = bv_corners(4)
    Y_min = bv_corners(1)
    Z_min = corners(2)

    num = 0

    'Set Radius on supports
    slice = sres
    Y_gap = (Y_max - Y_min) / slice
    Support_Rad_Y = Abs(Y_gap / 2)
    num_in_scan = 0

    swModel.ClearSelection2 True
    swBodySelect = swModel.GetBodies2(swSolidBody, False)
    swModel.ClearSelection2 True

    SpptCount = 0 'Restored in 1/14/2017
    SpptCount2 = 0:
    ReDim totalheight(0)
    ReDim totalheight2(0)

    'ZRay prevents ray trace from starting at the body edge
    Zray = Z_min - 0.00001 'gap between 2mm offset
    X_gap = (X_max - X_min) / slice
    Support_Rad_X = Abs(X_gap / 2)
    'Z direction update 01 July
```

```
        'Grid Definition for Support Radii
         'If Support_Rad_Y <= Support_Rad_X Then
            Support_Radius = Support_Rad_Y
            X_gap = Y_gap
            X_inc = X_min + Support_Rad_Y '- 2 / 2 / 2017
            Y_inc = Y_min + Support_Rad_Y '- 2 / 2 / 2017
         'Else
         '   Support_Radius = Support_Rad_X
         '   Y_gap = X_gap
         '   X_inc = X_min + Support_Rad_X
         '   Y_inc = Y_min + Support_Rad_X
         'End If

'*************************************************
'===============START OF LOOP===================
'*************************************************

        'Y_inc = Y_inc + Y_gap

'Go to Y where part exists to limit computing
While Y_inc <= corners(1)
Y_inc = Y_inc + Y_gap
Wend
Y_inc = Y_inc - Y_gap


While X_inc <= corners(0)
X_inc = X_inc + X_gap
Wend
X_min = X_inc - X_gap
X_inc = X_min

Do Until Y_inc > (Y_max * 0.99)

    Do Until X_inc > X_max * 0.99
    'If num = 0 Then
    'ElseIf num > 0 Then
        'Uniform Grid
        'Y_inc = Y_inc + Y_gap
        'Hexagonal Packed Grid
        'Y_inc = Y_inc + Y_gap * 0.86126809 + 0.0001
    'End If

'====================NEW CODE FOR POINTS INTERSECTIONS ======================
'===============3D MATLAB CODE - 10/14/2016 - Michael Barclift================
'--------------------------Ray Trace Projection Vectors----------------------------

    'Starts on current point
        shoot_center(0) = X_inc: shoot_center(1) = Y_inc: shoot_center(2) = Zray

    'Comparison to Directional Normal in Z
        Build_Direction(0) = 0: Build_Direction(1) = 0: Build_Direction(2) = 1#

'     'Visualize Grid
'     swModel.ClearSelection2 True
 '   bRet = swModel.Extension.SelectByID2("Grid_Points", "SKETCH", 0, 0, 0, False, 0, Nothing, 0)
 '    swModel.Insert3DSketch2 True
        'swModel.EditSketch
 '    Set swSketchPt(0) = swModel.CreatePoint2(X_inc, Y_inc, Z_min)
        'swModel.EditSketch False
    '    swModel.Insert3DSketch2 False

        oangle = 0
        hit_center = swModel.RayIntersections(swBodySelect, shoot_center, Build_Direction, swRayPtsOptsNORMALS, hitRadius, offset)
        'Debug.Print swModel.RayIntersections(swBodySelect, shoot_center, Build_Direction, swRayPtsOptsNORMALS, hitRadius, offset)
        center_vPts = swModel.GetRayIntersectionsPoints

    If hit_center > 0 Then

            'Debug.Print center_vPts(0), center_vPts(1), center_vPts(2), center_vPts(3), center_vPts(4), center_vPts(5), center_vPts(6), center_vPts(7), center_vPts(8)
            oangle = Arccos(center_vPts(8)) * 180 / 3.14159265359

            If oangle > sangle Then
              xc = center_vPts(3)
              yc = center_vPts(4)

              If SpptCount > 0 Then
              ReDim Preserve xc_array(SpptCount)
              ReDim Preserve yc_array(SpptCount)
              xc_array(SpptCount) = xc
              yc_array(SpptCount) = yc

              totalheight(UBound(totalheight)) = center_vPts(5) - (Z_min - 0.002) '2mm accounts for raft on build surface
              ReDim Preserve totalheight(UBound(totalheight) + 1)
              'Debug.print xc_array(0); "and"; xc_array(1)
              'Debug.print yc_array(0); "and"; yc_array(1)
              Else
              ReDim xc_array(SpptCount)
              ReDim yc_array(SpptCount)
              xc_array(SpptCount) = xc
              yc_array(SpptCount) = yc

              totalheight(UBound(totalheight)) = center_vPts(5) - (Z_min - 0.002)
              ReDim Preserve totalheight(UBound(totalheight) + 1)
              End If

              SpptCount = SpptCount + 1

            End If

            If hit_center > 2 Then
                    p = 2
                    While p < hit_center
                    oangle2 = Arccos(center_vPts(9 * p + 8)) * 180 / 3.14159265359

                    If oangle2 > sangle Then

                        xc = center_vPts(9 * p + 3)
                        yc = center_vPts(9 * p + 4)
                        zc = center_vPts(9 * p + 5)

                        If SpptCount2 > 0 Then
                        ReDim Preserve xc_array_up(SpptCount2)
                        ReDim Preserve yc_array_up(SpptCount2)
                        ReDim Preserve zc_array_up(SpptCount2)
                        ReDim Preserve zc_array_h(SpptCount2)
                        xc_array_up(SpptCount2) = xc
                        yc_array_up(SpptCount2) = yc
                        zc_array_up(SpptCount2) = center_vPts(9 * (p - 1) + 5)
                        zc_array_h(SpptCount2) = (zc - center_vPts(9 * (p - 1) + 5))

                        totalheight2(UBound(totalheight2)) = zc_array_h(SpptCount2)
                        ReDim Preserve totalheight2(UBound(totalheight2) + 1)

                        Else
                        ReDim xc_array_up(SpptCount2)
                        ReDim yc_array_up(SpptCount2)
                        ReDim zc_array_up(SpptCount2)
                        ReDim zc_array_h(SpptCount2)
                        xc_array_up(SpptCount2) = xc
                        yc_array_up(SpptCount2) = yc
```

```
                zc_array_up(SpptCount2) = center_vPts(9 * (p - 1) + 5)
                zc_array_h(SpptCount2) = (zc - center_vPts(9 * (p - 1) + 5))

                totalheight2(UBound(totalheight2)) = zc_array_h(SpptCount2)
                ReDim Preserve totalheight2(UBound(totalheight2) + 1)
                End If

                SpptCount2 = SpptCount2 + 1

                End If

              p = p + 2
              Wend
          End If


          End If

        num_in_scan = num_in_scan + 1

       'Z-Direction Update 01 July
        'If num_in_scan = 0 Then 'And (num Mod 2) = 1 Then
          'X_inc = X_min + X_gap
        'ElseIf num_in_scan = 0 And (num Mod 2) = 0 Then
          'X_inc = X_min + X_gap / 2
        'ElseIf num_in_scan > 0 Then
        If num_in_scan > 0 Then
          X_inc = X_inc + X_gap
        End If

        If X_inc > corners(3) Then
        X_inc = X_max
        End If

      Loop

      'swPlane.Show = True
      'Debug.Assert Not swPlane Is Nothing

      num = num + 1
      num_in_scan = 0
      X_inc = X_min

        'Uniform Grid
        Y_inc = (Y_inc + Y_gap)
        'Hexagonal Packed Grid
```

```
    'Y_inc = Y_inc + Y_gap * 0.86126809 + 0.0001

    If Y_inc > corners(4) Then
    Y_inc = (Y_max + 1) 'Added +1 to help termination criteria be more decisive
    End If


Loop

        If virtualsup.Value Then
        '=================================
        'Calculate Analytical Support Volume
        '==================================
        itr = 0
        calc_sv = 0
        supportsurfaces = UBound(totalheight) + UBound(totalheight2)

        'External Supports
        'While itr < UBound(totalheight)
        'calc_sv = calc_sv + 3.14159265359 * Support_Radius * Support_Radius *
totalheight(itr) * 0.9801 'due to 0.99 for clearancing
        'itr = itr + 1
        'Wend

        'Internal Supports
        'itr = 0
        'While itr < UBound(totalheight2)
        'calc_sv = calc_sv + 3.14159265359 * Support_Radius * Support_Radius *
totalheight2(itr) * 0.9801
        'itr = itr + 1
        'Wend


        eos_svol.Value = Format(calc_sv * 100 * 100 * 100, "0.00")
        Erase totalheight
        Erase totalheight2
        Erase xc_array_up
        Erase yc_array_up
        Erase zc_array_up
        Erase zc_array_h
        Erase xc_array
        Erase yc_array

        'If Not optim And eos_slice.Value <> vbNullString Then
        If eos_slice.Value <> vbNullString Then
        'Build Time
        tempvar(0) = eos_rr.Value
        tempvar(1) = eos_slice.Value
        tempvar(2) = eos_br.Value
        tempvar(3) = eos_svol.Value
        tempvar(4) = eos_compv.Value
        tempvar(5) = eos_mw.Value / 100
        eos_bt.Value = (tempvar(0) * tempvar(1)) * (1 / 60 * 1 / 60) + 1 / tempvar(2) *
(tempvar(3) + tempvar(4))
        eos_mc.Value = eos_bt * eos_mr.Value

        'Component Mass
        eos_smass.Value = (1 + tempvar(5)) * tempvar(3) * eos_wd.Value / 1000
        eos_pe.Value = tempvar(4) * (1 + tempvar(5)) * eos_wd.Value * eos_mp.Value / 1000
        eos_scost.Value = (CDbl(eos_smass)) * CDbl(eos_mp.Value)

        '85 is powder depreciation for 15 use Ti64 powder
        'eos_pdep.Value = 85 * (eos_pfk - eos_smass - (1 + tempvar(5)) * tempvar(4) / 1000)
        'eos_pdep.Value = 0

        tempvar(6) = (CDbl(eos_pe.Value) + CDbl(eos_scost.Value) + CDbl(eos_mc.Value)) '+
CDbl(eos_pdep))

        totalcost.Value = tempvar(6)

        totalcost.Value = Format(totalcost.Value, "0")
        eos_bt.Value = Format(eos_bt.Value, "0")
        eos_mc.Value = Format(eos_mc.Value, "0")
        eos_pe.Value = Format(eos_pe.Value, "0")
        eos_scost.Value = Format(eos_scost.Value, "0")
        eos_smass.Value = Format(eos_smass.Value, "0.00")
        End If

        Else
        '===============================
        'Create Circle as base for support
        '===============================
        itr = 0

        swModel.ClearSelection2 (True)
        Set swSketchMgr = swModel.SketchManager
        swSketchMgr.AddToDB = True
        swSketchMgr.DisplayWhenAdded = False
        swApp.CommandInProgress = True

      '========================
      'Loop for internal supports
      '=========================
      If SpptCount2 > 0 Then

      itr = 0

      If Hollow.Value Then


      Else

      Do Until itr > (SpptCount2 - 1)

              dblData(0) = xc_array_up(itr)
              dblData(1) = yc_array_up(itr)
              dblData(2) = zc_array_up(itr)
              dblData(3) = 0
              dblData(4) = 0
              dblData(5) = 1
              dblData(6) = Support_Radius * 0.99
              dblData(7) = zc_array_h(itr)
              dblData(8) = zc_array_h(itr)

              'Set swBody = swModeler.CreateBodyFromBox(dblData)
              Set swBody = swModeler.CreateBodyFromCyl(dblData)
              Set support = swModel.CreateFeatureFromBody3(swBody, False, 0)

              color =
support.GetMaterialPropertyValues2(swInConfigurationOpts_e.swThisConfiguration, "")
              color = support.GetMaterialPropertyValues2(1, Empty)

              color(0) = 0 '249 / 256
              color(1) = 1 '66 / 256
              color(2) = 1 '58 / 256
              color(3) = 1
              color(4) = 1
              color(5) = 0.8
              color(6) = 0.3215
              color(7) = 0
              color(8) = 0

              support.SetMaterialPropertyValues2 color, swAllConfiguration, Empty


      swModel.SelectionManager.EnableContourSelection = False
      swModel.ClearSelection
```

```
        support.Name = "Internal_Support_" + CStr(itr)

        itr = itr + 1
        Loop

        End If

        Erase xc_array_up
        Erase yc_array_up
        Erase zc_array_up
        Erase zc_array_h

        itr = 0
        boolstatus = swModel.Extension.SelectByID2("Internal_Support_" + CStr(itr),
"BODYFEATURE", 0, 0, 0, False, 4, Nothing, 0)

        Do Until itr > (SpptCount2 - 1)
        itr = itr + 1
        boolstatus = swModel.Extension.SelectByID2("Internal_Support_" + CStr(itr),
"BODYFEATURE", 0, 0, 0, True, 4, Nothing, 0)
        Loop

        Set myFeature =
swModel.FeatureManager.InsertFeatureTreeFolder2(swFeatureTreeFolderType_e.swFeatureTre
eFolder_Containing)
        boolstatus = swModel.SelectedFeatureProperties(0, 0, 0, 0, 0, 0, 0, 1, 0, "Internal Support
Structures")
     End If

        swModel.ClearSelection


          If SpptCount > 0 Then
                itr = 0
              Do Until itr > (SpptCount - 1)

                   dblData(0) = xc_array(itr)
                   dblData(1) = yc_array(itr)
                   dblData(2) = (Z_min - 0.002)
                   dblData(3) = 0
                   dblData(4) = 0
                   dblData(5) = 1
                   dblData(6) = Support_Radius * 0.99
                   dblData(7) = totalheight(itr)
                   dblData(8) = totalheight(itr)
                   'Set swBody = swModeler.CreateBodyFromBox(dblData)
                   Set swBody = swModeler.CreateBodyFromCyl(dblData)
                   Set support = swModel.CreateFeatureFromBody3(swBody, False, 0)

                   color =
support.GetMaterialPropertyValues2(swInConfigurationOpts_e.swThisConfiguration, "")
                   color = support.GetMaterialPropertyValues2(1, Empty)
                   color(0) = 1
                   color(1) = 0
                   color(2) = 1
                   color(3) = 1
                   color(4) = 1
                   color(5) = 0.8
                   color(6) = 0.3215
                   color(7) = 0
                   color(8) = 0

                   support.SetMaterialPropertyValues2 color, swAllConfiguration, Empty

                   swModel.SelectionManager.EnableContourSelection = False
                   swModel.ClearSelection

                   support.Name = "External_Support_" + CStr(itr)

                itr = itr + 1
                Loop


                   Erase xc_array
                   Erase yc_array
                   Erase totalheight

                   itr = 0
                   boolstatus = swModel.Extension.SelectByID2("External_Support_" + CStr(itr),
"BODYFEATURE", 0, 0, 0, False, 4, Nothing, 0)

                   Do Until itr > (SpptCount - 1)
                   itr = itr + 1
                   boolstatus = swModel.Extension.SelectByID2("External_Support_" + CStr(itr),
"BODYFEATURE", 0, 0, 0, True, 4, Nothing, 0)
                   Loop

                   Set myFeature =
swModel.FeatureManager.InsertFeatureTreeFolder2(swFeatureTreeFolderType_e.swFeatureTre
eFolder_Containing)
```

```
                     boolstatus = swModel.SelectedFeatureProperties(0, 0, 0, 0, 0, 0, 0, 1, 0, "External Support Structures")

            End If

            Min = Format((Timer - T), "0.00")

            MsgBox "Code ran in " & Min & " seconds", vbInformation

            swApp.CommandInProgress = False
            swSketchMgr.DisplayWhenAdded = True
            swSketchMgr.AddToDB = False

   swModel.ForceRebuild3 (True)

   Set swSupports = swModel.Extension.CreateMassProperty

   support_volume_raw = swSupports.Volume
   support_volume_actual = (support_volume_raw * 100 * 100 * 100)
   support_volume_actual = support_volume_actual - partvolume
   support_volume = support_volume_actual - 1587.5 '1615.91 volume of substrate
   support_volume = Format(support_volume, "0.00")
   eos_svol.Value = support_volume

            If eos_slice <> vbNullString Then

            'Build Time
            tempvar(0) = eos_rr.Value
            tempvar(1) = eos_slice.Value
            tempvar(2) = eos_br.Value
            tempvar(3) = eos_svol.Value
            tempvar(4) = eos_compv.Value
            tempvar(5) = eos_mw.Value / 100
            eos_bt.Value = (tempvar(0) * tempvar(1)) * (1 / 60 * 1 / 60) + 1 / tempvar(2) * (tempvar(3) + tempvar(4))
            eos_mc.Value = eos_bt * CDbl(eos_mr.Value)

            'Component Mass
            eos_smass.Value = (1 + tempvar(5)) * tempvar(3) * eos_wd.Value / 1000
            eos_pe.Value = tempvar(4) * (1 + tempvar(5)) * eos_wd.Value * eos_mp.Value / 1000
            eos_scost.Value = (CDbl(eos_smass)) * CDbl(eos_mp.Value)

            '85 is powder depreciation for 15 use Ti64 powder
            'eos_pdep.Value = 85 * (eos_pfk - eos_smass - (1 + tempvar(5)) * tempvar(4) / 1000)
            'eos_pdep.Value = 0
            tempvar(6) = (CDbl(eos_pe.Value) + CDbl(eos_scost.Value) + CDbl(eos_mc.Value)) '+
CDbl(eos_pdep))

            totalcost.Value = tempvar(6)

            totalcost.Value = Format(totalcost.Value, "0")
            eos_bt.Value = Format(eos_bt.Value, "0")
            eos_mc.Value = Format(eos_mc.Value, "0")
            eos_pe.Value = Format(eos_pe.Value, "0")
            eos_scost.Value = Format(eos_scost.Value, "0")
            eos_smass.Value = Format(eos_smass.Value, "0.00")
            End If

End If
End If

'Performance Tune-Up

   'swApp.UserControl = True
   swModel.Visible = True
   'swApp.Visible = True
   boolstatus = swModel.Extension.HideFeatureManager(False)
   swModelView.EnableGraphicsUpdate = True
   'swApp.Frame.KeepInvisible = False

   If Not optim Then
   UserForm2.Show vbModeless
   End If

End Function

Function deleteall()

boolstatus = swModel.Extension.SelectByID2("Build Volume", "SKETCH", 0, 0, 0, False, 0,
Nothing, 0)
boolstatus = swModel.Extension.SelectByID2("Build Surface", "PLANE", 0, 0, 0, True, 0, Nothing,
0)
boolstatus = swModel.Extension.SelectByID2("Build Platform", "BODYFEATURE", 0, 0, 0, True, 0,
Nothing, 0)
boolstatus = swModel.Extension.SelectByID2("Internal Support Structures", "FTRFOLDER", 0, 0,
0, True, 0, Nothing, 0)
boolstatus = swModel.Extension.SelectByID2("Support Structures", "BODYFEATURE", 0, 0, 0,
True, 0, Nothing, 0)
swModel.EditDelete
swModel.ClearSelection
```

```
deletesup

itr = 0
boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
 'boolstatus = swModel.Extension.SelectByID2("Inner_", "SKETCH", 0, 0, 0, False, 4, Nothing, 0)

While boolstatus
swModel.EditDelete
itr = itr + 1
boolstatus = swModel.Extension.SelectByID2("Inner_" + CStr(itr), "SKETCH", 0, 0, 0, False, 4,
Nothing, 0)
Wend

End Function

Private Sub CommandButton3_Click()

   Set swApp = Application.SldWorks
   Set swModel = swApp.ActiveDoc
   Set swSelMgr = swModel.SelectionManager
   Set swSelData = swSelMgr.CreateSelectData
   num = 0
   slice = TAP.Value
   'layert = 0.02 ----- SolidWorks is working in meters as default therefore this is 20 mm
   'layert = 0.001
   layert = (zmax / 1000) / slice

   Do Until num = slice
   If num = 0 Then
   ElseIf num > 0 Then
   corners(1) = corners(1) + layert
   End If

   swModel.Insert3DSketch2 True
   swModel.ClearSelection2 True

   Set swSketchPt(0) = swModel.CreatePoint2(corners(3), corners(1), corners(5))
   Set swSketchPt(1) = swModel.CreatePoint2(corners(0), corners(1), corners(5))
   Set swSketchPt(2) = swModel.CreatePoint2(corners(0), corners(1), corners(2))

   swModel.Insert3DSketch2 False

   bRet = swSketchPt(0).Select4(True, swSelData): Debug.Assert bRet
   bRet = swSketchPt(1).Select4(True, swSelData): Debug.Assert bRet
   bRet = swSketchPt(2).Select4(True, swSelData): Debug.Assert bRet
   Set swPlane = swModel.CreatePlaneThru3Points3(True)

   swPlane.Name = "AM_Slice_Plane_" & CStr(num)

   'swPlane.Show = True
   'Debug.Assert Not swPlane Is Nothing

   num = num + 1
   Loop

'swModel.Insert3DSketch2 True
'swModel.Rebuild (1)
'hello
'swModel.Insert3DSketch2 True
'swModel.Rebuild (2)

swModel.ForceRebuild3 (True)

End Sub

Private Sub CommandButton4_Click()

boolstatus = rotate_part(xrot.Value, yrot.Value, zrot.Value)

If eos_slice.Value <> vbNullString Then
eos_pfk.Value = eos_td.Value * (eos_compz.Value / 10) * (eos_bpx.Value / 10) * (eos_bpy.Value
/ 10) / 1000 * eos_ca.Value / 100
End If

eos_svol.Value = vbNullString

End Sub

Function rand(ubd, lbd) As Integer
Randomize
rand = Int((ubd - lbd + 1) * Rnd + lbd)
End Function
Function rotate_part_fast(Xangle, Yangle, Zangle)

Dim swApp As Object
Dim swModel As Object
Dim swFeatureManager As Object
Dim swModelDocExt As Object
Dim status As Boolean
Dim boolstatus As Boolean
Dim buildorientation As Object
Dim vBodyArr As Variant
Dim swBody As SldWorks.Body2

'Sub main()
Set swApp = Application.SldWorks
Set swModel = swApp.ActiveDoc
Set swFeatureManager = swModel.FeatureManager
Set swModelDocExt = swModel.Extension

   'Performance Tune-Up
   swApp.UserControl = False
   swApp.Visible = False
   boolstatus = swModel.Extension.HideFeatureManager(True)
   'swApp.DocumentVisible False, swDocumentTypes_e.swDocPART 'Open files in memory
   'swModel.Visible = False 'Loads part into memory
   swModelView.EnableGraphicsUpdate = False

   'Delete Old Support Structures, Bounding Box, Build Surface, Build Platform
   'deletesup
   deleteall

   status = swModelDocExt.SelectByID2("AM_Rotated_Orientation_" & CStr(numnum),
"BODYFEATURE", 0, 0, 0, True, 0, Nothing, 0)
   If status = True Then
   swModel.EditDelete
   End If

   If numnum > 0 Then
   status = swModelDocExt.SelectByID2("AM_Rotated_Orientation_" & CStr(numnum - 1),
"BODYFEATURE", 0, 0, 0, True, 0, Nothing, 0)
   If status = True Then
   swModel.EditDelete
   End If
   End If

   numnum = numnum + 1

Body = swModel.GetBodies2(swSolidBody, True)

status = swModelDocExt.SelectByID2(Body(0).Name, "SOLIDBODY", 0, 0, 0, True, 0, Nothing, 0)

swModel.ClearSelection2 True

status = swModelDocExt.SelectByID2(Body(0).Name, "SOLIDBODY", 0, 0, 0, False, 1, Nothing, 0)
```

```
If Xangle = vbNullString Then
xrot.Value = 0
End If

If Yangle = vbNullString Then
yrot.Value = 0
End If

If Zangle = vbNullString Then
zrot.Value = 0
End If

xrotv = Xangle / 180 * 3.14159265359
yrotv = Yangle / 180 * 3.14159265359
zrotv = Zangle / 180 * 3.14159265359

Set buildorientation = swFeatureManager.InsertMoveCopyBody2(0, 0, 0, 0, 0, 0, 0, zrotv, yrotv,
xrotv, False, 1)

swModel.ClearSelection2 True

buildorientation.Name = "AM_Rotated_Orientation_" & CStr(numnum)

swModel.ClearSelection2 True

swModel.ForceRebuild3 (True)

   swModel.SetDisplayWhenAdded False
   BoundingBoxCode
   swModel.SetDisplayWhenAdded True

DataInitialize

If eos_slice.Value <> vbNullString Then
eos_pfk.Value = eos_td.Value * (eos_compz.Value / 10) * (eos_bpx.Value / 10) * (eos_bpy.Value
/ 10) / 1000 * eos_ca.Value / 100
End If

   'Performance Tune-Up
   swApp.UserControl = True
   swApp.Visible = True
   boolstatus = swModel.Extension.HideFeatureManager(False)
   'swApp.DocumentVisible True, swDocumentTypes_e.swDocPART
   swModelView.EnableGraphicsUpdate = True

End Function
```

```
Function rotate_part(Xangle, Yangle, Zangle)

Dim swApp As Object
Dim swModel As Object
Dim swFeatureManager As Object
Dim swModelDocExt As Object
Dim status As Boolean
Dim boolstatus As Boolean
Dim buildorientation As Object
Dim vBodyArr As Variant
Dim swBody As SldWorks.Body2

'Sub main()
Set swApp = Application.SldWorks
Set swModel = swApp.ActiveDoc
Set swFeatureManager = swModel.FeatureManager
Set swModelDocExt = swModel.Extension

    'Performance Tune-Up
    swApp.UserControl = False
    swApp.Visible = False
   boolstatus = swModel.Extension.HideFeatureManager(True)
    'swApp.DocumentVisible False, swDocumentTypes_e.swDocPART 'Open files in memory
    'swModel.Visible = False 'Loads part into memory
    swModelView.EnableGraphicsUpdate = False

    'Delete Old Support Structures
    deletesup
    deleteall

    status = swModelDocExt.SelectByID2("AM_Rotated_Orientation_" & CStr(numnum),
"BODYFEATURE", 0, 0, 0, True, 0, Nothing, 0)
    If status = True Then
    swModel.EditDelete
    End If

    If numnum > 0 Then
    status = swModelDocExt.SelectByID2("AM_Rotated_Orientation_" & CStr(numnum - 1),
"BODYFEATURE", 0, 0, 0, True, 0, Nothing, 0)
    If status = True Then
    swModel.EditDelete
    End If
    End If
```

```
Body = swModel.GetBodies2(swSolidBody, True)

status = swModelDocExt.SelectByID2(Body(0).Name, "SOLIDBODY", 0, 0, 0, True, 0, Nothing, 0)

swModel.ClearSelection2 True

status = swModelDocExt.SelectByID2(Body(0).Name, "SOLIDBODY", 0, 0, 0, False, 1, Nothing, 0)

   If Xangle = vbNullString Then
   xrot.Value = 0
   Xangle = 0
   End If

   If Yangle = vbNullString Then
   yrot.Value = 0
   Yangle = 0
   End If

   If Zangle = vbNullString Then
   zrot.Value = 0
   Zangle = 0
   End If

xrotv = Xangle / 180 * 3.14159265359
yrotv = Yangle / 180 * 3.14159265359
zrotv = Zangle / 180 * 3.14159265359

Set buildorientation = swFeatureManager.InsertMoveCopyBody2(0, 0, 0, 0, 0, 0, 0, zrotv, yrotv,
xrotv, False, 1)

swModel.ClearSelection2 True

buildorientation.Name = "AM_Rotated_Orientation_" & CStr(numnum)

swModel.ClearSelection2 True

numnum = numnum + 1

swModel.ForceRebuild3 (True)

ModelInitialize
DataInitialize

   'Performance Tune-Up
   swApp.UserControl = True
   swApp.Visible = True
```

```
    boolstatus = swModel.Extension.HideFeatureManager(False)
    'swApp.DocumentVisible True, swDocumentTypes_e.swDocPART
    swModelView.EnableGraphicsUpdate = True

End Function

Private Sub CommandButton5_Click()
'============================
'Build Orientation Optimization
'============================

    'Performance Speed-Up
    'swModel.Visible = False

    optim = True
    virtualsup.Value = True

'Minimize: Build Height, Support Volume, Build Area, Bounding Box Volume
Dim obj_bh() 'Build Height
Dim obj_sv() 'Support Volume

ReDim obj_sv(1)
ReDim obj_bh(1)

'Subject to constraints:
'1) Part bounding box must be inside print volume
'2) Angles for X,Y,Z rotations must be between 0 and 360 degrees

Dim rotx_ul As Double 'Rotation along x - Upper Limit
Dim roty_ul As Double 'Rotation along y - Upper Limit
Dim rotz_ul As Double 'Rotation along z - Upper Limit
Dim rotx_ll As Double 'Rotation along x - Lower Limit
Dim roty_ll As Double 'Rotation along y - Lower Limit
Dim rotz_ll As Double 'Rotation along z - Lower Limit

Dim pop_size As Integer 'Samples for each iterations
Dim max_gen As Integer 'Maximum number of iterations
Dim j As Integer 'Index
Dim popx() 'Population of x angles array
Dim popy() 'Population of y angles array
Dim popz() 'Population of z angles array
Dim rpopx() 'Regenerated Population of x angles array
Dim rpopy() 'Regenerated Population of y angles array
Dim rpopz() 'Regenerated Population of z angles array
Dim mutants As Integer 'Number of mutants per population
Dim m1 As Double 'Mutation Factor 1
Dim m2 As Double 'Mutation Factor 2
Dim popmean As Double 'Population Mean

Dim converged As Boolean
Dim tol As Double
Dim tolv As Double

ReDim popx(1)
ReDim popy(1)
ReDim popz(1)

'=============
'Pre-Processing
'=============
'Rotation Constraints
rotx_ll = -180
roty_ll = -180
rotz_ll = -180
rotx_ul = 180
roty_ul = 180
rotz_ul = 180

'********************
'Optimization Settings
'********************
pop_size = 15 'Starts at zero base
max_gen = 40 'Starts at zero base

'Convergence Criteria
tol = 5 'Tolerance on volume difference
tolv = 200 'Tolerance on average population

'Excel Worksheet Set-Up and Connection
Dim exApp As Excel.Application
Dim sheet As Excel.Worksheet

  Set exApp = CreateObject("Excel.Application")
  exApp.Visible = True
  exApp.Workbooks.Add
  Set sheet = exApp.ActiveSheet

  sheet.Cells(1, 1).Value = "Iteration"
  sheet.Cells(1, 2).Value = "Angle X"
  sheet.Cells(1, 3).Value = "Angle Y"
  sheet.Cells(1, 4).Value = "Angle Z"
  sheet.Cells(1, 5).Value = "Support Volume"
```

```
    sheet.Cells(1, 6).Value = "Build Height"
    sheet.Cells(1, 7).Value = "Top Two"
    sheet.Cells(1, 8).Value = "Population Mean"

'=================
'Particle Swarm Optimization Algorithm
'=================
Dim feasible_rotx(): ReDim feasible_rotx(1)
Dim feasible_roty(): ReDim feasible_roty(1)
Dim feasible_rotz(): ReDim feasible_rotz(1)

Dim gen_max_sv(): ReDim gen_max_sv(1)
Dim gen_max_bh(1)
Dim gen_max_valx(): ReDim gen_max_valx(1)
Dim gen_max_valy(): ReDim gen_max_valy(1)
Dim gen_max_valz(): ReDim gen_max_valz(1)

Dim k1 As Integer
Dim k2 As Integer
Dim n As Integer: n = 0

Dim deltavx(): ReDim deltavx(pop_size)
Dim old_deltavx(): ReDim old_deltavx(pop_size)
Dim deltavy(): ReDim deltavy(pop_size)
Dim old_deltavy(): ReDim old_deltavy(pop_size)
Dim deltavz(): ReDim deltavz(pop_size)
Dim old_deltavz(): ReDim old_deltavz(pop_size)

Dim childx(0)
Dim childy(0)
Dim childz(0)
Dim bit As Integer
Dim md As Double 'Randomly Selected Mutation Factor

Dim Pbest(): ReDim Pbest(pop_size)
Dim Pbest_x(): ReDim Pbest_x(pop_size)
Dim Pbest_y(): ReDim Pbest_y(pop_size)
Dim Pbest_z(): ReDim Pbest_z(pop_size)

Dim w1 As Double: w1 = 0.5
Dim c1 As Double: c1 = 1.5
Dim c2 As Double: c2 = 1.5

'swModel.Visible = False

'Generate Random Population
```

```
For j = 0 To pop_size
popx(j) = rand(rotx_ll, rotx_ul)
popy(j) = rand(roty_ll, roty_ul)
popz(j) = rand(rotz_ll, rotz_ul)

ReDim Preserve popx(UBound(popx) + 1)
ReDim Preserve popy(UBound(popy) + 1)
ReDim Preserve popz(UBound(popz) + 1)
Next j

'Calculate Fitness and Feasibility of Population
While n < max_gen And converged = False

    For j = 0 To pop_size

      boolstatus = rotate_part_fast(popx(j), popy(j), popz(j))

    If Abs(corners(0)) > Abs(bv_corners(0)) Or Abs(corners(1)) > Abs(bv_corners(1)) Or
Abs(corners(2)) > Abs(bv_corners(2)) Or Abs(corners(3)) > Abs(bv_corners(3)) Or Abs(corners(4))
> Abs(bv_corners(4)) Or Abs(corners(5)) > Abs(bv_corners(5)) Then

      sheet.Cells(j + 2 + n * (pop_size + 1), 1).Value = n
      sheet.Cells(j + 2 + n * (pop_size + 1), 2).Value = "NOT FEASIBLE"

      Else

      boolstatus = generate_supports(eos_sangle.Value, eos_sres.Value) 'Input support angle
and resolution of supports

      feasible_rotx(UBound(feasible_rotx) - 1) = popx(j)
      feasible_roty(UBound(feasible_roty) - 1) = popy(j)
      feasible_rotz(UBound(feasible_rotz) - 1) = popz(j)
      obj_sv(UBound(obj_sv) - 1) = CDbl(supportsurfaces) 'Objective Function
      obj_bh(UBound(obj_bh) - 1) = CDbl(eos_compz.Value)

      sheet.Cells(j + 2 + n * (pop_size + 1), 1).Value = n
      sheet.Cells(j + 2 + n * (pop_size + 1), 2).Value = popx(j)
      sheet.Cells(j + 2 + n * (pop_size + 1), 3).Value = popy(j)
      sheet.Cells(j + 2 + n * (pop_size + 1), 4).Value = popz(j)
      sheet.Cells(j + 2 + n * (pop_size + 1), 5).Value = CDbl(supportsurfaces) 'Objective Function
      sheet.Cells(j + 2 + n * (pop_size + 1), 6).Value = CDbl(eos_compz.Value)

      ReDim Preserve feasible_rotx(UBound(feasible_rotx) + 1)
      ReDim Preserve feasible_roty(UBound(feasible_roty) + 1)
      ReDim Preserve feasible_rotz(UBound(feasible_rotz) + 1)
      ReDim Preserve obj_sv(UBound(obj_sv) + 1)
```

```
   ReDim Preserve obj_bh(UBound(obj_bh) + 1)

End If

Next j

'===Replace old personal bests===
If n = 0 Then
   ReDim Pbest(UBound(obj_sv))
   Pbest = obj_sv
   Pbest_x = popx
   Pbest_y = popy
   Pbest_z = popz

Else
   k1 = 0
   While k1 < (UBound(obj_sv) - 1)
      If obj_sv(k1) < Pbest(k1) Then
      Pbest(k1) = obj_sv(k1)
      Pbest_x(k1) = popx(k1)
      Pbest_y(k1) = popy(k1)
      Pbest_z(k1) = popz(k1)
      End If
      k1 = k1 + 1
   Wend
End If

'===Find global best===
k1 = 0
k2 = 1
While k2 < (UBound(obj_sv) - 1)
   If obj_sv(k1) > obj_sv(k2) Then
   k1 = k2
   End If
   k2 = k2 + 1
Wend
sheet.Cells(k1 + 2 + n * (pop_size + 1), 7).Value = "1"

'===Store global best===
If n = 0 Then
gen_max_sv(n) = obj_sv(k1)
gen_max_valx(n) = feasible_rotx(k1)
gen_max_valy(n) = feasible_roty(k1)
gen_max_valz(n) = feasible_rotz(k1)

ElseIf n > 0 And gen_max_sv(n - 1) > obj_sv(k1) Then
```

```
gen_max_sv(n) = obj_sv(k1)
gen_max_valx(n) = feasible_rotx(k1)
gen_max_valy(n) = feasible_roty(k1)
gen_max_valz(n) = feasible_rotz(k1)

ElseIf n > 0 Then
gen_max_sv(n) = gen_max_sv(n - 1)
gen_max_valx(n) = gen_max_valx(n - 1)
gen_max_valy(n) = gen_max_valy(n - 1)
gen_max_valz(n) = gen_max_valz(n - 1)
End If

ReDim Preserve gen_max_sv(UBound(gen_max_sv) + 1)
ReDim Preserve gen_max_valx(UBound(gen_max_valx) + 1)
ReDim Preserve gen_max_valy(UBound(gen_max_valy) + 1)
ReDim Preserve gen_max_valz(UBound(gen_max_valz) + 1)
sheet.Cells(j + 2 + n * (pop_size + 1), 9).Value = gen_max_sv(n)

'===Population Mean Calculation===
popmean = 0
k = 0
Do Until k > UBound(obj_sv)
popmean = popmean + obj_sv(k)
k = k + 1
Loop
popmean = popmean / (UBound(obj_sv) + 1)
sheet.Cells(j + 2 + n * (pop_size + 1), 8).Value = popmean

'================
'Post-Processing
'================

'Check for convergence

If n > 20 Then
   tol = gen_max_sv(n) - gen_max_sv(n - 10)
   converged = (tol <= 1)

   If converged = True Then
   popx(0) = gen_max_valx(n)
   popy(0) = gen_max_valy(n)
   popz(0) = gen_max_valz(n)
   MsgBox ("Solution Converged")
   End If

Else
```

```
    For j = 0 To pop_size
    deltavx(j) = w1 * old_deltavx(j) + c1 * (CDbl(rand(0, 1000)) / 1000) * (Pbest_x(j) - popx(j)) +
c2 * (CDbl(rand(0, 1000)) / 1000) * (gen_max_valx(n) - popx(j))
    deltavy(j) = w1 * old_deltavy(j) + c1 * (CDbl(rand(0, 1000)) / 1000) * (Pbest_y(j) - popy(j)) +
c2 * (CDbl(rand(0, 1000)) / 1000) * (gen_max_valy(n) - popy(j))
    deltavz(j) = w1 * old_deltavz(j) + c1 * (CDbl(rand(0, 1000)) / 1000) * (Pbest_z(j) - popz(j)) +
c2 * (CDbl(rand(0, 1000)) / 1000) * (gen_max_valz(n) - popz(j))

      'Check if X angle is feasible
      popx(j) = popx(j) + deltavx(j)
      If (popx(j) < rotx_ll) Then
        popx(j) = rotx_ll
        deltavx(j) = 0
      ElseIf (popx(j) > rotx_ul) Then
        popx(j) = rotx_ul
        deltavx(j) = 0
      End If

      'Check if Y angle is feasible
      popy(j) = popy(j) + deltavy(j)
      If (popy(j) < roty_ll) Then
        popy(j) = roty_ll
        deltavy(j) = 0
      ElseIf (popy(j) > roty_ul) Then
        popy(j) = roty_ul
        deltavy(j) = 0
      End If

      'Check if Z angle is feasible
      popz(j) = popz(j) + deltavz(j)
      If (popz(j) < rotz_ll) Then
        popz(j) = rotz_ll
        deltavz(j) = 0
      ElseIf (popz(j) > rotz_ul) Then
        popz(j) = rotz_ul
        deltavz(j) = 0
      End If

    'Reassign old velocities
    old_deltavx(j) = deltavx(j)
    old_deltavy(j) = deltavy(j)
    old_deltavz(j) = deltavz(j)

    Next j
```

```
        ReDim feasible_rotx(1)
        ReDim feasible_roty(1)
        ReDim feasible_rotz(1)
        ReDim obj_sv(1)
        ReDim obj_bh(1)

        End If
        n = n + 1
Wend

    If n = max_gen Then
        popx(0) = gen_max_valx(n)
        popy(0) = gen_max_valy(n)
        popz(0) = gen_max_valz(n)
    End If

'====================
'Optimization Complete
'=====================
boolstatus = rotate_part(popx(0), popy(0), popz(0))
boolstatus = generate_supports(eos_sangle.Value, eos_sres.Value)
optim = False

xrot.Value = Int(popx(0))
yrot.Value = Int(popy(0))
zrot.Value = Int(popz(0))

'Deallocate Memory
Erase feasible_rotx()
Erase feasible_roty()
Erase feasible_rotz()
Erase obj_sv()
Erase obj_bh()
Erase popx()
Erase popy()
Erase popz()
Erase old_deltavx()
Erase old_deltavy()
Erase old_deltavz()
Erase deltavx
Erase deltavy
Erase deltavz

'swModel.Visible = True

UserForm2.Show
```

```
End Sub
Private Sub CommandButton6_Click()

Dim swApp As Object
Dim swModel As Object
Dim swFeatureManager As Object
Dim swModelDocExt As Object
Dim status As Boolean
Dim buildorientation As Object
Dim vBodyArr As Variant

'Sub main()
Set swApp = Application.SldWorks
Set swModel = swApp.ActiveDoc
Set swFeatureManager = swModel.FeatureManager
Set swModelDocExt = swModel.Extension

If numnum = 0 Then

MsgBox ("Model is already at original state.")

Else
status = swModelDocExt.SelectByID2("AM_Rotated_Orientation_" & CStr(numnum - 1),
"BODYFEATURE", 0, 0, 0, True, 0, Nothing, 0)
If status = True Then
swModel.EditDelete
End If

deleteall

numnum = numnum - 1

ModelInitialize
DataInitialize
End If

End Sub

Private Sub eos_br_Change()

If eos_mr.Value = vbNullString Then

ElseIf eos_bt.Value = vbNullString Then
Else
eos_mc = eos_mr.Value * eos_bt.Value
```

```
End If

If eos_rr.Value = vbNullString Then
Else
eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value)
eos_bt.Value = Format(eos_bt.Value, "0")
End If

End Sub

Private Sub eos_bt_Change()
If eos_mr.Value = vbNullString Then
ElseIf eos_bt.Value = vbNullString Then
Else
eos_mc = eos_mr.Value * eos_bt.Value
End If
End Sub

Private Sub eos_ca_Change()
If eos_ca.Value = vbNullString Then
Else

eos_pfk.Value = eos_td.Value * (eos_compz.Value / 10) * (eos_bpx.Value / 10) * (eos_bpy.Value
/ 10) / 1000 * (eos_ca.Value / 100)
eos_pfk.Value = Format(eos_pfk.Value, "0")
eos_pfb.Value = eos_mp.Value * eos_pfk.Value
End If

'10/25/2016 - Addition
If eos_pe.Value = vbNullString Then

totalcost.Value = vbNullString
'ElseIf eos_pw.Value = vbNullString Then
totalcost.Value = vbNullString
ElseIf eos_mc.Value = vbNullString Then
totalcost.Value = vbNullString

Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")
```

```
End If

End Sub

Private Sub eos_compz_Change()

If eos_lt.Value = vbNullString Then
Else

eos_slice.Value = (zmax * 1000) / eos_lt.Value
eos_slice.Value = Format(eos_slice.Value, "0")
End If

If eos_svol.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_compmass.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_slice.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_rr.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_br.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_mp.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_mw.Value = vbNullString Then
'eos_pe.value = vbNullString

eos_pe.Value = (eos_compmass.Value) * eos_mp.Value
eos_pe.Value = Format(eos_pe.Value, "0")

eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value + eos_svol.Value)
eos_bt.Value = Format(eos_bt.Value, "0")

ElseIf eos_compmass.Value = vbNullString Then

eos_pe.Value = vbNullString

Else

eos_pe.Value = (eos_compmass.Value) * eos_mp.Value
eos_pe.Value = Format(eos_pe.Value, "0")
```

```
'eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value + eos_svol.Value)
'eos_bt.Value = Format(eos_bt.Value, "0")

'eos_pw.value = (eos_mw.value / 100) * (1 + eos_mw.value / 100) * eos_compmass.value *
eos_mp.value * (1 + eos_ss.value / 100)
'eos_pw.Value = (eos_mw.Value / 100) * (eos_compmass.Value + eos_smass.Value) *
eos_mp.Value
'eos_pw.Value = Format(eos_pw.Value, "0")

End If



If eos_pe.Value = vbNullString Then
totalcost.Value = vbNullString
'ElseIf eos_pw.Value = vbNullString Then
totalcost.Value = vbNullString
ElseIf eos_mc.Value = vbNullString Then
totalcost.Value = vbNullString

Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")

End If


End Sub

Private Sub eos_lt_Change()
eos_slice.Value = (zmax * 1000) / eos_lt.Value
eos_slice.Value = Format(eos_slice.Value, "0")

End Sub

Private Sub eos_mat_Change()
count = eos_mat.ListIndex

eos_mp.ListIndex = count
eos_lt.ListIndex = count
```

```
eos_td.ListIndex = count
eos_br.ListIndex = count
eos_wd.ListIndex = count

If eos_ca.Value = vbNullString Then

Else
eos_pfk.Value = eos_td.Value * (eos_compz.Value / 10) * (eos_bpx.Value / 10) * (eos_bpy.Value
/ 10) / 1000 * (eos_ca.Value / 100)
eos_pfk.Value = Format(eos_pfk.Value, "0")
eos_pfb.Value = eos_mp.Value * eos_pfk.Value
End If


If totalcost.Value <> vbNullString Then
        'Build Time
        tempvar(0) = eos_rr.Value
        tempvar(1) = eos_slice.Value
        tempvar(2) = eos_br.Value
        tempvar(3) = eos_svol.Value
        tempvar(4) = eos_compv.Value
        tempvar(5) = eos_mw.Value / 100
        eos_bt.Value = (tempvar(0) * tempvar(1)) * (1 / 60 * 1 / 60) + 1 / tempvar(2) *
(tempvar(3) + tempvar(4))
        eos_mc.Value = eos_bt * CDbl(eos_mr.Value)

        'Component Mass
        eos_smass.Value = (1 + tempvar(5)) * tempvar(3) * eos_wd.Value / 1000
        eos_pe.Value = tempvar(4) * (1 + tempvar(5)) * eos_wd.Value * eos_mp.Value / 1000
        eos_scost.Value = (CDbl(eos_smass)) * CDbl(eos_mp.Value)

        '85 is powder depreciation for 15 use Ti64 powder
        'eos_pdep.Value = 85 * (eos_pfk - eos_smass - (1 + tempvar(5)) * tempvar(4) / 1000)
        'eos_pdep.Value = 0

        tempvar(6) = (CDbl(eos_pe) + CDbl(eos_scost) + CDbl(eos_mc)) '+ CDbl(eos_pdep))

        totalcost.Value = tempvar(6)

        totalcost.Value = Format(totalcost.Value, "0")
        eos_bt.Value = Format(eos_bt.Value, "0")
        eos_mc.Value = Format(eos_mc, "0")
        eos_pe.Value = Format(eos_pe.Value, "0")
        eos_scost.Value = Format(eos_scost.Value, "0")
        eos_smass.Value = Format(eos_smass.Value, "0.00")
```

```
End If

End Sub

Private Sub eos_mc_Change()

'If totalcost.value = vbNullString Then
'totalcost.value = vbNullString

'If eos_pe.Value = vbNullString Then
'totalcost.Value = vbNullString
'ElseIf eos_pw.Value = vbNullString Then
'totalcost.Value = vbNullString
'ElseIf eos_mc.Value = vbNullString Then
'totalcost.Value = vbNullString

'Else

'totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
'totalcost.Value = Format(totalcost.Value, "0")

'End If

End Sub

Private Sub eos_mp_Change()
If eos_mw.Value = vbNullString Then
'eos_pw.Value = vbNullString

ElseIf eos_compmass.Value = vbNullString Then
'eos_pw.Value = vbNullString
ElseIf eos_smass.Value = vbNullString Then
'eos_pw.Value = vbNullString

Else

'Waste due to Powder Splattering in the AM Process
'eos_pw.Value = (eos_mw.Value / 100) * (eos_compmass.Value + eos_smass.Value) *
eos_mp.Value
'eos_pw.value = (eos_mw.value / 100) * (1 + eos_mw.value / 100) * eos_compmass.value *
eos_mp.value
'eos_pw.Value = Format(eos_pw.Value, "0")
```

```
End If

If eos_ca.Value = vbNullString Then

Else
eos_pfk.Value = eos_td.Value * (eos_compz.Value / 10) * (eos_bpx.Value / 10) * (eos_bpy.Value
/ 10) / 1000 * (eos_ca.Value / 100)
eos_pfk.Value = Format(eos_pfk.Value, "0")
eos_pfb.Value = eos_mp.Value * eos_pfk.Value
End If

If eos_mw.Value = vbNullString Then

'eos_pw.Value = vbNullString

ElseIf eos_compmass.Value = vbNullString Then
'eos_pw.Value = vbNullString
ElseIf eos_smass.Value = vbNullString Then

'eos_pw.Value = vbNullString

Else
'Powder Waste
'eos_pw.Value = (eos_mw.Value / 100) * (eos_compmass.Value + eos_smass.Value) *
eos_mp.Value

'1/9/2017 - eos_pw.value = (eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 +
eos_ss.value / 100)

'eos_pw.value = (eos_mw.value / 100) * (1 + eos_mw.value / 100) * eos_compmass.value *
eos_mp.value
'eos_pw.Value = Format(eos_pw.Value, "0")

End If

If eos_smass.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_compmass.Value = vbNullString Then
eos_pe.Value = vbNullString

Else
'Component Mass
eos_pe.Value = (eos_smass.Value + CDbl(eos_compmass.Value)) * CDbl(eos_mp.Value)
eos_pe.Value = Format(eos_pe.Value, "0.00")
End If
End Sub

Private Sub eos_mr_Change()
If eos_mr.Value = vbNullString Then
ElseIf eos_bt.Value = vbNullString Then
Else
eos_mc = eos_mr.Value * eos_bt.Value

End If

End Sub

Private Sub eos_mw_Change()
If eos_mw.Value = vbNullString Then
'eos_pw.Value = vbNullString
ElseIf eos_compmass.Value = vbNullString Then
'eos_pw.Value = vbNullString

ElseIf eos_smass.Value = vbNullString Then
'eos_pw.Value = vbNullString

Else

'eos_pw.Value = (eos_mw.Value / 100) * (eos_compmass.Value + eos_smass.Value) *
eos_mp.Value
'eos_pw.value = (eos_mw.value / 100) * (1 + eos_mw.value / 100) * eos_compmass.value *
eos_mp.value
'eos_pw.Value = Format(eos_pw.Value, "0")

End If

If eos_smass.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_compmass.Value = vbNullString Then
eos_pe.Value = vbNullString

Else

eos_pe.Value = (eos_smass.Value * eos_compmass.Value) * eos_mp.Value
eos_pe.Value = Format(eos_pe.Value, "0")

End If

'10/25/2016 - TotalCost
```

```
If eos_pe.Value = vbNullString Then
totalcost.Value = vbNullString
'ElseIf eos_pw.Value = vbNullString Then

totalcost.Value = vbNullString
ElseIf eos_mc.Value = vbNullString Then
totalcost.Value = vbNullString

Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")

End If

End Sub


Private Sub eos_rr_Change()
If eos_rr.Value = vbNullString Then
ElseIf eos_svol.Value = vbNullString Then

Else
eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value + eos_svol.Value)
eos_bt.Value = Format(eos_bt.Value, "0")
End If



'10/25/2016 - Addition


If eos_pe.Value = vbNullString Then
totalcost.Value = vbNullString

'ElseIf eos_pw.Value = vbNullString Then
totalcost.Value = vbNullString

ElseIf eos_mc.Value = vbNullString Then
totalcost.Value = vbNullString
```

```
Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")

End If

End Sub

Private Sub eos_ss_Change()
'eos_pe.value = 0
'If Not IsEmpty(eos_compmass.value) Then
'MsgBox ("Select Material")
'Else

If eos_ss.Value = vbNullString Then
eos_pe.Value = vbNullString
ElseIf eos_compmass.Value = vbNullString Then

eos_pe.Value = vbNullString

ElseIf eos_slice.Value = vbNullString Then
eos_pe.Value = vbNullString

ElseIf eos_rr.Value = vbNullString Then
eos_pe.Value = vbNullString

ElseIf eos_br.Value = vbNullString Then
eos_pe.Value = vbNullString

ElseIf eos_mp.Value = vbNullString Then
eos_pe.Value = vbNullString

ElseIf eos_mw.Value = vbNullString Then

'eos_pe.value = vbNullString
eos_pe.Value = (1 + eos_ss.Value / 100) * eos_compmass.Value * eos_mp.Value
eos_pe.Value = Format(eos_pe.Value, "0")
eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value) * (1 + eos_ss.Value / 100)
eos_bt.Value = Format(eos_bt.Value, "0")
```

```
ElseIf eos_compmass.Value = vbNullString Then
eos_pe.Value = vbNullString

Else

eos_pe.Value = (1 + eos_ss.Value / 100) * eos_compmass.Value * eos_mp.Value
eos_pe.Value = Format(eos_pe.Value, "0")

eos_bt.Value = (eos_rr.Value * eos_slice.Value) * (1 / 60 * 1 / 60) + 1 / eos_br.Value *
(eos_compv.Value) * (1 + eos_ss.Value / 100)
eos_bt.Value = Format(eos_bt.Value, "0")

'eos_pw.value = (eos_mw.value / 100) * (1 + eos_mw.value / 100) * eos_compmass.value *
eos_mp.value * (1 + eos_ss.value / 100)
'eos_pw.Value = (eos_mw.Value / 100) * eos_compmass.Value * eos_mp.Value * (1 +
eos_ss.Value / 100)
'eos_pw.Value = Format(eos_pw.Value, "0")

End If


'10/25/2016 Addition

If eos_pe.Value = vbNullString Then
totalcost.Value = vbNullString
'ElseIf eos_pw.Value = vbNullString Then
totalcost.Value = vbNullString
ElseIf eos_mc.Value = vbNullString Then
totalcost.Value = vbNullString
Else

totalcost.Value = (eos_compmass.Value) * eos_mp.Value + (eos_mw.Value / 100) *
eos_compmass.Value * eos_mp.Value + (eos_mr.Value * eos_bt.Value)
'totalcost.value = ((1 + eos_ss.value / 100) * eos_compmass.value * eos_mp.value) +
((eos_mw.value / 100) * eos_compmass.value * eos_mp.value * (1 + eos_ss.value / 100)) +
(eos_mr.value * eos_bt.value)
totalcost.Value = Format(totalcost.Value, "0")

End If

End Sub

Private Sub eos_td_Change()

'eos_pfk.value = eos_td.value * (zmax / 10) * (eos_bpx.value / 10) * (eos_bpy.value / 10) / 100
'eos_pfk.value = Format(eos_pfk.value, "0")
```

```
'eos_pfb.value = eos_mp.value * eos_pfk.value
'eos_compmass.value = eos_td.value * eos_compv.value / 1000
'eos_compmass.value = Format(eos_compmass.value, "0.00")

End Sub

Private Sub eos_wd_Change()
eos_compmass.Value = eos_wd.Value * eos_compv.Value / 1000
eos_compmass.Value = Format(eos_compmass.Value, "0.00")

End Sub


Private Sub MultiPage1_Change()

arcam_compx.Value = eos_compx.Value
arcam_compy.Value = eos_compy.Value
arcam_compz.Value = eos_compz.Value

opto_compx.Value = eos_compx.Value
opto_compy.Value = eos_compy.Value
opto_compz.Value = eos_compz.Value

arcam_compv.Value = eos_compv.Value
arcam_compv.Value = eos_compv.Value
arcam_compv.Value = eos_compv.Value

opto_compv.Value = eos_compv.Value
opto_compv.Value = eos_compv.Value
opto_compv.Value = eos_compv.Value

opto_pack.Value = eos_pack.Value
arcam_pack.Value = eos_pack.Value

End Sub

Private Sub opto_mat_Change()
count = opto_mat.ListIndex

opto_mp.ListIndex = count
opto_td.ListIndex = count
opto_mfr.ListIndex = count

End Sub
```

```
Private Sub CommandButton1_Click()
' --------------SolidWorks Macro Initialization-----------------

   Set swApp = Application.SldWorks
   Set swModel = swApp.ActiveDoc
   'Set swSelMgr = swModel.SelectionManager
   'Set swSelData = swSelMgr.CreateSelectData
   Set swModDocExt = swModel.Extension
   Set swMass = swModDocExt.CreateMassProperty

' --------------Document Configuration-----------------
  'Stop using default units
   swMass.UseSystemUnits = False

   'Set units to custom system
   uuni =
swModDocExt.SetUserPreferenceInteger(swUserPreferenceIntegerValue_e.swUnitSystem,
swUserPreferenceOption_e.swDetailingNoOptionSpecified,
swUnitSystem_e.swUnitSystem_Custom)

   'Set volume units to a given value
   'massi =
swModDocExt.SetUserPreferenceInteger(swUserPreferenceIntegerValue_e.swUnitsMassPropVo
lume, swUserPreferenceOption_e.swDetailingNoOptionSpecified,
swUnitsMassPropVolume_e.swUnitsMassPropVolume_Meters3)

   'Set length units in mass to a given value
   massi =
swModDocExt.SetUserPreferenceInteger(swUserPreferenceIntegerValue_e.swUnitsMassPropLe
ngth, swUserPreferenceOption_e.swDetailingNoOptionSpecified, swLengthUnit_e.swMETER)

' --------------Get Model Property Data-----------------
   partvolume = swMass.Volume
   ' Use only 7 decimal places

   partvolume = Round(partvolume, 7)
   ' value = Format(value, "0.00")

   ' Check if data is in correct format
   MsgBox ("Your Name is " & partvolume)

   'TextBox1.value = value
   'MsgBox ("Your Name is " & TextBox1)

   cboxnum = partvolume * matdensity
   cmassbox.Value = cboxnum
```

```
    cboxnum = partvolume * matcost
    mcostbox.Value = cboxnum

    ' Label3.Caption = TextBox1
    ' Label4.Caption = TextBox2

    ' TextBox1.EnterFieldBehavior = fmEnterFieldBehaviorSelectAll

    mcostbox.Locked = True
    cmassbox.Locked = True

    ' Debug.Print Label3.Caption
    ' Debug.Print Label4.Caption
    ' Debug.Print TextBox1
    ' Debug.Print TextBox2

End Sub

    Private Sub denbox_Change()

End Sub

Private Sub matbox_Change()

    If matbox.Text = "Stainless Steel" Then
    matdensity = 3
    matcost = 5

    ElseIf matbox.Text = "Inconel 718" Then
    matdensity = 6
    matcost = 10
    ElseIf matbox.Text = "Ti-6Al-4V" Then
    matdensity = 9
    matcost = 15
    End If
    denbox.Value = matdensity
End Sub

Private Sub mbox_Change()
If mbox.Text = "EOSINT M280" Then
End If
End Sub
```