caxman
computer aided technologies for additive manufacturing

HORIZON 2020

ANALYSIS TOOLS FOR AM, NON-LINEAR SETTING

DELIVERABLE D2.5

Circulation:	PU: Public
Lead partner:	CNR-IMATI
Contributing partners:	SINTEF, CNR-IMATI, CIMNE
Authors:	Lorenzo Tamellini, Rafael Vazquez, Massimiliano Martinelli, Federico Marini, Paola Pietra, Micol Pennacchio, Silvia Bertoluzza, Marco Attene, Marco Livesu, Vibeke Skytt, Oliver Barrowclough, Michele Chiumenti
Quality Controllers:	Christian Altenhofen, Georg Muntingh
Version:	1.0

Consisting of

SINTEF	STIFTELSEN SINTEF, Department of Applied Mathematics, Oslo, Norway
Fraunhofer	Fraunhofer IGD, Interaktive Engineering Technologien, Darmstadt, Germany
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz GmbH DFKI Innovative Factory Systems, Kaiserlautern, Germany
CNR-IMATI	Consiglio Nazionale Delle Ricerche Istituto di Matematica Applicata e Tecnologie Informatiche, Genova & Pavia , Italy
CIMNE	Centre Internacional de Metodes Numerics en Enginyeria Civil Engineering, Barcelona, Spain
ARCTUR	ARCTUR Racunalniski Inzeniring Doo, R&D, Nova Gorica, Slovenia
BOC	BOC ASSET Management GmbH, Innovation Group, Wien, Austria
Missler	Missler Software Missler Software Service Department, Ramonville St Agne, France
Jotne	Jotne EPM Technology AS, Aeronautics, Space and Defense, Oslo, Norway
STAM SRL	STAM SRL, R&D Department, Genova, Italy
TRIMEK SA	TRIMEK SA, R&D, Altube-Zuia (Alava), Spain
Tronrud	Tronrud Engineering AS, 3D Printing, Hønefoss, Norway
NOVATRA	NOVATRA, Varennes Saint Sauveur, France

DOCUMENT HISTORY

Version[1]	Issue Date	Stage	Content and Changes
0.9	14.07.2017	100 %	Ready to be submitted to Quality Control
1.0	01.08.2017	100 %	Ready to be submitted to Project Officer

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

One of the aims of WP2 within the CAxMan project is to provide the end-user with a seamless tool to perform the thermal and mechanical analysis of the object to be manufactured, using Isogeometric Analysis (IGA). In (Deliverable 2.3) we explained the requirements to develop IGA tools in the linear setting, with special attention to the interactions between the IGA solver and the CAD software. In this report, we analyse the same requirements for the problems in the non-linear setting. Since the CAD-IGA interaction is basically the same for linear and non-linear problems, the focus is mostly on the development of the IGA solver for the non-linear transient heat equation, with non-constant parameter, and non-linear elasticity in a regime of large deformations.

The second part of the report introduces some of the analysis tools that are necessary to develop the optimization loop. We focus on a proof-of-concept shape optimization loop, that is, the modification of some design parameters to optimize the shape of the piece to be printed. We then apply it to improve the design of a simplified version of the teeth of the NUGEAR test case. The objective in this specific example is to optimize the design in such a way that the thermal deformations during the printing process are minimal, thus reducing the machining operations, but we remark that more general optimization problems could be tackled with the same approach.

The development of the optimization loop does not only require design and analysis tools from WP2, but it also involves the meshing tools from WP3 and the printing simulation tools of WP4. A great part of this deliverable is devoted to the description of the necessary interactions.

TABLE OF CONTENTS

TABLE OF AUTHORS FOR EACH SUBSECTION

Section		Authors
	Executive summary	R. Vazquez (IMATI)
1	Introduction	R. Vazquez (IMATI)
2	Time-dependent heat equation solver	R. Vazquez, F. Marini (IMATI)
3	Non-linear elastic equation	L. Tamellini, M. Martinelli (IMATI)
4	Updates on GoTools-IGATools-TopSolid integration	M. Martinelli (IMATI)
5	Optimization	L. Tamellini, P. Pietra (IMATI)
5.1	Sparse grid-based optimization	L. Tamellini, S. Bertoluzza (IMATI)
5.2	Parametric geometry generation	O. Barrowclough (SINTEF)
6	Coating optimization for additive manufacturing	L. Tamellini, M. Pennacchio (IMATI), O. Barrowclough (SINTEF)
6.1	Geometry generation and boundary tessellation	O. Barrowclough (SINTEF)
6.2	Mesh generation	M. Livesu, M. Attene (IMATI)
6.3	Thermal analysis solver	M. Chiumenti (CIMNE)
6.4	Computation of functionals	V. Skytt (SINTEF), L. Tamellini (IMATI)

1 INTRODUCTION

This report is the second on analysis tools for design with additive manufacturing in WP2. It contains the developments and results obtained since the publication of (Deliverable 2.3) in month 16. The content of this report is mainly related to Task 2.4 “Analysis tools for design” and Task 2.5 “Tools for optimization”. It also touches some requirements from Task 2.1 “Interoperability to CAD”, as well as Task 2.2 “Tri-variate shape representation for design and analysis”.

In (Deliverable 2.3) we explained the requirements to develop IGA analysis tools in the linear setting, with special attention to the interactions between the IGA solver and the CAD software. The description of these requirements can be divided in two groups:

1. input information, such as the geometry description, the material properties, and the boundary and working conditions to run the simulation
2. output information, for visualization of results and post-processing

The first part of the present report explains the development of the IGA tools for the non-linear setting, in particular for the transient heat equation with non-constant physical properties, and non-linear elasticity for large deformations. Even if the problems are different from those explained in (Deliverable 2.3), the requirements for the interaction between the IGA solver and the CAD software basically remain the same. For this reason, this report focuses on the description of the mathematical problem and its solution with IGA.

We then shift our focus to the implementation of the optimization loop in CAxMan. Different types of optimization can be considered during the additive manufacturing process, depending on the goal to achieve and the parameters that can be modified to get it. After having recalled some basic mathematical tools to formulate and solve an optimization problem, we detail a proof-of-concept example targeted at improving the design of a simplified version of the teeth of the NUGEAR test case. More specifically, the objective is to optimize the design of the so-called “coating layer” in such a way that the thermal deformations during the printing process are minimal, thus reducing the machining operations.

The development of the shape optimization loop requires a set of design and analysis tools, and in particular, it is necessary to run a new simulation every time the shape of the geometry is changed. A possible method to reduce the computational cost of the optimization process consists in replacing the full-accuracy simulation with a surrogate model in the optimization loop. In this report, we therefore introduce sparse grids, which are an effective method to obtain a surrogate model / response surface for a system with several inputs (in this case, the design parameters). In particular, sparse grids allow for a good balance between: the number of full simulations that need to be executed to “train” the method, the accuracy of the surrogate model, and the speed of evaluation of such surrogate model.

As mentioned before, the goal of the optimization loop is to control the thermal deformation during the printing process. The simulation of this deformation is performed with the tools developed in WP4. To achieve this objective, the meshing and slicing tools developed in WP3 are necessary to discretize the geometry. Therefore, the optimization loop requires the communication among the three work-packages. The last part of this deliverable is devoted to the description of the necessary interactions between the different work-package tools.

The deliverable is organized as follows. In Section 2, we describe the time-dependent heat equation, and its solution with the IGA solver. Section 3 concerns the description and the solution of the non-linear elastic equation. In Section 4, we briefly discuss the interactions between some of the pieces of software used in WP2 (namely IGATools by IMATI, GoTools by SINTEF, TopSolid by Missler) and refer to (Deliverable 2.4) for a more thorough discussion. Finally, we detail the proof-of-concept optimization loop: we recall some fundamental mathematical tools to solve optimization problems and introduce sparse grids in Section 5, and we detail the optimization loop in Section 6, with particular emphasis on the interactions with other WPs.

2 TIME-DEPENDENT HEAT EQUATION SOLVER

The transient heat equation was already introduced in (Deliverable 2.3), although the procedure to compute its solution by numerical simulation was not detailed. We recall from that deliverable that the evolution of the temperature T is given by the heat equation

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = f, \qquad x \in \Omega, t \in [0, t_f]$$

where ρ is the mass density, c_p is the specific heat at constant pressure, k is the heat conductivity and f is a given heat source. Convection terms are neglected.

For the numerical solution of the heat equation, we use an IGA discretization in space, as already described in (Deliverable 2.3), while for the discretization in time we implemented a general θ-scheme, (Quarteroni, 2008). This is a general methodology to generate time-marching schemes, where the user can select from a range of different methods by setting the value of a scalar parameter, θ, between 0 and 1. In particular, the popular backward and forward Euler methods can be obtained setting $\theta = 1$ and $\theta = 0$ respectively, while the Crank-Nicolson method can be obtained by setting $\theta = 0.5$. The difference between these methods will be clarified later on.

Let us fix the time step Δt, and partition the time interval $[0, t_f]$ in N subintervals of length Δt. Denoting by T_n the temperature at time $t_n = n\Delta t$, the general θ-scheme is given by

$$\rho c_p \frac{T_{n+1} - T_n}{\Delta t} - \theta\left(\nabla \cdot (k \nabla T_{n+1})\right) - (1-\theta)(\nabla \cdot \left(k \nabla T_n\right)) = \theta f(t_{n+1}) + (1-\theta)(f(t_n))$$

In particular, the backward Euler scheme is given by

$$\rho c_p \frac{T_{n+1} - T_n}{\Delta t} - \nabla \cdot (k \nabla T_{n+1}) = f(t_{n+1}),$$

the forward Euler scheme is given by

$$\rho c_p \frac{T_{n+1} - T_n}{\Delta t} - \nabla \cdot (k \nabla T_n) = f(t_n),$$

and the second-order Crank-Nicolson scheme is given by

$$\rho c_p \frac{T_{n+1} - T_n}{\Delta t} - \frac{1}{2}(\nabla \cdot (k \nabla T_{n+1} + k \nabla T_n)) = 1/2(f(t_{n+1}) + f(t_n)).$$

Obviously, the accuracy of the methods depends on the size of the time step Δt: the smaller the time step, the more accurate the results. In particular, forward and backward Euler are said to be "first-order methods", because it can be proven that their approximation error is proportional to Δt, while Crank-Nicolson is said to be a "second-order method" because its error is proportional to the ***square*** of Δt. Being Δt typically smaller than 1, this implies that the error of Crank-Nicolson will be substantially smaller than that of Euler methods. Moreover, the methods differ also with respect to their stability: roughly speaking, backward Euler and Crank-Nicolson are guaranteed to reproduce correctly the behavior of the temperature for arbitrary long time intervals for any choice of Δt, while the same is true for forward Euler only if Δt is small enough.

In the CAxMan use cases, the material properties of the components to be analysed will depend on the temperature, and this dependency is expressed in the transient heat equation as follows:

$$\rho(T)c_p(T)\frac{\partial T}{\partial t} - \nabla \cdot (k(T)\nabla T) = f \quad x \in \Omega, t \in [0, t_f]$$

We remark that the fact that the physical properties depend on the temperature introduces non-linearities in our problem, since the terms appearing in the heat equation depend on the solution of the equation itself.

In order to deal with the non-linearities, a simple and efficient approach is to evaluate the material properties considering the temperature at the previous time step. For instance, for the backward Euler method this would result in solving, at each time step, the equation

$$\rho(T_n)c_p(T_n)\frac{T_{n+1} - T_n}{\Delta t} - \nabla \cdot (k(T_n)\nabla T_{n+1}) = f(t_{n+1})$$

This procedure allows solving a linear problem at each time step, without introducing any further loop to handle the non-linear terms.

The IGATools implementation has been validated by verifying that the code is able to reproduce the correct convergence properties of the θ-scheme. A problem with a manufactured solution has been solved numerically with IGATools and it has been verified that first-order error convergence is obtained for the backward and forward Euler methods, while second order is obtained for the Crank-Nicolson method.

Figure 1 shows some snapshots taken from the simulation of a thick quarter of a ring made of Titanium ($c_p = 520J/(kg \cdot K), k = 21.9W/(m \cdot K), \rho = 4507kg/m^3$), which is one of the materials that can be used by the EOS-M280 printers used in CAxMan.

The piece has an internal radius of 5cm, an external radius of 10cm and a height of 5cm. It is initially at thermal equilibrium at $T = 20C$, and at time $t = 0$ one of the two square faces is put in contact with a hot surface at $T = 50C$, while the other one is kept at $T = 20C$ and the remaining surfaces are assumed to exchange no heat with the external environment. As expected, the temperature propagates through the piece, and the new thermal equilibrium is reached after approximately 7 minutes.

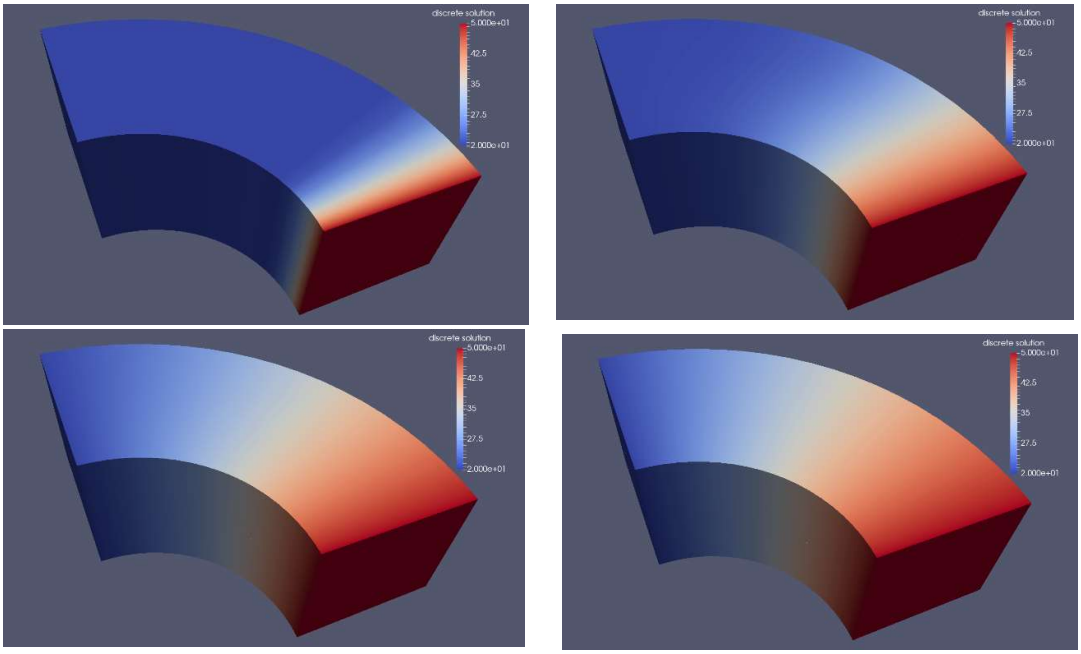

FIGURE 1: HEAT PROPAGATION THROUGH A THICK QUARTER OF ANNULUS (TOP-LEFT: 12 SECONDS; TOP-RIGHT: 1 MINUTE; BOTTOM-LEFT: 4 MINUTES; BOTTOM-RIGHT: 7 MINUTES). COLOR IS PROPORTIONAL TO TEMPERATURE IN CENTIGRADES.

3 NON-LINEAR ELASTIC EQUATION

In (Deliverable 2.3), the equations of linear elasticity have been discussed, and a solver for these equations is currently being incorporated into the CAxMan software ecosystem. Here, we discuss how to extend this approach to the more general non-linear elasticity case.

The goal of the elasticity computations is to determine the internal status of tension of a body according to the external loading. The stresses are proportional to the deformations. Therefore, in general, solving the elastic equation means finding the proper deformation that generates a suitable stress to counterbalance the external actions. In other words, both the deformations and the status of internal tensions that they generate are the unknowns of the problem, which need to be computed.

The crucial assumption in the linear regime is that the deformations from the position at rest of the body due to an external force is much smaller than the characteristic lengths of the body, so that the deformed and undeformed bodies are indistinguishable. This means that only the internal field needs to be computed, because even an infinitesimal deformation will generate a large enough reaction to balance external forces.

Instead, coming back to the non-linear regime, one needs to solve the so-called virtual work equation (Bonet, 2008)

$$\int_{vol}(div[\sigma]+\mathbf{f})\cdot\delta v dvol = 0$$

Where vol is the actual (unknown) configuration of the body, $dvol$ is an infinitesimal portion of such body, δv is any admissible displacement, $\mathbf{f}$ are external forces and $[\sigma]$ is the stress tensor, which depends on the actual deformation of the body through some constitutive equation. Thus, the unknown deformation appears both as an argument of $[\sigma]$ and in vol, resulting in a non-linear equation.

In CAxMan, we consider a hyperelastic compressible isotropic neo-Hookean material, in which the link between deformation and stress is analogous to the one in the linear regime (Deliverable 2.3, p. 9):

$$\sigma = \frac{\mu}{J}(b-I)+\frac{\lambda}{J}\log(J)I$$

where the deformation from rest to loaded position is described by a function $x=\Phi(X)$ whose gradient is $F=\frac{\partial\Phi}{\partial X}$, $J=\det(F)$ is the pointwise volume change, μ and λ are material parameters, $b=FF^T$, the superscript $\cdot^T$ denotes matrix transposition and I is the identity matrix.

Extensions of IGATools++ Isogeometric analysis software to solve these generalized equations are under development. It must be emphasized at this point that the end-users STAM and Novatra have decided that for most of CAxMan applications, the linear solver already available will be enough.

The approach to solve the equation consists of two nested iterative loops as shown in Algorithm 1. The outer loop gradually applies the external load. The total external forces $\mathbf{f}$ are divided into a sum of increments $\delta\mathbf{f}$, the equilibrium position due to $\delta\mathbf{f}$ is computed, then $\delta\mathbf{f}$ is increased to $2\times\delta\mathbf{f}$ and a new equilibrium position is computed, until the entire load is applied. This "gentle loading" is not necessary from a physical point of view, but is useful to ease the convergence of the inner loop, which computes the deformation due to the current load. More specifically, the inner loop takes the deformation for the previous loading as an initial guess and gradually increases it, until the deformation matches the current load. This operation is based on an "alternating principle": given a deformation, the algorithm computes the corresponding stress, the unbalance between this stress and the external load generates a further deformation, and so on.

```
f̃ = 0 ;
vol = rest volume;
Repeat
      f̃ = f̃ + δf
      Repeat
            Compute stress [σ] and new displaced volume vol
      Until [σ] balances f̃
Until f̃ is equal to f
```

ALGORITHM 1: NON LINEAR ELASTICITY ITERATIVE ALGORITHM

4 UPDATE ON GOTOOLS-IGATOOLS-TOPSOLID INTEGRATION

As discussed in (Deliverable 2.4), .g22 is a GoTools file format that will be used for WP2 internal interoperability among GoTools, IGATools and Missler's TopSolid. IGATools can already import geometries in the previous GoTools format, i.e., .g2, as detailed in (Deliverable 2.2). However, the novel .g22 format has been introduced due to the fact that the softwares upstream in the analysis process (i.e., TopSolid and GoTools) need to communicate to IGATools not only the geometry but also further information needed for analysis, such as boundary conditions and material properties. The extension of IGATools to support .g22 format is currently under development. Figure 2 shows an example of a block-structured geometry stored in .g22 format and read by IGATools. We refer to (Deliverable 2.4) for more information on the .g22 file format.

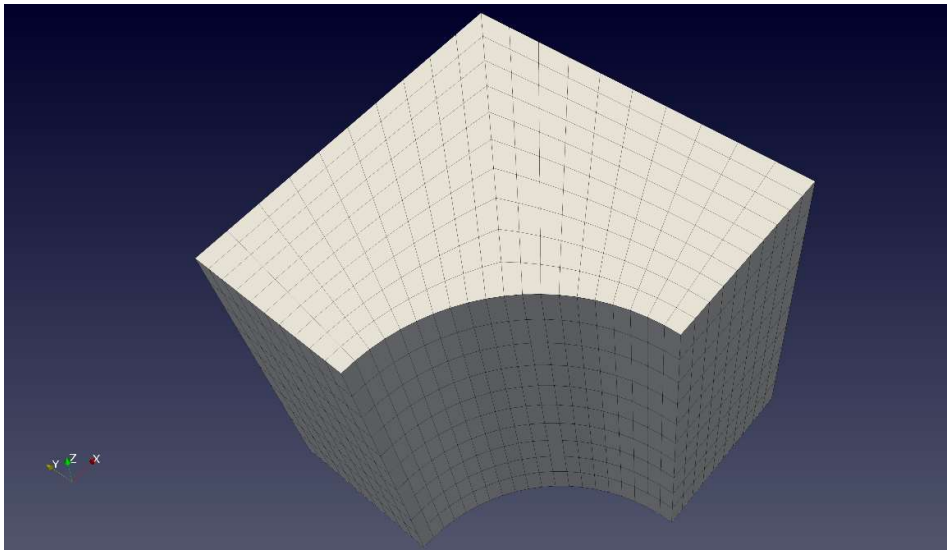

FIGURE 2: A BLOCK-STRUCTURED GEOMETRY STORED IN THE .G22 FORMAT, READ BY IGATOOLS

IGATools and GoTools need a strong interaction also to perform analysis of trimmed geometries, with GoTools functionalities being used by IGATools during the pre-processing of the trimmed elements of the geometry (assembly of the linear system matrix and right-hand side). In this phase, the so-called reparametrization procedure (see (Deliverable 2.3) and (Deliverable 2.4)) needs to be performed multiple times, and therefore it is crucial that GoTools can keep the computational times to a minimum (ideally, some minutes) to allow for analysis of complex geometries with multiple trimmed elements.

5 OPTIMIZATION

One of the main features of additive manufacturing is the possibility to print complex shapes that can deliver a much better performance under thermal or mechanical loading compared to their standard design counterparts. Thus, design optimization is a natural field for applying CAxMan analysis tools. In this section, we will first recall some fundamental mathematical aspects of optimization theory and then focus on a proof-of-concept optimization problem that concerns the optimization of the extra layer of material in the stock model that needs to be added to the actual object while printing, to allow the subsequent machining operation. While this is just a relatively easy proof-of-concept test, we remark that it is already complex enough so that the backbone of

the software chain of a generic optimization loop needs to be up and running. Extensions to optimize other kind of goals (e.g., working performance of the printed objects, manufacturing time/cost etc.) could be obtained without major changes in the overall strategy.

Optimization can be broadly divided in shape optimization and topology optimization. In the former, a predefined reference configuration is chosen, and can be bent, folded, distorted, etc. during the optimization loop, while keeping the same "macroscopic" configuration, e.g., without adding/removing features such as holes, branches, cuts etc. The addition and removal of such features is done during the topology optimization loop. In this deliverable, we deal with shape optimization only.

The most general mathematical formulation of an optimization problem is

$$\begin{aligned} &\min_{\mathbf{p}\in\mathbb{R}^N} && f(\mathbf{p}) \\ &st: && g(\mathbf{p}) \le 0 \\ & && h(\mathbf{p}) = 0 \end{aligned}$$

where $f(\mathbf{p})$ is the so-called objective function that one wishes to optimize and depends on $\mathbf{p}$, a set of N design parameters that can be varied (either CAD control points or geometrical measures such as angles or lengths) within a feasible set of choices. Such a feasible set is defined by the (non-linear) equations $g(\mathbf{p}) \le 0$ and $h(\mathbf{p}) = 0$. The function $g(\mathbf{p})$ is usually termed "inequality constraints function", while $h(\mathbf{p})$ is called "equality constraints function". In this introductory section, we assume for simplicity that all constraints are inequalities, i.e., $h(\mathbf{p})$ disappears in the equations above; problems with both equality and inequality constraints can be treated in a similar way. Moreover, we assume that the design parameters are not bound to assume only integer or fractional values, i.e., they are real numbers. In the following, we denote by N the number of parameters in the above optimization problem.

A large body of literature has been produced in the last 70 years about solving optimization problems (Nocedal, 2006). Unconstrained problems, i.e.

$$\min_{\mathbf{p}\in\mathbb{R}^N} f(\mathbf{p}), \quad \mathbf{p} \in \mathbb{R}^N$$

can be efficiently solved with iterative methods that perform a sequence of steps from an initial guess $\mathbf{p}_0$ to the optimal solution $\mathbf{p}_{opt}$. They can be broadly categorized in gradient methods (Newton and quasi-Newton methods, steepest descent, conjugate gradient) and gradient-free method (Nelder-Mead or simplex, genetic algorithms etc). The former are more efficient but require a good initial guess $\mathbf{p}_0$ and information about the gradient of the objective function, which might not be available. The latter are less efficient but more robust even for poor initial guesses, and do not require information on the objective gradient.

Constrained problems, i.e.,

$$\begin{aligned} &\min_{\mathbf{p}\in\mathbb{R}^N} f(\mathbf{p}) \\ &st: g(\mathbf{p}) \le 0 \end{aligned}$$

can be solved by several different strategies. A full survey of the methods exceeds the scope of this deliverable and we refer the interested reader to (Nocedal, 2006). Here we focus on penalization methods, where the initial constrained problem is replaced by a modified unconstrained problem:

$$\min_{\mathbf{p}\in\mathbb{R}^N} f(\mathbf{p}) + \tilde{g}(\mathbf{p}), \quad \mathbf{p} \in \mathbb{R}^N$$

where $\tilde{g} \gg 0$ if $g(\mathbf{p}) > 0$, i.e. if the parameters are unfeasible. In such a way, choosing a non-feasible $\mathbf{p}$ will be less likely. We will actually not restrict ourselves to a single method, but we will run several methods and choose the best solution among those found. Indeed, as will be clearer later, the vast majority of the computational cost will reside in solving a preliminary number of full model computer simulations for a selected set of geometries, while the call to the optimizer upon those values will only require a few seconds. The penalization methods that we will consider in this pool are:

- squared penalty, in which $\tilde{g} = M\sum_{j=1}^{N} \max\{g_j(\mathbf{p}), 0\}^2$, for $M \gg 1$
- augmented Lagrangian, which is an improved variation of the square penalty method
- log barrier, in which $\tilde{g} = -M\sum_{j=1}^{N} \log(-g_j(\mathbf{p}))$, for $M \gg 1$

Note that for squared penalty and augmented Lagrangian, $\tilde{g}(\mathbf{p})$ is positive only if $\mathbf{p}$ is outside the feasible region, while for the log-barrier method, $\tilde{g}(\mathbf{p})$ gets larger and larger as we move from the interior to the border of the feasible region, so that this method will not be able to catch solutions that lie on the boundary of the feasible domain.

The number of parameters clearly plays a crucial role in this setting: the more parameters, the better the final result but the more computational intensive the optimization procedure (more precisely, in view of the discussion above, the more full-model simulations will be needed). In principle, as already mentioned, the parameters to play with could be either the control points of the CAD file (all of them or a portion) or some "physical geometrical parameters" (lengths or angles).

In CAxMan, we consider the latter approach, which is closer to the designer intuition and allows to keep the number of parameters to a minimum, yet with a good "expressivity", meaning that their variations will have a strong impact on the objective function. The procedure that we will present in the following can be applied effectively up to about a dozen parameters, which is deemed enough for CAxMan purposes.

The crucial feature of the optimization problem we might consider in CAxMan is that we do not have an analytic expression of the objective function nor of the constraints, but we only know its value by solving an (expensive) analysis equation (thermal or mechanical). Therefore, it is vital to keep the number of evaluations of the objective function and constraints to a minimum. Moreover, we do not have access to the exact gradient of $f(\mathbf{p})$, but some sort of approximation thereof will be needed if we want to resort to gradient methods. On top of this, we remark that of course both $f(\mathbf{p})$ and $g(\mathbf{p})$ are non-linear functions of $\mathbf{p}$.

Thus, in CAxMan we propose a two-steps procedure. First, we build a surrogate model for the objective function, i.e., we solve the full analysis on a small and wisely selected set of parameters combinations and we build a polynomial approximation of $f(\mathbf{p})$ and $g(\mathbf{p})$ based on these values. Then, we replace $f(\mathbf{p})$ and $g(\mathbf{p})$ in the optimization problem with their approximations. The rationale here is that while the optimization loop might require hundreds of evaluations of the full model, it is reasonable to expect that a handful (a few dozens) of the full model evaluations should

be sufficient to build a decent approximation of $f(\mathbf{p})$ and $g(\mathbf{p})$. Solving an optimization problem based on polynomials will essentially add no cost to the overall procedure (a few seconds/minutes for the optimizer vs some hours for the solution of the full model). Moreover, we will be allowed to use gradient-methods in the optimization, because we can compute exactly and automatically the gradient of the polynomial surrogate models.

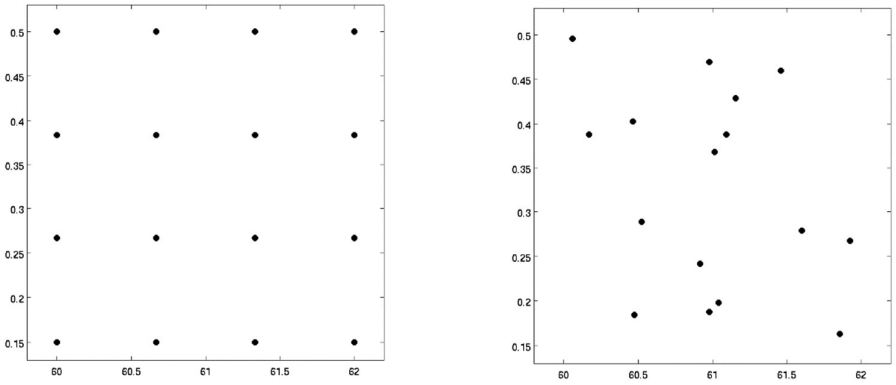

FIGURE 3: LEFT: CARTESIAN GRID; RIGHT: MONTE CARLO SAMPLING. BOTH GRIDS ARE ON THE PARAMETER SPACE THAT WILL BE USED FOR THE TEST CASE EXPLAINED IN THE NEXT SECTION. BOTH GRIDS HAVE 16 POINTS.

5.1 SPARSE GRID-BASED OPTIMIZATION

The most critical step here is the choice of the points in the parameter space where the full problem should be solved. The choice should cover the parameter space well, guarantee good accuracy of the resulting surrogate model and keep the number of points to a minimum. A naive choice would be to cover the parametric domain with a Cartesian grid (see Figure 3, left): however, the number of points in this sampling scheme grows exponentially in the number of parameters (i.e., a Cartesian grid with M points per parameter has M^N points), and therefore would quickly lead to a prohibitive number of full problems to be solved. Good coverage of the parameter space at a reduced computational cost is achieved with Monte Carlo sampling (see Figure 3, right), but the quality of the approximation might be fairly poor.

In CAxMan, we consider the sparse grid method (Bungartz, 2004), which has been successfully employed in optimization e.g. in (Valentin, 2016). This method devises a peculiar strategy to choose the points in the parametric domain, different from a Cartesian grid and in a non-aleatoric way, which reduces the number of points and guarantees good accuracy. Of course, the computational time of a single full-model run (i.e., the cost of solving a PDE) will still represent the major computational burden, i.e., if the computational budget of the user is very limited even advanced sampling methods such as sparse grids will allow to perform optimization on a very small number of parameters only.

The basic idea of sparse grids is as follows. On one hand, an approximation based on a **single** Cartesian grid refined along **all** parametric directions **at once** is in general very accurate but too expensive, as we have already discussed. On the other hand, building **many** small Cartesian approximations based on grids in which parametric directions are **never refined simultaneously** and combining them together could be computationally less demanding yet

delivering good approximations, because each of the components is accurate at least with respect to some parameters. Thus, several Cartesian grids are generated, and the resulting surrogate model is just a suitable linear combination of all the intermediate surrogate models. A somehow similar procedure is known in literature as “Richardson extrapolation” (Brezinski, 1991) where an accurate approximation is obtained by suitably combining several coarse approximations. Detailing the procedure with which the grids are selected and combined exceeds the scope of this deliverable and we refer to (Bungartz, 2004) for further details. Superposing all the points required by the method in the same axes results in Figure 4, which shows that the parametric space is covered in a "cross-like" fashion.

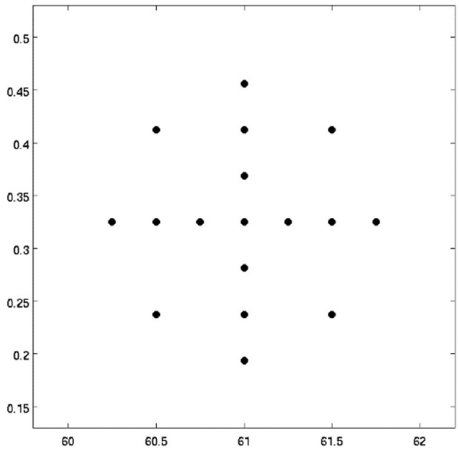

FIGURE 4: SPARSE GRID ON THE PARAMETER SPACE THAT WILL BE USED FOR THE TEST CASE EXPLAINED IN THE NEXT SECTION. NOTE THAT THIS SPARSE GRID HAS 17 POINTS, COMPARABLE TO THE NUMBER OF POINTS IN THE CARTESIAN AND MONTE CARLO GRIDS SHOWN IN THE PREVIOUS PICTURE

Another advantage of this procedure is that the selection of points can be made asynchronously. The points can be chosen adaptively as the results of the full model come in, or can be prescribed a-priori and then read in, once the simulations are done.

In CAxMan, we use the implementation of sparse grid optimization provided by the C++ software SG++, available at www.sgpp.sparsegrids.org and freely available under BSD licence, see (Valentin, 2016) and (Pfluger, 2010). This software provides several methods to generate sparse grids (adaptive, asynchronous) and comes equipped with optimization routines for constrained problems, which can be adapted to CAxMan needs with reasonable effort.

5.2 PARAMETRIC GEOMETRY GENERATION

It should be clear by now that the shape optimization algorithm proposed here needs to generate several geometries by specifying the values of the chosen optimization parameters, i.e., a specific set of geometric entities such as lengths, angles, radii, etc. This operation is called “parametric modeling” and an alternative to “direct modeling” (as is the case with spline-based modeling). GoTools mainly contains functionality for direct modeling, but has been extended to support the trial example detailed in the following sections. We mention in-passing that parametric modeling is often better at preserving the design intent, though it is much less flexible than direct modeling

and problems with conflicting parameters may occur. Therefore, in practice, parametric and direct modeling techniques are often used side-by-side.

6 COATING OPTIMIZATION FOR ADDITIVE MANUFACTURING

Although additive manufacturing has been conceived to produce objects that were "un-producible", or to ease the production of complex objects, the outcome of the additive manufacturing process often needs some further machining via subtractive manufacturing (see WP5 and WP6). This post-process ensures e.g. smoothness of the surfaces or adherence to the original design, due to 3D printer "errors"/"imprecision" or to the thermal deformation induced by the printing process, see (Deliverable 3.3).

In this preliminary optimization procedure, we assume that the printing errors are negligible, so we only need to deal with deviations from the nominal object induced by temperature. More precisely, we need to make sure that despite these deformations, there is enough material on each side to perform the machining, which will typically remove a layer 0.05mm thick from each face. On the other hand, we want to minimize the waste of metal powder as well as of machining time. Therefore, we solve this problem by an optimization approach.

In other words, we want to determine the best shape to be given as input to the printer such that the subsequent machining operations are minimal, keeping into account the thermal distortion effects, which will be predicted by the thermal analysis tools developed in Work Package 4, see (Deliverable 4.2). The optimal input shape for the 3D printer will differ from what we actually want to obtain at the end of the entire process, in the sense that it will be likely thicker in the areas where larger deformations are predicted.

In this optimization scenario, we propose to "embed" the target geometry in a coating volume, whose shape can be modified by varying some parameters within a feasibility range, and we will determine the optimal combination of such parameters. In mathematical terms, the problem to be solved is written as

$$\begin{aligned} &\min_{\mathbf{p}\in\Omega} \Delta vol(\mathbf{p}) \\ &st.\Delta\text{thickness}(\mathbf{p}) > 0.05mm \\ &\text{thermal simulation}(\mathbf{p}) = 0 \end{aligned}$$

where:

- $\mathbf{p}$ can be chosen in $\Omega = [p_{1,min}, p_{1,max}] \times [p_{2,min}, p_{2,max}] \times \ldots \times [p_{N,min}, p_{N,max}] \subset \mathbb{R}^N$;
- $\Delta vol(\mathbf{p})$ measures the difference in volume between the target geometry and the printed version of the coating prism corresponding to the values of the design parameters specified by $\mathbf{p}$;
- $\Delta\text{thickness}(\mathbf{p})$ measures the thickness that needs to be machined away on each face of the printed coating prism; as we said, the additional thickness needs to be larger than 0.05mm on each face due to the machining tool precision;
- $\text{thermal simulation}(\mathbf{p}) = 0$ means that we predict the shape of the printed coating prism by a thermal analysis simulation of the AM process as in WP4, i.e., by solving the equation that describes the space-time evolution of the thermal AM process.

In detail, we define a nominal geometry to mimic one of the teeth of the NUGEAR. The nominal geometry is shaped as a prism with trapezoidal side faces, and the coating volume that contains it is also a prism, this time however with non-parallel trapezoidal side faces, i.e., the bottom and the top faces have different lengths; see Figure 5 below. The optimization parameters are the angle α between the base and the flank of the tooth, i.e., the bottom angle of the trapezoid (which is assumed to be symmetric) and the overhang of the top face, δ, i.e., $N=2$ and $\mathbf{p}=[\alpha,\delta]$. The nominal geometry is chosen to be the geometry with $\alpha=60^{\circ}$ and $\delta=0$, and the ranges of variation of the parameters defining the coating volumes are $60^{\circ}\le\alpha\le 62^{\circ}, 0.15\le\delta\le 0.5$. Note that this choice ensures that the nominal geometry is contained within each coating volume (but it is not certain that the actual printed part will have enough extra material to allow the 0.05mm machining due to the thermal distortions. Finding a feasible shape is also one of the goals of the optimization). The basis of the nominal as well as coating geometries are rectangles of dimensions $10mm\times(1+\frac{8}{\sqrt{3}})mm$, and the height is 4mm. This slightly strange choice ensures that the width of the top of the tooth w is equal to 1 for the nominal geometry. For the coating volumes, w will vary depending on the values of α and δ, but will always satisfy $w\ge 1$.

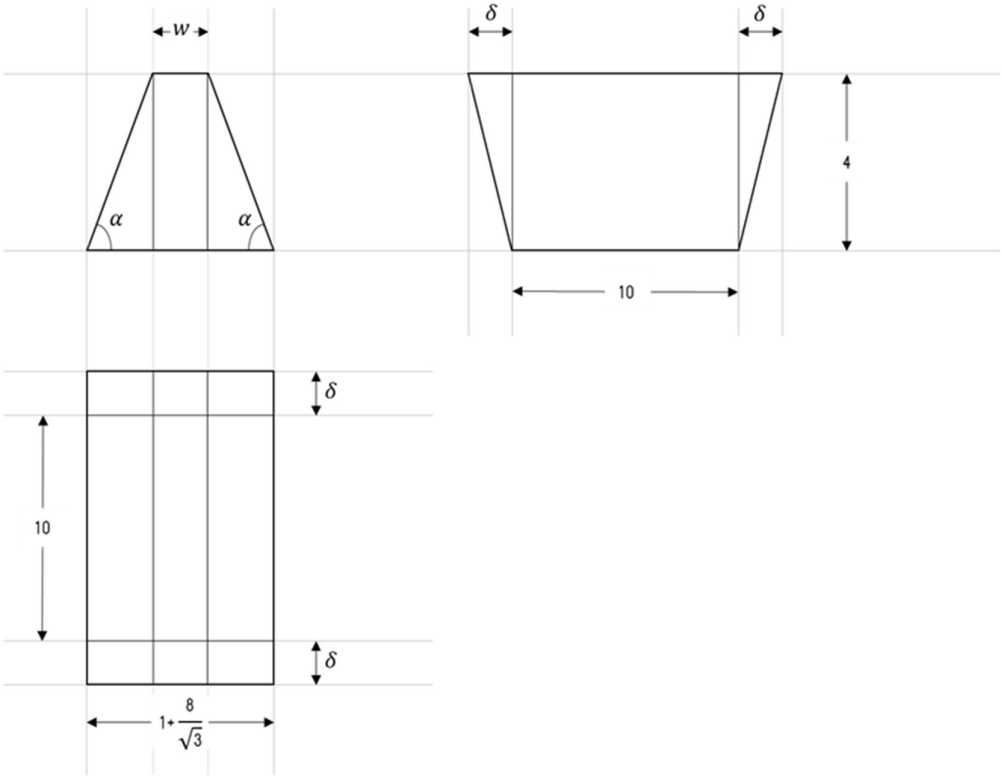

FIGURE 5: PARAMETRIC COATING VOLUME

In this proof of concept, we have chosen the optimization with asynchronous sparse grids construction, which considerably simplifies the data flow between the partners and software involved. In detail, the workflow is as follows:

0) The nominal geometry is tessellated.
1) The list of parameters combinations where evaluation of $\Delta vol(\mathbf{p})$ and $\Delta\text{thickness}(\mathbf{p})$ is needed to build the sparse grid approximation is generated by SG++ and saved to an ASCII file.
2) For each parameter combination, a geometry is generated and the boundary is tessellated.
3) Each boundary tessellation is sliced and a mesh suitable for the simulation is generated (Deliverable 3.3).
4) A full thermal model is solved on each mesh (Deliverable 4.2).
5) For each model run, $\Delta vol(\mathbf{p})$ and $\Delta\text{thickness}(\mathbf{p})$ are computed against the nominal geometry.
6) The values of $\Delta vol(\mathbf{p})$ and $\Delta\text{thickness}(\mathbf{p})$ are written to a file and read back in by SG++.
7) The SG++ constrained optimizers listed above are invoked.

Of course, in a later stage the communication between the steps will be performed by direct software communication (API) instead of file exchange. We remark that although very simple, this method could be easily generalized to more complex and generic geometries: for instance, using as optimization parameters the extra-thickness of the coating on a number of surfaces.

In the following, we give a few details on the main challenges of points 2 to 5.

6.1 GEOMETRY GENERATION AND BOUNDARY TESSELLATION

The first step of the optimization loop is to generate the nominal geometry as well as the geometries that will be tested to construct the surrogate model. These operations are performed by GoTools by parametric modeling. A subset of such modeling functionalities has been added to GoTools, and in particular, the coating prism is going to be stored in GoTools as a general, newly defined, Hexahedron class. Any instance of the Hexahedron class can be converted to a SplineVolume and it is thus compatible with the block-structured isogeometric representation described in (Deliverable 2.2).

The outcome of the procedure that will be passed to the subsequent stage of the optimization loop (i.e., the mesh generation for the printing process simulation) is a surface tessellation that can be defined from the boundary surfaces of the volume block. In detail, the procedure for generating the geometric tessellations is as follows:

1. For each parameter pair in the list, read in the parameter pair.
2. Construct a Hexahedron from the parameter pair, based on the geometry described above.
3. Convert the Hexahedron to a SplineVolume.
4. Extract the boundary surfaces of the SplineVolume as SplineSurfaces.
5. Tessellate the SplineSurfaces.
6. Store the result as a PLY file.

As already mentioned, in this first instance of the optimization loop, the PLY file format is being used for simplicity. Later versions will instead use the standardized formats agreed upon in the consortium.

6.2 MESH GENERATION

To generate the meshes for the simulation we exploit the processing tools developed in WP3. With reference to CAxMan (Deliverable 3.3), the processing tool P2.5 that has been developed to perform the slicing reads the aforementioned PLY. A set of parallel planes aligned along the Z-direction is produced to cut the part, and each pair of consecutive planes in the set is separated by a distance equal to the layer thickness (20 microns). The intersection between each of the planes and the tessellation produces one of the polylines that constitute the slices, and the whole set of slices is saved to a CLI file that the printer can process to proceed with the actual fabrication.

Notice that in this first instance of the optimization loop, we are using a simplified version of the slicing tool P2.5, which assumes the part is pre-oriented and the layer thickness is fixed.

Later versions will substitute the PLY file with an annotated tessellation, that is, a ZIP file containing the tessellated geometry along with all the necessary information (printing direction, layer thickness, etc.) according to the format described in (Deliverable 3.2).

Then, we have developed another processing tool (PT3.6) to be chained after the slicer in the process planning workflow to create a volumetric (tetrahedral) mesh out of the slices. Each slice in the input CLI file is extruded along the vertical direction, up to the next slice, and tested for intersections with it. The extruded slice is triangulated and intersection points are added to the geometry in order to guarantee a conforming surface mesh. Notice that also the inner (planar) area of each slice is triangulated. The resulting geometry is a non-manifold Piecewise Linear Complex (PLC) that externally conforms with the virtual prototype and internally conforms with all the slices contained in the CLI file. The PLC is used as a constraint to produce an accurate tetrahedral mesh that is forwarded to the simulation module.

Note that conformity is rather important in order to realize accurate simulations of the printing process. These simulations, indeed, have a dynamic nature and must be able to consider each single layer as a separate entity that can exist only at a certain time.

Our target printer (EOS M280) is capable of reproducing digital models at an impressive precision, with a layer thickness that can be as small as $h_1 = 20$ microns, see Figure 6-left. Though this precision is certainly desirable in the physical replicas, the resulting meshes are excessively complex to be used as Finite Element meshes in the simulation module.

On the other hand, taking a larger layer thickness $h_2 > h_1$ would introduce an error on the boundary (i.e., a staircase effect, see Figure 6-center) that is larger than the deformations due to the residual stresses, and thus would spoil the whole optimization. Therefore, as a first step, we have decided to produce a mesh which is just conformal with the initial STL model that is sent to the slicer (i.e., instead of extruding layers, the original trapezoidal face is meshed directly as shown in Figure 6-right). The meshing process strives to produce tetrahedra whose size is comparable with h_2, i.e., the "increased" layer thickness that would have been used to produce a small enough mesh.

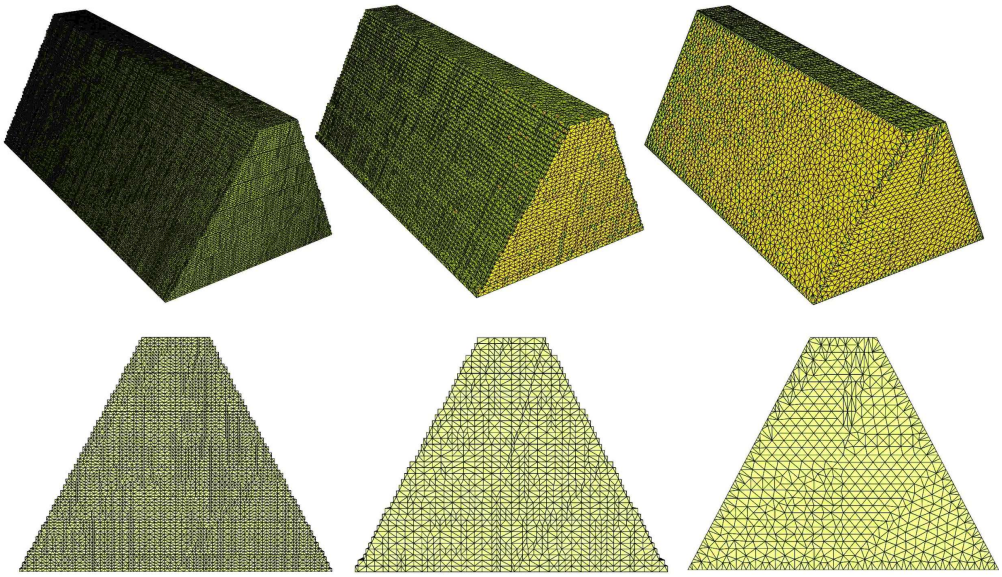

FIGURE 6: MESHES OF ONE EXAMPLE OF PARAMETRIZED GEOMETRY. LEFT: 20-MICRONS LAYERS; CENTER: LARGER LAYERS PRODUCE AN UNDESIRED STAIRCASE EFFECT; RIGHT: MESH CONFORMAL TO THE INITIAL STL.

6.3 THERMAL ANALYSIS SOLVER

The AM technique studied in this work is the so-called selective laser melting (SLM). It consists of uniformly spreading the material powder by a wiper forming the pre-deposited layer. Then the particles pertaining to the scanning path are melted by a high power laser, fusing and solidifying to form a new layer of the component. This process is repeated until the whole component is built.

Different computational frameworks have been developed for the numerical simulation of additive manufacturing techniques. They usually solve the coupled thermal and mechanical problem employing a high-fidelity scanning path for the laser. This approach provides a reliable and accurate prediction of the distortion and residual stresses but it is expensive in terms of computational resources and CPU time. If the manufacturing of large industrial components must be simulated, the computational cost makes it unfeasible to run at full precision. Consequently, the implementation of simplified methods is necessary to reduce the CPU time. In Table 1 below, a comparison between the different computational methods, in terms of accuracy, computational cost, and complexity, is presented (*=low, **=medium, ***=high). The main challenge is to strike a balance between accuracy and CPU time.

MODELING METHOD		ACCURACY	COMPUTATIONAL COST	COMPLEXITY
Thermo-mechanical analysis		***	*** (days)	***
Simplified methodologies	Inherent Shrinkage	*	** (several hours)	*
	Inherent Strain	**	* (few hours or minutes)	***

TABLE 1: COMPARSION OF COMPUTATIONAL METHODS FOR THERMAL ANALYSIS

In this work, the **Inherent Shrinkage Method** has been selected to deal with the numerical simulation of the Additive Manufacturing process by selective laser melting (SLM) with good balance between accuracy and computational requirements.

The main hypothesis within the Inherent Shrinkage Method consists of assuming that the thermal load (laser source) is very localized at the heat affected zone. Hence, the sintering process going on does not affect the remaining part of the domain. This given, the classical thermo-mechanical analysis is skipped, so that the original transient problem is replaced by a sequence of mechanical computations according to the deposition strategy. A further simplification consists of assuming a layer-by-layer deposition, skipping the high-fidelity simulation of the scanning sequence necessary to complete each layer of the domain. The Inherent Shrinkage Method considers that the main source of distortion is due to the material shrinkage of the thermal affected powder during the cooling/sintering phase. This contraction is introduced into the mechanical problem as user-defined process data in terms of sintering temperature and expansion coefficient (material property). The computation is defined by adding new layers to the computational domain following the sequence of powder layers spread one after the other until completing the manufacturing of the full geometry.

The final result coming from the manufacturing process simulation is presented in terms of **accumulated distortions** of the component according to the additive manufacturing sequence. Those distortions will define a distorted volume to be compared to the nominal geometry to be actually produced.

6.4 COMPUTATION OF FUNCTIONALS

The final step is to compute the value of $\Delta\text{thickness}(\mathbf{p})$ and of $\Delta vol(\mathbf{p})$ for each tested geometry.

To compute $\Delta thickness(\mathbf{p})$ we need to evaluate the distances between the distorted versions of the hexahedrons constructed for each parameter pair and the nominal hexahedron. As an output of the simulation, the position of the nodes of computational meshes corresponding to the distorted shape of each object is known and nodes corresponding to the surfaces of the object are separated giving one point set for each parameter sample. From the parametric modeling phase, a surface model representing the boundary of each object is also known. Thus, the information required to compute distances, and in particular the minimum and maximum signed distances, between the distorted mesh and the nominal shape exists.

The CAxMan infrastructure builds upon the infrastructure of CloudFlow (FP7-2013-NMP-ICT-FoF-609100), and we will adapt another CloudFlow result in order to use it in the optimization loop. CloudFlow organized a number of experiments and one of those was about comparing point clouds to CAD models. The purpose was to:

- Compare point cloud scans of manufactured parts to their nominal CAD shape.
- Check older parts for wear to support the decision making process around the need for repairs and upgrades.

This experiment has resulted in a workflow in the Cloud and the process is as follows:

1. Perform a coarse alignment of points to the CAD model by manually connecting a few points in the point set to positions in the CAD model.
2. Automatically perform registration starting from the initially aligned points.

3. Compute distances between the point set and the CAD model by projection. A closest point approach is chosen. The procedure is optimized regarding closest point computations with respect to trimmed surfaces and identification of the surface where a closest point belongs in a model with hundreds of surfaces. The distance computation is parallelized on several nodes.
4. Present results visually and as a report.

Our task is much simpler than the generic task of the CloudFlow experiment. The simulation is performed on relatively coarse meshes, thus the number of points is much lower than the CloudFlow expectations. Each hexahedron boundary shell consists of 6 non-trimmed surfaces. The set of points is extracted from the simulation meshes, which originates from the initial meshing of the hexahedrons. Thus, we do not expect a change of coordinate systems and the registration part of the procedure can be omitted. The distance computations will result in two numbers for each pair of design parameters, minimum and maximum signed distance, so point 4 in the description above is also redundant.

In the current setting, our computation reduces to point 3 above with a smaller point set and a simpler CAD model than initially expected. However, we need to compute signed distances. The closest point computations result in the actual closest point between the surface model and the initial point together with the corresponding parameter value. The sign of the pointwise distance can then be found by comparing the vector between these two points with the surface normal at the parameter value. In the current optimization loop, the distance computations will be performed offline, but for future use, most of the required functionality already runs as a service in the Cloud.

The input is initially given by two ASCII files. The first (.msh file) contains information on the original undistorted mesh points together with the connectivity of the tetrahedral elements. The second (.res) file contains the distortion results as 3D vectors, with the results restricted to the boundary of the mesh. In the first version of the optimization workflow we extract only the final distorted points (as a sum of the original mesh points and the distortions). For our simple example, with relatively small distortions, it is sufficient to run the closest point computations over all surfaces and all points. However, in later versions with more complex examples, and possibly larger distortions, we will consider utilizing other information present in mesh, such as the original undistorted points and neighbourhood information. This can improve both the speed and reliability of the computations, by providing a good initial guess of the closest point parameters, for neighbouring points. If, in future versions, we cannot expect the surface model and the point set to reside in the same coordinate system, a registration step must be introduced. In that case, we will seek to generate an automatic correspondence between certain mesh points and positions in the surface model. Manual alignment of points is not an option in an optimization loop.

To evaluate $\Delta vol(\mathbf{p})$, we need to compute the difference between the volume of the nominal geometry and the volume of the distorted ones. This operation is simple, as each of the two volumes can be computed by summing up the volumes of all tethraedra in the respective meshes.

7 REFERENCES

Bonet, J. W. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis.* Cambridge: Cambridge University Press.

Brezinski, C. R.-Z. (1991). *Extrapolation methods. Theory and Practice.* Amsterdam: North-Holland.

Bungartz, H. J. (2004). Sparse Grids. *Acta Numerica*, 147-269.

CAxMan Consortium. (2016). Deliverable 2.2. *IGA based shape representation, initial version*.

CAxMan Consortium. (2016). Deliverable 2.3. *Analysis tools for AM, linear setting*.

CAxMan Consortium. (2016). Deliverable 2.4. *Shape representation for AM*.

CAxMan Consortium. (2016). Deliverable 3.1. *Requirements: Process planning for AM*.

CAxMan Consortium. (2016). Deliverable 3.2. *AM Process Planning Workflows*.

CAxMan Consortium. (2016). Deliverable 3.3. *First implementation of process planning workflows*.

CAxMan Consortium. (2016). Deliverable 4.2. *Thermal analysis framework for MD process*.

Nocedal, J. W. (2006). *Numerical Optimization.* New York: Springer-Verlag New York.

Pfluger, D. (2010). *Spatially Adaptive Sparse Grids for Higher-Dimensional Problems.* Munchen: Verlag Dr. Hut.

Quarteroni, A. (2008). *Numerical Models for Differential Problems.* Milan: Springer-Verlag Italia.

Valentin, J. P. (2016). Hierarchical Gradient-Based Optimization with B-Splines on Sparse Grids. *Sparse Grids and Applications - Stuttgart 2014* (pp. 315-336). Springer.