Lightweight parametric design optimization for 4D printed parts

Rubén Paz[a,*], Eujin Pei[b], Mario Monzón[a], Fernando Ortega[a] and Luis Suárez[a]
[a]*Departamento de Ingeniería Mecánica, Universidad de Las Palmas de Gran Canaria, 35017 Las Palmas G.C., Spain.*
[b]*Department of Design, College of Engineering, Design and Physical Sciences, Brunel University London, Tower A, TOWA020, UB8 3PH, United Kingdom.*

Abstract. 4D printing is a technology that combines the capabilities of 3D printing with materials that can transform its geometry after being produced (e.g. Shape Memory Polymers). These advanced materials allow shape change by applying different stimulus such as heating. A 4D printed part will usually have 2 different shapes: a programmed shape (before the stimulus is applied), and the original shape (which is recovered once the stimulus has been applied). Lightweight parametric optimization techniques are used to find the best combination of design variables to reduce weight and lower manufacturing costs. However, current optimization techniques available in commercial 3D CAD software are not prepared for optimization of multiple shapes. The fundamental research question is how to optimize a design that will have different shapes with different boundary conditions and requirements. This paper presents a new lightweight parametric optimization method to solve this limitation. The method combines the Latin Hypercube design of experiments, Kriging metamodel and specifically designed genetic algorithms. The optimization strategy was implemented and automated using a CAD software. This method recognizes both shapes of the part as a single design and allows the lightweight parametric optimization to retain the minimum mechanical properties for both shapes.

Keywords: Lightweight design optimization, 4D printing, Shape Memory Polymers (SMPs), Finite Element Analysis (FEA), genetic algorithms (GAs), Kriging.

1. Introduction and problem definition

1.1. Introduction

Smart materials are defined as materials that can either change their shape or properties between different physical domains under the influence of certain stimuli from the environment [17]. Particularly, Shape Memory Materials (SMMs) have the ability to recover their original shape from a deformation when a particular stimulus is applied. This is known as the shape memory effect (SME). SMMs are either inorganic (Shape Memory Alloys, SMAs) or organic (Shape Memory Polymers, SMPs) [10]. The shape recovery is usually activated by the surrounding temperature (both in SMAs and SMPs), and other stimuli that have been used include an electric field, magnetic field, pH, UV light, or specific chemicals, etc. [13].

Additive Manufacturing (AM) is defined as a process of fusing materials layer upon layer to produce a three-dimensional object from CAD data [15]. The continuous evolution of these technologies and materials is expected to revolutionize the manufacturing industry. As conventional 3D printing technology matures, creeping up in the background is Four-Dimensional (4D) Printing [7]. The fourth dimension in 4D printing refers to the ability of material objects to transform its geometry after being produced, thereby providing additional capabilities and offering potential for performance-driven applications [8]. The technology “4D printing” can be summarized as the combination of using the AM process and Shape Memory Polymer (SMP) materials which provides

[*]Corresponding author. E-mail: ruben.paz@ulpgc.es. Phone: +34928459640

far-reaching opportunities that go beyond the potential application of conventional AM parts.

However, the design and optimization of 4D printed parts has not been studied in detail and some advances are needed to boost their use in different applications. Most of the research related to 4D printing has been focused on discovering new materials and suggesting specific applications, but there are no references associated with design optimization tools. In fact, only a few authors have considered the design, most of them focusing on design of alloy-based actuators [29], self-folding sheets with SMAs for origami engineering [12,37] and design optimization for deformation of SMAs [23,24].

Current optimization techniques available in the commercial 3D CAD software with Finite Element Analysis (FEA) tools are not prepared for the new concept of optimization for multiple shapes. For example, a specific 4D printed part may require a certain stiffness to hold a weight in its programmed shape, and releasing the weight when the stimulus is applied to recover its original shape, and finally supporting another load in its original shape that may require another minimum value of stiffness for its correct function. The key research question is how to optimize a design that will have different shapes with different boundary conditions and requirements.

The aim of a lightweight parametric optimization is to minimize the weight and thus reduce time and manufacturing costs. To achieve this, the optimization process must take both shapes into consideration to fulfill the mechanical requirements needed in both shapes. However, this option is not possible with the current commercial CAD-FEA optimization methods.

1.2. Problem definition (general approach)

The objective is to find the best combination of design variables of a parametric design to minimize the weight of a shape memory part produced by AM. The designer must first propose the design variables (parameters) that will change during the optimization process. These design variables are mostly associated with the geometric dimensions or CAD features such as wall thicknesses, bar dimensions, angles, etc. Since any type of CAD feature can be used as a design variable, these will depend on the specific geometry that is required for optimization. On the other hand, the optimal design must fulfill certain mechanical requirements for the boundary conditions related to the programmed shape (before the stimulus is applied), and other mechanical requirements associated with the original shape (once the recovery is completed). These mechanical requirements can be related to different properties such as the maximum stress that the material can withstand in different directions or different failure criteria such as Von Mises, Tresca, etc., the maximum displacement/strain allowed under certain conditions, maximum displacement of a specific point of the part and so on. Therefore, the optimization problem can be summarized as follows:

Minimize	*Weight (VAR1, VAR2, VAR3…)*
Subject to:	*Constraint 1 (programmed shape)*
	Constraint 2 (programmed shape)
	…
	Constraint N-1 (original shape)
	Constraint N (original shape)

The optimal design will reduce the weight but keeping the minimum properties required according to the constraints desired in both shapes of the part.

2. Methodology

This section presents the methodology developed to solve the problem described in the previous section. Section 2.1. explains the overall optimization concept., section 2.2. details how the methodology works with both shapes of the part during the optimization process to fulfill the mechanical requirements needed for each state, section 2.3. presents the strategies of the algorithm implemented to carry out the optimization process, and section 2.4. presents the software description in terms of automation of the methodology and workflow between the different tools used in the optimization process.

2.1. Overall concept

The overall optimization concept is illustrated in Figure 1. A commercial CAD-FEA software (SolidWorks) was used to evaluate the constraints of the optimization process and the weight of the part. The data provided by the CAD-FEA software are stored and used by the optimization algorithm to drive the optimization process.

The first step is to create a parametric design in the CAD software. The designer must define the geometry and the desired design variables to parameterize the model. Once the parameterization is completed, the next step is to set out the appropriate mechanical analysis (FEA) and the outputs of control (mechanical constraints and weight) according to the conditions that the part must withstand.

The next step is to run the optimization algorithm. The algorithm will prompt some input data to carry out the optimization process which is explained further in section 2.5. During the process of the optimization, the algorithm will define the values of the design variables and the design to be simulated. The geometry is updated in the CAD software and subsequently the simulations are carried out using the FEA tool. The results such as the weight and mechanical constraints are stored by the optimization algorithm and used internally to define the subsequent design. This is repeated as a continuous iterative process until an optimal solution is reached. The optimization strategies are explained in section 2.3.

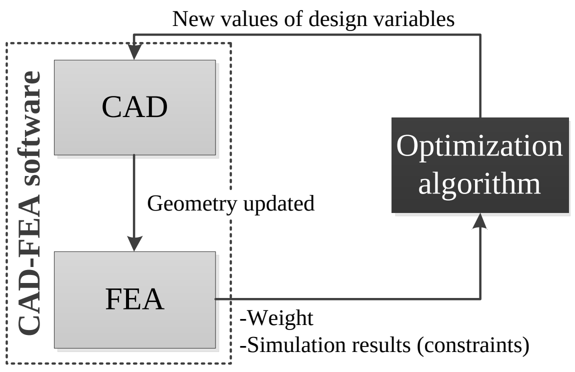

Fig. 1. Overall optimization concept.

2.2. Configurations to address the multiple shapes

In order to evaluate the multiple shapes as different states, the method processes different ‘configurations’ in the CAD model.

Each part has a list of features that define the final geometry, but these features can be activated or suppressed in an independent manner for each of the configurations, which means that the same part may have different shapes depending on the configuration. The use of ‘configurations’ allows better management of different shapes for the same design. The software associates each finite element analysis with only one configuration, and the algorithm assumes that each constraint is related only to one analysis (Figure 2).

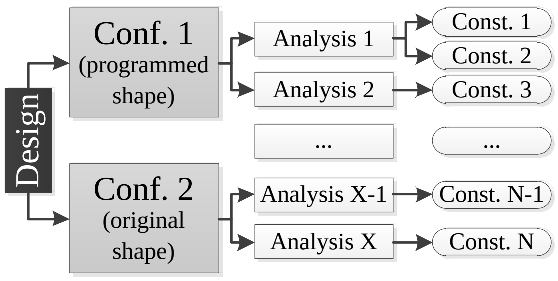

Fig. 2. Relation between different configurations, analysis and constraints in one design.

As 4D printed parts usually have 2 different shapes, configuration 1 is associated with the programmed shape and configuration 2 is associated with the original shape that will be recovered once a stimulus is applied. For each shape, the design must fulfill different constraints that may be related to different boundary conditions (analysis 1, analysis 2, etc.). For each design, the optimization process computes every single finite element analysis that is produced in terms of the programmed and original shape.

Within 3D-CAD modeling, the flex feature is available in the CAD software to differentiate the programmed and original shapes (configuration 1 and 2). This feature allows bending, twisting, tapering or stretching of the shape. For example, if the programmed shape has a ‘L’ profile (configuration 1) and the original shape a straight profile (configuration 2) (Figure 3), a bending feature could be used so that configuration 1 would be activated to achieve the ‘L’ shape, and suppressed in configuration 2 to obtain the straight shape, but always keeping the same values of the design variables.

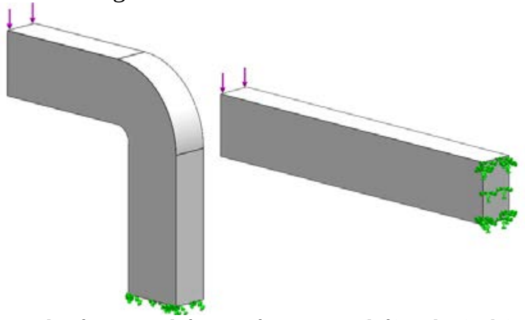
Fig. 3. Flex feature to define configuration 1 (left) and 2 (right).

2.3. Optimization algorithm

The optimization algorithm implemented in this paper is based on genetic algorithms (GAs) and FEA. There are several metaheuristic methods that are usually applied in optimization problems such as GAs

[3,27], Differential Evolution (DE) [41], Particle Swarm Optimization (PSO) [40], Gene Expression Programming (GEP) [38] or Ant Colony Optimization (ACO). These techniques have been applied in different fields, such as shape and topology optimization for structures [1,14,19,20]. On the other hand, although GAs and FEA are widely used for optimization purposes [39], they are commonly combined with metamodels to estimate the results of the FEA and then reducing the optimization time [28,34].

There are several parametric optimization tools available in commercial CAD-FEA software based on this concept, such as Catia (conjugate gradients or simulated annealing) [21], SolidWorks (Box-Behnken Design Of Experiments and the response surface method) [6,18], or ANSYS. This latter software includes different strategies for Design Of Experiments (DOE) such as a central composite or a Box-Behnken design, and also different metamodels such as the response surface method, Kriging [5] or Neural Networks. Among these, only the Kriging metamodel includes the refinement option, which is an interesting tool to enhance the accuracy of the estimations throughout the evolution of the algorithm.

Although none of these tools are capable of processing the optimization of multi-shape designs, the optimization strategies were analyzed to find the best strategies for this application. In this sense, the idea of the metamodel refinement (similar to ANSYS) was considered a key factor to improve the performance of the final methodology. On the other hand, Kriging is commonly used in optimization [22,30,33,42,44] as it is able to provide the best linear unbiased predictions. For these reasons, the Kriging metamodel was selected for this methodology. However, the refinement criterion applied in the optimization algorithm developed in this research was different from the one used in ANSYS). Section 2.3.2. explains these details.

In general terms, the optimization algorithm implemented can be divided into three clear stages: DOE, feasible/unfeasible border approximation, and final optimization.

The aim of the DOE is to simulate several designs of the search space to gather information about the behavior of the part such as the constraints and objective depending on the design variables. According to the tests carried out, the modified version of Latin Hypercube presented in this work was the best DOE as it was the one that allowed the metamodel creation with less sampling points. Section 2.3.1. explains these details.

The second stage (feasible/unfeasible border approximation) is an approach to add more sampling points and to improve the accuracy of the metamodel, focusing on the feasible/unfeasible border to carry out the refinement of the metamodel in an efficient manner. This second stage uses the Kriging method to predict the results and it is driven by specifically designed GAs (proximity penalty concept) to explore new zones along the feasible/unfeasible border. Moreover, the Kriging metamodel is always generated with the highest order possible of the regression model according to the available data to improve the accuracy of the metamodel.

The final stage of optimization uses the data gathered in the previous stages to accomplish an optimization based on the trained Kriging and GAs. Section 2.3.3. explains this stage.

2.3.1. Design of experiments (DOE)

The DOE is a stage in which different designs are simulated by FEA to gather data that will be useful for the metamodel generation. In previous studies [35], a more specific DOE was implemented to focus the sampling on specific areas close to the optimum, thus improving the performance of the metamodel. This DOE was driven by a GA with binary and ternary encoding. This coding was used to provide 2 and 3 level values respectively so that all the designs generated during this DOE had their design values in the minimum, maximum or middle values according to the limits selected. However, the coding of the GAs of other stages of the algorithm was with real numbers as the domain is continuous. This flexibility in terms of encoding was one of the reasons why GAs were used. To successfully create the metamodel with 1-order polynomial regression models, the sampling needed in the DOE is too high for this application where each sampling needed more computational time due to the simulation of both shapes (configuration 1 and 2). Moreover, the accuracy of the metamodel can be improved by using a higher order in the regression model, but this would result in more sampling (more computational time) or using a different DOE with a similar sampling effort but enhanced distribution of sampling points.

For this reason, a different DOE using the Latin Hypercube was implemented. This DOE is commonly combined with Kriging [26,32] because the distribution of data is appropriate for Kriging and consequently the sampling needed to define the metamodel can be reduced. It divides the domain search into different equal spaces and allocates randomly sampling points so that there is only one sampling point in each row and column of the search domain. In an n-dimensional problem, each sample would be the only one in each axis-aligned hyperplane containing it. Figure 4 represents this in a 2D example. If the number of sampling points is 4, then each dimension is divided into 4 equal parts. The Latin Hypercube will first add one random point and then the following 3 sampling points will also be randomly placed but keeping only one sample in each row and column, which guarantees an appropriate distribution of the sampling points all over the search domain.

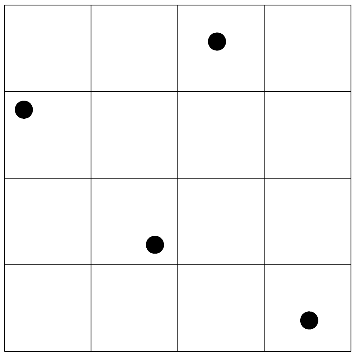
Fig. 4. Latin Hypercube DOE.

Further some modifications were implemented in the DOE of the methodology to improve its performance. First of all, the DOE consists of 2 steps.

Firstly, the minimum, central and maximum points of the search domain are added. This means that all the design variables will have the minimum value, then the middle value, and finally the maximum value. This first stage provides an initial database of the search domain.

Secondly, a Latin Hypercube DOE is applied using a Matlab function [25]. The number of sampling points added is ‘n’, being ‘n’ the number of design variables of the parametric design. Therefore, the sampling effort is proportional to the number of design variables. On the other hand, the Latin Hypercube is applied following the criterion of maximizing the minimum distance between points. After several tests, it was observed that the closer the points were to the border of the search domain, the more successful the Kriging generation and the more accurate the results (because interpolation is more accurate than extrapolation). In this sense, the minimum and maximum points added in the first step are important to reduce the use of extrapolations in favor of interpolations. To take this into account, the Latin Hypercube was modified to move the sampling points to the borders of the search domain. The optimization algorithm identifies the points of the standard Latin Hypercube DOE that are aligned to the border, and modifies the correspondent variables to their maximum or minimum values to place them on the border. Figure 5 shows the modified Latin Hypercube in the same problem depicted in Figure 4. The black points are the final sampling points according to the modified Latin Hypercube. The grey points are the points from the standard Latin Hypercube where the location was modified. For example, the point located in the last row and column was moved to the corner. In this process, only the points placed in ‘squares’ adjacent to the border are modified, and only within the constraints of the corresponding variables. Section 2.3.4. summarizes the improvements of this concept.

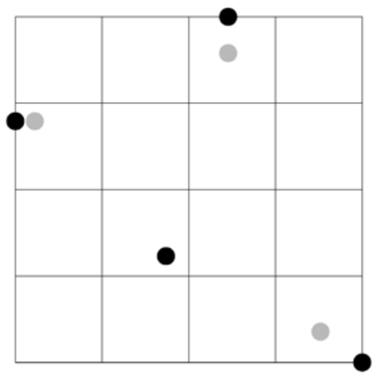
Fig. 5. Modified Latin Hypercube DOE.

2.3.2. Feasible/unfeasible border approximation

The next step in the optimization algorithm is to add more sampling points next to the feasible/unfeasible border. Since the objective (minimum weight) is always opposite to the mechanical constraints (stiffness, stress, etc.) the optimum solution will always be in the feasible/unfeasible border. This stage of the algorithm, through the use of the Kriging metamodel combined with a specific strategy based on genetic algorithms, allows the addition of new sampling points along the feasible/unfeasible border, improving the accuracy of the metamodel in the zones where the optimum points will be located. Taking a step further, we also include a refinement loop that follows a similar idea to the ANSYS metamodel. However, in this case the refinement is focused on the zones close to the location of the optimum (feasible/unfeasible border) and not in the zones with the highest estimation error. This allows the operator to improve the metamodel accuracy only in the zones of interest, thus reducing the sampling and the optimization time.

The first task consists of creating the Kriging metamodel from the data gathered in the previous stage. A Matlab subroutine process is used to create the metamodel [31]. Among the available correlation models (exponential, generalized exponential, Gaussian, linear, spherical and cubic spline), the generalized exponential model was selected as it is the most flexible in terms of shape of the function and commonly used when the spatial correlation between data is unknown (as it happens in this application). Regarding the regression model, the function will be always polynomial. However, there are several orders available (2, 1 and 0-order).

The 2-order regression model can achieve better estimations but requires more sampling or better distribution of the data to be able to create the metamodel. As the order of the regression model lowers, the metamodel can be created more easily (without so many data and poorer distribution) but the accuracy is reduced. For this reason, the optimization algorithm includes a loop that makes an attempt to generate a 2-order regression model. If this fails, then the 1-order regression model is used automatically, and finally the 0-order. This system enables a simpler regression model when the data is minimal and consequently, more complex regression models are produced when there are enough data.

Once the metamodel is created, a genetic algorithm (GA) drives the optimization. The genetic algorithm uses 100 generations with a 100 individual population size. The first population is randomly created. The estimations of the metamodel are used to calculate the fitness value of each individual being proposed during the GA evolution. In this case, the fitness function ('F') includes the weight ('w') of the design and different penalty factors that increase the value to penalize the individual when one or more constraints are not fulfilled (according to the metamodel predictions). This penalty factor ('PF') is calculated as the absolute error between the estimated value of the constraint ('EC') and its limit value ('LV'). This error is multiplied by 10E99 to amplify the penalization and it must be applied in all the constraints that are not fulfilled.

$F = w + PF$

$PF = |EC - LV| \cdot 10E99$

Once the fitness function of each individual is known, the GA applies a tournament selection of 2 individuals, an arithmetic crossover with 50% probability, mutation with 60% probability and 50% of maximum mutation amplitude, reparation and elitism. The arithmetic crossover is applied according to the following expression:

$Children\_1 = \alpha \cdot Parent\_1 + (1-\alpha) \cdot Parent\_2$

$Children\_2 = (1-\alpha) \cdot Parent\_1 + \alpha \cdot Parent\_2$

α = random value between -0.5 and 1.5

The mutation is applied by randomly selecting a gene or the design variable ('DV') of the individual that will suffer a "mutation". Its initial value of the gene or design variable ('DV_0') will be modified by adding a value that will randomly change from -0.5 to 0.5 times the domain of that variable, which is calculated as the difference between the maximum and minimum values of the design variable ('DVmax' and 'DVmin' respectively).

$DV = DV_0 + (DVmax - DVmin) \cdot A$

A = amplitude (random value between -0.5~ 0.5)

When the GA finishes its evolution, the best design is simulated by FEA and the results of weight and constraints are stored to update the metamodel using all the available data. Subsequently, the GA that has been produced earlier is applied again using the updated Kriging. This is repeated in a loop until 'n+1' points have been simulated and added to the database (being 'n' the number of design variables, so that the sampling is proportional to the number of design variables).

In the first 'n' iterations, the GA uses another penalty factor in the fitness function (proximity penalty factor, 'PPF'). The aim of this penalty factor is to penalize those individuals close to points already simulated. If the individual is close to a previously simulated point, then the fitness function will be penalized and consequently, the individual will not survive in the tournament selection. To apply this, the algorithm internally calculates a niche radius or 'radius of influence' ('Ri') that depends on the dimensions of the search domain (equivalent radius, 'Req') and on the number of 'niches' desired in the domain, which was established in '2n'.

$Req = 1/2 \cdot \Sigma^n (DVmax - DVmin)^{1/2}$

$Ri = Req / (2n)^{1/n}$

Once the radius of influence is defined, the algorithm calculates the distance between each individual proposed by the GA and the sampling points that have been added. If the distance between the individual of the GA and any of the sampling points ('Di') is lower than the radius of influence, then the proximity penalty factor is applied according to the following expression:

$PPF = \Sigma^i \ 1/Di \cdot 10E99$

This strategy forces the algorithm to explore new zones of the domain along the feasible/unfeasible border. The inspiration of this idea comes from the resource sharing method used in multimodal optimization [4,9]. Once 'n' points have been added in this stage, the exploration is considered enough and the optimization algorithm stops the application of the proximity penalty to enable more freedom for the optimum search.

In the iteration ‘n+1’, the GA evolves to an optimal design that is then simulated by FEA. Next, the accuracy of the predictions of the metamodel are checked. If the mean absolute percentage error (MAPE) of the estimations compared with the simulation results (weight and constraints) is higher than 5%, then the last point is added to the database and the metamodel is updated to improve its accuracy. Subsequently, the GA is run again. This is repeated in a loop until the MAPE of the estimations of the last point is lower than 5%, which guarantees a minimum accuracy of the metamodel the zones close to the location of the final solution.

2.3.3. Final optimization

In this final stage, the same GA is run again but using the metamodel updated with all the available data and without applying proximity penalty. This GA allows the new optimal solution to be generated according to the metamodel created with all the data gathered so far. The optimal design obtained according to the GA is simulated by FEA, and if it is the best design simulated so far, then it will be the final design. Otherwise, the metamodel is updated with this new point and the GA is run again. This is repeated until the algorithm achieves a design better than the existing solution obtained from the previous stages of the algorithm. However, if more than ‘5+n’ points are added in this stage and at least one of them is a feasible design by fulfilling all of the constraints, then the best design of the previous stages will be considered as the final solution. This condition was added because on some occasions, the algorithm can achieve a very good design in the previous stages that is difficult to improve afterwards.

Figure 6 summarizes the structure of the optimization algorithm. It is noted that stage 2 (Feasible/unfeasible border approximation) and stage 3 (Final optimization) use GAs to select the design that will be simulated by FEA. In the first ‘n’ iterations of stage 2, proximity penalty is applied to explore the feasible/unfeasible border. Therefore, the optimization algorithm draws on GAs applied in different iterative processes and with different purposes.

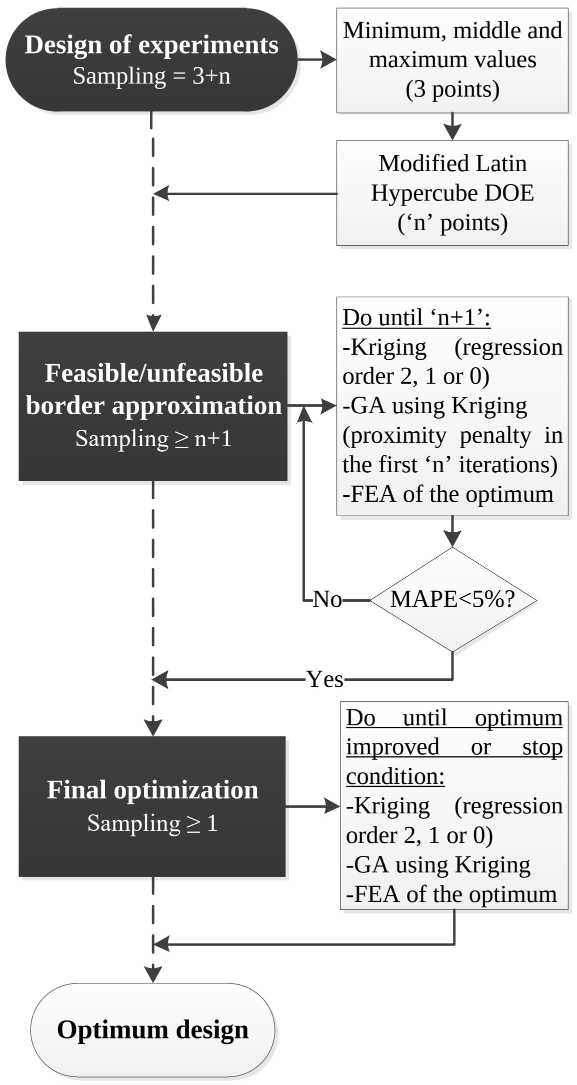
Design of experiments
Sampling = 3+n
Minimum, middle and maximum values (3 points)
Modified Latin Hypercube DOE (‘n’ points)
Feasible/unfeasible border approximation
Sampling ≥ n+1
Do until ‘n+1’:
-Kriging (regression order 2, 1 or 0)
-GA using Kriging (proximity penalty in the first ‘n’ iterations)
-FEA of the optimum
No
MAPE<5%?
Yes
Final optimization
Sampling ≥ 1
Do until optimum improved or stop condition:
-Kriging (regression order 2, 1 or 0)
-GA using Kriging
-FEA of the optimum
Optimum design

Fig. 6. Structure of the optimization algorithm

2.4. Comparison of results between the algorithm presented and previous versions

This section has presented a short summary of the comparison between the proposed algorithm and the previous versions. The final algorithm is the result of several versions that were improved step by step by testing them in different applications. The sampling strategy, refinement and metamodels used were modified according to the conclusions obtained from the results of the tests. On the other hand, although there are many different metaheuristic techniques to optimize, the preliminary tests carried out with GAs found that there were several configuration that obtained similar results and very close to the theoretical optimum. These tests depicted that the configuration of parameters of the GA was not crucial for the performance of the methodology, which meant that some parameters such as the number of generations may be high (conservative values) and the performance of the methodology will not be affected. This is because the time needed by the GA to evolve is not relevant compared with the sampling or metamodel refinement strategy. Therefore, the use of GAs provided enough accuracy and speed, apart from flexibility and robustness to accomplish different ideas to drive the optimization process according to the requirements for each step. The tuning of the GA was carried out by changing several parameters such as the type of penalty factor, the number of generations, the crossover and mutation probability and the mutation amplitude. After multiple tests, although several configurations obtained similar results, the parameters presented in section 2.3.2. were the ones selected.

Although this algorithm is the result of more than 15 versions, only the last ones are presented in this section, starting from a version already tested in previous work [35]. The first modification carried out was to implement a loop to use the highest order of regression model possible in the Kriging metamodel. This idea improved the results in terms of the quality of the optimal solution. The resulting version was selected as a reference to compare with the new versions developed. All the versions were tested in five different case studies with 5, 10, 15, 20 and 25 design variables respectively. Each version was run 15 times to obtain an average value of optimization time and weight of the optimum. We also applied the Mann-Whitney U-test between each version and version 1 to assess if the weight of the optimal solutions were statistically different. The overall purpose was to reduce the sampling to cut down the optimization time and to maintain the quality of the optimal solution.

The first modification that was applied was to reduce the number of sampling points of the DOE. As version 1 used a '3+n+n=3+2n' approach, in version 2, the number of sampling points of the DOE was reduced to '3+n/2+n2=3+n'. However, this reduction led to problems in the metamodel generation in stage 2. In the exploration stage, the algorithm evolved to produce designs that were already simulated and therefore almost the same design was unnecessarily simulated twice.

In order to avoid this, in version 3, the proximity penalty in stage 2 was applied to all the sampling points added before instead of just being applied to the points added in stage 2. The sampling was the same used in the previous version.

In version 4, the radius of influence (niche radius) to apply the proximity penalty was reduced for the sampling points added in the DOE, while the proximity penalty used for the points added in stage 2 was kept. This approach aims to allow more freedom of exploration in stage 2 but avoiding the repetition of sampling points of the DOE.

In version 5, the DOE strategy was changed to use Latin Hypercube. Moreover, the number of sampling points was increased to keep the same number as version 1, which means 3 points and 2n points of the Latin Hypercube DOE (3+2n).

In version 6 the number of sampling points was further reduced to establish the same sampling effort used by versions 2-4. Therefore, in the first stage, the sampling was '3+n'.

In version 7, the number of sampling points of the DOE was again set to '3+2n', but the Latin Hypercube strategy was modified to move the sampling points to the borders of the search domain. The aim was to avoid the extrapolation of results and to use interpolation instead.

Finally, in version 8, the previous approach was applied again but reducing the number of sampling points of the initial stage to '3+n'. Therefore, versions 2, 3, 4, 6 and 8 have the same sampling effort in the first stage ('3+n'), and versions 1, 5 and 7 have additional numbers of sampling points ('3+2n').

Table 1 summarizes the results for each version. The columns show the version, the number of sampling points used in the DOE, percentage increase of optimization time compared with the results obtained with version 1 and the percentage increase of optimum's weight compared with the results of version 1. The values presented are the average values of the 15 runs of each version for the 5 different case studies ($15\cdot5=75$ runs for each version).

- Version 1: DOE based on binary and ternary encoding ('3+2n' points).
- Version 2: '3+n' sampling points in the DOE.
- Version 3: Proximity penalty applied in all the sampling points added before.
- Version 4: Radius of influence reduced in the proximity penalty of sampling points added in the DOE.
- Version 5: DOE based on Latin Hypercube ('3+2n' points).
- Version 6: DOE based on Latin Hypercube ('3+n' points).

- Version 7: DOE based on Latin Hypercube (‘3+n’ points).

Table 1. Comparison between the different versions of optimization algorithm developed

Version	No. points (DOE)	Increase of optimization time	Increase of optimum’s weight
1	3+2n	-	-
2	3+n	-31.0%	1.4%
3	3+n	-29.5%	1.5%
4	3+n	-38.1%	1.3%
5	3+2n	-7.7%	0.3%
6	3+n	-14.5%	0.8%
7	3+2n	-35.4%	1.9%
8	3+n	-40.2%	1.5%

Version 8 (algorithm presented in section 2.3.) achieves an average reduction of 40.2% on the optimization time compared with version 1, while the quality of the optimum is worsened by 1.5%. It can be observed that the use of Latin Hypercube improved the results compared with version 1. The modification proposed to the Latin Hypercube significantly enhanced the results in terms of the optimization time compared to the standard Latin Hypercube (around 25 points). However, the quality of the solution is worsened by 1.6 points.

2.5. Software description

The optimization algorithm with the strategies described before was implemented in the Application Programming Interface (API) of the CAD-FEA software using Visual Basic. Through this tool, the optimization is driven in a completely automated manner, which means that the multiple geometry updating and FEA simulations needed for the optimization process are automatically generated once the code is run and the input data is established.

Some specific tasks that are carried out by the Matlab Windows Application are automatically activated during the optimization process. The API sends the available data to Matlab such as the design variables, constraints and weight of all the points simulated to create the Kriging metamodel and to calculate the predictions. These results are sent back to the API and used for the fitness function evaluation in the GAs. Apart from the creation of the metamodel and calculation of the predictions, Matlab is also used to sort the data in the GAs as well as to save the final results and create some graphics that summarize the evolution of the optimization process. Figure 7 represents the overall workflow.

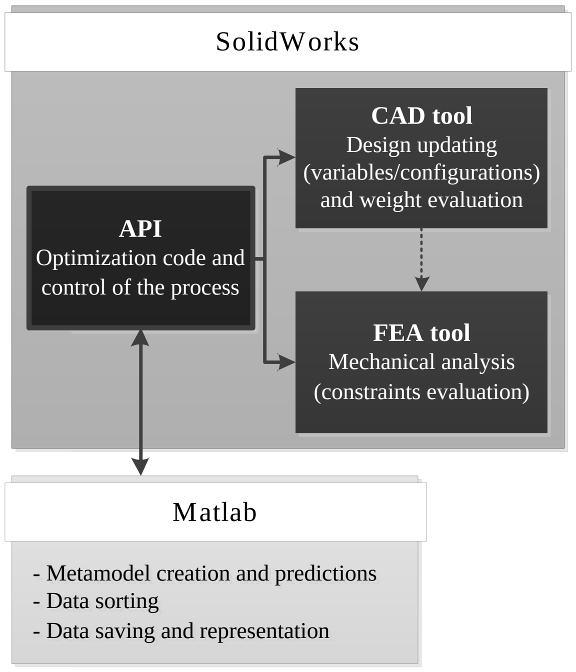
SolidWorks
CAD tool
Design updating (variables/configurations) and weight evaluation
API
Optimization code and control of the process
FEA tool
Mechanical analysis (constraints evaluation)
Matlab
- Metamodel creation and predictions
- Data sorting
- Data saving and representation

Fig. 7. Workflow of the optimization process.

The sequence to apply the optimization method is as follows (Figure 8):

- Parameterization of the geometry: Definition of the geometry in the CAD software, including the design variables. The design variables must be linked to the global variables with the right nomenclature (‘VAR1’, ‘VAR2’, etc.). The global variables must be defined in the equation manager of the CAD software in the right order and before the rest of equations. The API will access the equation manager to change the value of the variables, which are linked to the corresponding dimensions of the part.
- Definition of the 2 configurations of the geometry: The programmed and original shapes must be defined using two different configurations that must be named with integers. One configuration will have the flex feature ‘suppressed’ and the other being ‘activated’.
- Definition of the mechanical analysis related to each configuration: Definition of the boundary conditions (loads, constraints, contacts, materials, etc.) of the analysis associated with each configuration. The name of the analysis must follow a certain convention (Analysis 1, Analysis 2, etc.) to guarantee the workflow between the API and the FEA tool.

- Definition of sensors to get the relevant data from the FEA results: Definition of the sensors to store the FEA data is needed for the optimization process (constraints and objective). The sensors must be named also following certain conventions (Constraint 1, Constraint 2… and Mass).
- Running of the optimization program: Once the optimization is run, the program prompts some input data such as the number of design variables, the lower and upper limit values of each variable, the number of FEAs, the number of constraints, limit values, feasible zones and associated analysis of each constraint, the maximum element size for the mesh of each analysis and finally the configuration number related to each analysis.
- Final result: The optimization algorithm automatically changes the geometry, accomplishes the FEA simulations and manages the dataflow between the CAD, FEA and API tools and Matlab. Once the optimization is finished, the CAD shows the optimal design on the screen.

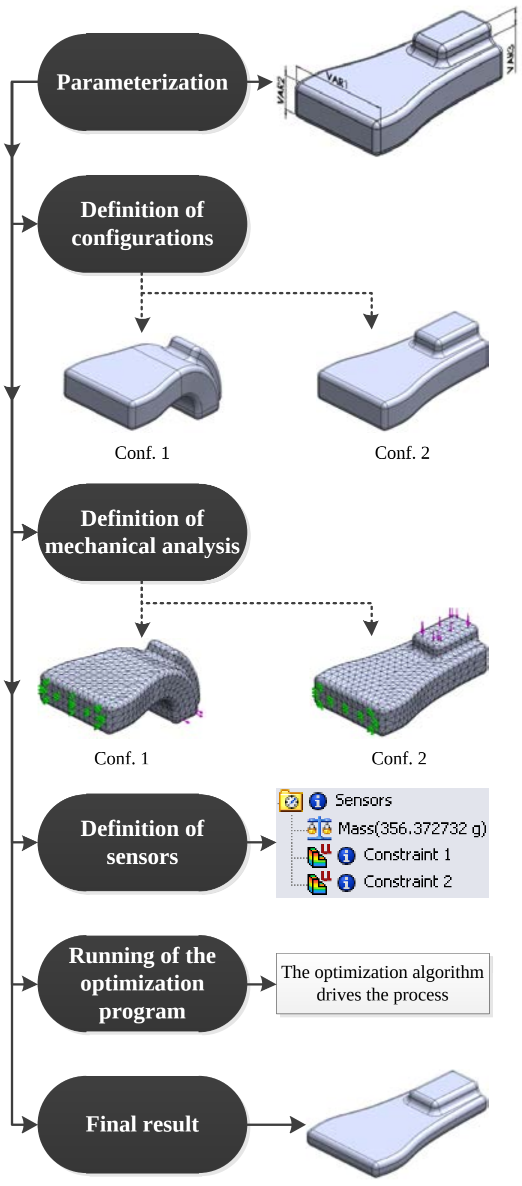
Parameterization
VAR1
VAR2
VAR3
Definition of configurations
Conf. 1
Conf. 2
Definition of mechanical analysis
Conf. 1
Conf. 2
Definition of sensors
Sensors
Mass(356.372732 g)
Constraint 1
Constraint 2
Running of the optimization program
The optimization algorithm drives the process
Final result

Fig. 8. Sequence to apply the optimization method.

3. Case study

This section presents a simple case study in which the new optimization algorithm was applied.

3.1. Geometry and design variables

The geometry of this case study is a simple parallelepiped (a prism whose external faces are all parallelograms) that will have a programmed ‘L’ shape (configuration 1) and then will be stimulated to recover its original straight shape (configuration 2). The part will have a hollow geometry with an internal longitudinal wall (Figure 9). The geometry will be longitudinally symmetrical. The thickness of the external walls will be homogenous (the same thickness in all the points), while the thickness of the central wall will change linearly. A total of 8 design variables (parameters) were defined to control the dimensions of the part, the external walls of the part (5 variables) and the dimensions of the internal wall (3 design variables). These dimensions will be the parameters to be modified during the optimization process. The global dimensions were fixed (90x10x8mm). The center of the part, which is the zone where the shape memory will be applied, was kept solid (11mm length), as well as the end where the part will be fixed for the mechanical tests (10mm length).

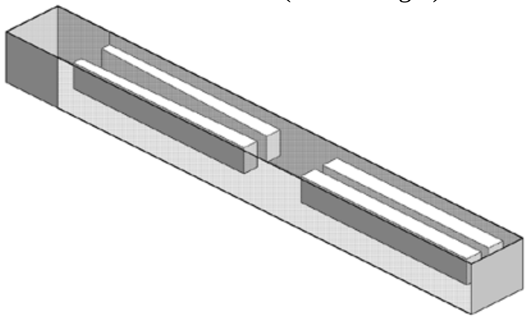
Fig. 9. Geometry to be optimized (internal wall and 4 hollows).

The design variables and limits must be set out according to the manufacturing capabilities, so that any design of the search domain can be manufactured. This is a main advantage of the parametric optimization. The designer can adapt the limits of the parameters according to the manufacturing limitations.

In this case, the design variables and limits (in mm) were defined as follows:

- VAR1: upper thickness [0.8-3.5]
- VAR2: lower thickness [0.8-3.5]
- VAR3: lateral thickness [1.5-2.5]
- VAR4: thickness in the fixed end [1.5-5]
- VAR5: thickness in the end of the load [1.5-5]
- VAR6: half of the thickness of the reinforcing wall in the fixed end [0.75-2] (symmetry applied)
- VAR7: half of the thickness of the reinforcing wall in the middle of the part [0.75-2] (symmetry applied)
- VAR8: half of the thickness of the reinforcing wall in the end of the load [0.75-2] (symmetry applied)

The parameterization is carried out by first defining VAR1, VAR2, etc. as global variables in the equation manager (and in the correct order), and then link the dimensions we want to parameterize with the associated global variable.

3.2. Material

The material used was Polylactic Acid (PLA) for 3D printing, which has shape memory properties. The material was characterized using standard flexural tests applied in 3D printed samples. These samples were produced with the same manufacturing parameters described in section 3.5. Therefore, imperfections related to manufacturing are considered through the appropriate definition of the flexural modulus. According to the experimental results, the flexural modulus was 2950.83MPa. The Poisson’s ratio was 0.36 [16]. The density of PLA was established according to measurement of several 3D printed samples (1.176g/cm^3). These values were introduced in the FEA tool to define the material properties.

3.3. Objective and constraints

The part must support a load of 160N (applied at 2.5mm from the border) in its L-shape state (configuration 1) with a maximum deflection of 8.5mm (constraint 1). Once the original shape (configuration 2) is recovered, the part must support a maximum load of 40N keeping the deflection under 6.5mm (constraint 2). Symmetry was applied in order to simplify the FEA simulations. No penetration contact was established between the punch and the part. Figure 10 shows the boundary conditions of configuration 1 (programmed shape). Figure 11 depicts the analysis of configuration 2 (shape recovered after the stimulus application). The mesh was defined by using a curvature-based mesher with parabolic tetrahedral solid elements. The maximum and minimum element size was 2mm and 0.4mm respectively.

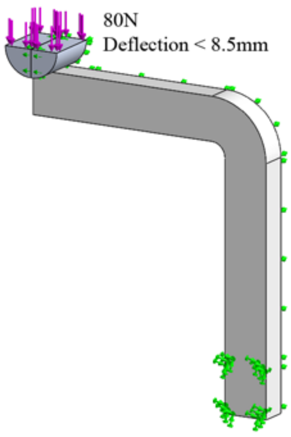

Fig. 10. FEA of configuration 1 ('L' shape).

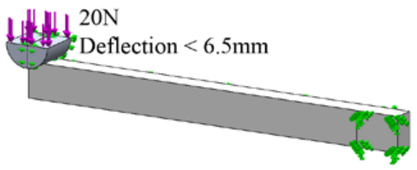

Fig. 11. FEA of configuration 2 (original shape).

The optimization problem of this case study can be summarized as follows:

Minimize Weight (VAR1, VAR2, …, VAR8)
Subject to: Const.1<8.5mm (programmed shape)
Constraint 2<6.5mm (original shape)

As the mathematical formulas of the objective and constraints are not known, the weight and constraints of each design are obtained from the CAD model and FEA results respectively. This can lead to high CPU time. However, the proposed methodology takes advantage of the Kriging metamodel to evaluate the values of the weight and constraints from the available data without changing the geometry or accomplishing the FEA simulations.

3.4. Optimization and results

The optimization was run on a Dell Precision T3500 CPU model with an Intel(R) Xeon(R) W3530-2.80GHz processor and 6GB RAM. The optimum solution was obtained in 102 minutes with a total number of 23 designs evaluated. The computational time needed in the DOE was 48 minutes (11 sampling points). In the second stage (feasible/unfeasible border approximation), the time needed was 41 minutes, with 9 sampling points. Finally, the last stage took 13 minutes, with 3 new sampling points. The optimal design was the last one, with a total mass of 3.448g (real weight of 2x3.448g due to the symmetry applied). The deflection with the programmed 'L' shape (configuration 1) was 8.498mm (constraint 1), while the limit value was 8.5mm. The deflection with the straight shape (original shape after recovery, configuration 2) was 5.952mm (constraint 2). However, the limit value for this constraint was 6.5mm, which means that constraint 1 was more restrictive. As expected, the optimum fulfills all the constraints established in the problem definition. Table 2 shows the values of constraint 1, constraint 2 and mass obtained for each design simulated during the optimization process. All the designs that did not fulfill constraint 2, also did not fulfill constraint 1, which verifies the previous statement (constraint 1 was more restrictive).

Table 2. Values of the constraints and mass of the designs evaluated during the optimization process.

Design no.	Constraint 1 (mm) (<8.5)	Constraint 2 (mm) (<6.5)	Mass (g)
1	9.542	7.056	2.851
2	7.909	5.481	3.769
3	7.731	5.344	4.174
4	8.285	5.787	3.652
5	8.002	5.555	3.717
6	7.988	5.582	3.813
7	7.750	5.355	4.099
8	8.155	5.755	3.701
9	7.791	5.386	3.938
10	8.059	5.606	3.706
11	7.885	5.457	3.710
12	9.162	6.432	3.050
13	9.464	6.829	2.978
14	9.513	6.989	2.925
15	8.588	6.147	3.417
16	8.788	6.125	3.454
17	8.369	5.916	3.560
18	8.521	6.099	3.424
19	8.557	6.194	3.411
20	8.601	6.264	3.384
21	8.548	6.137	3.423
22	8.600	6.250	3.474
23	8.498	5.952	3.448

The design variables of the optimum are shown in Table 3.

Table 3. Values of the design variables of the optimum design.

VAR1 (mm)	VAR2 (mm)	VAR3 (mm)	VAR4 (mm)	VAR5 (mm)	VAR6 (mm)	VAR7 (mm)	VAR8 (mm)
0.806	1.566	1.743	5.000	1.582	1.973	0.750	2.000

Figure 12 shows the relative value of the constraints and mass of the designs evaluated in the last 2 stages of the algorithm (points 12-23). The relative values of the constraints were calculated dividing the value obtained by the associated limit value. Therefore, the values of constraint 1 were divided by 8.5 and the values of constraint 2 by 6.5. In the case of the mass, the reference value is the mass of the optimum. Consequently, the mass values were divided by 3.448.

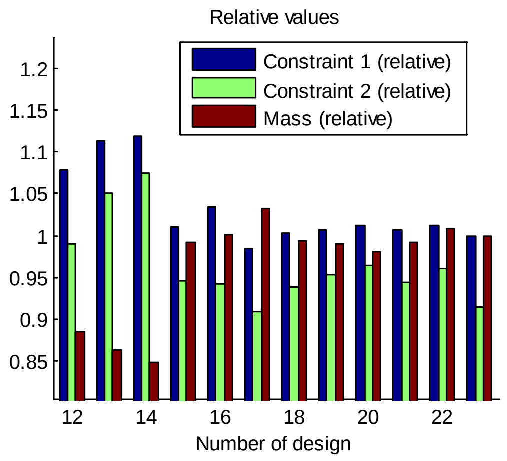

Fig. 12. Relative values of the designs 12-23.

From Figure 12, it can be observed that constraint 1, which was the most restrictive, tends to show values close to 1 as the algorithm evolves. At the same time, constraint 2 tends to show values lower than 1. Therefore, the algorithm evolves to the design with the lower mass that fulfills the constraints.

Figure 13 shows the results (displacements) of the FEA simulation of the optimum design in its programmed shape (configuration 1). Figure 14 shows the displacement results of the optimum in its original shape (configuration 2).

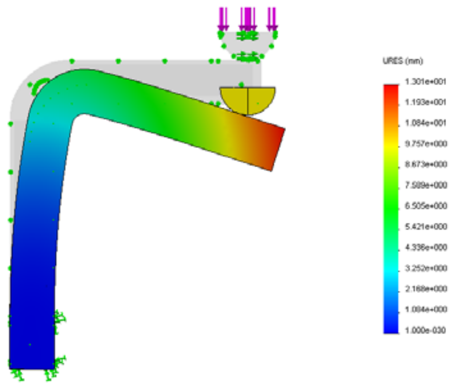

Fig. 13. Displacements of the optimum (configuration 1).

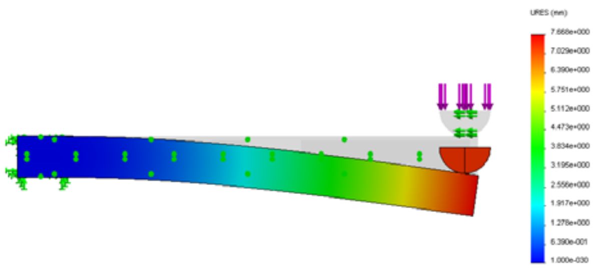

Fig. 4 Displacements of the optimum (configuration 2).

3.5. Manufacturing

The optimum design was manufactured in a BQ Prusa i3 extrusion-based 3D printer. The filament diameter was 1.75mm, the nozzle diameter 0.4mm, the temperature for the layer deposition was 220°C (225°C for the first layer), the layer thickness 0.4mm, 3 perimeters in the contour, 3 solid layers at the top and bottom, 100% fill density and manufacturing speed of 40mm/s in the perimeter and 50mm/s in the infill (travel at 80mm/s).

3.6. Experimental tests

The part was heated at 65°C to deform it and obtain the ‘L’ shape of configuration 1. Subsequently, the part was tested (Figure 15). The results of the test of configuration 1 are shown in Figure 16.

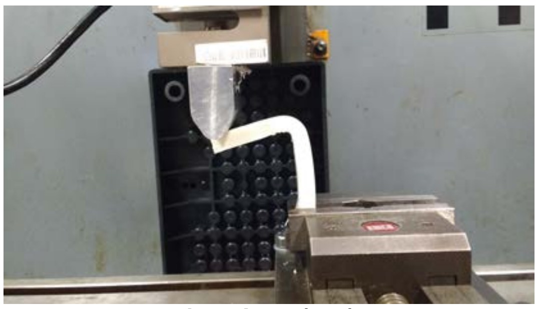

Fig. 15. Mechanical test of configuration 1.

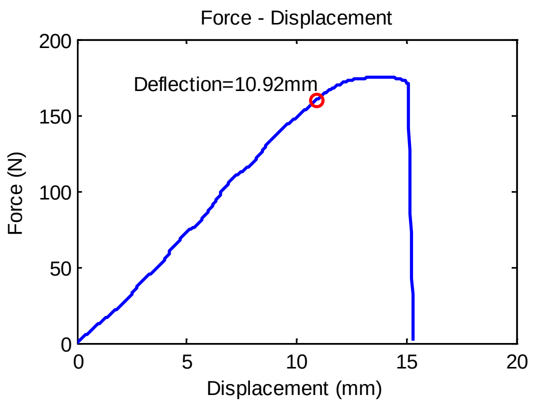

Fig. 16. Results of the test of configuration 1.

The deflection obtained for 160N load was 10.92mm. However, the result according to the FEA simulations was 8.5mm. This difference was due to a loss of the material property of the PLA component during the programming of the 'L' shape when it encountered a large deformation.

Next, the part was heated again to recover its original straight shape (configuration 2) and it was tested (Figure 17). The results of this test are shown in Figure 18.

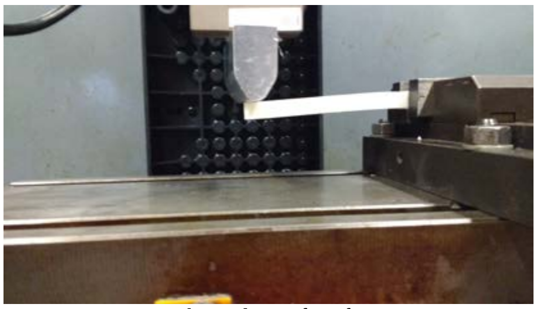
Fig. 17. Mechanical test of configuration 2.

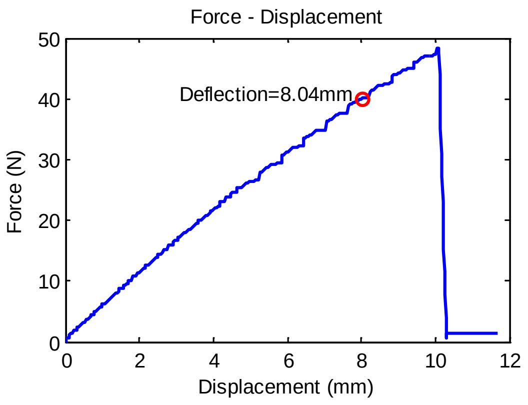

Fig. 18. Results of the test of configuration 2.

The deflection obtained for 40N load was 8.04mm, while the result according to the FEA simulations was 5.94mm. After several subsequent tests with standard flexural samples, it was observed that the main cause of this difference was the loss of material property during the programming stage. The large displacements and strains applied in the L-shape programming led to delamination process (Figure 19), reducing the properties of the sample. Further, tests revealed that the loss of stiffness was around 20% after the shape programming was applied. On the other hand, it was observed that the ratio between the displacement in configuration 1 and 2 was 1.4, for both simulations and experimental results, which confirmed that the difference between the experimental and simulation results is related to this loss of property. Therefore, if the material definition is properly fitted according to the real properties of the programmed material, the proposed optimization algorithm can be successfully applied to optimize the weight of the 4D part.

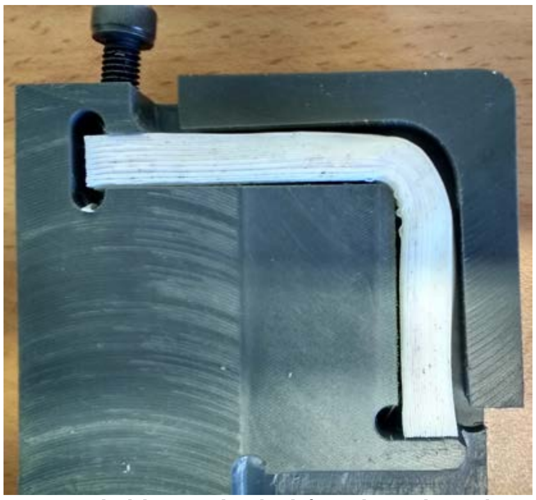
Fig. 19. Sample delaminated in the deformed zone during the programming stage.

4. Conclusions

The design methodology proposes a novel lightweight parametric optimization of shape memory parts taking into account both shapes of the design (programmed and original shapes). This tool, which was implemented in a commercial 3D CAD and FEA software, can be effectively used to optimize the weight and consequentially reduce material and manufacturing costs, but yet ensuring that the minimal mechanical requirements is maintained for both states of the shape memory part (programmed and original shapes). Through the use of this algorithm, users will be able to optimize a design that has different shapes with different boundary conditions and requirements to fulfill the mechanical requirements needed in both instances. This approach of optimization was developed further by automating the work flow which would encourage and promoted the uptake of 4D printing.

While this paper focuses on 2 different configurations, the algorithm is capable of handling many more configurations as desired. This tool can be very powerful to model several situations involved in the shape recovery. For example, a third configuration could be defined with an appropriate configuration and associated Finite Element Analysis conducted to examine whether the part is able to recover under certain boundary conditions. Therefore, this optimization algorithm optimizes the design variables according to the mechanical requirements needed in the programmed and original shapes, and also optimizes the design variables to achieve the shape recovery force needed during the recovery process. If the recovery process is to be analyzed in detail, it can be discretized in different steps related to different configurations, each of which represents a different situation/shape during the recovery. On the other hand, some shape memory materials can retain 3 different shapes and a third configuration could also be very useful. Further research will be carried out to investigate more complex geometries and bending methods. In the case of very complex 4D products, the computer processing time may be too high. As multi-core CPUs and GPUs become more available with higher performance and lower cost, this methodology could be accelerated [2,11,43]. Although SolidWorks was used due to the availability of the flex feature and the configurations module, more advanced FEA software such as Abaqus could also be investigated to allow the simulation of more complex parts and include acceleration techniques.

This proposed methodology can be applied in many different sectors such as medical, automotive, toy, furniture, interior design, textile or telecommunications. The capabilities of 4D printing combined with this methodology have the potential to become useful in areas of high-value engineering [36].

References

[1] Adeli H, Balasubramanyam KV. A synergic man-machine approach to shape optimization of structures. Comput Struct. 1988;30(3):553–61.

[2] Belloch JA, Gonzalez A, Martínez-Zaldívar FJ, Vidal AM. Multichannel massive audio processing for a generalized crosstalk cancellation and equalization application using GPUs. Integr Comput-Aided Eng. 2013 Jan 1;20(2):169–82.

[3] Bolourchi A, Masri SF, Aldraihem OJ. Studies into Computational Intelligence and Evolutionary Approaches for Model-Free Identification of Hysteretic Systems. Comput-Aided Civ Infrastruct Eng. 2015;30(5):330–346.

[4] Della Cioppa A, De Stefano C, Marcelli A. On the role of population size and niche radius in fitness sharing. IEEE Trans Evol Comput. 2004 Dec;8(6):580–92.

[5] Dressler M. Art of Surface Interpolation [Internet]. [Kunštát]: Technical University of Liberec, Faculty of Mechatronics and Interdisciplinary Engineering Studies; 2009 [cited 2014 Dec 29]. Available from: http://m.dressler.sweb.cz/AOSIM.pdf

[6] Eriksson L. Design of Experiments: Principles and Applications. MKS Umetrics AB; 2008. 486 p.

[7] Eujin Pei. 4D Printing: dawn of an emerging technology cycle. Assem Autom. 2014 Sep 9;34(4):310–4.

[8] Garrett TAC Skylar Tibbits, and Banning. The Next Wave: 4D Printing [Internet]. Atlantic Council. [cited 2016 May 18]. Available from: http://www.atlanticcouncil.org/publications/reports/the-next-wave-4d-printing-and-programming-the-material-world

[9] Glibovets NN, Gulayeva NM. A review of niching genetic algorithms for multimodal function optimization. Cybern Syst Anal. 2013;49(6):815–20.

[10] Gupta V, Yadav N, Gupta S. Shape Memory Materials: An Innovative Way to Improve Properties of Cotton. PARIPEX-Indian J Res. 2014 Jun 1;III(VI):293–6.

[11] Guthier B, Kopf S, Wichtlhuber M, Effelsberg W. Parallel Implementation of a Real-time High Dynamic Range Video System. Integr Comput-Aided Eng. 2014 Apr;21(2):189–202.

[12] Hartl D, Lane K, Malak R. Computational design of a reconfigurable origami space structure incorporating shape memory alloy thin films. In: ASME 2012 Conference on Smart Materials, Adaptive Structures and Intelligent Systems [Internet]. American Society of Mechanical Engineers; 2012 [cited 2016 Jul 4]. p. 277–285. Available from: http://proceedings.asmedigitalcollection.asme.org/data/Conferences/ASMEP/75654/277_1.pdf

[13] Huang WM, Ding Z, Wang CC, Wei J, Zhao Y, Purnawali H. Shape memory materials. Mater Today. 2010 Jul;13(7–8):54–61.

[14] Interactive Layout Optimization of Trusses. J Comput Civ Eng [Internet]. 1061 Issue: object: doi: . /jccee5.1987.1.issue-3, revision: rev:1479256513100-15515:doi:10.1061/jccee5.1987.1.issue-3 [cited 2017 Jan 13];1(3). Available from: http://ascelibrary.org/doi/abs/10.1061/%28ASCE%290887-3801%281987%291%3A3%28183%29

[15] ISO/ASTM 52900:2015 - Additive manufacturing -- General principles -- Terminology [Internet]. ISO. [cited 2016 May 18]. Available from:

http://www.iso.org/iso/catalogue_detail.htm?csnumber =69669

[16] Jamshidian M, Tehrany EA, Imran M, Jacquot M, Desobry S. Poly-Lactic Acid: Production, Applications, Nanocomposites, and Release Studies. Compr Rev Food Sci Food Saf. 2010 Sep 1;9(5):552–71.

[17] Khoo ZX, Teoh JEM, Liu Y, Chua CK, Yang S, An J, et al. 3D printing of smart materials: A review on recent progresses in 4D printing. Virtual Phys Prototyp. 2015 Jul 3;10(3):103–22.

[18] Kleijnen JPC. Design of Experiments: Overview [Internet]. Rochester, NY: Social Science Research Network; 2008 Aug [cited 2014 May 6]. Report No.: ID 1262179. Available from: http://papers.ssrn.com/abstract=1262179

[19] Kociecki M, Adeli H. Shape optimization of free-form steel space-frame roof structures with complex geometries using evolutionary computing. Eng Appl Artif Intell. 2015 Feb;38:168–82.

[20] Kociecki M, Adeli H. Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures. Eng Appl Artif Intell. 2014 Jun;32:218–27.

[21] König O, Wintermantel M. CAD-based evolutionary design optimization with CATIA V5. Proc 1st Weimar Optim Stoch Days WOST Weimar. 2004;1–30.

[22] Kunakote T, Bureerat S. Surrogate-Assisted Multi-objective Evolutionary Algorithms for Structural Shape and Sizing Optimisation. Math Probl Eng. 2013 Jul 10;2013:e695172.

[23] Langelaar M, Keulen F van. Modeling of shape memory alloy shells for design optimization. Comput Struct. 2008 May;86(9):955–63.

[24] Langelaar M, van Keulen F. Design optimization of shape memory alloy active structures using the R-phase transformation. In: The 14th International Symposium on: Smart Structures and Materials & Nondestructive Evaluation and Health Monitoring [Internet]. International Society for Optics and Photonics; 2007 [cited 2016 Jul 4]. p. 65250W–65250W. Available from: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1300885

[25] Latin hypercube sample - MATLAB lhsdesign - MathWorks España [Internet]. [cited 2015 Jul 24]. Available from: http://es.mathworks.com/help/stats/lhsdesign.html

[26] Laurent L, Boucard P-A, Soulier B. Generation of a cokriging metamodel using a multiparametric strategy. Comput Mech. 2012 Apr 26;51(2):151–69.

[27] Lee H-G, Yi C-Y, Lee D-E, Arditi D. An Advanced Stochastic Time-Cost Tradeoff Analysis Based on a CPM-Guided Genetic Algorithm. Comput-Aided Civ Infrastruct Eng. 2015;30(10):824–842.

[28] Lee J-H, Hwang S-C, Park JH, Lee K-H. Structural Design Examples Using Metamodel-based Approximation Model. In: Proceedings of the 9th WSEAS International Conference on Applied Computer and Applied Computational Science [Internet]. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS); 2010 [cited 2014 Mar 22]. p. 153–156. (ACACOS'10). Available from: http://dl.acm.org/citation.cfm?id=1844074.1844101

[29] Liang C, Rogers CA. Design of Shape Memory Alloy Actuators. J Mech Des. 1992 Jun 1;114(2):223–30.

[30] Liu H. Taylor kriging metamodeling for simulation interpolation, sensitivity analysis and optimization. 2009 [cited 2014 Mar 21]; Available from: http://etd.auburn.edu/etd/handle/10415/1621

[31] Lophaven SN, Nielsen HB, Søndergaard J. A Matlab Kriging Toolbox. Tech Univ Den Kongens Lyngby Tech Rep No IMM-TR-2002-12 [Internet]. 2002 [cited 2014 Jul 18]; Available from: http://www.imm.dtu.dk/~hbn/dace/dace.pdf

[32] Okobiah O, Mohanty S, Kougianos E. Fast Design Optimization Through Simple Kriging Metamodeling: A Sense Amplifier Case Study. IEEE Trans Very Large Scale Integr VLSI Syst. 2014 Apr;22(4):932–7.

[33] Pan F, Zhu P. Design optimisation of vehicle roof structures: benefits of using multiple surrogates. Int J Crashworthiness. 2011;16(1):85–95.

[34] Pant S, Limbert G, Curzen NP, Bressloff NW. Multi-objective design optimisation of coronary stents. Biomaterials. 2011 Nov;32(31):7755–73.

[35] Paz R, Monzón MD, Benítez AN, González B. New lightweight optimisation method applied in parts made by selective laser sintering and Polyjet technologies. Int J Comput Integr Manuf. 2015 Jul 9;0(0):1–11.

[36] Pei E. 4D printing – revolution or fad? Assem Autom. 2014 Apr 1;34(2):123–7.

[37] Peraza-Hernandez E, Hartl D, Galvan E, Malak R. Design and optimization of a shape memory alloy-based self-folding sheet. J Mech Des. 2013;135(11):111007.

[38] Rashidi S, Ranjitkar P. Bus dwell time modeling using gene expression programming. Comput-Aided Civ Infrastruct Eng. 2015;30(6):478–489.

[39] Roman Gatzi MU. Structural Optimization Tool using Genetic Algorithms and Ansys. 2000;

[40] Shabbir F, Omenzetter P. Particle swarm optimization with sequential niche technique for dynamic finite element model updating. Comput-Aided Civ Infrastruct Eng. 2015;30(5):359–375.

[41] Vincenzi L, Savoia M. Coupling response surface and differential evolution for parameter identification problems. Comput-Aided Civ Infrastruct Eng. 2015;30(5):376–393.

[42] Zhao L, Choi KK, Lee I. Metamodeling Method Using Dynamic Kriging for Design Optimization. AIAA J. 2011;49(9):2034–46.

[43] Zhou Y, He F, Qiu Y. Optimization of parallel iterated local search algorithms on graphics processing unit. J Supercomput. 2016 Jun 1;72(6):2394–416.

[44] Zhu P, Zhang Y, Chen G-L. Metamodel-based lightweight design of an automotive front-body structure using robust optimization. Proc Inst Mech Eng Part J Automob Eng. 2009 Sep 1;223(9):1133–47.