EXTENDED ABSTRACT
Coupling optimisation algorithms to Finite Element Methods (FEM) is a very promising way to achieve optimal metal forming processes. However, many optimisation algorithms exist and it is not clear which of these algorithms to use. This paper investigates the sensitivity of a Sequential Approximate Optimisation algorithm (SAO) proposed in [1-4] to an increasing number of design variables and compares it with two other algorithms: an Evolutionary Strategy (ES) and an Evolutionary version of the SAO (ESAO). In addition, it observes the influence of different Designs Of Experiments used with the SAO. It is concluded that the SAO is very capable and efficient and its combination with an ES is not beneficial. Moreover, the use of SAO with Fractional Factorial Design is the most efficient method, rather than Full Factorial Design as proposed in [1-4].

INTRODUCTION
During the last decades, Finite Element (FEM) simulations of metal forming processes have become important tools for designing feasible production processes. More recently, several authors recognised the potential of coupling FEM simulations to mathematical optimisation algorithms to design optimal metal forming processes instead of only feasible ones. However, many optimisation algorithms exist and it is not clear which of these algorithms to use.

The present research is based on the Sequential Approximate Optimisation (SAO) algorithm which uses both Response Surface Methodology (RSM) and Design and Analysis of Computer Experiments (DACE). The efficiency of this algorithm decreases with increasing number of design variables because of the high number of FEM simulations required. To overcome such a problem two solutions have been investigated.

The first one is to use an Evolutionary Strategy (ES), which is good to find the global optimum and is able to deal with a higher number of design variables. Unfortunately, it requires many function evaluations. Behind this idea was to create an algorithm called ESAO, which is an Evolutionary version of the SAO, to keep the benefit of the ES while using the metamodels of the SAO to avoid too many function evaluations.

The second solution consists in comparing different Designs Of Experiments (DOE) for a fixed number of design variables of the SAO. The DOE used are: Full Factorial Design, Fractional Factorial Design or Latin Hypercube Design.

In the first part of this paper, the three algorithms will be briefly explained: SAO, ES and ESAO. In Section II the sensitivity of the three algorithms to an increasing number of design variables taken into account in optimisation problems is investigated. Section III is dedicated to a comparison between the different DOE’s and conclusions are presented in Section IV.
I THE OPTIMISATION ALGORITHMS

In this first part the three optimisation techniques used in the present research are briefly described.

I-1. Sequential Approximate Optimisation algorithm (SAO)

The Sequential Approximate Optimisation algorithm uses both Response Surface Methodology (RSM) and Design and Analysis of Computer Experiments (DACE) metamodelling techniques [1-4]. SAO is summarised in Fig. 1.

The first step, modelling, consists of carefully choosing the design variables, constraints, and objective function.

Then it uses a Design Of Experiment (DOE) strategy to define starting points. The two designs combined in the SAO are Latin Hypercube Design (LHD), defining points in a spacefilling way, and Full Factorial Design (FFD) selecting all the corner points of the domain as illustrated in Fig. 2.

The third step is running the FEM simulations for each of the DOE points. Once the objective function values are known for each point, the algorithm fits metamodels based on polynomial regression (RSM) or Kriging interpolation (DACE). All of these metamodels are then validated using cross validation before the best one is selected. This final metamodel is used to run a multistart SQP algorithm. This is the optimisation step.

Finally, if the evaluation of the optimum, i.e. FEM calculation of the optimal design variables found, is not satisfying we can improve the metamodel by adding DOE points to the previous ones. This is the sequential improvement.

The SAO algorithm has been implemented in MATLAB [1-4] and can be used in combination with any FEM code. Furthermore, the fitting of the DACE/Kriging metamodels uses the MATLAB Kriging toolbox implemented by Lophaven, Nielsen and Søndergaard [5].

I-2. Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)

The employed ES has been developed to solve optimisation problems with a nonlinear objective function. It is summarised in Fig. 3. In general, the method should be applied if derivative based methods, e.g. quasi-Newton BFGS or conjugate gradient, (supposedly) fail due to a rugged search landscape (e.g. discontinuities, sharp bends and ridges, noise, local optima, outliers).

The CMA-ES has been developed in such a way that we almost have no parameters to define. Basically one just needs to choose the size of the population λ.

When the algorithm starts, a population of new search points (λ individuals) is generated by sampling a multivariate normal distribution based on the covariance matrix. Normal distribution is used for many reasons; one of them is that it implies invariance (does not favour any direction). The normal distribution is based on the eigenvectors and deviations of the covariance matrix in order to introduce mutation in the new generation.
Once this population is created, it evaluates the $\lambda$ objective function values and selects the best $\mu$ points. These $\mu$ parents are recombined using weighted recombination. In order to update the covariance matrix, a step size control has been introduced that prevents premature convergence. The last step before sampling the next generation is the covariance matrix adaptation that increases the probability of successful steps to appear again.

The CMA-ES algorithm has been implemented in MATLAB by N. Hansen, from Technical University of Berlin [6] and can also be used in combination with any FEM code to evaluate the $\lambda$ children.

I-3. Evolutionary SAO (ESAO)

ES are known to be effective optimisation techniques but they involve a large number of FEM simulations what we absolutely want to avoid. The idea here is to combine both the SAO and the CMA-ES so that the advantages of one algorithm counterbalance the disadvantages of the other. The new algorithm, ESAO, is described in Fig. 4 and resembles the Metamodel Assisted Evolutionary Strategy proposed by Emmerich et al. [7].

The ESAO algorithm starts with the first steps of the SAO: choosing the design variables, constraints, and objective function, creating the DOE, running FEM simulation and fitting metamodels.

Then, on the selected metamodel, we introduce the CMA-ES. The starting $\mu$ parents of the ES are chosen to be the best $\mu$ points of the initial DOE. Then we generate $\lambda$ offspring individuals and evaluate them on the metamodel. Following with the CMA-ES, we select the best $\mu$ points based on the metamodel, add them to the DOE points and run the corresponding FEM simulations in order to update the metamodels and increase their accuracy.

**Figure 4. The Evolutionary Sequential Approximate Optimisation algorithm**

Then the recombination step occurs, followed by covariance matrix adaptation and new population sampling. The new generation will be evaluated on the updated metamodels again and so forth until a stopping criterion is met.

We note that the evaluation of the $\lambda$ individuals is made on the metamodel thus avoiding $\lambda$-$\mu$ highly time consuming FEM simulations. Then we run the FEM simulations only on the best $\mu$ selected points, becoming the parents of the new generation. This is the key difference between the CMA-ES and the ESAO. In the CMA-ES all of the $\lambda$ search points are evaluated by FEM simulations.

II Comparison between SAO, CMA-ES and ESAO

In the first part of this section one can find the description of the test function and parameters used to obtain the results exposed in the second section. The results are used to compare the algorithms and investigate their sensitivity with respect to the number of variables.

II-1. Test function

All simulations presented in this part are realised with the following Generalised Rosenbrock Function [Equation (1)] with global optimum described in [Equation (2)]:

$$f(x) = \sum_{i=1}^{N-1} \left( 100 \left( x_i^2 - x_{i+1} \right)^2 + (1 - x_i)^2 \right)$$  \hspace{1cm} (1)

$$f(x) = 0 \text{ for } x_i = 1, \quad i = 1, \ldots, N$$  \hspace{1cm} (2)

Where $x$ is the vector containing the design variables and $N$ is the size of $x$, i.e. the number of design variables. Figure 5 illustrates the 2-dimensional case.

**Figure 5. Rosenbrock function in two dimensions**
The Rosenbrock function is considered to be difficult, because it has a ridge. The top of the ridge is very sharp, and it runs around a parabola. To find the valley is trivial, however convergence to the global optimum is difficult.

The problem we choose to optimise is the following:

Minimise: \( f(x) \)
With: \( x_i \in [0, 1.5] \),
For \( i = 1, \ldots, N \) and \( N = [2, 3, 5, 7, 10] \)

In our method we also need to define the number of starting DOE points and the maximum number of points we can use during the improvement part. In this test we decide to start with \( 10N \) DOE points and improve until \( 20N \) points.

Regarding the ES strategy we also have to define \( \lambda \) and \( \mu \). These parameters change depending on \( N \). Table 1 represents the corresponding values. To summarise, we keep the best 25% of the \( \lambda \) search points to generate the next population.

Table 1. Values of \( \lambda \) and \( \mu \) depending on the number of design variables.

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \lambda )</th>
<th>( \mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

II-2. Results

First we want to investigate the sensitivity of the three algorithms from two to ten variables. Up to five variables we can use the SAO algorithm, with LHD and FFD but for more than five variables it becomes impossible with the condition we defined: \( 10N \) DOE points to start. One difficulty with factorial designs is that the number of combinations increases exponentially with the number of variables one wants to manipulate. FFD is made of \( 2^N \) corner points. Hence in 10 dimensions it would be \( 2^{10} = 1024 \) corner points, which is not an option to consider. For this reason we do not use the FFD for more than five variables and consider LHD only. The corresponding results will be denoted as LHD (Latin Hypercube Design).

Figure 6 represents the objective function improvement vs. the number of variables. For each of the three algorithms, we use the same starting DOE and the same starting objective function value. One can see in solid line the results with standard DOE’s (FFD + LHD), and with dashed lines we use only LHD. The same results are reported in Table 2.

By observing these first results, we can see that the SAO algorithm is quite constant and efficient when using only LHD. Standard SAO, using LHD and FFD, is very effective when the number of variables is below five. Once again, this is totally related to the number of corner points. A lot of corner points implies fewer points inside the domain, and consequently a less accurate metamodel.

CMA-ES also gives good results but it is not constant. This is the hazard of ES’s. We will see later that it needs more iterations than SAO to obtain the same results.

![Figure 6. Objective Function Improvement depending of the Number of Design Variables](image-url)
Table 2. Values of $\lambda$ and $\mu$ depending on the number of design variables

<table>
<thead>
<tr>
<th>Var. Values</th>
<th>Initial Optimal values</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAO</td>
<td>CMA-ES</td>
<td>ESAO</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2</td>
<td>0.27</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>2.00</td>
<td>99%</td>
</tr>
<tr>
<td>5</td>
<td>4.00</td>
<td>20%</td>
</tr>
</tbody>
</table>

ESAO is not as good as expected. One of the reasons is the combination of a not very accurate metamodel with random sampling by CMA-ES.

In the following figures we can see the evolution of the objective function values from $10N$ to $20N$ DOE points.

Figure 7 shows the results for three and five variables. The graphs represent the evolution of the optimal values for standard DOE (FFD+LHD) and only for LHD for each of the three algorithms. Figure 8 corresponds to seven and ten variables and contains only LHD as explained previously.

In order to be able to compare the algorithms, for each number of variables the three algorithms start with the same $10N$ DOE points and the same objective function value. We notice two different starting values for the first two graphs. This is because the two DOE’s used do not necessarily contain points in common and then do not give the same first optimal value.

In addition we can observe that some curves stop before $20N$ DOE points. This is because they give very good results very quickly. What we want to know is just which of the methods is the most efficient and for that we do not always need to achieve the $20N$ FEM simulations.

In each of the graphs, the SAO algorithm is the first one to obtain the best improvement. We are now sure that, in the case of this test function, SAO is better than ES and the combination of both algorithms. Indeed ESAO does not give any improvement.

Moreover for an increasing number of design variables, we definitely have to use LHD alone, rather than its combination with FFD as proposed in [1-4].

III COMPARISON OF DESIGNS OF EXPERIMENT

Results from Section II point out that there are some differences to investigate between the DOE’s. In this part we use the SAO algorithm and new DOE’s to compare their efficiency.

One more thing we have to observe is the influence of the position of the optimum. Indeed if we play with corner points in the DOE, perhaps the position near or far from such corner points may be non-negligible.
III-1. Design Of Experiment (DOE)

We already explained that the SAO algorithm is based on both Latin Hypercube Design (LHD) and Full Factorial Design (FFD). In this part we will use another design called Fractional Factorial Design illustrated in Fig. 9-b. The difference between Full and Fractional is that we do not use all the corner points but only a fraction, half of them for instance.

In order to be able to compare the different DOE’s we use five variables in the new experiments. Hence we will have the following designs:
- Full Factorial \(2^5 = 32\) corner points,
- Fractional Factorial \(2^{5-1} = 16\) corner points,
- Fractional Factorial \(2^{5-2} = 8\) corner points,
- Latin Hypercube Design \(0\) corner points.

Both Fractional Factorial Designs will be denoted as ‘Frac 8’, eight corner points, and ‘Frac16’, sixteen corner points.

Each of the Full and Fractional Factorial Designs are combined with LHD in order to obtain the desired number of starting DOE points: \(10 \times 5 = 50\) points.

III-2. Results

In this section we use the same test function as previously, the Rosenbrock function. But the domain of the variables will change. We will compare the three following cases:
- 5 variables in \([0, 1.5]\)
- 4 variables in \([0, 1.5]\), 1 in \([-1, 1]\) (2\textsuperscript{nd} variable)
- 4 variables in \([0, 1.5]\), 1 in \([-1, 1]\) (5\textsuperscript{th} variable)
- 3 variables in \([0, 1.5]\), 2 in \([-1, 1]\) (2\textsuperscript{nd} & 5\textsuperscript{th} variables)

In Fig. 10 one can observe the evolution of the objective function values vs. the number of DOE points for each of the four different DOE’s. It is important to note that in each picture, four different DOE’s are compared, consequently the optimisation starts with different objective function values.

Even if SAO gives quite good results with all of the DOE’s, the ‘Frac 8’ (Fractional Factorial Design \(2^{5-2}\)) is always the first one to find the best results.

When the optimum is inside the domain the FFD is not accurate enough because of the use of too many corner points. We can see it starts with a good optimal value, lucky choice of the DOE points, but it does not improve it whereas in only 20 FEM simulations ‘Frac 8’ finds better optimal values.
Figure 10. Comparison of the DOE’s: (a) Optimum inside the domain, (b) 2nd variable on the boundary, (c) 5th variable on the boundary, (d) Two variables on the boundary.

IV CONCLUSIONS

In the scope of a project called Optimisation of Forming Processes, an algorithm to solve optimisation problems for metal forming processes using time consuming FEM simulations has been developed.

The efficiency of this Sequential Approximate Optimisation (SAO) algorithm decreases with increasing number of design variables because of the high number of FEM simulations required. To overcome such a problem two solutions have been investigated in the present research.

The first one was the combination of the SAO algorithm with an ES to avoid too many function evaluations; the second one was to use different DOE’s in order to find more accurate metamodels using less function evaluations.

Even if these comparisons have not yet been applied to a real industrial study, it can be concluded that the SAO algorithm outperforms both the ES and the ESAO algorithm, i.e. it gives more accurate results for less iterations. This is due to the accuracy of the metamodels which is not precise enough to be combined with a random method; it alters too much the evolution of the CMA-ES.

In the third section, it is especially emphasised that when the number of design variables is larger than five in the SAO algorithm, then the choice of the DOE becomes crucial. The comparison of different DOE’s shows that the SAO algorithm
associated with a fractional factorial design is the most efficient method.

To conclude SAO is very capable and efficient and the use of corner points or the combination with an ES is not beneficial in the case of the Rosenbrock function. It is recommended that some other test are performed to validate this work and in addition a comparison to an industrial metal forming case using FEM simulations as function evaluations would be of great importance.

ACKNOWLEDGEMENTS

This research has been carried out in the framework of the projects “Optimisation of Forming Processes MC1.03162” and the First Europe “METAOPT”.

The first project is part of the research programme of the Netherlands Institute for Metals Research (NIMR). The industrial partners cooperating in this project are gratefully acknowledged for their useful contributions to this research.

The support of the Région Wallonne and the Fonds Social Européen under the project First-Europe EPH3310300R075F/415823 is also gratefully acknowledged.

REFERENCES


