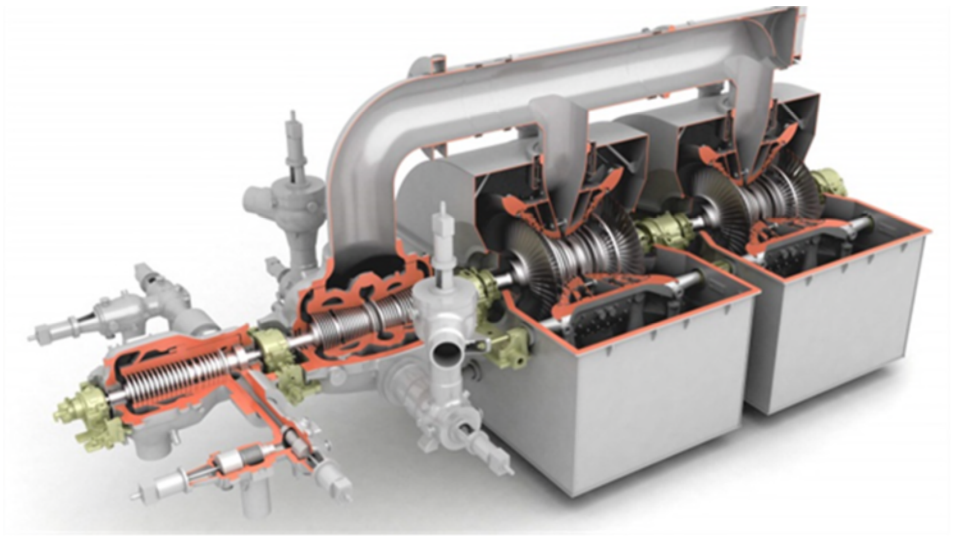
EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling



Blade path optimisation in steam turbine design

Alissa Kamilova

Contents

1. Introduction

Background

Blade paths describe the best arrangement for of the different kinds of blades in turbines, as well as their physical components and materials

Steam turbines are used in 70% of global power generation (nuclear, thermo-solar, fossil fuels, etc.). Most new power plants require a bespoke, custom-optimised steam turbine which has the highest achievable efficiency. Siemens Power and Gas design their steam turbines by using a computational model which maximises the efficiency of a turbine while keeping within the mechanical, thermodynamic, and aerodynamic specifications. The design process is normally done in two stages; a preliminary design is first produced using computer software, and this is then refined by experienced engineers. This procedure is done for high pressure and medium pressure turbines, since they generally consist of smaller blades and are not subject to problems such as vibrations which have to be dealt with separately. The design outcome is the optimal arrangement of the blades in the turbines, as well as their different physical components and materials. This is called the *blade path*, as it refers to the path through which steam flows during turbine operation. As seen in Figure 1, the blade path is formed of different stages. Each stage contains a *stator* blade, which remains static during normal operation, and a *rotor* blade, which rotates in order to allow *mass flow* through the turbine.

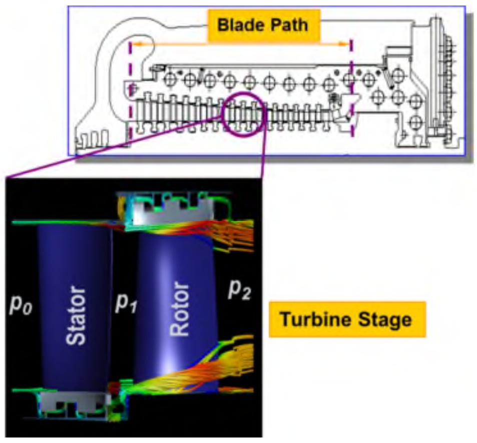


Figure 1: Blade path example for a high pressure turbine. The path, shown in the top image, is formed by a number of turbine stages. Each turbine stage, shown in the bottom image, is formed by a stator blade, remaining in place throughout the operation of the turbine, and a rotor blade, rotating to transport steam through the turbine.

3DV program

The program 3DV uses the optimisation routine E04UCF to maximise the efficiency of a turbine and provide its optimal blade path

The software used for generating the preliminary blade paths is called 3DV and it uses the external NAG optimisation routine E04UCF to solve the relevant flow problem and produce a design that will allow the highest efficiency. To reach a solution quicker, it is necessary to provide a starting point for the program. Normally this involves supplying the optimal design for a similar turbine that has already been approved, and then use these specifications as a starting point for the new turbine.

Furthermore, even though some of the variables are discrete, such as the types of materials to use, they are all considered continuous until a path is generated, and then are discretised accordingly. A normal usage of 3DV will have the user change between the continuous optimisation and the discretisation process several times before settling on a particular design. This is not always a quick process and it depends highly on the experience of the user, since there is currently no way of ensuring that the design chosen cannot be improved upon. Additionally, 3DV always stalls at some point, indicating that the current efficiency is the highest one found by the software. However, if it is then restarted, the program can find a more optimal point, until it stalls again. This process is repeated a number of times, where the number depends on the experience of the person in charge of the design. The lack of standardisation in producing blade paths affects the reliability of the final turbine design. Our aim is to investigate the stalling behaviour of 3DV and explore possible ways to avoid it.

2. Stalling behaviour

During a typical 3DV code run, the optimisation will stall several times before indicating that a more optimal point could not be found

An example of a typical output with 3DV is shown in Figure 2. The blue line indicates the efficiency evaluated at each point produced by the software when it is first run. Then, the solution found at the last iteration is used as a starting design and the program is restarted. The results of this procedure are shown in the orange line. Note that, particularly after evaluation 100, the efficiency seems to reach its highest point, but this is not selected as the optimum by 3DV, lower values are produced until stalling once more. Finally, when the code is restarted for the second time, now using the specification obtained at the end of the green curve, 3DV stops immediately, indicating the point could not be improved upon.

After not finding a more optimal point, the code can be restarted and most times it will improve the objective function

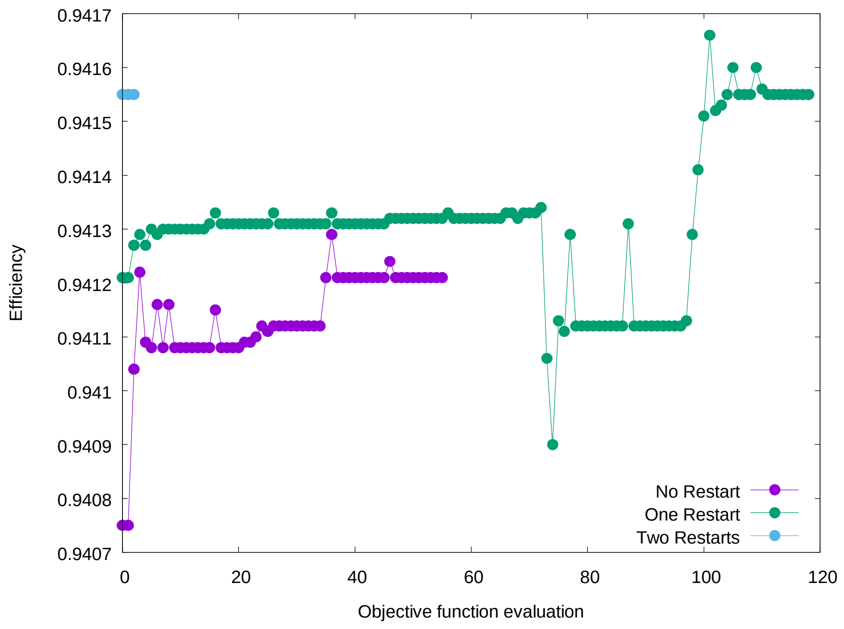


Figure 2: Results of the optimisation routine, first for the initial run (no restart), then restarting the program using the solution produced by the initial run (1 restart) and finally restarting it again, using the solution reached in the orange line (2 restarts). Note how although the efficiency reaches a point which cannot be improved upon, there were larger values of efficiency in previous iterations which were disregarded by the program.

It is clear from Figure 2 that the algorithm finds designs where the efficiency is higher, but does not accept them. Additionally, there is repeated stalling behaviour throughout the optimisation procedure.

Glossary of terms

- Objective function: A function $f(x)$ which will be minimised or maximised to solve an optimisation problem.
- Objective function gradients: First derivative of the objective function, $\nabla f(x)$
- Constraints: A condition that an optimisation problem solution must satisfy.
- Linesearch: A strategy to find a local minimum (or maximum) of an objective function $f(x)$, in which a descent direction is found along which the objective

function will be reduced, and then a step size Δx will be computed to determine how far x should move along that direction.

- Linearisation: A procedure that finds a linear approximation to a function at a given point.
- Curvature: The amount by which a geometric object, such as a surface, deviates from being a flat plane, or a curve deviates from being a straight line.

Sequential Quadratic Programming

The optimisation routine E04UCF uses a Sequential Quadratic Programming (SQP) method to solve the optimisation problem. This routine is set up as a minimisation procedure, with the objective function

$$f(x) \coloneqq -F(x),$$

where $F(x)$ is the efficiency of the turbine. As with any optimisation method, the goal is to find a new solution x_{k+1} , provided we know the current solution x_k, such that

$$x_{k+1} = x_k + \alpha p$$

and

$$f(x_{k+1}) < f(x_k)$$

where $\alpha > 0$ is the step length, p is the search direction and $f(x)$ is the function to minimise evaluated at point x. This way the line search will be performed in the direction p with step of size α.

E04UCF uses Sequential Quadratic Programming (SQP), which performs a number of minor iterations per each major iteration

What distinguishes SQP methods from other optimisation algorithms is that p is calculated as the solution of a Quadratic Programming (QP) subproblem in which the constraints of the original problem are linearised. Furthermore, accurate information about the gradients of the objective and constraints is essential for the correct functioning of the method, since this will determine the step direction and whether or not a point is accepted.

Comments

- Each iteration k of the SQP method is called a major iteration. During each major iteration, the QP subproblem requires a number of minor iterations to reach to an acceptable search direction. Theoretical results indicate that when the maximum efficiency obtainable is reached, the number of minor iterations will converge to 1.
- The QP subproblems are solved at low computational cost because they are linear, at least in terms of the way the constraints are written. Since linear approximations are used for all of them, the algorithm is built to handle linear constraints in a more natural way than nonlinear ones.
- 3DV also produces monitoring output which includes valuable information such as whether or not the objective function has decreased from one iteration to another, as well as the step length at each iteration. It is important to note that a very common occurrence for the example shown in Figure 2 is step lengths $\alpha \approx 10^{-4}$, with a decreasing efficiency, which means the algorithm has found a direction that was deemed acceptable, but then only allowed small steps to be taken towards it.

3DV solutions to our problem have very small step lengths and sometimes exhibit a decrease in the overall efficiency

3. Maratos effect and how to avoid it

The stalling behaviour, output of 3DV and the SQP algorithm construction suggest that the optimisation is suffering from a well-known problem in SQP methods called the *Maratos effect* [1]. It occurs when the SQP algorithm takes steps which satisfy the acceptance criteria for the search direction obtained in the QP subproblem, but fail the ones pertaining to the major iteration. In this case, it causes the algorithm to accept points where the efficiency decreases and more constraints are violated.

The Maratos effect occurs when the curvature of the constraints is not properly captured by the linearisations of the constraints

The Maratos effect occurs because the linearisation of the constraints performed for the minor iterations does not capture all the information on the curvature of constraints in the full problem. As a consequence, the subproblem will produce points that are not in the solution space of the original problem. A graphical interpretation of what occurs during the Maratos effect is shown in Figure 3.

In our case, the problem solved by 3DV involves highly nonlinear equations, which have in practice not been correctly represented when performing the optimisation.

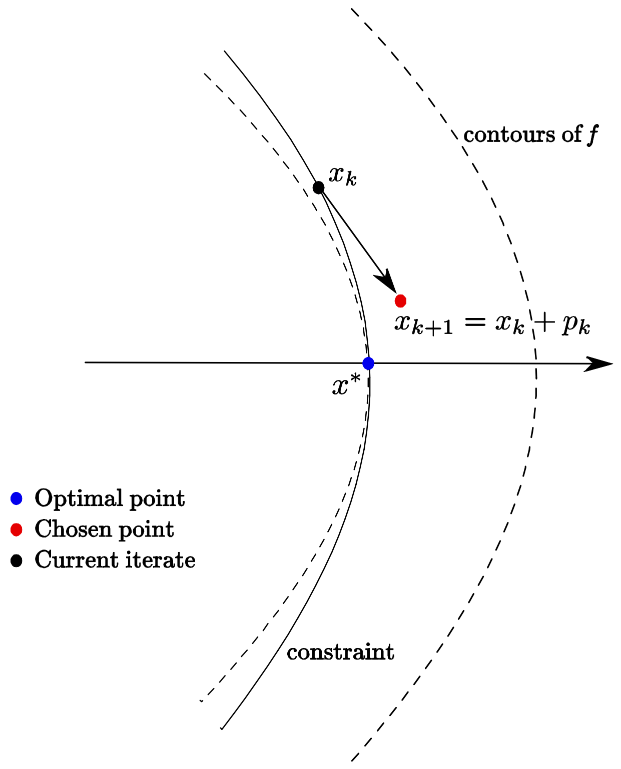


Figure 3: Example of the Maratos effect, reproduced from [1]. The curvature of the constraint is not captured properly by the QP subproblem, so the chosen point moves away from the optimal point, instead of towards it.

Interior-point methods

Since we have no access to the details of the E04UCF routine implementation, possible corrections directly for the Maratos effect cannot be performed. Instead, we chose to implement a completely different numerical scheme which utilises an interior-point method.

These types of methods have shown good practical performance for solving highly nonlinear optimisation problems, although the theory is not as robust. During each iteration, an auxiliary problem is solved, and this solution is guaranteed to be inside the region where the solution of the original problem is found, hence the name interior-point.

Interior point methods avoid the Maratos effect by incorporating a filter which prevents the cycling of results

Furthermore, the curvature information of the constraints is encoded in the second-order derivative approximation performed internally, since the optimisation problem does not supply it. Additionally, a list of previous objective function values is stored. At each iteration the new value is compared against this list and rejected if it is found to be equal to any entry. This is called a *filter* and it prevents the cycling of optimisation results.

Ipopt is a package that implements an interior point method, which allows the user to tailor the algorithm to the problem being solved

Implementing Ipopt

We use a software package called Ipopt for the implementation of our interior-scheme into 3DV. The aim is to have a code which computes exactly the same things in the same output format as with 3DV. The software package was chosen because it is very well documented and has a vast array of options that can be modified in order to personalise the implementation as much as possible to the particular problem at hand.

We find that it is necessary to rewrite most of the input functions into the correct format for Ipopt. This changes the structure of the code in a way that the output values of the constraint gradients do not provide the same results as with the previous optimisation. We are able to incorporate the Ipopt package library successfully into 3DV and we find that we are able to reproduce the output plots, files and blade paths made by the standard 3DV.

4. Discussion, Conclusions & Recommendations

We investigated the optimisation program 3DV used by Siemens Power and Gas for calculating blade path designs for high pressure and medium pressure turbines. After running multiple examples, we identified a stalling behaviour The method used by the optimisation routine uses a sequential linear programming algorithm to perform the optimisation in 3DV. We found that the linearisation of the constraints performed by every subproblem within each major iteration did not represent the curvature of the constraints correctly, which is a common problem in these methods known as the Maratos effect. This led the algorithm to accept suboptimal points that didn't achieve the highest efficiency and violated some of the constraints. Since Siemens are using a commercial optimiser, it was not possible to apply any corrective measures to prevent the Maratos effect directly in the algorithm.

Instead, we investigated interior point methods as an alternative optimising algorithm. These do not suffer from the Maratos effect, although they are not considered as robust as SQP methods. We attempted to integrate Ipopt, an interior point filter line-search algorithm software package into 3DV. Ipopt is well documented and has a vast array of options that allow personalisation to each particular problem, allowing the user to have better control over the algorithmic behaviour. We successfully linked the libraries required for Ipopt with the ones that 3DV is compiled with, and the full 3DV output was produced in the same way as with the previous optimisation (including output files, blade path diagrams and plots).

The objective function gradient presented errors of $\approx 10^{-4}$ in both E04UCF and Ipopt

However, we encountered problems when rewriting the codes to fit Ipopt specifications. In particular, we suspect that the gradient of the objective function provided by 3DV is not correct, as both the previous optimisation and the Ipopt implementation show relative errors of $\approx 10^{-4}$ when comparing their respective inner derivative checker. Since the efficiency is requested to an accuracy of at least four decimal places, these errors have a great impact on the final result. Additionally, the pressure constraint between the stages is not satisfied, preventing the incorporation of more stages in the blade path.

Suggestions for future work include a detailed investigation into the objective function gradient calculation to identify why discrepancies occur in both algorithmic approaches. We recommend taking a simpler approach in future implementations, in which the optimisation code is written separately from 3DV and import the objective function output, as well as the constraint data through an interface. This would be a better starting point for a full Ipopt working implementation, and would allow optimisation results to be obtained and analysed before proceeding with the integration with 3DV.

5. Potential Impact

Siemens Power and Gas use the 3DV optimisation algorithm for all their low and high pressure blade path designs; however it has remained relatively unchanged for over 15 years. Investigation into the stalling behaviour of 3DV has provided insight into the model used and the properties of the implemented optimisation routine. Additionally, we have found issues within the coding itself, such as the calculation of the objective function gradient and the implementation of the constraints, which could improve performance significantly if they are properly addressed. The implementation of an alternative optimisation algorithm such as the interior-point method will provide an opportunity for comparison and will be a good indication on the next steps necessary to take to improve the performance of 3DV. Ultimately, a more appropriate algorithm for blade path optimisation will reduce cost, time, and effort in all turbine designs.

Armin de Lazzer, Senior Key Expert Blading Technology, Siemens said: *Turbine blade path design for present steam conditions requires the aid of an optimization methodology not only to design for highest efficiency but also to find a solution satisfying all design constraints, given the complexity and non-linearity of the aero-mechanical problem. Once a working and sufficiently stable optimization methodology has been found, engineers tend to compensate remaining limitations of the methodology by using expert knowledge; this is not a desirable condition for the company since design output becomes dependent on individuals. For Siemens, Alissa's work provides us with very valuable understanding of the behaviour of the present design system and where it comes from.*

Her analysis gives us clear guidance on how the system should be improved to suit our needs better. Within her work, she has shown that mathematically based selection of an optimization approach suited to the specific problem type has the potential not only to improve the individual design result but also to denote a huge step towards consistency of designs across individual designers. We are convinced that the implementation of the methodology proposed by Alissa within the design systems will improve the design result. For us at Siemens, Alissa's project served as an eye-opener to the benefit of systematic mathematical assessment of engineering optimisation problems, and as a cornerstone for forthcoming developments of fully automated mixed discrete-continuous and multi-objective blade path optimization at Siemens. These results justify our decision to continue our participation as a partner in this program.

References

1. Nocedal, J. et al (2016) *Numerical Optimisation.* 2nd ed. Springer, New York, NY, 441-443.