Supervised Learning for Generating Fair Curves with Curvature Boundary Conditions

Samuel Sudhof, University of Stuttgart, Germany

Norimichi Uemura, Interstellar Technologies, Japan

Masaharu Uchiumi, Muroran Institute of Technology, Japan

Abstract. Fairness measures are means to guess if a curve is suited for an aerodynamic application if no knowledge of the flow field is available. Fair curves are proposed as a means to reduce the number of parameters for generating a supersonic rotor blade. A specific new measure of fairness is proposed with the goal of being numerically less demanding to evaluate than previously known measures of fairness. Gradient search for an optimum of this fairness value is nevertheless computationally expensive. Therefore, a neural network was trained using Adamax optimization that can produce curves of acceptable quality instantly for applications where real time response is needed.

1. Introduction

Systems analysis for future low-cost small satellite launchers with gas-generator cycle engines indicate that there is a strong demand for single stage supersonic turbines. This caused interest in the development of a reliable design method for suitable rotor blades with fast turnaround times.

A well-known method for designing supersonic turbine rotor blades was first published by Boxer et al [1] and later expanded upon by Goldman and Scullin [2]. Under the assumption of isentropic central flow and disregarding shock-boundary interactions, the authors describe a mathematically viable source-vortex flow-field in the rotor.

This method offers a way to analytically calculate how blade parameters such as pitch and throat width influence the overall blade and flow shape, however, during the development of Vulcain, it was found that such predictions are not completely reliable [3]. More recently, supersonic rotor blade profiles were optimized for desirable flow properties using three-dimensional computational fluid dynamics (CFD) [4] [5]. The resulting shapes of such optimizations consistently differ from profiles obtained by the aforementioned analytical method.

Notably, the analytical method proposes that large parts of the pressure and suction sides of the profile are concentric circles, which is a logical consequence of the demand for a source-vortex flow. The optimizations depart from this by having a strongly curved section near the inlet. Further, the successful designs of the first turbine stages for Vulcain 2 have an inflection on the suction side near the outlet [6].

In the development of a computer-assisted method for blade design with fast turnaround times, even imperfect information can be useful if it allows a reduction in the number of parameters or it allows
better initial guesses for blade parameters, as such improvements that lower the need for time-consuming CFD simulations. Therefore, a new method should be informed by the best guess available through analytic means.

By using concentric circular arcs, the source-vortex model suggests specific ratios of curvature between locations on the pressure side of the profile and locations on the suction side. A parametric system is being designed to make use of these curvature suggestions, while implementing the ability to modify them.

In the interest of fast turnaround, the focus will not be on finding perfect curves for the profiles, but instead to use as few parameters as possible to specify well-performing profiles.

A precise description of the parametric system will be a topic for a later publication, as this paper discusses a method to generate a suitable curve segment for a blade profile using only boundary conditions of location, direction and curvature at the start and end points.

2. Definition of a Curve Segment

The curve segments discussed here will be quartic Bézier curves [7]. All quartic Bezier curve can be expressed using aforementioned three boundary conditions and two open parameters like this:

Let the control points of a Bézier curve be \( \vec{P}_m \) with \( m \in \{0,1,2,3,4\} \) and \( \vec{P}_m = (P_{m,x}, P_{m,y}) \). Let the curvature be \( k(u) \) with \( k(0) \) and \( k(1) \) known boundary conditions. Let \( n = 4 \) be the degree of the curve.

\[
\begin{align*}
G &= \| \vec{P}_4 - \vec{P}_0 \| \\
H &= \| \vec{P}_3 - \vec{P}_4 \| \\
g &= \frac{\| \vec{P}_4 - \vec{P}_0 \|}{G} \\
h &= \frac{\| \vec{P}_4 - \vec{P}_0 \|}{H} \\
P_{ax} &= P_{1x} - P_{0x} \\
P_{ay} &= P_{1y} - P_{0y} \\
P_{bx} &= P_{4x} - P_{3x} \\
P_{by} &= P_{4y} - P_{3y} \\
P_{2x} &= \frac{P_{bx}(L_a - E_{1a} + E_{2a}) - P_{ax}(L_b - E_{1b} + E_{2b})}{P_{ax}P_{by} - P_{ay}P_{bx}} \\
P_{2y} &= \frac{L_b - E_{1b} + E_{2b} + P_{by}P_{2x}}{P_{bx}} \\
L_a &= \frac{G^2 n \cdot k(0)}{n - 1} \\
L_b &= \frac{H^3 n \cdot k(1)}{n - 1}
\end{align*}
\]

The input parameters \( G \) and \( H \), together with the tangency boundary condition define the location of the control points \( \vec{P}_1 \) and \( \vec{P}_2 \). The equations above can be used to find the coordinates of the Point \( \vec{P}_2 \).

With this, the quartic curve is fully defined. The derivation is omitted here for brevity, but it is described in some more detail by Sudhof [8]. As this citation is not available on the date of this conference, the relevant content is summarized here:

On page 22 of The NURBS book [7] is a listing of the first and second end derivatives of Bézier curves. The expression for the curvature of a parametric curve, which is a function of first and second derivatives, can be found in many textbooks on this topic. For example, Pressley [12] writes about it in great detail.

By inserting the definitions given by Piegel and Tiller [7] into the curvature given by Pressley’s equation (2.2) one obtains one expression for the curvature at each end of the curve, two expressions in total. After substituting in the definitions for \( G \) and \( H \), there are two unknowns: the coordinates of the
control point \( P_2 \). It then only remains to solve a system of two linear equations for those two unknowns, \( P_{2x} \), and \( P_{2y} \).

Empirically, the following parameters have proven useful for specifying \( G \) and \( H \):

\[
p = g^2 + h^2
\]

\[
b = \frac{g^2}{g^2 + h^2}
\]

The variable names \( p \) and \( b \) are specified with a graphical understanding of their effect on the curve as “pointedness” and “(left-right) bias” in mind. In the following, only the parameters \( p \) and \( b \) were manipulated to obtain fair curves.

These parameters have the advantage that their useful areas are known independently of the problem. Values of \( p \) are geometrically meaningful in the range \([0, \infty[\), but it is defined in such a way that for relevant curves, parameter values higher than 1 are unlikely to be the optimal solution. Values of \( b \) are only meaningful in the range \([0,1]\).

To make the problem symmetrical towards both parameter boundaries, \( b \) was changed to have just \( g \) rather than \( g^2 \) in the numerator, after this analysis was concluded. This new version of the parameters will be used in future publications.

2.1. The Concept of Fairness in Curve Geometry

“Fairness” in the context of curve geometry means “beauty”, and like beauty, it does not have a single accepted definition. Two common definitions are described by Moreton and Séquin [9] with some information on how they can be numerically determined.

Fairness is a useful concept for profile curves, because it allows the selection of a curve that likely to be useable for an aerodynamic surface without requiring any CFD simulation. While such a simulation could be used to optimize for better parameters \( p \) and \( b \) than those that result from optimizing just the curve fairness, an evaluation of curve fairness takes a fraction of a second, while an evaluation of the flow in CFD takes anywhere from ten minutes to several days. Therefore, the turnaround time of blade development can be reduced.

As the boundary conditions of the curves include prescribed curvatures, the second fairness measure described by Moreton and Séquin [9] would be appropriate, i.e. the integral of the square of the derivative of curvature along the curve. Unfortunately, none of the methods for evaluating this fairness measure have proven to be accurate and fast enough for the present kind of problem. In the extremes of the range of the parameters \((p, b)\), numerical evaluation methods tend to underrepresent the fairness value (higher fairness value is worse). This causes gradient search to gravitate towards false minima, i.e. minima where the result of the numerical evaluation has a minimum, but the true integral value does not. An adaptive refinement method was able to alleviate this issue, but not to the point of making the results useable in an optimization.

Instead, a new measure of fairness was used, which is much less susceptible to such issues. In the following, fairness \( S_3 \) is defined on the local derivative of curvature \( k'(u) = \frac{dk(u)}{ds(u)} \) in two different cases.

\( S_3 = \max(|k'(u)|) = |k'(u_{max})| \) (case 1) unless there is an extremum with the opposite sign of this maximum. If there is an extremum with the opposite sign of \( k'(u_{max}) \), then let the largest extremum with the opposite sign of \( k'(u_{max}) \) be \( k'(u_{max2}) \). Then \( S_3 = |k'(u_{max})| + 10 \cdot |k'(u_{max2})| \) (case 2).

The second case is in place to penalize curves that have unnecessary undulations in the curvature derivative. More common methods of describing curve fairness, such as the one referenced above also discourage undulations on the curvature derivative. In practice, the new fairness measure was able to generate curves that are similar in shape to the more sophisticated established methods, while being numerically much easier to evaluate with high precision.

Especially near the edges of these parameter ranges, the curves can behave erratically, forming very sharp turns or cusps. The technical implementation of the new fairness function contains special handling of such features, based on the known [7] properties of Bézier curves, but with more extreme parameter values, these features become more common and more localized, which makes it harder to
evaluate the function correctly. As the fairness function is evaluated thousands of times for each individual data point, even an error rate of one in one million evaluations will cause the dataset to contain a large amount of false optimum curves.

3. Conventional Optimization for Fairness

Because of the aforementioned problems with extreme parameter values, a balance has to be struck between the desire to find the best possible curves in every single case and the numerical effort of correctly evaluating the fairness function. Therefore, the interval for the parameters \((p,b)\) was limited from \([0,1]\) to \([0.02,0.98]\). Very few optimal curves have parameter values near or at the edge of the chosen interval.

To start the search, an 80x80 regular grid of samples is taken of the search domain. Subsequently, a gradient search using an interior-point method in a subdomain with the size 0.08x0.08 centred on the lowest sample is conducted, and the result of this search is the final optimum parameter choice.

At the beginning of the research for this paper, optimizing the parameters \((p,b)\) of a single curve would take more than a minute on a recent consumer-grade desktop CPU, but in the final iteration, the time was cut down to 15 seconds by various optimizations. Multiple such calculations can be carried out simultaneously to take advantage of multi-core processors.

Even 15 seconds is quite long when this method is to be used in an interactive program, so it was speculated that a neural network trained using a recent learning method may be able to predict the optimal parameters \((p,b)\) if given enough examples of optimized curves.

4. Machine Learning

Since the optimization for fairness is indifferent of scaling and rotation, a data set of curves that all run between the start and end points \(P_0 = (1,0)\) and \(P_4 = (-1,0)\) was created. Initially, the other boundary conditions were chosen arbitrarily from a wide array of possible values, but as most of the resulting curves where irrelevant for blade development, a narrow range of input values was used that specifically represented the kind of curves on the suction side near the inlet and outlet of a rotor profile. These curves are marked “1” and “2” in figure 1.

In the case of a straight line, the start and end tangents would both be \((-1,0)\). To specify the tangents for one sample, the straight tangents are rotated by angles described by the first two open parameters.

- The first angle is chosen randomly from \([1°,35°]\). The start tangent gets rotated by negative one half of this value, and the end tangent gets rotated by positive one half of the value.
- A second parameter modifies only the start tangent. This tangent is additionally rotated by an angle chosen randomly from \([-12°,6°]\).

The last open parameter of the samples defines the curvature at the start point. The method by which it does this consists of two steps. The first is to determine what a “typical” curvature may be. The way this was done here is by finding the normals of the start and end tangents and finding the intersection point \(M\) of those normals. The typical curvature is defined as \(1/|MP_0|\). The final step is to multiply this typical curvature by a factor, which was chosen randomly from \([0.8,5]\).

The curvature of the end point was always chosen to be zero.

Figure 2 gives three examples of curves specified in this manner with the two open parameters optimized for the fairness measure \(S_3\) described above.

5.1. Network Design

57238 samples of optimized curves were generated. The samples where split with 1/5 of the samples (11448) set aside as a validation set and 45790 used for training a network.
Formally, this is a regression problem with 3 inputs, 2 outputs and an initially unknown number of hidden layers in between. All inputs were normalized so that the ranges described above are each mapped to the range [0,1]. Likewise, with the ranges for the parameters $p$ and $b$.

The settings of the network had to be chosen carefully to avoid some issues that are inherent with this data. First, the data contains a few optima at or very near the lower end of the possible parameter range. This eliminates the possibility of using a sigmoid activation function, because those parameters at the edge of the interval can never be reached. Any activation function that is not strictly limited in its output value range (such as leaky rectified linear unit) is likewise unusable, because values of $b$ below 0 or above 1 as well as values of $p$ below 0 are not meaningful. Therefore, rectified linear unit (ReLU) was chosen as the activation method of each layer with added upper limit, so that all results fall in the valid parameter range.

Predictably, this caused many optimizers to struggle, as methods based on gradient decent can get stuck if they ever reach a point at the low end of the range. Multiplying any learning step to a weight that holds the value 0 will not change this weight.

Furthermore, the neural network only had access to optimized results, not to the objective function itself. This provided a problem in the evaluation of loss. Loss had to be determined on the distance of the network result from the optimized result. However, the network may guess a value that is close to a local (not global) minimum and therefore represent a very low fairness value (i.e. a good curve). The optimization method may then judge, based on the distance between the local and the global minimum, that a large penalty must be applied. In order to mitigate this, the loss function was chosen to be mean absolute error, rather than mean squared error, thereby reducing the penalty for a small number of very high error predictions.

TensorFlow version 2.1.0 with GPU support [10] was chosen as the machine learning library, and all optimizers supported by TensorFlow for regression problems where evaluated. Adamax [11] with standard settings as described in the citation produced the best results. It particularly excelled at limiting the number of stuck neurons.

After some iteration it was found that a model with 4 hidden layers, of 300 neurons each modeled the data without overfitting.

The mean absolute error on the learning set was 0.0109 and 0.0105 on the validation set at this stage. The resulting curves of this network where evaluated using the objective function, and it was found that a significant number of curves predicted by the network had better fairness than the respective curves from the input data set, although in almost all cases the difference was very small.

As described above, the initial gradient search optimization was carried out by first evaluating a grid of samples. This sampling comprises the bulk of the total processing time, and the effort rises with the...
square of the resolution. The 80x80 search grid is therefore a compromise and cannot be expected to find the global minimum in every case.

To eliminate this problem from the data, the network prediction was used as the initial value of a new gradient search for each sample. If the result of this new gradient search had better fairness than the result of the initial gradient search, the training data would be updated to use the new search result rather than the original search result. The network training could then be run again on the improved training data, leading to new predictions that could in turn be used as initial points for gradient search.

This process was iterated 3 times with the final loss (mean absolute error) being reduced to 0.0087 for the training set and 0.0085 for the validation set.

99.5% and 95.5% of all predictions from the final network for \( p \) and \( b \), respectively, differed by less than 0.05 from the best known curves.

5. Results

Overall, the network performed well at predicting the curve parameters. Figure 3 shows a histogram of the fairness of 10,000 samples from the validation set. It can be seen that there is some reduction in the number of very high quality curves with fairness value \( S_3 \) lower than 0.5, and an increase in curves in the fairness range between 1.5 and 4.5, which are generally good, but not excellent curves. It is especially encouraging that all of the difficult samples (with fairness >10) were recreated by the network with minimal loss of quality. Figure 2 gives example curves to help visualize the kind of curvature distributions associated with these fairness values.

In the validation set, the average fairness of the best known input parameters was 3.90, while the average fairness of the network predictions was 3.94.
As the iterative improvement of the learning data mostly fixed a small number of outliers in the data, it did not significantly improve the average fairness.

It is hard to make specific statements about the aerodynamic performance of any curve without context, but the fairness of these curves shows that none of them have any obvious defects. Sudden turns, kinks or cusps in a curve would result in fairness values several orders of magnitude higher. The relatively narrow data range of the input data and the low number of input and output values do put the results into perspective, however.

A simple test was devised to determine if the network is generating useful information or merely memorizing inputs. For each entry in the validation set, the parameters values $p$ and $b$ were determined by nearest neighbour interpolation of the training data. The results of this showed that interpolation was somewhat worse than the machine learning results with an average fairness of 3.98, indicating that the performance of the network cannot entirely be attributed to memorization.

6. Conclusion
While in the current iteration, the network produced curves of somewhat lower quality than the optimization, it is able to produce curves that are reasonable approximations of the best available curves. Gradient search takes more than 10 seconds to yield one curve but thousands of curves can be generated by the network within a second.

Therefore, some uses of the network or a similarly developed network are foreseen. For example, the curves could be used as live preview curves for an interactive blade design program or as starting values for a gradient search. As the generation of the sample grid to determine the best starting point takes the bulk of the time needed for the gradient search, this could yield very good results, while taking only about 0.5 seconds per curve.

Further, if a method of evaluating the more traditional measures of fairness that operates with acceptable speed and precision is implemented, the work on this paper may be adapted to the new input data with ease.

One of the challenges to generating input data was to select the range of random data in a way that represents the typical range of curves in profile problems. A better way to generate training data would be to gather the data from regular blade designs. As a large number of such data is generated in a profile optimization, an effort will be made to preserve these results as a more representative source of learning data.

7. References

[1] E. Boxer, J. Sterrett and J. J. Wlodarski, "Application of Supersonic Vortex-flow Theory to the Design of Supersonic Impulse Compressor or Turbine-blade Sections," 1952.
[2] L. Goldman and V. J. Scullin, "Computer Program for Design of Two-Dimensional Supersonic Turbine Rotor Blades With Boundary-Layer Correction," 1971.
[3] L. Eriksson, "Development of a Supersonic Turbine Stage for the HM60 Engine," in 20th Joint Propulsion Conference, 1984.
[4] L. Griffin and D. Dorney, "Design and Analysis of Turbines for Space Applications," in 33rd AIAA Fluid Dynamics Conference and Exhibit, 2003.
[5] K. Kawatsu, N. Tani, M. Shimagaki, M. Uchiumi, N. Yamanishi, K. Mitsuhashi and T. Mizuno, "Multi Objective Optimization of a Supersonic Axial Turbine Blade Row Shape for Rocket Engine Turbopump," in 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2011.
[6] S. Andersson, M. Lindeblad and U. Wahlen, "Performance Test Results for the Vulcain 2 Supersonic/Transonic Turbine," in 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 1998.
[7] L. Piegel and W. Tiller, The NURBS Book, 2nd ed., Heidelberg: Springer, 1997.
[8] S. Sudhof, Development Techniques for Supersonic Turbines, manuscript in preparation, 2020.

[9] H. P. Moreton and C. H. Séquin, Functional optimization for fair surface design, vol. 26, ACM, 1992.

[10] A. Agarwal, P. Barham, E. Brevedo, Z. Chen, C. Citro and G. S. Corrado, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/.

[11] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv, p. 1412.6980 [cs.LG], 2014.

[12] A. Pressley, Elementary Differential Geometry, London: Springer, 2010.