Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018

ICCFD10-129

Multiblock structured grids for direct numerical simulations of transonic wing sections

M. Zauner and N. D. Sandham
Corresponding author: m.zauner@soton.ac.uk

University of Southampton, UK.

Abstract: Direct numerical simulations of transitional and turbulent flows around airfoils at moderate and high Reynolds number require large and complex grids consisting of billions of grid points. Advances in computational resources towards exa-scale computing (10^{18} floating point operations per second) and powerful algorithms that aim to exploit the full potential of modern high-performance computing architectures allow an increase of size and complexity of such simulations. However, high-order numerical methods for structured curvilinear grids require continuous metric terms up to the second order of derivatives or higher. An evaluation of the requirements on grid-generation tools, stressing scalability, precision and flexibility, suggested the need for hand-crafted grids. In the present contribution, we outline a method based on polynomial functions and identify the benefits of such techniques for large structured multi-block high-fidelity grid generation around airfoil geometries, also providing an open-source tool for airfoils with sharp as well as blunt trailing edges.

Keywords: Grid Generation, Computational Fluid Dynamics, Airfoil, Open Source.

1 Introduction

So far, direct numerical simulation (DNS) of aircraft wing sections is limited to relatively low Reynolds numbers compared to real flight conditions. As DNS needs to resolve the smallest length scales of turbulent structures (Kolmogorov scales), each spatial dimension needs $N_L \approx \frac{L}{\eta_m}$ grid points to discretise a characteristic length scale L. Assuming homogeneous isotropic turbulence we can write the microscale η_m as

$$\eta_m = \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}} = \left(\frac{\nu^3 L}{U^3}\right)^{\frac{1}{4}}, \tag{1}$$

where ν, U and ϵ denote the kinematic viscosity, the characteristic velocity and the dissipation rate, respectively. Hence, the number of grid points in three dimensions scales with the factor

$$N_L^3 \approx \left(\frac{L}{\left(\frac{\nu^3 L}{U^3}\right)^{\frac{1}{4}}}\right)^3 = Re_L^{\frac{9}{4}}. \tag{2}$$

That means to increase the Reynolds number for a DNS by a factor of 16, the grid would be theoretically 512 times larger. Equation (2), based on isotropic turbulence, tends to overestimate the grid size, as the laminar and potential flow regions do not contain turbulent structures and can have a lower grid resolution. Nevertheless, the increase of the overall computational costs is still considerable. Since 2003 the Reynolds number (in this contribution always based on the axial chord length c at zero incidence, unless otherwise stated) of resolved DNS has risen from $Re = 10{,}000$ [1] to $Re = 500{,}000$ [2]. Various recent publications, such as [3], [2], [4], [5] and many more show the challenges, but also the potential of high-fidelity simulations

to investigate complex transonic flow phenomena over wings in order to improve the aerodynamic efficiency and extend the flight envelope for next-generation aircraft at moderate Reynolds numbers.

A new generation of DNS codes, such as OpenSBLI [6], aims to exploit modern high-performance computing architectures more effectively and advance towards exa-scale computing to eventually solve problems at high Reynolds numbers. DNS of transitional and turbulent flows around airfoils involving the application of high-order central-difference methods for structured curvilinear grids, require continuous metric terms up to the second order of derivatives or higher. Otherwise, artificial disturbances are introduced into the Navier-Stokes equations, increasing the numerical error. Commercial software packages often apply methods that solve elliptic partial differential equations to increase the smoothness of the grid. Those approaches are problematic for geometries with a blunt trailing edge in a three-block CH-grid configuration (figure 1), because the whole grid cannot be described by a single matrix. Furthermore, in a direct numerical simulation, the grid has to cover a wide range of length scales. On the one hand, it is necessary to resolve boundary- and shear-layers sufficiently well, whereas on the other hand one does not want to over-resolve the potential flow, in order to keep the computational efforts as low as possible. Without intervening manually, grid smoothing algorithms tend to shift points from the highly resolved regions into the freestream region. The memory requirements to smooth and live-render large DNS grids can be challenging as well. With respect to analyses of the acoustic field or global instabilities, it is also beneficial to control the spacing in the free-stream region, avoiding too coarse a grid spacing.

Simulations were carried out by [7], using the predecessor code of OpenSBLI and splitting the computational domain into three blocks consisting of one C-block ($B2$) around Dassault Aviation's V2C airfoil geometry and two H-type blocks ($B1$ & $B3$) enclosing the wake region and outflow. The arrangement of all three blocks is sketched in figure 1. There are three block interfaces in between $B1$ and $B3$, $B2$ and $B1$ as well as $B2$ and $B3$. The close up of figure 1 indicates how the interfaces to block 2 are related to the two corners of the blunt trailing edge. In figure 1, the coordinates of the Cartesian coordinate system are denoted by x and y in free-stream direction and normal to it, respectively. The curvilinear coordinates are denoted by ξ and η in the circumferential and radial direction, respectively. Curves that consist of grid points with constant values of η are called ξ-gridlines, whereas curves of constant ξ-values are denoted as η-gridlines (grey-dashed curves in figure 1). The large grey arrows indicate the direction of increasing grid point number. $B2$ is generated first and then $B1$ and $B3$. An evaluation of the requirements on the grid generation tool, stressing scalability, precision and flexibility suggested the need for hand-crafted grids of the kind used by [7].

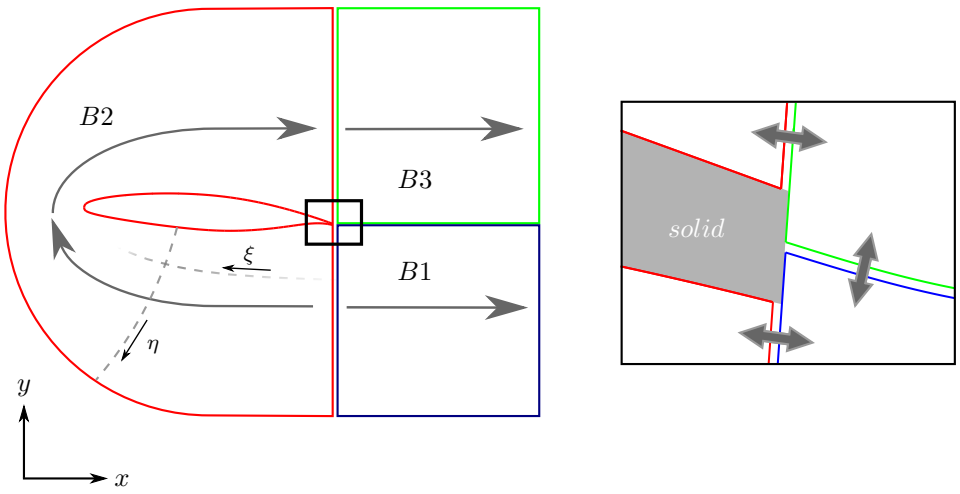

Figure 1: Segmentation of the grid into three blocks.

An in-house tool was developed to generate large high-fidelity grids for airfoils on mainstream single-core computers or laptops. The grid spacing is described by high-order polynomials, while the gridlines are generated by a blending procedure. The approach has been successfully tested and applied for DNS at moderate Reynolds numbers, using grids that consist of more than 10 million points in the 2D $x-y$-plane. An optimised grid resolution requires good understanding of the flowfield. More precisely, the user needs to know critical regions and the length-scales of turbulent structures. This poses a big problem, as it is often not possible to identify those regions and length-scales beforehand. As we want to reduce the need for extensive grid studies involving expensive 3D simulations, we apply an error-severity indication tool, developed by [8], to quantify and compare the quality of grids a posteriori. This approach analyses the Fourier spectrum of derivative quantities such as vorticity in order to identify grid-to-grid point oscillations that do not decay with expected rates. Regions can then be identified for more focused grid refinement. The grid analysis and refinement strategy is discussed in detail by [9].

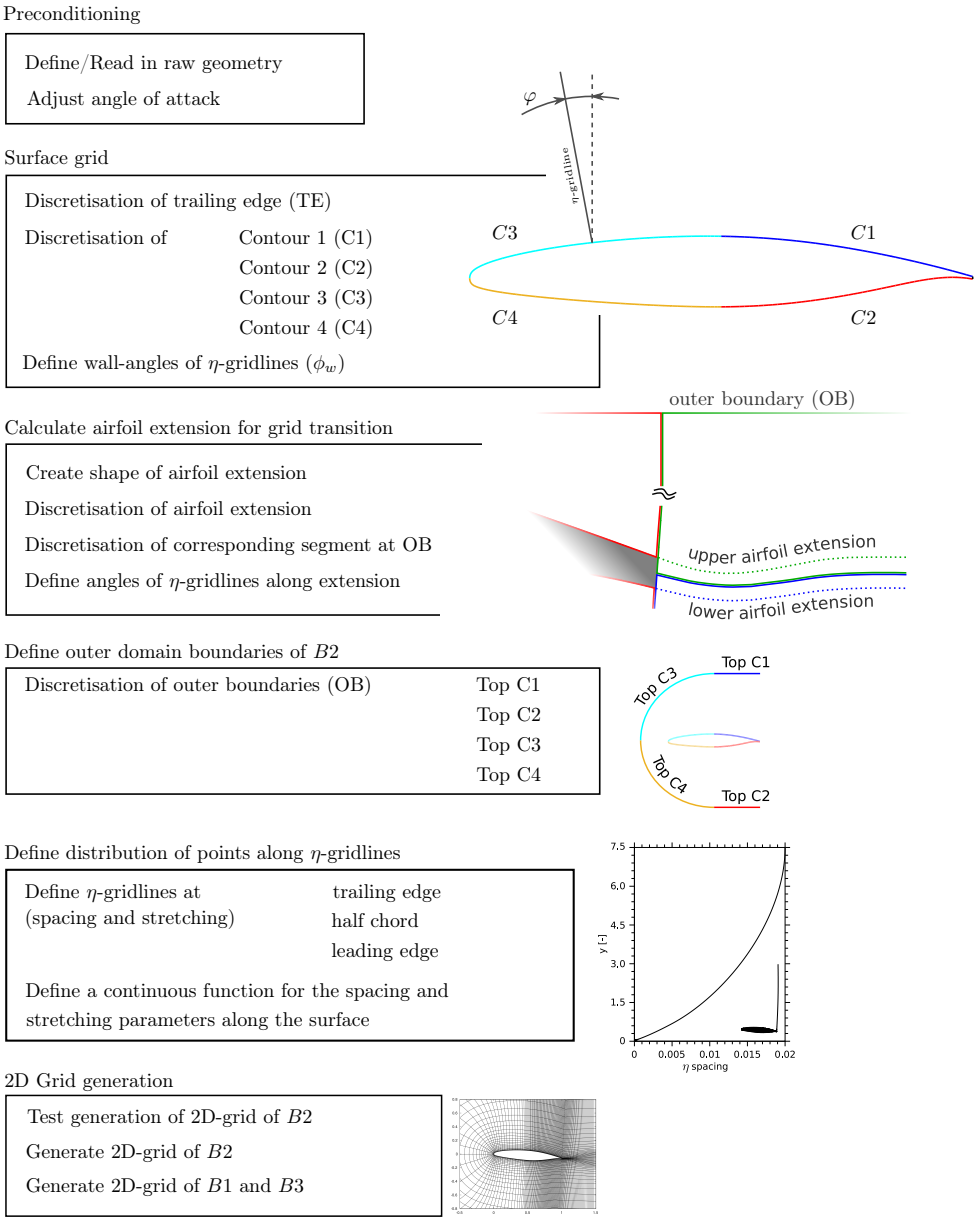

Figure 2: Flow chart outlining the grid-generation process.

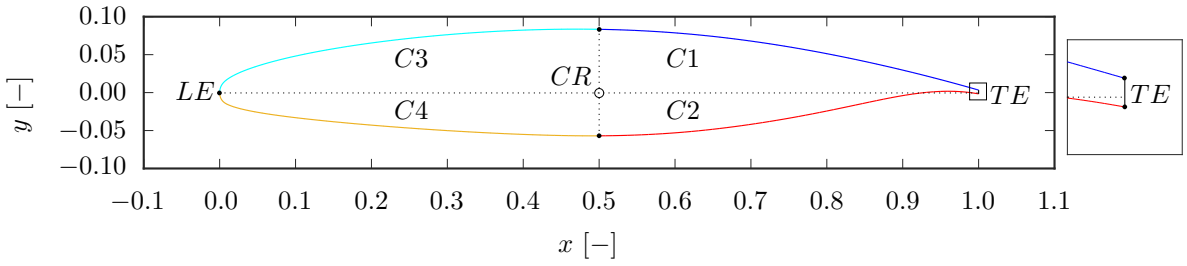

Figure 3: Segmentation of Dassault Aviation's V2C airfoil.

The full grid-generation process is described in the following section. Figure 2 can be seen as a guideline of the work flow. It outlines a process consisting of six main steps. After setting the angle of attack and general parameters regarding the 2D domain size, the airfoil surface and the grid around the trailing edge is defined. The domain boundaries are discretised in a next step, which is followed by the design of the η-gridlines. In a final step the full 2D grid is generated and written out. An open-source version of the grid-generation tool for Matlab$^{\mathrm{TM}}$(discussed in section 3) also follows the structure shown in figure 2.

2 Grid-generation process

2.1 Preconditioning and airfoil surface

In order to keep numerical errors due to the applied interpolation methods low, the raw-input surface geometry of the airfoil is specified with significantly higher resolution compared to the final grid (approximately 20 times higher). In order to increase the flexibility regarding the distribution of grid points, the airfoil is divided into four sections corresponding to figure 3, labelled as $C1$, $C2$, $C3$ and $C4$. For the flow cases in [7], sections $C1$ and $C2$ needed to be highly resolved to capture the laminar/turbulent transition process, whereas sections $C3$ and $C4$ allow higher stretching due to the laminar boundary layer. The centre of rotation (CR) to adjust the angle of attack (α) is located at half chord, on the mean chord line (the horizontal dotted line in figure 3 with $y = 0$). The reference length is the axial chord length of the airfoil at zero inclination and independent of the angle of attack. After the airfoil is rotated by an angle of attack (the red contour in figure 4), it is vertically shifted, so that the leading edge (LE) again coincides with the abscissa of the Cartesian coordinate system (the green contour in figure 4).

The discretisation of the airfoil starts at the blunt trailing edge (TE). An equidistant distribution of points (constant spacing) is applied for the linear segment, connecting both corners. The number of grid points resolving the TE needs to be chosen carefully, as the TE spacing also defines the height of the wall-closest cell of the η-gridlines containing the corners of the TE. In order to discretise the airfoil, each section is unwound and the surface distance s is calculated. The distribution of N gridpoints along the unwound section is defined by a polynomial of 6^{th} order and its derivatives according to

$$A\xi^6+ \quad B\xi^5+ \quad C\xi^4+ \quad D\xi^3+ \quad E\xi^2+ \quad F\xi+ \quad G = \quad s(\xi), \tag{3}$$

$$6A\xi^5+ \quad 5B\xi^4+ \quad 4C\xi^3+ \quad 2D\xi^2+ \quad 2E\xi+ \quad F \quad = \quad \Delta s(\xi), \tag{4}$$

$$30A\xi^4+ \quad 20B\xi^3+ \quad 12C\xi^2+ \quad 4D\xi+ \quad 2E \quad = \quad \Delta s'(\xi), \tag{5}$$

$$120A\xi^3+ \quad 60B\xi^2+ \quad 24C\xi+ \quad 4D \quad = \quad \Delta s''(\xi), \tag{6}$$

where the spacing and its derivatives are denoted by Δs, $\Delta s'$, $\Delta s''$ and $\Delta s'''$, letters $A - G$ denote the unknown coefficients and the variable ξ is in the range of $\xi \in [1, N]$. We can now define Δs and $\Delta s'$ at the first ($\xi = 1$) and last point ($\xi = N$) of the section. In addition we can choose $\Delta s''$ at the front- or back-end

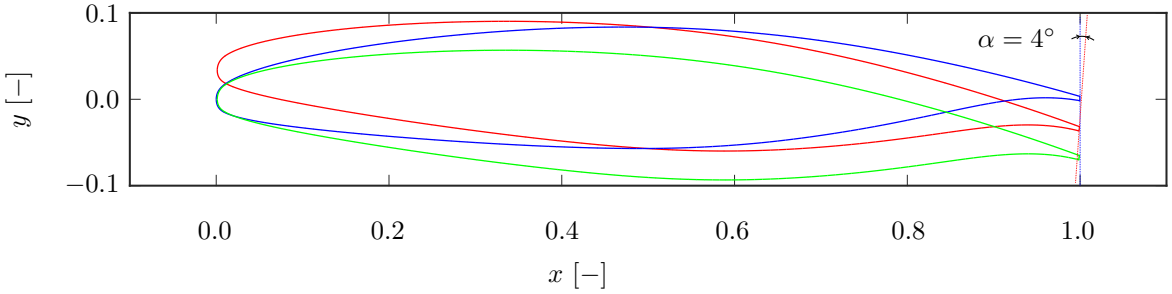

Figure 4: Transformation of the original profile (blue) to an airfoil with 4° angle of attack. The purely rotated profile is described by the red contour, whereas the green contour considers also a translation in order to have the origin at the leading edge.

as well. The parameter $\Delta s''$ allows us to fine-tune the spacing independently of other parameters, whereas Δs and $\Delta s'$ need to agree with adjacent sections. Applying the boundary conditions to equations (3) to (6) gives a system of seven linear equations that can be written in matrix form as

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ N^6 & N^5 & N^4 & N^3 & N^2 & N & 1 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 6N^5 & 5N^4 & 4N^3 & 3N^2 & 2N & 1 & 0 \\ 30 & 20 & 12 & 4 & 2 & 0 & 0 \\ 30N^4 & 20N^3 & 12N^2 & 4N & 2 & 0 & 0 \\ 120i^3 & 60i^2 & 24i & 4 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{bmatrix} = \begin{bmatrix} 0 \\ s(N) \\ \Delta s(1) = \frac{\partial s}{\partial \xi}|_{\xi=1} \\ \Delta s(N) = \frac{\partial s}{\partial \xi}|_{\xi=N} \\ \Delta s'(1) = \frac{\partial^2 s}{\partial \xi^2}|_{\xi=1} \\ \Delta s'(N) = \frac{\partial^2 s}{\partial \xi^2}|_{\xi=N} \\ \Delta s''(i) = \frac{\partial^3 s}{\partial \xi^3}|_{\xi=i} \end{bmatrix} \tag{7}$$

This system can be solved by multiplying the right-hand-side vector by the inverse of the coefficient matrix on the left-hand-side. The parameter i depends on whether the section's front- or back-end is fine-tuned and is set to 1 or N, respectively. After having calculated all unknowns, equation (3) provides a function for the surface distance $s(\xi)$ of each grid point on that segment. We also know the Cartesian coordinates of N_{raw} discrete points on the raw input surface section as a function of the surface distance $s_{raw}(\xi_{raw})$. Therefore, we can calculate the Cartesian coordinates of the grid points on the surface, applying a cubic spline interpolation method according to [10]:

$$\begin{aligned} x(\xi) &= spline\{s_{raw}(\xi_{raw}), x_{raw}(\xi_{raw}), s(\xi)\},\ \xi \in [1, N]\ and\ \xi_{raw} \in [1, N_{raw}] \\ y(\xi) &= spline\{s_{raw}(\xi_{raw}), y_{raw}(\xi_{raw}), s(\xi)\},\ \xi \in [1, N]\ and\ \xi_{raw} \in [1, N_{raw}]. \end{aligned} \tag{8}$$

This calculation scheme is also used to allocate the grid points at the outer domain boundaries and along η-gridlines within the domain. In order to achieve the optimal distribution of grid points, it might take some iterations to adjust the derivatives of the spacing and the number of points within that section. Due to the simplicity of the procedure it is possible to compare the changes after each iteration in a simple line plot as illustrated in figure 5, where only the number of grid points is changed. Furthermore, it is possible to calculate the spacing that would be needed to comply with wall units that are typical for the applied CFD-method (DNS, LES, RANS, etc.). The required spacing in the wall-normal direction Δn can be calculated

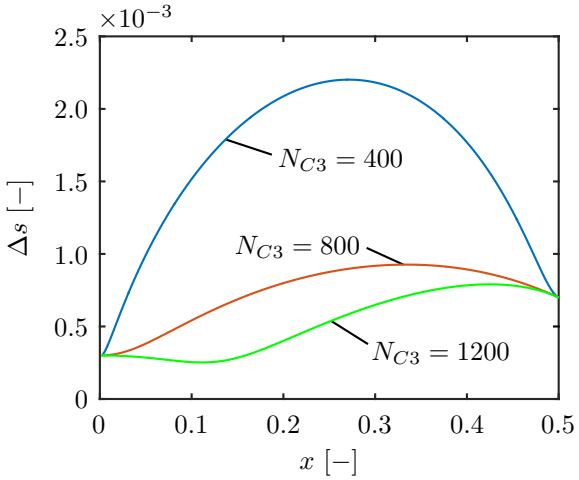

Figure 5: Spacing between grid points in the ξ-direction on the surface of section $C3$ as a function of x for different numbers of points.

corresponding to the skin-friction coefficient C_f according to

$$\Delta n = \Delta y^+ \cdot l^+ = \Delta y^+ \cdot \frac{\nu}{u_\tau} = \Delta y^+ \cdot \nu \cdot \sqrt{\frac{\rho}{\tau_w}} = \Delta y^+ \cdot \nu \cdot \sqrt{\frac{2}{C_f \cdot U_\infty^2}}, \tag{9}$$

where Δy^+ denotes the spacing in the wall-normal direction normalised by the wall unit l^+. In order to fully resolve turbulent boundary layers in DNS, that spacing should be $\Delta y^+ < 1$ at the wall. The spacing in the wall-tangential direction x^+ is usually larger, but depends on the flowfield. In our case we typically aim for $\Delta x^+_{wall} \approx 5 \cdot \Delta y^+_{wall} \Rightarrow \Delta s \approx 5 \cdot \Delta n_{wall}$. The skin-friction coefficient C_f is often unknown a-priori, but can be estimated, with open-source wing-design tools like XFOIL [11]. The calculated target distribution can now be plotted as a function of x on top of the current resolution to support the iterative process of adjusting the polynomial.

After the whole airfoil surface is discretised, the wall angle φ_w, describing the inclination of η-gridlines at the wall, is calculated for each section. For wall-normal gridlines, the wall angle would be defined as

$$\varphi_w(\xi) = \tan^{-1}\left(\frac{\partial x(\xi)}{\partial y(\xi)}\Big|_{\eta=0}\right). \tag{10}$$

However, in concave sections in $C2$, gridlines would intersect with each other. Furthermore, the wall-normal vectors at the corners are not aligned with the wall-tangential vector of the blunt trailing edge. The wall angle at the TE corners is chosen to correspond to the angle of attack α, whereas it can be wall-normal near the leading edge. At $x \approx 0.5$ we choose vertical η-gridlines ($\varphi_w \approx 0$). The transition can be accomplished by high-order polynomials, blending, or a combination of both. On the one hand, polynomials can be defined precisely at the boundaries and therefore allow segments of the grid to be connected without creating discontinuities. On the other hand, the shape of high-order polynomials is hard to control. A blending procedure is a simple way to combine the strength of high-order polynomials at the boundaries with simple functions that describe the middle pieces. As an example figure 6 illustrates the blending from a wall-normal φ_w close to the LE (red curve) to vertical gridlines ($\varphi_w = 0$) towards the half-chord for section $C3$. We assume we have two vectors of the same length, where each discretises a continuous function. For this example, both vectors have a length corresponding to the resolution $N_{C3} = 400$ of the surface section $C3$ (figure 3). The first vector $\varphi_{w,I}$ contains values of the angle corresponding to wall-normal gridlines (the red curve), whereas the second vector $\varphi_{w,II}$ has the same size, but contains only zeros (the green line). In figure 6, both vectors are displayed as a function of their index ξ. A kernel vector F_b is now defined by a polynomial

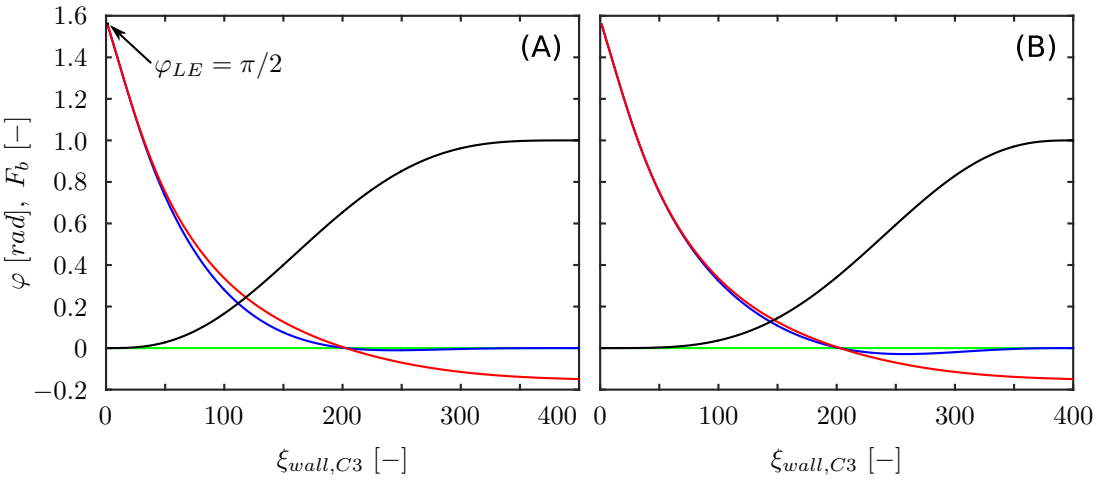

Figure 6: The blue function for φ_w is calculated by blending the wall-normal angles (red curve) with zero (green line), applying a polynomial blending kernel F_b according to the black curve. Plot (A) and (B) show the effect of different blending functions F_b.

function having the same size. The kernel is $F_b(1) = 0$ at one end and $F_b(N_{C3}) = 1$ at the other end. In order get a smooth transition from the values of $\varphi_{w,I}$ at the beginning and $\varphi_{w,II}$ at the end of section $C3$, both vectors are blended according to

$$\varphi(\xi) = \varphi_{w,I}(\xi) + F_b * [\varphi_{w,II}(\xi) - \varphi_{w,I}(\xi)], \text{ with } \xi \in [1, \mathrm{N_{C3}}]. \tag{11}$$

The kernel F_b defines where exactly and how fast the transition should take place. The effect of changing the kernel-function is indicated in figure 6 comparing (A) and (B). The blending factor F_b (black curve) in (A) increases faster than in (B) and therefore the final blue curve approaches the green zero line earlier. As we aim to keep the η-gridlines as far as possible wall-normal, we try to delay the transition from the red curve to the green line a bit longer, without overshooting the zero-line too much. Too fast changes of the function describing ϕ, or overshooting the zero-line significantly, can again result in gridline intersections. Eventually, the blue curve in figure 6(B) follows the red curve, corresponding to wall-normal η-gridlines, for longer, before settling down to zero and meets our requirements perfectly. This blending procedure is frequently applied during the course of the grid generation, as it enables smooth transitions between constant values, polynomials or other discretised functions in a simple way. Furthermore, it can be used for fine-tuning.

2.2 Airfoil extension

Towards the outflow of $B1$ and $B3$, the grid-design philosophy is to transition the curvilinear gridlines to horizontal and vertical ξ- and η-gridlines, respectively (i.e. a Cartesian grid). The stretching in the η-direction in the region close to the interface of $B1$ and $B3$ further downstream of the trailing edge is reduced so that the distribution of grid points is uniform across the wake.

A grid transition region is defined ($1 \lesssim x \lesssim 1.5$), where the ξ-gridlines that contain the corners of the blunt TE (wall-gridline) are designed as smooth extensions of the airfoil profile (the dotted curves in figure 7). The ξ-gridlines transition to horizontal lines. The extensions of the wall-gridlines in the grid transition zone are designed as high-order polynomials. In the case of the V2C airfoil, both extensions converge. Blending techniques are applied to better control the shape and limit the unfavourable compression of the wake segment between both extensions. Towards the end of the grid transition zone, those extensions diverge, so that the resolution in the wake is as uniform as possible in the η-direction. This is also achieved by an additional blending procedure. It is noted that the design of those extension gridlines requires a careful trade-off, as either a too fast transition to horizontal gridlines or an exaggerated compression of the region

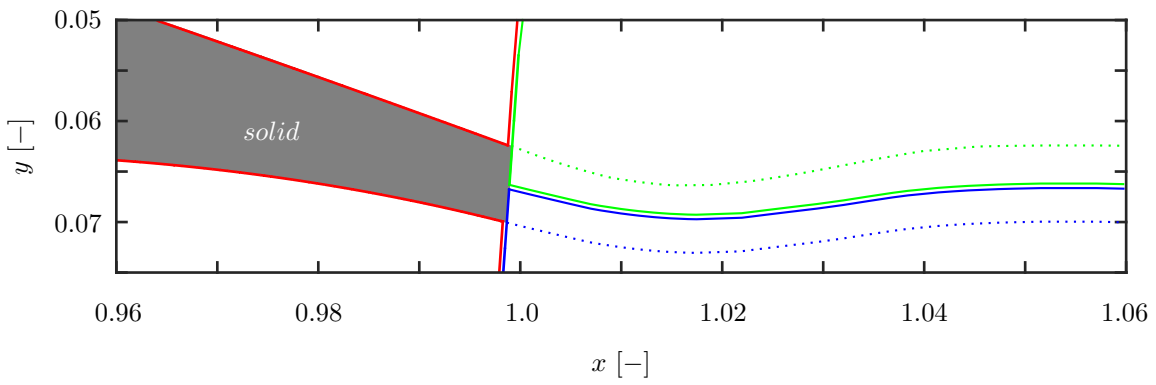

Figure 7: Sketch of a blunt trailing edge illustrating the extension of the contour by ξ-gridlines (dotted lines).

between both smooth extension-gridlines can affect the simulation time-step significantly.

The final gridlines forming the continuous airfoil extension are discretised in the ξ-direction in the same way as the airfoil surface in section 2.1. The same spacing function ($\Delta = f(\xi)$) is used to allocate the grid points along the corresponding segments of the outer domain boundaries of $B1$ and $B3$.

2.3 Outer domain boundaries of $B2$

The outer domain boundaries (OB) of $B2$ consist of a semicircle with a radius of $R = 7.5c$, centred at $x = 0.5c$ and $y = 0$, that is linked by two horizontal lines on both ends to its corner points at $x = 1c$ and $y = \pm 7.5c$ (figure 1). The outer boundaries are discretised in the same way as was described for the surface boundaries. The spacing of each section can be defined independently from the surface, as we ease the restrictions of linear η-gridlines. Angles φ_{OB} have to be defined also at the OB in a similar way as done for φ_w. We can define those angles normal to the outer boundaries, but near $x = -7c$ and $y = 0$ it is necessary to blend the angle that is normal to the boundary with an angle corresponding to horizontal η-gridlines in order to avoid gridline intersections.

2.4 Discretisation of η-gridlines

After defining the wall and outer domain boundaries, the η-gridlines need to be specified. The shape of each η-gridline is defined in three steps that are illustrated in figure 8. The aim is to find a smooth, contained transition from a wall-optimised curve (the green curve in figure 8(A)) to a boundary-optimised curve (the green curve in figure 8(B)). In order to get the wall-optimised curve in figure 8(A), a blending between the blue line, agreeing with the defined gridline-angle at the wall (φ_w), and a red line, connecting the point at the wall (I) with the corresponding point at the outer boundary (II) is performed. In figure 8(B), the outer-boundary optimised curve is calculated in the same way, considering a blue line associated with the pre-defined angle at the outer boundary (φ_{OB}). In order to generate the final η-gridline (the magenta curve) for that specific point on the surface, both green curves are blended again according to figure 8(C). The discretisation of each gridline is also defined by a polynomial function and involves the same procedures that are applied at the airfoil surface in the ξ-direction (section 2.1). In order to fine-tune and better control the distribution of points, each η-gridline can be divided into two sections (representing the near- and far-field). Each section is again discretised applying polynomial functions that can be connected without creating discontinuities. This is done for grids in [7], but requires additional iterations of the process.

Representative η-gridlines and all required control parameters (spacing, stretching, etc.) are manually defined at the trailing edge, half chord and leading edge. Based on those locations, continuous functions $(f(\xi), \xi\epsilon[1, N])$ for each parameter are generated along the wall, the outer boundaries and the airfoil extensions. In the case of splitting the η-gridline into two parts, additional functions for parameters need to be

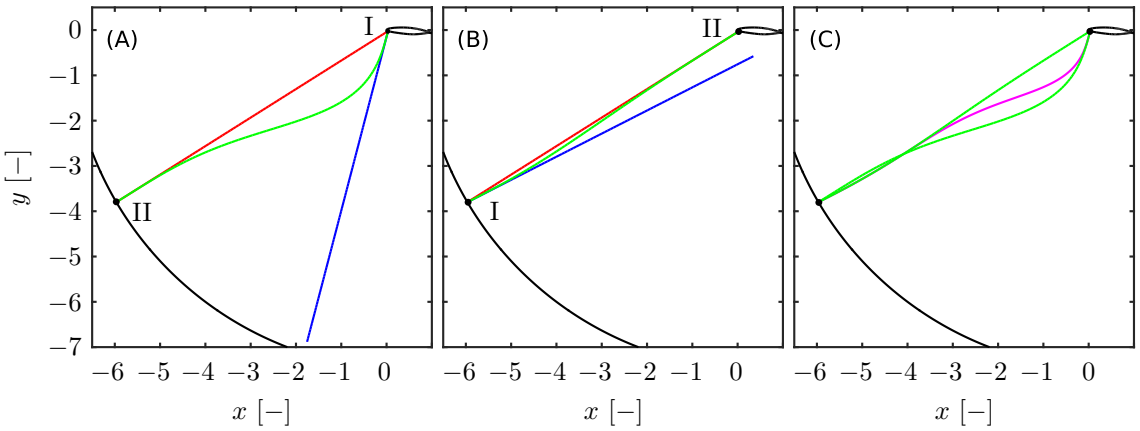

Figure 8: Illustration of the three steps to generate the η-gridlines in block 2.

defined at the interface. Appendix A shows a simple example of how to optimise the distribution of grid points along η-gridlines.

2.5 2D grid generation

The final grid is generated by looping through all grid points of the airfoil surface in the clock-wise direction, as indicated by the arrows in figure 1. For each wall grid point, the corresponding η-gridline is generated and discretised. In order to keep the memory requirements low, each gridline is written out after each iteration. Although not implemented yet, this process can be fully parallelised. After $B2$ is generated, $B1$ and $B3$ are created in two steps, looping trough all points along the airfoil-extension gridlines (dotted curves in figure 7) in the positive x-direction. Firstly, the η-gridlines between the convergent-divergent airfoil extension and the outer boundaries are calculated in the same way as for $B2$. In order to connect those η-gridlines, a segment in the passage between the extension gridlines is generated by a blending procedure considering two lines corresponding to the pre-defined inclination of the η-gridlines along the extension gridlines. For this connection segment an equidistant spacing is applied that corresponds to the spacing of the first cells of the adjacent η-gridlines. All three segments are merged and consequently split into two parts according to $B1$ and $B3$. Secondly, after the transition to a Cartesian topology, the η-gridlines are vertical with a fixed distribution of grid points in the y-direction. Only the spacing in the x-direction changes. The η-gridlines can be simply copied, by just varying the x-coordinates of each point. Each block is written out as double-precision ASCII-files. A separate Python- or Fortran-script (depending on the CFD code) compiles the binary grid file that is read in by the DNS code, corresponding to the spanwise extent and resolution. The 3D domain is homogeneous and periodic in the spanwise direction.

2.6 Proof of continuity

The continuity of the grid is checked by looking at sensitive metric terms in representative regions of the domain. Figure 9 shows $(\partial\eta/\partial y)/\partial\xi$ for the wall-gridline of the V2C profile, including three close-ups of the corners at the blunt trailing edge (left and right hand side plots) and the leading edge (centre plot). No discontinuities are presented in this metric term. Finally, figure 10 shows continuity of $(\partial\eta/\partial y)/\partial\xi$ along the η-gridline that goes through both corners at the blunt trailing edge. Again, no discontinuities are detected.

3 Open-source software package

The code can be found online in a Github repository (`https://github.com/ZaunerM/PolyGridWizZ`). A DOI is assigned to the current release (version 0.0.1-beta)[12]: 10.5281/zenodo.1245598. A fully-parallelised

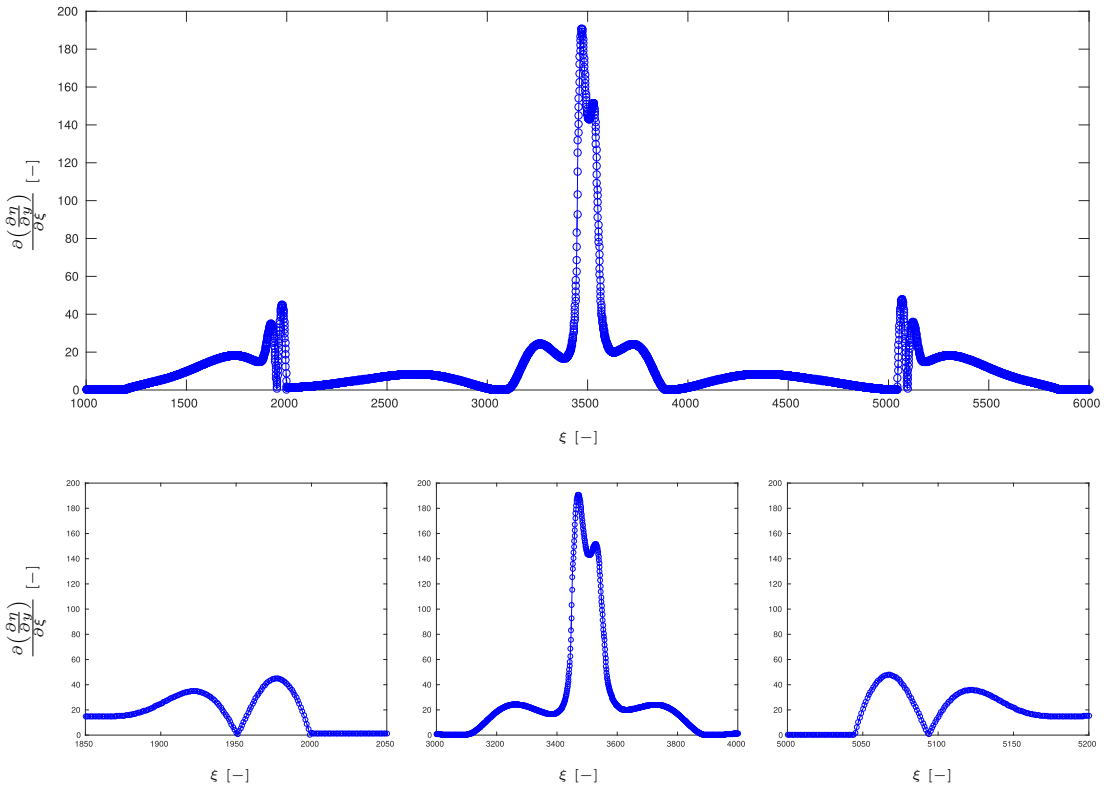

Figure 9: The upper plot shows the evolution of $(\partial\eta/\partial y)/\partial\xi$ along the wall-gridline of the grid around a V2C profile. The lower plots show close-ups of the sections close to the corners (left- and right-hand side plots) and around the leading edge (centre plot).

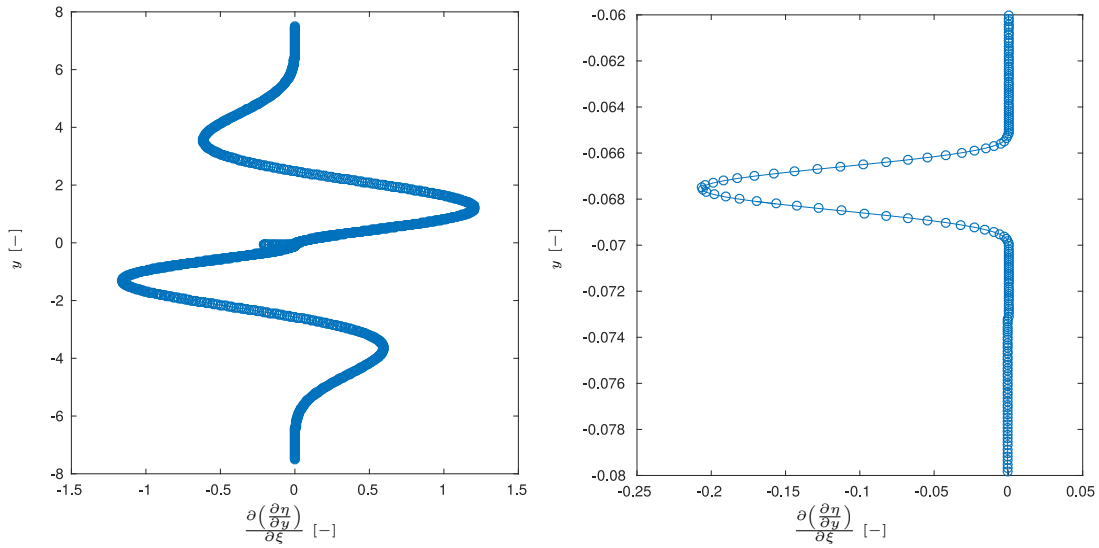

Figure 10: Evolution of $(\partial\eta/\partial y)/\partial\xi$ as a function of y along the η-gridline going through the trailing edge. The right hand side plot shows a close-up of the trailing-edge section.

version is planned to be re-written in Python, including supporting features for the design process of polynomial grid-point distributions.

The published beta version is able to read in files containing the raw geometry of an airfoil with blunt or to sharp trailing edges, or generate NACA four-digit airfoil surfaces (the default is a NACA4412 profile with a sharp trailing edge). A guide-line is provided at the top of the source code. Some sections require several iterations to optimise the grid. It is recommended to read through all comments and try to understand the main operation of each section.

Examples of grids with blunt and sharp trailing edge are included in the release and were tested for direct numerical 2D simulations using the legacy Fortran version of SBLI[7], and OpenSBLI[6] on CPU and GPU computing architectures, respectively. The Fortran script `BuildGridBinary.f` as well as the Python script `convert_hdf5.py` build the binary grid files for SBLI and OpenSBLI, respectively.

4 Conclusions

A grid generator for structured multi-block high-fidelity grids has been developed. The presented approach is straight-forward and has benefits regarding the scalability, flexibility and precision of large grids. The grid can be subdivided into arbitrary sections (here for example $C1 - C4$ of the airfoil), and discretised by piecewise continuous polynomials, which allows target-oriented local grid refinement. The precision (order of continuity of metric terms) of the grids can be easily increased by increasing the order of polynomials that are used to define spacing-functions, blending-kernels and gridlines. Memory requirements hardly increase with the size of the grid.

A beta version of the grid generator is available (`https://github.com/ZaunerM/PolyGridWizZ`) that can generate grids with continuous metric terms up to second order of derivatives, for airfoils with sharp as well as blunt trailing edges. This code is considered as a proof of concept rather than a fully-developed software-package and currently requires user understanding in order to fully exploit its potential. The source-code contains all the important information for the grid and can be archived in a cheap way and modified at a later stage. A future fully-parallelised version is planned to be written in Python, including

features supporting the design process of polynomial grid-point distributions. Proper software-engineering can significantly improve the user-friendliness and performance of the software package.

All parameters, including the nomenclature, are summarised in appendix B.

Acknowledgements

We acknowledge ARCHER (Leadership grant entitled "Transonic flow over an aerofoil"), EPSRC (grant EP/M022692/1 entitled "Unsteady aerodynamics of wings in extreme conditions"), UK Turbulence Consortium (grant EP/L000261/1) as well as the Iridis cluster for the computational resources. MZ was supported by European Commission Horizon 2020 project grant entitled "ExaFLOW: Enabling Exascale Fluid Dynamics Simulations" (671571).

Appendix A

This appendix gives an example for optimising the distribution of grid points along η-gridlines. Figure A.1 shows a screenshot of the Matlab^{TM} grid-generation code with all sections collapsed. In order to generate the default grid, one can simply run the whole code. It will stop before line 2145. In order to write out the grid files for block 2, the sections starting at line 2223 need to be executed. After that, depending on a blunt or sharp trailing edge, the sections beginning at lines 2293 or 2563 need to be executed, respectively. The default grid is generated for a NACA4412 with a sharp trailing edge. The sections, between line 1539 and 1901 (framed blue in figure A.1) define the distribution of grid points along η-gridlines at three locations (trailing edge, half chord and leading edge). Three options for the distributions grid points are available at line 1586 (`Style=1 % <- Choose Version 1/2/3`). Starting with `Style=1`, the grid points are allocated using a single sixth-order polynomial, which is shown by the blue line in figure A.2. Figure A.2(A) shows the spacing between grid points as a function of η along the η-gridline originating at the leading edge ($\eta = 0$) and ending at the outer boundary ($\eta = 689$). While option `Style=1` allows the control parameter (second derivative of the spacing) to be set at the wall, option `Style=2` enables fine-tuning at the outer boundary. It is recommended in both cases to set the control parameter to zero, so that uniform spacing can be maintained close to the boundaries.

As the boundary layer around the leading edge is generally thin, one might want to reduce the grid spacing in that region. On the other hand, smaller cells ahead of the stagnation point can affect the time step significantly. In order to increase the resolution of the boundary layer without limiting the time step, the stretching near the wall needs to be increased. This can be done easily by splitting the η-gridlines in two. Selecting `Style=3` at line 1586, a preliminary distribution is calculated according to option `Style=2`. As a next step, one needs to set the flag at line 1834 to `suggest='t'` so that all derivatives are calculated at the interface between the two parts of the η-gridline, based on the preliminary distribution function. The location of the interface can be either set corresponding to a specific wall distance (`control_dist`) or η (`index`). Those derivatives of the preliminary distribution function describe the boundary conditions for the two polynomials of the divided η-gridline. If no other changes are made, the distributions of grid points along the η-gridline for option `Style=3` correspond to option `Style=2`.

As we only want to modify the gridlines around the leading edge, we start with the corresponding section between $1785 - 1890$. The interface between the two segments (denoted by `index`) is set at $\eta = 125$, corresponding to a wall distance of about $n \approx 0.05$. To increase the stretching at the wall for the preliminary function, we reduce the spacing at the outer boundary by 50% (`prof_deriv1_top=prof_deriv1_top3*0.5`) and obtain a distribution function according to the grey line in figure A.2(A). As a next step, the derivatives are calculated at the interface. Setting the draft flag to `draft='t'`, we can reset the spacing at the outer boundary to its original value and reduce the wall spacing by 90%. As soon as the new input file is generated (that is done in the last section at line 1990), the `suggest` and `draft` flags need to be set `'f'`. Otherwise, the wall spacing at the leading edge will be reduced every time the code is executed. Another option would be to reset the spacings at the wall and outer boundary at the beginning of the code in section `I2`. Calculating both polynomial distribution functions for both segments gives us the red lines in figure A.2. The grid spacing increases rapidly within the first 125 points and then maintains a more or less uniform spacing for the next 200 points, before it increases again towards the outer boundaries. It could be fine-tuned by optimising

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  PolyGridWizZ (Polynomial Gridgeneration Wizard)   %%%
%%%                 (Beta-version 0.0.1             )  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%    Developed by Markus Zauner and Neil D. Sandham  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%              UNIVERSITY OF SOUTHAMPTON             %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%    Corresponding author:      Markus Zauner        %%%
%%%                           m.zauner@soton.ac.uk     %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  developed for Matlab 8.6.0.267246 (R2015b)        %%%
%%%     tested for Matlab 9.4.0.813654 (R2018a)        %%%
%% INTRODUCTION%%...%%
%% Guide:%%...%%
%% Reset everything & calc Carpenter coefficients%%...%%
%% P1: Read-in settings          (User input required)%%...%%
%% P2: Setting angle of attack   (User settings required)%%...%%
%% P3: Basic domain settings%%...%%
%% I1: Read input (Very important! -> Main input)%%...%%
%% S1: Save input-file%%...%%
%% I2: Additional user-input (Always check)%%...%%
%% Start calculation of airfoil surface
%%   Discretisation of trailing edge (TE) %%...%%
%%   Contour 1%%...%%
%%   Contour 2%%...%%
%%   Contour 3%%...%%
%%   Contour 4%%...%%
%%   Define wall-angels of eta-gridlines <- change blending-speed by modifying 'fac'%%...%%
%%   Merge sections and check contour (no user input needed)%%...%%

%% Calculate transition region downstream of the airfoil
%%   Create shape of airfoil extension (Needs a bit of playing and depends strongly on the a
%%   Discretisation of airfoil extension and corresponding outer domain bound%%...%%
%% Define angels so that the eta-gridlines transition to vertical downstream of the airfoil
%% Calculate gridlines at the outer boundary of the C-block
%%   Top of Contour 1%%...%%
%%   TOP of Contour 2%%...%%
%%   TOP of Contour 3%%...%%
%%   TOP of Contour 4%%...%%

%% Merging zones for generating Block 2 (No input needed)%%...%%

%% Define polynomial describing boundary conditions for eta-gridlines
%%   Define and Test Eta-Gridline at TE%%...%%
%%   Saving Parameters %%...%%
%%   Define and Test Eta-Gridline at Center%%...%%
%%   Saving Parameters %%...%%
%%   Define and Test Eta-Gridline at LE%%...%%
%%   Saving Parameters %%...%%
%%   EPS FIGURES in order to compare grids (just uncomment, if needed)%%...%%
%%   Generate Distribution function (no input needed)%%...%%
%% Final preparation for creating Blocks %%...%%
%% Loop trough every point in xi-direction (This can take a while)
%%      Test Loop for Block 2 %%...%%
%%      Final loops
%%          Generate Block 2%%...%%
%%          Generate Block 1 and 3 (blunt)%%...%%
%%          Generate Block 1 and 3 (sharp)%%...%%
%% Finished
```

Figure A.1: Screenshot of the Matlab^{TM} code (collapsed sections).

the number of grid points or the derivatives at the interface. Comparing both curves in figure A.2(B), shows that the number of grid points within $n < 0.02$ increases by almost a third for the new distribution function, whereas the grid is significantly coarsened in the incoming freestream region that tends to limit the simulation time step. As a last step, the position of the interface (`index`) needs to be changed to the same value at the center and trailing edge.

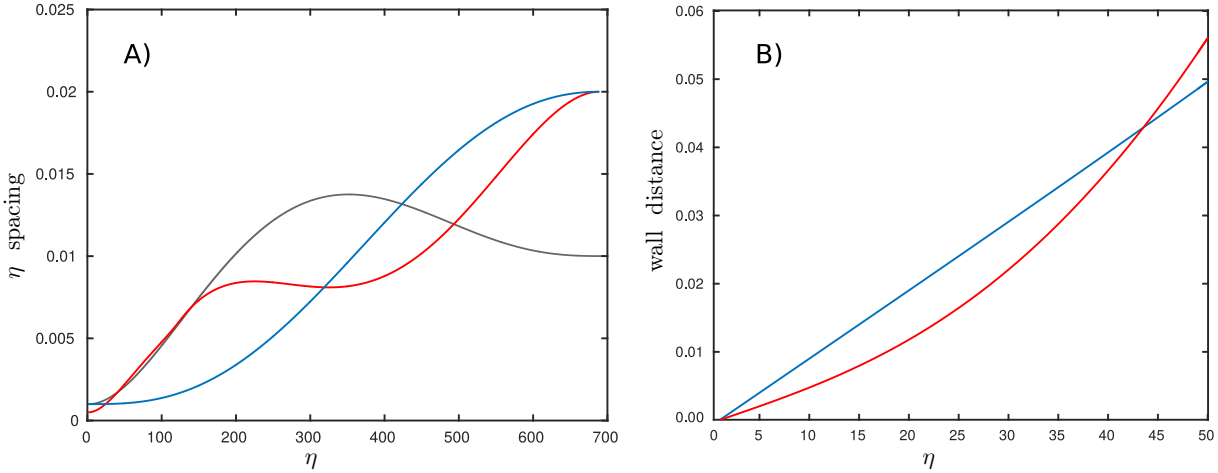

Figure A.2: The spacing between grid points (A) and the wall distance of grid points (B) along the η-gridline at the leading edge are shown as a function of η. The blue and red lines denote distribution functions using `Style=1` and `Style=3`, respectively. The grey line in plot (A) shows the preliminary distribution that is calculated in order to obtain the first part of the red line.

Appendix B

This appendix summarises all control parameters. Table B.1 lists all parameters to specify the discretisation in the ξ-direction along the airfoil surface for the trailing edge (TE) and section 1 ($C1$) to section 4 ($C4$). The $\times$-symbols indicate the grid segments that are influenced by each parameter. Besides the airfoil sections $C1 - C4$, some parameters also influence the airfoil extension (AE), for example. The discretisation of the trailing edge also defines the wall spacing of the η-gridlines at the corners of the trailing edge. Symbols with asterisks denote fine-tuning parameters. In the case of 6^{th}-order polynomials only one fine-tuning parameter can be used per segment. The second column also lists the names of corresponding variables in the open-source code. Part 1 of table B.2 shows all parameters necessary to discretise the outer boundaries of $B2$, whereas part 2 lists all parameters required by the airfoil extension. The outflow boundaries of $B1$ and $B3$ are denoted by OF. Table B.3 shows the parameters needed to define an η-gridline (two segments). The first section contains the number of grid points in both segments (near- and far-field) and the wall distance of the interface ($n_c = 0.7$ for grids in [7]). The second, third, and fourth sections denote the spacing in the η-direction and derivatives of the spacing. The spacing-parameters can be defined for several locations separately. Each location is assigned a number. In the provided version of the code, the parameters are defined at the trailing edge (#= 1), half chord (#= 2) and leading edge (#= 3). Locations (#= 1) and (#= 2) are the same for suction as well as pressure side. It should be borne in mind that the wall-spacing at the trailing-edge corners is defined by the resolution of the blunt trailing edge.

Param.	Code	$C1$	$C3$	$C4$	$C2$	TE	AE	η_{TE}	note
N_{TE}	NTEw					x		x	Resolution of blunt trailing edge
N_{C1}	Nc1	x							No. of grid points in section $C1$
Δs_{TEu}	dsTEu	x					x		ξ-spacing ($\frac{\partial s}{\partial \xi}$) at upper TE
$\Delta s'_{TEu}$	ddsTEu	x					x		$\frac{\partial^2 s}{\partial \xi^2}$ at upper TE
$\Delta s''_{TEu}$	dddsTEu	x*							$\frac{\partial^3 s}{\partial \xi^3}$ at upper TE
Δs_{Cu}	dscu	x	x						ξ-spacing ($\frac{\partial s}{\partial \xi}$) at upper half-chord
$\Delta s'_{Cu}$	ddscu	x	x						$\frac{\partial^2 s}{\partial \xi^2}$ at upper half-chord
$\Delta s''_{Cu}$	dddscu	x*							$\frac{\partial^3 s}{\partial \xi^3}$ at upper half-chord
φ_{Cu}	phi_center1	x	x						Wall angle at upper half-chord
N_{C3}	Nc3		x						No. of grid points in section $C3$
Δs_{LE}	dsLE		x	x					ξ-spacing ($\frac{\partial s}{\partial \xi}$) at LE
$\Delta s'_{LE}$	ddsLE		x	x					$\frac{\partial^2 s}{\partial \xi^2}$ at LE
$\Delta s''_{LE}$	dddsLE		x*						$\frac{\partial^3 s}{\partial \xi^3}$ at LE
N_{C4}	Nc4			x					No. of grid points in section $C4$
Δs_{Cl}	dscl			x	x				ξ-spacing ($\frac{\partial s}{\partial \xi}$) at lower half-chord
$\Delta s'_{Cl}$	ddscl			x	x				$\frac{\partial^2 s}{\partial \xi^2}$ at lower half-chord
$\Delta s''_{Cl}$	dddscl			x*					$\frac{\partial^3 s}{\partial \xi^3}$ at lower half-chord
φ_{Cl}	phi_center2			x	x				Wall angle at lower half-chord
N_{C2}	Nc2				x				No. of grid points in section $C2$
Δs_{TEl}	dsTEl				x		x		ξ-spacing ($\frac{\partial s}{\partial \xi}$) at lower TE
$\Delta s'_{TEl}$	ddsTEl				x		x		$\frac{\partial^2 s}{\partial \xi^2}$ at lower TE
$\Delta s''_{TEl}$	dddsTEl				x*				$\frac{\partial^3 s}{\partial \xi^3}$ at lower TE

* optional (only one per section can be used)

Table B.1: Summary of the input parameters required to mesh the airfoil surface. The first and second columns list the parameters and their name in the code, respectively, whereas columns $3-7$ show, which sections (according to figure 3) are influenced by each parameter. Column 8 and 9 indicate whether the parameter impacts the airfoil extension in $B1$ and $B3$ or the discretisation of the η-gridlines (here denoted by η_{TE}), respectively.

Param.	Code	C1	C3	C4	C2	AE	OF	note
R	rad	x	x	x	x			Radius of semicircle defining the C-block
Δs_{TTu}	dsTEut	x				x		ξ-spacing ($\frac{\partial s}{\partial \xi}$) at upper corner of $B2$
$\Delta s'_{TTu}$	ddsTEut	x				x		$\frac{\partial^2 s}{\partial \xi^2}$ at upper corner of $B2$
$\Delta s''_{TTu}$	dddsTEut	x*						$\frac{\partial^3 s}{\partial \xi^3}$ at upper corner of $B2$
Δs_{TCu}	dscut	x	x					ξ-spacing ($\frac{\partial s}{\partial \xi}$) at upper half-chord
$\Delta s'_{TCu}$	ddscut	x	x					$\frac{\partial^2 s}{\partial \xi^2}$ at upper half-chord
$\Delta s''_{TCu}$	dddscut	x*						$\frac{\partial^3 s}{\partial \xi^3}$ at upper half-chord
Δs_{TLE}	dLE_tan		x	x				ξ-spacing ($\frac{\partial s}{\partial \xi}$) at $(x/y) = (-7/0)$
$\Delta s'_{TLE}$	ddLE_tan		x	x				$\frac{\partial^2 s}{\partial \xi^2}$ at $(x/y) = (-7/0)$
$\Delta s''_{TLE}$	dddLE_tan		x*					$\frac{\partial^3 s}{\partial \xi^3}$ at $(x/y) = (-7/0)$
Δs_{TCl}	dsclt				x			ξ-spacing ($\frac{\partial s}{\partial \xi}$) at upper half-chord
$\Delta s'_{TCl}$	ddsclt				x			$\frac{\partial^2 s}{\partial \xi^2}$ at upper half-chord
$\Delta s''_{TCl}$	dddsclt			x*				$\frac{\partial^3 s}{\partial \xi^3}$ at upper half-chord
Δs_{TTl}	dsTElt				x	x		ξ-spacing ($\frac{\partial s}{\partial \xi}$) at lower corner of $B2$
$\Delta s'_{TTl}$	ddsTElt				x	x		$\frac{\partial^2 s}{\partial \xi^2}$ at lower corner of $B2$
$\Delta s''_{TTl}$	dddsTElt				x*			$\frac{\partial^3 s}{\partial \xi^3}$ at lower corner of $B2$
$N3_{\varphi,t}$	N3tphic			x				Control Point for approximating angles at outer boundaries
$N4_{\varphi,t}$	N4tphic				x			Control Point for approximating angles at outer boundaries
$\varphi_{t3,c}$	ddphi3t_LEc			x				Second derivative of the boundary angle at N3tphic
$\varphi_{t4,c}$	ddphi4t_LEc				x			Second derivative of the boundary angle at N4tphic
N_{out}	Nw						x	Tot. no. of points in ξ of $B1$ and $B3$
N_{blend}	Nbuffer					x	x	No. of points for airfoil extension
$dist_{blend}$	xTEc_save					x	x	Approx. length of the transition region
N_k	Nk					x		Speed up transition to horiz. grid lines
F_{div}	StretchW					x	x	Divergence factor of the wake-region**
Δs_{wt}	dsTEc					x	x	ξ-spacing at the end of the transition
$\Delta s'_{wt}$	ddsTEc					x	x	$\frac{\partial^2 s}{\partial \xi^2}$ at the end of the transition
$\Delta s''_{wt}$	dddsTEc					x		$\frac{\partial^3 s}{\partial \xi^3}$ at the end of the transition
x_{out}	xOut						x	x-position of outlet
Δs_{out}	dsOut						x	ξ-spacing at outlet
$\Delta s'_{out}$	ddsOut						x	$\frac{\partial^2 s}{\partial \xi^2}$ at outlet
$\Delta s''_{out}$	dddsOut						x	$\frac{\partial^3 s}{\partial \xi^3}$ at outlet

* optional

** Stretching factor that causes the upper and lower airfoil downstream continuation-lines to diverge

Table B.2: Summary of the input parameters required to mesh the outer boundaries (part 1) and to define the airfoil-extension (part 2). The first two columns list the parameters and code variables, whereas column $3-8$ show which grid sections, corresponding to the segmentation of the airfoil surface (figure 3), are influenced by each parameter. Column 8 and 9 indicate whether the parameter impacts the grid transition region in the wake of $B1$ and $B3$ (AE) or the outflow region (OF), respectively.

Param.	Code	note
N_{NF}	Ny1	No. of grid points in the nearfield
N_{FF}	Ny2	No. of grid points in the farfield
n_c	control_dist#	Wall distance of nearfield boundary at TE
$N_{const,1}$	const_line	Force number of points being on wall-optimised η-line
$N_{const,2}$	const_line2	Force number of points being on OB-optimised η-line
$N_{blend,LE}$	blend_C	Points left & right of LE used for blending ϕ_{OB}
Δn_w	prof_deriv1_wall#	Δn at the airfoil surface
Δn_c	prof_deriv1_cont#	Δn at the nearfield interface
Δn_b	prof_deriv1_top#	Δn at the outer boundary
$\Delta n'_w$	prof_deriv2_wall#	$\frac{\partial^2 n}{\partial \eta^2}$ at the airfoil surface
$\Delta n'_c$	prof_deriv2_cont#	$\frac{\partial^2 n}{\partial \eta^2}$ at the nearfield interface
$\Delta n'_b$	prof_deriv2_top#	$\frac{\partial^2 n}{\partial \eta^2}$ at the outer boundary
$\Delta n'_b$	prof_deriv2_top#	$\frac{\partial^2 n}{\partial \eta^2}$ at the outer boundary
$\Delta n''_w$	prof_deriv3_wall#	$\frac{\partial^3 n}{\partial \eta^3}$ at the airfoil surface
$\Delta n''_c$	prof_deriv3_cont#	$\frac{\partial^3 n}{\partial \eta^3}$ at the nearfield interface
$\Delta n''_b$	prof_deriv3_top#	$\frac{\partial^3 n}{\partial \eta^3}$ at the outer boundary

Table B.3: Summary of the input parameters required to generate η-gridlines. The parameters are defined for several axial chord positions. A number (represented by the #-symbol) is assigned to each position.

References

[1] S. Bourdet, A. Bouhadji, M. Braza, and F. Thiele. Direct Numerical Simulation of the Three-Dimensional Transition to Turbulence in the Transonic Flow around a Wing. *Flow, Turbulence and Combustion (formerly Applied Scientific Research)*, 71(1-4):203–220, 2003, doi:10.1023/B:APPL.0000014932.28421.9e.

[2] M. Gageik, I. Klioutchnikov, and H. Olivier. Comprehensive mesh study for a Direct Numerical Simulation of the transonic flow at Rec = 500,000 around a NACA 0012 airfoil. *Computers and Fluids*, 122:153–164, 2015, doi:10.1016/j.compfluid.2015.08.030.

[3] L. Jones, R. Sandberg, and N. Sandham. Direct numerical simulations of forced and unforced separation bubbles on an airfoil at incidence. *Journal of Fluid Mechanics*, 602:175–207, 2008, doi:10.1017/S0022112008000864.

[4] S. M. Hosseini, R. Vinuesa, P. Schlatter, A. Hanifi, and D. S. Henningson. Direct numerical simulation of the flow around a wing section using high-order parallel spectral methods. *International Journal of Heat and Fluid Flow*, 16, 2016, doi:10.1016/j.ijheatfluidflow.2016.02.001.

[5] P. Balakumar. Direct Numerical Simulation of Flows over an NACA-0012 Airfoil at Low and Moderate Reynolds Numbers. *47th AIAA Fluid Dynamics Conference*, (June):1–19, 2017, doi:10.2514/6.2017-3978.

[6] D. J. Lusher, S. P. Jammy, and N. D. Sandham. Shock-wave/boundary-layer interactions in the automatic source-code generation framework OpenSBLI. *Computers & Fluids*, 0:1–5, 2018, doi:10.1016/j.compfluid.2018.03.081.

[7] M. Zauner, N. De Tullio, and N. D. Sandham. Direct numerical simulations of transonic flow around an airfoil at moderate Reynolds numbers. *Submitted to the AIAA Journal*, 2018.

[8] C. T. Jacobs, M. Zauner, N. De Tullio, S. P. Jammy, D. J. Lusher, and N. D. Sandham. An error indicator for finite difference methods using spectral techniques with application to aerofoil simulation. *Computers & Fluids*, 168:67–72, 2018, doi:10.1016/j.compfluid.2018.03.065.

[9] M. Zauner, C. T. Jacobs, and N. D. Sandham. Grid refinement using spectral error indicators with application to airfoil DNS. In *Submitted to ECCM-ECFD Conference proceedings*, Glasgow, 2018.
[10] C. de Boor. A Practical Guide to Splines. *Mathematics of Computation*, 34(149):325, 1980, doi:10.2307/2006241.
[11] M. Drela. XFOIL: An analysis and design system for low Reynolds number airfoils, 1989.
[12] M. Zauner and N. D. Sandham. PolyGridWizZ Beta-version 0.0.1. 2018, doi:10.5281/ZENODO.1245598.