Shape Optimization of Wind Turbine Blades using the Continuous Adjoint Method and Volumetric NURBS on a GPU Cluster

Konstantinos T. Tsiakas*, Xenofon S. Trompoukis, Varvara G. Asouti, Kyriakos C. Giannakoglou
Parallel CFD & Optimization Unit, School of Mechanical Engineering
National Technical University of Athens, Greece
Email: tsiakost@gmail.com, xeftro@gmail.com, vasouti@mail.ntua.gr, kgianna@central.ntua.gr

Summary

This paper presents the development and application of the continous adjoint method for the shape optimization of wind turbine blades for maximum power output. A RANS solver, which includes the Spalart–Allmaras turbulence model, is the flow (primal) model based on which the adjoint system of equations is derived. The latter includes the adjoint to the turbulence model PDE. The solution of the primal and adjoint equations provides the objective function gradient w.r.t. the design variables. A volumetric Non-Uniform Rational B–Splines (NURBS) model is used to parameterize the shape to be designed. In addition, the same tool provides the means to accordingly deform the computational mesh. In order to reduce the computational cost, the aforementioned tools, developed in the CUDA programming environment, run on a cluster of Graphics Processing Units (GPUs) using the MPI protocol. Optimized GPU memory handling and GPU dedicated algorithmic techniques make the overall optimization process up to 50x faster than the same process running on a CPU. The developed software is used for the shape optimization of an horizontal axis wind turbine blade for maximum power output.

Keywords: *Shape Optimization, Continous Adjoint Method, Wind Turbines, GPUs, NURBS Morphing, CUDA, MPI*

1 Introduction

The design of wind turbines, and in particular their blade shapes, is a major application field in CFD. Though CFD methods are widely used for the aerodynamic analysis of wind turbines[1], their use in optimization of their bladings is still limited. The major drawback of CFD based optimization is its computational cost, especially when dealing with turbulent flows around complex geometries. The huge meshes needed for the aerodynamic analysis of wind turbine blades make the use of stochastic, population-based optimization methods rather prohibitive. An alternative is the use of gradient-based optimization methods, such as steepest descent or quasi–Newton methods. In such a case, the gradient of the objective function needs to be computed. To this end, the adjoint method can be used and this makes the cost of computing the gradient independent of the number of design variables and approximately equal to that for solving the primal equations.

Over and above to any gain from the use of the less costly methods to compute the objective function gradient, a good way to reduce the optimization turnaround time is by using GPUs. Both the flow and adjoint solvers are ported on GPUs, exhibiting a noticeable speed–up compared to their CPU implementations[2,3]. Though the use of a modern GPU can greatly accelerate CFD computations, its memory capacity is limited compared to a modern CPU RAM, posing a limitation when using GPUs for industrial applications. To overcome this problem, many GPUs, on different computational nodes if necessary, can be used to perform the computation in parallel, by making use of the CUDA environment together with the MPI protocol.

The geometry of wind turbine blades is quite complex, consisting of airfoil profiles varying largely along the spanwise direction. As a result, employing a scheme that parameterizes the exact geometry of the blade and incorporating it within the optimization process is not an easy task. Here, a volumetric NURBS model is used to parameterize the space around the blade rather than the blade itself[4]. This model additionally undertakes mesh deformation, which would have to be carried out by a different method if a direct surface parameterization model was used. The main cost of the parameterization model is the computation of the B–Spline basis functions and their derivatives, which are herein required for the objective function gradient, according to the chain rule. In order to reduce this cost, their computation is also carried out on the GPUs. The Navier-Stokes equations and their adjoint are described in section 2. Section 3 describes the implementation of volumetric NURBS. Section 4 describes the GPU implementation of the primal and adjoint solvers

and the parameterization model and, finally, section 5 presents the application of the developed method and software to a horizontal axis wind turbine blade.

2 Navier-Stokes and Adjoint Equations – Objective function and its gradient

The flow model is based on the incompressible flow equations using the Spalart-Allmaras turbulence model.

2.1 Primal equations

The primal equations are the incompressible Navier-Stokes equations. The pseudo-compressibility approach, as introduced by Chorin[5], is applied. In order to predict the flow around the rotating blades in steady state, a multiple reference frame technique is used, where the equations are solved in a moving frame for the absolute velocity components. The Navier-Stokes equations are

$$R_{U_n} = \frac{\partial f_{nk}^{inv}}{\partial x_k} - \frac{\partial f_{nk}^{vis}}{\partial x_k} + S_n = 0 \tag{1}$$

where $U_n = [p \;\; v_1^A \;\; v_2^A \;\; v_3^A]^T$ is the vector of the state variables, $v_i^A, i = 1,2,3$ are the absolute velocity components and p is the pressure divided by the density. The inviscid and viscous fluxes f_{nk} and source terms $S_n$are given as

$$f_{nk}^{inv} = \begin{bmatrix} \beta v_k^R \\ v_k^R v_1^A + p\delta_{1n} \\ v_k^R v_2^A + p\delta_{2n} \\ v_k^R v_3^A + p\delta_{3n} \end{bmatrix} \qquad f_{nk}^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}$$

$$\tau_{mk} = (\nu + \nu_t)\left(\frac{\partial v_m^A}{\partial x_k} + \frac{\partial v_k^A}{\partial x_m}\right) \quad S_n = \begin{bmatrix} 0 \\ \omega_2 v_3^A - \omega_3 v_2^A \\ \omega_3 v_1^A - \omega_1 v_3^A \\ \omega_1 v_2^A - \omega_2 v_3^A \end{bmatrix} \tag{2}$$

where ν and ν_t stand for the kinematic and turbulent viscosity and ω stands for the blade rotational velocity. In equation 2, v_i^R denote the relative velocity components. The absolute and relative velocity vectors are linked through $v_i^A = v_i^R - v_i^F$ with $v_i^F = \varepsilon_{ijk}\omega_j d_k$ and $d_k = x_k - x_k^C$ are the components of the position vector from the origin which lies on the rotation axis.

Equations 1 are solved together with the Spalart-Allmaras turbulence model[6] PDE in a decoupled time–marching scheme.

2.2 Continuous Adjoint Method Formulation

The selected objective function F is the power output of the wind turbine blading for constant ω. Its maximization is, in fact, equivalent to the maximization of the torque w.r.t. the axis of the wind turbine shaft. If r_k denotes the components of the unit vector aligned with this shaft, F can be expressed as

$$F = \int_{S_{Blade}} \varepsilon_{klm}\left(x_l - x_l^C\right)\left(pn_m - \tau_{mq}n_q\right) r_k dS \tag{3}$$

where S_{Blade} denotes the blade surface. In equation 3, n_q are the components of the unit vector normal to the blade surface and pointing towards the blade.

By introducing the adjoint mean–flow variables Ψ_n $(n = 1,\ldots,4)$ and the adjoint turbulent variable $\tilde{\nu}^a$, the augmented objective function is defined as

$$F_{aug} = F + \int_\Omega \Psi_n R_{U_n} d\Omega + \int_\Omega \tilde{\nu}^a R_{\tilde{\nu}} d\Omega \tag{4}$$

Upon convergence of the primal equations, F_{aug} is equal to F. To compute the variations of F_{aug} w.r.t. the design variables b_i, we start by differentiating equation 4, which yields

$$\frac{\delta F_{aug}}{\delta b_i} = \frac{\delta F}{\delta b_i} + \frac{\delta}{\delta b_i}\int_\Omega \Psi_n R_{U_n} d\Omega + \frac{\delta}{\delta b_i}\int_\Omega \tilde{\nu}^a R_{\tilde{\nu}} d\Omega \tag{5}$$

By developing and eliminating the integrals including the variations in the flow quantities w.r.t. b_i, the field adjoint equations and their boundary conditions arise. All remaining integrals form the expression of the gradient of F w.r.t. b_i. The field adjoint equations read

$$R_{\Psi_n} = -\underbrace{A_{mnk}\frac{\partial \Psi_m}{\partial x_k}}_{Conv(\Psi)} - \underbrace{\frac{\partial \phi_{nk}^{vis}}{\partial x_k}}_{Diff(\Psi)} - \underbrace{S_n^{adj}}_{Source1(\Psi)} + \underbrace{T_n^{adj}}_{Source2(\tilde{\nu}_a)} = 0 \tag{6}$$

with

$$A_{nmk} = \begin{bmatrix} 0 & \beta\delta_{1k} & \beta\delta_{2k} & \beta\delta_{3k} \\ \delta_{1k} & v_1^R + v_1^A\delta_{1k} & v_1^A\delta_{2k} & v_1^A\delta_{3k} \\ \delta_{2k} & v_2^A\delta_{1k} & v_2^R + v_2^A\delta_{2k} & v_2^A\delta_{3k} \\ \delta_{3k} & v_3^A\delta_{1k} & v_3^A\delta_{2k} & v_3^R + v_3^A\delta_{3k} \end{bmatrix} \tag{7}$$

$$S_n^a = \begin{bmatrix} 0 \\ \omega_2\Psi_4 - \omega_3\Psi_3 \\ \omega_3\Psi_2 - \omega_1\Psi_4 \\ \omega_1\Psi_3 - \omega_2\Psi_2 \end{bmatrix} \qquad \phi_{nk}^{vis} = \begin{bmatrix} 0 \\ \tau_{1k}^a \\ \tau_{2k}^a \\ \tau_{3k}^a \end{bmatrix} \tag{8}$$

$$\tau_{mk}^a = (\nu + \nu_t)\left(\frac{\partial \Psi_{m+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_m}\right) \tag{9}$$

where δ_{ij} is the Knonecker's symbol.

In equation 6, the terms marked as $Conv(\Psi)$ and $Diff(\Psi)$ correspond to the adjoint convection and diffusion respectively, $Source1(\Psi)$ corresponds to the adjoint source terms resulting from the frame rotation and $Source2(\tilde{\nu}^a)$ includes the contribution of the adjoint turbulence model to the adjoint mean–flow equations. The derivation of the adjoint turbulence model equation can be found in a

previous work[7] published from the same group and will not be repeated here.

After solving the primal and adjoint equations, $\frac{\delta F}{\delta bi}$ can be computed once the geometric sensitivities $\frac{\delta x_l}{\delta b_i}$ and $\frac{\partial}{\partial x_k}\left(\frac{\delta x_l}{\delta b_i}\right)$ at the mesh nodes become available. The final expression of the sensitivity derivatives reads

$$\begin{aligned}\frac{\delta F}{\delta b_i} = &\int_{S_{Blade}} I_l^{torque}\frac{\delta x_l}{\delta b_i}dS + \int_{S_{Blade}} I_l^{SMF}\frac{\delta x_l}{\delta b_i}dS + \\ &\int_\Omega I_l^{VMF1}\frac{\delta x_l}{\delta b_i}d\Omega + \int_\Omega I_{lk}^{VMF2}\frac{\partial}{\partial x_k}\left(\frac{\delta x_l}{\delta b_i}\right)d\Omega + \\ &\int_\Omega I_l^{VSA1}\frac{\delta x_l}{\delta b_i}d\Omega + \int_\Omega I_{lk}^{VSA2}\frac{\partial}{\partial x_k}\left(\frac{\delta x_l}{\delta b_i}\right)d\Omega + \\ &\int_\Omega I^{VSA3}\frac{\delta \Delta}{\delta b_i}d\Omega \end{aligned} \tag{10}$$

where Δ is the distance from the nearest wall, I_l^{torque} are terms resulting from the differentiation of the objective function, $I^{SMF}, I^{VMF1}, I^{VMF2}$ from the differentiation of the mean–flow equations and $I^{VSA1}, I^{VSA2}, I^{VSA3}$ from the differentiation of the turbulence model equation.

2.3 Discretization and Numerical Solution

The primal and adjoint equations are solved on hybrid meshes (consisting of tetrahedra, pyramids, prisms or hexahedra) using the vertex–centered finite volume method and the time–marching technique. The numerical fluxes crossing the finite volume intefaces are computed with second–order accuracy. The primal inviscid numerical flux crossing the interface between nodes P and Q reads

$$\Phi^{PQ} = \frac{1}{2}\left(f_{nk}^{inv,P} + f_{nk}^{inv,Q}\right)n_k^{PQ} - \frac{1}{2}\left|\overline{A}_{nmk}^{PQ} n_k\right|\left(U_m^R - U_m^L\right)$$

where n_k^{PQ} are the components of the unit vector normal to the finite volume interface between nodes P and Q and pointing to node Q and the jacobian $\overline{A}^{PQ}$ is computed based on the Roe–averaged[8] flow variables. U^R and U^L are the flow variables on the right and left sides of the finite volume interface, obtained by extrapolating U^Q and U^P respectively.

The adjoint inviscid numerical fluxes are computed using a non–conservative scheme.

$$\Phi_n^{adj,PQ} = -\frac{1}{2}A_{mnk}^P\left(\Psi_n^P + \Psi_n^Q\right)n_k - \frac{1}{2}\left|\overline{A}_{mnk}^{PQ} n_k\right|\left(\Psi_n^R - \Psi_n^L\right)$$

$$\Phi_n^{adj,QP} = \frac{1}{2}A_{mnk}^Q\left(\Psi_n^P + \Psi_n^Q\right)n_k + \frac{1}{2}\left|\overline{A}_{mnk}^{PQ} n_k\right|\left(\Psi_n^R - \Psi_n^L\right)$$

Thus, the adjoint flux entering the finite volume of node P is not the same as the flux exiting the finite volume of node Q.

For the computation of the viscous fluxes, the derivatives of any primal flow or adjoint quantity W on the finite volumes interface (between nodes P and Q) are computed as

$$\left(\frac{\partial W}{\partial x_k}\right)^{PQ} = \overline{\left(\frac{\partial W}{\partial x_k}\right)} - \left[\overline{\left(\frac{\partial W}{\partial x_k}\right)}t_m^{PQ} - \frac{W^Q - W^P}{\sqrt{(x_m^Q - x_m^P)^2}}\right]t_k^{PQ} \tag{11}$$

where

$$\begin{aligned} t_m^{PQ} &= \frac{x_m^Q - x_m^P}{\sqrt{(x_m^Q - x_m^P)^2}} \\ \overline{\left(\frac{\partial W}{\partial x_k}\right)} &= \frac{1}{2}\left[\left(\frac{\partial W}{\partial x_k}\right)^P + \left(\frac{\partial W}{\partial x_k}\right)^Q\right] \end{aligned} \tag{12}$$

The discretized equations are linearized and solved iteratively w.r.t. the correction of the primal/adjoint variables (delta formulation) using a point–implicit Jacobi method.

3 Parameterization through volumetric NURBS

Volumetric NURBS are rational trivariate (in 3D) B–Splines defined on non-uniform knot vectors, used to parameterize the volume around the blade. Let (ξ,η,ζ) be the three parametric directions and X_m^{ijk} and w^{ijk} denote the (ijk)th control point coordinates and weight. Given the parameteric coordinates of a point as well as the knot vectors and control points coordinates/weights, its physical coordinates $x_m (m = 1,2,3)$ can be computed as

$$x_m(\xi,\eta,\zeta) = \frac{\sum_i^{N_\xi}\sum_j^{N_\eta}\sum_k^{N_\zeta}\Xi_{i,p_\xi}(\xi)H_{j,p_\eta}(\eta)Z_{k,p_\zeta}(\zeta)X_m^{ijk}w^{ijk}}{\sum_i^{N_\xi}\sum_j^{N_\eta}\sum_k^{N_\zeta}\Xi_{i,p_\xi}(\xi)H_{j,p_\eta}(\eta)Z_{k,p_\zeta}(\zeta)w^{ijk}} \tag{13}$$

where, Ξ_{i,p_ξ} is the ith B-Spline basis function of degree p_ξ defined on the knot vector $K_\xi = \{\xi_0,\ldots,\xi_{m_\xi}\}$ (H_{j,p_η} and Z_{k,p_ζ} are defined similarly), N_ξ is the number of control points in the ξ direction and it must hold that $m_\xi = N_\xi + p_\xi + 1$[9]. Knots must be arranged in a non-decreasing order.

Specifying the control points, weights and knot vectors, a point inversion, via the Newton-Raphson method, is used to calculate the parametric coordinates of the mesh nodes. The so–computed parametric coordinates as well as the knot vectors remain fixed during the optimization. All variations of geometric quantities, such as $\frac{\delta x_l}{\delta b_i}$ and $\frac{\partial}{\partial x_k}\left(\frac{\delta x_l}{\delta b_i}\right)$, involved in the computation of the objective function gradient can be computed by using closed–form expressions resulting from the differentiation of equation 13.

During the optimization loop, the control point coordinates and weights are updated and equation 13 is used to provide the deformed computational mesh and blade shape.

4 Implementation on GPUs

Nowadays, GPUs have become powerful parallel co–processors to CPUs, offering more than one order of magnitude more floating point operations per second (FLOPS) with lower memory latency compared to modern CPUs.

Although the GPU hardware capabilities are superior to the CPU ones, directly porting a CPU code on GPU does not necessarily yields the desired high speed-ups, due to different architecture features. The Navier-Stokes/adjoint equations solver this paper makes use of, efficiently exploits the high computing capabilities that modern GPUs have, running on a GPU at least 50 times faster than the equivalent CPU solver. Such high parallel efficiency mainly results from (a) the use of Mixed Precision Arithmetics (MPA), which allows the l.h.s. matrices to be computed using double-precision and stored using single-precision arithmetics[2], without harming the accuracy of the solver and (b) the minimization of random accesses to the relatively high latency device memory by concurrently running threads.

For maximum speed–up, the primal and adjoint solvers employ different algorithmic techniques for the computation of the nodal residuals and l.h.s. coefficients. In previous work of the authors[3], it is shown that, when processsing large amount of data on a GPU, minimizing memory usage and non–coalesced memory accesses is more important than minimizing the number of (rather redundant) re–computations of the same quantity. Thus, the primal solver, in which the memory consuming Jacobians per finite volume interface need to be computed for the l.h.s. coefficients at each pseudo–iteration, uses a one–kernel scheme. According to this scheme, a single kernel is launched, associating each GPU thread with a mesh node. Each thread computes and accumulates the numerical fluxes crossing all boundaries of this node's finite volume and their Jacobians and, thus, forms residuals and l.h.s. coefficients. On the contrary, since for the solution of the adjoint equations the l.h.s. coefficients depend only on the primal solution field, the Jacobians are computed once, before the iterative solution of the adjoint equations. Thus, the adjoint solver employs a two–kernel scheme in which the less memory consuming adjoint numerical fluxes are computed by the first kernel (which associates GPU threads with finite volume interfaces) and accumulated by the second kernel (which associates GPU threads with mesh nodes).

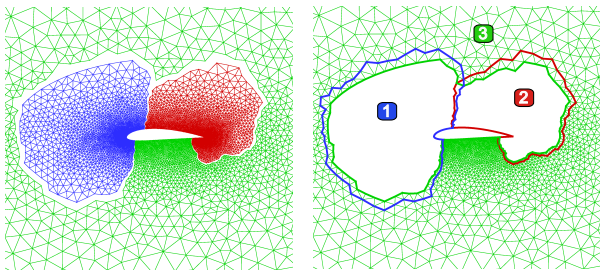
The primal/adjoint solvers run on a cluster of GPUs. In order to run a case in many GPUs, the mesh is partitioned in overlapped subdomains and each subdomain is associated with one GPU. For instance, figure 1(a) shows a triangular mesh generated around an isolated airfoil partitioned in three overlapped subdomains. The shared regions of the mesh subdomains are marked in white in figure 1(a). The whole mesh (i.e. including the overlapped regions) of the 3rd subdomain, with the boundaries shared with subdomains 1, 2, can be seen in figure 1(b). To further reduce the wall–clock time, computations and data transfers overlap. For instance, when computing the primal/adjoint spatial gradients, each GPU associated with a subdomain performs the same sequence of steps. As an example, the GPU associated with the 3rd subdomain performs the following steps:

Step A: launches a kernel only for the computation of the gradients at the nodes interface with subdomains 1 and 2 (i.e. nodes lying on the blue and red lines of figure 1(b)),

Step B: performs the data interchange between the subdomains (asssigned to different GPUs) and

Step C: launches a kernel for the computation of the gradients at the remaining nodes of the subdomain.

Steps A, B can be performed simultaneaously with step C so that computations and data transfers overlap. Data transfers among GPUs on different computational nodes use the MPI protocol. The communication of GPUs on the same node is performed through the shared (on–node) CPU memory.



(a) (b)

Figure 1: Mesh with triangular elements around an isolated airfoil partitioned in three overlapped subdomains.

The computations of the parametric coordinates of the mesh nodes and the objective function gradients, which are computationally intensive and memory demanding, also run on the GPUs. Since $\frac{\delta x_l}{\delta b_i}$, which is needed for $\frac{\delta F}{\delta b_i}$, are geometric quantities independent of the primal/adjoint solution, they could be computed and stored once. However, the memory needed for storing $\frac{\delta x_l}{\delta b_i}$ often exceeds that required for the solution of the primal and adjoint equations. Hence, their storage is avoided and they are re–computed at the end of each optimization cycle using pre–allocated GPU memory.

The optimization algorithm is shown, as a flowchart, in figure 2. Steps performed exclusively on CPU or GPU are clearly marked. Expensive processes associated with the computation/update of the mesh geometrical data, such as computing node distances from the nearest wall, are performed on the GPU, while others such as computing the cells volumes are performed at the same time on the CPU. Thus, all available computing resources are exploited

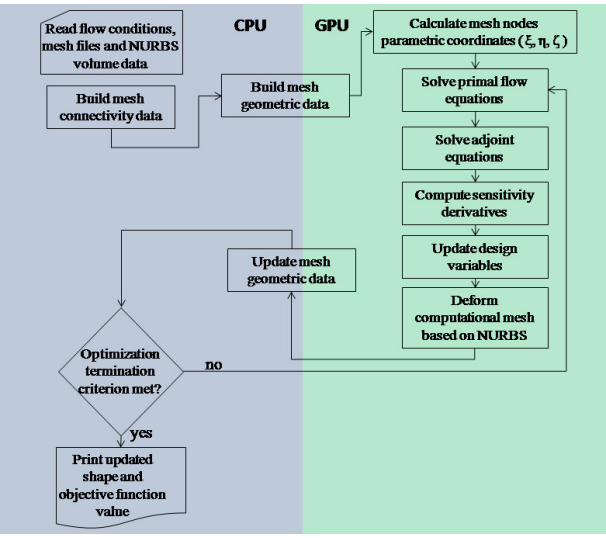


Figure 2: Flowchart demonstrating the optimization algorithm steps. Steps performed on the CPU and the GPU are distinguished.

and the wall clock time needed to perform these tasks is reduced.

5 Optimization of the Wind Turbine Blade

This software was used for the shape optimization of the MEXICO[10] horizontal axis wind turbine (HAWT) blade for maximum power output, when operating with $10\frac{m}{s}$ farfield velocity and 0° yaw. The mesh consists of about 2.5×10^6 nodes. The primal/adjoint solver run on 4 NVIDIA Kepler K20 GPUs, lying on two different nodes. For the parameterization of the blade, a NURBS control volume is used, as shown in figure 3. The computationally expensive steps of solving the primal and adjoint field equations need approximately $15min$ and $10min$ per optimization cycle, respectively.

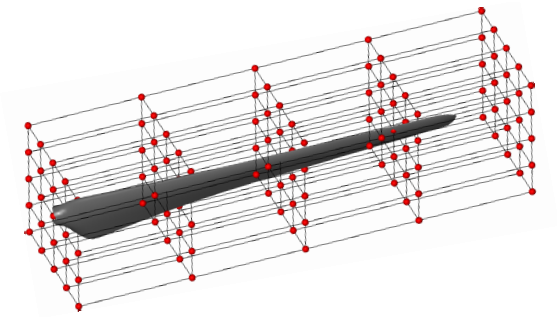


Figure 3: Parameterization of the HAWT blade.

The pressure coefficient distribution on the blades is shown in figure 4, while relative velocity streamlines in the tip vortex region are plotted in figure 5. Figure 6 shows a comparison of the chordwise distribution of the pressure coefficient with experimental results measured in the MEXICO study[10].

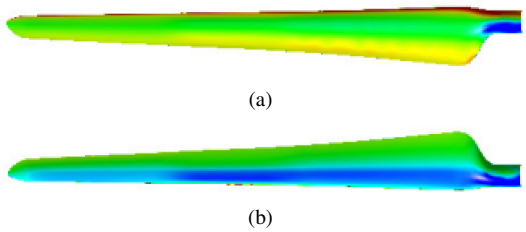


Figure 4: Pressure coefficient (c_p) distribution on the blade's pressure (a) and suction (b) side. The pressure coefficient is defined as $c_p = \frac{p-p_{far}}{\frac{1}{2}V_{far}^2+\omega^2R^2}$, R being the local radius and far indexing farfield flow quantities.

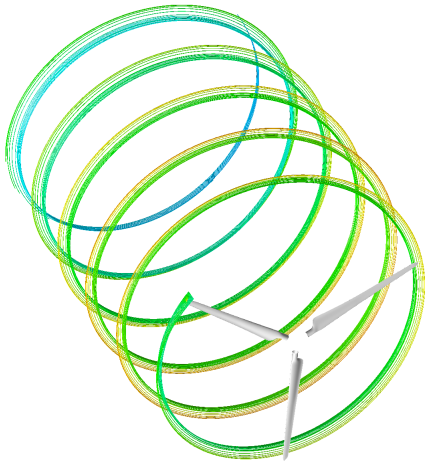


Figure 5: Relative velocity streamlines in the tip vortex region. Streamlines are coloured by the relative velocity magnitude.

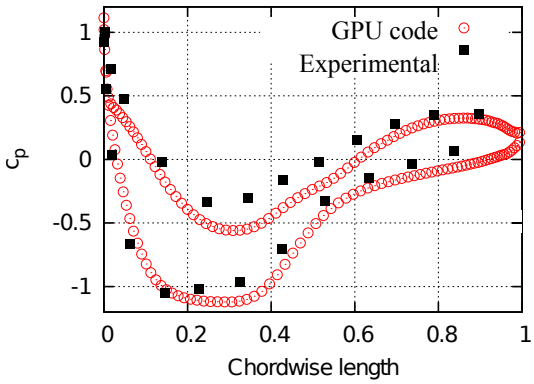


Figure 6: Comparison of the pressure coefficient distribution at 60% of the blades span with experimental results[10].

Optimization convergence history and a detailed analysis of the optimized blade will be included in the full paper.

Acknowledgements

This study has been co–financed by the European Union (European Social Fund–ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALES. Investing in knowledge society through the European Social Fund.

References

[1] Hansen, M., Sørensen, J., Voutsinas, S., Sørensen, N., and Madsen, H. State of the art in wind turbine aerodynamics and aeroelasticity. *Progress in Aerospace Sciences* **42**(4), 285 – 330 (2006).

[2] Kampolis, I., Trompoukis, X., Asouti, V., and Giannakoglou, K. CFD-based analysis and two-level aerodynamic optimization on Graphics Processing Units. *Computer Methods in Applied Mechanics and Engineering* **199**(9-12), 712–722 (2010).

[3] Asouti, V., Trompoukis, X., Kampolis, I., and Giannakoglou, K. Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units. *International Journal for Numerical Methods in Fluids* **67**(2), 232–246 (2011).

[4] Martin, M. J., Andres, E., Lozano, C., and Valero, E. Volumetric b-splines shape parametrization for aerodynamic shape design. *Aerospace Science and Technology* **37**(0), 26 – 36 (2014).

[5] Chorin, A. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* **2**(1), 12–26 (1967).

[6] Spalart, P. and Allmaras, S. A one–equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale* **1**, 5–21 (1994).

[7] Zymaris, A., Papadimitriou, D., Giannakoglou, K., and Othmer, C. Continuous adjoint approach to the Spalart-Allmaras turbulence model for incompressible flows. *Computers & Fluids* **38**(8), 1528–1538 (2009).

[8] Roe, P. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* **43**(2), 357–372 (1981).

[9] Piegl, L. and Tiller, W. *The NURBS Book (2Nd Ed.)*. Springer-Verlag, Inc., New York, (1997).

[10] Schepers, J. and Snel, H. Final report of IEA Task 29, Mexnext (Phase I): Analysis of Mexico wind tunnel measurements. Technical report, ECN, (2012).