# 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 and Kyriakos C. Giannakoglou

**Abstract** This paper presents the development and application of the continuous adjoint method for the shape optimization of wind turbine blades aiming at maximum power output. A RANS solver, coupled with 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 equation. The primal and adjoint fields are used for the computation of 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. The latter is also used for deforming the computational mesh at each optimization cycle. In order to reduce the computational cost, the aforementioned tools, developed in the CUDA 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.

## 1 Introduction

Wind turbines design, 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 [3], their use in shape optimization 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

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

huge meshes (with millions of nodes) 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 computation of the gradient of the objective function is required. To do so, 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 accelerating the solution of the primal and adjoint equation using GPUs. Both the flow and adjoint solvers are ported on GPUs, exhibiting a noticeable speed–up compared to their CPU implementations [4, 1]. 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 over and above of the blade itself [5]. 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 aforementioned methods and the corresponding software is applied for the shape optimization of the blades of a horizontal axis wind turbine.

## 2 Navier-Stokes, Adjoint Equations and Sensitivity Derivatives

The flow model is based on the incompressible flow equations using the Spalart-Allmaras turbulence model. The derivation of the adjoint equations along with the discretization of the resulting equations follows.

## 2.1 Flow (primal) equations

The flow equations used are the incompressible Navier-Stokes equations by applying the pseudo-compressibility approach, as introduced by Chorin [2]. 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 flow equations read

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

where $U_n = [p \ \upsilon_1^A \ \upsilon_2^A \ \upsilon_3^A]^T$ is the vector of the state variables, $\upsilon_i^A, i = 1,2,3$ 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

$$f_{nk}^{inv} = \begin{bmatrix} \beta \upsilon_k^R \\ \upsilon_k^R \upsilon_1^A + p\delta_{1n} \\ \upsilon_k^R \upsilon_2^A + p\delta_{2n} \\ \upsilon_k^R \upsilon_3^A + p\delta_{3n} \end{bmatrix}, \ f_{nk}^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}, \ S_n = \begin{bmatrix} 0 \\ \omega_2 \upsilon_3^A - \omega_3 \upsilon_2^A \\ \omega_3 \upsilon_1^A - \omega_1 \upsilon_3^A \\ \omega_1 \upsilon_2^A - \omega_2 \upsilon_3^A \end{bmatrix}$$

with $\omega$ the blade rotational velocity and the stresses are

$$\tau_{mk} = (\nu + \nu_t) \left( \frac{\partial \upsilon_m^A}{\partial x_k} + \frac{\partial \upsilon_k^A}{\partial x_m} \right)$$

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

Equations 1 are solved together with the Spalart-Allmaras turbulence model [9] PDE ($R_{\tilde{\nu}} = 0$) according to a decoupled time–marching scheme.

## 2.2 Continuous Adjoint Formulation

For the wind turbine application under consideration, the objective function $F$ is the power output of the turbine blading for constant rotational velocity.

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 the shaft, $F$ can be expressed as

$$F = \int_{S_{Blade}} \varepsilon_{klm} \left( x_l - x_l^C \right) \left( p n_m - \tau_{mq} n_q \right) r_k dS \tag{2}$$

where $S_{Blade}$ denotes the blade surface. In equation 2, $n_q$ are the components of the unit vector normal to the blade surface, pointing towards the blade.

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

$$F_{aug} = F + \int_\Omega \Psi_n R_{U_n} d\Omega + \int_\Omega \tilde{\nu}^a R_{\tilde{\nu}} d\Omega \tag{3}$$

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 3, which yields

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

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. The remaining integrals form the expression of the gradient of $F$ w.r.t. $b_i$. The field adjoint equations read

$$R_{\Psi_n} = -\underbrace{A_{mnk}\frac{\partial \Psi_m}{\partial x_k}}_{Conv(\Psi)} - \underbrace{\frac{\partial \phi^{vis}_{nk}}{\partial x_k}}_{Diff(\Psi)} - \underbrace{S^{adj}_n}_{Source1(\Psi)} + \underbrace{T^{adj}_n}_{Source2(\tilde{\nu}_a)} = 0 \tag{5}$$

with

$$A_{nmk} = \begin{bmatrix} 0 & \beta\delta_{1k} & \beta\delta_{2k} & \beta\delta_{3k} \\ \delta_{1k} & \upsilon_1^R + \upsilon_1^A\delta_{1k} & \upsilon_1^A\delta_{2k} & \upsilon_1^A\delta_{3k} \\ \delta_{2k} & \upsilon_2^A\delta_{1k} & \upsilon_2^R + \upsilon_2^A\delta_{2k} & \upsilon_2^A\delta_{3k} \\ \delta_{3k} & \upsilon_3^A\delta_{1k} & \upsilon_3^A\delta_{2k} & \upsilon_3^R + \upsilon_3^A\delta_{3k} \end{bmatrix}$$

$$S^a_n = \begin{bmatrix} 0 \\ \omega_2\Psi_4 - \omega_3\Psi_3 \\ \omega_3\Psi_2 - \omega_1\Psi_4 \\ \omega_1\Psi_3 - \omega_2\Psi_2 \end{bmatrix} \qquad \phi^{vis}_{nk} = \begin{bmatrix} 0 \\ \tau^a_{1k} \\ \tau^a_{2k} \\ \tau^a_{3k} \end{bmatrix}$$

where $\delta_{ij}$ is the Kronecker's symbol and

$$\tau^a_{mk} = (\nu + \nu_t)\left(\frac{\partial \Psi_{m+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_m}\right) \tag{6}$$

are the adjoint stresses.

In equation 5, 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[10] 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

$$\begin{aligned}\frac{\delta F}{\delta b_i} = &\int_{S_{Blade}} \varepsilon_{klm} p n_m r_k \frac{\delta x_l}{\delta b_i} dS + \int_{S_{Blade}} \varepsilon_{klm}\left(x_l - x_l^C\right) p r_k \frac{\delta}{\delta b_i}(n_m dS) - \\ &\int_{S_{Blade}} \varepsilon_{klm} \tau_{mq} n_q r_k \frac{\delta x_l}{\delta b_i} dS - \int_{S_{Blade}} \varepsilon_{klm}\left(x_l - x_l^C\right) \tau_{mq} r_k \frac{\delta}{\delta b_i}(n_m dS) + \\ &T^{MF} + T^{SA}\end{aligned} \tag{7}$$

where the terms $T^{MF}$ and $T^{SA}$ correspond to the differentiation of the flow equations and the turbulence model respectively. These terms are herein omitted in the interest of space. The reader may find them in [10] irrespective of the objective function used.

## 2.3 Discretization and Numerical Solution

The primal and adjoint equations are discretized on hybrid meshes (consisting of tetrahedra, pyramids, prisms or hexahedra) using the vertex–centered finite volume method and solved using a time–marching scheme. The numerical fluxes crossing the finite volume interfaces 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 [7] 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.

On the other side, the adjoint inviscid numerical fluxes are computed using a non–conservative scheme,

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

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

For 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

$$\left(\frac{\partial W}{\partial x_k}\right)^{PQ} = \overline{\left(\frac{\partial W}{\partial x_k}\right)} - \left[\overline{\left(\frac{\partial W}{\partial x_k}\right)} t_m^{PQ} - \frac{W^Q - W^P}{\sqrt{(x_m^Q - x_m^P)^2}}\right] t_k^{PQ} \tag{8}$$

where

$$t_m^{PQ} = \frac{x_m^Q - x_m^P}{\sqrt{(x_m^Q - x_m^P)^2}}, \quad \overline{\left(\frac{\partial W}{\partial x_k}\right)} = \frac{1}{2}\left[\left(\frac{\partial W}{\partial x_k}\right)^P + \left(\frac{\partial W}{\partial x_k}\right)^Q\right]$$

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 $(\xi, \eta, \zeta)$ be the three parametric directions and $X_m^{ijk}$ and $w^{ijk}$ 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

$$x_m(\xi,\eta,\zeta) = \frac{\sum_i^{N_\xi}\sum_j^{N_\eta}\sum_k^{N_\zeta} \Xi_{i,p_\xi}(\xi) H_{j,p_\eta}(\eta) Z_{k,p_\zeta}(\zeta) X_m^{ijk} w^{ijk}}{\sum_i^{N_\xi}\sum_j^{N_\eta}\sum_k^{N_\zeta} \Xi_{i,p_\xi}(\xi) H_{j,p_\eta}(\eta) Z_{k,p_\zeta}(\zeta) w^{ijk}} \tag{9}$$

where, $\Xi_{i,p_\xi}$ is the $i$th B-Spline basis function of degree $p_\xi$ defined on the knot vector $K_\xi = \{\xi_0, \ldots, \xi_{m_\xi}\}$ ($H_{j,p_\eta}$ and $Z_{k,p_\zeta}$ are defined similarly), $N_\xi$ is the number of control points in the $\xi$ direction and it must hold that $m_\xi = N_\xi + p_\xi + 1$ [6]. Knots must be arranged in 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 in 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 are given by closed–form expressions resulting from the differentiation of equation 9.

During the optimization loop, the control point coordinates and weights are updated and equation 9 is used to deform the computational mesh and the 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 a 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[4], 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[1], it is shown that, when processing 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, 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 (GPU threads associated with finite volume interfaces) and accumulated by the second kernel (GPU threads associated 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 sub-domains and each sub-domain is associated with one GPU. For instance, figure 1 (left) shows a triangular mesh generated around an isolated airfoil partitioned in three overlapped sub-domains. The shared regions of the mesh sub-domains are marked in white in figure 1. The whole mesh (i.e. including the overlapped regions) of the 3rd sub-domain, with the boundaries shared with sub-domains 1, 2, can be seen in figure 1 (right). 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 sub-domain performs the same sequence of steps. As an example, the GPU associated with the 3rd sub-domain performs the following steps:

*Step A:* Launches a kernel only for the computation of the gradients at the nodes interface with sub-domains 1 and 2 (i.e. nodes lying on the blue and red lines of figure 1 (right)).

*Step B:* Performs the data interchange between the sub-domains (assigned to different GPUs).

*Step C:* Launches a kernel for the computation of the gradients at the remaining nodes of the sub-domain.

Steps A, B can be performed simultaneously 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.

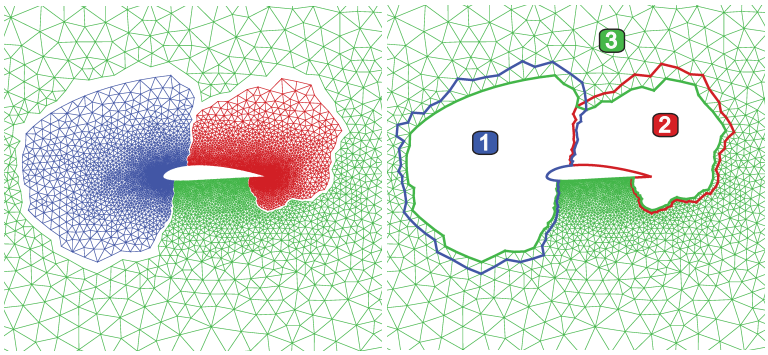


**Fig. 1** Mesh with triangular elements around an isolated airfoil partitioned in three overlapped subdomains.

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

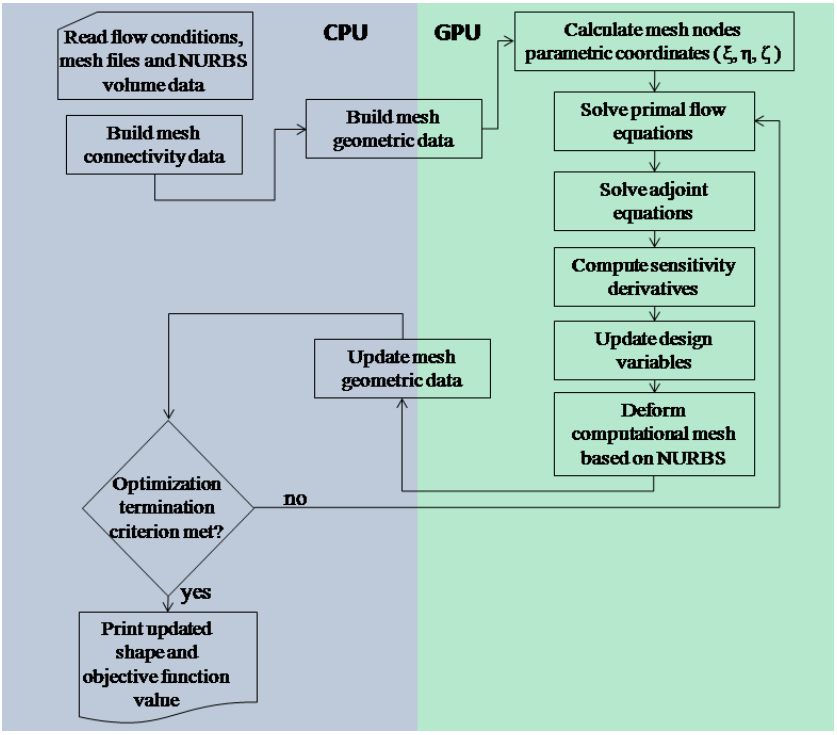


**Fig. 2** Flowchart demonstrating the optimization algorithm steps. Steps performed on the CPU and the GPU are distinguished.

## 5 Optimization of the Wind Turbine Blade

The developed software described in the previous sections, was used for the shape optimization of the MEXICO[8] horizontal axis wind turbine (HAWT) blade for maximum power output, when operating at $10\frac{m}{s}$ farfield velocity and $0^o$ yaw angle. For the parameterization of the blade, a $5\times5\times5$ NURBS control volume is used, as shown in figure 3. All boundary control points are kept fixed in order to ensure $C^1$ continuity while the remaining ones are allowed to move along the z axis (figure 3) leading to 27 ($3\times3\times3$) design variables in total. The computational mesh consists of about $2.5\times10^6$ nodes and both the primal and adjoint solvers run on 4 NVIDIA Kepler K20 GPUs, lying on two different nodes. On this platform each optimization cycle needs approximately $25min$, $15min$ for the solution of the primal equations and $10min$ for the adjoint. The convergence history of the optimization is shown in figure 4. The optimized blade yields 3% increased torque compared to the reference blade. The improvement is minor due to the degrees of freedom used, i.e. the NURBS control points were allowed to move only in the z direction.


**Fig. 3** Parameterization of the HAWT blade.

Since the differences between the optimized and the reference blade are not visible in a 3D surface comparison, the blade profiles at three spanwise positions of the blade are compared instead (figure 6).

Figure 6 presents the comparison of the chordwise distribution of the pressure coefficient for the starting and the optimized blade, along with the experimental results (from [8]) for the same spanwise positions. It is clear that most differences appear in the lower part of the blade.

The relative velocity streamlines in the tip vortex region are plotted in figure 7.

Figures 8 and 9 show the axial velocity and turbulent viscosity in a transversal slice through the wing turbine origin.

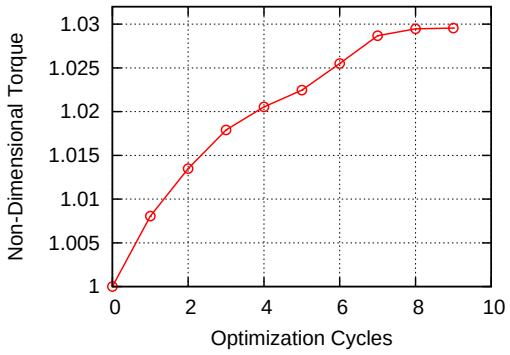


**Fig. 4** Optimization convergence history. On the vertical axis, the objective function (power output to be maximized) is divided by the value this function takes on for the starting blade geometry.

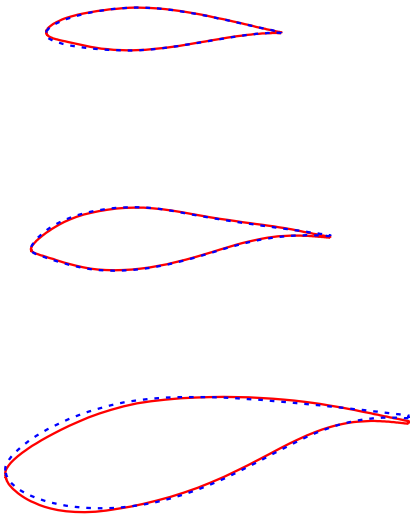

**Fig. 5** Comparison of the optimized blade profile (solid/red line) with the starting (dashed/blue) at 35%, 60% and 82% (from bottom to top) of the wind turbine blade span.

## 6 Conclusions

This paper presented the development and use of the continuous adjoint method for the shape optimization of a HAWT blade for maximum torque. Since wind turbine blades are complex geometries, the parameterization was based on volumetric NURBS method, which also contributes to the mesh deformation at each optimization cycle. In order to reduce the optimization turnaround time, the solution of both

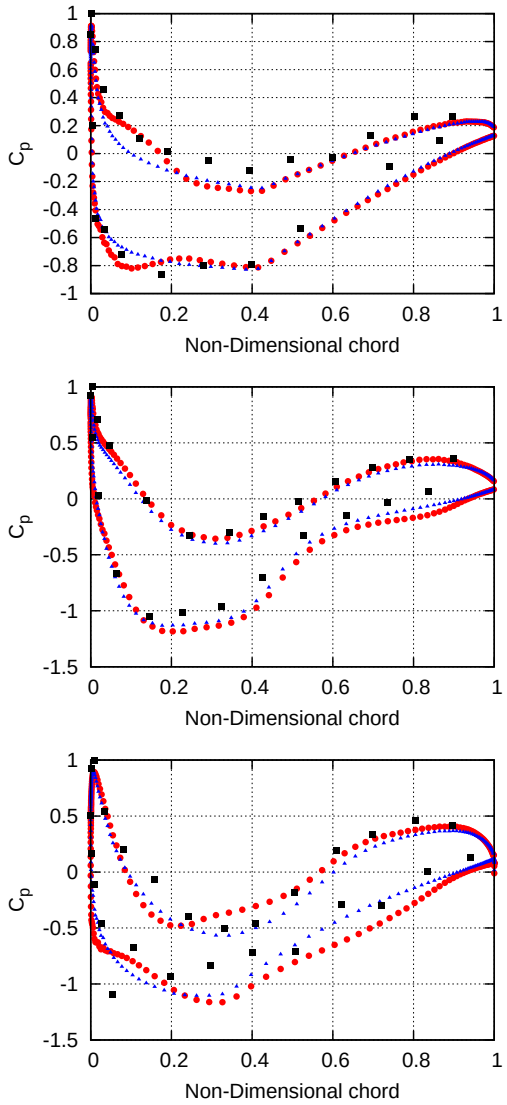


**Fig. 6** Comparison of the pressure coefficient for the starting (red circles) and the optimized blade (blue triangles), along with the available experimental data (black squares) on the starting geometry, [8] at 35%, 60% and 82% (from bottom to top) of the wind turbine blade span. The pressure coefficient is defined as $c_p = \frac{p-p_{far}}{\frac{1}{2}(V_{far}^2+\omega^2 R^2)}$, with $R$ the local radius and $far$ indexing farfield flow quantities.

the flow and the adjoint equations is carried out on 4 Nvidia Tesla K20 GPUs. In particular, each optimization cycle requires approximately 15*min* for the primal and 10*min* for the adjoint equations solution.

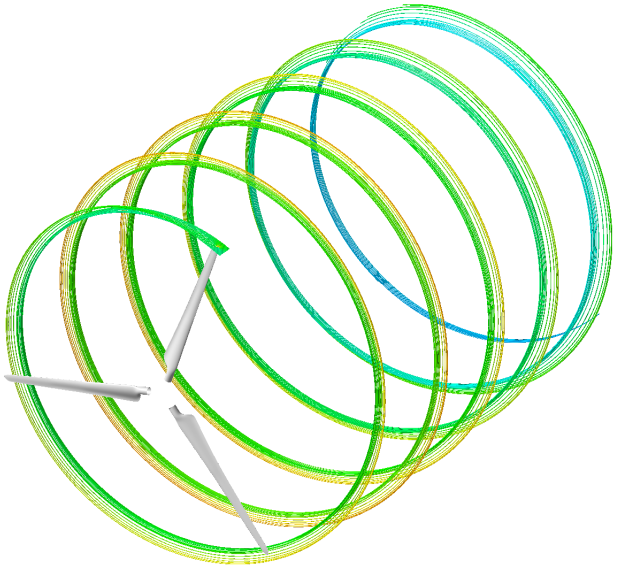

**Fig. 7** Relative velocity streamlines (coloured based on the relative velocity magnitude) in the tip vortex region.

## 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.

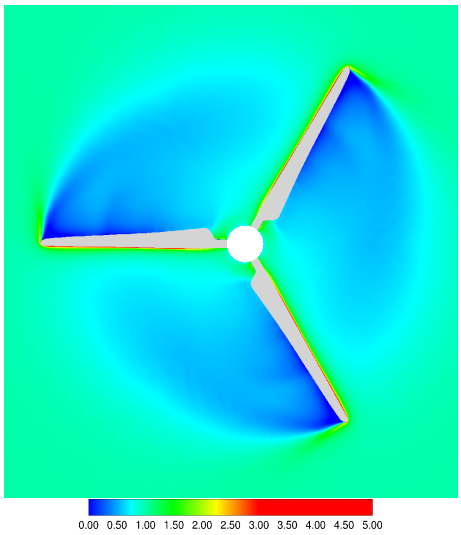


**Fig. 8** Axial velocity in a transversal slice through the wing turbine origin. The velocity values are normalized with respect to the farfield velocity magnitude.

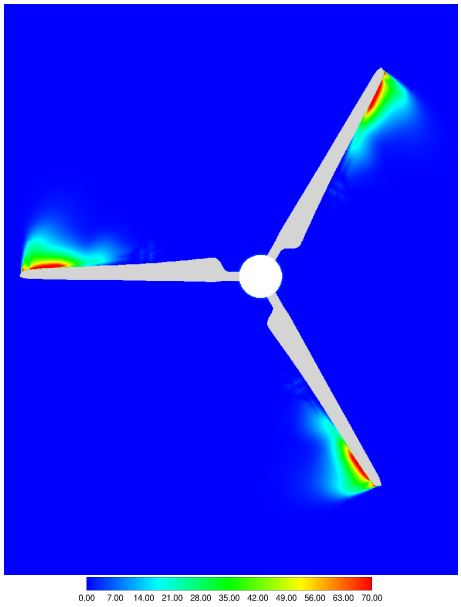


**Fig. 9** Turbulent viscosity in a transversal slice through the wing turbine origin.

## References


1. Asouti, V., Trompoukis, X., Kampolis, I., 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)
2. Chorin, A.: A numerical method for solving incompressible viscous flow problems. Journal of Computational Physics **2**(1), 12–26 (1967)
3. Hansen, M., Srensen, J., Voutsinas, S., Srensen, N., Madsen, H.: State of the art in wind turbine aerodynamics and aeroelasticity. Progress in Aerospace Sciences **42**(4), 285 – 330 (2006). DOI http://dx.doi.org/10.1016/j.paerosci.2006.10.002. URL http://www.sciencedirect.com/science/article/pii/S0376042106000649
4. Kampolis, I., Trompoukis, X., Asouti, V., 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)
5. Martin, M.J., Andres, E., Lozano, C., Valero, E.: Volumetric b-splines shape parametrization for aerodynamic shape design. Aerospace Science and Technology **37**(0), 26 – 36 (2014). DOI http://dx.doi.org/10.1016/j.ast.2014.05.003. URL http://www.sciencedirect.com/science/article/pii/S127096381400090X
6. Piegl, L., Tiller, W.: The NURBS Book (2Nd Ed.). Springer-Verlag, Inc., New York (1997)
7. Roe, P.: Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics **43**(2), 357–372 (1981)
8. Schepers, J., Snel, H.: Final report of IEA Task 29, Mexnext (Phase I): Analysis of Mexico wind tunnel measurements. Tech. rep., ECN (2012)
9. Spalart, P., Allmaras, S.: A one–equation turbulence model for aerodynamic flows. La Recherche Aérospatiale **1**, 5–21 (1994)
10. Zymaris, A., Papadimitriou, D., Giannakoglou, K., Othmer, C.: Continuous adjoint approach to the Spalart-Allmaras turbulence model for incompressible flows. Computers & Fluids **38**(8), 1528–1538 (2009)