MOX–Report No. 13/2011

Numerical simulation of sailing boats: dynamics, FSI, and shape optimization

Lombardi, M.; Parolini, N.; Quarteroni, A.; Rozza, G.

MOX, Dipartimento di Matematica "F. Brioschi"
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox@mate.polimi.it http://mox.polimi.it

Numerical simulation of sailing boats: dynamics, FSI, and shape optimization

Matteo Lombardi, Nicola Parolini, Alfio Quarteroni and Gianluigi Rozza

Abstract The numerical simulation of free-surface flows around sailing boats is a complex topic that addresses multiple mathematical tasks: the correct study of the flow field around a rigid hull, the numerical simulation of the hull dynamics, the deformation of the sails and appendages under transient external conditions like gusts of wind or wave patterns and, overall, the coupling among all these components. In this paper, we present some recent advances that have been achieved in different research topics related to yacht design and performance prediction. In particular, we describe the numerical algorithms that have been developed in the framework of open-source libraries for the simulation of free-surface hydrodynamics and boat dynamics, as well as for the analysis of the fluid-structure interaction between wind and sails. Moreover, an algorithm for shape optimization, based on the solution of the adjoint problem and combined with the Free Form Deformation (FFD) method for the shape parametrization and mesh motion, is presented and discussed. Theoretical and methodological aspects are described and the first preliminary results are reported.

1 Introduction, Motivation, and Problem Setting

The analysis of the dynamics of a boat advancing into the sea and the corresponding numerical simulations is an interesting and challenging problem [16]. From an engineering point of view it is useful to evaluate the dynamic behaviour of a hull in as many configurations as possible. This can be a helpful tool for boat design, where it could be used to improve the hull performance. This task can be accomplished in two (complementary) ways: experimentally in a towing tank or numerically by Computational Fluid Dynamics (CFD) simulations. Not surprisingly, experimental measurements are usually very expensive since they require the construction of as many scale-models as the number of configurations to be analysed and also technical equipments and facilities are particularly onerous. Furthermore it is possible to reproduce only a limited range of configurations due to practical limitations with the experimental set-up. For these reasons, numerical computations are becoming more and more relevant in this field and much effort is being devoted in the industrial and academic research to make them as accurate as possible. When a numerical simulation approach is used, it is easy to switch from a model to another one and the spectrum of configurations that it is possible to analyse is a priori unlimited. On the other hand, the physical problem is very complex and the most advanced mathematical models have to be used to simulate the problem correctly.

The computation of the flow around ships is a subject studied since long time, at first through the analysis of simplified models and more recently with use of increasingly complex and accurate numerical models. The earliest important theoretical results concerning ship hydrodynamics date

Matteo Lombardi, Gianluigi Rozza, and Alfio Quarteroni
MATHICSE, CMCS Chair of Modelling and scientific computing, EPFL, Lausanne (Switzerland), e-mail: matteo.lombardi@epfl.ch, gianluigi.rozza@epfl.ch, alfio.quarteroni@epfl.ch

Nicola Parolini and Alfio Quarteroni
MOX, Politecnico di Milano, Milano (Italy) e-mail: nicola.parolini@polimi.it, alfio.quarteroni@epfl.ch

back to the end of the nineteenth century. Froude, Kriloff and Reynolds were the fathers of the first theories about ship motion. Froude investigated the role of the wave component of the resistance defining the important dependence on a parameter, the nowadays called Froude number. In 1898, a thin ship theory was proposed in [53] to describe the wave generated by an advancing ship and an analytical estimation of the wave resistance (based on the Mitchell integral) was derived. More complex models date back to the 60s with the solution of potential flow equations, thanks to important efforts in the aerodynamic fields (see, e.g., the pioneering work by Hess and Smith [32]). Those models were later extended to the solution of ship hydrodynamic problems [18, 19] and added mass models. Although based on simple irrotational and inviscid flow models, potential flow panel codes are still commonly used in the ship hydrodynamics community [75] thanks to their low computational cost and because in several situations they manage to reproduce the main features of the flow quite accurately. With the continuous growth of computation power and improvement of the numerical models for the solution of partial differential equations, more and more complex ship hydrodynamics problems can be simulated accounting for viscous (and turbulent) flows and moving domain (for boat dynamics). Indeed, in the last two decades, numerical methods based on the solution of complex PDEs, Navier–Stokes (and Euler) equations, have been successfully applied to naval engineering problem (see, *e.g.*, [33, 22, 1, 36, 65]) and are, nowadays, standard numerical tools adopted in the design process for ship performance evaluation.

The Navier–Stokes equations can be written as

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = \nabla \cdot T(\mathbf{u}, p) + \mathbf{G} \quad \text{in } \Omega \times (0, T) \tag{1}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad \text{in } \Omega \times (0, T), \tag{2}$$

where $\mathbf{u}$ is the velocity field, p is the pressure, $T(\mathbf{u}, p) = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) - pI$ is the stress tensor, ρ the density, μ the viscosity coefficient and $\mathbf{G}$ the gravity term. In case of incompressible fluids, the second equation reduces to $\nabla \cdot \mathbf{u} = 0$. Finally, Ω is the computational domain occupied by the fluid, while (O, T) is time interval in which the problem will be studied. The Reynolds number $Re = \frac{\rho U L}{\mu} = \frac{U L}{\nu}$ is a dimensionless indicator of the ratio of inertial forces to viscous forces and thus quantifies the relative importance of these two types of forces for given flow conditions. Another fundamental dimensionless number is the Froude number $Fr = \frac{U}{\sqrt{g L}}$ which involves the ratio between the gravity force and the inertial forces.

Typically, the Reynolds numbers that characterize ship hydrodynamics problems are large (order of magnitude 10^7-10^9); the flow is indeed fully turbulent around a large portion, to a first approximation all of it, of the boat and suitable turbulence models have to be used to correctly estimate the forces acting on it. A Direct Numerical Simulation (DNS) of such flows is unaffordable with the computational power available today. Although the Large Eddy Simulation (LES) [79] and the Detached Eddy Simulation (DES) [84] are gaining interest also in the ship hydrodynamics community [2, 90], the Reynolds Averaged Navier Stokes equations (RANS) still represent the most common approach to solve this kind of problems. In the RANS approach, velocity and pressure are decomposed in two components, an average one and a fluctuation part, and the Navier–Stokes equations are solved only with respect to the averaged variables, while the effect of the fluctuation part is modelled via some extra turbulence models [24, 88, 9].

If we split the flow variables into an averaged part and a fluctuation part

$$\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}', \quad p = \bar{p} + p', \tag{3}$$

where $\overline{\mathbf{u}} = \frac{1}{\Delta t} \int_t^{t+\Delta t} \mathbf{u}\, dt$ and Δt is a value higher than the turbulent time scale but smaller than the one chosen for the actual numerical discretization of the equations, and we substitute into equations (1-2) (under the assumption of incompressibility), the RANS equations are obtained as:

$$\frac{\partial}{\partial t}(\overline{\rho}\overline{\mathbf{u}}) + \nabla \cdot (\overline{\rho}\overline{\mathbf{u}} \otimes \overline{\mathbf{u}}) = \nabla \cdot T(\overline{\mathbf{u}}, \overline{p}) + \nabla \cdot \mathbf{R} \quad \text{in } \Omega \times (0, T), \tag{4}$$

$$\frac{\partial \overline{\rho}}{\partial t} + \overline{\mathbf{u}} \cdot \nabla \overline{\rho} = 0 \quad in\ \Omega \times (0, T), \tag{5}$$

$$\nabla \cdot \overline{\mathbf{u}} = 0 \quad \text{in } \Omega \times (0,T), \tag{6}$$

where $\mathbf{R}$ is the so-called Reynolds stress tensor $R_{ij} = \rho \overline{u_i' u_j'}$ which derives from the non-linear convective term. From a physical point of view, this term corresponds to the extra mixing due to the turbulent eddies not represented by the averaged variables. Many different turbulent models exist in literature, and they can be grouped in the following three classes:

- algebraic models,
- models based on differential equations (one or two equations models),
- models based on the Reynolds stress tensor transport equation.

In this work, we have considered the SST (Shear Stress Transport) [51] model, which is a combination of the *K-epsilon* [54] and *K-omega* [50] models. This model has been validated extensively in the literature [88] and successfully applied to naval hydrodynamics problems [65, 91].

Two main classes of methods for the solution of free surface flows are present in the specialized literature: front tracking methods and front capturing methods. In the formers, only one phase is simulated, the water, and the boundary of the domain follows the free-surface evolution. In this case, the computational domain Ω in equations (1) and (2) does depend on time. The main drawback is that complex free-surface behaviour, like spray or overturning waves, cannot evidently be simulated, and the mesh quality can rapidly deteriorate. By front tracking methods, on the contrary, both air and water are simulated, and the computational domain and mesh is kept fixed. An extra variable is introduced so to specify for every cell of the domain whether it belongs to the air or water. The most common methods that belong to this class are the Volume of Fluid (VOF) method [34], [5] and the Level Set (LS) method [61], [60]. The volume of fluid method is based on a *homogeneous* approach, where a unique velocity and pressure field is defined for both fluids. A new variable c, called *volume fraction* of the water, is defined on the whole domain and such that $c = 0$ if the element corresponds to an air zone, $c = 1$ if the element corresponds to a water zone and a value in the middle if both fluid are present in the element. Furthermore, c must be bounded between 0 and 1 With this approach, it is thus possible to solve a unique fluid-dynamic problem for both fluids and Navier–Stokes equations (1-2), under the assumption of incompressibility, become:

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = \nabla \cdot T(\mathbf{u}, p) + \mathbf{G} \quad in\ \Omega \times (0,T), \tag{7}$$

$$\nabla \cdot \mathbf{u} = 0 \quad in\ \Omega \times (0,T), \tag{8}$$

where N_{el} is the number of different fluids, i.e. two in this case, and ρ and μ are defined as

$$\rho = c\rho_{\text{water}} + (1-c)\rho_{\text{air}}, \quad \mu = c\mu_{\text{water}} + (1-c)\mu_{\text{air}}, \tag{9}$$

where μ is present in the tensor T. For the evolution of the interface between air and water, a further transport equation for the *volume fraction* variable has to be solved:

$$\frac{\partial c}{\partial t} + \nabla \cdot (c\mathbf{u}) = 0. \tag{10}$$

In order to keep the interface sharp, ad-hoc numerical terms can be introduced into those equations. In this work, the Volume of Fluid (VOF) model is adopted and solved using the Multidimensional Universal Limiter with Explicit Solution (MULES) algorithm, in which a compression term is added to the standard VOF formulation in order to keep the interface sharp (see [78] for details).

In case of mesh motion, for example when simulating the boat dynamic or when solving the Fluid-Structure interaction problem, the Arbitrary Lagrangian Eulerian (ALE) formulation has to be implemented [35, 17, 28, 29]. This correction takes care of the fact that the mesh is moving and thus corrects the convective term with the actual mesh motion velocity.

Equations (4) thus become

$$\frac{\partial}{\partial t}(\overline{\rho}\overline{\mathbf{u}}) + \nabla \cdot (\overline{\rho}(\overline{\mathbf{u}} - \mathbf{w}) \otimes \overline{\mathbf{u}}) = \nabla \cdot T(\overline{\mathbf{u}}, \overline{p}) + \nabla \cdot \mathbf{R} \quad \text{in } \Omega(t) \times (0,T), \tag{11}$$

where $\mathbf{w}$ is the mesh velocity. The same correction has to be applied to the turbulence and volume fraction equations.

An important feature of this research project is the use of open source codes. All over the world, the scientific community is investing a great effort in the development and validation of free software like *FreeFem* [68], *DealII* [45], *LifeV* [25], *OpenFOAM* [59]. The latter is growingly very rapidly and is being extensively used and developed by many academic and industrial research centres. It features many of the key-points needed for our kind of applications, more notably a Navier–Stokes solver, turbulence models, VOF models, and parallel scalability. Furthermore its modular organization allows to easily transfer specific development to other applications/solvers, yielding a larger impact of the research carried out by single research groups on the global scientific community. For these reasons, the authors have chosen to use OpenFOAM as the base library to solve the flow problem.

OpenFOAM (Open Field Operation and Manipulation) is a C++ object oriented programming toolbox capable of solving several continuum mechanics problems, among which the solution of partial differential equation like the Navier–Stokes equations [38]. It works on structured and unstructured meshes and the space discretization is done by means of finite volume method. The time marching scheme is based on a pseudo PISO algorithm, where pressure and velocity are solved in a segregated way and, for every time step, sub-iterations are needed to reach convergence [24].

An accurate solution of the flow problem is the starting point for the solution of more complex physical behaviour such as the rigid-body boat dynamics ([4, 16]) or the fluid-structure interaction between sails and wind [87, 73, 20, 89]. Another valuable development direction of numerical tools for hydrodynamic design is represented by the integration of the flow solver in a shape optimization process. In the naval engineering community, shape optimization techniques are receiving an increasing interest, both when considering gradient-based optimization schemes [85, 48] and derivative-free algorithms [18, 8].

In this paper, we present a numerical model that has been developed for the simulation of free-surface boat dynamics problems and the Fluid-Structure interaction algorithm devised for the prediction of the flying shape of downwind sails. We also introduce a shape optimization technique for hydrodynamic drag reduction based on gradient methods techniques (with and without the solution of the adjoint problem), which makes use of the FFD method for the shape parametrization and mesh motion. Finally, a set of simulations obtained with the different methods proposed are presented and discussed.

2 Hull Dynamics

In order to simulate the dynamics of the hull, it is necessary to estimate the loads exerted by the fluids (air and water) on the structure and the dynamic response of the structure to these loads. Our attention has been focused, in particular, on the resolution of the equations of motion of the boat and the coupling algorithm for the fluid-structure interaction.

The motion of a boat, once neglecting the deformation of the hull (provided that they are small) is governed by the equations of motion of a rigid body: those can be split in a translation motion of a point, which for simplicity is usually chosen the barycentre (equivalent in this case to the center of mass, i.e. the mean location of all the mass in the system) and the rotation of the body around it.

The equation for the variation of the linear momentum referred to the barycentre is

$$m\ddot{\mathbf{X}}_G = \mathbf{R}, \tag{12}$$

where m is the mass of the boat. $\mathbf{R}$ is the total force acting on the boat obtained via:

$$\mathbf{R} = \mathbf{F}_{Flow} + m\mathbf{g} + \mathbf{F}_{ext}, \tag{13}$$

where F_{flow} is the hydrodynamic force, $m\mathbf{g}$ the gravity force and $\mathbf{F}_{ext}$ corresponds to the resultant of the external forces. The hydrodynamic force $\mathbf{F}_{Flow}$ can be evaluated integrating the viscous and pressure stresses acting on the wetted surface of the boat:

$$\mathbf{F}_{Flow} = \sum_{j=1}^{n} (-p_j \mathbf{n}_j + \tau_j) S_j, \tag{14}$$

where the sum is done over all faces of the volume that defines the surface of the boat S_j is the corresponding area.

In order to solve the equation for the variation of angular momentum first of all is necessary to define two reference system: one referred to the fixed domain, that we will call the *principal reference system* and one that rotates with boat, *secondary reference system*. It can be shown that the union of the unit-vector $\mathbf{i}, \mathbf{j}, \mathbf{k}$ of the secondary reference system expressed in the principal system generates the rotation matrix form one system to the other.

The equation of variation of angular momentum, referred to the principal reference system, can be finally expressed as

$$\bar{\bar{T}} \bar{\bar{I}}_G \bar{\bar{T}}^{-1} \dot{\omega} + \omega \times \bar{\bar{T}} \bar{\bar{I}}_G \bar{\bar{T}}^{-1} \omega = \mathbf{M}_G, \tag{15}$$

where ω is the angular velocity, $\bar{\bar{I}}_G$ is the matrix of the *moment of inertia*

$$\bar{\bar{I}}_G = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{YX} & I_{YY} & I_{YZ} \\ I_{ZX} & I_{ZY} & I_{ZZ} \end{bmatrix}$$

and $\mathbf{M}_G$ is the total moment around the rotation center acting on the boat expressed in the principal reference system. The total moment can be expressed as

$$\mathbf{M}_G = \mathbf{M}_{Flow} + \mathbf{M}_{ext}, \tag{16}$$

where $\mathbf{M}_{ext} = (\mathbf{X}_{ext} - \mathbf{X}_G) \times \mathbf{F}_{ext}$ is the moment due to the external forces and $\mathbf{M}_{Flow}$ the moment due to the hydrodynamic forces, which, analogously as for the hydrodynamic forces, equation (14), corresponds to

$$\mathbf{M}_{Flow} = \sum_{j=1}^{n} (\mathbf{X}_j - \mathbf{X}_G) \times (-p_j \mathbf{n}_j + \tau_j) S_j. \tag{17}$$

These equations form a system of 6 second order ordinary differential equations, three for the translation motion and three for the rotation. The translation equations are not coupled and can be resolved separately. The rotation ones, except when the rotation is around one single axis, are coupled non-linear equations and have to be resolved as a coupled system. The rotation are handled via Euler angles and the dynamic problem solved via Adams-Bashforth/Crank-Nicholson schemes, see [46] for a detailed analyses.

As mentioned in the previous sections, the RANS model has been adopted to simulate the fluid-dynamic behaviour of the flow around the boat, and particular care has been paid to model the boundary layer around the hull. The free-surface dynamics has been tracked using a VOF model. The coupling between the fluid solver and the hull dynamics solver has been done explicitly: for every time step, the fluid solver advances using the hull position at the previous time step, then the fluid forces at time t_{n+1} are evaluated and the new boat position is obtained. Finally the fluid mesh is adjusted and the next time step can be computed. For this kind of applications, such a simple and explicit coupling algorithm is usually stable since the transient fluid solver requires a quite restrictive time step constraint while the hull dynamic time-scale is usually much higher. A more general and detailed analyses of Fluid- Structure Interaction coupling algorithm is presented in the following sections. For more details on the hull dynamics model refer to [4, 64, 16].

3 Wind/Sail Fluid-Structure Interaction

The problem of modelling and simulating the interaction between fluids and solids arises in many different applied fields. Examples are the interaction of the blood flow within arterial walls, the flow around the sails of a sailing boat, the deformation of the wing of an aeroplane or a Formula-1 car. Although extensively studied in literature, due to its complexity this problem is still subject

of intensive investigation. A complete review of the existing literature goes far beyond the scope of this work, however we refer for example to [58, 12, 13, 71] and the references therein. Here we would simply like to mention the different approaches that can be chosen and the reasons behind those choices. The first classification depends on whether one is interested only in the equilibrium solution, or to the whole transient behaviour. In case of wind/sail fluid structure interaction, both situations are of interest.

Secondly, one should consider which level of accuracy is needed for the simulation of the fluid and solid parts, and if any simplified model could be meaningful. For example, in upwind sailing configuration, it is usually not indispensable to solve the full Navier–Stokes equations around the sails; indeed, simpler models, like the one based on potential flow, are accurate enough to obtain realistic flow fields. The same argument can be applied to the structural part, where assumptions like linear elasticity model or small deformation can lead to simplified, and much faster, numerical methods.

In this work, a full RANSE solver has been used, since, in the configurations to be analysed, flow separations may occur and viscous stresses may be important: phenomena which can not be simulated accurately with potential flow solver. As structural solver, different options may be considered. For cases where a linear elasticity model is adequate, classical simple structural solver, like the one already implemented in OpenFOAM, can be used. For the sails simulations, a specific solver for membranes/shells specifically tuned for sails must to be used. More details are given in the following subsection.

Another important aspect to be taken into consideration is the amplitude of the deformation and the added-mass [10] effect; this determines the FSI coupling strategy to be used in order to obtain stable simulations. In this respect, we refer to *partitioned strategies* when we solve iteratively the fluid and structure sub-problems (and never assemble the full matrix). This allows the possibility to couple two existing codes, one solving the fluid equations and the other the structure problem. We refer instead to *monolithic* schemes when we solve the non-linear coupled problem all at once. In this case, a global linear system is assembled and solved for all the unknowns of the problem: fluid and structural variables as well as all the geometrical informations (like the interface position). The complete fluid-solid system has to be solved as a whole and memory issue may arise due to the size of the global matrix. Other possible models that are in between these two approaches exist: for instance one can formally write the monolithic problem and then use a block preconditioner to solve it. In this way one recovers partitioned procedures [12].

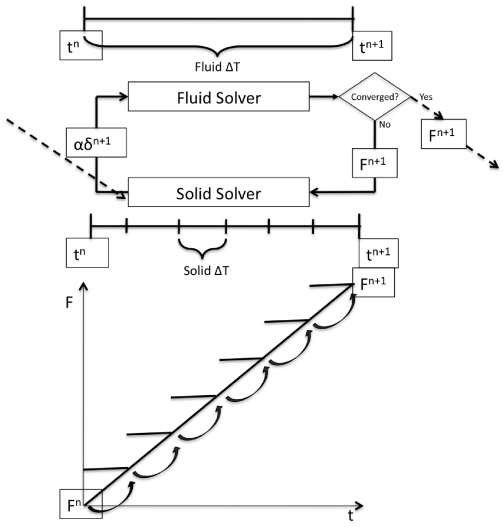


Fig. 1 Schematic flow-chart of the Fluid-Structure interaction coupling algorithm.

In case of small deformation and/or high density ratio between air and fluid, a simple coupling strategy can be used, e.g. the so called *weakly coupled partitioned approach* where the two solvers

can be run in a *segregated* way. For example at first the fluid solver is run, then the interface conditions (the stresses) are passed from the fluid mesh to the solid mesh, and then the solid solver is run; the solid interface conditions (the interface positions and its rate of deformations) are then passed back to the fluid solvers and the fluid mesh and boundary conditions are updated accordingly; a new time step can the be computed. Unfortunately, when the added mass effect gets more important, it has been proven (see e.g. [10]) that this kind of schemes becomes unconditionally unstable and it is necessary to sub-iterate, with relaxation, for every time step. The *strongly coupled partitioned approach*, consists in introducing sub-iterations at every time step between the two solvers. Usually the interface conditions are imposed via a relaxation procedure, for example with the *Aitken* relaxation method, see [13]. A potential drawback of this approach is that the number of sub-iterations may become large, of the orders of a few tens depending on the case, and thus the computational cost may rise very quickly. In our case, since the fluid and structural solvers are in general two different codes and the FSI problem is stiff, the strongly coupled partitioned approach has been adopted.

In figure 1 a schematic flow chart of the strongly coupled partitioned FSI algorithm adopted in this work is shown. For every time step, sub-iteration cycles have to be computed. At first, the structural solver is run using as boundary condition the fluid solver stresses from the previous time step. The new position of the sail is passed to the fluid and an iteration of the fluid solver is computed. The updated fluid stresses are then transferred back to the solid and the structural solver is run again. Once again, the new deformed sail position is passed to the fluid and the new flow solution is computed. If the difference between the solid deformation and fluid stresses at the previous and current sub-iteration is below a given tolerance, the FSI is considered converged and a new time step is computed, otherwise, a new sub-iteration cycle is carried out. In order to accelerate the convergence and ensure stability, the deformation passed from the solid to the fluid are relaxed with the one obtained at the previous sub-iteration cycle via Aitken relaxation technique. Furthermore, since the structural solver is explicit, a smaller time step has to be used for the solid solver than for the fluid solver. For every sub-iterations, the solid solver performs many iterations, until it covers the whole fluid solver time interval; the boundary condition imposed at every sub-step is obtained via linear interpolation of the fluid stresses at time t^n and t^{n+1}.

3.1 The Sails Structural Solver

In this section, we present the shell model used for the simulation of the sail deformation. Consider a shell body of constant thickness h immersed in a fixed reference frame $\{\mathbf{i}_i\}$, $i = 1, 2, 3$. Within the context of the inextensible director shell theory, the geometry of the shell in the reference configuration is described by the mapping

$$\mathbf{X} = \boldsymbol{\Phi}(\bar{\boldsymbol{\xi}}) = \bar{\boldsymbol{\Phi}}(\xi^1, \xi^2) + \xi^3 \mathbf{L}(\xi^1, \xi^2) \quad -\frac{h}{2} \leq \xi^3 \leq \frac{h}{2}, \tag{18}$$

where $\mathbf{X}$ is the position vector of a material point in the shell body identified by the convective system of coordinates ($\bar{\boldsymbol{\xi}} = (\xi^1, \xi^2, \xi^3)$; $\bar{\boldsymbol{X}} = \bar{\boldsymbol{\Phi}}(\xi^1, \xi^2)$ is the position vector of points belonging to the shell middle surface $\mathcal{M}(\xi^3 = 0)$, with boundary $\partial\mathcal{M}$, and the unit vector $\mathbf{L}(\xi^1, \xi^2)$ denotes the director field. In the present formulation, $\mathbf{L}$ is assumed to be normal to the middle surface in the original configuration but, according to the Mindlin-Reissner assumption, it is not forced to remain normal during the deformation.

A shell deformed configuration at time $t \in [0, T]$ is defined by the mapping $\mathbf{x} = \chi_t(\mathbf{X})$ where χ_t represents the motion. The displacements $\mathbf{s}$ at a generic point $\mathbf{X}$ follows from the kinematic description as:

$$\mathbf{s}(\bar{\boldsymbol{\xi}}) = \mathbf{x}(\bar{\boldsymbol{\xi}}) - \mathbf{X}(\bar{\boldsymbol{\xi}}). \tag{19}$$

For a generic solid medium with density ρ and subjected to the body forces $\mathbf{B}$, the equation of motion (momentum conservation) in the reference configuration, can be written as:

$$\rho_0 \frac{\partial^2 \mathbf{s}}{\partial t^2} = \nabla_R \cdot \mathbf{S} + \mathbf{B}, \tag{20}$$

where $\nabla_R \cdot$ is the divergence operator in the reference configuration and $\mathbf{S} = J\sigma\mathbf{F}^{-T}$ is the first Piola-Kirchhoff stress tensor, σ is the Cauchy stress tensor, $\mathbf{F}$ is the deformation gradient and J is the determinant of $\mathbf{F}$.

From the numerical point of view, the structural problem is solved by the Galerkin Finite Element method [72]. More specifically, the MITC4 elements [19] has been used for the space discretization. These shell elements are three-dimensional solid but the typical constraints of the shell kinematics are forced. To avoid shear locking a mixed interpolation of the tensorial component has been adopted [11].

The assumption of large displacements and small strains are considered and a linear elastic constitutive model with an isotropic material is implemented.

After space discretization, the semi-discrete equations of motion governing the structural dynamic response of a system can be written as:

$$\mathbf{M}\ddot{\mathbf{s}} + \mathbf{C}\dot{\mathbf{s}} + \mathbf{F}(\mathbf{s}) = \mathbf{P}, \tag{21}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ the damping matrix and $\mathbf{F}$ the vector of internal force, $\mathbf{P}$ is the vector representing the external loads. $\ddot{\mathbf{s}}$, $\dot{\mathbf{s}}$ and $\mathbf{s}$ are respectively the acceleration, velocity and displacement vectors, collecting all the nodal degrees of freedom employed to describe the solid kinematics.

The central difference explicit method [6] has been chosen for the time integration of equation (21). A variable time increment version of the algorithm is used, in which, inside the single time marching step, the velocity is approximated at the midpoints of the time interval with the following formula:

$$\dot{\mathbf{s}}^{n+1/2} = \frac{\mathbf{s}^{n+1} - \mathbf{s}^n}{\Delta t^{n+1/2}}, \tag{22}$$

while the acceleration can be written as:

$$\ddot{\mathbf{s}}^n = \frac{\dot{\mathbf{s}}^{n+1/2} - \dot{\mathbf{s}}^{n-1/2}}{\Delta t^n}, \tag{23}$$

where $\Delta t^{n+1/2} = t^{n+1} - t^n$ and $\Delta t^n = t^{n+1/2} - t^{n-1/2}$. Finally, the acceleration can be directly expressed in terms of the displacements:

$$\ddot{\mathbf{s}}^n = \frac{\Delta t^{n-1/2}\left(\mathbf{s}^{n+1} - \mathbf{s}^n\right) - \Delta t^{n+1/2}\left(\mathbf{s}^n - \mathbf{s}^{n-1}\right)}{\Delta t^{n+1/2}\Delta t^n \Delta t^{n-1/2}}. \tag{24}$$

Using the approximations (22)-(24) and the equation of motion (21), the nodal velocity can be updated as:

$$\dot{\mathbf{s}}^{n+1/2} = \dot{\mathbf{s}}^{n-1/2} + \Delta t^n \mathbf{M}^{-1}\left(\mathbf{F}^n - \mathbf{P}^n - \mathbf{C}\dot{\mathbf{s}}^{n-1/2}\right). \tag{25}$$

At every time step the displacements $\mathbf{s}$, the external forces $\mathbf{P}^n$ (or $\mathbf{P}^{n+1}$, depending on the coupling) and the viscous forces are known; the internal nodal forces $\mathbf{F}^n$ can be determined by the constitutive law and the strain-displacement relation. So the right-hand side of (25) is known and the velocity can be obtained, the displacement can be subsequently computed using (22).

If the mass matrix $\mathbf{M}$ is diagonal (e.g. due to a mass-lumping reduction) the update of the nodal velocities and nodal displacements can be obtained without solving any linear system. This choice has significant computational advantages: the problem (21) is transformed into an uncoupled system of algebraic equations in which each solution component may be computed independently, with no need for assembly or factorization of any global matrix.

The central difference method is not unconditionally stable so that the time interval Δt needs to be small to satisfy the CFL (Courant-Friedrichs-Levy) condition.

At the moment a simple homogeneous constitutive law for the sail material property is being used but more complex models are under implementations. A modern sailing boat sail is in fact a complex masterpiece of modern technology, with the use of carbon fibres inside two PET (polyethylene

terephthalate) films cooked in a vacuum oven so to get very light but also very resistant sails. The disposition and orientation of the fibres is done in a way that the material properties of the sails varies greatly from zone to zone so that the sail shape under load is the closest to the optimum shape (chosen at the sail design stage).

4 Shape Optimization

For the design of a sailing boat, an efficient numerical scheme ought to be combined with a shape optimization strategy through the minimization of suitable cost functionals [56, 37]. A general abstract formulation of shape optimization problems can be set up as follows: given a set of admissible shapes $\mathcal{O}_{ad}$, find the optimal shape $\hat{\Omega} \in \mathcal{O}_{ad}$ s.t.

$$\hat{\Omega} = \underset{\Omega \in \mathcal{O}_{ad}}{\arg\min}\ J(\Omega, y(\Omega)) \quad \text{with} \quad \mathcal{R}(y(\Omega)) = 0, \tag{26}$$

where $J(\cdot,\cdot)$ is the cost functional to be minimized, $y(\Omega)$ are the state variables, and $\mathcal{R}$ represents the state equations (i.e. the Navier–Stokes equations in the current context).

Many possible ways to achieve this target exist, each bringing its own advantages and disadvantages and therefore some clear choices have to be made. In figure 2, a schematic diagram of the different methodology for shape optimization cited in this section is drawn. In particular, the two approaches adopted in this work have been highlighted with solid and dashed-dot arrows. The results obtained are reported in section 6.

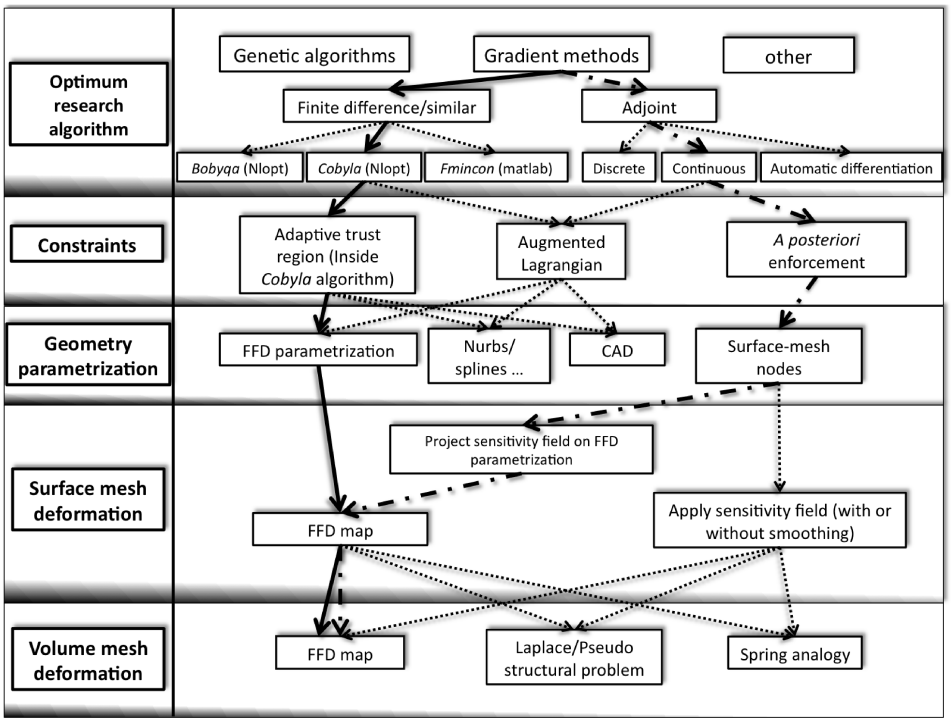


Fig. 2 Diagram showing a range of possible approaches to perform shape optimization, subdivided in the five main steps to be performed. The solid and dashed-dot arrows represent the two approaches developed in this article, while the thin-dotted ones are other possible methodologies taken in consideration by the authors but not used in this work.

4.1 Optimum research algorithm

The next step is the choice of the optimization algorithm. Given a certain cost functional and parameter space, many different ways to reach the optimum can be pursued [56],[37], e.g. the gradient-like methods, genetic algorithms (GA) and neural networks. Gradient-like methods [57] require the gradients of the cost functional and constraints (dependent variables) with respect to the shape design (independent) variables. These gradients, commonly referred to as *sensitivity* derivatives, provide a mechanism for changing the design variables in order to improve the objective function without violating the given constraints. Sensitivity derivatives may be approximated by finite differencing; however, besides being computationally expensive, unless carefully monitored, this approach can produce inaccurate gradient approximations. In this case, the number of control points must be small. The preferable approach [72] is to derive the analytical expression of the sensitivity field, however this requires the solution of the *adjoint* field. The latter can be obtained writing the adjoint equations either at continuous level and using the same approach as for the discretization of the direct fluid problem, or at discrete level, thus via a proper manipulation of the already assembled discrete elements. Another possible technique to obtain the adjoint field is through automatic differentiation [55] algorithms.

In this case, the control points are precisely all the surface mesh points, which may be many. In fact, the advantage of this technique is that the cost of the optimization does not depend linearly on the number of control points, the drawback being that for every optimization step also the adjoint problem has to be solved.

Once the direct and adjoint fields have been computed, the sensitivity field, i.e. the gradient of the cost functional for every surface mesh point, can be obtained. After a proper choice of the sensitivity-descent step size, the surface mesh points position can be updated, and a new optimization step can be computed.

The adjoint approach should thus be very fast to converge to the optimal solution since for every iteration the best descent direction is chosen. A weakness of this approach is that there is no control over the shape that the geometry may assume, since every node is moved independently. If applied directly, this approach may not maintain the smoothness and regularity of the geometry or other features that one may want to preserve, e.g. symmetry. Usually some smoothing iterations of the gradient field are performed and, only after this regularizing steps, the geometry is deformed. In the following, a combination of adjoint-gradient method with surface parametrization technique to improve mesh quality and surface smoothness is proposed.

Another weakness of gradient like methods is that they may converge to local optimum and not to the global one. Genetic algorithms (GAs) [31], 30], on the other hand, have proven their strength in avoiding local extrema and numerical noise in aerodynamic optimization. In problems with constraints, however, they may require a very high number of parameters evaluations to converge. In our case, the cost of GAs methods is still not affordable thus we are oriented in the use of a deterministic approach and we will focus at first only on gradient-like methods.

In this work, two approaches for shape optimization have been chosen. The first one consists in using optimization algorithms that do not require the direct evaluation of the gradient of the cost functional. Thanks to the open source library *Nlopt* [39], many different derivative-free methods are available. For our test cases we have used the *Cobyla* (Constrained Optimization By Linear Approximations) and *Bobyqa* (Bound Optimization By Quadratic Approximation) methods. The *Cobyla* algorithm constructs successive linear approximations of the objective function and constraints via a simplex of n+1 points (in n dimensions), and optimizes these approximations in a trust region at each step [69]. The *Bobyqa* algorithm instead performs gradient-free optimization using an iteratively constructed quadratic approximation of the objective function [70]. As stated before, the drawback of using gradient-free methods is that the cost of the optimization is linearly dependent on the number of control points and, due to the high cost of every cost function evaluation (i.e. the cost of the solution of a full Navier–Stokes simulation), only a small number of control points can be used. It is thus infeasible to use as control points the location of all nodes of the surface mesh to be optimized, which may be in the order of tens of thousands, while a FFD parametrization fulfils very well this requirement.

The second approach that has been explored is based on the solution of the adjoint Navier–Stokes equations for the direct evaluation of the gradient of the cost functional. In this work the continuous adjoint formulation of the Navier–Stokes have been adopted. Here a short description on this approach is reported; for a more detailed analysis we refer to [62, 63].

With the introduction of a Lagrange multiplier λ, problem (26) can be rewritten as an unconstrained minimization problem for the Lagrangian functional L given by:

$$L = J + \int_{\Omega} \lambda \mathcal{R} d\Omega. \tag{27}$$

In the specific case of drag minimization around an object under a steady laminar Newtonian flow, the state equations governing the problem are the Navier–Stokes equations (1) (without the time derivative term) and the cost functional can be expressed as:

$$J = -\int_{\Gamma_{Body}} (T(\mathbf{u}, p)\mathbf{n}) \cdot \hat{\mathbf{u}}_f d\Gamma = \int_{\Gamma_{Body}} (p \cdot \mathbf{n} - 2\nu\sigma(\mathbf{u})\mathbf{n}) \cdot \hat{\mathbf{u}}_f d\Gamma, \tag{28}$$

where Γ_{body} is the boundary associated to the shape to be optimized, $\sigma(u) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ is the *strain rate tensor* and $\hat{\mathbf{u}}_f$ is the unit-vector in the main flow direction. The Lagrange multiplier is given as $\lambda = (\mathbf{v}, q)$, where $\mathbf{v}$ and q are the so-called adjoint velocity and pressure respectively.

If we define as design variables β the deformation of every boundary point in the normal direction , the Lagrange problem (27) can be expressed as $L = L(\mathbf{u}, p, \mathbf{v}, q, \beta)$, and its total variation becomes:

$$\delta L = \delta_{\mathbf{u}} L + \delta_p L + \delta_{\mathbf{v}} L + \delta_q L + \delta_\beta L. \tag{29}$$

The first order optimality condition for the Lagrange functional entails that the first four terms on the right-hand side of equation (29) are equal to zero. The third and fourth term correspond to Navier–Stokes equations and are indeed equal to zero since fields $(\mathbf{u}, p)$ satisfy those equations. The first and second term are equal to zero if the so-called *adjoint Navier–Stokes* equations are satisfied by adjoint variables $(\mathbf{v}, q)$. If we decompose the cost functional into contributions from the boundary J_Γ and from the interior of the domain J_Ω, after integration by parts, the adjoint Navier–Stokes equation are defined as:

$$(\nabla\mathbf{v}^T)\mathbf{u} - (\mathbf{u} \cdot \nabla)\mathbf{v} = -\nabla q + \nabla \cdot (2\nu\sigma(\mathbf{v})) - \frac{\delta J_\Omega}{\delta \mathbf{u}} \quad \text{in } \Omega \tag{30}$$

$$\nabla \cdot \mathbf{v} = -\frac{\delta J_\Omega}{\delta p} \quad \text{in } \Omega, \tag{31}$$

with the following boundary conditions:

$$\begin{cases} \mathbf{v} = -(\frac{\delta J_{\Gamma_{Body}}}{\delta p} \cdot \mathbf{n}) = -\hat{\mathbf{u}}_f & \text{on } \Gamma_{Body} \\ \mathbf{v} = 0 & \text{on } \Gamma_{in} \bigcup \Gamma_w \\ T(\mathbf{v}, q) \cdot \mathbf{n} + (\mathbf{u} \cdot \mathbf{n})\mathbf{v} = 0 & \text{on } \Gamma_{out} \end{cases} \tag{32}$$

They have been derived under the assumption that in the primal problem, Dirichlet boundary conditions on the velocity have been imposed on the inlet and external wall $(\Gamma_{in} \bigcup \Gamma_w)$ as well as a free-stress condition on the outlet (Γ_{out}). The

Furthermore, since in our case the cost functional does not depend on volume values but only on boundary integrals, problem (30-31) reduces to

$$(\nabla\mathbf{v}^T)\mathbf{u} - (\mathbf{u} \cdot \nabla)\mathbf{v} = -\nabla q + \nabla \cdot (\nu\nabla\mathbf{v}) \quad \text{in } \Omega \tag{33}$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } \Omega. \tag{34}$$

The adjoint problem (33-34) has a structure similar to that of the primal Navier–Stokes equations and the solution of its numerical counterpart has been implemented and solved with the OpenFOAM library. Albeit linear, problem (33-34) is computationally not much cheaper than the primal

Navier–Stokes equations being often a stiffer problem and thus the final cost is strongly affected by the actual algorithm implemented in the solver.

Equation (29) thus reduces to

$$\delta L = \delta_\beta J + \int_\Omega (\mathbf{v}, q)\delta_\beta \mathcal{R} d\Omega. \tag{35}$$

Since there is no explicit dependence of the cost functional on the deformation, the term $\delta_\beta J$ is equal to zero.

Finally, it is possible to show, see [83] for details, that the volume integral in (35) can be expressed as a boundary integral defined on the portion of the boundary to be optimized

$$\delta L = \int_{\Gamma_{Body}} \mathbf{g} \cdot \mathbf{n} d\Gamma, \tag{36}$$

reducing its computation cost to a simple post-processing step. It can be demonstrated [62] that, after some simplification, the sensitivity field $\mathbf{g}$ can be approximated as:

$$\mathbf{g} = \mathbf{A}(2\nu\sigma(\mathbf{u}) : \sigma(\mathbf{v}))\mathbf{n}, \tag{37}$$

where A represents the area of every face.

4.2 Parameters constraints

Another issue to be addressed is how to treat the control parameters constraints. Usually the object to be deformed cannot deform freely but is subject to some geometrical constraints, like keeping the constant or imposing a minimum size of certain parts of the shape to be optimized. One of the most common methods to enforce a constraint is to add a penalization term inside the cost function whose magnitude is adjusted dynamically until the constraint is satisfied. In this respect, a very popular method is the augmented-Lagrangian method [74].

Another more empirical technique consists instead in recovering the constraint *a posteriori*, i.e. after the shape deformation due to the sensitivity field, via a second mesh motion. For example, in case of a volume constraint, after each sensitivity-based deformation, the shape can be inflated/deflated homogeneously, i.e. same amount in the normal direction of every face, until the required volume is obtained.

Finally, some optimum research algorithm, like *Cobyla*, satisfy the constraints modifying the trust region dynamically at each step based on the information available at the previous iterations. Notice that during the initial stages the constraints may not be respected, but as soon as more data are available and the magnitude of deformation becomes smaller, the constraints requirements are fulfilled. In this work the *a posteriori* recovery technique [69] and the adjustable trust region technique have been used.

4.3 Geometry parametrization

At this point, decision on how to model the geometry (and how to control the morphing process) has to be taken. Usually, industrial geometries are drawn using specific *CAD* programs, e.g. *Rhino* [14], *AutoCad* and *Maya* [26], *SolidWork* [82], *etc.* Those geometries are then exported in a suitable format and imported into the mesh generator, where a space discretization of the domain is produced.

One can therefore, at every iteration of the shape optimization algorithm, modify the geometry directly inside the CAD program, and then re-export it to the mesh generator. By so doing, one has to link the shape optimization method with the CAD program, create a new geometry assembly (a process which may not be easy to automatize on complex geometries) and finally generate a new

mesh. In case of unstructured meshes this last part, although very expensive, may be relatively easily automatized, while in case of structured meshes, like the one typically used by the authors for the hydrodynamic solver component, the automation is usually not feasible. For those reasons we have decided not to follow this approach.

Another possible way to proceed is to deform directly the computational mesh. By this approach one can decide whether to modify just the nodes that define the mesh-surface of the geometry that one wants to optimize, or to deform the whole mesh. In the first case, a suitable numerical algorithm has to be used in order to modify the volume mesh according to the imposed surface-mesh displacement. In the mesh motion framework, many alternatives exist. The most common one is to solve a Laplace problem on the whole domain using as boundary conditions the surface displacements. Another popular mesh-motion technique is the so-called spring analogy model [21]. Many variants and sophisticated numerical modification of those schemes can be implemented in order to help preserving the mesh quality. The important advantage of this choice is that the mesh must not be recreated but simply modified. The disadvantage is that nothing assures that the quality of the mesh is maintained, and, in case of large deformations, the mesh can even become invalid. In this case either special untangling schemes are adopted, see for example [15], or the mesh has to be fully regenerated. Furthermore, the solution of the internal mesh nodes displacement may have a not negligible computational cost.

For the actual shape parametrization, different techniques can be used, like *B-splines* [7], *NURBS* [67], or similar. By changing the value of the shape parameters, representing for example the position of the control points of the spline, the shape is modified. An "interpolator" to traduce the continuous shape displacement to the discrete level, i.e. to the mesh, is needed.

Another possible technique is that based on the so called Free Form Deformation (FFD). Originally introduced in the late 70s [80], FFD have been at first used in computer graphics; only in recent years they have been employed in optimal design problems [3].

Recently, this technique has been coupled with Reduced Basis methods [44, 77, 47] and used also for simple FSI problems [42, 43]. A free-form deformation operates on a bivariate Bezier control area, built around (and regardless of) the shape we want to optimize, manipulating a lattice of control points. In this way, design parameters are directly connected neither to geometrical properties nor to the shape boundary, see figure 3. This is at the same time a great advantage and disadvantage of this technique: on the positive side it does not bound the optimization process to a restricted subset related to geometrical quantities; on the other hand it is very difficult to control some geometrical quantities, like, e.g. the length of an edge, the angle of attack of a foil, in a "classical" way.

A brief description follows in the simple case of a 2D computational domain. Given a fixed rectangular 2D domain D s.t. $\Omega \subset D$, we assume the existence of a differentiable and invertible map $\Psi : (x_1, x_2) \to (s, t)$ such that $\Psi(D) = (0, 1)^2$; by this *freezing* procedure, FFD can be defined in a simpler way in the coordinates (s, t) of the spline parameter space $(0, 1)^2$. We thus select a lattice of $(L+1) \times (M+1)$ unperturbed control points $\mathbf{P}_{l,m} = [l/L, m/M]^T$, $l = 0, \ldots, L$, $m = 0, \ldots, M$, and modify the object by moving control points to a new position. The corresponding perturbed control points $\mathbf{P}^o_{l,m}(\boldsymbol{\mu}_{l,m}) = \mathbf{P}_{l,m} + \boldsymbol{\mu}_{l,m}$ are thus specified by a set of $(L+1)(M+1)$ parameter vectors $\boldsymbol{\mu}_{l,m} \in \mathbb{R}^2$, giving in all $2(L+1)(M+1)$ possible degrees of freedom. In general, among the control points $\mathbf{P}_{l,m}$, we indicate the effectively free scalar-valued parameters chosen as design variables as $\mu_1, \ldots, \mu_P$ – each corresponding to the displacement of a control point in the s or t direction – and define the parametric map $\tilde{T}(\cdot, \boldsymbol{\mu}) : D \to \mathbb{R}^2$ by which the uploaded geometry is computed as follows:

$$\tilde{T}(\mathbf{x}; \boldsymbol{\mu}) = \Psi^{-1} \left(\sum_{l=0}^{L} \sum_{m=0}^{M} b^{L,M}_{l,m}(\Psi(\mathbf{x})) \mathbf{P}^o_{l,m}(\boldsymbol{\mu}_{l,m}) \right), \tag{38}$$

where $b^{L,M}_{l,m}(s, t) = b^L_l(s) b^M_m(t)$ are tensor products of one-dimensional *Bernstein basis polynomials* $b^L_l(s) = \binom{L}{l} s^l (1-s)^{L-l}$ and $b^M_m(t) = \binom{M}{m} t^m (1-t)^{M-m}$ defined on the unit square $(s, t) \in [0, 1] \times [0, 1]$. Finally, we have $\Omega_o(\boldsymbol{\mu}) = T(\Omega; \boldsymbol{\mu})$, by using the restriction $T = \tilde{T}|_\Omega$.

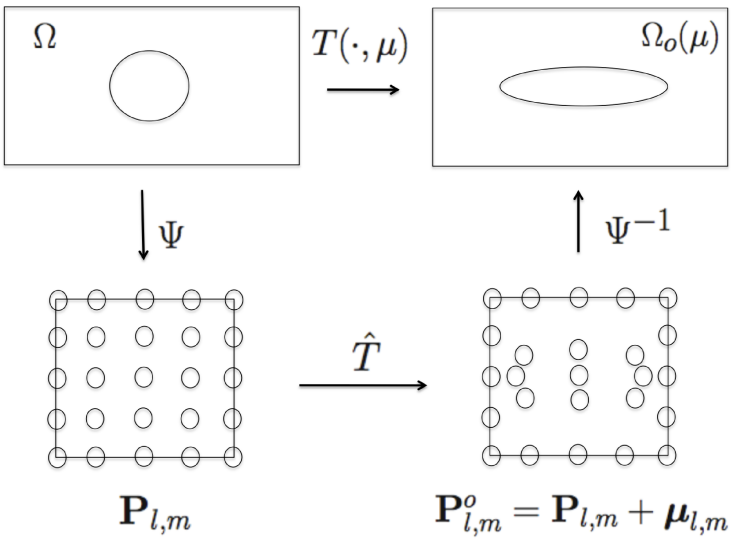


Fig. 3 Schematic illustration of the Free From Deformation (FFD) technique. Through a series of maps and transformation, the geometry is deformed following the movement of the control points.

4.4 Surface and volume mesh deformation

Once the new value of the shape parameters is obtained, the surface mesh describing the shape has to be deformed. If the shape has been parametrized via FFD, the new position of the surface mesh nodes is obtained applying the FFD map to every node. If, instead, the adjoint approach has been used and the sensitivity field is known for every surface mesh node, two approaches can be adopted: either move the mesh-points directly, or "project" them to some other shape parametrization. In the first case, usually some smoothing of the sensitivity field has to be done otherwise the nodes near the edges and corners tend to be over-deformed, the mesh anisotropy may create small oscillations and in general the smoothness of the deformed shape is not guaranteed. Another possibility is to use once again the FFD parametrization, projecting the sensitivity field on the FFD control parameters. This time we are not limited to a small lattice and can indeed choose a very fine one, in order to reduce the loss of information due to the projection below an acceptable threshold. This projection can be accomplished for example by a least square interpolation of the surface mesh points and the FFD control points:

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\boldsymbol{\mu}_{ad}}{\arg\min} \sum_{i\in Surfnodes} (\mathbf{x}_i - \tilde{T}(\Psi(\mathbf{x}_i);\boldsymbol{\mu})\,)^2, \tag{39}$$

where $\tilde{T}(\Psi(\mathbf{x}_i);\boldsymbol{\mu})$ is the FFD map as defined in section 4, equation (38), and $\boldsymbol{\mu}_{ad}$ is the domain of admissible $\boldsymbol{\mu}$ such that the FFD control points $\mathbf{P}^o_{l,m}$ stay inside the domain $[0,1]\times[0,1]$. Furthermore, stricter constrains on the deformation $\boldsymbol{\mu}$ are usually imposed in order to avoid excessive mesh deformation. Without the interval constraints, the least square minimization reduces to the solution of a linear problem, with size the number of FFD control points, of the form

$$F^T F\hat{\boldsymbol{\mu}} = F^T\mathbf{x}, \tag{40}$$

where F is the FFD map matrix and $\mathbf{x}$ is the vector containing the displacement of the surface mesh points. The solution of the same problem but with interval constraints is a typical problem of linear programming and its solution requires that of an optimization sub-problem by itself. In order to save computation time, we have chosen to use a numerical trick to maintain the magnitude of the control points small: we impose some extra line to problem (40) thus requiring m to be close to zero (this turns out to add an identity matrix at the bottom of F and a vector of 0 at the bottom of x).

Although not equivalent to the constrained minimization problem, this technique allows to obtain all the control points displacements with a single solution of the now modified problem (40). We believe that this inaccuracy introduced is justified by the fact that the error committed is anyway smaller than the one introduced by the other numerical steps of the optimization procedure.

Once the new shape has been evaluated, the volume mesh can be deformed either by solving a modified Laplace problem or directly using the FFD map. We believe, and our numerical results are in agreement with our conjecture, that using the FFD map is the best solution in this case. The mesh motion technique is in fact mesh-independent and thus is not affected by the typical problem generated by Laplace-like methods with small and skewed elements and in general produces meshes with higher quality.

5 Numerical Results

In this section an overview of the numerical results that have been obtained is presented. Results concern free-surface flows and hull dynamics, shape optimization and fluid-structure interaction problems. In figure 4 a flow diagram of the entire simulation and optimization processes for a sailing boat is drawn. Obviously, some parts are shared by more than one part, like the Navier–Stokes solver which is used for simulating both the water field and air field or the mesh motion technique which are found in both the boat dynamics and the optimization steps. For every sub-section, a diagram of the numerical/methodological tools needed by that section is outlined.

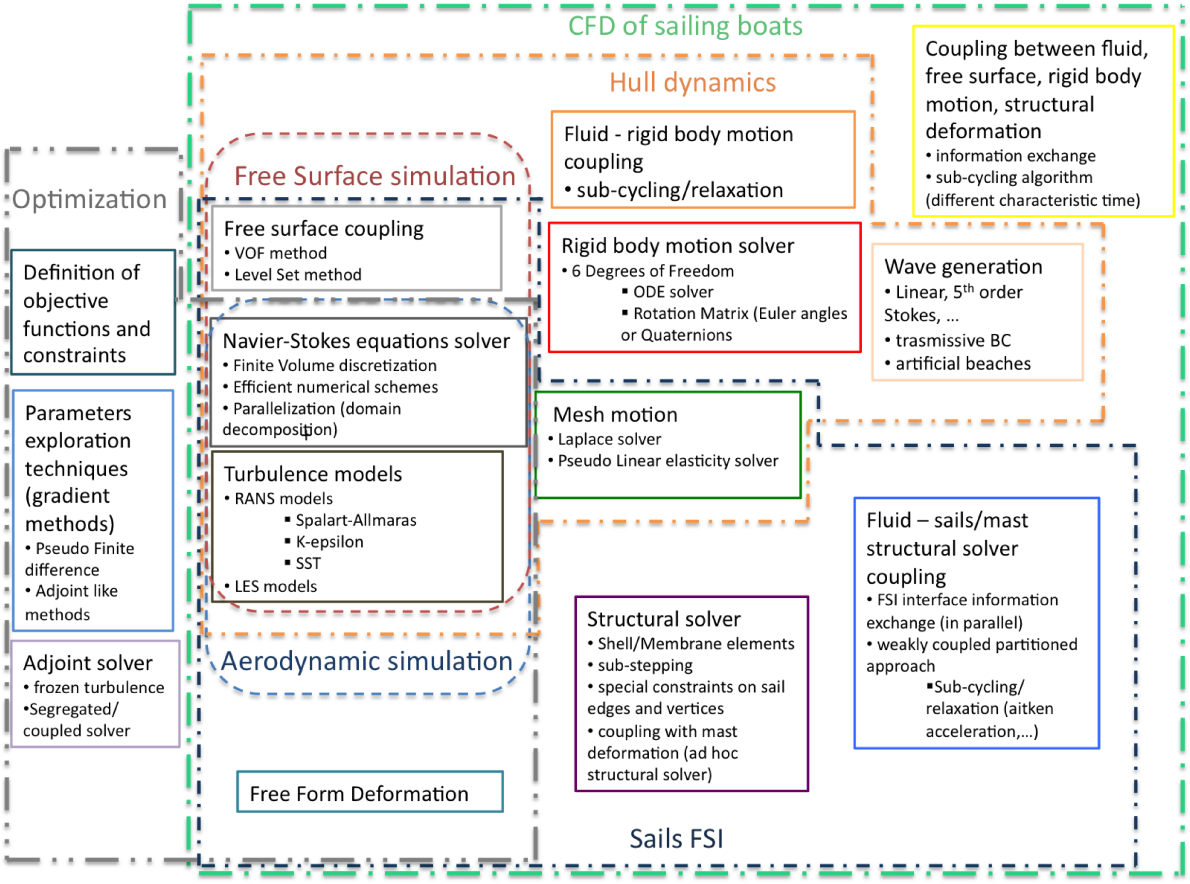


Fig. 4 Overview of the different methodologies and tools considered in the this work.

5.1 Hull Dynamics

As described in the previous sections, the numerical tools needed for the simulation of the hull dynamics are (figure 5): a Navier–Stokes solver with free-surface capabilities, suitable turbulence models, a dynamic model for the rigid body of the hull and a mesh motion solver. Furthermore, if one is interested in the simulations of waves, a mathematical model for the imposition of the correct boundary condition in order to generate and maintain the wave on the whole domain is needed [49, 23, 41].

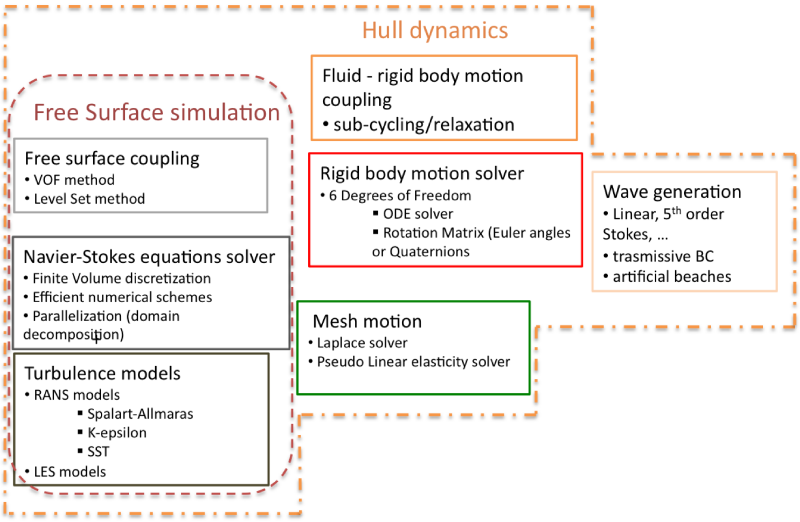


Fig. 5 Overview of the different methodologies and tools considered for the simulation of the hull dynamics.

The free-surface solver and its interaction with the rigid-body hull dynamic module have been tested on a classical benchmark problem, namely the Series 60 hull with a block coefficient $C_B = 0.6$ (see [86]). We have considered a flow at a Reynolds number Re=$4 \cdot 10^6$ and a Froude number Fr=0.316. First, the steady solution at a fixed attitude has been computed on three-different grid resolutions with a number of cells of around 70000, 200000 and 650000, respectively and a time step $dt = 0.001$ s. The wave profiles on the hull obtained with the different grid resolutions are presented in figure 6 and compared with the experimental measurements. These results together with the drag coefficients reported in Table 1 show a good convergence to the experimental data.

Grid	C_p	C_v	C_t
Coarse	0.00274	0.00344	0.00619
Medium	0.00179	0.00372	0.00552
Fine	0.00163	0.00388	0.00551
Experiment			0.00542

Table 1 Pressure, viscous and total drags coefficient for different grid resolutions.

Starting from the steady solution obtained with the medium resolution grid, we have activated the hull dynamics module and left the hull free to sink and pitch. The time evolution of the two degrees of freedom towards the hull running attitude (hydrodynamic equilibrium) are presented in figure 7.

Finally, we tested the behaviour of the model simulating the dynamics of the hull in waves. This was achieved imposing at the inflow an incoming wave modelled using the fifth order Stokes wave expansion, see [66] for more details. The wave model sets a time dependent wave elevation and the correspondent orbital velocity at the inflow boundary of the domain. We have considered a wave amplitude $\lambda = 0.01L$ where L is the boat length, about 3.5m for the problem at hand, and a wave

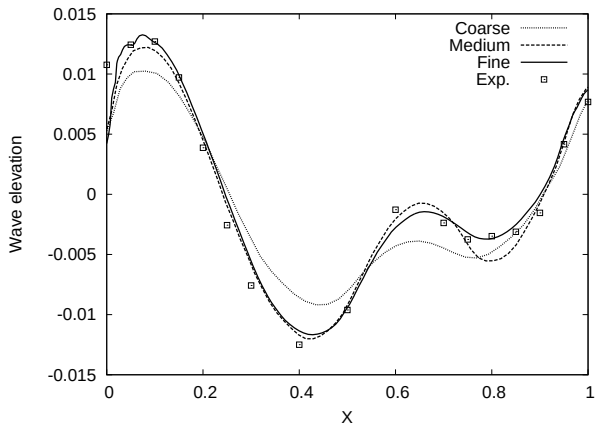


Fig. 6 Wave profiles along the Series 60 hull for different grid resolutions

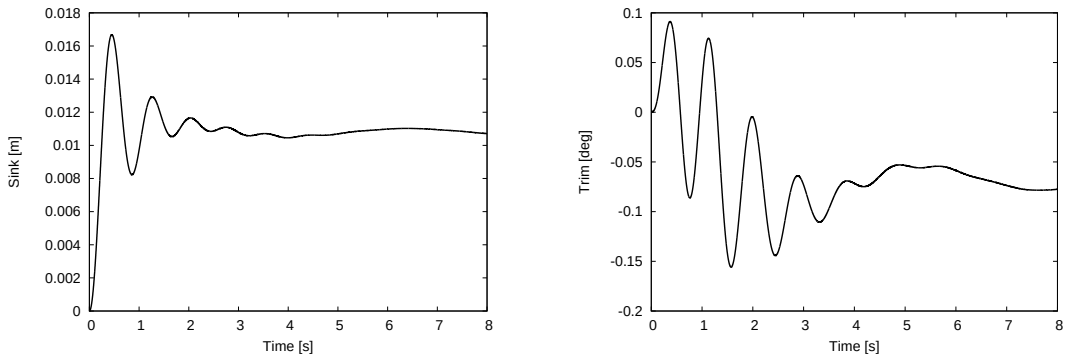


Fig. 7 Sink (left) and trim (right) evolutions of the Series 60 hull towards hydrodynamic equilibrium.

frequency correspondent to an incoming wave length of 0.2L. In figure 8 the free-surface elevation is represented: the incoming wave and its typical pattern modification in the boat wake can be clearly observed.

A comparison between the time evolution of drag and pitching moment for the two dynamics case (with and without the incoming wave) is reported in figure 9.

Free-surface simulation of this type has been extensively adopted during the past few years in the framework of a collaboration of our research group with the Alinghi Team for the design of the boats which participated to the last three America's Cup campaigns preparation [64, 65, 16].

In particular, more recently, the focus has been on the simulation of the free-surface flows around the Alinghi multi-hull catamaran designed for 33rd America's Cup. A large scale simulation analysis was performed to predict the drag and lift on the hulls in many sailing conditions and attitudes. All those data were then integrated in a Velocity Predictor Program (VPP) that estimates the performance of a sailing yacht, for given boat design and sailing conditions.

For this analysis, different simulation software were used, including the OpenFOAM free-surface solver described above, the Ansys CFX free-surface solver [27] and a linear free-surface potential solver, with the viscous component predicted via ITTC formulas, developed by the Alinghi Design team. In figure 10, the drag prediction obtained with the different codes are compared with the experimental results. As can be seen, OpenFOAM data trend is in line with the other numerical solvers, predicting drag values slightly higher than CFX. The potential flow solver under-predicts the drag values, which is acceptable since the viscous contribution is obtained through empirical

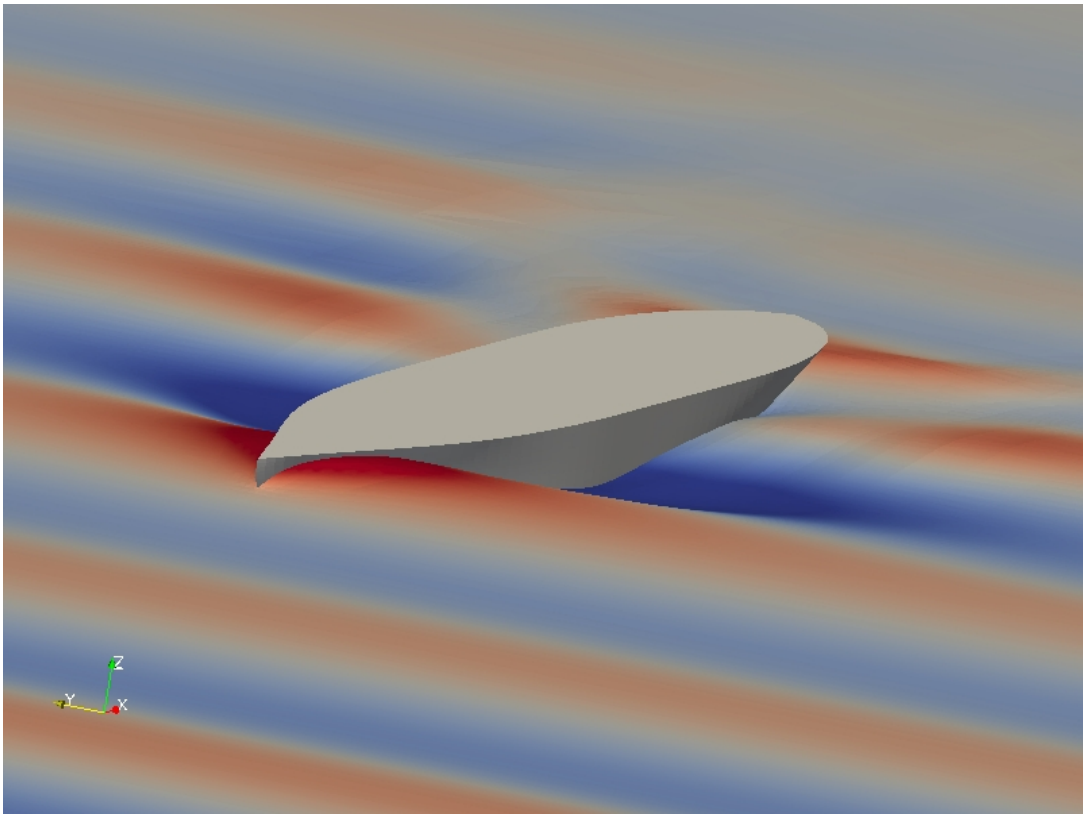


Fig. 8 Series 60 dynamic simulation with waves: free-surface elevation.

formulas and flow solver is based on a more simplified model. The drag values obtained via experimental test in a towing tank are quite close to the numerical values obtained (less than 5% of error on average), with higher values for small trim angles and lower values for high trim angles.

5.2 High Performance CFD: Parallel Scalability

An investigation on the parallel scalability of the code has also been carried out and some important features that allowed us to fully benefit of the parallel computational power have been higlighted. We compared the parallel performance of the free-surface solver implemented in OpenFOAM on a clusters available at the Ecole Polytechnique Fèdèrale de Lausanne. The technical specifications of the cluster are: 56 bi-processor nodes of Intel Xeon Nehalem quads-cores at 2.66 GHz with 24 GB of RAM and connected via InfiniBand.

A transient free-surface Navier–Stokes simulation for the Series 60 hull with a large grid (about 4 millions cells) has been considered as test case.

A brief scalability analyses has the been performed on the Nehalem cluster and results are reported in table (2).

nodes*core	# partitions	Wall Clock Time
4*4	16	647
2*8	16	664
4*8	32	350
8*4	32	296
8*8	64	177

Table 2 OpenFOAM scalability test on the Nehalem cluster.

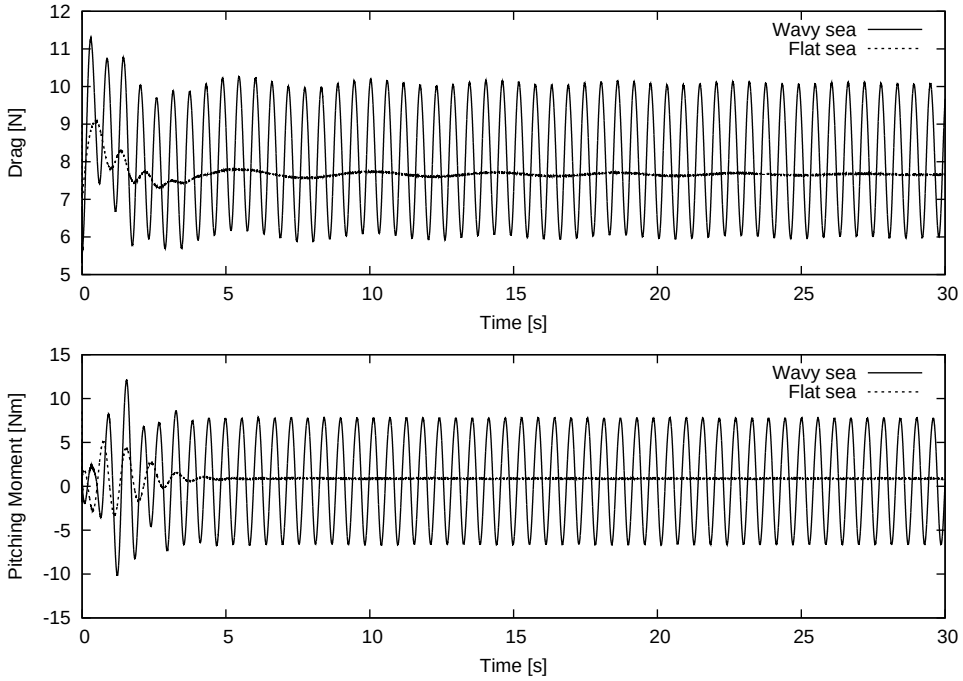


Fig. 9 Drag (top) and pitching moment (top) for the dynamic Series 60 simulations in wavy and flat sea.

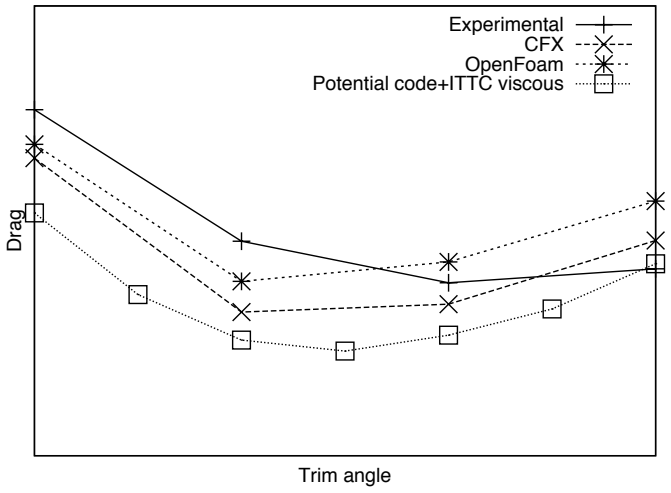


Fig. 10 Drag prediction results on the Alinghi AC33 catamaran hull obtained with different numerical tools and compared with the experimental results. No axis values are reported due to confidentiality agreement with the Alinghi team.

As reported in the table, OpenFOAM, for a fixed number of partitions, is significantly slower when using all cores of a node rather than twice the number of nodes but using only half of the cores available per node. this is due to the fact that when using all 8 cores a bit of memory communication bottleneck arises. When keeping the number of cores per node fixed and increasing

just the number of nodes, the scalability factor is instead very high (at least for the number of partition tested): namely 95-98%.

OpenFOAM has been installed also on the EPFL cluster BlueGene-P [30], whose major strength is not the speed of every single node, only 800 MHz, rather the possibility to use a very large number of cores, up to 16000 in this case, and the highly optimized memory communications. In order to be able to use this cluster efficiently, the solver must scale well up to a high number of partitions. Special attention has thus to be taken in compiling the code properly so to benefit from the special architecture.

Two different grid decomposition methods have been considered in this case: the former, referred to as *Xdir* decomposition, simply subdivides the grid in slices (each containing approximately the same number of grid cells) in a given direction (in this case the stream-wise direction X); the latter is based on the *Metis* [40] library, which decomposes the grid trying to minimize the number of faces on the partition interfaces.

The results obtained with the the partitioner Xdir shows a deterioration of the scalability property of the solver when increasing the number of partitions beyond 64, see figure 11 . This is due to the fact that with this partitioner, as the number of domains increases, the number of faces on the partition interface increases linearly. Since the amount of data passed from core to core is proportional to this value, it is easy to reach the memory communication bottleneck. With the Metis partitioner, much better scalability is achieved: 87.5%, up to 512 partitions. When using a even higher number of partition, i.e. 1024, no gain is observed compared to the run with 512 partitions. This is due to the fact that, the test case having 4 million cells, when using 1024 partitions every partitions has about only 4 thousand cells. This number is too small for the solver to be efficient requiring a too short computational time for every node when compared to the communication lag time. If a bigger test case were to be chosen, the solver would be able to scale efficiently up to a higher number of partitions. For BlueGene-P as well, the fact of using just one, two or all four cores of every node has some effects on the scalability, although not as much as on classical clusters.

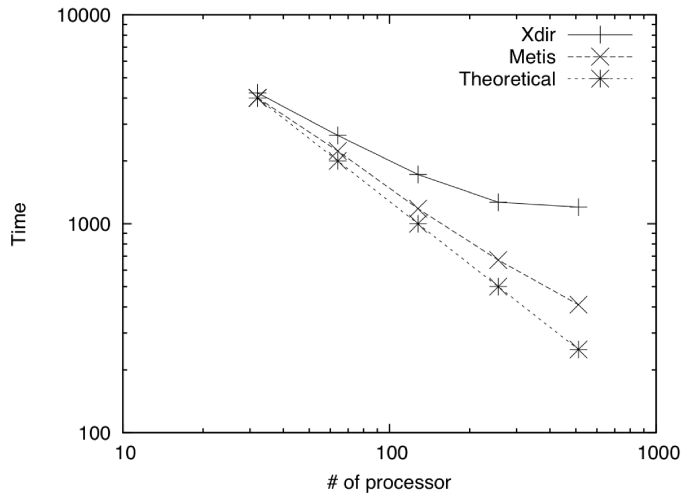


Fig. 11 Scalability trend of the solver OpenFOAM on the Bluegen/P cluster for two different partitioning method.

5.3 Fluid-Structure Interaction

In figure 12, the main tools for the simulation of the Fluid-structure interaction of a sail is reported. The Navier–Stokes solver and the turbulence model are the same one used in the previous section, while the mesh motion is in general more critical due to the higher deformation, thus special care has to be taken to preserve the quality of the mesh. The structural solver for the simulation of the sail deformation under the aerodynamics loads and the coupling algorithm have been introduced in section 3.

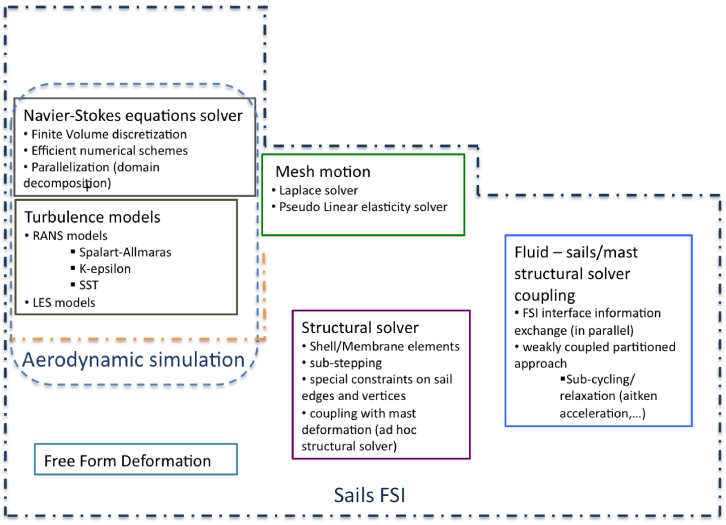
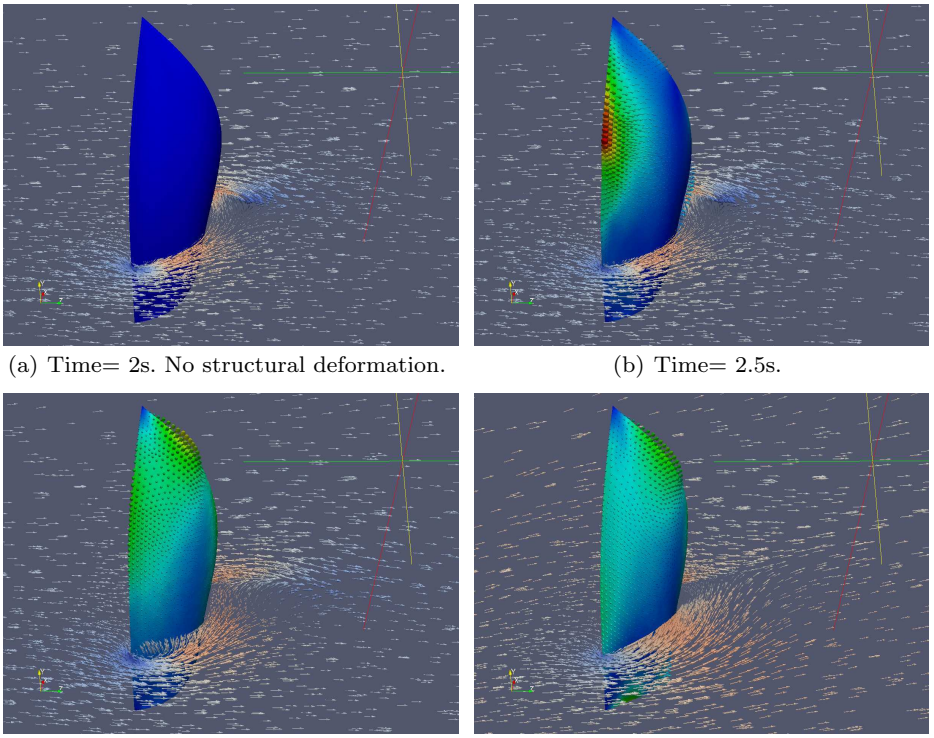


Fig. 12 Overview of the different methodologies and tools considered for the simulation of the Fluid-Structure interaction of the sail.

The fluid structure interaction between wind and sails has been analysed. An external structural solver capable to simulate the membranes/shells structural deformations have been used. The coupling algorithm between the two solver is the one described in section 3.1, with the extra complexity of the communication and data interpolation from the two independent solvers. In this case the fluid solver has been considered as the master program, being responsible of the interpolation of the stresses from the fluid mesh to the structural solver and the interpolation of the sail displacement to the fluid mesh, as well as starting and pausing the structural solver when needed. The meshes used for the flow solver, generated with the commercial code ICEM [52] are made of hexahedra, tetrahedra and pyramids and are not geometrically conforming with the one of the the shell solver. As initial condition, the steady converged solution of the flow field obtained while keeping the sail fixed has been used. At first a single sail, a gennaker, immersed in a steady atmospheric flow has been simulated; in figure 13 the flow field and the sail deformation are presented at different time-steps.

Many different configurations are currently under investigation and the results are so far very encouraging. For example, a variation of the structural property of the sail has been investigated: setting the width of the sail to a higher value, produce a final shape that is rounder near the bottom and flatter near the top, see figure 14, and in general less deformed. This is due to the fact that now the sail is heavier and more rigid: the weight of the sail plays now a more important role. Furthermore, from a numerical point of view, this configuration is easier to compute from a FSI point of view, since the inertia of the sail is higher: in good agreement with this concept, the number of FSI sub-iterations for every time step drop from about 50 to 20 (for a relative convergence criteria of 0.05%).

In these simulations the three vertices of the sails are kept fixed, but the possibility to insert more realistic constraints is possible. More complex configurations, such as multiple sails (main sail



(a) Time= 2s. No structural deformation. (b) Time= 2.5s.

(c) Time= 4s. (d) Time= 6s.

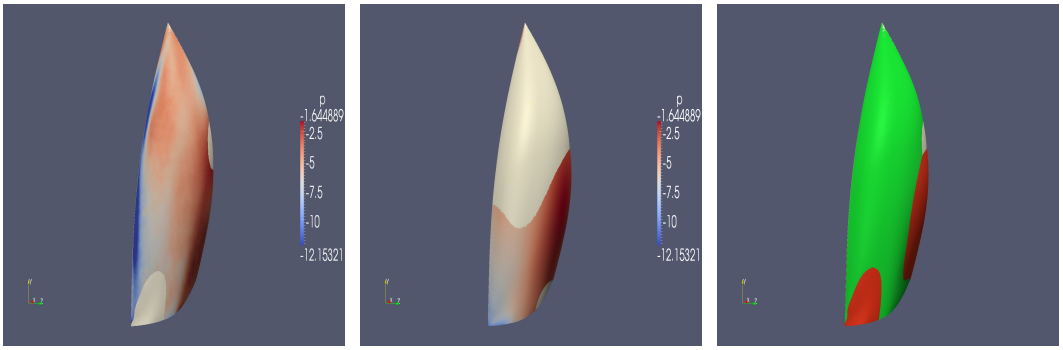
Fig. 13 Gennaker FSI: velocity vectors on a slice at Y=15 m and displacement vectors and displacement magnitude on the sail surface.

plus gennaker), mast deformation and full sailing boat dynamics are currently under investigation and will be reported in a future paper.

In terms of efficiency, the resolution of a full transient FSI problem is much more demanding than the sole aerodynamic simulations. To give a general idea of the FSI overhead hereafter we provide some data of the computation time for one of the simulation described above run on 32 i7 2.66GHz processors. The cost for one time step of the aerodynamic simulation is about 8 seconds. The cost of a single FSI sub-iteration step comprehends the aerodynamic solver, about 8 seconds, the mesh motion, about 2-5 seconds (depending on the mesh motion technique) plus the structural solver. The latter, being an explicit solver in our case, usually requires a much smaller time step than the fluid one and thus it requires several sub-iterations. The constraint on the structural solver time step depends heavily on the structural property of the sail: a very stiff material induces very fast modes on the structures and thus a very small time step constraint is required to be stable. In our case, the structural solver required about 2-5 seconds for FSI sub-iteration. Finally, for every time-step, there are about 20-50 FSI-sub-iterations, mainly depending on the time step chosen, convergence criteria and structural property of the sail, and thus the total time for a FSI time step is about 30-100 times longer than the one of a simple aerodynamic simulation.

6 Shape Optimization Test Cases

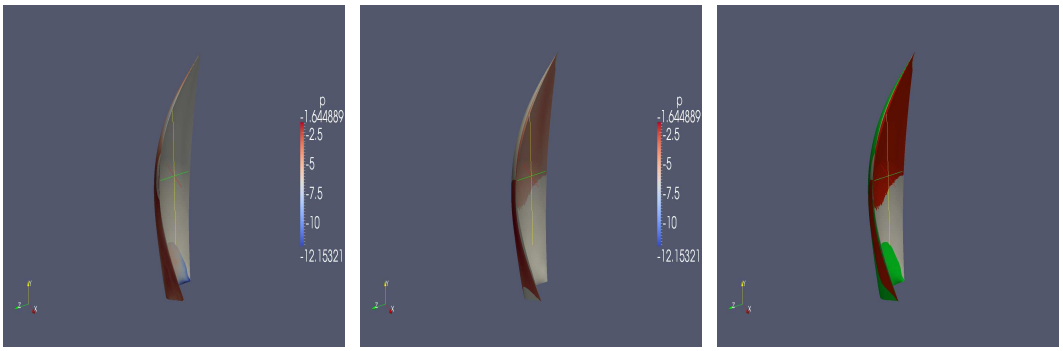
For the shape optimization, besides the solver needed for the solution of the Navier–Stokes equations, a solver for their adjoint counter-part may be required, as well as a proper geometric parametrization technique (figure 15). Once the objectives function and constraint are defined, a module that takes care of the parameters exploration and that adjusts the mesh accordingly has to be implemented.



(a) Base run final shape (pressure) vs initial position (white)

(b) Higher width run final shape (pressure) vs initial position (white)

(c) Base run (green) vs Higher width run (red) vs initial position (white)



(d) Base run final shape (pressure) vs initial position (white)

(e) Higher width run final shape (pressure) vs initial position (white)

(f) Base run (green) vs Higher width run (red) vs initial position (white)

Fig. 14 Comparison of the final shape (i.e. at convergence) of the sail for different structural properties and versus the initial shape. View from the front and from the side.

The Free Form Deformation method has been chosen for the geometry deformation. An analysis of the FFD method aimed at shape optimization algorithm on ship-related geometries, like the bulb and the rudder of a sailing boat, has been done at first with the use of the code COMSOL [81] for the solution of the Navier–Stokes problem and used the Matlab Optimization toolbox for the exploration of the parameters set. This work has showed the good potential of this approach, although it also outlined the necessity to use much more efficient numerical tools to obtain reasonable computation time even for simple test problems.

In fact, although Matlab is a very powerful program with many built-in functions, it is fairly slow compared to an efficient C/C++/Fortran code. Furthermore, since our final goal is to have an automatized program for shape-optimization, the code have been rewritten in C++ in order to allow us a better integration with OpenFOAM.

A FFD class has been implemented directly into OpenFOAM so that a call can be made inside the code, using the flexibility of the C++ class structure, to update the position of the mesh points with the FFD module. This allows us to use both approaches of mesh motion based on the FFD: one can either decide to move all points, both surface and internal nodes, or just to move the surface meshes and then use the other options, i.e. laplacian solvers, already implemented in OpenFOAM for the movement of the internal nodes. Special care has also to be taken so that the code works in a parallel environment as well.

The test case that we have chosen is a 2D cylinder immersed in a flow in laminar regime, $Re = 40$, with the drag as cost functional to minimize under the constraint of fixed volume. The shape has been parametrized via a FFD lattice made of 5x5 control points set around the cylinder inside the domain. Out of the total 50 degrees of freedom (25x2, because every control points can move in both longitudinal and vertical direction) we have chosen to keep fixed the external points, so that the points on the FFD lattice border are not deformed and thus the FFD map can be used

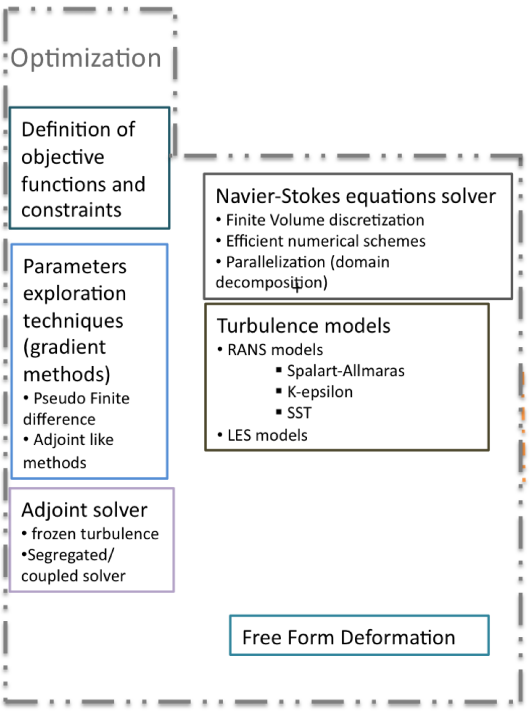


Fig. 15 Overview of the different methodologies and tools considered for the shape optimization problem.

also for the volume mesh deformation. Thanks to the symmetry with respect to the horizontal axis of the problem and geometry, the same symmetry condition can be imposed on the control points movement in the vertical direction. We have thus chosen to select only 6 control points, 3 in the longitudinal direction and 3 in the vertical direction (which counts as double thanks to the symmetry), as represented in figure 16.

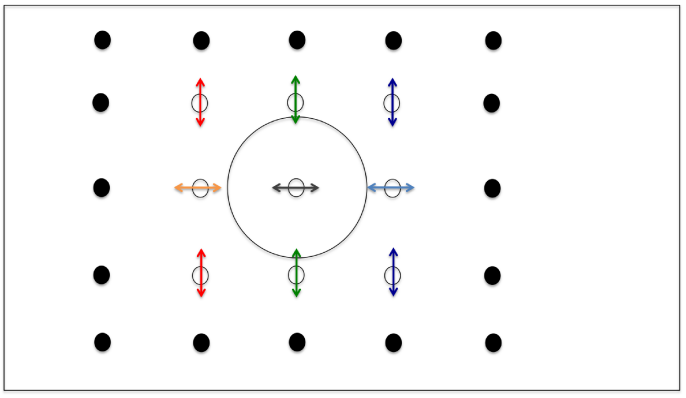
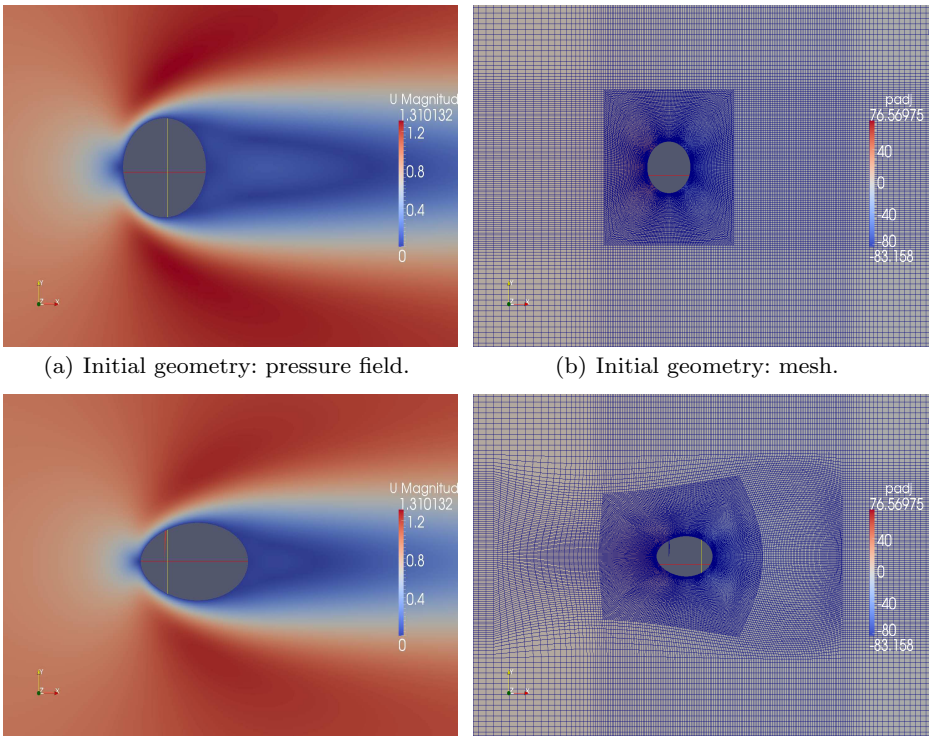


Fig. 16 FFD lattice over the 2D domain for the shape optimization of a cylinder. Although the lattice is composed by 5x5=25 control points, only 6 FFD points have actually been chosen as control points. The full black points are kept fixed, while the empty one can move in the arrows directions. Control points with the same colours means that the two movement have the same value but opposite sign (symmetrical displacement).

Besides the volume constraint, a bound-constrain of [-0.4;0.4] has been imposed to the value of the control points. The optimization problem has been solved via the *Cobyla* algorithm and the results are reported in figures 17-18. In the first figure, the qualitative comparison between initial and final shape and corresponding meshes is reported. As can be seen, the magnitude of the deformation is quite big, although only 6 control points have been used, and the mesh deformation is very smooth. Quantitatively, the new shape has reduced the drag by 11-12% in about 80 iterations, but more than half of them have been spent in very small adjustment to satisfy with high accuracy, i.e. 10^{-4}, the volume constrained. With a less restrictive constraint about 40 iterations would have been sufficient.

The exploration of the parameters space at first does not necessarily produce a better geometry, neither fulfil the volume constraints. As more information are available and the control parameters influence is better defined, the descent to the (local) optimum is achieved and the calibration of the penalty terms for the volume constraint imposed so that the volume condition is satisfied.



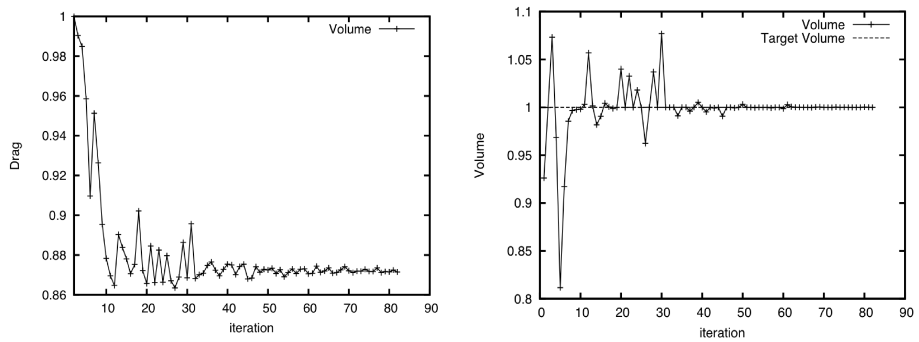
(a) Initial geometry: pressure field. (b) Initial geometry: mesh.

(c) Optimized geometry: pressure field. (d) Optimized geometry: mesh.

Fig. 17 Shape Optimization of a 2D cylinder in laminar regime. Comparison between the initial geometry and optimized ones.

We then consider a similar test case, where we look for the drag minimizing shape in 3D, Re=40, starting from an initial spherical shape and considering the volume fixed. In this case the adjoint-based gradient method has been used and the volume constraint has been recovered at every iteration via a-posteriori mesh motion. The sensitivity field has been projected to a double FFD lattice around the sphere (see figure 19): an external one to be able to capture the macro-deformation and have them "absorbed" by a wide region of the domain, and an inner one to capture more details. In this case as well, the outer control points of the lattices are kept frozen, so that the FFD mapping can be used for both surface and volume mesh deformation.

The methodology has proven to be robust and efficient, see figure 20-21. The magnitude of the deformation is very important and, although no symmetry condition is imposed this time, the optimal shape is symmetric with respect to the span-wise and vertical directions (as expected). The sensitivity field and the volume constraints tend to deform the shape in opposite ways; indeed, the sensitivity field tends to make the shape smaller while the volume constraints recovery re-inflates



(a) Drag. (b) Volume.

Fig. 18 Shape Optimization of a 2D cylinder in laminar regime. Cost functional: drag minimization. Constraint: fixed volume. Method: *Cobyla*. The data in the plot have been non-dimensionalized with the initial (i.e. not-deformed) values.

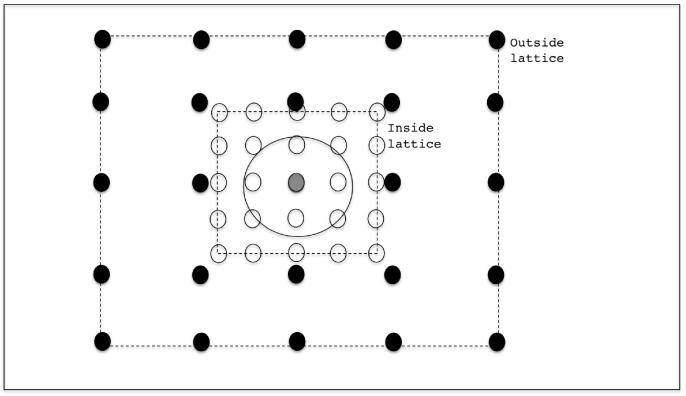
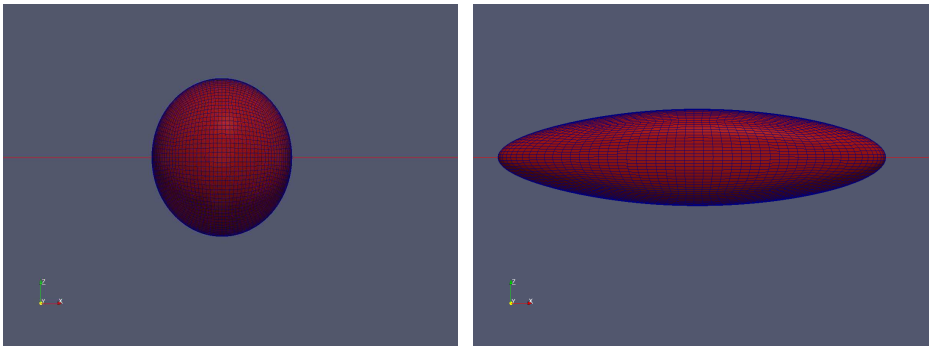


Fig. 19 2D section of the two-levels FFD lattice over the 3D domain for the shape optimization of a sphere. External lattice: black dots. Internal lattice: white dots.

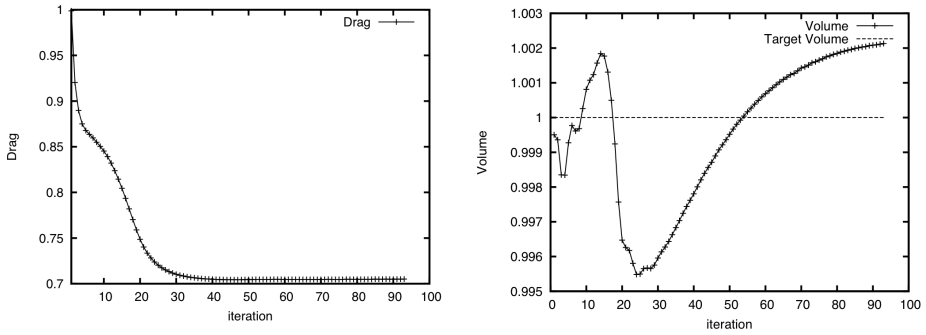
it. Thus the optimization stopping criteria is reached when the sensitivity field is uniform in the normal direction all over the shape. At the beginning, the sensitivity field mainly tends to squeeze vertically the sphere and the volume constraints, acting uniformly, makes the resulting deformation equal to a vertical/span-wise contraction and longitudinal expansion. At convergence, being the shape already elongated, the sensitivity field requires a uniform contraction in all directions which is counter-balanced by the volume constraint.

The optimized shape is about 30% more efficient than the initial one while maintaining the same volume. The optimum search, thanks to the gradient evaluation, is monotonic and stops after about 80 iterations, although after 40 iterations a value very close to the optimum is already reached. In this case the descent step size has been kept fixed and relatively small: with a higher value or, even better, a dynamic step size, the convergence could have been even faster. The error on the volume constrains is always very small and never higher than 0.7%.



(a) Initial shape. (b) Final shape.

Fig. 20 Shape Optimization of a 3D sphere in laminar regime.



(a) Drag. (b) Volume.

Fig. 21 Shape Optimization of a 3D sphere in laminar regime. Cost functional: drag minimization. Constraint: fixed volume. Method: gradient-like. The data in the plot have been non-dimensionalized with the initial (i.e. undeformed) values.

7 Perspectives

In this section we report some perspectives in the numerical models and simulation of sailing boat dynamics and its shape optimization.

Further methodological and code development has to be carried out in order to achieve the targets highlighted in the previous sections. Although a substantial work has already been accomplished, further developments are needed in order to be able to simulate more accurately the full dynamic of a sailing boat. After the successful shape optimization test cases, more realistic geometries and conditions have be to analysed. Below a more detailed description of the future development plan is reported.

7.1 Fluid-Structure Interaction

An accurate and stable coupling technique has to be proposed for more complex and fast-varying dynamics, and an analysis on the interpolation of the data from non-conformal meshes has to be conducted. At the moment the structural and fluid solvers are computed in serial, i.e. one after the other, while a better synchronized strategy, relying on running media at the same time and exchanging information only at given times, may be investigated. Other mesh motion techniques can also be explored, in particular it would be interesting to couple the FFD techniques developed for the shape optimization analyses to this kind of problem. The capability of the FFD technique to keep the mesh valid also for large deformations could in fact be very useful in case of large motion,

like the one of a gennaker under a gust. A coupled local FFD/laplacian may indeed perform very well.

From a programming point of view, the FSI codes available need further development in order to be able to handle realistic flow conditions and geometries. The structural solver also needs to be methodologically developed to take into account the complex constitutive law of modern sails and account for more realistic constraints. Furthermore the mast deformation has to be taken into account as well as real sailing boat sail configurations, i.e. gennaker-main sail-mast.

7.2 Full Sailing Boat Dynamics

Once both the sail FSI and the free surface simulations have been validated, the simulation of the full dynamic of the sailing boat can be finally computed.

To accomplish it, a proper way to couple the FSI solution of the sails with the 6-dof motion of the hull has to be set up, taking into account all the physical constraints, like where are the sails forces applied on the hull, and numerical issue, for example the fact that the boat cannot freely move into the domain, but it has to be kept more or less at the centre. One way to achieve this point would be to fix the longitudinal position of the boat and insert an acceleration term inside the Navier–Stokes like if we were in a non constant-reference system, or we could just keep the average speed fixed and let the boat floats (if it does not drift away otherwise the mesh may easily become invalid).

7.3 Optimization

The performance of shape optimization tools for turbulent flows is of interest and subject of many studies nowadays. Although recently the full adjoint formulation of the Navier–Stokes equations with turbulent models has been proposed [92, 93], the most common approach in literature is to *freeze* the turbulent viscosity obtained by the direct solver and use it in the laminar Navier–Stokes adjoint equations. Special care has to be taken when the direct turbulent model uses wall functions, since this is not seen "naturally" by the laminar adjoint equations and may lead to un-physical results.

Complex geometries may prove critical in terms of smooth deformation and validity of the deformed mesh and thus may require special care in the motion algorithm and smoothing techniques. The possibility to use combined local-FFD have to be developed to further enhance the effectiveness of the methodology. Also the coupling of the proposed approach with Reduced Basis/Reduced models [76] techniques is a subject of current interest. Due to the very heavy computational load entailed by every direct and adjoint computation, the possibility to use reduced models may greatly benefit the effectiveness of the proposed methodology and could allow to tackle even more complex problems. Instead of computing a full RANS simulation for every geometry, an approximated solution could indeed be extrapolated thanks to RB techniques, thus reducing drastically the computational cost.

8 Conclusions

This work represents an overview of state-of-the art numerical models for the simulation and shape optimization of sailing boats combined with the experience about methodological development and numerical simulations accomplished by the authors.

A preview of the complexity of the task of a full simulation of a sailing boat has been highlighted and the main tools to achieve everyday more complex and accurate simulations have been described. Successively, some of the results obtained so far by the authors with open-source codes have been

presented. Promising results in the simulation of free-surface flows and hull dynamics have been achieved and soon more complex configurations will be tackled.

The set up for the simulation of the Fluid-Structure interaction of a sail under steady wind conditions has been established and the results obtained are very promising. The full transient FSI simulation of a gennaker has been successfully computed for different sail structural properties and the shape deformation obtained seems qualitatively very reasonable.

The basic tool for automatic shape optimization has been presented and the combination of classical optimization methods with modern shape parametrization techniques, the FFD parametrization, have been proposed. The FFD control points have been used both as optimization control parameters and/or as shape parametrization space on which to project the sensitivity field. This last approach is innovative and has proven its effectiveness in some initial test cases. More challenging and realistic configurations are foreseen for this topic as well.

Finally, the work here presented has direct applications on many real life cases, not only for sports competitions but also for helping sails/boat manufacturers and naval and ocean engineering groups in general. Furthermore, the flexibility of the fluid code used and the numerical tools under development may be relatively easily modified to be applied to many other fields, like the shape optimization of the geometry of cardiac bypass and stents or the shape and structural composition of aeroplane wings under aerodynamic loads.

The most significant contribution of the methodology is the combination of several techniques to provide a platform for the simulation of complex systems from a geometrical and multi-physics point of view.

Acknowledgements The authors would like to thank the EPFL university and the Alinghi Design Team, for providing the financial support and the technical expertise necessary to accomplish this work.

We are also grateful to Prof. A. Frangi, Dr. M. Cremonesi and Dr. A. Giampieri from the the Structural Engineering Department (DIS) of Politecnico di Milano, for sharing their structural code and the collaboration undergone for the sail simulations.

The financial support for CADMOS and the Blue Gene/P system is provided by the Canton of Geneva, Canton of Vaud, Hans Wilsdorf Foundation, Louis-Jeantet Foundation, University of Geneva, University of Lausanne, and Ecole Polytechnique Fèdèrale de Lausanne.

This work has been partially supported by Regione Lombardia and CILEA Consortium through a LISA Initiative (Laboratory for Interdisciplinary Advanced Simulation) 2010 grant assigned to the second author.

References

1. B. Alessandrini and G. Delhommeau. A fully coupled Navier-Stokes solver for calculation of turbulent incompressible free surface flow past a ship hull. *International Journal for Numerical Methods in Fluids*, 2:pp. 125–142, 1999.
2. N. Alin, R. E. Bensow, C. Fureby, T. Huuva, and U. Svennberg. Current capabilities of DES and LES for submarines at straight course. *Journal of Ship Research*, 54:pp. 184–196, 2010.
3. M. Andreoli, A. Janka, and J.A. Désidéri. Free-form-deformation parametrization for multilevel 3D shape optimization in aerodynamics. *INRIA, Rapp. de rech. no. 5019*, 2003.
4. R. Azcueta. *Computation of turbulent free-surface flows around ships and floating bodies.* PhD thesis, Technical University of Hamburg-Harburg, 2001.
5. R. Azcueta. Computation of turbulent free-surface flows around ships and floating bodies. *Ship Technology Research*, 49, 2002.
6. T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures.* Wiley, 2000.
7. V. Braibant and C. Fleury. Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering*, 44(3):pp. 246–267, 1984.
8. E. F. Campana, D. Peri, Y. Tahara, and F. Stern. Shape optimization in ship hydrodynamics using computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 196:pp. 634–651, 2006.
9. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics.* Scientific Computation. Springer Berlin Heidelberg, 2007.
10. P. Causin, J. F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanical Engineering*, 194:pp. 4506–4527, 2005.

11. D. Chapelle and K-J. Bathe. *The finite element analysis of shells - Foundamentals*. Springer, 2003.
12. P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, and A. Quarteroni. Fluid structure interaction simulations of physiological blood flow in the aorta. Technical report, MATHICSE, EPFL, 2010.
13. S. Deparis. *Numerical Analysis of Axisymmetric Flows and Methods for Fluid-Structure Interaction Arising in Blood Flow Simulation*. PhD thesis, EPFL, 2004.
14. Rhino: 3D design software for CAD, CAE, and CAM designers. Available from: `http://www.rhino3d.com/`.
15. D. Detomi. *Mesh Modifications for Finite Element Methods in Complex Three-Dimensional Domains*. PhD thesis, Politecnico di Milano, 2004.
16. D. Detomi, N. Parolini, and A. Quarteroni. Numerical models and simulations in sailing yacht design. In M. Peters, editor, *Computational Fluid Dynamics for Sport Simulation*, pages 1–31. Springer, 2009.
17. J. Donea, S. Giuliania, and J.P. Halleuxa. An arbitrary Lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33:pp. 689–723, 1982.
18. R. Duvigneau and M. Visonneau. Hybrid genetic algorithms and neural networks for fast CFD-based design. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.
19. E. N. Dvorkin, D. Pantuso, and E. A. Repetto. A formulation of the MITC4 shell element for finite strain elasto-plastic analysis. *Computer Methods in Applied Mechanics and Engineering*, 125:pp. 17–40, 1995.
20. J. Burns Fallow. America's Cup sail design. *Journal of Wind Engineering and Industrial Aerodynamics*, 63:pp. 183–192, 1996.
21. C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163:pp. 231–245, 1998.
22. J. R. Farmer, L. Martinelli, and A. Jameson. A fast multigrid method for solving incompressible hydrodynamic problems with free surfaces. *AIAA Journal*, 32(6):pp. 1175–1182, 1993.
23. J.D. Fenton. A fifth-order stokes theory for steadywaves. *Port, Coastal and Ocean Engineering*, 111:pp. 216–234, 1985.
24. J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, second edition, 1999.
25. LifeV: finite element library. Available from: `http://www.lifev.org/`.
26. AutoCad: 3D Design & Engineering Software for Architecture. Available from: `http://usa.autodesk.com/`.
27. ANSYS CFX CFD Software for Fluid Flow Modeling. Available from: `http://www.ansys.com/products/fluid-dynamics/cfx/`.
28. L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE–FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:pp. 4097–4116, 2004.
29. L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and Simulation of the Circulatory System*. MS&A. Springer, 2009.
30. A. Gara, M. A. Blumrich, D. Chen, G. L. T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. Overview of the BlueGene/L system architecture. *IBM Journal of Research and Development*, 2010.
31. D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
32. J. L. Hess and A. M. O Smith. Calculation of potential flow about arbitrary bodies. *Progress in Aerospace Sciences*, 8:pp. 1–138, 1967.
33. T. Hino. Computation of viscous flows with free surface around an advancing ship. In *Proc. of the 2nd Osaka International Colloquium on Viscous Fluid Dynamics in Ship and Ocean Technology (Osaka)*, 1992.
34. C. W. Hirt and B. D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:pp. 201–225, 1981.
35. T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29(3):pp. 329–349, 1981.
36. S. R. Idelsohn, E. Onate, and C. Sacco. Finite element solution of free-surface ship-wave problems. *International Journal for numerical methods in engineering*, 45:pp. 503–528, 1999.
37. A. Jameson and L. Martinelli. Aerodynamic shape optimization techniques based on control theory. In *Computational Mathematics Driven by Industrial Problems*, pages 151–221. Springer Berlin / Heidelberg, 2000.
38. H. Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows*. PhD thesis, Imperial College (London), 1995.
39. S. G. Johnson. The nlopt nonlinear-optimization package. Available from: `http://ab-initio.mit.edu/nlopt`.
40. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:pp. 359–392, 1998.
41. M.H. Kim, M.S. Celebi, and D.J. Kim. Fully nonlinear interactions of waves with a three-dimensional body in uniform currents. *Applied Ocean Research*, 20:pp. 309–321, 1998.

42. T. Lassila, A. Quarteroni, and G. Rozza. A reduced basis model with parametric coupling for steady fluid-structure interactions. Submitted, 2010.
43. T. Lassila and G. Rozza. Reduced formulation of a steady-fluid structure interaction problem with parametric coupling. In R. A. E. Makinen, K. Valpe, P. Neittaanmaki, and T. Tuovinen, editors, *Proc. of the 10th Finnish Mechanical days*, December 2009.
44. T. Lassila and G. Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Computer Methods in Applied Mechanical Engineering*, 199:pp. 435–465, 2010.
45. Deal II: A Finite Element Differential Equations Analysis Library. Available from: `http://www.dealii.org/`.
46. M. Lombardi. Simulazione numerica della dinamica di uno scafo. Master's thesis, Politecnico di Milano, 2006.
47. A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *submitted*, 2010.
48. L. Martinelli and A. Jameson. An adjoint method for design optimization of ship hulls. In *Proc. of the 9th International Conference on Numerical Ship Hydrodynamics (Ann Arbor, Michigan)*, 2007.
49. B. Le Méhauté. *An introduction to hydrodynamics and water waves.* Springer Verlag., 1976.
50. F. R. Menter. Improved two-equation k-omega turbulence models for aerodynamic flows. *NASA STI/Recon Technical Report N*, 1992.
51. F. R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the SST turbulence model. *Heat and mass transfer*, 4:pp. 625–632, 2003.
52. ANSYS ICEM CFD meshing software. Available from: `http://www.ansys.com/Products/Other+Products/ANSYS+ICEM+CFD`.
53. J. H. Michell. The wave resistance of a ship. *Philosophical Magazine*, 45:pp. 106–123, 1898.
54. B. Mohammadi and O. Pironneau. *Analysis of the K-epsilon turbulence model.* Wiley, 1993.
55. B. Mohammadi and O. Pironneau. Mesh adaption and automatic differentiation in a CAD-free framework for optimal shape design. *International Journal for Numerical Methods in Fluids*, 30:pp. 127–136, 1999.
56. B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids.* Oxford University Press, 2009.
57. J. C. Newman, A. C. Taylor, P. A. Newman, and W. Hou. Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *Journal of Aircraft*, 36(1):pp. 87–96, 1999.
58. F. Nobile. *Numerical approximation of fluid-structure interaction problems with application to haemodynamics.* PhD thesis, EPFL, 2001.
59. OpenFOAM: Open Field Operation and Manipulation. Available from: `http://www.openfoam.com/`.
60. S. Osher and R. Fedkiw. *The level set method and dynamic implicit surfaces.* Springer-Verlag, 2002.
61. S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithm based on hamilton–jacobi formulations. *Journal of Computational Physics*, 79:pp. 12–49, 1988.
62. C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, pages pp. 861–877, 2008.
63. C. Othmer, E. de Villiers, and H. Weller. Implementation of a continuous adjoint for topology optimization of ducted flows. In *18th AIAA Computational Fluid Dynamics Conference (Miami, Florida)*, 2007.
64. N. Parolini. *Computational fluid dynamics for naval engineering problems.* PhD thesis, EPFL, 2004.
65. N. Parolini and A. Quarteroni. Mathematical models and numerical simulations for the America's cup. *Computer Methods in Applied Mechanics and Engineering*, 194:pp. 1001–1026, 2005.
66. S. Piazza. Simulazioni numeriche della dinamica di uno scafo in mare ondoso. Master's thesis, Politecnico di Milano, 2007.
67. L. Piegl and W. Tiller. *The NURBS book.* Springer, 1997.
68. O. Pironneau, F. Hecht, A. Le Hyaric, and J. Morice. Freefem++: Free finite element method. Available from: `http://www.freefem.org/`.
69. M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation, 1994.
70. M. J. D. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, Cambridge, 2009.
71. A. Quaini and A. Quarteroni. A semi-implicit approach for fluid-structure interaction based on an algebraic fractional step method. *Mathematical models and methods in applied sciences*, 17(6):pp. 957–983, 2007.
72. A. Quarteroni. *Numerical Models for Differential Problems.* MS&A. Springer, 2009.
73. H. Renzsch, O. Müller, and K. Graf. Flexsail – a fluid structure interaction program for the investigation of spinnakers. In *21st International HISWA Symposium (Amsterdam)*, 2010.
74. J. F. Rodrigues, J. E. Renaud, and L. T. Watsen. Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. *Structural and Multidisciplinary Optimization*, 15:pp. 141–156, 1998.
75. B. S. Rosen, J. P. Laiosa, W. H. Davis, and D. Stavetski. Splash free-surface code methodology for hydrodynamic design and analysis of iacc yachts. In *Proc. of 11th Chesapeake Sailing Yacht Symposium (Annapolis)*, 1993.
76. G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *rch. Comput. Methods Engrg.*, 15:pp. 229–275, 2008.

77. G. Rozza, T. Lassila, and A. Manzoni. Reduced basis approximation for shape optimization in thermal flows with a parametrized polynomial geometric map. In J. Hesthaven and E. Ronquist, editors, *Selected papers from the ICOSAHOM 09 Conference (NTU Trondheim, Norway), 22-26 June 2009*, volume 76 of *Lecture Notes in Computational Science and Engineering*. Springer,, 2010.
78. H. Rusche. *Computational fluid dynamics of dispersed two-phase flows at high phase fractions*. PhD thesis, Imperial College (London), 2002.
79. P. Sagaut. *Large Eddy Simulation for Incompressible Flows*. Springer, 2009.
80. T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. *Comput. Graph.*, 20(4):151–160, 1986.
81. The COMSOL Multiphysics simulation software. Available from: `http://www.comsol.fr/`.
82. SolidWorks: 3D CAD Design Software. Available from: `http://www.solidworks.com/`.
83. O. Soto and R. Loehner. On the computation of flow sensitivities from boundary integrals, 2004.
84. P. R. Spalart. Detached-eddy simulation. *Annual review of Fluid Mechanics*, 41:pp. 181–202, 2009.
85. Y. Tahara, F. Stern, and Y. Himeno. Computational fluid dynamics-based optimization of a surface combatant. *Journal of Ship Research*, 48:pp. 273–287, 2004.
86. Y. Toda, F. Stern, and J. Longo. Mean-flow measurement in the boundary layer and wake and wave field of a series 60 cb = 0.6 ship model. part 1:froude numbers 0.16 and 0.36. *Journal of Ship Research*, 36(4):pp. 360–377, 1992.
87. D. Trimarchi, S. R. Turnock, D. J. Taunton, and D. Chapelle. The use of shell elements to capture sail wrinkles, and their influence on aerodynamic loads. In *The Second International Conference on Innovation in High Performance Sailing Yachts (Lorient, France)*, 2010.
88. D. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, La Canada, 1998.
89. A. M. Wright, A. R. Claughton, J. Paton, and R. Lewis. Off-wind sail performance prediction and optimisation. Technical report, The Royal Institution of Naval Architects, 2010.
90. J. Yang, T. Michael, S. Bhushan, A. Hanaoka, Z. Wang, and F. Stern. Motion prediction using wall-resolved and wall-modeled approaches on a cartesian grid. In *Proc. of the 28th Symposium on Naval Hydrodynamics (USA, Pasadena)*, 2010.
91. P. J. Zwart, P. G. Godin, J. Penrose, and S. H. Rhee. Simulation of unsteady free-surface flow around a ship hull using a fully coupled multi-phase flow method. *Journal of Marine Science and Technology*, 13:pp. 346–355, 2008.
92. A.S. Zymaris, D.I. Papadimitriou, K.C. Giannakoglou, and C. Othmer. Continuous adjoint approach to the spalart–allmaras turbulence model for incompressible flows. *Computers & Fluids*, pages 1528–1538, 2008.
93. A.S. Zymaris, D.I. Papadimitriou, K.C. Giannakoglou, and C. Othmer. Adjoint wall functions: A new concept for use in aerodynamic shape optimization. *Journal of Computational Physics*, pages 5228–5245, 2010.

MOX Technical Reports, last issues

Dipartimento di Matematica "F. Brioschi",
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

13/2011 Lombardi, M.; Parolini, N.; Quarteroni, A.; Rozza, G.
Numerical simulation of sailing boats: dynamics, FSI, and shape optimization

12/2011 Miglio E.; Cuffaro M.
Tectonic evolution at mid-ocean ridges: geodynamics and numerical modeling

11/2011 Formaggia, L.; Minisini, S.; Zunino, P.
Stent a rilascio di farmaco: una storia di successo per la matematica applicata

Formaggia, L.; Minisini, S.; Zunino, P.
Stent a rilascio di farmaco: una storia di successo per la matematica applicata

10/2011 Zunino, P.; Vesentini, S.; Porpora, A.; Soares, J.S.; Gautieri, A.; Redaelli, A.
Multiscale computational analysis of degradable polymers

09/2011 Pigoli, D.; Sangalli, L.
Wavelets in Functional Data Analysis: estimation of multidimensional curves and their derivatives

08/2011 Ieva, F.; Paganoni, A.M.; Secchi, P.
Mining Administrative Health Databases for epidemiological purposes: a case study on Acute Myocardial Infarctions diagnoses

07/2011 Arioli, G.; Gamba, M.
Automatic computation of Chebyshev polynomials for the study of parameter dependence for hyperbolic systems

06/2011 Secchi, P.; Stamm, A.; Vantini, S.
Large p Small n Data: Inference for the Mean

05/2011 Arioli, G.; Gamba, M.
An algorithm for the study of parameter dependence for hyperbolic systems