OPENFOAM HYDRODYNAMICS FOR YACHT DESIGN: THE CASE OF LUTRA80 SINGULARITY

O.GULINSKY¹, A.KONYNENDYK², A.ROGATKIN², V.VOROBYOV¹

1. MOSCOW INSTITUTE OF PHYSICS AND TECHNOLOGY, RUSSIA,
   E-MAIL ADDRESS: BEDELBAEVA_AIGUL@MAIL.RU
2. LUTRA YACHT DESIGN, NETHERLANDS,
   E-MAIL ADDRESS: INFO@LUTRAYACHTDESIGN.COM

Abstract. The boat concept of Lutra 80 Singularity was to create a true dual-purpose yacht that rewards its owner with racecourse performance and a high level of interior luxury. The hull form and underwater appendages combine with an aggressive sail plan showed impressive results on the racecourse. This task has required a significant amount of tank and tunnel tests. Not all the planned tests could be carried out within a limited time and with a certain constraint on the budget. This report presents the results of a series of CFD experiments based on the OpenFOAM platform. We used the solver interFoam and its extension for dynamic mesh interDyMFoam to study canoe body without appendages.

Figure 1. Lutra 80 Sungularity on a racecourse.
1. **Introduction**

Lutra 80 Singularity was designed by Lutra Yacht Design Group. The boat concept was to create a true dual-purpose yacht that rewards its owner with racecourse performance and a high level of interior luxury. In racing mode, the Singularity concept provides a sophisticated competitive edge.

The hull form and underwater appendages combine with an aggressive sail plan showed impressive results on the racecourse. This task has required a significant amount of tank and tunnel tests. Nevertheless, certainly not all the planned tests could be carried out within a limited time and with a certain constraint on the budget. The purpose of the present project is the following. Using the results of real tank tests we wanted to work out and test reliable and stable solver based on the OpenFOAM platform version 2.1.x. With this in hand we plan to perform a series of numerical experiments based on the Lutra 80 concept. This particular report contains some results of the first part of the project. We used the solver interFoam and its extension for dynamic mesh interDyMFoam to study canoe body without appendages.

![Figure 2. Tank testing.](image)
The main result of hydrodynamic tank tests are the resistance (drag) curves at various heeling angles.

![Drag curve obtained in tank testing.](image)

**Figure 3.** Drag curve obtained in tank testing.

Therefore, the main purpose of our computations at this stage is the calculation of these curves. Besides the problem of simulating a turbulent environment, an important feature of the challenge is the need to work in a two-fluid (air-water) medium. All this makes it necessary to apply most of the tools of the OpenFOAM code.
2. Equations

Our presentation in this section largely follows the J.M. McDonough lectures [2] and the D.C. Wilcox book [1].

2.1. Navier-Stokes equations.

The starting point is the Navier-Stokes equation, which is now believed to embody the physics of all fluid flows, including turbulent ones (in the simple form appropriate for analysis of incompressible flow of a fluid whose transport properties may be assumed constant):

$$\nabla \cdot \mathbf{U} = 0,$$

$$U_t + \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla P + \nu \Delta \mathbf{U} + F_B$$

where $P$ is the reduced, or kinematic (divided by constant density) pressure, and $F_B$ is a general body-force term (also scaled by constant density). The second equation also can be rescaled in terms of the Reynolds number and written as follows:

$$U_t + \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla P + \frac{1}{Re} \Delta \mathbf{U} + \tilde{F}_B,$$

(2.1)

where $\tilde{F}_B$ is a dimensionless body force, often termed the Grashof number in mathematical treatments which is closely related to a Froude number under the present scaling.

It is worth noting here that the Reynolds number

$$Re = \frac{\rho UL}{\mu},$$

where $\rho$ and $\mu$ are, respectively, the fluid density and dynamic viscosity, $U$ is a velocity scale i.e., a typical value of velocity, and $L$ is a typical length scale, is defined in a sense with some degree of uncertainty ("typical"), so it makes sense to talk about Reynolds numbers instead of a single number for the same problem.

Direct numerical simulation (DNS) corresponds to solving the Navier-Stokes equations, using sufficient resolution to capture all physically important scales from the largest to the dissipation scales. It is to be emphasized that no turbulence modeling is employed for DNS and there is no closure problem. But DNS is still not feasible for practical engineering problems as it requires much computational time and moreover it is not enough stable.

2.2. N.-S. in Fourier space. It is convenient to expose the issue of direct calculations in terms of the Fourier representation of the equations and it is enough for this reason to consider the 2-D dimensionless form of N.-S. equations (2.1) in the absence of body forces. Let

$$\mathbf{U} = (u, v)^T, \quad \mathbf{k} = (k_1, k_2)^T,$$

$$u(x, y, t) = \sum_{k} a_k(t) \varphi_k(x, y), \quad v(x, y, t) = \sum_{k} b_k(t) \varphi_k(x, y),$$

$$p(x, y, t) = \sum_{k} c_k(t) \varphi_k(x, y), \quad \varphi_k(x, y) = e^{ik \cdot x} = e^{i(k_1x + k_2y)}.$$
Simple calculations lead us to the following equations for the Fourier coefficients

\[ \dot{a}_k + \sum_{l,m} A_{klm}^{(1)} a_l a_m + \sum_{l,m} B_{klm}^{(1)} a_l b_m = -k_1 c_k - \frac{|k|^2}{Re} a_k, \quad \forall \quad -\infty < k < \infty \tag{2.2} \]

where the (1) superscript denotes the x-momentum equation and \( A_{klm}^{(1)} \) and \( B_{klm}^{(1)} \) are Galerkin triple products. Clearly, an analogous result holds for the y-momentum equation for \( b_k \). We can also represent \( c_k \) in terms of only the \( a_k \) and \( b_k \) which implies that (2.2) and similar equation for \( b_k \) can be expressed in a form that is independent of the pressure.

The Fourier-space representation can be used to deduce qualitative mathematical and physical features of N.S. flows. Neglecting all nonlinear terms in (2.2) with \( c_k \) also eliminated in light of previous remark we get

\[ \dot{a}_k = -\frac{|k|^2}{Re} a_k, \tag{2.3} \]

the solution of which is

\[ a_k(t) = a_k(0) \exp\left(\frac{|k|^2}{Re} t\right). \tag{2.4} \]

Clearly, this solution decays in time and approaches zero as \( t \to \infty \) with the rate \( |k|^2/Re \). In particular, for fixed \( Re \) higher wavenumber Fourier modes decay faster than do lower ones. On the other hand, if \( |k| \) is fixed, then the rate of decay of \( a_k \) decreases with increasing \( Re \). So we can associate the rate of decay of \( a_k \) with physical viscous dissipation, in particular, increasing \( Re \) implies decreasing viscous dissipation.

We will need this observation in the discussion of qualitative mathematical and physical features of the RANS and LES models.

To consider effects of the nonlinear terms we can drop (without lost at the qualitative level) the linear viscous dissipation term from (2.2) and, as in the previous case, we also drop the term containing \( c_k \) :

\[ \dot{a}_k + \sum_{l,m} A_{klm}^{(1)} a_l a_m + \sum_{l,m} B_{klm}^{(1)} a_l b_m = 0. \tag{2.5} \]

And, making an even greater simplification, we keep only the quadratic term \( (l=m) \)

\[ \dot{a}_k = -A_{kll}^{(1)} a_k^2 \tag{2.6} \]

and come to the initial-value problem for the Bernoulli (or Riccati) equation with the solution

\[ a_k(t) = \frac{1}{A^{(1)}_k t + \frac{1}{a_k(0)}}. \tag{2.7} \]

If \( A^{(1)} \) and \( a_k(0) \) are of opposite signs, \( a_k(t) \to \infty \) can occur in finite time. In 2D it is known that this does not actually occur but this possibility can not be eliminated in 3D.

\textit{Anyway, this implies a potential for an ill behavior of N.-S. solutions and so the DNS method.}
2.3. RANS methods.

The idea of the approach is to separate the solutions \( \mathbf{U} = (u, v, w)^T \) of the N.-S. equation on an average "smooth" part \( \mathbf{u} \) and a "disturbance" \( \mathbf{u}' \). Appealing to the ergodic arguments, (that is the ensemble averaging is equivalent to time averaging), which, however, is difficult to justify strictly, this method converts the N.-S. equations to the form

\[
\nabla \cdot \mathbf{u}^2 = -\nabla p + \nu \Delta \mathbf{u} + \mathbf{R}(\mathbf{u}, \mathbf{u}'),
\]

where \( \mathbf{R}(\mathbf{u}, \mathbf{u}') \) is the Reynolds stress tensor. Finally, it is common to ignore the fact that \( \mathbf{u} \) is independent of time and write

\[
\mathbf{u}_t + \nabla \cdot \mathbf{u}^2 = -\nabla p + \nu \Delta \mathbf{u} + \mathbf{R}(\mathbf{u}, \mathbf{u}').
\]

Since \( \mathbf{R}(\mathbf{u}, \mathbf{u}') \) is not determined, a wide range of Reynolds-averaged NavierStokes (RANS) approaches entail extensive modeling. Starting with the Boussinesq hypothesis, the main idea behind determining the Reynolds stress tensor is the introduction of the turbulent eddy viscosity \( \nu_T \) and as a consequence the following form of the equations

\[
\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla \cdot ([\nu + \nu_T] \nabla \mathbf{u}).
\]

We note that the celebrated Boussinesq hypothesis leads to an additional linear diffusive term and the mathematical consequences of this are substantial. The balance of nonlinear advection and linear diffusion is significantly upset by adding a diffusive term, that can be easily seen in the Fourier form representation of the N.S. equations (see (2.2),(2.3), (2.5) and related discussion). So the resulting equations are far more dissipative and certainly it might be bad for physics.

All these lead for the following consequences:

1. RANSs equations have robust (computationally and mathematically) solution.
2. Boussinesq hypothesis leads to an increase of the influence of the diffusive term and hence RANSs solution fundamentally differs from a physically authentic N.-S. solution.

The turbulent eddy viscosity is not a physical property of the fluid and hence \( \nu_T \) is to be determined by turbulence models. We just mention some versions of these methods used in this study. In their basic form \( k - \varepsilon \) RANS models consist of a PDE for each of turbulence kinetic energy, \( k \), and turbulence kinetic energy dissipation rate \( \varepsilon \). Together, these two quantities provide velocity and length scales needed to directly construct eddy viscosity at each point in a computational domain. In this sense the \( k - \varepsilon \) models can be viewed as being "closed" because these models possess sufficient equations for constructing eddy viscosity with no direct appeal to experimental results needed. We also note that \( k \) and \( \varepsilon \) are not the only variables employed in two-equation models. For example, there are also \( k - \omega \) (see section 2.4) and \( k - \omega^2 \) models, among others.

We also note that in all cases the model of turbulence presents equations that are even more complicated than the N.S. equations themselves. Below we collect the complete set (a simple version) of equations of the "standard" \( k - \varepsilon \) RANS model. The system of equations consist of the mean flow continuity and momentum equations, with the latter containing the eddy viscosity, the equations for turbulence kinetic energy and its dissipation rate, and those corresponding to the Boussinesq hypothesis and eddy viscosity in this context. Thus, we have the following:
\[ \nabla \cdot \mathbf{u} = 0, \]
\[ \mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p - \nabla \cdot ((\nu + \nu_T) \nabla \mathbf{u}), \]
\[ k_t + \mathbf{u} \cdot \nabla k = \mathcal{P} - \varepsilon + \nabla \cdot ((\nu + \nu_T/\sigma_k) \nabla k), \]
\[ \varepsilon_t + \mathbf{u} \cdot \nabla \varepsilon = C_{\varepsilon 1} \frac{\varepsilon}{k} \mathcal{P} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} + \nabla \cdot ((\nu + \nu_T/\sigma_\varepsilon) \nabla \varepsilon), \]
\[ \mathcal{P} = (2\nu_T \overline{s}_{ij} - \frac{2}{3} k \delta_{ij}) \frac{\partial u_i}{\partial x_j}, \]
\[ \overline{s}_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \]
\[ \nu_T = C_{\nu} k^2. \]

2.4. Two-Phase Flow problems.

A yacht hull design problem is commonly referred to a hydrodynamic one. However, this can lead to confusion, because in fact it is substantially a two-phase flow problem. Indeed, the wave drag force is essential for conventional ships and not for submarines in a deep water. Consequently, we need to simulate a boat moving in a two-phase medium and, what is most important, an interaction between the air and water phases. On the other hand, the forces acting on hull are not equally affected by this two phases. As the density of air is much less than the density of water, it makes no sense to model an exact air velocity field and a pressure distribution. However, what we do need to determine is the air-water interface precise position and the air-water mixing proportions in a region where such mixing exists.

So our approach is in a sense similar to the system-reservoir model in statistical mechanics, which means that we are not interested in the precise behavior of the air but only its effect on the water phase.

In the conventional volume-of-fluid (VOF) method [3], which is implemented in inter-DyMFOam code, two immiscible fluids are considered as one effective fluid throughout the domain, the physical properties of which (\( \rho_w, \nu_w \) are the density and viscosity of the water and \( \rho_a, \nu_a \), respectively, for air) are calculated as weighted averages
\[ \rho = \rho_w \alpha + (1 - \alpha) \rho_a, \]
\[ \mu = \alpha \rho_w \nu_w + (1 - \alpha) \rho_a \nu_a, \]
\[ \nu = \frac{\mu}{\rho}, \]

based on the distribution of the liquid volume fraction \( \alpha \), thus being equal to the properties of each fluid in their corresponding occupied regions and varying only across the interface
\[ \alpha_t + \nabla \cdot (\mathbf{u} \alpha) + \nabla \cdot [\mathbf{u}_r, \alpha(1 - \alpha)] = 0, \]
\[ \mathbf{u}_r = \frac{\nabla \alpha \cdot \mathbf{u}}{\|\nabla \alpha\|} \mathbf{u}. \]

We note that the transport equation for \( \alpha \) contains an artificial compression term, where \( \mathbf{u}_r \) is the restored vector of relative velocity. The matter is that one of the critical issues
in numerical simulations of free surface flows using the VOF model is the conservation of the phase fraction. Accurate calculation of the phase fraction distribution is crucial for a proper evaluation of surface curvature. It is not a simple task to assure boundedness and conservativeness of the phase fraction but with this compression term the equation remains conservative. And that is not all, since the solution must belong to the interval $(0, 1)$. The restriction is achieved by using a special solver MULES (Multidimensional Universal Limiter for Explicit Solution).

So, by introducing the new variable $\alpha$, two phases are coupled in one set of equations consisting of the mean flow continuity and momentum equations

$$\nabla \cdot \mathbf{u} = 0,$$

$$(\rho \mathbf{u})_t + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \nabla \cdot [(\mu + \mu_t) \nabla \mathbf{u}],$$

along with SST k-omega two-equation turbulence model [4] applied in our study, which is given by the following (written in conservation form):

$$\frac{\partial (\rho k)}{\partial t} + \frac{\partial (\rho u_j k)}{\partial x_j} = P - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j}\right],$$

$$\frac{\partial (\rho \omega)}{\partial t} + \frac{\partial (\rho u_j \omega)}{\partial x_j} = \frac{\gamma}{\nu_t} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1) \frac{\rho \sigma_\omega^2}{\omega} \left[ \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \right] + \frac{2}{3} (1 - F_1) \rho \sigma_\omega^2 \omega \frac{\partial k}{\partial x_j} \omega \frac{\partial \omega}{\partial x_j},$$

$$P = \tau_{ij} \frac{\partial u_i}{\partial x_j},$$

$$\tau_{ij} = \mu_t \left(2 S_{ij} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \rho \sigma_\omega^2 \delta_{ij} \text{ (form of the Boussinesq hypothesis )},$$

$$S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right).$$

(2.11)

The turbulent eddy viscosity is computed from

$$\mu_t = \rho \frac{a_1 k}{\max(a_1 \omega, \Omega F_2)},$$

where $\Omega = \sqrt{W_{ij} W_{ij}}$ is the vorticity magnitude with

$$W_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right).$$
Additional functions are given by

\[ F_1 = \tanh(\arg_1^4), \]

\[ \arg_1 = \min \left( \max \left( \frac{\sqrt{k}}{\beta^{*} \omega_d}, \frac{500 \nu}{d^2 \omega}, \frac{4 \rho \sigma_2 k}{C D_{k \omega} d^2} \right), \right. \]

\[ C D_{k \omega} = \max \left( \frac{2 \rho \sigma_2}{\omega} \frac{1}{\partial k / \partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-20} \right), \]

\[ F_2 = \tanh(\arg_2^2), \]

\[ \arg_2 = \max \left( \frac{\sqrt{k}}{\beta^{*} \omega_d}, \frac{500 \nu}{d^2 \omega} \right), \]

where \( d \) is the distance from the field point to the nearest wall and \( \beta, \sigma, a, \gamma \) are the closure constants.

The turbulent kinematic viscosity is defined by \( \nu_t = \mu_t / \rho \).

2.5. \textbf{LES methods.}

As we have seen above RANS methods are not able to produce solutions to the N.S. equations. This is obviously a very serious shortcoming of any turbulence modeling procedure. In contrast, large-eddy simulation (LES) procedures potentially converge to DNS as discretization step sizes are refined and thus their solutions can be expected to converge to N.-S. solution.

All of LES methods have been constructed via the classical approach to LES consisting of the following steps:

1) Decompose flow variables into large- and small-scale parts, with the large-scale part defined by a filtering process;

2) Filter the governing equations, and substitute the decomposition from part 1) into the nonlinear terms to construct the unclosed terms to be modeled. The idea is to obtain a system of equations for resolved-scale variables that is as close as possible to the N.S. equations;

3) Model the unresolved stresses, i.e. construction of subgrid-scale (SGS) models;

4) Solve equations for the large-scale contribution, while essentially ignoring the small-scale part.

For the velocity vector \( \mathbf{u}(x,t) = (u,v,w)^T \) the filtering process is represented by a spatial filter

\[ \tilde{\mathbf{u}}(x,t) = \int_K G(x, \lambda) \mathbf{u}(\lambda,t) d\lambda, \quad (2.12) \]

where the filter kernel \( G \) is often taken to be a Gaussian, and \( K \) is a subdomain of the solution domain \( \Omega \) such that the volume of \( K \) is approximately \( h^3 \), with \( h \) being the discrete step size of the numerical approximation.

It is useful to identify the decomposition

\[ \mathbf{u}(x,t) = \tilde{\mathbf{u}}(x,t) + \mathbf{u}'(x,t) \quad (2.13) \]
with the Fourier representation
\[ u(x,t) = \sum_{|k|>0} a_k(t)\varphi_k(t) + \sum_{|k|=k_c+1}^{\infty} a_k(t)\varphi_k(t). \quad (2.14) \]

We note that \( \tilde{u}(x,t) \) represents the spatial low-pass filter, supported by the chosen discretization of the governing equations. In this representation it is possible to provide a heuristic argument for convergence of LES solutions to solutions to the N.S. equations: (2.14) shows that as discrete resolution is increased \( (k_c \to \infty) \), the computed result should in principle converge to the true function.

By straightforward applying the filtering procedure, one can deduce the filtered form of the momentum equations:
\[ \tilde{u}_t + \nabla \cdot (\tilde{u}\tilde{u}) = -\nabla\tilde{p} + \nu \Delta \tilde{u}. \quad (2.15) \]

Filtering of the momentum equations leads to significant difficulty only in the nonlinear terms \( \nabla \cdot (\tilde{u}\tilde{u}) \) and the difficulties associated with the nonlinear terms are similar to those arising in the RANS case.

Using the LES decomposition (2.13) for the nonlinear terms
\[ \nabla \cdot (\tilde{u}\tilde{u}) = \nabla \cdot \left( \left( \tilde{u} + u' \right) \left( \tilde{u} + u' \right) \right) \]
one can obtain the SGS stress tensor \( (\tilde{u} + u')(\tilde{u} + u') \).

By analyzing a single component of the tensor
\[ (\tilde{u} + u')(\tilde{v} + v') = \tilde{uv} + \tilde{uv}' + \tilde{vv}' + \tilde{uu}', \]
we note that it consists of the Leonard stress \( \tilde{uv} \), which can be computed directly without any modeling required, the cross stress \( \tilde{uv}' + \tilde{vu}' \), that must be modeled because they contain small-scale factors, and the Reynolds stresses \( u'v' \), which are fundamentally different from the Reynolds stress components arising in RANS formalisms, despite their common name. It is now the usual practice to model SGS stress as a single entity instead of modeling each of the contributions separately.

Introducing the LES subgrid-scale stress \( \tau_{SGS} \) by
\[ \tilde{uu} + [(\tilde{u} + u')(\tilde{u} + u') - \tilde{uu}] = \tilde{uu} - \tau_{SGS} \]
we can rewrite (2.15) in the form
\[ \tilde{u}_t + \nabla \cdot (\tilde{u}\tilde{u}) = -\nabla\tilde{p} + \nu \Delta \tilde{u} - \nabla \cdot \tau_{SGS}. \quad (2.16) \]

This is the equation for the large-scale part of a LES formulation. We observe that only \( \tau_{SGS} \) needs to be modeled, and in that sense (2.16) takes on the same appearance as would a time-dependent RANS method (see (2.10)). But it should be emphasized that presence of the time-derivative term in (2.16) is rigorously correct in contrast to the RANS case. The LES decomposition retains the time dependence in both large and small scales because temporal averaging has been replaced with spatial filtering in the construction process.
Thus, in contrast with RANS models, which are formally required to model everything from the integral scales through the dissipation range, LES models need to represent only the high-wavenumber part of the inertial subrange.

We note that $\nabla \cdot \tau_{SGS}$ is typically modeled in a way reminiscent of the Boussinesq hypothesis, so from a mathematical viewpoint, the arguments given above is nothing but a justification of mechanism for stabilizing of numerical solutions of the equation (2.16) by increasing (numerical) dissipation.

There are many different approaches, ranging from simply solving the governing equations with numerical methods that are strongly dissipative and thus replacing physical dissipation with numerical dissipation, to quite sophisticated procedures which attempt to directly model the subgrid-scale variables (SGS models). We will illustrate all of the above by describing the Smagorinsky model. Because of its simplicity, this oldest of LES SGS models is still widely used and is seen as quite adequate for flows far from solid boundaries, especially if the large scale is well resolved with the cut-off wavenumber lying fairly deep within the inertial subrange.

This model is based on the Boussinesq hypothesis employed extensively in RANS models and takes the form

$$\tau_{SGS} = 2\nu_{SGS} \tilde{S}, \quad (2.17)$$

where $\tilde{S}$ is the large-scale strain-rate tensor, and $\nu_{SGS}$ is the subgrid-scale eddy viscosity. The eddy viscosity is calculated using a formula analogous to the mixing-length formulation of RANS methods:

$$\nu_{SGS} = (C_S \Delta)^2 |\tilde{S}|. \quad (2.18)$$

Here, $\Delta$ is the filter width proportional to grid spacing, and $C_S$ is the Smagorinsky constant, typical values of which are $O(10^{-1})$. This model is implemented in the OpenFOAM code (see code below) with a slight change in relation to the formulas (2.17),(2.18), consisting in the introduction of a regularization parameter $k$ (see below the description).

The comments from Smagorinsky.H file:
Description The Isochoric Smagorinsky Model for incompressible flows.
Algebraic eddy viscosity SGS model founded on the assumption that local equilibrium prevails. Thus,

\begin{verbatim}
  B = 2/3*deltaCoeff * V^1/3 - 2*nuSgs*dev(D)
  Beff = 2/3*deltaCoeff * V^1/3 - 2*nuEff*dev(D)
  where
  D = symm(grad(U))
  k = (2*ck/ce)*delta^2*||D||^2
  nuSgs = ck*sqrt(k)*delta
  nuEff = nuSgs + nu
  delta = deltaCoeff * V^1/3
\end{verbatim}

2.6. Hybrid RANS-LES models.
As we have mentioned, RANS two-equation eddy-viscosity models are highly dissipative. This means that they are not likely to be triggered into unsteady mode unless the flow instabilities are strong, such as vortex shedding behind bluff bodies. At the same time, if
the very largest turbulence structures or some quasi-periodic non-turbulent structure are allowed to be resolved, the flow will be more accurately captured.

The concept of Scale Adaptive Simulation (SAS) allows the simulation of unsteady flows with both RANS and LES content in a single model environment. The functionality of SAS is similar to Detached Eddy Simulation (DES). In broad terms, DES sets out to selectively use either RANS or subgrid-scale (SGS) turbulence modeling in the flow domain in question based on certain criterion. It provides a steady state solution in stable flow regions (like boundary layers) and unsteady structures in unsteady region within a single model framework. In DES, the switch between the RANS and LES is dictated by the ratio of the RANS to the LES turbulent length scales. The latter length scale is defined from the grid (the length of the largest cell side). The difference between DES and SAS is that LES activity in DES is enforced by the grid limiter, whereas SAS allows a breakdown of the large unsteady structure by adapting the turbulence model to the locally resolved length scale.

SST- SAS provides two independent scales to the source terms of the underlying two-equation model. In addition to the standard input in form of the velocity gradient tensor \( \frac{\partial U_i}{\partial x_j} \), SAS models use another scale determined by the second derivative of the velocity field. The resulting von Karman length scale \( L_{vK} \) allows the model to automatically adjust in order to improve the resolution of flow elements. The mechanism of the SAS model, which is responsible for this transition, is the SAS term in the \( \omega \)-equation of SST k-omega model. The SAS term switches itself on when the ratio of the modeled turbulent length scale \( k_1/2/\omega \) to the von Karman length scale increases. The idea of the method is that the SAS term detects the unsteadiness and increases the production of \( \omega \) which in turn leads to a decrease of the turbulent viscosity.

SAS offers a single framework which covers steady state RANS as well as LES regions without an explicit switch in the model formation. All this construction results in the additional term \( Q_{SAS} \) which is simply added to the \( \omega \)-equation of the SST model (2.11):

\[
\frac{\partial (\rho \omega)}{\partial t} + \frac{\partial (\rho u_j \omega)}{\partial x_j} = \frac{\gamma}{\nu_t} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[ (\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + \\
2(1 - F_1) \frac{\rho \sigma_\omega}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} + Q_{SAS},
\]

where

\[
Q_{SAS} = \rho \max \left[ \zeta_2 k S^2 \left( \frac{L}{L_{vK}} \right)^2 - C \frac{2k}{\sigma_\Phi} \max \left( \frac{\|\nabla \omega\|^2}{\omega^2}, \frac{\|\nabla k\|^2}{k^2} \right), 0 \right],
\]

\[
L_{vK} = \frac{kS}{|\Delta u|}, \quad S = \sqrt{2S_{ij} S_{ij}}, \quad S_{ij} = \left( (\nabla u)_{ij} + (\nabla u)_{ji} \right)/2.
\]

The model constant are taken from the \( k - \sqrt{k}L \) model and are given in [5]. As a result of this term, the predicted turbulent length-scale \( L \) is largely proportional to the von Karman length scale \( L \sim L_{vK} \).
3. Numerical implementation

In this section, we show some feature of the numerical implementation of the RANS kOmegaSST method. The results are shown on Fig. 17.

3.1. Computational domain and boundary conditions.

Below we provide a successful combination of boundary conditions for which we have not come immediately:

<table>
<thead>
<tr>
<th>inletWater</th>
<th>inletAir</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>type fixedValue; value uniform (-9 0 0);</td>
</tr>
<tr>
<td>$P$</td>
<td>type zeroGradient;</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>type fixedValue; value uniform 1;</td>
</tr>
<tr>
<td>$k$</td>
<td>type fixedValue; value uniform 0.00015;</td>
</tr>
<tr>
<td>$\omega$</td>
<td>type fixedValue; value uniform 2;</td>
</tr>
<tr>
<td>$\nu_t$</td>
<td>type fixedValue; value uniform 5e-07;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>outlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
</tr>
<tr>
<td>$P$</td>
</tr>
<tr>
<td>$\alpha$</td>
</tr>
<tr>
<td>$k$</td>
</tr>
<tr>
<td>$\omega$</td>
</tr>
<tr>
<td>$\nu_t$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sides</th>
<th>hull</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>type fixedValue; value uniform (-9 0 0);</td>
</tr>
<tr>
<td>$P$</td>
<td>type zeroGradient;</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>type zeroGradient;</td>
</tr>
<tr>
<td>$k$</td>
<td>type symmetryPlane;</td>
</tr>
<tr>
<td>$\omega$</td>
<td>type symmetryPlane;</td>
</tr>
<tr>
<td>$\nu_t$</td>
<td>type symmetryPlane;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>atmosphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
</tr>
<tr>
<td>$P$</td>
</tr>
<tr>
<td>$\alpha$</td>
</tr>
<tr>
<td>$k$</td>
</tr>
<tr>
<td>$\omega$</td>
</tr>
<tr>
<td>$\nu_t$</td>
</tr>
</tbody>
</table>

Figure 4. Virtual tank scheme.
It should be emphasized that the boundary conditions for \( \alpha \) affect greatly the stability of the calculation.

They determine the mass flow of the fluid and are chosen to provide the mass conservation of the water in the virtual tank. The static zeroGradient type conditions on inlet do not work in this case because they cause very undesirable effects, such as "draining" of the virtual tank or its overflow. With zeroGradient type boundary condition, \( \alpha \) on the boundary could be equated to the value in the adjacent cells which in turn can cause changes in the water level if there are (computational or physical nature) surface waves.

In a successful combination at the moment \( \alpha \) is equal 1 below the water level and \( \alpha = 0 \) in the air.

\( p \) and \( U \) boundary conditions are typical for the free stream problems: at the inlet \( U \) is fixedValue type, \( p \) is zeroGradient type, and at the outlet \( U \) is zeroGradient and \( p = 0 \).

3.2. Meshes.

The mesh spatial step size is an important parameter for the accuracy of numerical approximation. It is clear that one should decrease the average cell size to achieve an acceptable accuracy of the approximation.

It should be noted that we prefer to talk about the accuracy of approximation instead of convergence for the following reasons.

First of all, because of the additional linear diffusive terms, solutions of RANS equations can not in principle converge to the solution of N - S equation. We also note that stability of the computational process deteriorates with decreasing mesh size, making it necessary to use the regularization parameters, which in turn coarsening solution. On the other hand, some parameters which are commonly used for the analysis of the tank testing results (such as drag force) do not depend noticeably on precise wave shapes.

Regarding fidelity of turbulent picture we note that one of a useful way of checking the accuracy is the comparison of wave turbulent patterns. A physically real wave pattern can give a guess that both mathematical model and its numerical approximation behave well at least at a large scale. However, in order to obtain an accurate wave shape one has to construct a really fine mesh, which still produces a stable picture but requires unacceptable computational time. Then after verification of the procedure on a fine mesh one can reduce the mesh to a coarser one to obtain necessary results in a reasonable time.
In our simulations we studied two types of meshes: coarse and fine (90k cells, 530k cells respectively).

Figure 5. Fine mesh slice.

Next figure shows the first layer for the coarse (upper picture) and fine mesh in dimensionless quantity $y^+$ scale.

Figure 6. Fine (top) and coarse (bottom) mesh and $y^+$. 

15
The estimations indicate that the value $u^+$ is about 26-37 and so in both cases the first layer of grids are in the outer (a fully-turbulent) layer. Clearly, for a detailed prediction of turbulence in the near the wall region it is desirable that the first layer of grid is in the viscous sub-layer. However, this would require non-comparable costs in time, that did not make sense at this stage.

**Figure 7.** The law of the wall.

The Fig. 8 shows the wave pattern obtained by using two different grids. It shows that the coarser mesh reproduces only the main feature of wave pattern whereas the fine mesh produces a much more rich picture.

**Figure 8.** Fine (top) and coarse (bottom) mesh and wave pattern.
At the same time, as seen in Fig. 9, 10, 11, 12, the main parameters (center of mass position and drag force) differ insignificantly with mesh refinement.

![Figure 9. Center of mass. V = 18 knots, fine mesh.](image1)

![Figure 10. Center of mass. V = 18 knots, coarse mesh.](image2)

![Figure 11. Drag force. V = 18 knots, fine.](image3)

![Figure 12. Drag force. V = 18 knots, coarse.](image4)

We think that the reason is that these two grids are in the same region of turbulence. This fact allows us to make a fairly accurate construction of the resistance curve with a coarse grid (see Fig. 17).
3.3. **Dynamic mesh.**

Dynamic mesh is crucial to determine the exact position of the boat. In particular, the dynamic mesh allows to analyze trim and heel boat balances and to optimize mass distribution and weight. Using the static mesh and interFoam solver can lead to significant errors. Especially these errors affect the study of transition processes from submerged regime to planning.

We use the Laplacian mesh solver with diffusion coefficient proportional to inverse distance to the hull. Mesh on all surfaces is static, and all mesh changes are allowed only in the volume of the computational domain.

3.4. **Regularization parameters.**

Special attention was paid to the study of the regularization parameters contained in the package. An example of the effect of these parameters are shown on Fig. 2, 3. aLim and cDamp are constants in sixDoFRigidBodyMotion.C code which correspond to accelerationLimit and accelerationDampingCoeff and can be initialized in the pointDisplacement file. The accelerationLimit parameter is aimed to cut the acceleration when it comes greater than specified value. The accelerationDampingCoeff scales acceleration and angular acceleration by \((1 - cDamp)\).

![Figure 13. V = 24 knots, aLim = 1, cDamp = 0.9](image13.png)

**Figure 13.** \(V = 24\) knots, \(aLim = 1, cDamp = 0.9\)

![Figure 14. V = 24 knots, aLim = 1, cDamp = 0.7](image14.png)

**Figure 14.** \(V = 24\) knots, \(aLim = 1, cDamp = 0.7\)
3.5. **Restraints and Constraints.**

During the actual tests, and therefore also in a virtual tank the orientation vector \( z \)-coordinate (trim) and the center of mass \( z \)-coordinate are the basic monitored parameters. Remaining degrees of freedom must be fixed to avoid oscillations and "drift" in \( x, y \) coordinates. The solver sixDoFRigidBodyMotion, which provides movement of an object at test, contains two predetermined instruments *Restraints* (strings) and *Constrains*.

If the initial position of the boat is set far from unknown in advance equilibrium, the transition period can destroy the computational process. In this case we applied a string to smooth the process and then used the obtained result to improve the initial position for the next iteration. It is impossible to use strings for constraining degrees of freedom since strings use to change the equilibrium. For these purposes, the code includes *Constrains*. We only allowed movement along \( z \)-axis and rotation around \( y \)-axis. To apply these features one should set two main parameters, tolerance and relaxation, which control the process of holding the body in the borders of constraints. In the spirit of regularization procedures, to ensure a smoothness of the process, *Constrains* do not fix the position of the coordinates completely but set some small values.

![Figure 15. V = 18 knots, center of mass Y coordinate, tolerance = 1](image1.png)

**Figure 15.** \( V = 18 \) knots, center of mass \( Y \) coordinate, tolerance = 1

![Figure 16. V = 18 knots, center of mass Y coordinate, tolerance = 10^{-6}](image2.png)

**Figure 16.** \( V = 18 \) knots, center of mass \( Y \) coordinate, tolerance = 10\(^{-6}\)

In our calculations constraints parameters were the following:

```java
fixedLine1
{
  tolerance 1;
  relaxationFactor 1;
  fixedLineCoeffs
  {
    refPoint (-12.8 0 -0.5);
    direction (0 0 1);
  }
}

fixedAxis1
{
  tolerance 1;
  relaxationFactor 1;
  fixedAxisCoeffs
  {
    axis ( 0 1 0 );
  }
}
```
3.6. **Time step setting.**

Another important tool affecting the stability of the calculation is the automatic time step setting in interDyMFoam which works as follows. Before computing the next approximation of fields on a new time step, the Courant numbers $Co$ in each cell is calculated

$$Co = \frac{u\Delta t}{\Delta x},$$

where $\Delta t$ is the current time step and $\Delta x$ is the cell size. After that a new time step $\Delta t$ is reduced if maximum $Co$ over all cells is greater than a predetermined value MaxCo and it is increased otherwise.

Time step $\Delta t$ is controlled by this procedure in such a way that $Co$ is kept close to MaxCo. When dealing with the virtual tank testing one faces a complication. We noted that air cells have a higher Courant number than water cells. This effect appears particularly near the hull where cells must be sufficiently accurate in order to resolve a low water speed in this area. At the same time, water and air have essentially different scales of ”near-hull” layers. So, following this procedure with $Co$ which is much higher in air than in water, we have to set $\Delta t$ much smaller than it is necessary to simulate water fraction. This leads to an increase in computational time and additional numerical errors. For these reasons, in our experiments we modified the interDyMFoam code in order to take into account only Courant numbers in the water cells for the control of the time step calculations. Due to this modification Courant number maximum is taken only over cells with $\alpha > \text{minPhase1Co}$, where minPhase1Co is a new constant to be determined. Experiments showed that minPhase1Co should be greater than 0.8 for time step stability.

3.7. **Some failures and confusions.**

1. It should be noted that in the current code the time step depends on the frequency of recording the results to disk. This setting is somewhat dangerous, mainly because it is not represented in the setDeltaT file and can be considered as a ”side-effect” for not so advanced users. One can face a problem when successful up to this calculations cease to work for some strange reason despite the fact that all the settings have not been changed. At the same time we note that this situation can be controlled by setting runTimeModifiable to ”no” in the controlDict file.

2. Here we describe another case when the automatic time step setting must be used carefully. If the processes under study are of a scale smaller than this mechanism sets then it can not be tracked. In our particular case the boat did not come out of the transition (computing) process at speeds of less than 3 meters per second.

In the dynamic mesh calculations problems may arise if as a result of the initial computational errors an object is moving faster than the grid. In our case the boat used to get too much speed at the initial stage and calculation collapsed. We have overcome this difficulty by ”manual” time step reduction at the initial stage. It should be noted that, as we later found out in (and only in) interDyMFoam, it is possible to control the time step with the help of the parameter meshCourantNumber (the mesh Courant number).
4. Results

First, using the RANS kOmegaSST method, we did detailed calculations required for a precise plotting the drag curve (see Fig. 17). Reports supplying the necessary information about these calculations are contained in Appendix. This method was chosen because it provides a stable solution at different flow speeds due to the strong influence of the numerical turbulent eddy viscosity.

![Figure 17. Drag curve by RANS kOmegaSST.](image)

At the same time, based on the preceding analysis, it can be assumed that the results obtained by the RANS kOmegaSST method reflect a strong dissipative nature of the method. These properties, as can be foreseen, should manifest itself especially at high speeds, when shown a more complex picture of turbulence. Therefore, given the ability of the LES and Hybrid methods to restore, in principle, a more complex picture of turbulence, we applied some of this methods (the Smagorinsky model, the oneEqEddy model, the kOmegaSST-SAS model) to calculate several key points on the curve. We tried these models on grids with different refinement degrees and with different ways of determining the delta - a parameter that specifies the minimum vortices scale, which LES model allows for.

It should be noted that the wave pattern, generated by these methods at 18kt speed, at first sight, differs slightly (see Fig. 18), if at all different, despite the fact that the pictures correspond to an accurate grid cases.
Figure 18. Wave pattern generated by different methods.
However, the situation differs considerably in plotting the resistance curves, calculated by different methods. Fig. 19 shows the results for the drag force, generated by different methods in the most interesting area of speeds 14-22 kt. Taking in account that the boat comes to a stable planning in the area 18-22 kt, it is important to make sure that the turbulence models correctly handle this regime and predict a correct position of the boat on the water.

**Figure 19. Drag forces with different methods**

In the area 14 - 18 knots of speed the oneEqEddy and the kOmegaSSTSAS indicate a similar results - 2% difference from each other - on each of the two tested grids. The Smagorinsky model stands out and gives a 5% less drag on the coarse grid, which we associate with the simplicity (roughness) of the model. With a more accurate grid the Smagorinsky model collapsed.

As it turned out, the method "cubeRootVol" to determine the parameter delta in OpenFOAM LES models - the cube root of the cell volume - is more reliable. The second - the "smooth" method - consists in smoothing the field in such a way that ratio of values of delta in adjacent cells is not more than 1.15. Only the kOmegaSSTSAS model was able to fulfill calculations with the "smooth" method and gave about 3% higher drag than in the "cubeRootVol" case. We believe that the problems with the use of smoothing are related to our configuration of the grid. Even if smoothing provides some advantages, using it one has to be very careful with the construction of the grid.
The most significant difference between the RAN method and LES-Hybrid method was showed, as expected, at speeds of 18-22. We did calculations with

1. kOmegaSST-SAS model, "smooth" parameter, coarse mesh,
2. kOmegaSST-SAS model, "cubeRootVol" parameter, coarse mesh,
3. oneEqEddy model, "cubeRootVol" parameter, coarse mesh.

In all three cases the values of the drag force exceed by 9-11% the value given by the kOmegaSST model. We believe that the LES and hybrid models give in this area more reliable result.

The difference between the results generated by different LES models on the same grid was less than the difference between the results generated by the same models with different grids. We obtained 7% less drag force in average applying the finer grid. It can be assumed that this effect is due to the refinement of the boundary layer only, rather than the mesh refinement in the volume of the computational domain. To test this hypothesis we used (stable) RANS kOmegaSST model and constructed a grid that coincides in the volume with the coarser mesh, but with a thin boundary layer (see Fig. 20).

![Fine sublayer mesh y+](image)

**Figure 20.** Fine sublayer mesh $y^+$.  

The resulting full drag indeed substantially and significantly changed (see on Fig. 19 the RANS kOmegaSSTfinesublayer red point at 18kt speed) in the direction of decreasing. However, we can not claim that it is closer to the true solution of the Navier-Stokes result.

**References**

5. Appendix

Velocity 10 knots.

Time
\[
\text{deltaT 0.002;}
\]
\[
\text{minPhase1Co 0.9;}
\]
\[
\text{maxCo 0.4;}
\]
\[
\text{maxAlphaCo 10;}
\]
\[
\text{maxDeltaT 0.5;}
\]

PIMPLE
\[
\text{momentumPredictor no;}
\]
\[
\text{nCorrectors 2;}
\]
\[
\text{nNonOrthogonalCorrectors 0;}
\]
\[
\text{nAlphaCorr 1;}
\]
\[
\text{nAlphaSubCycles 2;}
\]
\[
\text{cAlpha 1;}
\]
\[
\text{correctPhi yes;}
\]

Floating object
\[
\text{mass 23800;}
\]
\[
\text{momentOfInertia (100000 1250000 1250000);}
\]
\[
\text{accelerationLimit 1;}
\]
\[
\text{accelerationDampingCoeff 0.7;}
\]

Turbulence
\[
\text{simulationType RASModel;}
\]
\[
\text{RASModel kOmegaSST;}
\]

Mesh
\[
\text{Points: 101714}
\]
\[
\text{Cells: 90611}
\]
\[
\text{Faces: 268697}
\]

\[\text{Figure 21. Total drag force vs time. Velocity 10 knots.}\]
Figure 22. ΔT vs time. Velocity 10 knots.

Figure 23. Orientation vs time. Velocity 10 knots.

Figure 24. Center of mass vs time. Velocity 10 knots.

Figure 25. Acceleration vs time. Velocity 10 knots.
Velocity 18 knots.

**Time**
deltaT 0.002;
minPhase1Co 0.9;
maxCo 0.4;
maxAlphaCo 10;
maxDeltaT 0.5;

**PIMPLE**
momentumPredictor no;
nCorrectors 2;
nNonOrthogonalCorrectors 0;
nAlphaCorr 1;
nAlphaSubCycles 2;
cAlpha 1;
correctPhi yes;

**Floating object**
mass 23800;
momentOfInertia (100000 1250000 1250000);
accelerationLimit 1;
accelerationDampingCoeff 0.7;

**Turbulence**
simulationType RASModel;
RASModel kOmegaSST;

**Mesh**
Points: 101714
Cells: 90611
Faces: 268697

Figure 26. Total drag force vs time. Velocity 18 knots.
Figure 27. $\delta T$ vs time. Velocity 18 knots.

Figure 28. Orientation vs time. Velocity 18 knots.

Figure 29. Center of mass vs time. Velocity 18 knots.

Figure 30. Acceleration vs time. Velocity 18 knots.
Velocity 24 knots.

**Time**
deltaT 0.002;
minPhase1Co 0.9;
maxCo 0.4;
maxAlphaCo 10;
maxDeltaT 0.5;

**PIMPLE**
momentumPredictor no;
nCorrectors 2;
nNonOrthogonalCorrectors 0;
nAlphaCorr 1;
nAlphaSubCycles 2;
cAlpha 1;
correctPhi yes;

**Floating object**
mass 23800;
momentOfInertia (100000 1250000 1250000);
accelerationLimit 1;
accelerationDampingCoeff 0.9;

**Turbulence**
simulationType RASModel;
RASModel kOmegaSST;

**Mesh**
Points: 101714
Cells: 90611
Faces: 268697

**Figure 31.** Total drag force vs time. Velocity 24 knots.
Figure 32. deltaT vs time. Velocity 24 knots.

Figure 33. Orientation vs time. Velocity 24 knots.

Figure 34. Center of mass vs time. Velocity 24 knots.

Figure 35. Acceleration vs time. Velocity 24 knots.