**DISCLAIMER FOR FRONT PAGE OF MATERIALS TO BE MADE AVAILABLE VIA ETI INTERNET SITE**

1. "Save to the extent set out in paragraph 2 below, this document and its contents are made available to you via the ETI's Internet Site "as is" without any representations, conditions, warranties or other assurance of any kind. The ETI and the authors, together with their employees, directors, servants or agents exclude to the maximum extent permissible by law all representations, warranties, conditions or other assurance whatsoever (whether express or implied) regarding the use of this document or its content including any warranties of title, merchantability, accuracy, completeness, non-infringement or that the document or its contents are of satisfactory or any particular quality or fit for any particular purpose. Any person accessing this document and using it or any of its contents accepts all risk in doing so.
2. Notwithstanding any statement to the contrary contained on the face of this document, the ETI confirms that the authors of the document have consented to its publication by the ETI."

| | |
|---|---|
| Project | ReDAPT |
| Deliverable | MD1.2 |
| Authors | DD Apsley, PK Stansby, T Stallard, I Afgan, J McNaughton |
| Circulation | University of Manchester, EdF |
| To be approved by | Prof. Dominique Laurence |
| Date | 21 September 2011 |
| Version | 2 (revision 5) |

## EXECUTIVE SUMMARY

This document is the second in a series describing progress with the application of EdF's open-source CFD solver *Code_Saturne* toward a full marine-current-turbine simulation, including turbulence and waves, as part of the wider ReDAPT project.

In order to compute flow about a moving rotor in an otherwise stationary domain it has been necessary to develop our own sliding-interface method from scratch. A detailed description of this interfacing method is given in this report, together with validation studies for its implementation in *Code_Saturne* and an in-house research code *STREAM* and guidelines (for example, maximum timestep and inner-loop convergence criteria) for its use.

An idealised geometry has been produced, based on the turbine used for flume experiments by Prof. Bahaj's group at the University of Southampton. The model geometry includes both rotor and support structure with computational meshes of between 2 and 4 million nodes. A description of this model is given and details of the blockage corrections necessary for simulating open-water conditions are included for reference.

Reynolds-averaged-Navier-Stokes (RANS) calculations with the standard $k$-$\varepsilon$ turbulence model and large-eddy-simulation (LES) calculations with the Smagorinsky model and a basic synthetic eddy model (SEM) for inflow turbulence have been undertaken and are reported here. Calculations were performed on EdF's Blue Gene computer using 2048 processors and the University of Manchester's Computational Shared Facility (CSF) with 256 cores.

A parametric study of the effects of input turbulence and operating point (tip-speed ratio, TSR) has been undertaken. Both RANS and LES calculations have been performed with turbulence levels of 0% (nominal), 10% and 20%, with TSR values of 4, 6 (design) and 8. RANS calculations gave good results for mean thrust coefficient, but low values for power coefficient and tended to diffuse vortex structures in the wake. LES calculations gave slightly better predictions of power and more information about the wake vortex structures and the response of the fluctuating loads to a change in inflow turbulence levels.

**CONTENTS**

# 1. INTRODUCTION

## *1.1 Scope of this Document*

This report explains the work undertaken to complete milestone MD1.2; specifically:

- Description of a new sliding-mesh interface method and its implementation.
- 3-d simulation of a rotor using the sliding-mesh method (with imposed rotation):
  – including blade set, nacelle and support mast;
  – based on the geometry of Bahaj et al., (2007a,b);
  – RANS turbulence modelling;
  – LES turbulence modelling.

## *1.2 Specific Tasks Associated With This Project*

The specific milestones for the CFD work on the ReDAPT project are as follows. Major items in this report are underlined.

| | |
|---|---|
| MD1.1 | Ideal turbine geometry.<br>Imposed rotation of a single cylindrical mesh (Coriolis forces or ALE).<br>RANS turbulence.<br>No waves.<br>Report to identify development necessary for sliding mesh. |
| **MD1.2 (This Report)** | **Ideal turbine geometry.**<br>**Rotation via sliding mesh (including a description of the method).**<br>**RANS and LES turbulence.**<br>**No waves.** |
| MD1.3 | Real turbine geometry.<br>Sliding mesh.<br>RANS turbulence.<br>Waves. |
| MD1.4 | Real turbine geometry.<br>Sliding mesh.<br>LES turbulence.<br>Waves.<br>Comparison of loads, velocity and near-wake turbulence with field data. |

## *1.3 Staff on the Project*

The research staff employed on this project are a PhD student (James McNaughton) and a post-doctoral research associate (Dr Imran Afgan).

James McNaughton started in September 2009 and has PhD supervisors Dr David Apsley and Dr Alistair Revell. Overall responsibility for the University of Manchester's contribution to ReDAPT rests with Prof. Peter Stansby. Other Manchester or ex-Manchester staff who have contributed informal assistance in the project include Dr Juan Uribe (*Code_Saturne* expertise), Dr Tim Stallard (related PerAWAT project), Olivier Cozzi (moving-mesh/free-surface methodology), Yacine Addad (preceded Dr Afgan as RA on this project).

## 2. SLIDING-MESH INTERFACE METHOD

### *2.1 Motivation*

In order to simulate a rotating marine-current turbine in the vicinity of a solid lower boundary and wave-affected free surface the basic mesh regions will be as shown in Figure 2.1. These comprise an inner cylindrical block that rotates with the turbine blades and an outer block that includes the stationary support structure.

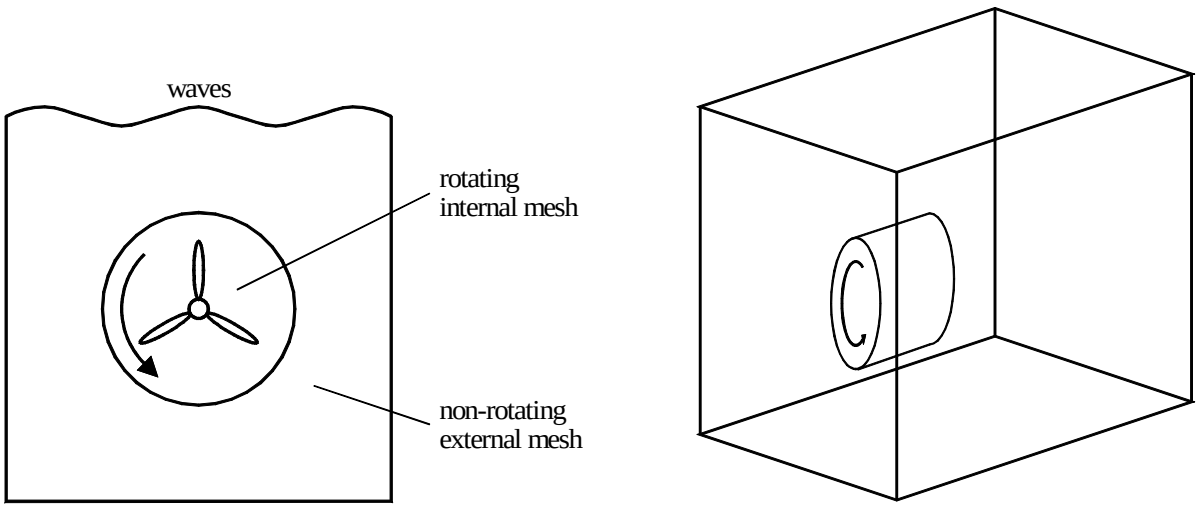



**Figure 2.1:** mesh regions with a sliding interface.

The arbitrary Lagrangian-Eulerian (ALE) method is used to handle the mesh movement within the rotating block. If the free surface is subject to wave motion then the ALE method can also be used for the vertical stretching of the upper part of the outer block; otherwise the outer mesh is stationary.

Between the rotating cylindrical part and the outer mesh there are non-conforming and sliding interfaces (on front, back and curved surfaces of the cylinder). We have developed our own generic method for handling sliding interfaces. This is described in Sections 2.2–2.4 below.

Note that sliding interfaces are not a new CFD challenge. For understandable commercial reasons the procedures implemented in commercial codes like Fluent and StarCCM+ are not fully documented in the open literature. However, some documentation is available for the interfacing methods employed in other open-source and university research codes. OpenFOAM's general grid interface (GGI, see Beaudoin and Jasak, 2008) divides adjacent faces into weighting factors and divides the fluxes into neighbouring cells. OpenFOAM also has a patching method (Petit et al., 2009) where meshes on either side of the interface are joined together at each time-step. The two methods are quite similar with the second being more conservative by formulation, albeit more computationally expensive. The method used by Blades and Marcum (2007) uses halo cells on either side of the interface which are projected a distance based on the host and neighbouring meshes. The halo-cell method is quite common and is also used by Steijl and Barakos (2008) and Fenwick and Allen (2006).

Our own interface method also uses halo (elsewhere, "ghost") data, but the method is slightly simpler in that only halo nodes (rather than whole cells) are constructed and they are used only to assign values on cell faces abutting the interface. Because we have used it in development and testing we will describe the implementation of the interface routines in both our own multi-block-structured-mesh code *STREAM* (Lien et al., 1996) and EdF's unstructured-mesh *Code_Saturne* (Archambeau, 2004).

### *2.2 General Outline of Method*

A general interface is depicted in Figure 2.2 below. In this case two regions of cells, here designated L and R, meet at a non-conforming (and generally sliding and curved) interface.

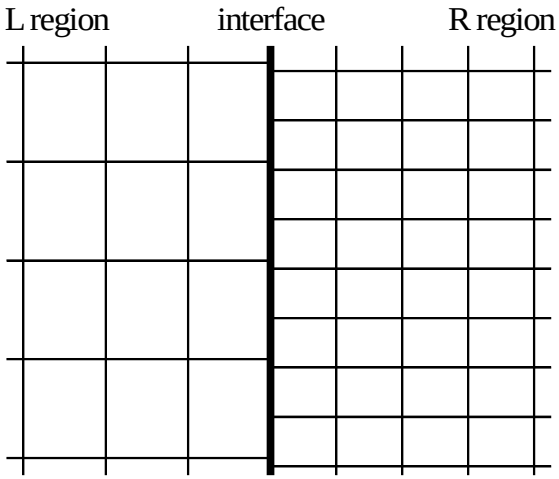



**Figure 2.2:** general non-conforming interface.

The basic idea of the method is to treat the interface as an *internal Dirichlet boundary* for the individual region on each side. Linkage between the two regions is achieved using the values of variables on the boundary cell faces. The process for finding these cell-face-centre values is illustrated in Figure 2.3.

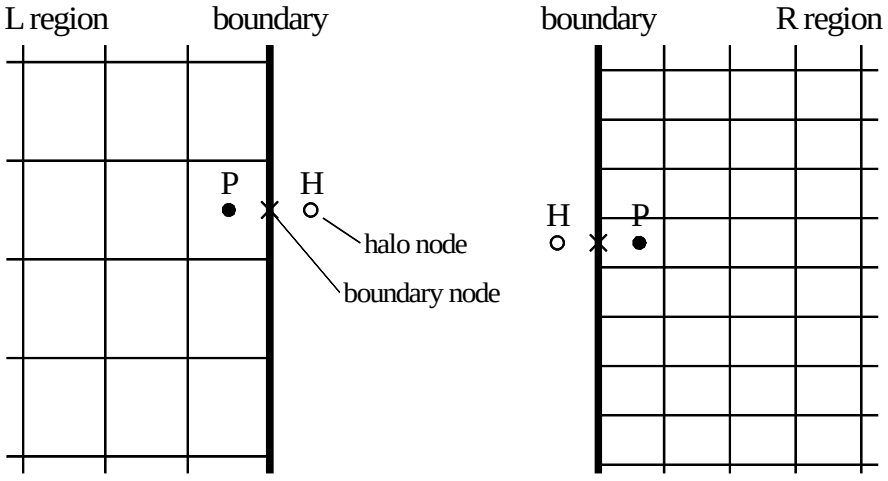



**Figure 2.3:** Location of boundary and halo nodes.

(1) For each cell abutting the boundary locate a *halo node* in the opposite region by extrapolation:

$$\boldsymbol{x}_H = \boldsymbol{x}_B + (\boldsymbol{x}_B - \boldsymbol{x}_P)$$

Here, P is the node at cell centre, B is the cell-face centre on the boundary and H is the halo node.

(2) For each variable $\phi$, interpolate (the method varies between codes: see Sections 2.3 and 2.4) from values in the opposite region to find the value at the halo node.

(3) The boundary value is then recovered by simple linear interpolation:

$$\phi_B = \frac{1}{2}(\phi_P + \phi_H)$$

The following should be noted.

- Where interpolation to produce a value at the halo node implicitly involves a contribution from the boundary face value, the updating of that boundary value is necessarily iterative.
- For the method to work properly the halo node should ideally be located in the first line of cells abutting the interface. This requires the cells immediately on either side of the interface to be of roughly comparable depth perpendicular to the interface. In most cases this is automatically done by the grid generator.
- The mesh motion in one timestep should not be so large that the halo nodes "skip" cells in the neighbouring region. Numerical experiment suggests that this limit should be even more conservative; we prefer to ensure that the interface slides by no more than half a cell in one timestep.
- Linear interpolation means that the method is formally second-order accurate in space.
- Because the interface passes values of $\phi$ and not flux($\phi$) the method is not guaranteed to be conservative. However, numerical experiment confirms that global mass-flux errors are less than 0.01%.

The method of producing a value of a variable at a halo node is code-specific and is described below for two CFD codes: our own in-house multi-block structured code *STREAM* and EdF's unstructured *Code_Saturne*. Specifically:

- *STREAM* uses a "which-cell-am-I-in?/linearly-interpolate" approach to find values at the halo node. This is straightforward and very efficient but is restricted to structured meshes.
- *Code_Saturne* uses a "which-is-the-closest-node?/Taylor-series-expansion" approach to find values at the halo node.

### *2.3 Implementation in STREAM*

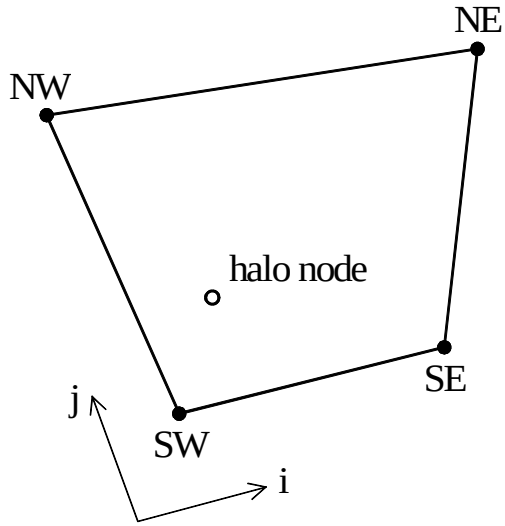



**Figure 2.4:** structured mesh – location within a box (illustrated here for 2-d meshes).

Given the location of a halo node, $\boldsymbol{x}_G$, the interpolation routine determines which "box", i.e. octet (3-d) or quartet (2-d) of nodes (see Figure 2.4), this point lies within and the fractional distances $f_i$, $f_j$, $f_k$ from "west", "south" and "bottom" faces of this box, respectively. These fractional distances are computed from the projected displacements of the halo node onto the associated faces of the box. (Note that these interpolation boxes have the nodes as *vertices* and are not the same as the computational finite volumes, which have nodes at their centres.) A point is formally within a particular box if

$$0 \le f_i \le 1, \qquad 0 \le f_j \le 1, \qquad 0 \le f_k \le 1$$

The search routine to find which box a halo node lies within is optimised by:

- only searching in designated blocks on either side of the interface;
- for each test box doing an initial quick check to confirm that the point lies within the range of *x*, *y*, *z* values at its corners;
- cycling through the boxes starting at the last used by this halo node; if the halo node hasn't moved very far then the relevant box will be found immediately.

An advantage over the "nearest-node" approach is that once the appropriate box has been found then searching can stop, whereas for "nearest-node" searches one must obviously continue testing all possibilities. The major restriction is that the mesh must be (multi-block) structured.

Multi-linear interpolation is then used to find the value of each variable $\phi$ at the halo node; e.g. in 2-d (see Figure 2.4 for node arrangements):

$$\phi = (1 - f_j)[(1 - f_i)\phi_{SW} + f_i\phi_{SE}] + f_j[(1 - f_i)\phi_{NW} + f_i\phi_{NE}]$$

The extension to 3 dimensions is obvious. The simple multilinear interpolation also ensures that as the halo node moves from one box to its neighbour then $\phi$ changes continuously.

Some simple 2-d (rotating cylinder) and 3-d (rotating cube) examples are illustrated below. The latter shares essentially the same mesh-interface topology as the marine current turbine, but with a considerably simpler rotating body.

Figures 2.5, 2.6 and 2.7 show simulations of the laminar flow about a 2-d rotating cylinder with Reynolds number $\mathrm{Re} \equiv U_0 D / \nu = 200$ and non-dimensional rotation rate $\alpha \equiv \Omega R / U_0 = 0.5$ ($D$ is diameter, $R$ is radius and $U_0$ is the approach-flow velocity). Detailed computations have been performed for a variety of rotation rates at this Reynolds number by Stansby and Rainey (2001) and Mittal and Kumar (2003). The latter made a comprehensive study of flow behaviour for $0 < \alpha < 5$, showing two-sided vortex shedding to exist for $\alpha < 1.91$, with very high lift coefficients for the larger values of $\alpha$ by the Magnus effect. A detailed grid- and timestep-dependence study has not been carried out here, the main purpose being to investigate the efficacy of the sliding-interface approach; however, results for fluctuating lift at this rotation rate are consistent with those of both journal papers.

Figure 2.5 shows the two-part mesh: an inner fine mesh which rotates with the cylinder and an outer coarser mesh that is stationary. The two are separated by a sliding interface.

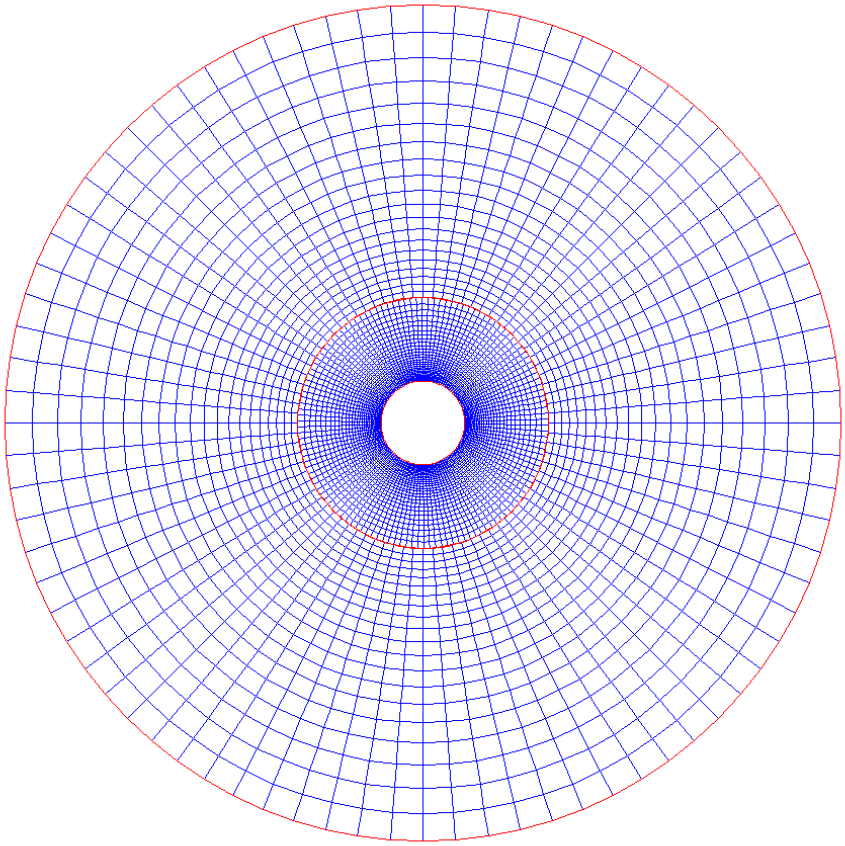


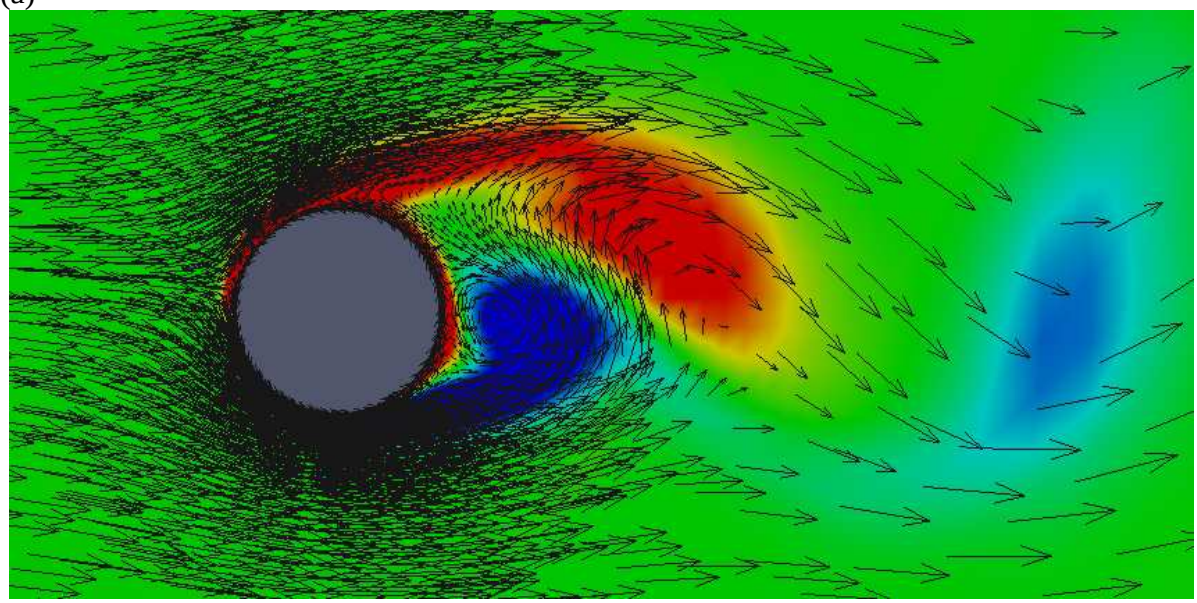
**Figure 2.5:** 2-d rotating cylinder – computational mesh.

Figure 2.6 shows shaded plots of flow variables (pressure and vorticity), overlaid by velocity vectors and streamlines respectively. The sliding interface cannot be detected in these plots, indicating that the interface is successfully transmitting flow information.

Figure 2.7 shows computed lift coefficient, comparing results for two timestepping schemes (Crank-Nicolson and Gear's method) and also the results with a completely stationary mesh (but still rotating cylinder walls). The non-dimensional timestep used is $\Delta t.U_0 / D = 0.004$. There are small differences in extreme values between stationary and rotating meshes, but the basic lift-coefficient distributions – peak and trough values of 0.4 and 2.1, with a Strouhal

number $fD/U_0 = 0.195$ – are consistent with the results of Mittal and Kumar (2003).

(a)



(b)

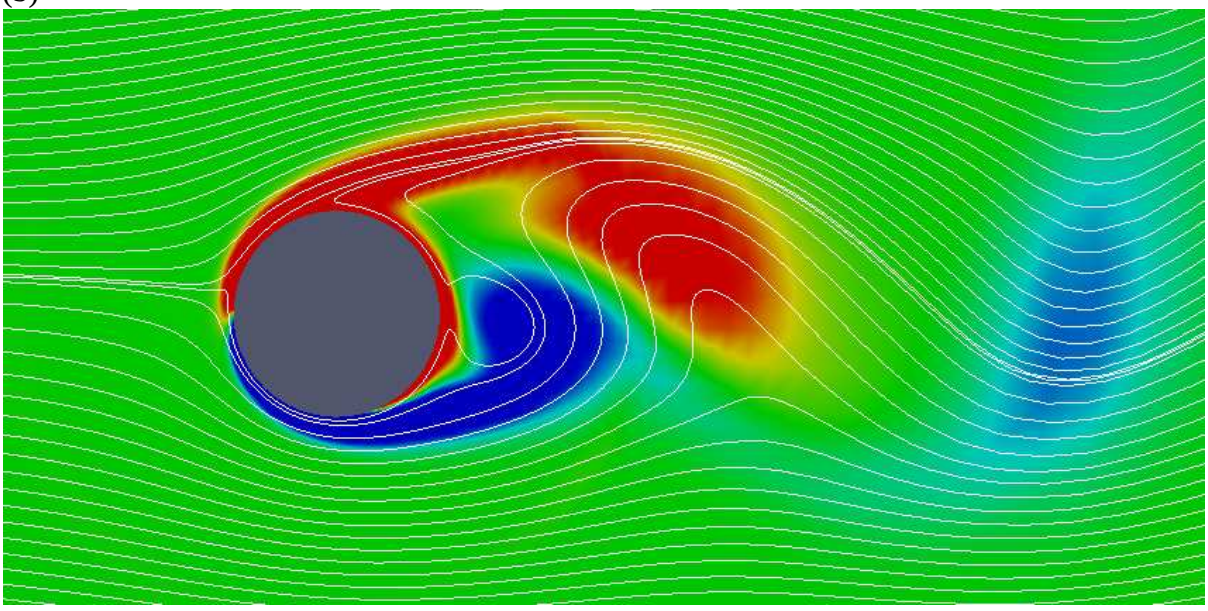


**Figure 2.6:** 2-d rotating cylinder: (a) pressure and flowfield; (b) vorticity and streamlines.

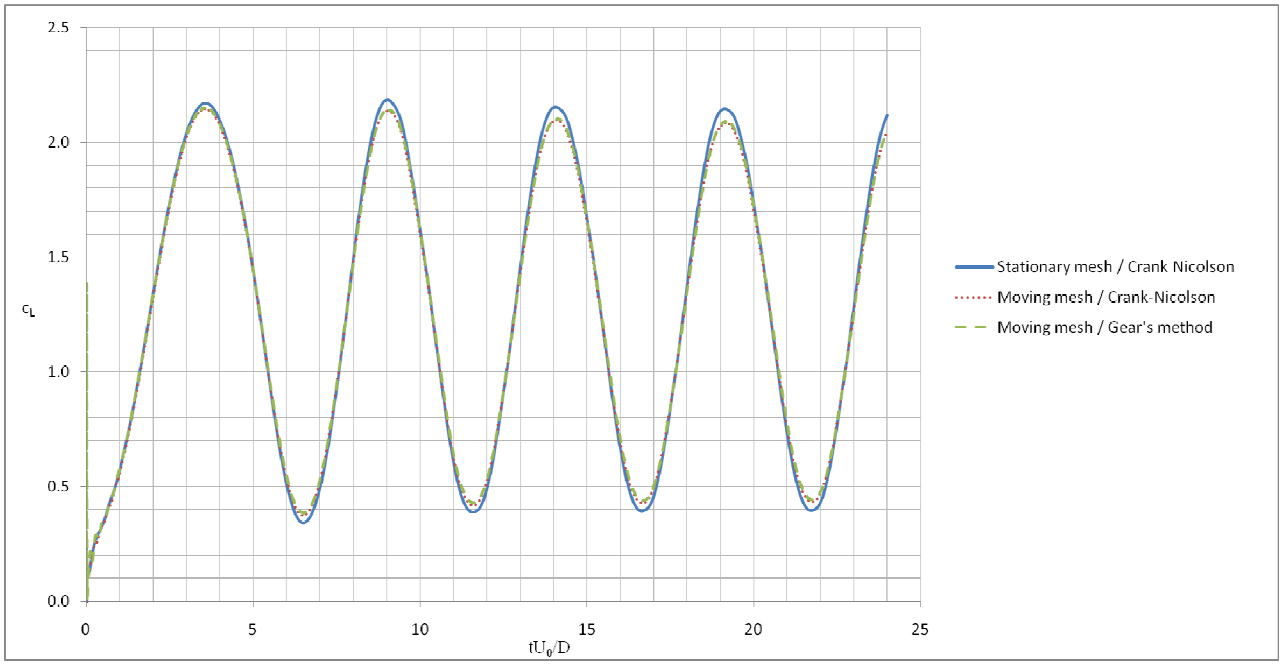



**Figure 2.7:** 2-d rotating cylinder – lift coefficient.

A second case considered was that of a rotating cube. The multiblock structure for this is shown in Figure 2.8. A central cylindrical core including the cube spins inside an outer cuboid domain. There are sliding interfaces on the upstream, downstream and curved surfaces of the cylindrical mesh: the same topology as will be employed for a marine current turbine. The flow is laminar with Reynolds number (based on cube side $L$ and approach-flow velocity $U_0$) of 100. The non-dimensional angular velocity $\Omega L/U_0$ is 1.0. The non-dimensional timestep $\Delta t\ U_0/L = 0.005$ and the Crank-Nicolson timestepping method is employed.

Figure 2.9 shows the streamwise pressure distribution and streamlines. As for the previous test case there is no detectable signature of the sliding interface.

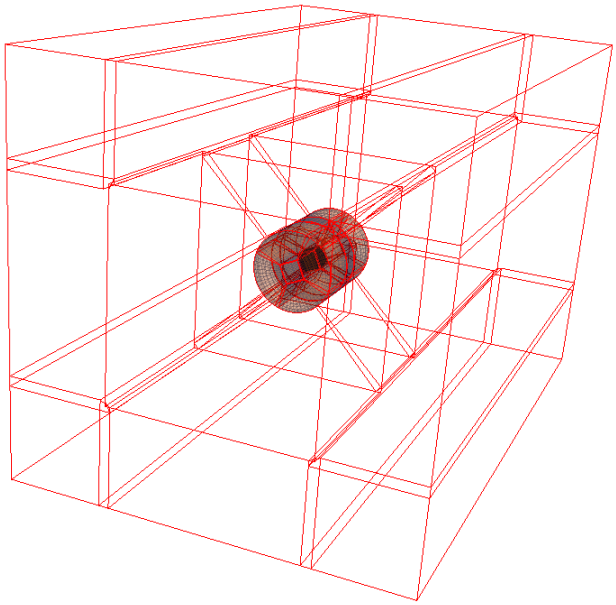
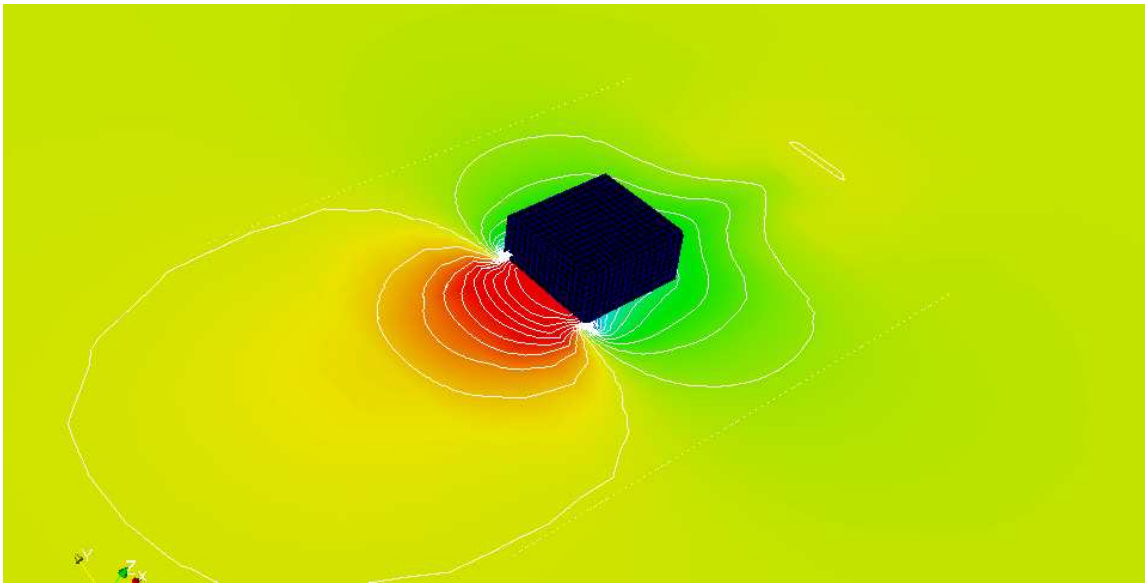


**Figure 2.8:** 3-d rotating cube – mesh blocks.

(a)



(b)

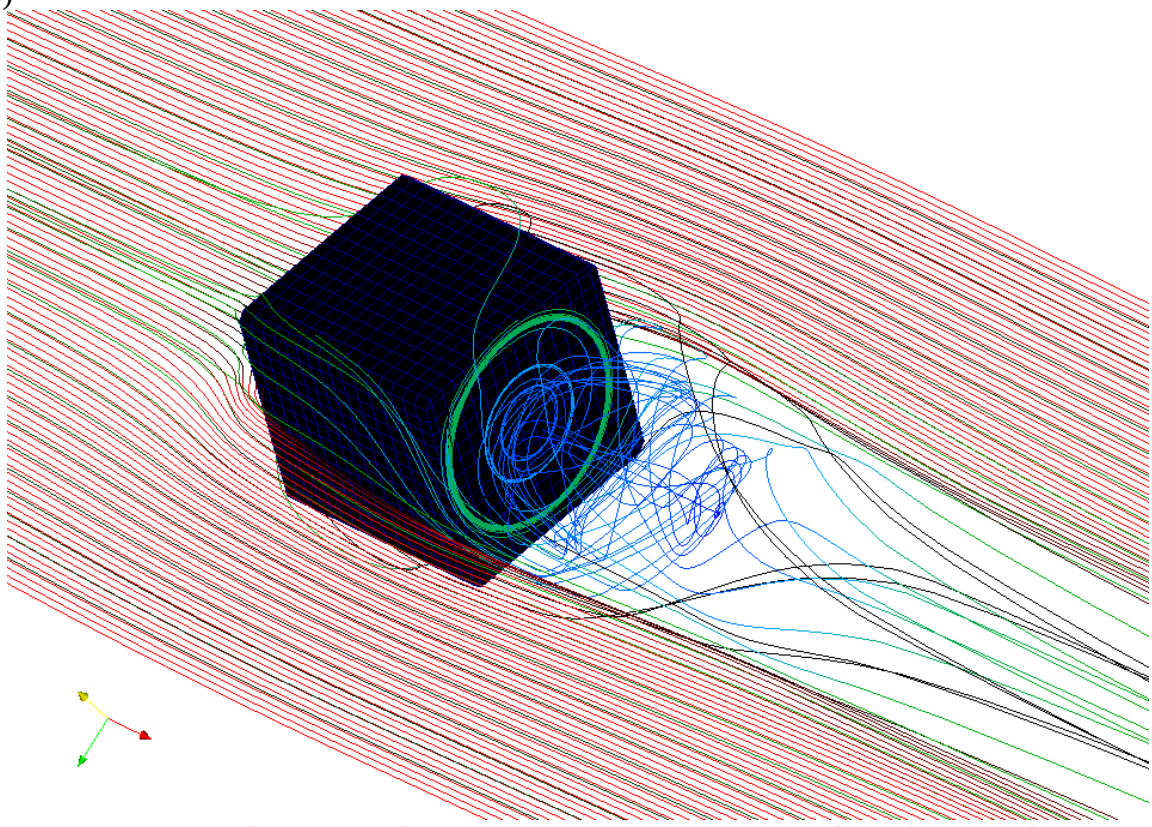


**Figure 2.9:** 3-d rotating cube: (a) pressure on a streamwise slice; (b) streamlines.

*2.4 Implementation in Code_Saturne*

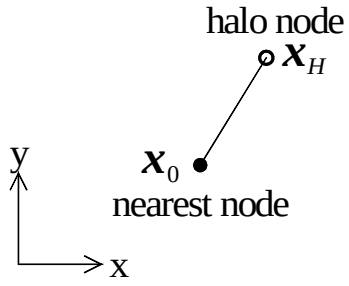



**Figure 2.10:** unstructured mesh – location relative to nearest node.

In our current implementation in *Code_Saturne* the value of a variable $\phi$ at a halo node $\boldsymbol{x_H}$ is determined by (see Figure 2.10):

- finding the *nearest* node $\boldsymbol{x_0}$;
- using a truncated Taylor-series expansion using the computed derivatives at $\boldsymbol{x_0}$:

$$\phi_H = \phi(\boldsymbol{x}_0) + (\boldsymbol{x}_H - \boldsymbol{x}_0) \bullet \nabla\phi(\boldsymbol{x}_0)$$

A potential deficiency is that $\phi$ will change discontinuously as the halo node is "passed" from one nearest node to its neighbour as the mesh slides. In practice, this does not seem to be a problem, but alternative interpolation schemes are being considered.

The method described in the previous section is implemented in Code Saturne v2.0.1. A tree structure of the key subroutines is shown in Figure 2.11. Modified subroutines are shown in italics and completely new subroutines underlined. A description of the subroutines follows.

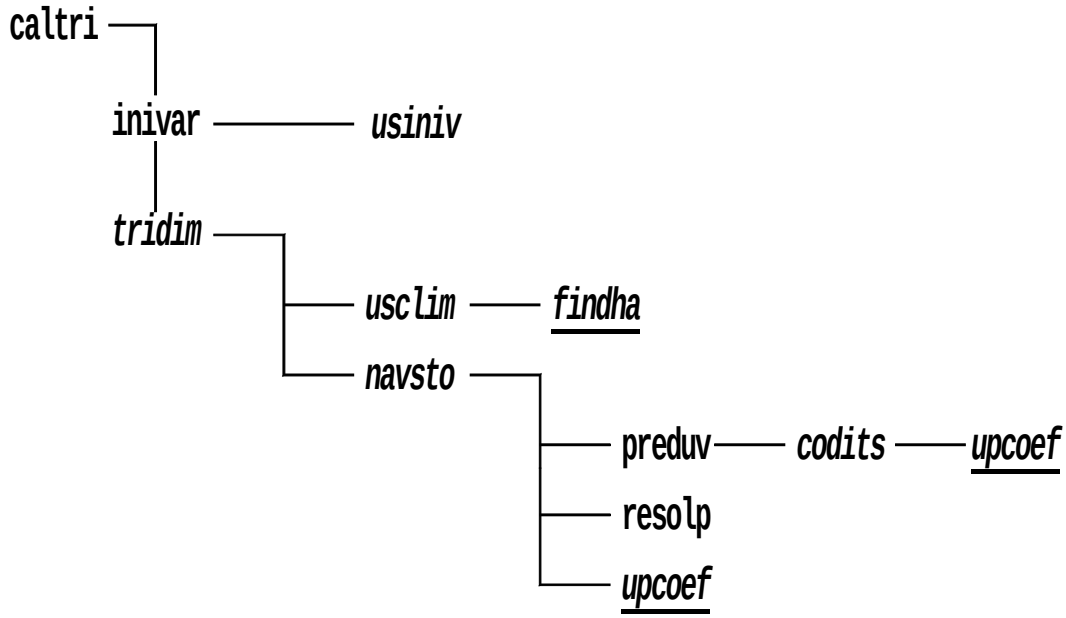



**Figure 2.11:** tree structure of key subroutines used in internal-interface development; italics indicates "modified" and underlining indicates "new".

**caltri**

The main calling tree of Code Saturne. It controls the fluid solver as well as writing the post-processing files.

***codits***

Iterative solver for a variable.

***findha***

The search routine for closest cell centre to the halo node. Each side of the interface searches through the cells only on the other side.

***navsto***

The Navier-Stokes solver. Calls the velocity prediction and pressure-correction subroutines as well as making the velocity correction. After velocity and pressure corrections the internal interface is updated by calls to ***upcoef***.

**preduv**

The velocity prediction.

**resolp**

The pressure correction.

***tridim***

Calls the main parts of the fluid solver. One call of ***tridim*** essentially corresponds to one time step. ***tridim*** calls the boundary-condition updaters as well as the Navier-Stokes solvers and mesh modifications using ALE. The subroutine has been modified so that calls to the turbulence subroutines are made within the inner loop. An inner-loop convergence criterion based on normalised residuals has been added.

***usclim***

Defines the boundary conditions. Here it is also used to define the internal interface as a Dirichlet boundary. The search routine (***findha***) is called and the values of all variables on the interface are updated.

***usiniv***

Called once at the start of a calculation and once after a restart. It is used to initialise the variables in the flow. In this application cells on either side of the internal interface are identified and their cell-centre coordinates stored.

Because *Code_Saturne* is very highly parallelised and cells abutting the interface may be attached to many different processors, memory management has to be put in place to assemble pointers to all interface-abutting cells. An investigation of the overheads associated with interfacing and the associated mesh movement has been performed and is illustrated in Figure 2.12. The case is a 3.1-million-cells calculation of flow about a sphere and was performed on the University's Red Queen research cluster; (speed-up is normalised by the computation time for one 12-core "blade"). The Figure indicates that *Code_Saturne* parallelises very well, but that overheads associated with the ALE method can easily exceed 25% of processor time.

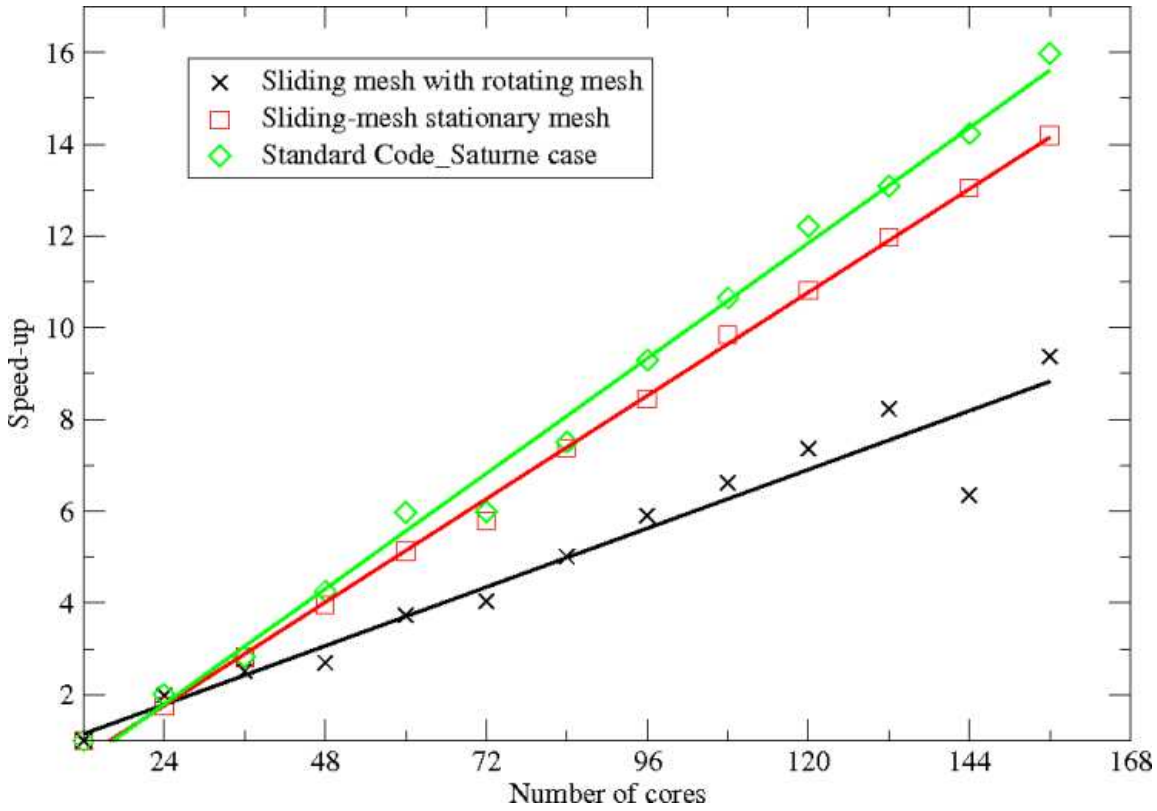
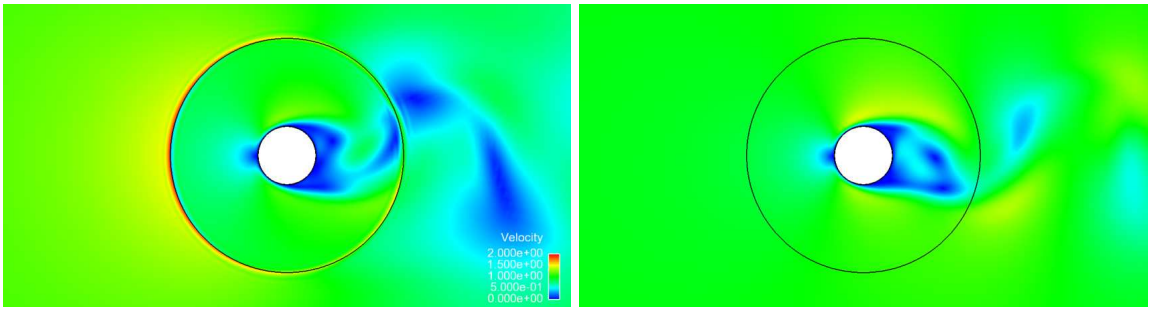



**Figure 2.12:** speed-up of different interface methods (flow about a sphere; 3.1 million cells)

Validation test cases have been computed to examine the behaviour of the interfacing method in *Code_Saturne*. These are 2-d flow about a stationary or rotating cylinder at Re = 200 and 3-d flow about a sphere.

The default number of iterations of the inner loop (NTERUP) in *Code_Saturne* is 1, making each timestep explicit. Implicit schemes are inherently more stable and when there is a sliding interface values on the interface must necessarily be obtained iteratively. Testing suggests that a minimum number NTERUP = 5 is necessary for the cylinder test case (see Figure 2.13 for example). As this is likely to be test-case- (and timestep-) dependent, we have also developed an inner-loop convergence criteria based on the normalised sum of residuals to establish convergence within a timestep more reliably.




NTERUP = 3 NTERUP = 5

**Figure 2.13:** effect of changing the number of inner-loop iterations on continuity at an interface.

By examining the fluctuating lift coefficient for a rotating cylinder we also investigated how far the interface could be allowed to slide in one timestep. A CFL-like condition – that halo cells should not pass more than one donor cell in a timestep – was envisaged. However our findings were that this was not sufficient and, in general, it is recommended that the interface should not slide by more than half a cell in one timestep.

The effect of different relative rotation rates (for a definition of α see Section 2.3) and different timestepping schemes were also investigated. Figure 2.14 shows values of fluctuating lift as a function of time at $\alpha = 0.5$ and $\alpha = 1.0$. With 2nd-order timestepping the values for both $\alpha = 0.5$ and $\alpha = 1.0$ compare very favourably with the results of Mittal and Kumar (2003), with very little difference between results with our own interface method (with and without rotating the mesh) and the normal "pasting" approach for composite meshes of *Code_Saturne*. However, it also indicates that a second-order timestepping method is highly desirable for accuracy in this case.

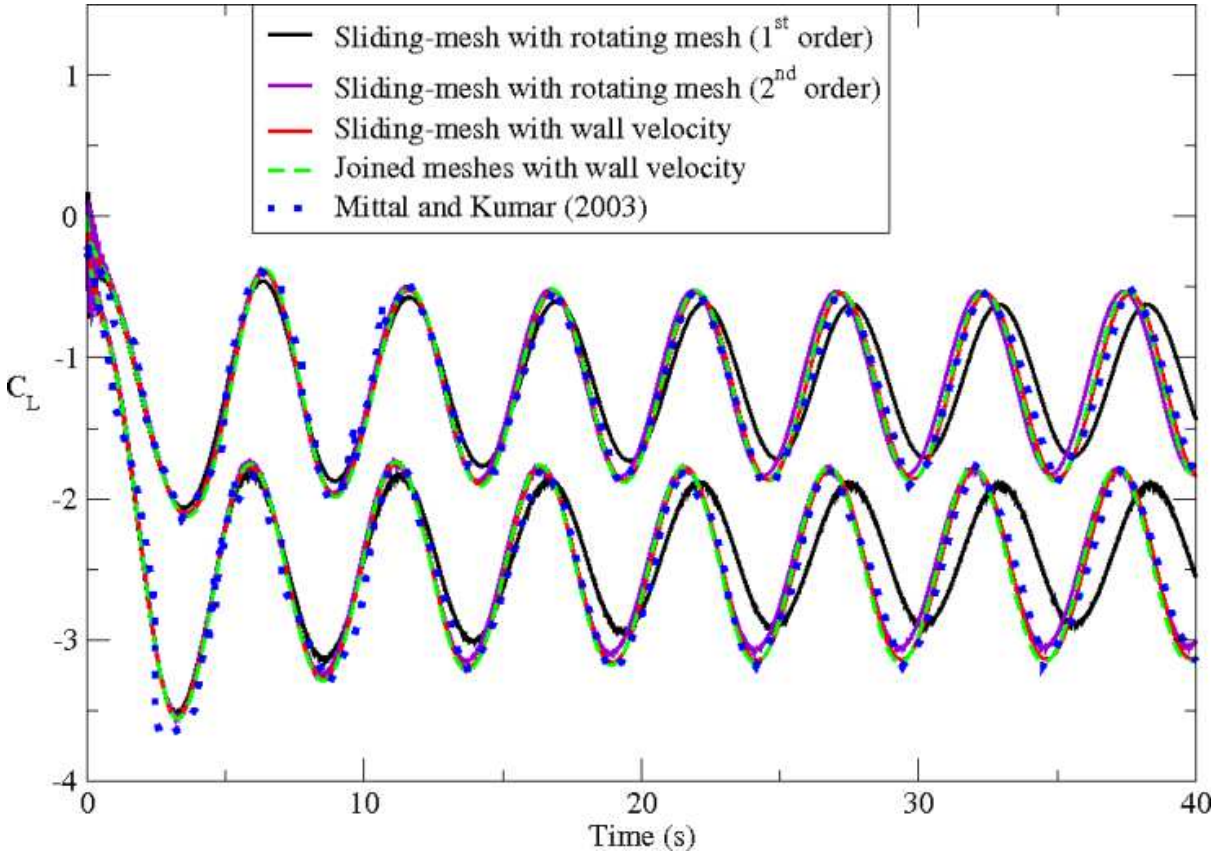



**Figure 2.14:** time-dependent lift coefficient for three methods of simulating a rotating cylinder; upper graphs: $\alpha = 0.5$; lower graphs: $\alpha = 1.0$.

The new interfacing method in *Code_Saturne* has also been tested for flow about a rotating sphere. The flow is obviously important to fans of ball sports, but also in ballistics and aerosol science. Experimental data with which to compare can be found in Oesterlé and Dinh (1998), whilst computations can be found in Kim (2009). The experiments considered Reynolds numbers and non-dimensional rotation rates in the ranges

$$10 < Re < 140 \qquad \text{and} \qquad 1 < \alpha < 6$$

The mesh and boundary conditions are shown in Figure 2.15 and an overview of the computed flow field in Figures 2.16 and 2.17. The position of the interface has been marked in the latter: its position is gratifyingly not detectable in the flow-field plots.

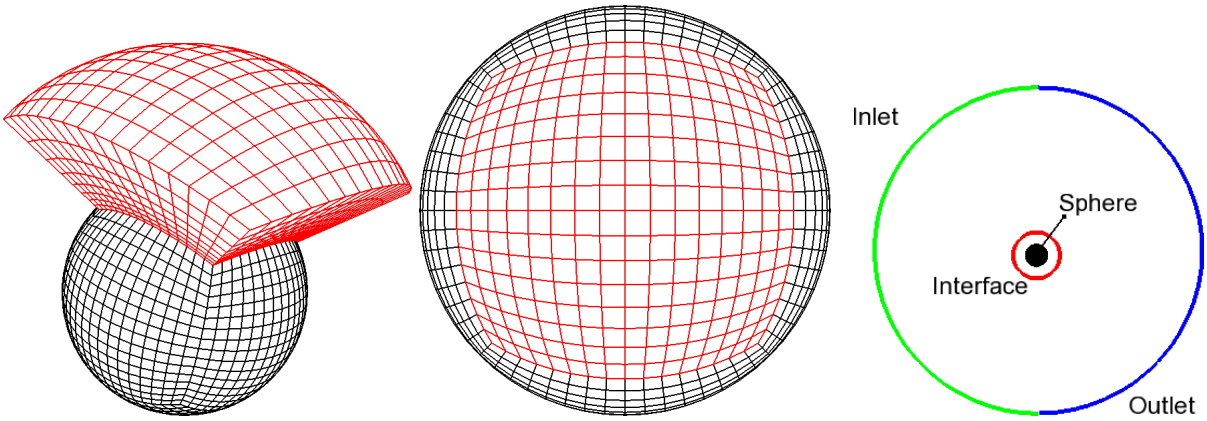



**Figure 2.15:** mesh and boundary conditions for computations of flow about a sphere.

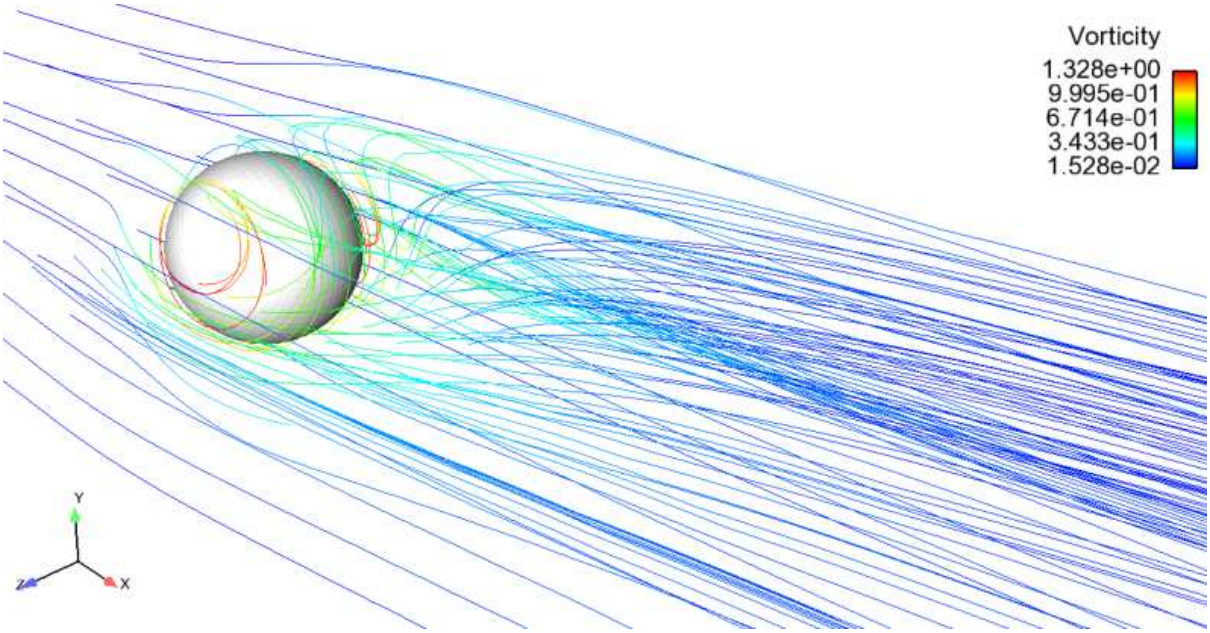
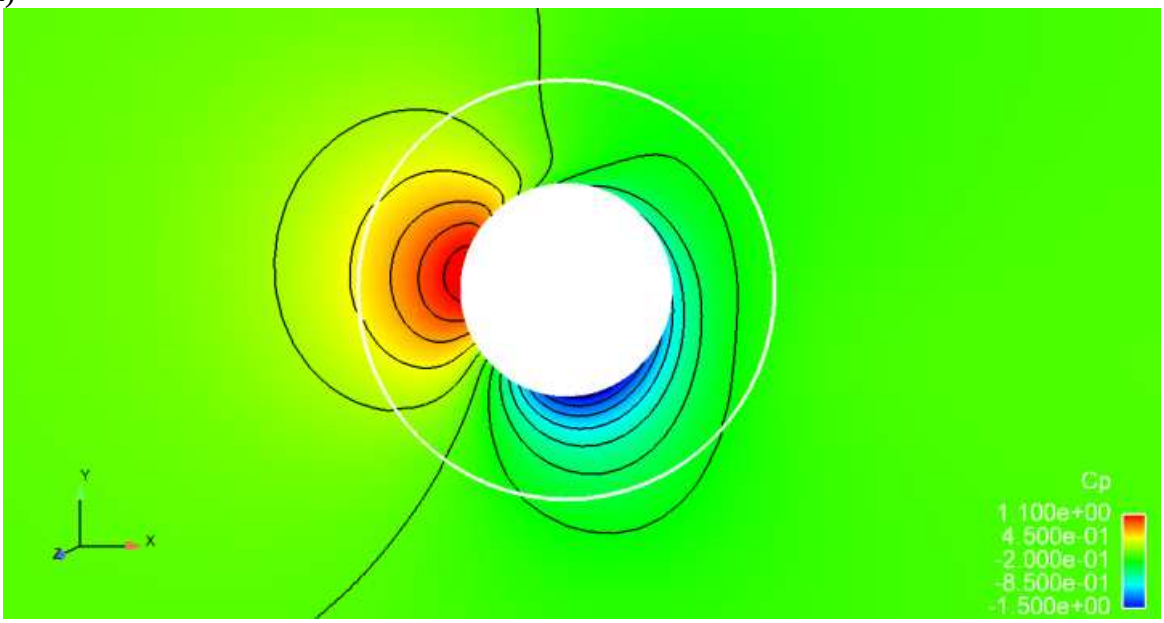



**Figure 2.16:** computed flow about a sphere: streamlines coloured by vorticity

(a)




(b)

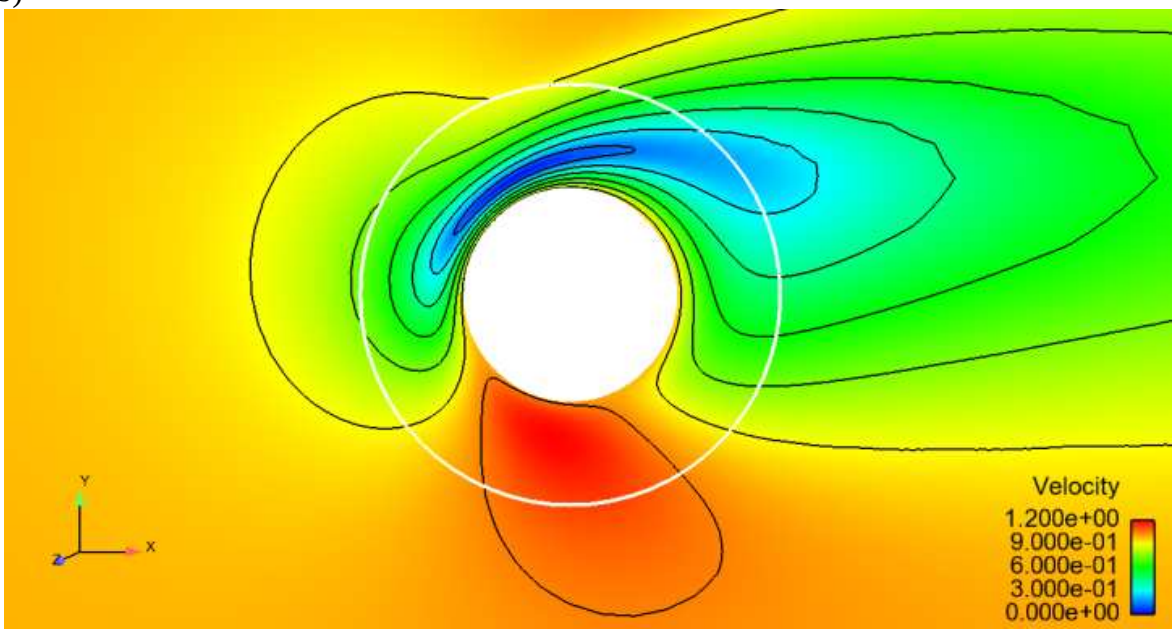



**Figure 2.17:** computed flow about a sphere; (a) pressure coefficient; (b) velocity magnitude.

Computations have focused on the particular case of Re = 25.45, $\alpha = 1.0$ (case Ba12 in Oesterlé and Dinh, 1998). Although this flow could obviously be computed with a single non-rotating mesh and imposed wall velocity of the sphere our object was to test the interfacing routines and to this end we have generated a mesh in two parts, corresponding to the regions inside and outside an internal interface at twice the sphere radius. We consider three strategies for computing the flow:

- case A: sliding-mesh interface routines with inner mesh rotating with the sphere;
- case I: interface routines but mesh actually stationary (sphere surface rotating);
- case J: usual *Code_Saturne* pre-processor routines to join or "paste" the meshes.

Preliminary results for drag and lift coefficients are shown in Table 2.1. Results thus far are only preliminary and grid- and timestep convergence has clearly not been demonstrated. In particular, we have only tried first-order implicit timestepping and results from the cylinder calculation have already shown that second-order timestepping is to be preferred for accuracy at moderate sizes of timestep. However, our own interfacing routines agree well with the more complex method used by the *Code_Saturne* pre-processor to "paste" two meshes. The rotating-mesh method – which is not strictly needed for this case – will clearly require either a smaller timestep or a more accurate timestepping method.

| Case: | $C_D$ | $C_L$ |
|---|---|---|
| A: interface routines with rotating inner mesh | 2.467 | 0.842 |
| I: interface routines with stationary mesh, rotating sphere | 2.506 | 1.011 |
| J: meshes joined by *Code_Saturne* | 2.509 | 1.011 |
| Oesterlé and Dinh (1998) | – | 1.196 |

**Table 2.1:** Drag and lift coefficients for flow about a rotating sphere.

## 3. MARINE CURRENT TURBINE: GEOMETRY AND SIMULATION DETAILS

### *3.1 Description of Experiment*

The MCT device to be simulated is that of Bahaj et al. (2007b). In that series of experiments power and thrust coefficients were measured for a range of tip speed ratios and pitch settings in a towing tank (and also a cavitation tunnel, although these will not be simulated here). The 3-bladed MCT had a rotor diameter of 800 mm and nacelle diameter 100 mm. The support diameter was also 100 mm. Blades were constructed from NACA 63-8xx profiles; chord, pitch and thickness distributions are given at 17 stages along the blade. The experiments showed considerable sensitivity to the overall pitch of the blades; in all our simulations we have used the optimal blade root pitch setting of 20°, although the mesh-generation scripts could be adapted to use other values. The towing tank had breadth 3.7 m and depth 1.8 m, with the rotor centred 0.84 m below the free surface. The area blockage ratio is 7.5% and Bahaj et al. made thrust-dependent blockage corrections before presenting their data, which we have also done for consistency. Power and thrust coefficients were measured at tip-speed ratios (TSRs) of up to 12, with optimal power takeoff at a TSR of about 6.

### *3.2 Computational Mesh*

A geometry and mesh has been built with Fluent's meshing tool Gambit to conform as closely as possible to that in the experimental measurements of Bahaj et al. (2007b). A complete geometry has now been modelled, including the support. This is composed of 5 separate meshes (see Figure 3.1) – an inner turbine mesh and 4 outer meshes. The latter, non-rotating, parts are pasted together with *Code_Saturne*'s pre-processor, with the interface between these and the rotating turbine mesh being handled by our own sliding-interface procedure. A typical cell count is given in Table 1. (Prior to the full-geometry simulations a number of different grid resolutions were tested with RANS models on the rotating blade and nacelle geometry, without the mast and various different meshes of up to 4 million cells have been used during development.) The domain extends a distance 3*D* upstream and 10*D* downstream of the rotor, where *D* is the rotor diameter.

| Region | Number of cells |
|---|---|
| Turbine | 1 165 878 |
| Mast | 252 768 |
| Upstream | 231 360 |
| Surround | 205 600 |
| Downstream | 242 900 |
| **Total:** | **2 098 506** |

**Table 1:** numbers of cells in different portions of the mesh.

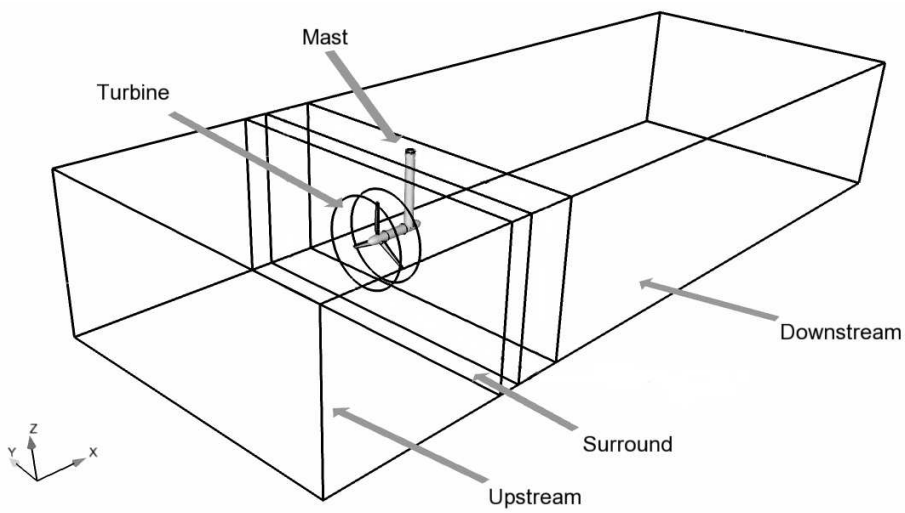



**Figure 3.1:** subdivision of the flow domain into different mesh regions

Individual blades are meshed using a C-mesh with 62 points around the profile and ten cells on the blunt trailing edge (Figure 3.2). Polyhedral elements are required outside the prism layer on the blade tips to ensure the outer surface can be meshed in a structured manner. This is to enforce regularity at the sliding-mesh interface. The circumferential sliding interface is non-conforming, with 240 cells on the inside and 160 cells on the outside. This flexibility of the interface method assists us in reducing the overall cell count in less important regions of the flow. The mesh near a blade is illustrated in Figure 3.2 and the overall surface mesh on all solid boundaries, rotating or otherwise, is shown in Figure 3.3.

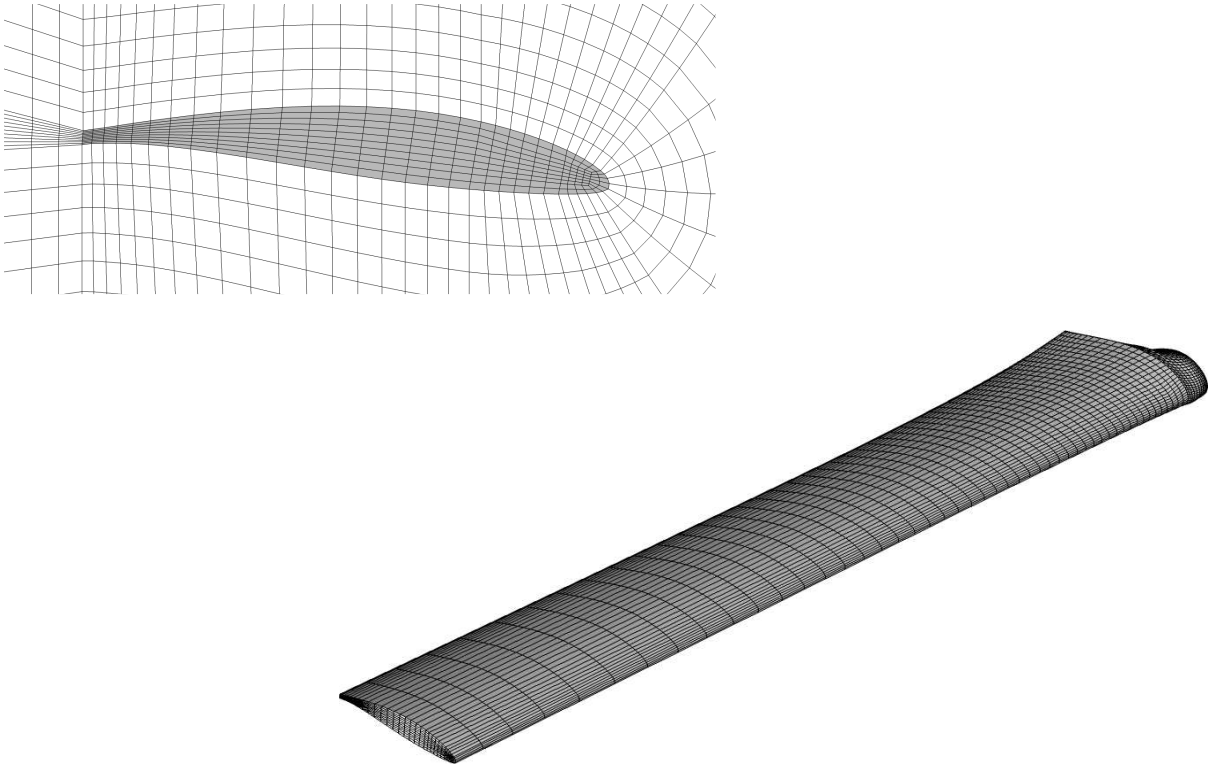


**Figure 3.2:** detail of the mesh about a blade.

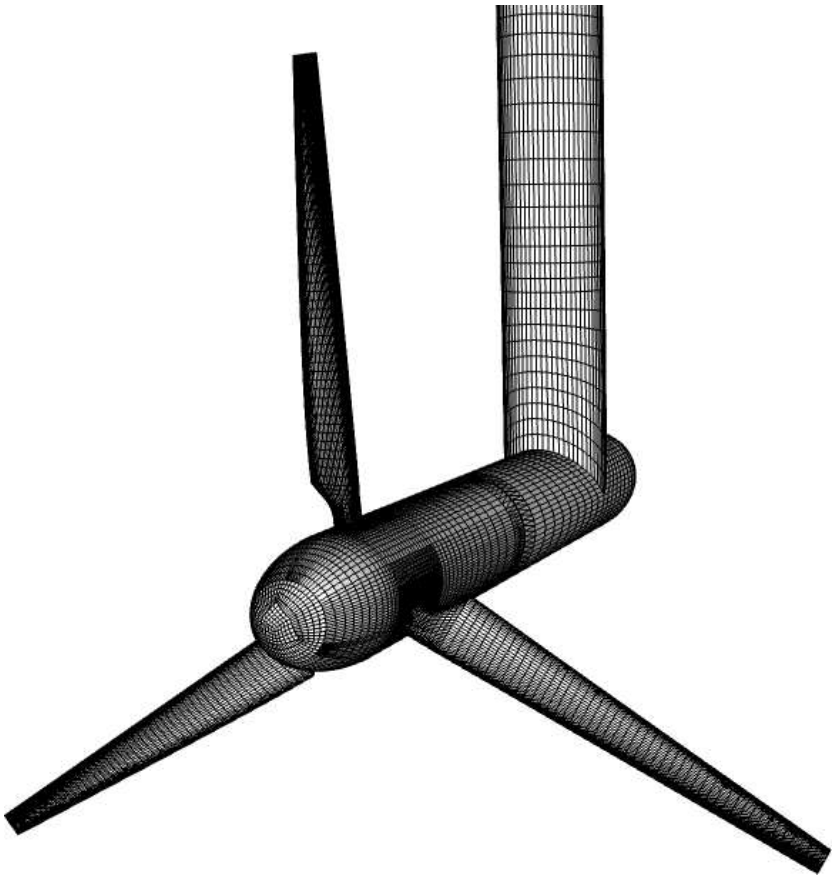


**Figure 3.3:** overall surface mesh.

### *3.3 Flow Parameters*

For reference we note here the definition of the principal performance parameters:

Tip-speed ratio: $\mathrm{TSR} = \dfrac{\Omega R}{U_0}$

Thrust coefficient: $C_T = \dfrac{thrust}{\frac{1}{2}\rho U_0^2 A}$

Power coefficient: $C_P = \dfrac{power}{\frac{1}{2}\rho U_0^3 A}$

Here, $\Omega$ is the angular velocity, $R$ the rotor radius, $U_0$ the approach velocity (or towing speed) and $A$ the area swept out by the rotor. Power is determined from torque × angular velocity.

In *Code_Saturne*, where all variables are entered in metre-second units, the following conditions were applied:

| | |
|---|---|
| inlet velocity: | $U_0 = 1$ m s$^{-1}$ |
| turbulence intensity: | 0%, 10% or 20% (0% replaced by 1% for RANS) |
| rotor diameter: | $D = 0.8$ m (radius $R = 0.4$ m) |
| angular velocity: | $\Omega = 10, 15, 20$ rad s$^{-1}$ (1 revolution ↔ 0.628, 0.419, 0.314 s) |
| density: | $\rho = 1000$ kg m$^{-3}$ |
| viscosity: | $\mu = 1.0\times10^{-3}$ kg m$^{-1}$ s$^{-1}$ |

These correspond to non-dimensional parameters

$$\mathrm{Re} \equiv \frac{\rho U_0 D}{\mu} = 8.0\times10^5$$

$$\mathrm{TSR} \equiv \frac{\Omega R}{U_0} = 4,\ 6 \text{ or } 8$$

### *3.4 Blockage Correction*

Although we simulate exactly the same flow domain as in the experiments, i.e. our side boundaries correspond to those of the towing tank, Bahaj et al. (2007a) did not present the raw values of $C_P$, $C_T$ and TSR in their reports but instead made a blockage correction to present them as equivalent "open-water" measurements. In order to compare like with like we make the same correction, which depends on the area blockage ratio $A/C$ and the thrust coefficient $C_T$. Details are given in Bahaj et al.'s paper. The sequence of calculations follows.

- Solve (numerically, e.g. by repeated bisection) for $r$:

$$r - \frac{\frac{A}{C}(r^2-1)}{1+\sqrt{1+\frac{A}{C}(r^2-1)}} = \sqrt{\frac{r^2-1}{C_T}}$$

  ($r$ corresponds to $U_3/U_2$ in Bahaj et al.'s notation)

- Find

$$\frac{U_1}{U_0} = \frac{r+1}{1+\sqrt{1+\frac{A}{C}(r^2-1)}} \times \sqrt{\frac{C_T}{r^2-1}}$$

$$\frac{U_0}{U_0'} = \frac{U_1/U_0}{(U_1/U_0)^2 + C_T/4}$$

  ($U_1$ is the velocity at the disk according to actuator-disk theory; a prime denotes an equivalent "open-water" quantity.)

- The blockage-corrected coefficients are then:

$$C_P' = C_P\left(\frac{U_0}{U_0'}\right)^3$$

$$C_T' = C_T\left(\frac{U_0}{U_0'}\right)^2$$

$$\mathrm{TSR}' = \mathrm{TSR}\left(\frac{U_0}{U_0'}\right)$$

In the present case the area blockage ratio $A/C = 0.075$. With a typical thrust coefficient $C_T = 0.8$ this gives $U_0/U_0' = 0.973$ and multiplicative blockage-correction factors 0.922, 0.947 and 0.973 for power coefficient, thrust coefficient and TSR respectively. Note that the corrections depend on $C_T$ and can become very significant for large thrust.

### *3.5 RANS Simulations*

Using the model described in Section 3 flow simulations were undertaken with the standard $k$-ε model, using *Code_Saturne*'s "scalable" wall functions. Dirichlet conditions were applied to all transport variables at inflow, with a fixed-pressure boundary condition is employed at outflow. Side-wall boundaries and top and bottom of the domain were modelled as slip walls (the most appropriate boundary condition to simulate a towing tank).

Three different turbulent intensities ($i$) were considered: 1%, 10% and 20%. The inlet turbulence scalars were related to these by

$$k_0 = \tfrac{3}{2}(U_0 i)^2, \qquad \varepsilon_0 = \frac{C_\mu^{3/4} k_0^{3/2}}{l_m}$$

where the inlet mixing length $l_m$ is taken as 0.7 times the turbine axis immersion depth.

Three different TSRs have also been considered: 4, 6 and 8.

In all cases a slope-limited blend of central and upwind differencing is used for advective fluxes together with first-order, fully-implicit timestepping.

The requirement that the interface slides no more than ½ cell in one timestep sets the following restriction on timestep size:

$$\Omega \Delta t \le \tfrac{1}{2} \times \frac{2\pi}{N_{cell}} \qquad \text{or} \qquad \Delta t \le \frac{\pi}{N_{cell}\Omega}$$

In a typical case, $N_{cell}$ = 160, Ω = 15 rad s$^{-1}$, giving a maximum timestep of 0.0013 s. For our calculations a timestep of 0.001 s was used (except for TSR = 8, when $\Delta t$ = 0.0008). The maximum CFL number was about 70. Note that for fully-implicit timestepping there is no automatic CFL restriction. Other timescales are:

- one blade rotation (0.4189 s, or $U_0 t/D$ = 0.524 when TSR = 6);
- one domain pass-through time (10.4 s, or $U_0 t/D$ = 13)

Different cases were run on EdF's Blue Gene high-performance computer using 2048 processors and on the University of Manchester's CSF with 256 processors.

### *3.6 LES Simulations*

LES calculations have been undertaken, using the same geometry and mesh (just over 2 million cells) as the RANS calculations (including both rotor and support). Whilst the pressure distribution on the blades is relatively unchanged the less-diffusive behaviour in the wake means that vortical structures should be better preserved (which is of importance when considering fluctuating forces on both rotor and mast and the operation of turbines in arrays).

In all cases to be reported here we have used the standard Smagorinksy model for the unresolved scales, with coefficient $C_s$ = 0.065. As with the RANS calculations a slope-determined blend of central and upwind differencing is used for advective fluxes. Second-order timestepping is used with a timestep of 0.0001 s (the maximum Courant number being about 14).

Basic wall functions are used at solid boundaries. These are defined by iterating the log-linear

velocity profile:

$$\frac{u}{u_\tau} = f(\frac{u_\tau y_p}{\nu}), \qquad \text{where} \qquad f(y^+) = \begin{cases} \frac{1}{\kappa}\ln y^+ + B, & y^+ > y^+_{\text{lim}} \\ y^+ & y^+ < y^+_{\text{lim}} \end{cases}$$

for the friction velocity $u_\tau$ (from which the wall stress $\tau_w = \rho u_\tau^2$ can then be determined). $y^+$ values vary from about 7 to 400. No Van Driest damping is used.

As for the RANS calculations, three TSRs are considered: 4, 6 and 8.

Inflow turbulence intensities of 0, 10% and 20% have been employed. The non-zero turbulence intensity cases were incorporated using a basic synthetic eddy method (SEM) already built into *Code_Saturne*. Fluctuating velocities at inlet are given by

$$u_\alpha(t) = U_{\alpha 0} + r_\alpha(t) i$$

where $\mathbf{U}_0$ and $i$ are the prescribed mean velocity and turbulence intensity and $r_\alpha$ is a time series of random fluctuations with mean zero, variance 1 and an appropriate Lagrangian timescale. In this approach the turbulence is necessarily isotropic. In the future it is hoped to do simulations with the more advanced "divergence-free synthetic-eddy method" (DFSEM) currently being developed by PhD student R. Poletto, which is aimed improving the level of pressure fluctuations at inlet.

Calculations were performed on EdF's Blue Gene high-performance computer using 2048 processors. On this a single timestep took about 16-20 s, meaning a real time of about 0.48 s (just over one rotation at TSR = 6) can be computed in a day.

## 4. MARINE CURRENT TURBINE: RESULTS

Tables 2 and 3 gives the main quantitative outputs of the simulations: namely the thrust and power coefficients as a function of:

- turbulence model (RANS or LES);
- tip speed ratio (TSR);
- inflow turbulence intensity.

To compare with experiment the second part of the table gives blockage-corrected results using the formulae of Section 3.4. This also leads to a small correction of TSR. Experimental results in the table have been read from curve fits, but values in the later graphs are taken directly from a University of Southampton data report (Bahaj, private communication)

Uncorrected

| TSR | Model | Inlet turbulence intensity | $C_T$ | $C_P$ |
|---|---|---|---|---|
| 4 | *k*-ε RANS | 1% | 0.5283 | 0.2323 |
| | | 10% | 0.5420 | 0.2047 |
| | | 20% | 0.5554 | 0.1837 |
| | LES | 0% | 0.5599 | 0.3359 |
| 6 | *k*-ε RANS | 1% | 0.7291 | 0.2365 |
| | | 10% | 0.7350 | 0.1958 |
| | | 20% | 0.7437 | 0.1592 |
| | LES | 0% | 0.8137 | 0.4174 |
| | | 10% | 0.8178 | 0.4032 |
| | | 20% | 0.8198 | 0.4183 |
| 8 | *k*-ε RANS | 1% | 0.8662 | 0.1340 |
| | LES | 0% | 0.9209 | 0.3030 |

Blockage-corrected (experimental results read from curve fits)

| TSR | Model | Inlet turbulence intensity | $C_T$ | $C_P$ |
|---|---|---|---|---|
| 3.95 | *k*-ε RANS | 1% | 0.514 | 0.223 |
| 3.94 | | 10% | 0.527 | 0.196 |
| 3.94 | | 20% | 0.539 | 0.176 |
| 3.94 | LES | 0% | 0.544 | 0.321 |
| 4 | Experiment | 0% | 0.60 | 0.40 |
| 5.87 | *k*-ε RANS | 1% | 0.697 | 0.221 |
| 5.86 | | 10% | 0.702 | 0.183 |
| 5.86 | | 20% | 0.709 | 0.148 |
| 5.83 | LES | 0% | 0.769 | 0.384 |
| 5.83 | | 10% | 0.773 | 0.370 |
| 5.83 | | 20% | 0.774 | 0.384 |
| 6 | Experiment | 0% | 0.80 | 0.440 |
| 7.75 | *k*-ε RANS | 1% | 0.812 | 0.122 |
| 7.71 | LES | 0% | 0.855 | 0.271 |
| 8 | Experiment | 0% | 0.92 | 0.390 |

**Table 2. Thrust and power coefficients**

Figures 4.1 and 4.2 summarise the computed thrust and power coefficients as a function of TSR; it includes the complete experimental data set as provided by Prof. Bahaj.

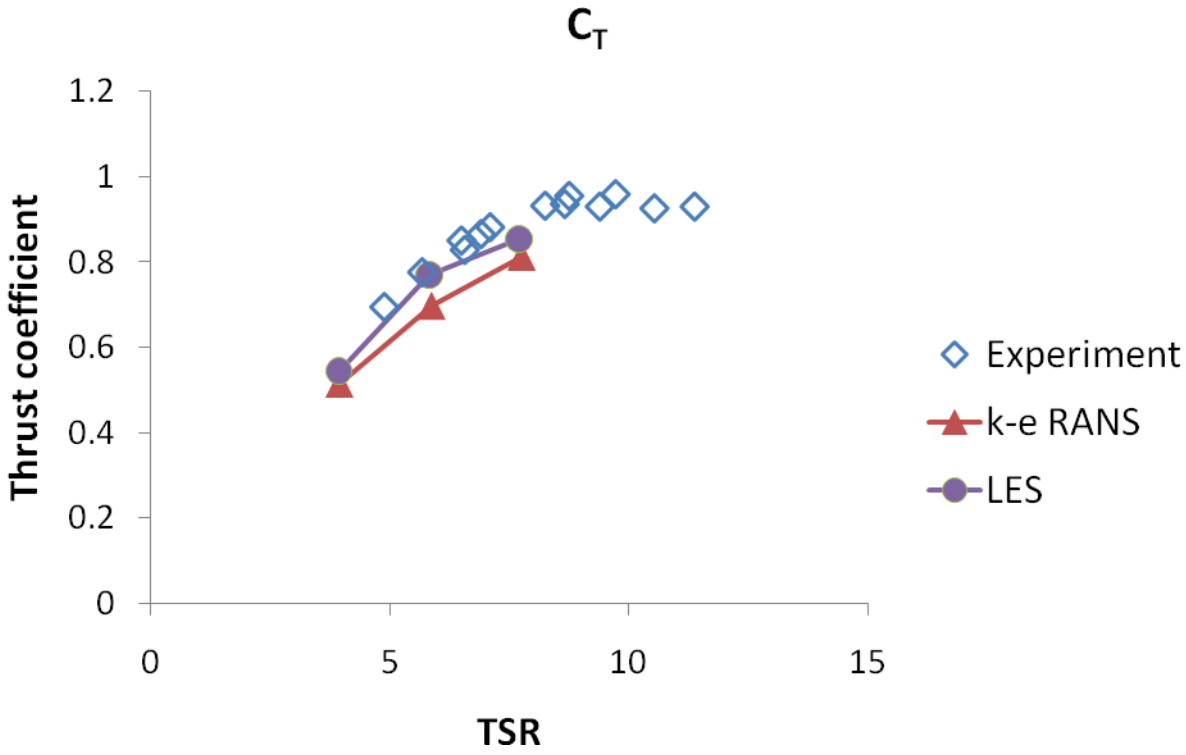



**Figure 4.1:** Thrust coefficient vs TSR at turbulence intensity 0%.

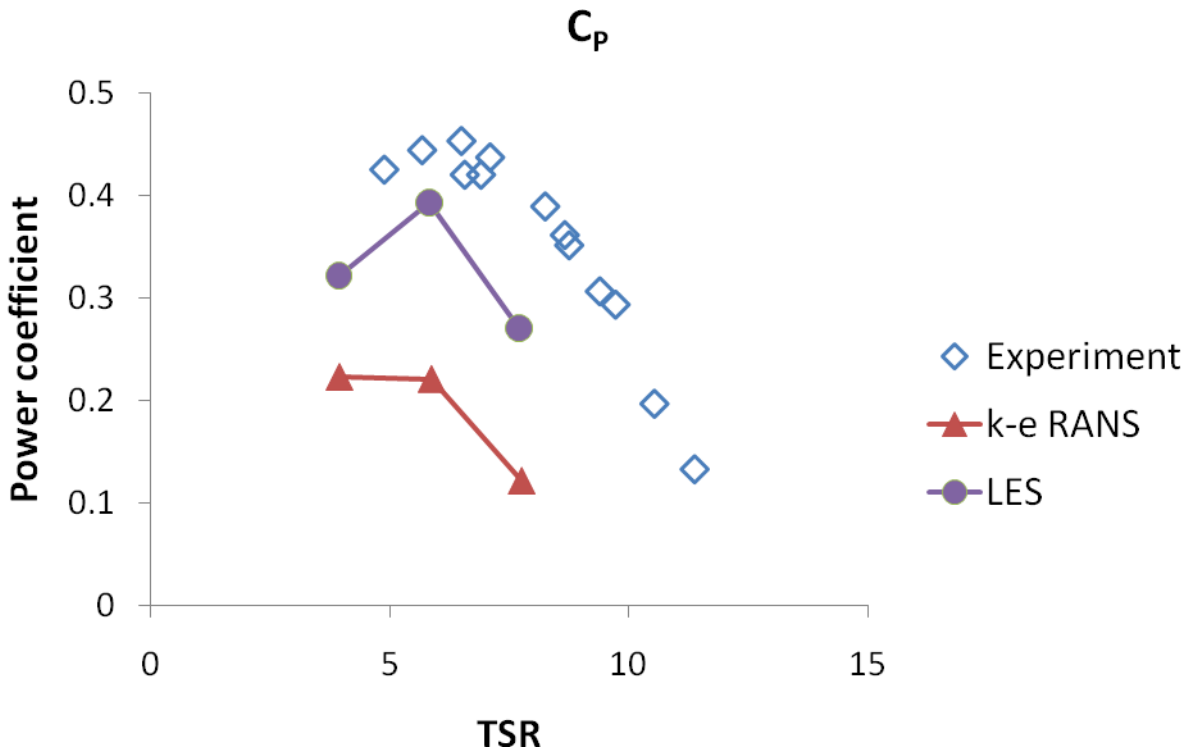



**Figure 4.2:** Power coefficient vs TSR at turbulence intensity 0%.

In general the thrust coefficient is well predicted by both methods, with LES giving slightly better agreement with experiment up to the maximum-power point (TSR = 6), but slightly less satisfactory results at off-design incidence. This is presumably related to the ability to predict boundary-layer behaviour on the blades in conditions approaching stall and merits further investigation of mesh dependence and wall-boundary treatment here.

The power coefficient is more sensitive to the computed pressure and stress distributions and is significantly under-predicted by the *k*-ε model. The LES simulations are an improvement, underpredicting the maximum $C_P$ value by about 10%; nevertheless, the underprediction at higher rotation speeds is quite significant. Note that, due to the large thrust, there is quite a significant blockage correction (about 0.894) applied to the power coefficient here, a feature that Bahaj et al. (2007) commented on.

A parametric study of the effect of inflow turbulence was also carried out and the effect on the *mean* thrust and power coefficients is presented in Figure 4.3. The effect on these *mean* coefficients is relatively modest, except for the computed reduction in power coefficient with the *k*-ε model, where an increased level of turbulence leads to thicker boundary layers and a reduction in lift on the blades.

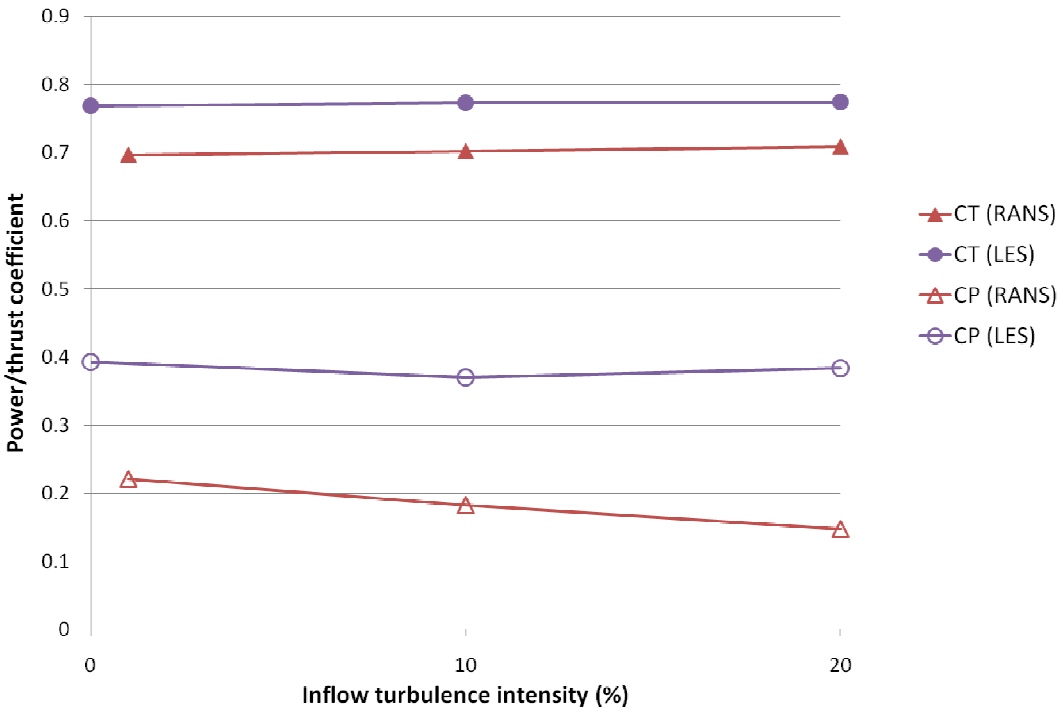
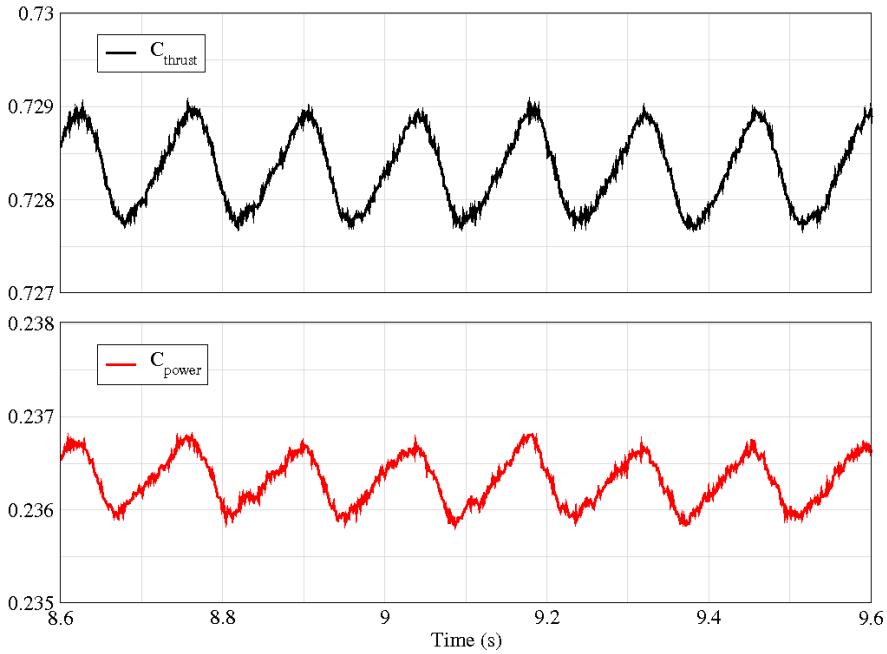



**Figure 4.3:** Thrust and power coefficients vs inflow turbulence intensity.

Inflow turbulence may, however, have a significant effect on the fluctuating loads on the turbine. This is illustrated in Figure 4.4 for RANS and Figures 4.5 for LES. The dominant mode in most figures is expected to be the 3-cycles-per-0.42 s associated with the individual blades passing in front of the support mast (temporarily raising the back pressure). This is observed in the RANS calculations (Figure 4.4). For LES, however, whilst this frequency is clear in fluctuating moments about the *y* and *z* axes (Figure 4.5b) it is not observed in the main moment giving rise to the power coefficient (Figure 4.5a). The reason is unclear.

(a) 1% inflow turbulence




(b) 10% inflow turbulence

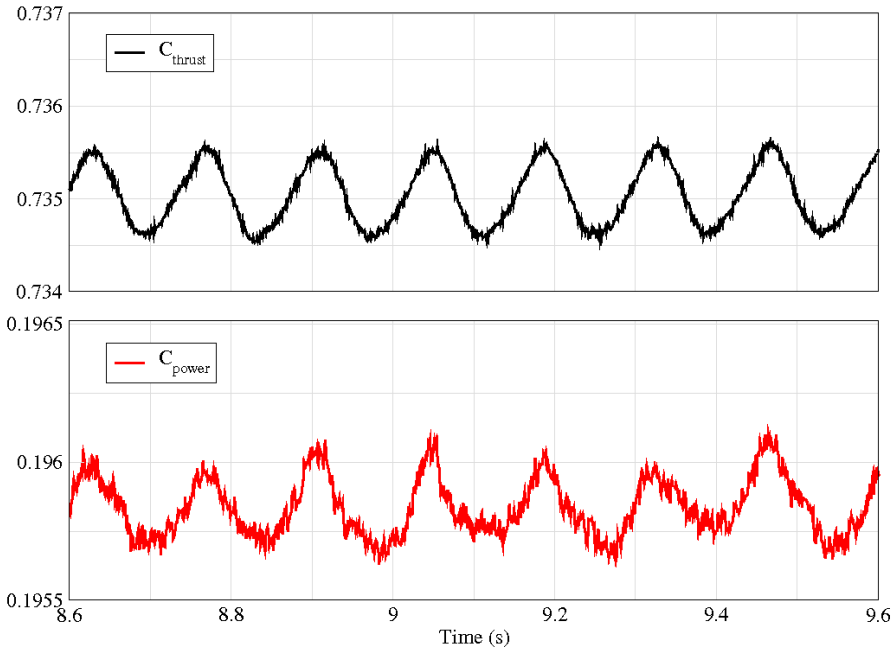
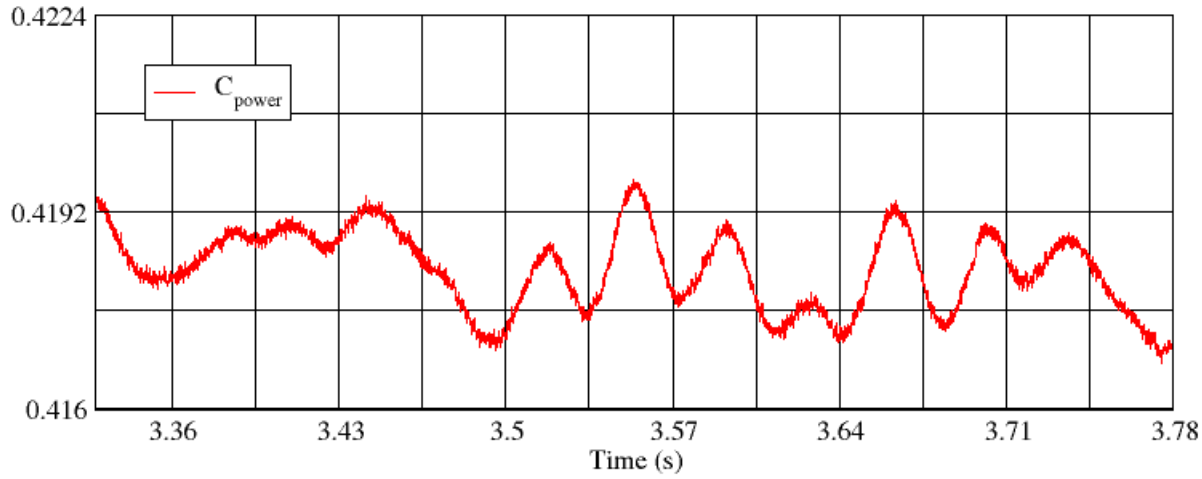



**Figure 4.4:** Time-variation of thrust and power coefficients for RANS computations

(a) Power coefficient (proportional to torque coefficient about the streamwise axis)




(b) Non-streamwise-axis torque coefficients

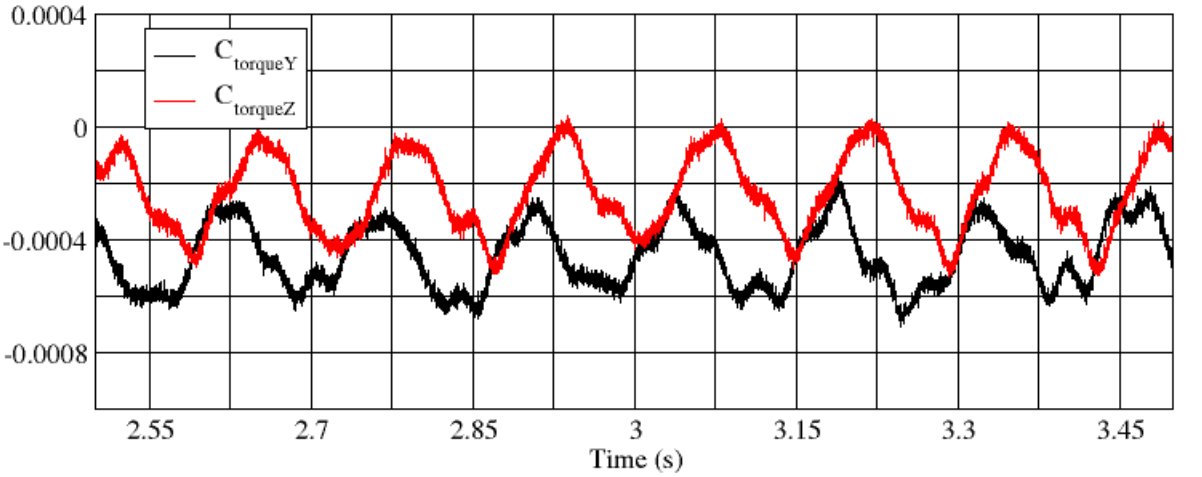



**Figure 4.5:** Time-variation of coefficients for LES computations (TSR = 6; 0% turbulence)

The effect of turbulence is complex: part of the variation is due to the fluctuating pressure field on the blades, whilst another part is due to the wake interaction with the mast. The RANS calculations suggest that for reasonable turbulence levels (say 10%) the regular fluctuations in load brought about by the changes of orientation with respect to the support mast are larger in magnitude than those associated with turbulence. However, the reason for the additional frequencies apparent in the LES power-coefficient time series are, at present, unclear.

The pressure coefficient on the blades is plotted (for RANS calculations) in Figures 4.6 and 4.7, whilst the pressure coefficient on all solid surfaces is shown in Figure 4.8. Note that there is a difference in normalisation here. On the blades in order to conform to standard aerofoil practice the normalising dynamic pressure is based on the relative approach velocity at each individual radius *r*; thus:

$$c_p = \frac{p - p_{ref}}{\frac{1}{2}\rho(U_0^2 + \Omega^2 r^2)}$$

For the global pressure distribution, however, the dynamic pressure is simply taken as that in the approach flow:

$$c_p = \frac{p - p_{ref}}{\frac{1}{2}\rho U_0^2}$$

and, as a result of the large speeds at the blade tips, the pressure coefficients in Figure 4.8 are in places considerably greater in magnitude than 1.

Plotting the pressure coefficient as a function of TSR in Figure 4.6 we note that there is a considerable reduction in lift (as evidenced by the difference in pressures on suction and pressure surfaces) when TSR = 8. This is particularly prevalent at the larger radii and is the reason for the considerable drop in power coefficient here.

Plotting the pressure coefficient as a function of inflow turbulence intensity in Figure 4.7 shows a relatively small influence of this parameter, with most of the differences occurring in the lower-speed region near the hub which does not contribute significantly to overall power.

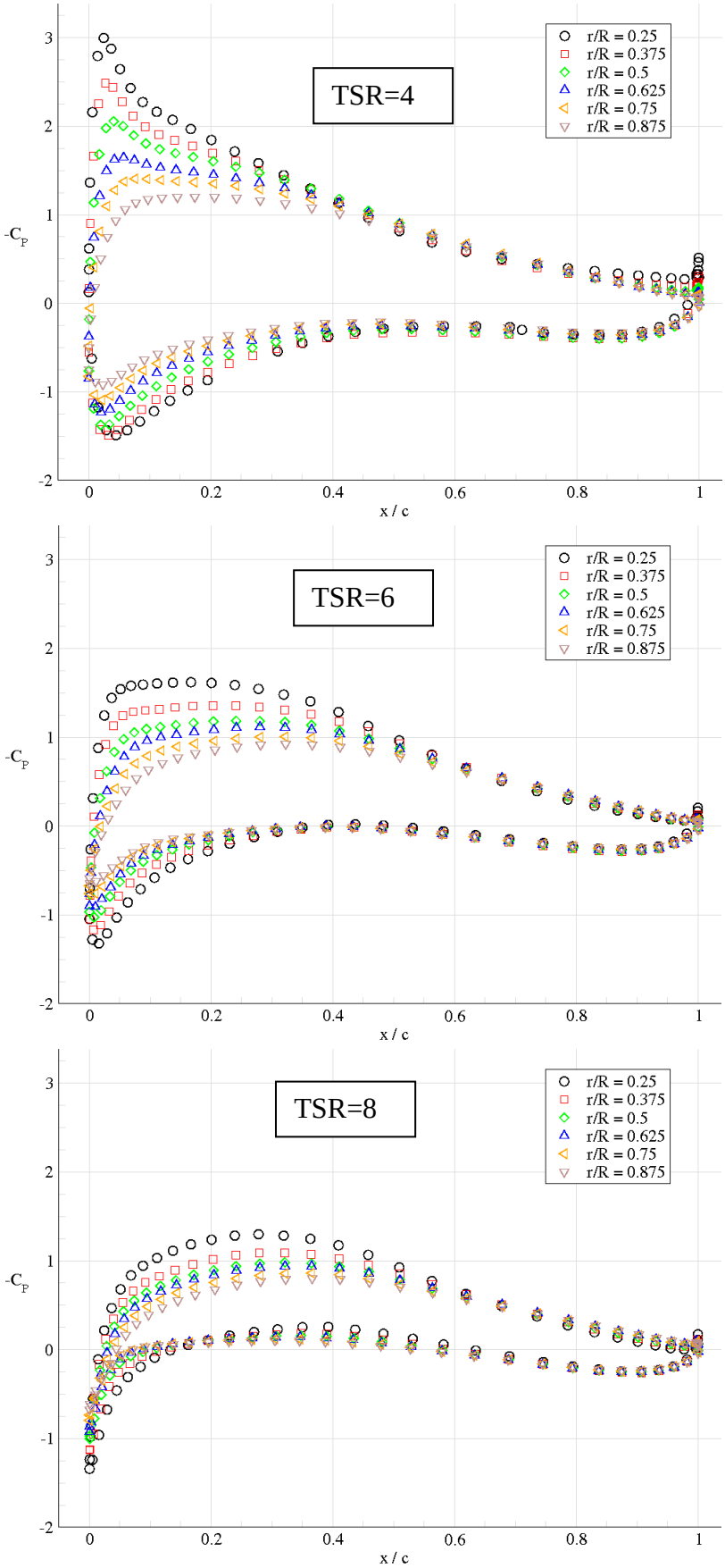



**Figure 4.6:** pressure coefficient on blade, 1% turbulence, RANS simulations; normalisation based on local approach dynamic pressure, i.e. $c_p = (p - p_{ref}) / \frac{1}{2}\rho(U_0^2 + \Omega^2 r^2)$

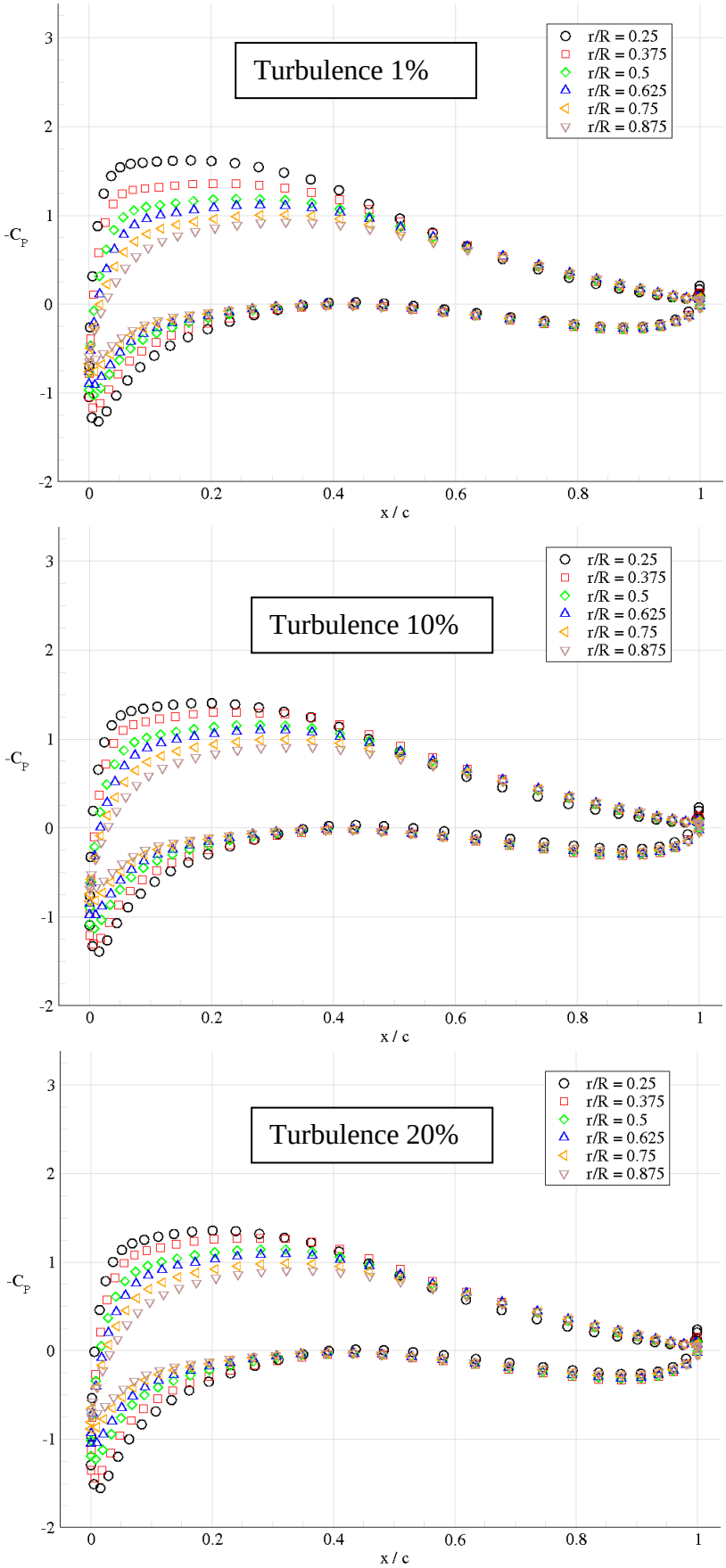
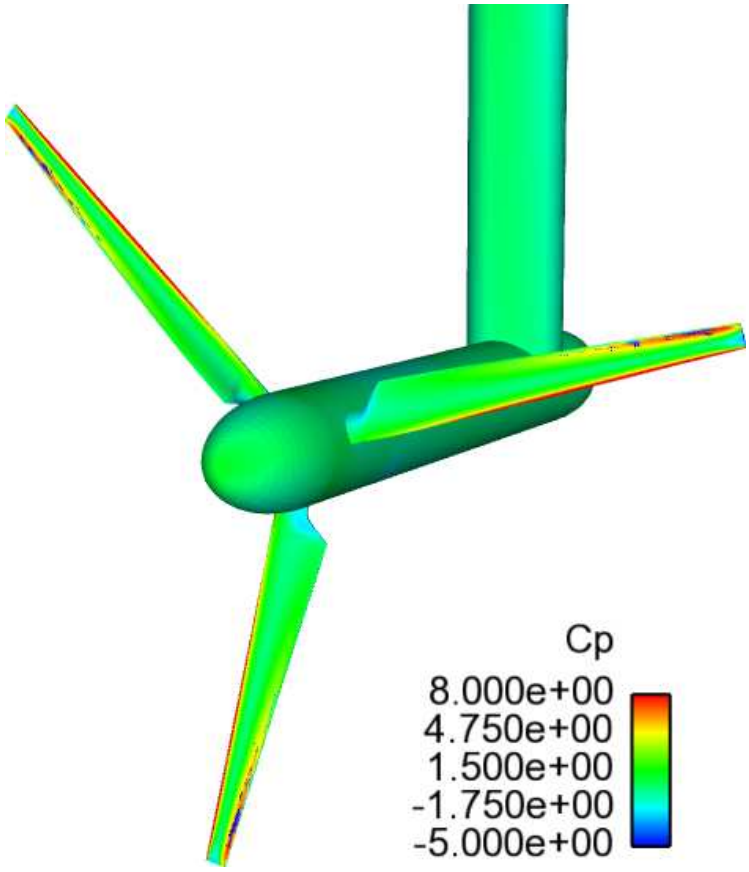



**Figure 4.7:** pressure coefficient on blade, TSR = 6, RANS simulations; normalisation based on local approach dynamic pressure, i.e. $c_p = (p - p_{ref}) / \frac{1}{2}\rho(U_0^2 + \Omega^2 r^2)$

(a) All surfaces




(b) Individual blade

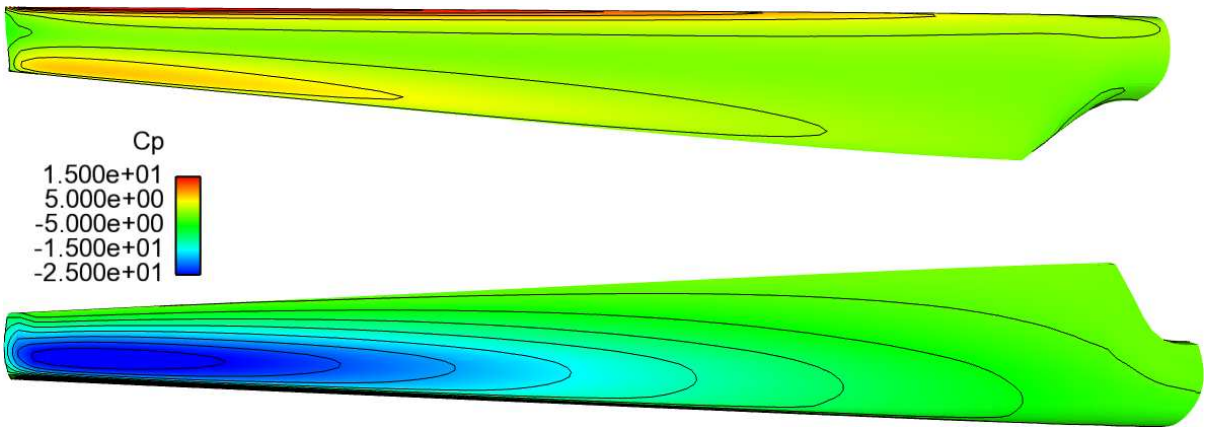



**Figure 4.8:** pressure coefficient on model surfaces (RANS simulations); normalisation based on upstream dynamic pressure, i.e. $c_p = (p - p_{ref}) / \frac{1}{2}\rho U_0^2$; scales are not intended to indicate maximum and minimum values.

Figures 4.8 and 4.9 show the vortex structure in the wake of the turbine for RANS and LES simulations. Note that blade-tip-generated vortex structures are better preserved downstream in LES than they would be by the more diffusive RANS calculation and there is a strong interaction with the support mast. Note also the well-established vortex occurring behind the latter.

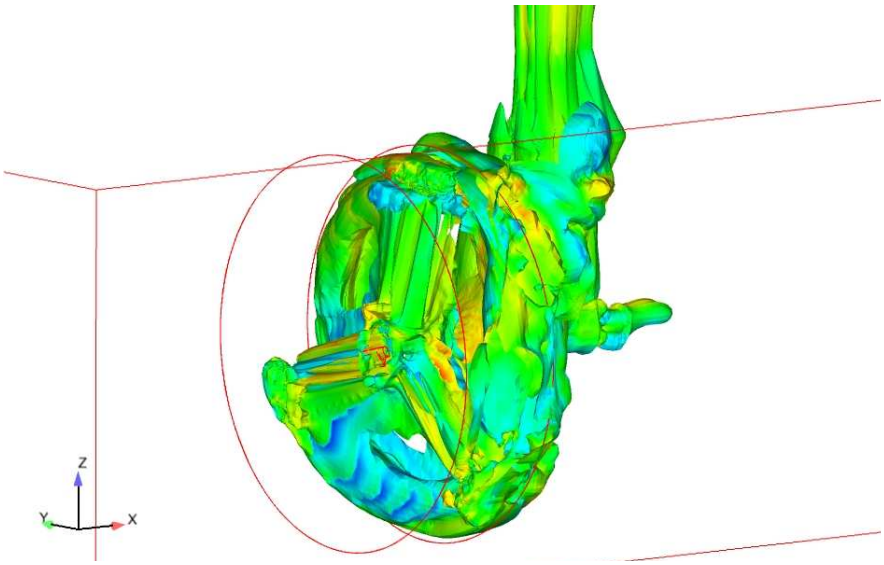



**Figure 4.8:** mean vorticity field (*k*-ε RANS)

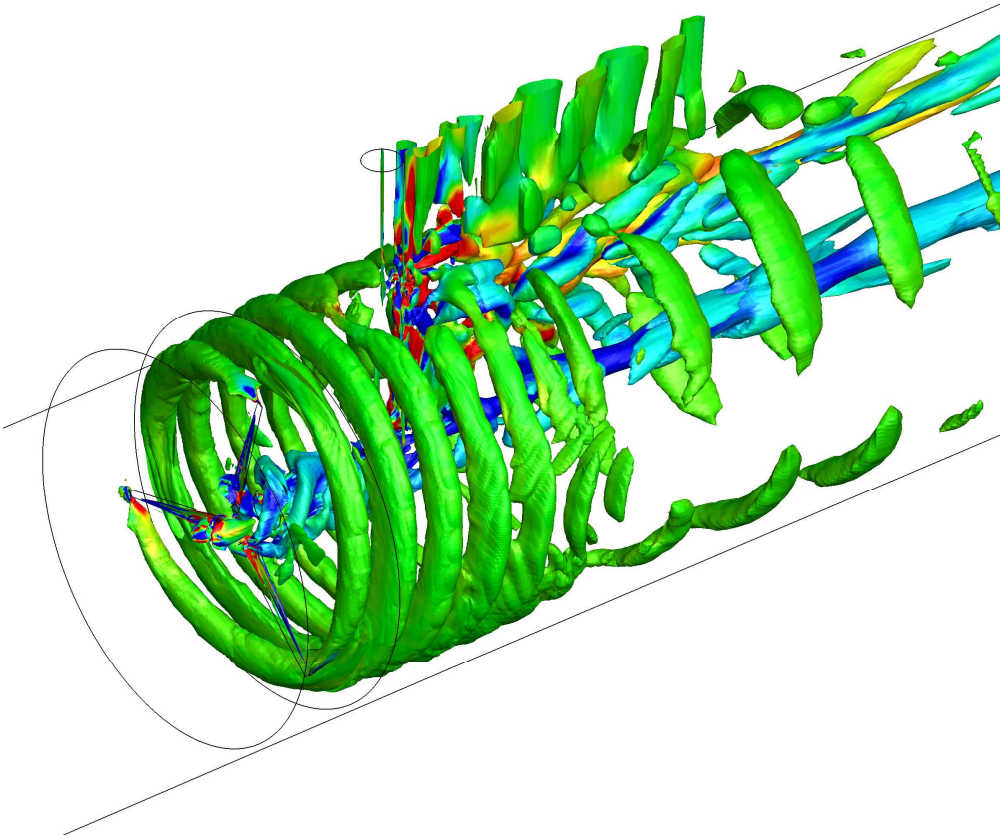


**Figure 4.9:** iso-*Q* field (LES)

## 5. NEXT STAGES

Despite a delay due to the unavailability of EdF's Blue Gene computer for a period this summer, parametric studies of the effect of operating point (TSR) and inflow turbulence models have now been performed for both RANS and LES calculations. Mean and fluctuating thrust and power coefficients have been determined as functions of time. On the meshes used RANS calculations take of the order of 1-2 days (on a high-performance computer), but LES calculations may take more than a week. It would be highly desirable to "tidy up" our results by performing more extensive grid-dependence checks.

RANS calculations are less computationally intensive and a basic $k$-$\varepsilon$ model has been shown to provide acceptable values for the thrust coefficient but a low power coefficient and considerable diffusion in the wake. Alternative RANS turbulence models available in *Code_Saturne* (including SST $k$-$\omega$) will also be investigated, as will non-uniform mean and turbulent velocity profiles. Alternative wall boundary conditions – "standard" wall functions and low-Re models – will also be investigated as part of the grid-dependence checks.

LES calculations show an improvement in predictions of the power coefficient, although still underprediction for off-design incidence. It is hoped to improve the specification of inflow turbulence – in particular, more realistic pressure fluctuations – by incorporating the work on divergence-free synthetic-eddy models currently being developed by PhD student R. Poletto. Following grid-dependence checks the dynamic LES model currently available in Code_Saturne will also be investigated.

The next deliverable (MD1.3) specifies two major new items:

- waves;
- real turbine geometry.

Both of these will be investigated in the first instance with time-dependent RANS. For the first we will be extending the work initiated by Olivier Cozzi and Yacine Addad to include free-surface movement in the outer domain. For the last item a CAD model of the TGL turbine has already been passed to us for meshing.

## REFERENCES


Archambeau, F., Mechitoua, N. and Sakiz, M., 2004, "*Code_Saturne*: a Finite-Volume Code for the Computation of Turbulent Incompressible Flows – Industrial Applications", International Journal on Finite Volumes, 1.

Bahaj, A.S., Batten, W.M.J. and McCann, G., 2007a, "Experimental verifications of numerical predictions for the hydrodynamic performance of horizontal axis marine current turbines", Renewable Energy, 32, 2479–2490.

Bahaj, A.S., Molland, A.F., Chaplin, J.R. and Batten, W.M.J., 2007b, "Power and thrust measurements of marine current turbines under various hydrodynamic flow conditions in a cavitation tunnel and a towing tank", Renewable Energy, 32, 407–426.

Batten, W.M.J., Bahaj, A.S., Molland, A.F. and Chaplin, J.R., 2008, "The prediction of the hydrodynamic performance of marine current turbines", Renewable Energy, 33, 1085–1096.

Beaudoin, M. and Jasak, H., 2008, "Development of a Generalized Grid Interface for turbomachinery simulations with OpenFOAM", Open Source CFD International Conference 2008, Berlin, Germany.

Blades, E. L. and Marcum, D. L., 2007, "A sliding interface method for unsteady unstructured flow simulations", International Journal for Numerical Methods in Fluids, 53, 507-529.

Fenwick, C. L. and Allen, C. B., 2006, "Development and validation of sliding and non-matching grid technology for control-surface representation", Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 220, 299-315.

Kim, D., 2009, "Laminar flow past a sphere rotating in the transverse direction", Journal of Mechanical Science and Technology, 23, 578-589.

Lien, F.S., Chen, W.L. and Leschziner, M.A., 1996, "A multiblock implementation of a non-orthogonal, collocated finite volume algorithm for complex turbulent flows", International Journal for Numerical Methods in Fluids, 23, 567–588.

Mittal, S. and Kumar, B., 2003, "Flow past a rotating cylinder", Journal of Fluid Mechanics, 476, 303-334.

Oesterlé, B. and Dinh, T. B. ,1998), "Experiments on the lift of a spinning sphere in a range of intermediate Reynolds numbers", Experiments in Fluids, 25, 16-22.

Petit, O., Page, M., and Beaudoin, M., 2009. The ERCOFTAC centrifugal pump OpenFOAM case-study", 3rd IAHR International Meeting of the Workgroup on Cavitation and Dynamic Problems in Hydraulic Machinery and Systems, Brno, Czech Republic.

Stansby, P.K. and Rainey, R.C.T., 2001, "A CFD study of the dynamic response of a rotating cylinder in a current", Journal of Fluids and Structures, 15, 513-521.

Steijl, R. and Barakos, G., 2008, “Sliding-mesh algorithm for CFD analysis of helicopter rotor-fuselage aerodynamics”, International Journal for Numerical Methods in Fluids, 58, 527-549.