ARCHER2-eCSE04-1 Project Report: **Green multi-rotor systems: a pre-exascale multi-code framework for rotor-resolved analysis and design of floating wind farm and multi-propeller electrical aircraft propulsion**

Wendi Liu[1], Sergio Campobasso[2,*], Alex Skillen[3], Charles Moulinec[1]

1. Scientific Computing Department, Science and Technology Facilities Council, Daresbury Laboratory, Warrington WA4 4AD, UK
2. Engineering Department, Lancaster University, Lancaster, LA1 4WR, UK
3. Department of Mechanical Aerospace and Civil Engineering, School of Engineering, The University of Manchester, UK

ABSTRACT

The project aim is to develop a pre-exascale framework for blade-geometry resolved computational fluid dynamics analyses (CFD) of open multi-rotor systems, such as floating offshore wind farms and propeller-based distributed electric propulsion systems, coupling the open-source code OpenFOAM and ARCTIC CFD codes through the Multiscale Universal Interface library. These simulations exploit the strongest points of both codes. In this project, we developed a coupled framework for both steady and unsteady state simulations with both one-to-one and one-to-many coupled instances. Test cases show smooth and continuous flow fields across the coupled boundaries. A scaling test has been performed by using the flow past four cylinders case leading to 180M cells in total. It shows good scalability when the MPI tasks go from 96 to 1,536.

Keywords: Multi-code framework; High-performance computing; Fluid-fluid coupling.

Table of contents

1. Introduction 2
2. Code coupling strategy 3
3. Newly developed functionalities for the ARCTIC-OpenFOAM coupling 7
3.1. Newly developed functionalities in ARCTIC v7 8
3.2. Newly developed functionalities in OpenFOAM v2106 11
3.3. Newly developed functionalities in MUI v1.2 11
4. Test cases on the coupled framework 11
4.1. Flow past a single cylinder at Re=40 in steady mode 11
4.2. Flow past a single cylinder at Re=40 in unsteady mode 13
4.3. Flow past a single cylinder at Re=150 14
4.4. Flow past four cylinders at Re=40 in steady mode 16
4.5. Flow past four cylinders at Re=40 in unsteady mode 18
4.6. Flow past four cylinders at Re=150 19
5. Scalability test 19

* Corresponding author. E-mail address: m.s.campobasso@lancaster.ac.uk

6. Conclusions and impact 21
Appendix I. Folder structure 22
Appendix II. Installation of framework 23
Appendix III. Workflow on running a new case by using the framework 24
Appendix IV. Test case between ARCTIC and ARCTIC domains -- flow past a flat plate 25
Appendix V. Test case between OpenFOAM and OpenFOAM domains – flow past a circular cylinder 27
Appendix VI. Modules used in ARCHER2 29
References 29

1. Introduction

The project aims to develop a scalable and accurate rotor geometry-resolved multi-code framework to improve floating offshore wind turbine (FOWT) farm and propeller-based distributed electric propulsion (DEP) [1] aerodynamic analysis and performance prediction. This is done to be able to analyse floating wind farms with hundreds of FOWTs, complete aircraft with DEP, and other complex multi-rotor systems. Such a framework is required because FOWT aerodynamics is not well understood, mainly because of their motion due to waves. The framework involves three software, namley ARCTIC, OpenFOAM and MUI. The first one, ARCTIC v7 (the incompressible version of COSA [2] used on ARCHER [3] and other PRACE tier-0/1 clusters) is a finite volume structured multi-block turbulent Navier-Stokes Fortran code that enables highly accurate predictions of rotor aerodynamics [2, 4-6] and unsteady aircraft aerodynamics [7]. The ARCTIC code uses geometric multigrid for pressure calculation to achieve convergence acceleration. The code has a steady, a time-domain and a nonlinear frequency-domain solver; its incompressible flow solver differs from the compressible one only by not solving the total energy equation, and solving the continuity equation with a generalised artificial compressibility method. The code uses some serial LAPACK routines, has a pure MPI and a hybrid MPI/OpenMP parallelisation, and parallel IO, all developed in one Hector dCSE, one ARCHER eCSE [8] and one CINCECA project. Production jobs have been/are being run on ARCHER and CINECA's GALILEO and MARCONI 100 in ARCHER RAP and HPC Europa3 projects. OpenFOAM v2106 [9] is a widely used open-source finite volume method that uses a co-located methodology on an unstructured polyhedral grid with arbitrary grid elements. The code is cell-centred and is built in a modular fashion using a wide platform of C++ libraries featuring a variety of numerical methods (space- and time-discretisation, matrix solution schemes, interpolation methods), flow physics models (e.g. compressible/incompressible models, multi-phase and reactive flows) that can be used for multi-physics simulations. OpenFOAM also features the volume of fluid method for the simulation of surface gravity waves and their interaction with floating bodies such as the platforms of floating wind turbines, one of the key features of interest in this project. Multiscale Universal Interface (MUI) v1.2 library [10] is a header-only concurrent framework for coupling heterogeneous solvers by domain decomposition. MUI is written in C++ and has wrappers for C, Fortran and Python. It has demonstrated good scalability up to 512 nodes of ARCHER [11] and allows an arbitrary number of codes to communicate with each other over MPI via a cloud of point data. The Radial Basis Function (RBF) spatial sampler of MUI is essential to ensure the conservation of all variables at the interfaces between code domains. The current MUI library offers more functionalities than

data exchange, such as several algorithms for code coupling and high-order interpolation schemes. Each of these three codes has different capabilities. The open-source computational fluid dynamics (CFD) code OpenFOAM will be used to compute the less detailed far-field flow around the FOWTs or the propellers, and one or several ARCTIC instantiations will be employed for the detailed flow field around the rotors. OpenFOAM will bring to the framework wave and wave/structure interaction (WSI) modelling for FOWT platform hydrodynamics not available in ARCTIC.

2. Code coupling strategy

Two-way fluid domain couplings between ARCTIC and OpenFOAM solvers for both steady-state and unsteady simulations have been implemented in this study. For steady-state simulation, ARCTIC employs a steady multigrid integration scheme and OpenFOAM uses muiSimpleFoam solver, which is a steady-state solver based on the SIMPLE algorithm for incompressible laminar/turbulent flows and with fluid-fluid coupling function embedded. Figure 1 shows the flow chart of the steady-state coupling between ARCTIC and OpenFOAM. There is one data exchange between ARCTIC and OpenFOAM through the MUI library per iteration (details of the data exchange strategy will be explained later). In each iteration ARCTIC and OpenFOAM send their fluid data in the area of interest that was solved by the previous iteration to the other solver. Once both ARCTIC and OpenFOAM receive the data from each other, they both update their boundary conditions according to the received data, solve the respective flow field and move to the next iteration.

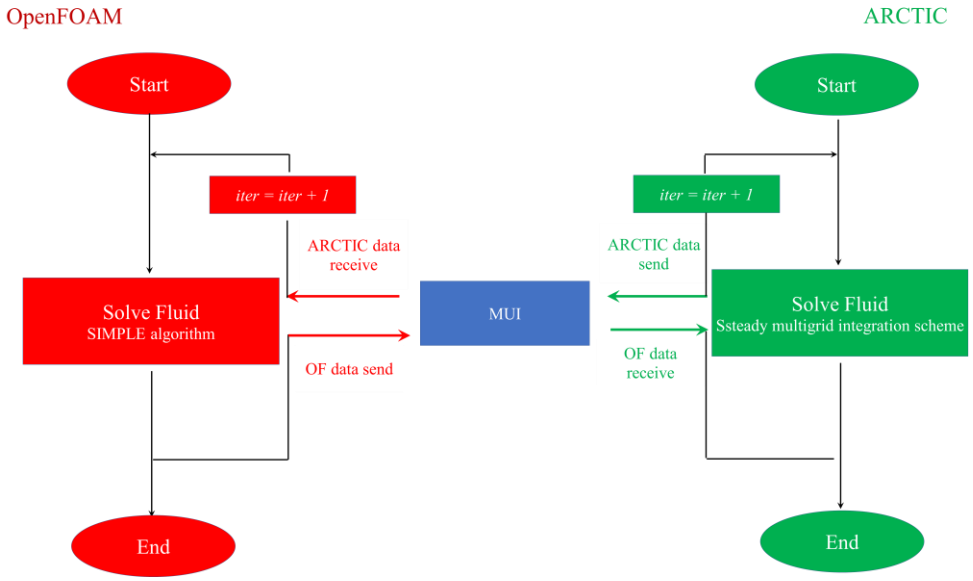

Figure 1 Flowchart on the steady-state coupling between ARCTIC and OpenFOAM

For unsteady simulations, ARCTIC employs dual time stepping with Runge-Kutta integration scheme and OpenFOAM uses muiPimpleFoam solver, which is a transient solver based on merged PISO-SIMPLE algorithm for incompressible laminar/turbulent flows and with fluid-fluid coupling function embedded. Figure 2 shows the flow chart of the unsteady coupling between ARCTIC and OpenFOAM. In this coupling framework, we implemented a sub-iteration loop within each time step, but outside the PIMPLE loop, to the OpenFOAM muiPimpleFoam solver. There is one data exchange between ARCTIC and OpenFOAM through the MUI library per sub-iteration. At each sub-iteration (assuming it is not the last sub-iteration of a given time step), OpenFOAM exchanges data with ARCTIC, updates the boundary value according to the received data, conducts the PIMPLE loop to update the fluid

field with the updated boundary value and then moves to the next sub-iteration without time marching. A similar procedure is also conducted by ARCTIC at each sub-iteration. In the present implementation, the number of sub-iterations value "muiTotalSubIteration" in OpenFOAM muiPimpleFoam solver should be set as the same value as the "lmax" value in ARCTIC, so that they can keep the same pace in time stepping and move to the next time step at the same time.

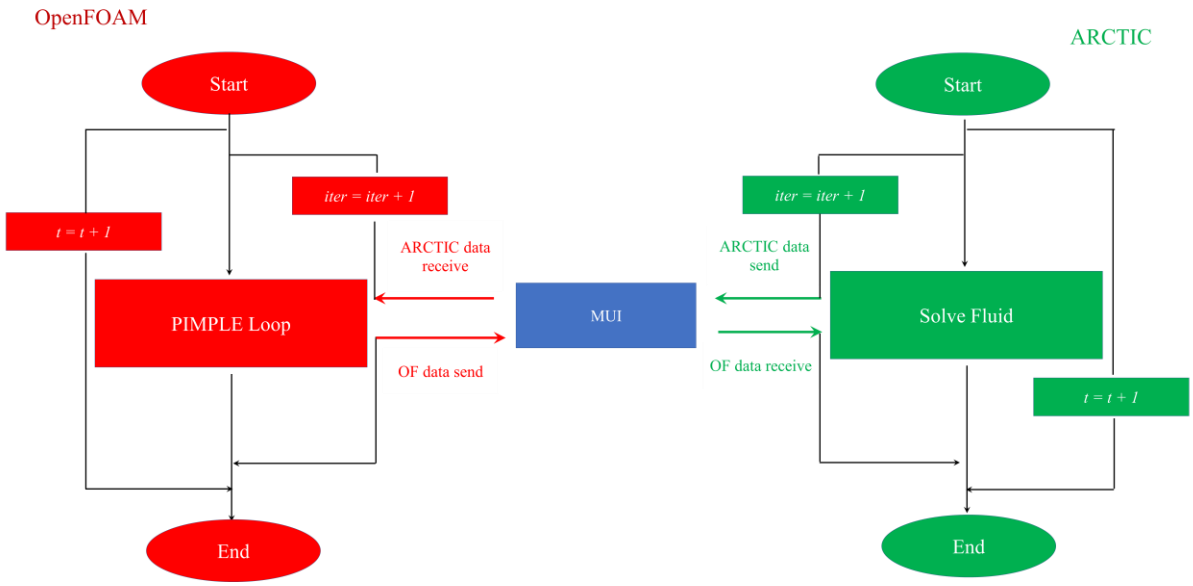

Figure 2 Flowchart on the unsteady coupling between ARCTIC and OpenFOAM

The way the data are exchanged between ARCTIC and OpenFOAM as well as how their respective boundaries are updated accordingly to the received data are important questions for the success of this multi-solver fluid-fluid coupling. ARCTIC and OpenFOAM have different boundary treatment methods. The ARCTIC solver employs a ghost cell approach at boundaries. There are two layers of auxiliary cells outside of the boundary and the fluxes at the boundary are calculated by 2nd order central differencing with cell centre values of these two layers of auxiliary cells as well as the cell centre values of two layers of internal cells. Instead, for the OpenFOAM solver, there is no ghost cell beyond the boundary. The boundary fluxes are determined directly by the boundary condition. Figure 3 shows the schematic plot of data exchange between ARCTIC and OpenFOAM. The fluid quantities that need to be exchanged between ARCTIC and OpenFOAM are the pressure, three components of velocity and turbulence quantities if a turbulent model is switched on. The shape of the OpenFOAM coupled boundary and the shape of the ARCTIC coupled boundary (boundary type BC=32) are expected to overlap with each other, i.e. these two coupled boundaries are representations of the same physical space in different solvers. The OpenFOAM mesh and ARCTIC mesh do not have to be conformal. For the data flow from OpenFOAM to ARCTIC, the MUI library searches for the cells in the OpenFOAM domain that are located near the ARCTIC auxiliary cells. For each ARCTIC auxiliary cell, MUI identifies the two nearest cell centres in the OpenFOAM domain and uses a weighted average method to these two nearest cell centres to determine the value of a certain fluid quantity for this ARCTIC auxiliary cell. Once every ARCTIC auxiliary cell of coupled boundary receives the data from OpenFOAM, ARCTIC calculates the fluid field with the updated auxiliary cell values. For the data flow from ARCTIC to OpenFOAM, the MUI library searches for the face nodes of the coupled boundary in the ARCTIC domain that is located near the OpenFOAM coupled boundary face centres. For each face centre of the OpenFOAM coupled boundary, MUI identifies two nearest face nodes in the ARCTIC domain and uses the same weighted average method as

above for these two nearest face nodes to determine the value of a certain fluid quantity for this OpenFOAM face centre. Once every OpenFOAM face centre of the coupled boundary received the data from ARCTIC, OpenFOAM calculates the fluid field with the updated boundary values.

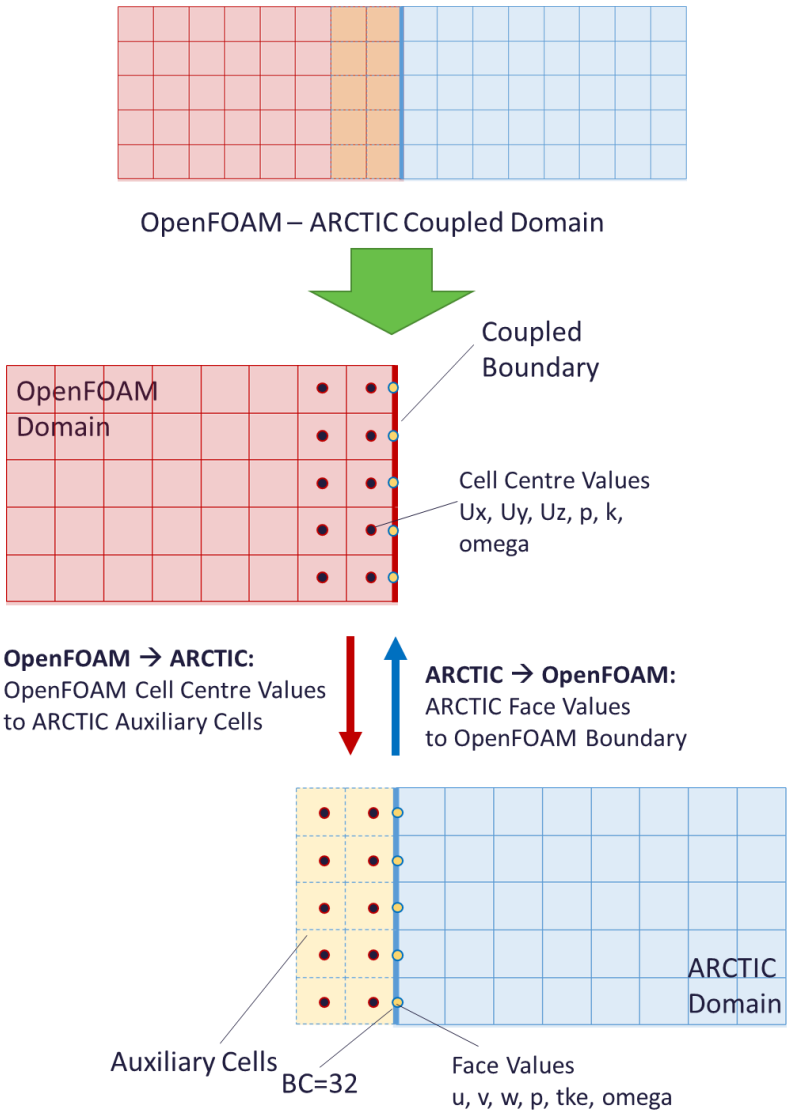

Figure 3 Schematic plot on data exchange between ARCTIC and OpenFOAM

ARCTIC and OpenFOAM handle differently the received fluid quantity after the data exchange. For OpenFOAM, each coupled boundary face centre collects data from the peer solver (ARCTIC in this case) through MUI "fetch" function. As shown in Figure 4, once OpenFOAM receive values of face centres from ARCTIC, these values are assigned to corresponding face centres (C_f in Figure 4) directly. OpenFOAM will calculate the cell centre values (C_c in Figure 4) with this boundary face value.

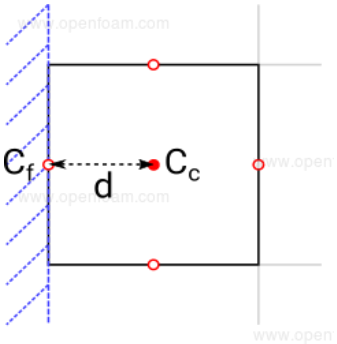

Figure 4 OpenFOAM evaluation scheme [12]

For ARCTIC, as shown in Figure 5, fluid data are collected from the peer solver (OpenFOAM in this case) through the MUI "fetch" function. These fluid data are stored temporarily by newly defined variables "qfetch" and "mutfetch". Here, "temporary" means that the data stored in "qfetch" and "mutfetch" are only valid for the present iteration. For a new iteration, the MUI "fetch" function should be called to receive a new set of data for "qfetch" and "mutfetch". This data collection is conducted at the very beginning of each iteration so that the data will be available for the remaining activities of the ARCTIC solver during the iteration,

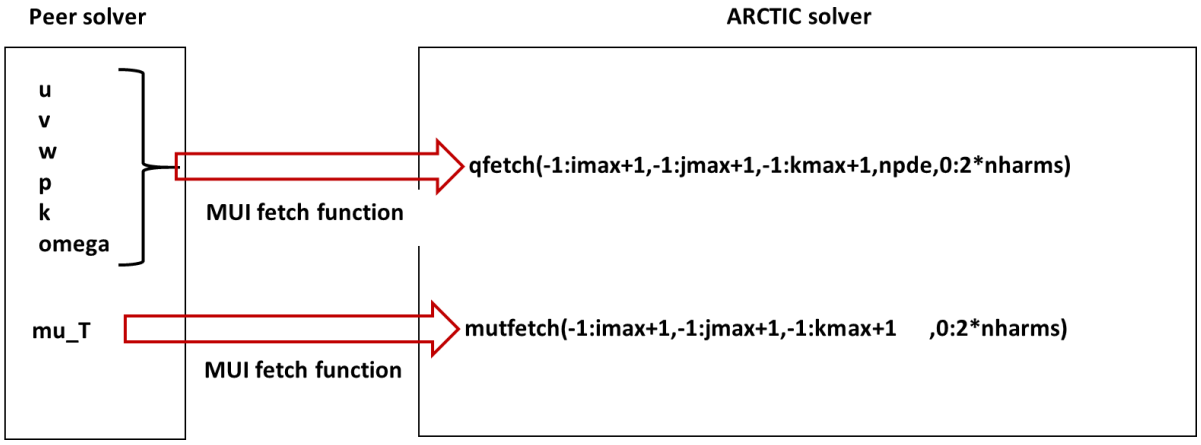

Figure 5 Diagram of ARCTIC collecting data from the other solver (peer solver) through the MUI library

During the steady multigrid integration scheme or dual time stepping with the Runge-Kutta integration scheme, boundary condition-related subroutines are called a number of times for synchronisation. If coupled boundary (BC=32) is defined, the corresponding subroutine is called to assign values to the auxiliary cells of the coupled boundary. Figure 6 shows the diagram on ARCTIC assigned the collected data through the coupled boundary subroutine. What is done by the coupled boundary subroutine is to read data from "qfetch" and "mutfetch" and assign the read data to "q" and "mut", respectively.

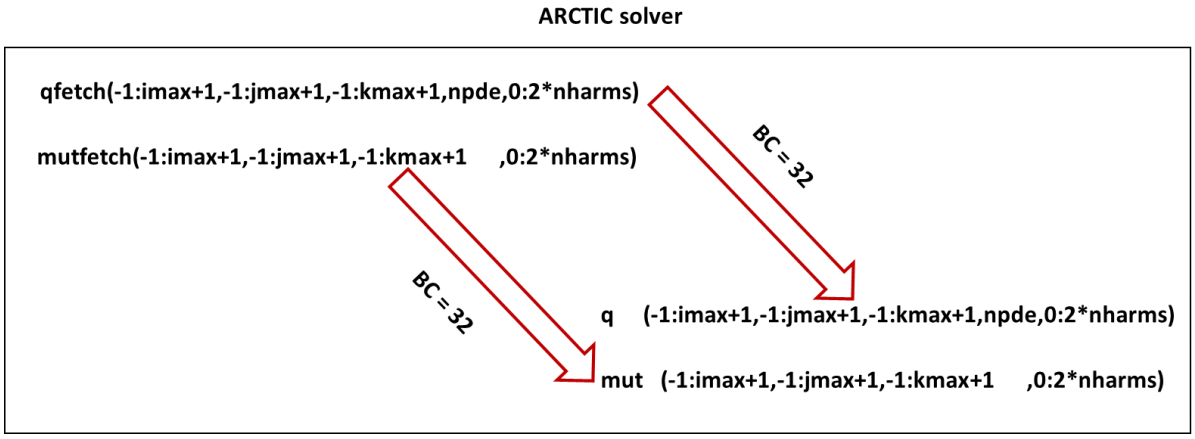

Figure 6 Diagram on ARCTIC assigned the collected data through BC=32

We have considered combining the data collection activity into this coupled boundary subroutine, so that the ARCTIC solver collects fluid data by using MUI "fetch" function and is directly assigned to "q" and "mut". However, as mentioned, the boundary condition related subroutines is called a number of times during one iteration. If these two actions are combined, MUI will search, collect and transfer massive data from OpenFOAM to ARCTIC a number of times within one iteration. That will increase MPI communication overhead significantly. Moreover, the data collected from OpenFOAM are not expected to change within one iteration. Thus, to communicate only once at the very beginning and store it in memory at each iteration is a better choice.

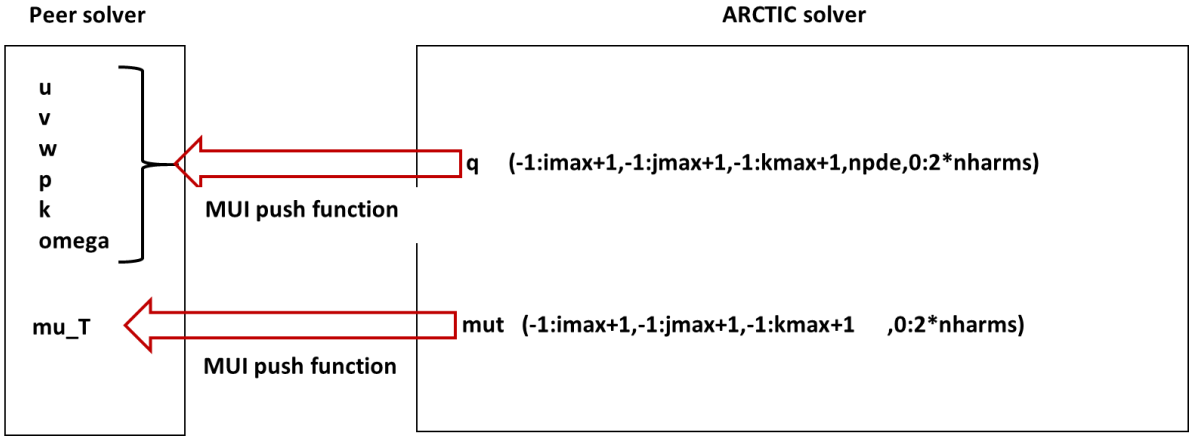

Figure 7 Diagram of ARCTIC sending data to the other solver (peer solver) through the MUI library. It read data from "q" and "mut" and sends it to the peer solver

For the data collection in the ARCTIC solver, as shown in Figure 7, MUI "push" function will be called to collect values from "q" and "mut" and send them to the peer solver.

3. Newly developed functionalities for the ARCTIC-OpenFOAM coupling

Several new functions have been implemented in ARCTIC, OpenFOAM as well as in the MUI coupling library to enable the coupled simulation. In this section, new functions that have been developed in this project are reviewed.

3.1. Newly developed functionalities in ARCTIC v7

3.1.1. Coordinates of auxiliary cells

As indicated in Section 2, ARCTIC employs two layers of auxiliary cells for each block boundary. MUI searches for the two nearest cell centres in the OpenFOAM domain according to the location of auxiliary cells of a coupled boundary. This requires each auxiliary cell of a coupled boundary to have its coordinate information available, which was not the case before this study as auxiliary cells where just a memory without any coordinate information.

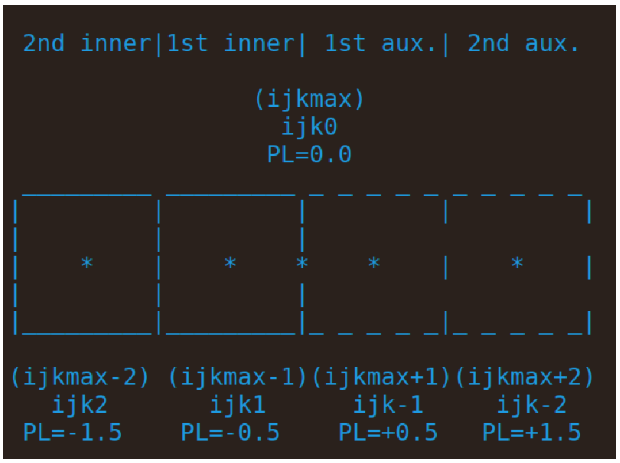

Figure 8 Schematic plot on coordinates of auxiliary cells

We have implemented the definition of cell centre coordinates of auxiliary cells as shown in Figure 8 and employed the linear extrapolation method to use the coordinate of the nearest two internal cell centres to get the coordinates of the two auxiliary cells. In Figure 8, "PL" means Pseudo-Location, which is the weight of each cell centre on the calculation of extrapolation. The Fortran function "Case Select" is used to define more extrapolation methods. It is extendible and easy to maintain.

3.1.2. Coupled boundary (BC=32)

As described in Section 2, the coupled boundary type with the mask number of 32, which assign values received from the peer programme, is implemented. The flag "-DMUI" is needed during the compilation stage to enable this boundary type. Otherwise, an error message will be given.

3.1.3. MUI_ARCTIC module

A new module named "MUI_ARCTIC" is defined to support the MUI-related functions within the ARCTIC solver. It contains the subroutine MUI_INT(), which is a collection of necessary MUI-related initialisation functions. It is important to call it at the beginning of the execution. It also contains some customised subroutines for ARCTIC to make it easier the use of MUI-related functions. An example is for the READ_MUI_CONTROL_FILE() subroutine, which allows users to define coupling-related variables from a standalone input file named "mui_input.dat".

3.1.4. MUI Push/commit/fetch

These are functions related to MUI sending and receiving data within ARCTIC. They are defined in the module of "parallelutils", and are called in the subroutine "smooth" of the main file (arctic.f).

3.1.5. Utility functions

Some utility functions have been defined in util.f to make the code easy to maintain. An example is the conversion functions of "q" and "mut". These functions are needed because of the different dimensions of the "q" value and "x", "y" and "z" values, especially in the auxiliary cells. By defining these functions, MUI send/receive as well as the data assignment for the coupled boundary are much easily handled.

3.1.6. MUI smart send/receive functions

MUI smart send/ receive functions are efficient ways to reduce communication overhead by avoiding all-to-all communications. The following two functions, i.e. "MUI_ANNOUNCE_SEND_SPAN()" and "MUI_ANNOUNCE_RECV_SPAN()" are defined in the module of "parallelutils", and are called in the main file (arctic.f). For these functions, users need to give the value of "send_shape" and corresponding values of the dimensions of the shape in the "mui_input.dat". MUI smart send function only searches for nodes inside the defined shape to reduce the search/communication overhead. The available send shapes that have been implemented are box, hollow_box, sphere, hollow_sphere, cylinder and hollow_cylinder. Users can pick the most suitable shape according to the case and define the shape variables (such as the centre coordinate and radius of a sphere). For smart receive, users do not have to give any input value before the simulation. The smart receive function will only look for the auxiliary cells of coupled boundary, which do not need any active definitions from users.

3.1.7. Makefile: make on Ubuntu and ARCHER2; make with MUI embedded

We have updated the Makefile so that it can be compiled on ARCHER2 and Ubuntu OS. The list of modules of ARCHER2 running can be found in Appendix VI. The compilers used in Ubuntu OS are GNU Fortran 9.4.0 and Open MPI 4.0.3. To compile in ARCHER2 with parallel mode, the user can run the command "make mpi HOST=archer2", or compile in Ubuntu OS by running "make mpi HOST=ubuntu". We have also defined flags to make it easy to switch on/off the MUI-related functions. These flags have been packaged in the Makefile file. Users can simply use "make mpi MUI=1" compile ARCTIC with coupled mode and enable all MUI-related coupled functions. If no coupling is asked for by the simulation, users can use "make mpi MUI=0" to compile ARCTIC as a standalone solver, or just simply use "make mpi" since MUI-related functions are switched off by default.

3.1.8. Dimension conversion

ARCTIC uses normalised values for the calculation, while OpenFOAM uses dimensional values. There is a need to conduct dimension conversion before data exchange. In this study, we convert all values into dimensioned values before send to the peer solver. Dimension conversion functions are implemented in ARCTIC to facilitate these activities. They are embedded in the subroutines of sending and receiving data from OpenFOAM via MUI. Before sending off the ARCTIC values, dimension conversion is carried out to all fluid quantities to obtain the dimensioned value. After receiving the OpenFOAM data, ARCTIC immediately normalises the dimensionalised value before storing them. Table 1 shows an example of non-dimensionalisation by ARCTIC solver for the case of the flow past a flat plate. Users can find the non-dimensionalisation variables in the "mui_input.dat", such as l_{ref}, P_{ref}, U_{ref}, ρ_{fluid}, etc.

Table 1 Example of non-dimensionalisation by ARCTIC solver on the flow past flat plate case

Property	Dimension				Nondimensionalisation of ARCTIC[†]			
	Symbol	Expression	Value[‡]	Unit	Symbol	Expression	Value[§]	Unit
Density	ρ_{fluid}	–	1000	$\frac{kg}{m^3}$	ρ'	$\rho' = \frac{\rho}{\rho_{fluid}}$	1	–
Static pressure	P_∞	–	1	$\frac{kg}{m \cdot s^2}$	P'	$P' = \frac{P}{P_{ref}} = \frac{P}{P_\infty}$	1	–
Free stream velocity	U_∞	–	1	$\frac{m}{s}$	U'	$U' = \frac{U}{U_{ref}} = \frac{U}{U_\infty}$	1	–
Fluid kinematic viscosity	ν_L	–	10^{-6}	$\frac{m^2}{s}$	ν_{rat}	$\nu_{rat} = \frac{\nu_T}{\nu_L}$	0.1	–
Turbulence kinematic viscosity	ν_T	–	10^{-7}	$\frac{m^2}{s}$				
Reynolds Number	–	–	–	–	Re	$Re = \frac{U_\infty l_{ref}}{\nu_L}$	10^6	–
Turbulence kinetic energy	k	$k = \frac{3}{2}U_\infty^2 I^2 \approx U_\infty^2 I^2$	10^{-6}	$\frac{m^2}{s^2}$	k'	$k' = \frac{k}{k_{ref}} = \frac{k}{U_{ref}^2} = \frac{\frac{3}{2}U_\infty^2 I^2}{U_{ref}^2} \approx I^2$	10^{-6}	–
Turbulence specific rate of dissipation	ω	$\omega = \frac{k}{\nu_T}$	10	$\frac{1}{s}$	ω'	$\omega' = \frac{\omega}{\omega_{ref}} = \frac{\omega\, l_{ref}}{U_{ref}} = \frac{k\, l_{ref}}{\nu_T\, U_{ref}} = \frac{k\, Re\, \nu_L}{\nu_T\, U_{ref}^2} = \frac{k\, Re}{\nu_{rat}\, U_{ref}^2} \approx \frac{I^2\, Re}{\nu_{rat}}$	10	–

[†] OpenFOAM incompressible solvers use dimension values except for pressure.
[‡] The dimensional values on the farfield/inlet boundary of OpenFOAM solver for the flow past flat plate case.
[§] The non-dimensional values on the farfield/inlet boundary of ARCTIC solver for the flow past flat plate case.

3.2. Newly developed functionalities in OpenFOAM v2106

3.2.1. MUI embedded OpenFOAM v2106

We embedded MUI v1.2 into OpenFOAM v2106 for the coupling between OpenFOAM and ARCTIC at the beginning of this project, so that to establish a foundation for the coupling.

3.2.2. Fluid-fluid coupling functions

We have written new MUI coupling initialising functions "muiPushFetchInit.H" and MUI coupling function "muiPushFetch.H" to enable data to be sent to the peer solver, for data to be received from the peer solver and to assign received data to coupled boundaries. Users can define coupling-related variables in the input file of "muiFluidCouplingDict".

3.2.3. muiSimpleFoam

We have created a steady-state solver based on the SIMPLE algorithm for incompressible laminar or turbulent flows and with the fluid-fluid coupling function embedded. This solver is used for steady-state coupled simulations with ARCTIC.

3.2.4. muiPimpleFoam

We have created a transient solver based on the merged PISO-SIMPLE algorithm for incompressible laminar or turbulent flows with the fluid-fluid coupling function embedded. This solver is used for unsteady coupled simulations with ARCTIC.

3.3. Newly developed functionalities in MUI v1.2

3.3.1. Fortran wrapper for multi-domain unifaces

In order to enable one OpenFOAM solver coupled with many ARCTIC solvers for potential FOWT farm simulations, we have implemented multi-domain interfaces function for the MUI Fortran wrapper. This implementation has been pushed back to the MUI main repository in GitHub and will be available in the next MUI release.

4. Test cases on the coupled framework

A number of test cases towards the simulation of FOWTs and propeller-based DEP have been performed.

4.1. Flow past a single cylinder at Re=40 in steady mode

We have started our test cases with flow past a stationary circular cylinder with the Reynolds number equals to 40 based on the far-field velocity and the cylinder diameter. A steady-state simulation is carried out with OpenFOAM muiSimpleFoam solver and ARCTIC "flow-mode" set to steady.

This test case is composed of two domains. The OpenFOAM domain (shown as the red grid in Figure 9) has outer circular boundaries with a radius of 20m and an inner coupling boundary with a radius of 1.235m. The outer boundary of the OpenFOAM domain consists of two boundary patches, i.e. the inlet boundary located in the second and the third quadrant, and the outlet boundary located in the first and the fourth quadrant. Each boundary contains 96 cells along the circumferential direction. There are 96 cells along the radial direction of the OpenFOAM domain as well, but with a progression ratio of 0.9746 to allow the same cell height as that of the ARCTIC mesh at the coupling boundary. The front and back boundaries of the OpenFOAM domain are set as symmetry boundary conditions. The ARCTIC domain

(shown as black grids in Figure 9) contains four blocks. Each block occupies a quarter of the cylinder with 48 cells along the circumferential direction and 32 cells along the radial direction. The outer circular coupled boundary has a radius of 1.235m and the inner wall boundary has a radius of 0.5m. Both of these two domains extend from 0m to 0.1m with 8 cells along the z-axis direction. The total number of cells for the OpenFOAM domain is 147,456 and the total number of cells for the ARCTIC domain is 49,152.

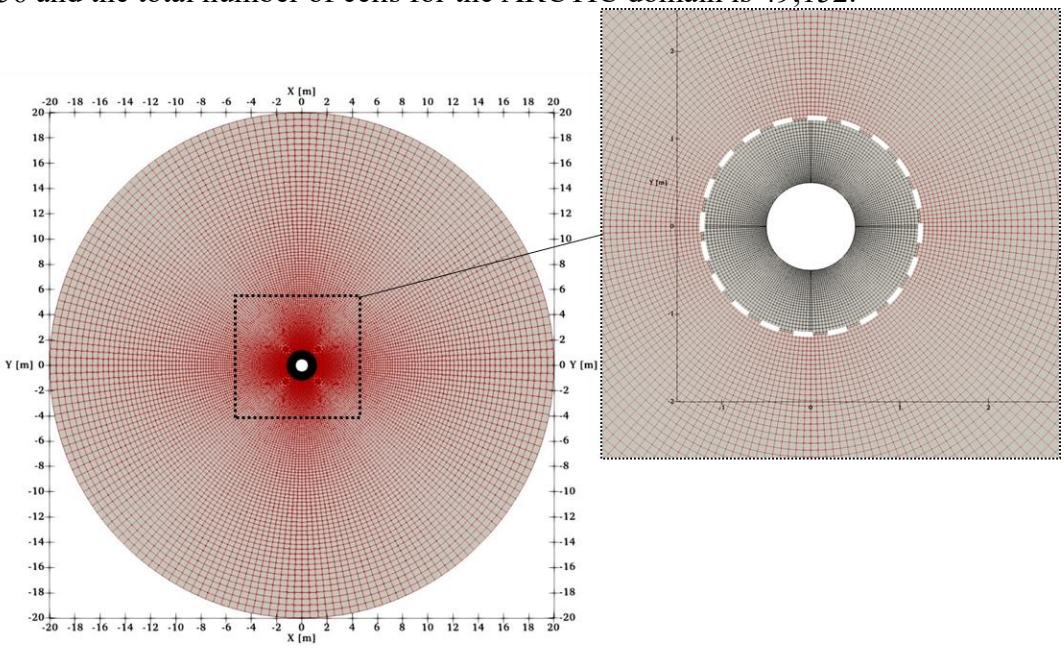

Figure 9 Coupled meshes for single cylinder case. The red grid is for the OpenFOAM domain and the black grid is for the ARCTIC domain. The white dashed circle indicates the coupling boundary between OpenFOAM and ARCTIC domains

The simulation is based on non-dimensioned values, i.e. on the OpenFOAM side, the variable is normalised based on the ARCTIC normalisation. Non-dimensional reference values are all set to 1.0 so that no dimension conversions are needed for this case. For the OpenFOAM domain with muiSimpleFoam solver, the inlet boundary has zero gradient pressure and constant uniform velocity along the x-axis direction with a magnitude of 1.0 m/s. The outlet boundary has a constant pressure of 1.0 m^2/s^{-2} (the pressure in OpenFOAM is normalised by fluid density) and a zero gradient velocity. The coupling interface is set as fixed values for both pressure and velocity. These values will be overwritten by the MUI received values from ARCTIC, and the values written in the input files are just placeholders without any affection for the coupling and simulation. The kinematic viscosity is set as 0.025 m^2/s, so that to ensure the Reynolds number of this case is 40, which is based on the diameter of the cylinder (i.e. 1.0 m). Turbulence models were switched off for this simulation. The total iteration number is set to 20,000. For the ARCTIC domain, coupled condition (BC=32) is set for the outer boundary, and the non-slip wall condition (BC=1500) is set for the inner cylinder. The Reynolds number "reyno" is set to 40. Reference values "qinf" and "pinf" are all set as 1.0. The velocity direction "alpha" is set to 0.0. The CFL numbers "cfl" and "cfli" are set as 2.0 and 4.0. The total iteration number "lmax" is set to 20,000, which is the same as the OpenFOAM side. Tolerance is set as 10^{-120} so that to ensure ARCTIC always loop over the iteration numbers set in "lmax". By doing so, both ARCTIC and OpenFOAM will always keep the same pace on iterations and time marching.

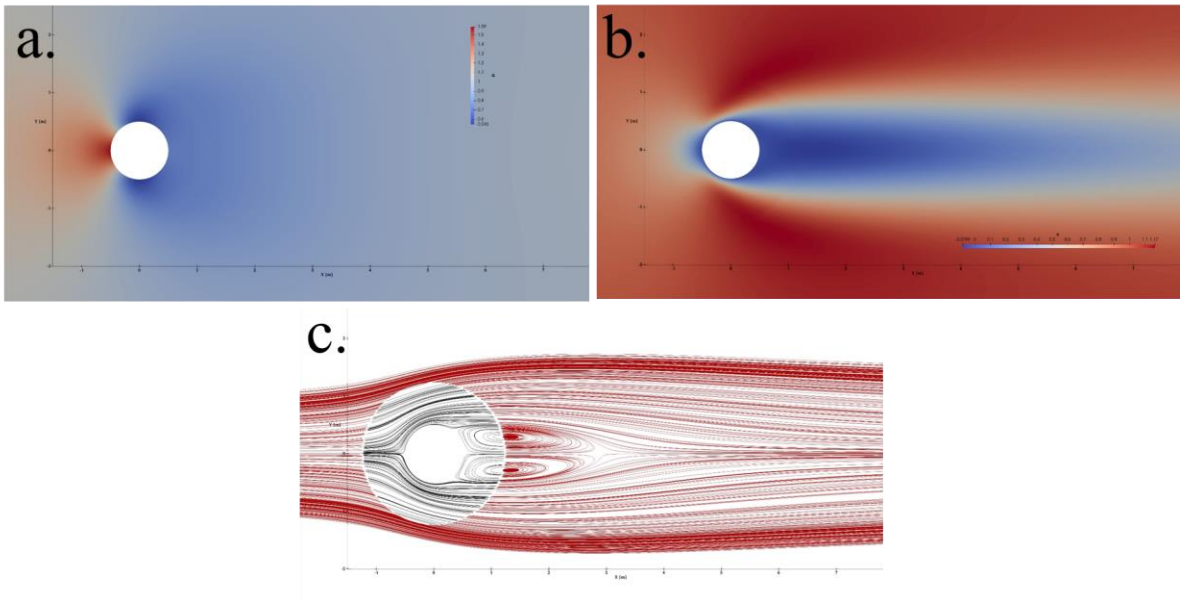

Figure 10 *Re*=40 steady-state simulation on single cylinder case with a): pressure contour, b): velocity contour and c): streamline that red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

Simulation results are shown in Figure 10. Both pressure and velocity contours show smooth and continuous fluid quantities across the coupled boundary. The streamlines are plotted using ParaView with different seeds for different domains, which is equivalent to plotting separately. There are two possibilities on why it cannot be plotted as continuous streamlines across the coupled boundary. First, there are small gaps between the ARCTIC and OpenFOAM grids, which is normal as the boundary faces are shifted by half a cell width due to the different I/O scheme between the two codes, i.e. OpenFOAM export cell centre values and ARCTIC export cell vertex values for post-processing. The other possibility is that the plotted grids are circumferentially discontinuous at the interdomain interface, again unavoidable. It is the postprocessing issue rather than the coupled simulation. Although the streamlines do not look not continuous at the coupled interface, we still can see a pair of vortices generated at the wake of the cylinder across the coupled boundary, which shows the success of multi-code fluid coupling.

4.2. Flow past a single cylinder at Re=40 in unsteady mode

We have also simulated the same configuration with the same Reynolds number, mesh and physical setup as in Section 4.1, but under unsteady conditions. OpenFOAM unsteady solver muiPimpleFoam is employed. ARCTIC "flow-mode" is set to unsteady. Both the ARCTIC and the OpenFOAM domains use the results from Section 4.1 as the starting point for simulation.

The time step size for both domains is set to 0.1 s and the simulation last for 1,000 time steps. There are 50 sub-iterations within each time step and the data exchanges between the two domains are performed at the beginning of each sub-iteration. As mentioned in Section 2, ARCTIC employs dual time stepping with Runge-Kutta integration scheme. There will be about 50 sub-iterations within each time step for ARCTIC only (without any coupling) single domain simulations for this case. The overhead for a coupled case will be the simulation time from the other solver (OpenFOAM solver in this case) and the communication between the two solvers. At the moment, a staggering coupling scheme is implemented. In the future,

once a parallel coupling scheme is implemented (i.e. both ARCTIC and OpenFOAM can solve the present iteration at the same time), the overhead will be reduced significantly.

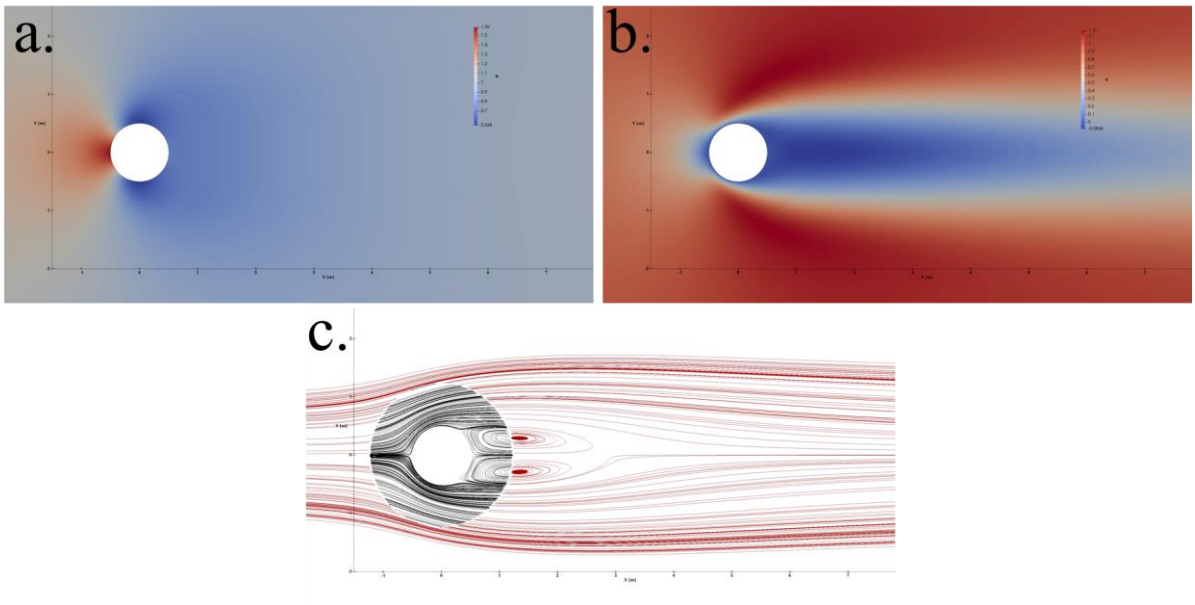

Figure 11 *Re*=40 unsteady simulation on single cylinder case with a): pressure contour, b): velocity contour and c): streamline that red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

Simulation results with 1,000 time steps are shown in Figure 11. Like the results shown in Section 4.1, smooth and continuous contours are obtained and a pair of vortices are generated at the wake of the cylinder across the coupled interface. These show that the unsteady coupling between the two fluid domains is working for a steady-state configuration.

4.3. Flow past a single cylinder at Re=150

We have further increased the Reynolds number to 150 to show the unsteady vortex shedding generated from one domain and past to the other. We keep the same mesh and physical/numerical setup as in Section 4.2, but change the kinematic viscosity to 0.00667 m^2/s. OpenFOAM unsteady solver muiPimpleFoam is employed. ARCTIC "flow-mode" is set to unsteady. Both the ARCTIC and the OpenFOAM domains use the results from Section 4.1 as the starting point to perform a steady-state simulation with *Re*=150 for 20,000 iterations. Once the steady-state simulation with *Re*=150 is done, the results from both domains are used as the starting point for the unsteady simulation with *Re*=150.
The time step size for both domains is set to 0.025 *s* and the simulation lasts for 2,500 time steps. There are 50 sub-iterations within each time step and the data exchanges between the two domains are performed at the beginning of each sub-iteration.

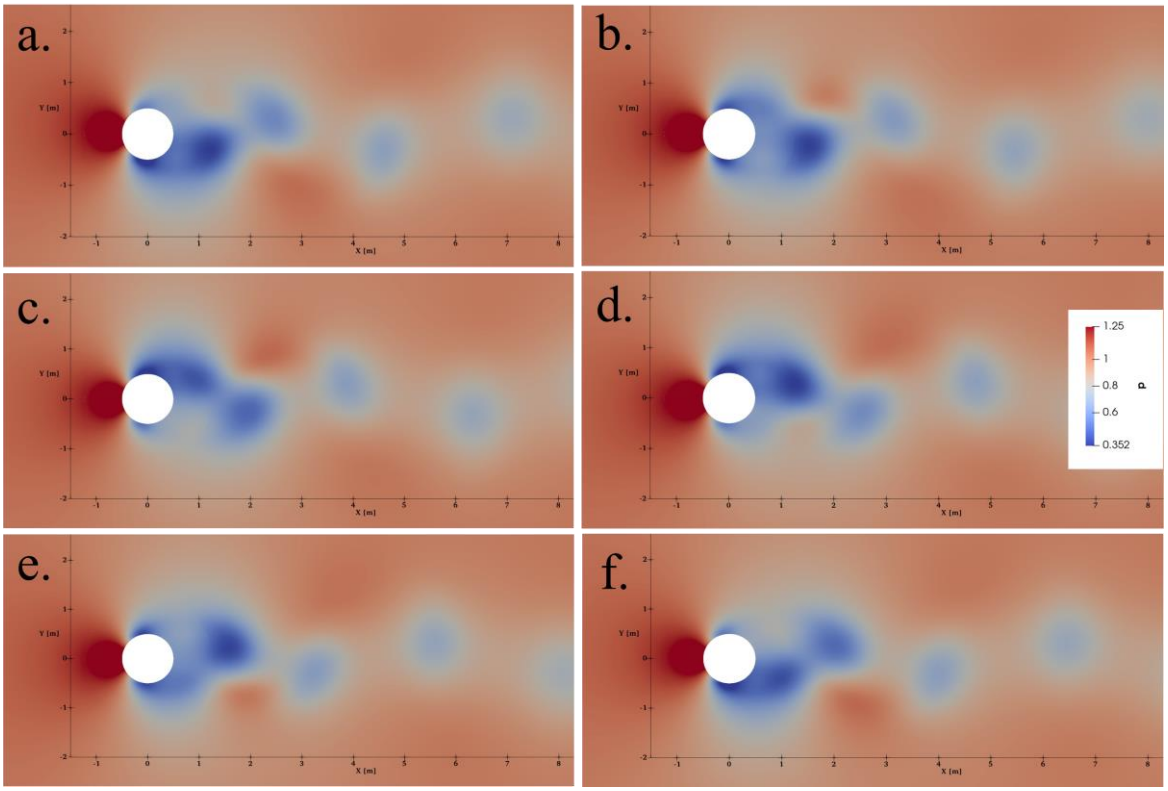

Figure 12 Pressure contours on *Re*=150 unsteady simulation on single cylinder case for a): *t*=55 s, b): *t*=56 s, c): *t*=57 s, d): *t*=58 s, e): *t*=59 s and f): *t*=60 s

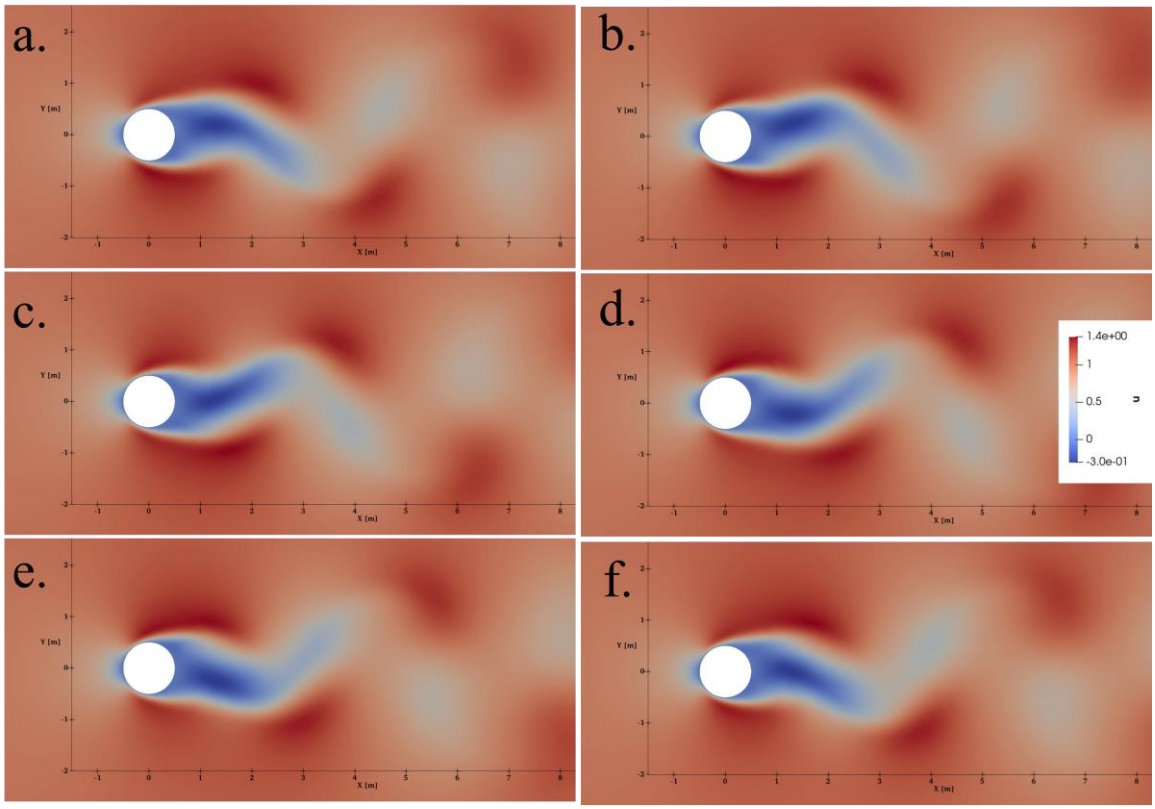

Figure 13 Velocity contours on *Re*=150 unsteady simulation on single cylinder case for a): *t*=55 s, b): *t*=56 s, c): *t*=57 s, d): *t*=58 s, e): *t*=59 s and f): *t*=60 s

Pressure contours at different time instances are shown in Figure 12. Separation points and low-pressure regions can be seen to be transmitted downstream across the coupling interface at the cylinder wake. Similar observations can be found in Figure 13. All of these plots show smooth and continuous contours across the coupling interface.

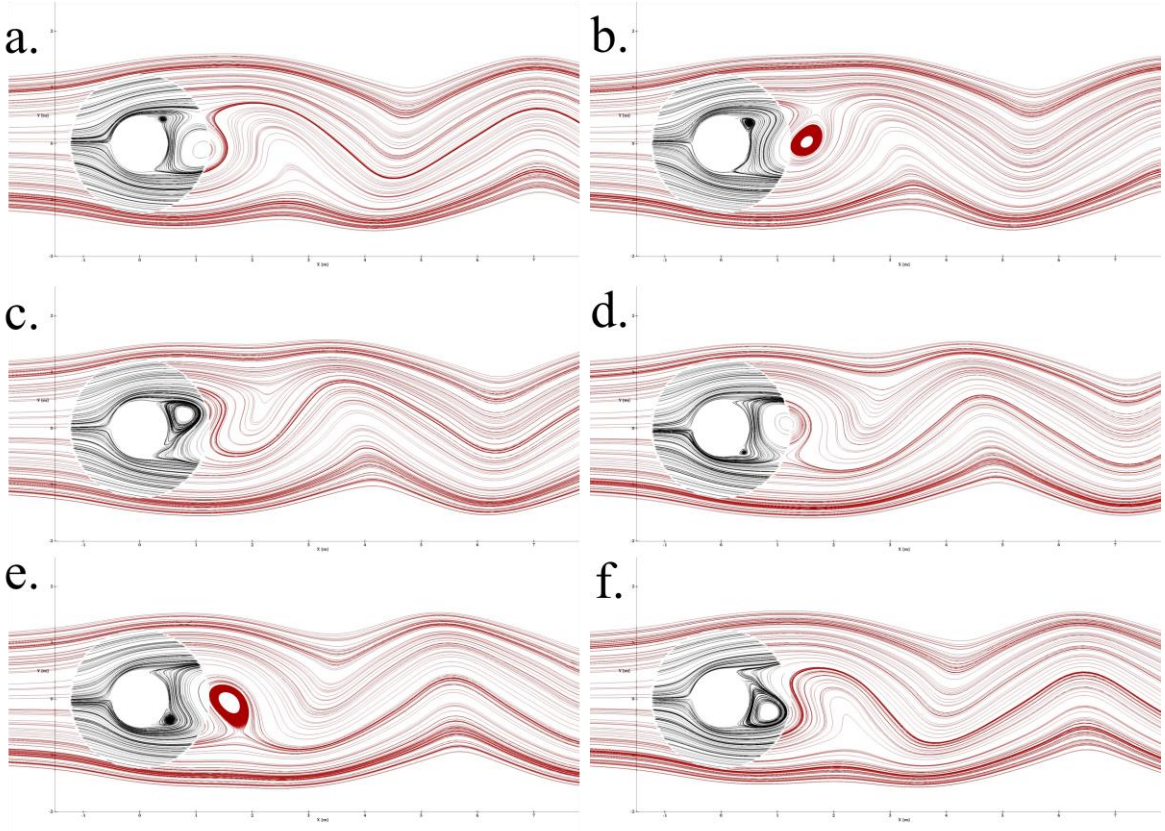

Figure 14 Streamlines on *Re*=150 unsteady simulation on single cylinder case for a): *t*=55 s, b): *t*=56 s, c): *t*=57 s, d): *t*=58 s, e): *t*=59 s and f): *t*=60 s. Red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

Streamlines at different time instances are shown in Figure 14. It is clear that the vortex generated inside the ARCTIC domain, goes across the coupling interface and is transmitted downstream in the OpenFOAM domain.

4.4. Flow past four cylinders at Re=40 in steady mode

We then further increased the number of cylinders to four. In this test case, one OpenFOAM domain is coupled with four ARCTIC domains, which are composed of five domains in total. The OpenFOAM domain (shown as red grids in Figure 15) has outer boundaries that extend from -102 m to 102 m in both *x*-axis and *y*-axis directions. The OpenFOAM domain is meshed using both structured hexahedron and unstructured pentahedron cells. The outer boundary of the OpenFOAM domain consists of four boundary patches, i.e. the inlet boundary located in the minimum *x*-axis, the outlet boundary located in the maximum *x*-axis, the top boundary located in the maximum *y*-axis and the bottom boundary located in the minimum *y*-axis. There are 11 layers of hexahedron grids that are close to each coupling interface. Each cylinder boundary contains 192 cells along the circumferential direction and keeps the same cell height as that of the ARCTIC mesh at the coupling interface. The top, bottom, front and back boundaries of the OpenFOAM domain are set as symmetry boundary conditions. The ARCTIC domains (shown as non-red grids in Figure 15**Error! Reference**

source not found.) have the same topology and number of cells as that in Figure 9. All five domains extend from 0 m to 0.1 m with 8 cells along the *z*-axis direction. The total number of cells for the OpenFOAM domain is 2,280,144 and the total number of cells for each of the ARCTIC domains is 49,152 and there are four such domains in this test case. The centres of the four cylinders at the *x*-*y* plan are (-2, 2), (2, 2), (-2, -2) and (2, -2) respectively.

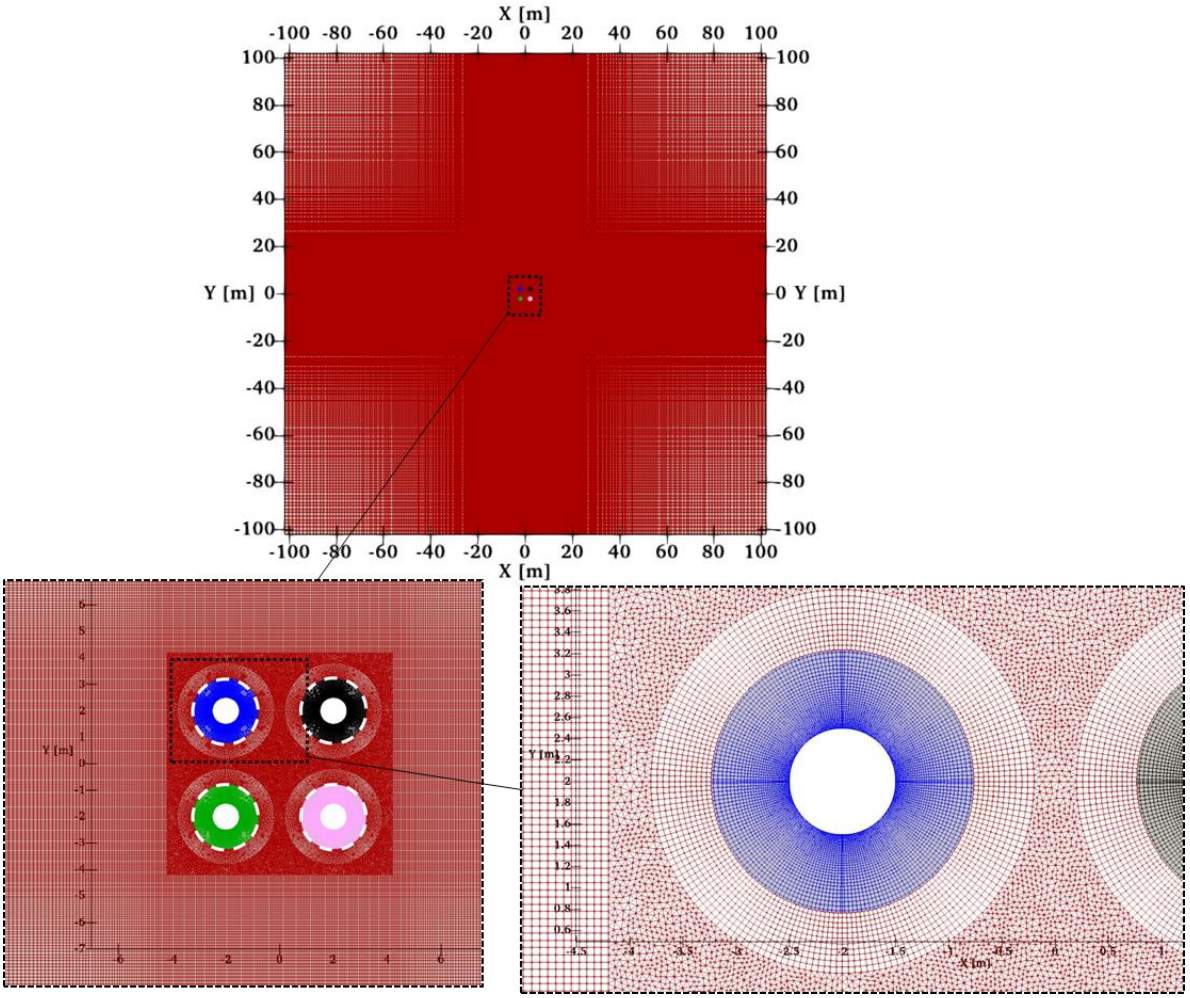

Figure 15 Coupled four-cylinder meshes. The red grid is for the OpenFOAM domain, the blue grid is for the first ARCTIC domain located in the second quadrant, the black grid is for the second ARCTIC domain located in the first quadrant and the green grid is for the third ARCTIC domain that located in the third quadrant and the pink grid is for the fourth ARCTIC domain that located in the fourth quadrant. The white dashed circles in the lower left plot indicate the coupling interfaces between OpenFOAM and ARCTIC domains

The physical and numerical setups are as same as that in Section 4.1. Pressure and velocity contours as well as streamline plots in Figure 16 show smooth results across all coupled interfaces.

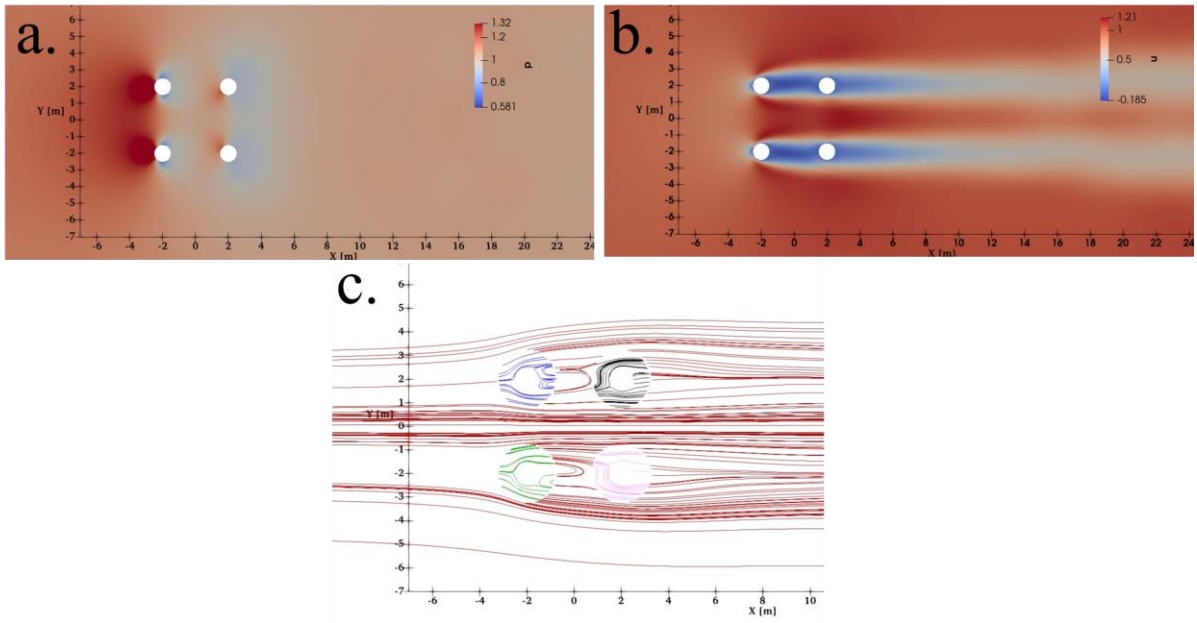

Figure 16 *Re*=40 steady-state simulation on four cylinders case with a): pressure contour, b): velocity contour and c): streamline. Red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

4.5. Flow past four cylinders at Re=40 in unsteady mode

We have used the same configuration as in Section 4.4, but ran the simulations in unsteady mode for testing. The same meshes are employed as that in Section 4.4, and the physical/numerical setup is as same as that in Section 4.2. Simulation results include pressure contour, velocity contour and streamline are shown in Figure 17. Smooth results are shown.

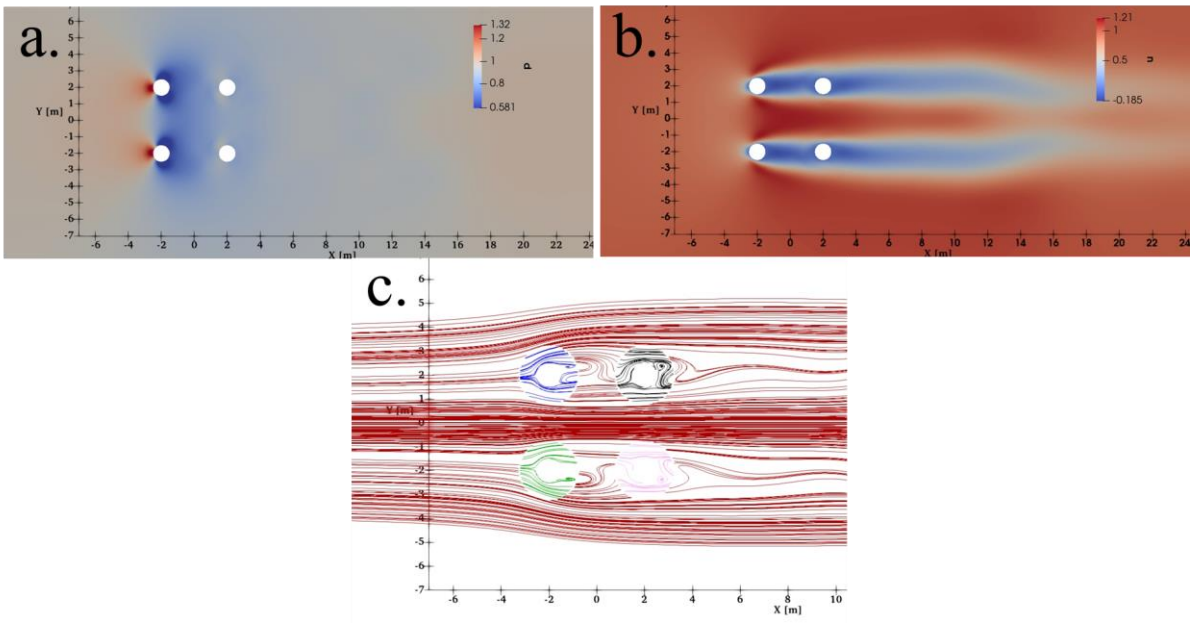

Figure 17 *Re*=40 unsteady simulation at *t*=1 s in the four cylinder case with a): pressure contour, b): velocity contour and c): streamline. Red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

4.6. Flow past four cylinders at Re=150

The last test case is the flow past four cylinders at *Re*=150. The same meshes were employed as that in Section 4.5. Physical and numerical setup as well as the simulation procedure are the same as that in Section 4.3. Smooth and continuous results were obtained as shown in Figure 18.

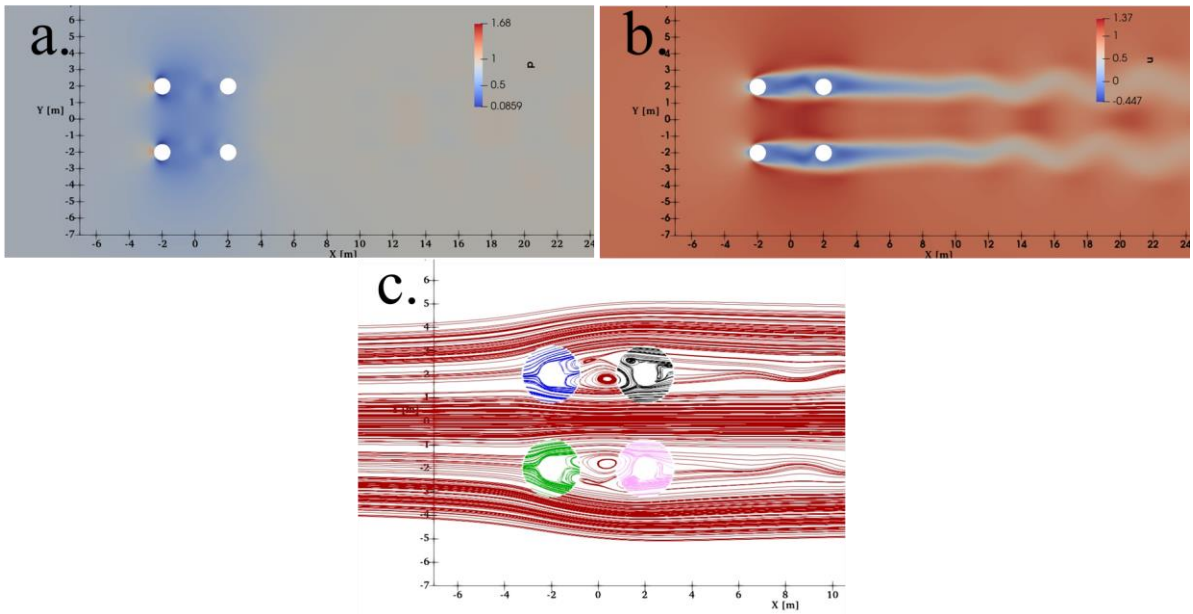

Figure 18 *Re*=150 unsteady simulation at *t*=3 s of four cylinders case with a): pressure contour, b): velocity contour and c): streamline. Red streamlines are generated from the OpenFOAM domain and black streamlines are generated from the ARCTIC domain

5. Scalability test

We have carried out a scalability test for the full coupling framework in case of the flow past four cylinders at *Re*=40. This scalability test configuration is composed of five domains. The OpenFOAM domain (shown as red grids in Figure 19) has outer boundaries that extend from -102 m to 102 m in both *x*-axis and *y*-axis directions. The OpenFOAM domain is meshed using both structured hexahedron and unstructured pentahedron cells. The outer boundary of the OpenFOAM domain consists of four boundary patches, i.e. the inlet boundary located in the minimum *x*-axis, the outlet boundary located in the maximum *x*-axis, the top boundary located in the maximum *y*-axis and the bottom boundary located in the minimum *y*-axis. There are 14 layers of hexahedral cells that are close to each coupling interface. Each coupled interface contains 400 cells along the circumferential direction and keeps the same cell height as that of the ARCTIC mesh at the coupling interface. The top, bottom, front and back boundaries of the OpenFOAM domain are set as symmetry boundary conditions. The ARCTIC domains (shown as non-red grids in Figure 19Error! Reference source not found.) have 4 blocks in the circumferential direction, 2 blocks in the radial direction and 32 blocks in the axial direction. All blocks have an equal number of cells with 100 cells along the circumferential direction, 24 cells along the radial direction and 16 cells along the axial direction. All five domains extend from 0 m to 5.12 m with 512 cells along the axial direction. The total number of cells for the OpenFOAM domain and the ARCTIC domains are shown in Table 2.

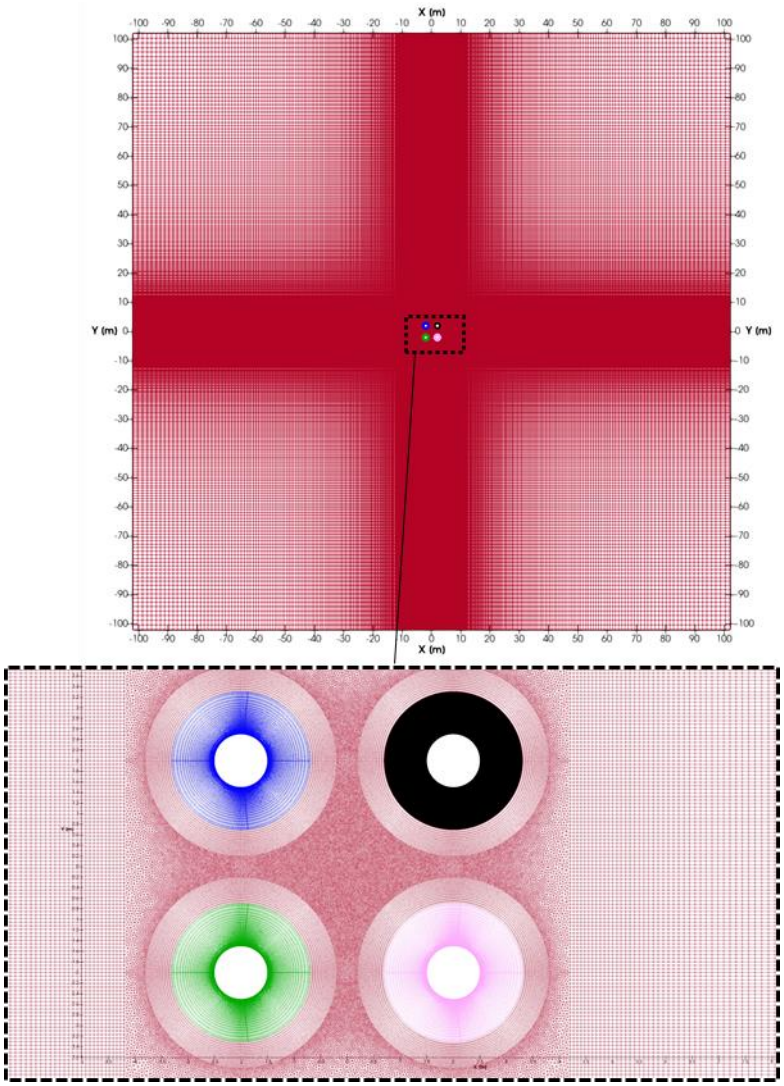

Figure 19 Coupled four-cylinder meshes for scalability test

Table 2 Cell numbers of each domain for the scalability test

Cell Numbers ARCTIC	9,830,400	per domain (x4)
Cell Numbers OpenFOAM	170,704,896	-

The numerical setup of the scalability test follows the same settings as the test case of the flow past four cylinders at *Re*=40 in case of steady-state mode as reported in Section 4.4. Figure 20 shows the parallel speedup v.s the number of MPI tasks on ARCHER2 with detailed information in Table 3. The maximum number of MPI tasks, i.e. 1536, is only limited by the number of predefined geometric partitions required by the multi-block code, e.g. 256 per each ARCTIC domain. Very good performance is observed over the entire range of selected core counts.

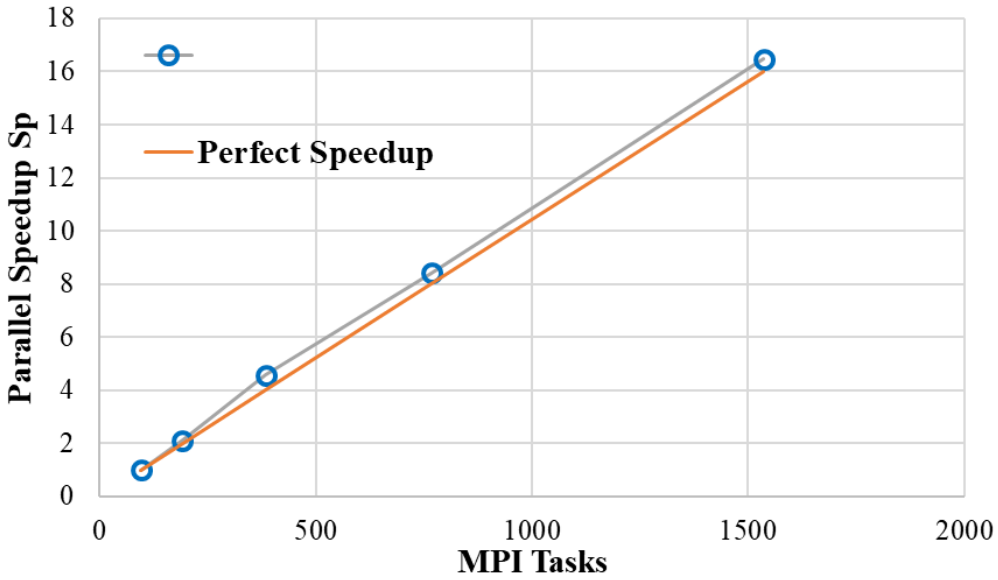

Figure 20 Parallel speedup vs MPI tasks for the scalability test of the coupled framework

Table 3 Scalability test results

MPI tasks total	MPI tasks OF domain	MPI tasks ARCTIC domain #1	MPI tasks ARCTIC domain #2	MPI tasks ARCTIC domain #3	MPI tasks ARCTIC domain #4	Averaged wall-clock time per iteration [s]	**Parallel speedup Sp**	**Parallel efficiency Ep**	Perfect Speedup
96	32	16	16	16	16	82.4	**1**	**1**	1
192	64	32	32	32	32	39	**2.11**	**1.06**	2
384	128	64	64	64	64	18	**4.58**	**1.14**	4
768	256	128	128	128	128	9.8	**8.41**	**1.05**	8
1536	512	256	256	256	256	5	**16.48**	**1.03**	16

6. Conclusions and impact

We have developed and implemented a new simulation framework by coupling the open-source OpenFOAM and ARCTIC computational fluid dynamics analyses (CFD) codes by means of the Multiscale Universal Interface library for pre-exascale blade-geometry resolved CFD of open multi-rotor systems, such as floating offshore wind farms and propeller-based distributed electric propulsion systems. A number of test cases have been performed. They include both steady- and unsteady-state simulations, including flow past one stationary cylinder and four stationary cylinders with Reynolds numbers equal to 40 (steady flow) and 150 (unsteady flow). These test cases show a smooth and continuous flow field across the coupled interface for both steady and unsteady simulations. The preliminary scalability test shows good scalability for the coupled framework up to 1536 MPI tasks, and it is expected that a case which would be n times larger than the configuration with 1 OpenFOAM (mesh n times larger) plus $4 \times n$ ARCTIC instances for $4 \times n$ cylinder would show very good performance up to n times 1536 MPI tasks. This case could be seen as a simplification of the flow in and around an off-shore wind turbine farm.

The project's deliverables benefit the renewable energy field in the development and deployment of high-fidelity methods for offshore wind energy. The new multi-rotor simulation technology also benefits sustainable aviation for the analysis of propeller-based DEP.
The OpenFOAM-MUI-ARCTIC framework will be installed as a single module maintained by the PI, Dr Sergio Campobasso. Access for the UK and international researchers will be sought through EPSRC support (including the Pioneer scheme), mainly.
The multi-code functionality will also benefit onshore wind R&D, enabling presently unfeasible rotor-resolved analysis of 100+ turbine farms. Resolving with adequate fidelity both rotor aerodynamics and complex terrain-dependent atmospheric boundary layer (ABL) flows will allow accurate prediction of wake-induced energy losses, as such ABL flows affect both the time- and space-dependent wind resource available to each turbine, and rotor wake kinematics and recovery. A lot of the UK and international research focus on the growing area of arrays of tidal stream turbines, where the rotor-resolved OpenFOAM-MUI-COSA framework allows to reliably resolve velocity gradients due to irregular bathymetry, surface gravity wave perturbations and turbine wake.
In all the aforementioned applications, improved turbine designs, lower energy losses in multi-rotor environments, and lower cost of energy will result in more promoting knowledge and technology transfer, especially because all the actors of this project have solid links with industry.
Recent investments in floating wind R&D in the UK and abroad aim at improving fidelity of both experimental and simulation-based research on FOWT aero-hydrodynamics by using physical rotors rather than surrogate thrusters in ocean basin tests, and geometry-resolved rotors rather than actuator line-based rotor surrogates in CFD simulations. The new computational aerodynamic technology for multi-rotor systems enabled by this project is key for rotor-resolved floating wind farm simulations needed to understand and minimise energy losses due to rotor/wake interactions. For the first time, the developed multi-code framework will enable performing these highly resolved multi-physics simulations, using codes with optimal and scalable functionalities for both rotor aerodynamics and platform hydrodynamics.
This project's lessons also benefit the STFC Daresbury Laboratory (STFC-DL) Co-Is, who are MUI developers and are actively involved in deploying and further developing MUI for code coupling in other ongoing eCSE projects. Consolidation of MUI will also benefit present and future projects of the UKRI ExCALIBUR programme which uses MUI.

Acknowledgement

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (https://www.archer2.ac.uk/ecse/). We thank the useful discussions with Dr Stephen Longshaw.

Appendix I. Folder structure

A Git repository has been created at the beginning of this eCSE project to enable more efficient and reliable exchange and testing of newly developed code functionalities between STFC Daresbury Laboratory and Lancaster University. The folder structure of this Git repository is as follows.

There are five top-level folders. They are src, config, thirdParty, testCase and utility.

1. src

This folder contains the source file of ARCTIC v7.

2. config

This folder contains the configure file that will be used on ARCHER2. Simulations on ARCHER2 can simply load the correct modules by source this file.

3. thirdParty

This folder contains third-party codes that will be used for the coupling between ARCTIC and OpenFOAM.

3.1. Sub-folder MUI

This sub-folder contains the MUI source code with different version numbers. The sub-folder "V1.2" contains the source file of MUI v1.2 that is used in the coupling.

3.2. Sub-folder MUIArcticWrapper

This subfolder provides the customised MUI FORTRAN wrapper for ARCTIC in "mui_c_wrapper_arctic.cpp" and "mui_f_wrapper_arctic.f90". It also contains the "MUI_ARCTIC" module with useful functions for the ARCTIC MUI embedding.

3.3. Sub-folder OF2106MUI

It contains the source code on the MUI embedded OpenFOAM v2106. Detailed instruction has been provided through the README.md file in this subfolder.

3.4. Sub-folder OFArcticSolver

It contains the source code of the steady-state (muiSimpleFoam) and unsteady (muiPimpleFoam) OpenFOAM solvers that are able to couple with ARCTIC with data exchange functions.

4. testCase

It contains the test cases presented in this report. It includes the flow past flat plate (ARCTIC-ARCTIC coupling only), and flow past cylinder(s).

5. utility

It contains utility codes that will be useful for mesh modifications, running on HPC systems and post-processing for the coupled framework.

Appendix II. Installation of framework

Step One: Obtain the Git repository.
The Git repository needs to be obtained at the beginning of the installation.
Once Obtained the Git repository, fetch submodules by running the following command:

"git submodule update --init –recursive"

Step Two: Install MUI Embedded OpenFOAM v2106
Go to the subfolder "cd thirdParty/OF2106MUI"
Follow the "README.md" file to install the MUI Embedded OpenFOAM v2106.

Step Three: Install OpenFOAM solvers muiSimpleFoam and muiPimpleFoam
Source the installed MUI Embedded OpenFOAM v2106 by running the following command:
"source thirdParty/OF2106MUI/OpenFOAM-v2106/etc/bashrc"
Go to the subfolder "cd thirdParty/OFArcticSolver/src/application"
Compile the muiSimpleFoam solver by running the following command:
"wmake muiSimpleFoam"
Compile the muiPimpleFoam solver by running the following command:
"wmake muiPimpleFoam"
There is a unit test case in the subfolder "cd thirdParty/OFArcticSolver/unitTest" to verify the installation of the solver.

Step Four: Install customised MUI Fortran wrapper and MUI_ARCTIC module.
Go to the subfolder "cd thirdParty/MUIArcticWrapper/src"
Compile the code by running the following command:
"make HOST=archer2 DEBUG=0"
There is a unit test case in the subfolder "cd thirdParty/MUIArcticWrapper/unitTest" to verify the installation of the modules.

Step Five: Compile the ARCTIC v7 code.
Go to the folder "cd src"
Compile the code by running the following command:
"make mpi HOST=archer2 DEBUG=0 MUI=1"

After these five steps, the coupled framework has been compiled successfully and is ready to simulate.

Note for framework compile without coupling functions:
If the framework is used for single domain simulations, such as using ARCTIC to perform simulations without involving OpenFOAM, the code compilation will be much simple. The user can go to Step Five directly and use a different command:
"make mpi HOST=archer2 DEBUG=0 MUI=0"
or simply:
"make mpi"

Appendix III. Workflow on running a new case by using the framework

Step One: Generate meshes for every domain.
Step Two: In ARCTIC "input.dat", set up the case as normal. Set the correct boundary condition for the coupled boundary, i.e. BC=32. Keep a note on the "lmax" value, as it will be used in the OpenFOAM setting. For unsteady simulations, keep a note of "dt", "ntime" and "wflow_freq" values, as they will be used in the OpenFOAM setting.
Step Three: In ARCTIC "mui_input.dat", follow the test case as an example to set the values.
Step Four: In OpenFOAM, set up the case as normal.
Step Five: In OpenFOAM "controlDict", if the simulation is steady-state, set the "deltaT" as 1.0 and set the "endTime" as the value of "lmax" of the ARCTIC solver; if the simulation is unsteady, set the "deltaT" as the value of "dt" of ARCTIC solver, set "endTime" as the value of "dt" multiply by "ntime" (since the endTime is the physical time with the unit of "second", but "ntime" is the number of time steps in ARCTIC solver), set "writeInterval" as the value of "wflow_freq" of ARCTIC solver.

Step Six: In OpenFOAM "muiFluidCouplingDict", follow the test case as an example to set the values. if the simulation is steady-state, set the "muiTotalSubIteration" as 1; if the simulation is unsteady, set the "muiTotalSubIteration" as the value of "lmax" of the ARCTIC solver.
Step Seven: Load correct modules and source correct configure files before a run.
Step Eight: Run the case by using the following command or some equivalent commands:

```
"mpirun -np 2 -wdir ${arctic_domain_1_folder} ./arctic.mpi : \
        -np 2 -wdir ${arctic_domain_2_folder} ./arctic.mpi : \
        -np 2 -wdir ${arctic_domain_3_folder} ./arctic.mpi : \
        -np 2 -wdir ${arctic_domain_4_folder} ./arctic.mpi : \
        -np 2 -wdir ${openfoam_domian_folder} muiSimpleFoam -parallel -coupled"
```

Note: It will increase the stability of the simulation if using a low Reynolds number steady-state simulation results as the starting point for the simulation.

Appendix IV. Test case between ARCTIC and ARCTIC domains -- flow past a flat plate

This test case is flow past a flat plate. Both single domain (ARCTIC) simulation and two ARCTIC domains coupling are performed. The simulation domain extends from x=-0.333m to x=2.333m; y=0m to y=1m; z=0m to z=1m. The flat plate is located at the y=0 m plane and extends from x=0m to x=2.333m. The domain contains 65 cells along the x-axis and 97 cells along the y-axis and 25 cells along the z-axis. The cells closed to the minimum y-axis direction have higher mesh density to ensure a high resolution within the boundary layer. The total number of cells for each domain is 157,625.
The 1st simulation is the coupling between the left ARCTIC domain (extend from x=-0.333 m to x=1.0 m) and the right ARCTIC domain (extend from x=1.0 m to x=2.333 m) using the MUI library. The coupling interface between the two domains is located at x=1.0 m. The right domain has the same topology as the left domain (i.e. through grid modification code to translate the mesh and change the index in the input.dat to ensure the wall boundary is start at i=1). The second simulation is using a single domain with ARCTIC itself. This is achieved through the grid modification code to stretch the grid of the original mesh along the x-axis direction to ensure the domain is extended from x=-0.333 m to x=2.333 m. The index in the input.dat has also changed to ensure the wall starts roughly from x=0 m. The second simulation is for reference purposes. Figure 21 shows the blocks and meshes for the two simulations. It should be noted that the mesh for the single domain simulation is coarser than the coupled simulation along the x-axis direction for the reason above. It may lead to some disagreement on the results when comparing these two directly.

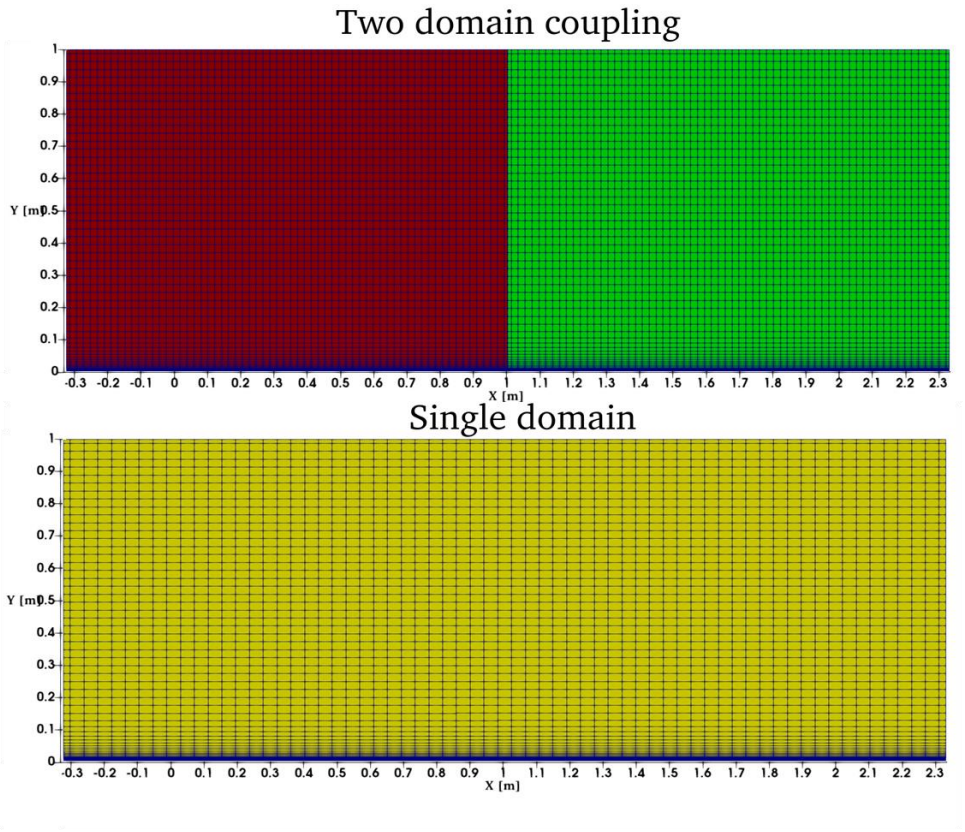

Figure 21 Computational domain for flow past a flat plate with ARCTIC-ARCTIC coupling (top) and single ARCTIC run (bottom). Red colour represents the left simulation domain of the coupling simulation, green colour represents the right domain of the coupling simulation and yellow colour represents the single block simulation

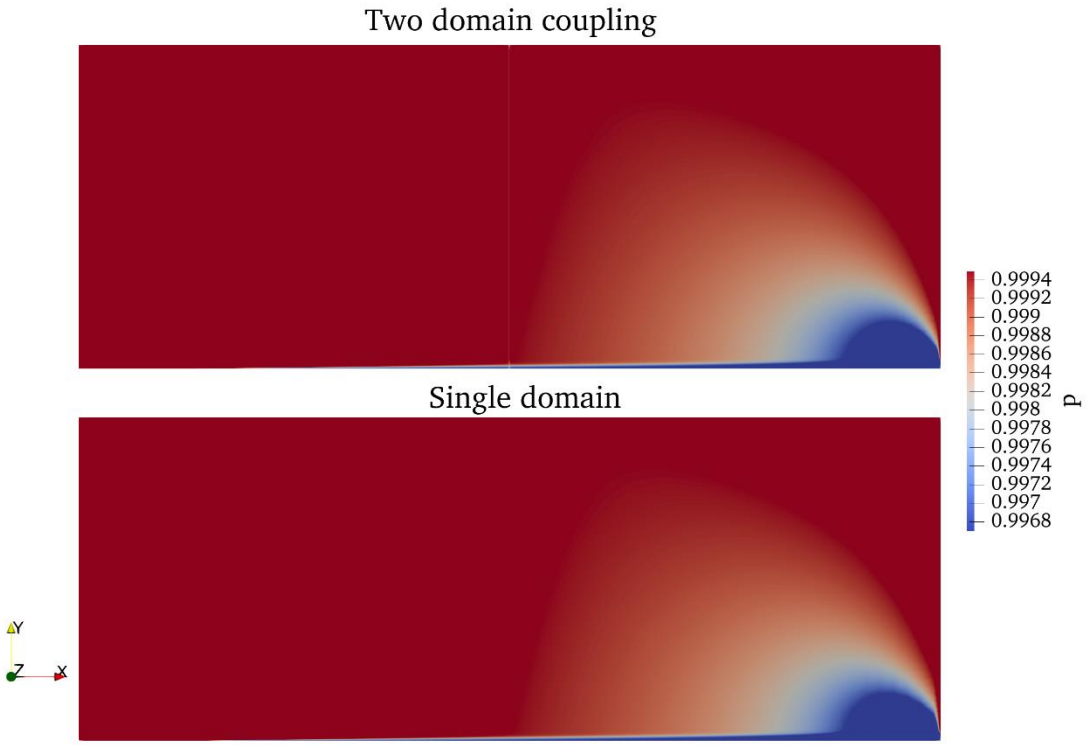

Figure 22 Pressure contour (p) on two domain coupling (top) and single domain (bottom)

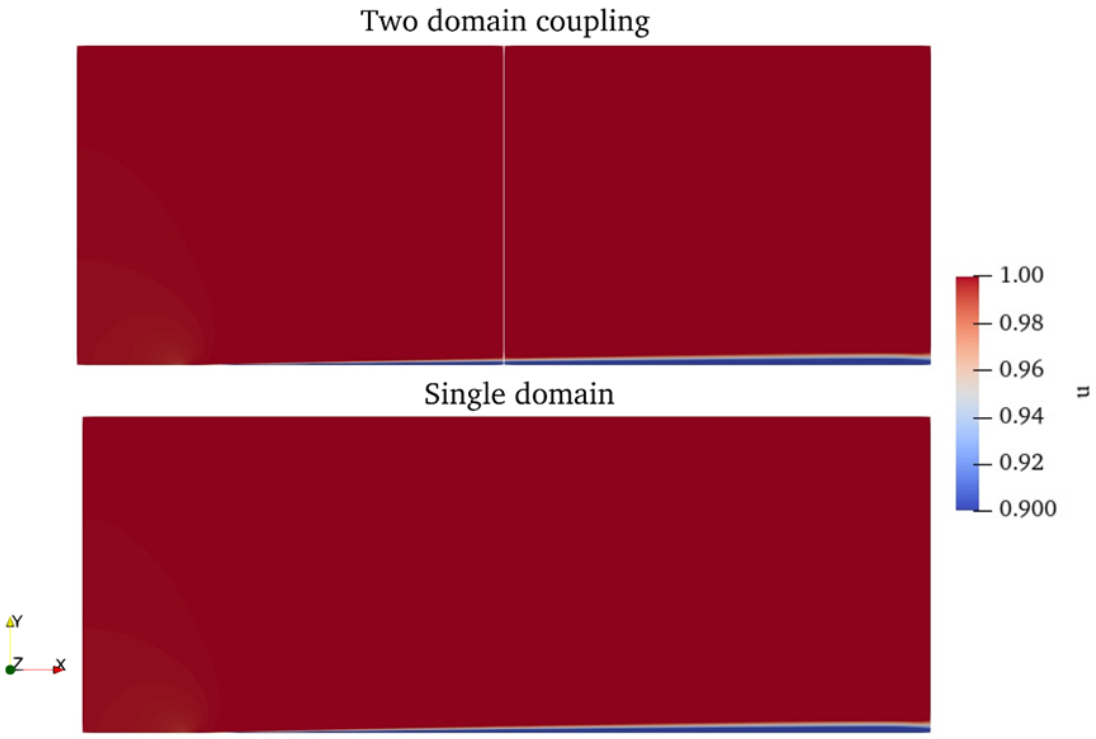

Figure 23 Contour of *x*-axis velocity (u) on two domain coupling (top) and single domain (bottom)

Pressure contours and velocity contours are plotted in Figure 22 and Figure 23. Good agreement is reached between the single domain simulation and coupled domain simulation.

The low-pressure area in both simulations does not look correct. It may be because the exit boundary condition used here is not the best-suited one for this problem. One would need an internal flow outlet boundary condition, but an external flow outlet boundary condition is used here, which assumes fairly uniform flow at the exit.

Appendix V. Test case between OpenFOAM and OpenFOAM domains – flow past a circular cylinder

This test case is the flow past a stationary cylinder at a Reynolds number of 40. It is composed of two domains, all of which solved by OpenFOAM. The outer domain (shown as the black grid in Figure 24) has outer circular boundaries with a radius of 20 m and an inner coupling boundary with a radius of 1.235 m. The outer boundary of the outer domain consisted of two boundary patches, i.e. the inlet boundary located in the second and the third quadrant, and the outlet boundary located in the first and the fourth quadrant. Each boundary contains 96 cells along the circumferential direction. There are 96 cells along the radial direction of the outer domain as well, but with a progression ratio of 0.9746. The inner domain (shown as the red grid in Figure 24) has outer circular boundaries with a radius of 2.5 m and an inner wall boundary with a radius of 0.5 m. There are 192 cells along the circumferential direction and 56 cells with a progression ratio of 0.945 along the radial direction of the inner domain. Both of these two domains extend from 0m to 0.1 m with 8 cells along the *z*-axis direction. The front and back boundaries of both domains are set as symmetry boundary conditions. The total number of cells for the outer domain is 147,456 and

the total number of cells for the inner domain is 86,016. The physical and numerical setup is the same as that in Section 4.1.

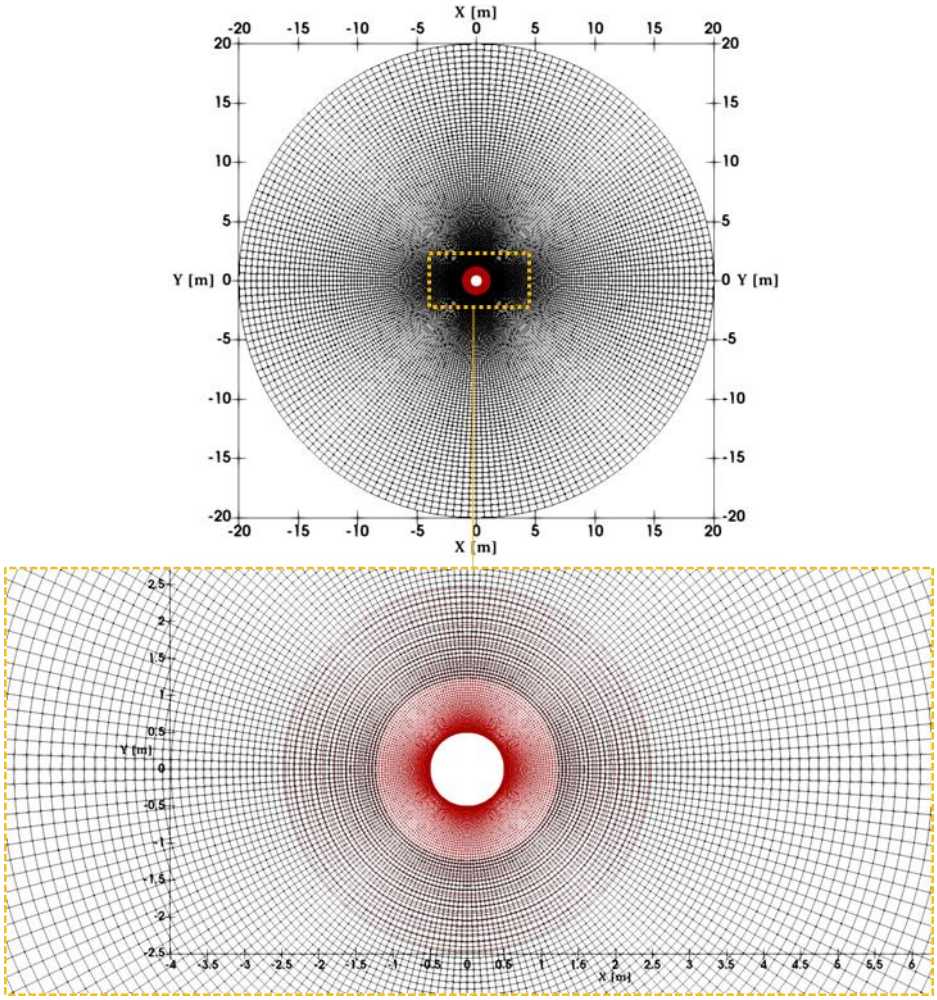

Figure 24 Coupled single cylinder meshes for OpenFOAM-OpenFOAM coupling test. The red grid is for the inner domain and the black grid is for the outer domain

Pressure contour, velocity contour as well as streamlines are shown in Figure 25. A smooth and continuous fluid field is observed.

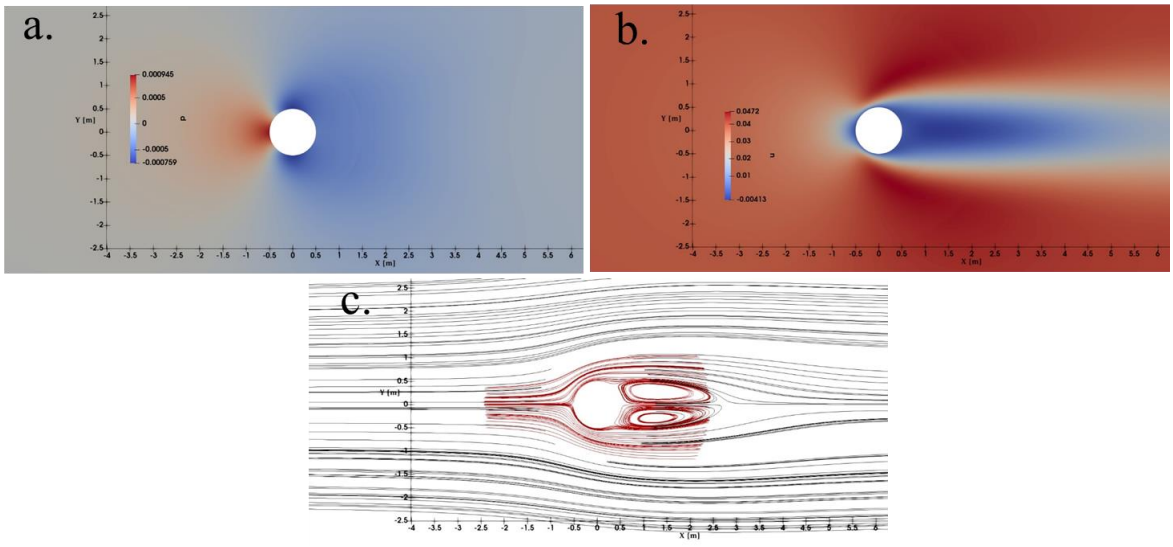

Figure 25 *Re*=40 unsteady simulation for OpenFOAM-OpenFOAM coupling test with a): pressure contour, b): velocity contour and c): streamline. Red streamlines are generated from the inner domain and black streamlines are generated from the outer domain

Appendix VI. Modules used in ARCHER2

Currently Loaded Modules:

1) gcc/10.2.0
2) craype/2.7.6
3) craype-x86-rome
4) xpmem/2.2.40-7.0.1.0_2.7__g1d7a24d.shasta
5) bolt/0.7
6) epcc-setup-env
7) load-epcc-module
8) PrgEnv-gnu/8.0.0
9) mkl/21.2-2883
10) cray-python/3.8.5.0
11) cmake/3.21.3
12) cray-libsci/21.04.1.1
13) boost/1.72.0
14) perftools-base/21.02.0
15) perftools
16) cray-ucx/2.7.0-1
17) craype-network-ucx
18) cray-mpich-ucx/8.1.9

References

1. Kim, H.D., A.T. Perry, and P.J. Ansell. *A review of distributed electric propulsion concepts for air vehicle technology*. in *2018 AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*. 2018. IEEE.
2. Drofelnik, J., A. Da Ronch, and M.S. Campobasso, *Harmonic balance Navier-Stokes aerodynamic analysis of horizontal axis wind turbines in yawed wind.* Wind Energy, 2018. **21**(7): p. 515-530.
3. ARCHER, *COSA*, in *Software Documentation*. 2019: ARCHER. p. www.archer.ac.uk/documentation/software/cosa/.
4. Ortolani, A., et al., *Computational Fluid Dynamics Analysis of Floating Offshore Wind Turbines in Severe Pitching Conditions.* Journal of Engineering for Gas Turbines and Power, 2020. **142**(12).
5. Ortolani, A., et al. *Cross-comparative analysis of loads and power of pitching floating offshore wind turbine rotors using frequency-domain Navier-Stokes CFD and blade element momentum theory*. in *Journal of Physics: Conference Series*. 2020. IOP Publishing.
6. Cavazzini, A., et al., *Harmonic Balance Navier–Stokes Analysis of Tidal Stream Turbine Wave Loads*, in *Recent Advances in CFD for Wind and Tidal Offshore Turbines*. 2019, Springer. p. 37-49.
7. Drofelnik, J., A. Ronch, and M. Sergio Campobasso, *Feasibility of the Navier-Stokes harmonic balance method for modelling aircraft unsteady aerodynamics.* 2018.
8. ARCHER. *Improving the ease-of-use and portability of the COSA 3D solver for open rotor unsteady aerodynamics eCSE project eCSE03-002*. 2017; Available from: https://www.archer.ac.uk/community/eCSE/eCSE03-02/eCSE03-02.php.
9. Jasak, H., A. Jemcov, and Z. Tukovic. *OpenFOAM: A C++ library for complex physics simulations*. in *International workshop on coupled methods in numerical dynamics*. 2007. IUC Dubrovnik Croatia.
10. Tang, Y.-H., et al., *Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers.* Journal of Computational Physics, 2015. **297**: p. 13-31.

11. Skillen, A., et al. *Profiling and application of the multi-scale universal interface (MUI)*. in *7th European Conference on Computational Fluid Dynamics, Glasgow, UK*. 2018.
12. OpenFOAM. *Boundary conditions*. 2022 [cited 2022; Available from: www.openfoam.com/documentation/guides/v2112/doc/openfoam-guide-boundary-conditions.html.