ExaQUte

Exascale **Q**uantification of **U**ncertainties for **Te**chnology and Science Simulation

D1.4 Final public Release of the solver

Document information table

Contract number:	800898
Project acronym:	ExaQUte
Project Coordinator:	CIMNE
Document Responsible Partner:	CIMNE
Deliverable Type:	Software release
Dissemination Level:	Public
Status:	Final version

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **800898**

Authoring

Prepared by: CIMNE				
Authors	Partner	Modified	Version	Comments
Quentin Ayoul-Guilmard Sundar Ganesh Fabio Nobile	EPFL	All	1.0.0	XMC
Rosa M. Badia Jorge Ejarque	BSC			PyCOMPSs
Luca Cirrottola Algiane Froehly	INRIA			ParMmg
Brendan Keith Anoop Kodakkal	TUM			Wind generator
Marc Nuñez Carlos Roig Riccardo Rossi Riccardo Tosi	CIMNE			Kratos Multiphysics, XMC
Cecilia Soriano				Coordination

Change Log

Versions	Modified Page/Sections	Comments
1.0.0	All	Submitted version

Approval

Aproved by:				
	Name	Partner	Date	OK
Task leader	Riccardo Rossi	CIMNE	30/11/2020	OK
WP leader	Riccardo Rossi	CIMNE	30/11/2020	OK
Coordinator	Riccardo Rossi	CIMNE	30/11/2020	OK

Executive summary

This deliverable presents the final software release of Kratos Multiphysics, together with the XMC library, Hyperloom and PyCOMPSs API definitions [13]. This release also contains the latest developements on MPI parallel remeshing in ParMmg. This report is meant to serve as a supplement to the public release of the software.

Kratos is "a framework for building parallel, multi-disciplinary simulation software, aiming at modularity, extensibility, and high performance. Kratos is written in C++, and counts with an extensive Python interface". XMC is "a Python library for parallel, adaptive, hierarchical Monte Carlo algorithms, aiming at reliability, modularity, extensibility and high performance". Hyperloom and PyCOMPSs are environments for enabling parallel and distributed computation. ParMmg is an open source software which offers the parallel mesh adaptation of three dimensional volume meshes.

Table of contents

1 Introduction **6**

2 Software structure **6**
2.1 Multiphysics software 6
2.2 Uncertainty Quantification 7
2.3 ParMmg 7

3 Examples **8**
3.1 Uncertainty Quantification and Ensemble Average 8
3.2 MPI fluid dynamics simulations and parallel adaptive refinement with ParMmg 9

4 Current results **9**

Nomenclature / Acronym list

Acronym	Meaning
API	Application Program Interface
Kratos	Kratos MultiPhysics
UQ	Uncertainty Quantification

1 Introduction

The Kratos software [10, 11] is designed for dealing with a multitude of different physical problems, spacing from computational fluid dynamics to convection diffusion, from structural applications to fluid structure interaction. For more information, we refer to the Kratos documentation.

Kratos is mainly used in the project as the base multiphysics solver in each of the developments of the project. Kratos is designed in an structured manner by means of applications, which allows to develop and use a wide variety of numerical tools, methods and solvers in the same base framework. For the purpose of the project, the main application used as main solver is the FluidDynamicsApplication, a finite-element CFD framework. This application of Kratos can deal with incompressible Navier-Stokes problems on classical finite-element unstructured meshes and also on immersed meshes, which meets the requirements of the project considering the physical problem to be solved. Other applications will be used, which are detailed on section 2.1.

As explained in previous software release deliverables [1, 2, 4, 19], Kratos can solve Uncertainty Quantification problems. UQ is the field of science which studies how uncertainties evolve and propagate in problems, which are characterized by random parameters. UQ methods have been developed within the XMC Python library [3], and Kratos is the default solver software of the library. Further details about the XMC library can be found in [1, 4].

Both XMC and Kratos are designed for large scale computing in distributed environments. The scheduling in distributed environments is handled by external libraries: Hyperloom [8] and PyCOMPSs [5, 15, 18]. XMC presents a natural integration with these libraries. Due to this inner integration of XMC with Hyperloom and PyCOMPSs, the API definitions of the schedulers have been integrated in the software release.

This release also contains the latest developments on the MPI paralell remeshing software developed by INRIA: ParMmg. An interface with Kratos has been developed, and it enables the use of the remeshing software along with the multiphysics solvers.

Therefore, this final software release [13] contains Kratos, XMC, ParMmg, Hyperloom and PyCOMPSs API definitions. The complete changelog is available in the Kratos releases page.

2 Software structure

2.1 Multiphysics software

Kratos Multiphysics 8.1 release can be retrieved from the Kratos releases page. The main applications that are used in the project are:

- ExaqauteSandboxApplication: Contains custom processes and utilities used to run the examples in the ExaQUte project.
- FluidDynamicsApplication: Contains all CFD formulations and utilites, including body-fitted and embedded incompressible Navier-Stokes solvers.
- LinearSolversApplication: Contains a wide selection of different linear solvers (LU, multi-grid, eigen-solvers) to solve the linear equation on the FEM analysis.

- MappingApplication: contains the core developments in mapping data between non matching grids. For instance, two different meshes with different discretization.
- MeshingApplication: Contains several utilities to perform meshing and remeshing, as well as the wrapper to use Mmg and ParMmg.
- MetisApplication: Wrapper for Metis library to be used within Kratos.
- MultilevelMontecarloApplication: Contains a wrapper with the XMC library and utilites to perform UQ.
- StatisticsApplication: Contains utilities to compute statistical measures of the physical quantites involved in the analyses.
- TrilinosApplication: Wrapper for Trilinos library to be used within Kratos.

2.2 Uncertainty Quantification

Within Kratos, different applications are organized in different folders. As described in previous works [2, 19], one folder is entirely dedicated to UQ and it is called *MultilevelMonteCarloApplication*. The folder is located in:

```
Kratos/applications/MultilevelMonteCarloApplication
```

This folder contains XMC in:

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   XMC
```

and Hyperloom and PyCOMPSs API definitions in:

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   PyCOMPSs/exaqute
```

2.3 ParMmg

ParMmg is the MPI parallel version of the metric-based refinement sofware Mmg. Mmg is an open source software for bidimensional and tridimensional surface and volume remeshing. For this release, ParMmg offers the parallel mesh adaptation of three dimensional volume meshes [6, 9, 12].

In this release of the solvers, an interface in Kratos with the recent developments on the MPI parallel remeshing software ParMmg is included. This allows to perform parallel refinement operations from the Kratos solver without effort from the user, as the interface contains the library calls to ParMmg and takes care on adapting the data structures from Kratos to ParMmg.

These remeshers require the definition of a metric field, which will set the desired size and orientation of the final elements on the remeshed mesh. A metric is a tensor of size *dxd* where d is the number of dimensions. The computation of the metrics is available in the MeshingApplication of Kratos.

The releases of Mmg and ParMmg can be downloaded from:

```
https://github.com/MmgTools/mmg/releases/tag/v5.5.2
https://github.com/MmgTools/ParMmg/releases/tag/v1.2.0
```

3 Examples

Kratos examples has been released together with the software release, and can be found here [14]. In the following sections, examples on UQ and adaptive refinement will be presented, which will also serve as an example of usage of Kratos Multiphysics, with examples directly related to the project.

3.1 Uncertainty Quantification and Ensemble Average

Different algorithms have been developed during the project within the XMC library: Monte Carlo, Multilevel Monte Carlo, Adaptive Monte Carlo, Adaptive Multilevel Monte Carlo, Continuation Multilevel Monte Carlo, Asynchronous Monte Carlo and Asynchronous Multilevel Monte Carlo. We refer to previous works for details [1, 2, 4, 19].

In addition to previous deliverable, XMC can be exploited as well to solve unsteady and ergodic problems, applying the ensemble average method [16].

Referring to the Examples release [14], all problems are located in the

```
Examples/multilevel_monte_carlo
```

folder and they are organized in two subfolders: *validation* and *use_cases*. The former validates the software against literature, while the latter shows the sequence of problems that were solved in order to arrive to the final UQ application: the *Commonwealth Advisory Aeronautical Council* (CAARC) benchmark problem [7]. Each subfolder contains as many folders as problems that were solved and analyzed. From the problem name, the user can easily understand the nature of the solution strategy: UQ, ensemble average or both.

Problem descriptions and instructions to run each problem can be found within each folder of interest, under the form of *README.md* file.

By default, all problems run in serial. In case the user is interested in running in parallel, in every folder bash files are added. These can be used as a reference to run using Hyperloom or PyCOMPSs. Moreover, in each execution script[1] and in the XMC file

```
Kratos/applications/MultilevelMonteCarloApplication/external_libraries/
   XMC/xmc/distributedEnvironmentFramework.py
```

the appropriate import has to be changed from

```
from exaqute.ExaquteTaskLocal import *
```

to

```
from exaqute.ExaquteTaskHyperLoom import *
```

or

```
from exaqute.ExaquteTaskPyCOMPSs import *
```

For further details about running in distributed environments, we refer to the MultilevelMonteCarloApplication documentation and to the PyCOMPSs and Kratos tutorial.

[1] Execution scripts are called *run_mc_Kratos.py* or *run_mlmc_Kratos.py*.

3.2 MPI fluid dynamics simulations and parallel adaptive refinement with ParMmg

Prior to this release, the remesher Mmg was already available, and it was covered in the previous Deliverable 1.3 [2]. It is reminded that there is a set of examples for the serial remesher, showcasing the definition of different metrics and the use of Mmg software in the Kratos Examples release [14], located in:

```
Examples/mmg_remeshing_examples
```

On this release, a new set of examples is added with the recent developments on ParMmg, which are located in:

```
Examples/parmmg_remeshing_examples
```

These examples can only be run in MPI. The first example is a basic fluid dynamics channel flow with a body-fitted cylinder. The script solves a few seconds of simulation, and then the information on the velocity and pressure field is used to adaptively refine the mesh. This process is repeated several times, improving gradually the accuracy on the solution. This is an example of how the solution of a previous simulation can be used to improve future simulations. It is also a showcase of how all involved operations can be performed in parallel: the physics simulation, the metrics computation, the remeshing operation and the mapping of quantities. Further instructions on how to run the example are located in the *README.md* file.

The second example is an embedded simulation of a high-rise building, which is highly related to the project. The script starts with a background mesh, and the skin of the building is loaded by computing the distance to the skin on the nodes of the background mesh. The information of this distance can be used as well to refine adaptively. The process of computing the distance and remeshing can be repeated a few times to improve the resolution of the geometry. After finishing the remeshing process, the embedded simulation is started on the improved background mesh. In this case, the remeshing is performed before starting the fluid dynamics simulation as a pre-process. This is also a perfect showcase of what would be needed in an optimization loop, as every optimization step modifies slightly the geometry. In each optimization step, the mesh could be adapted to the new design to keep an accurate representation of the geometry. Further instruction on how to run the example are located in the *README.md* file.

4 Current results

The work developed and released has been run and tested with several benchmarks, which have been run in different supercomputers. The software has been tested up to 192 working nodes (9216 CPUs), and showed a very good parallel efficiency. Part of the work has already been submitted to a peer reviewed journal, while other works will be submitted soon.

Regarding the adaptive refinement strategies, some figures are presented next. Firstly, Figure 4.4 shows the body-fitted cylinder case, which was run with 32 MPI processors. It uses the information of the velocity field at every time step to perform the refinement. With this example, the user can test other combinations of physical quantities such as the pressure, to get a different adaptation. Next, Figures 4.5 and 4.6 show a before-after

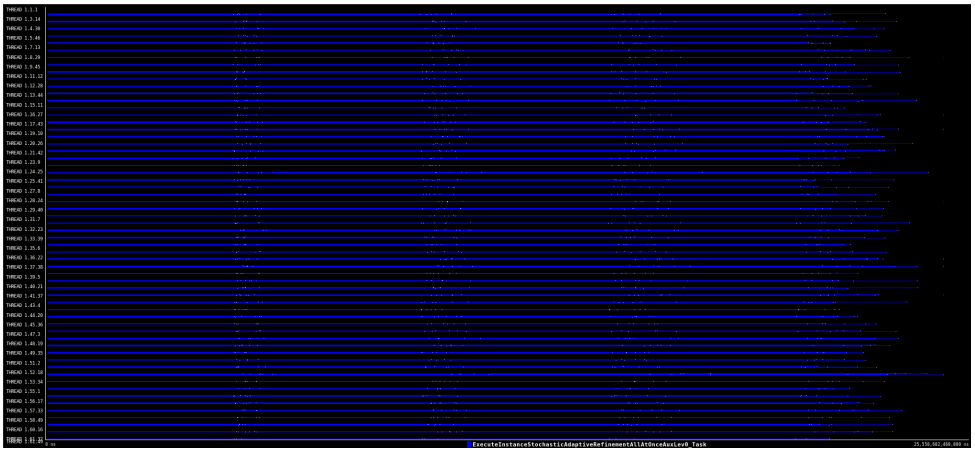


Figure 4.1: Execution trace of asynchronous Monte Carlo algorithm, running with PyCOMPSs with 41 worker nodes / 1968 CPUs. Blue lines are job executions, and they completely fill the machine during the whole runtime. The problem is the wind engineering two-dimensional benchmark (see [14] for details).

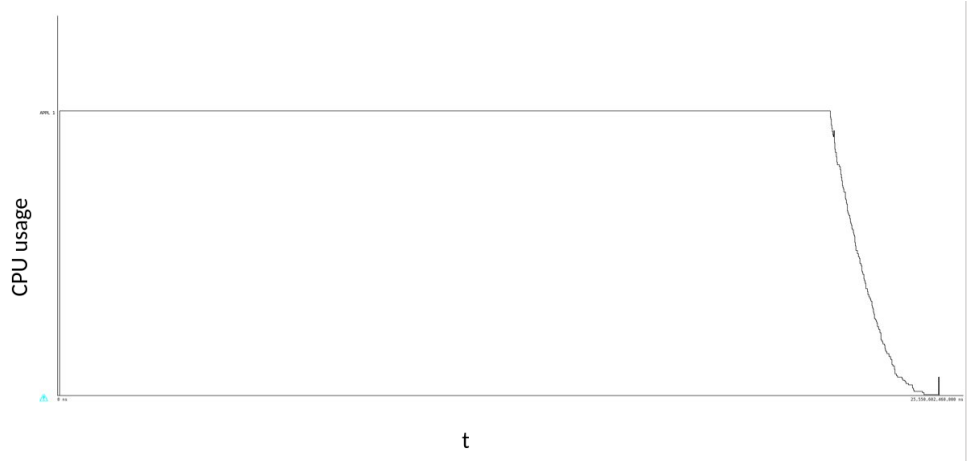


Figure 4.2: CPU usage of asynchronous Monte Carlo algorithm, running with PyCOMPSs with 41 worker nodes / 1968 CPUs. CPU usage is constant and equal to 100% of the machine during the whole runtime. The problem is the wind engineering two-dimensional benchmark (see [14] for details).

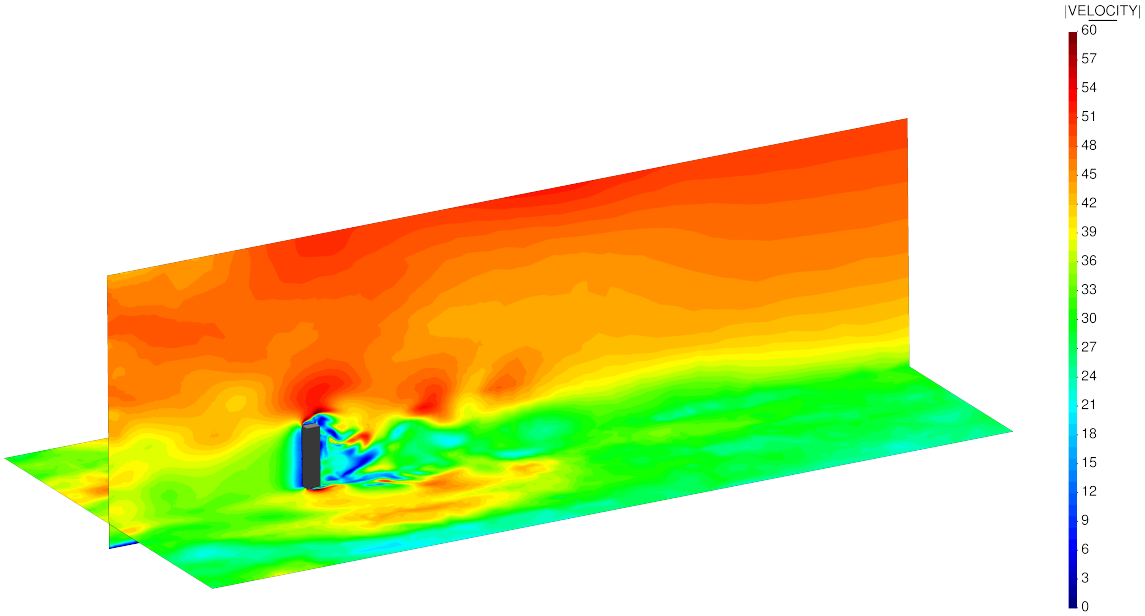
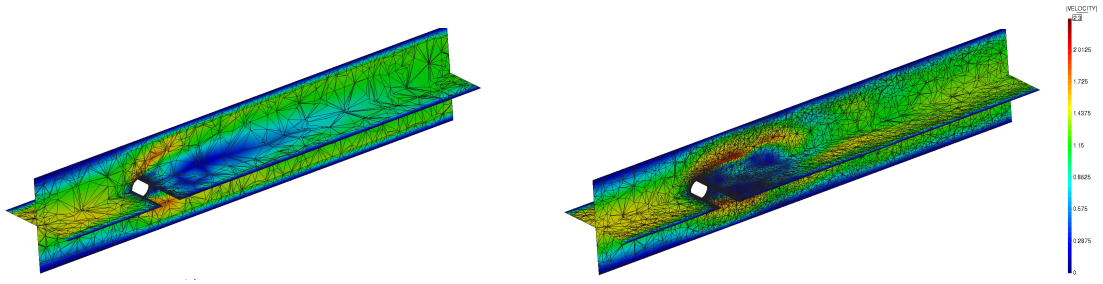


Figure 4.3: Velocity field around the CAARC building. The problem has been run with the Mann wind model [17].

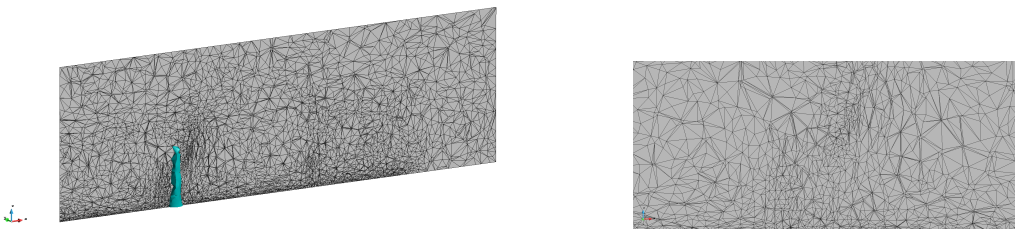
comparison of an adaptive refinement strategy using the distance field with 256 MPI processors. It can be seen that discretization is not good enough to correctly represent the building in the initial mesh. Adapting the mesh improves the description of the building, which is then used to start an embedded fluid dynamic analysis of the incompressible Navier-Stokes flow around the building. A snapshot of the velocity field in the embedded simulation on the remeshed mesh is shown in Figure 4.7.



(a) Initial mesh and its velocity field at T=5.0 s with 100k elements.

(b) Remeshed mesh at the fourth iteration using the velocity from previous solutions at T=5.0 s with 700k elements.

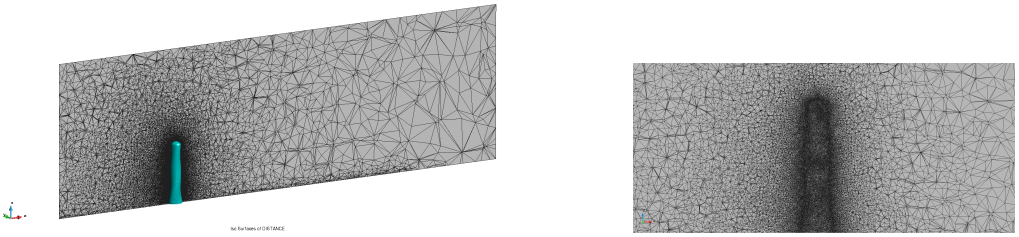
Figure 4.4: Comparison of the initial and final mesh after four iterations on a consecutive resolution and refinement process of a body-fitted cylinder mesh.



(a) Vertical cut and isosurface representation of a high-rise building using a distance function on an unadapted mesh. It can be seen that the resolution of the mesh is not enough to properly account for the geometry.

(b) Zoom on the mesh where the building is located. If compared to Figure 4.6 (b), it can be seen that the nodal presence is not enough to obtain a proper representation of the building

Figure 4.5: Vertical cuts on an unadapted background mesh with 200k elements where a high-rise building is loaded with a distance function. This mesh serves as the initial ground to load the geometry, and it is therefore not utilized to solve the fluid dynamics problem.



(a) Vertical cut and isosurface representation of a high-rise building using a distance function on an adapted mesh. Adapting the mesh increases the

(b) Zoom on the mesh where the building is located. Since the metric was built according to a distance function representing the building, smaller elements are accomplished near the skin, where the distance is close to 0.

Figure 4.6: Vertical cuts on an adapted background mesh with 2M elements where a high-rise building is loaded with a distance function. An embedded solver is used to solve the incompressible Navier-Stoke flow around the building. The refinement was performed with 256 MPI processors.

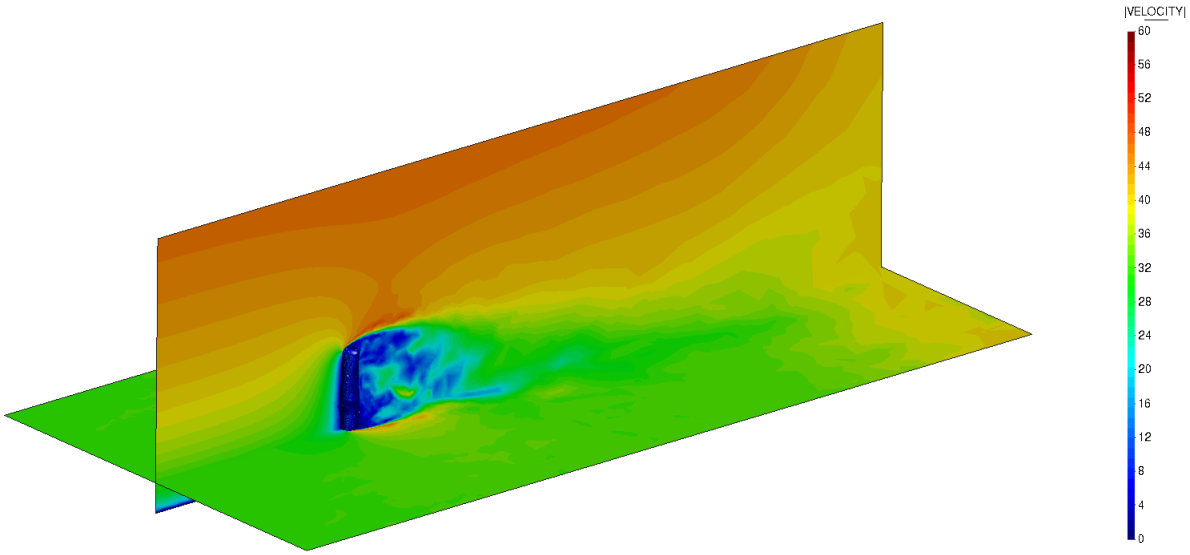


Figure 4.7: Velocity field around the embedded building with a steady logarithmic wind inlet. The geometry is represented by a distance function and the problem is solved using an embedded solver after remeshing with ParMmg. The analysis was run with 256 MPI processors.

References

[1] R. Amela, Q. Ayoul-Guilmard, R. M. Badia, S. Ganesh, F. Nobile, R. Rossi, and R. Tosi. D5.2 Release of ExaQUte MLMC Python engine, May 2019.

[2] Q. Ayoul-Guilmard, R. M. Badia, J. Ejarque, M. Núñez, S. Ganesh, F. Nobile, C. Soriano, C. Roig, R. Rossi, and R. Tosi. D1.3 First public Release of the solver. Technical report, BSC, CIMNE, EPFL, 2020.

[3] Q. Ayoul-Guilmard, S. Ganesh, F. Nobile, R. Rossi, R. Tosi, R. M. Badia, and R. Amela. XMC, 2020. URL `https://doi.org/10.5281/zenodo.3235832`.

[4] Q. Ayoul-Guilmard, M. Núñez, S. Ganesh, F. Nobile, R. Rossi, and R. Tosi. D5.4 Report on MLMC for time dependent problems. Technical report, ExaQUte, 2020.

[5] R. M. Badia, J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent. COMP superscalar, an interoperable programming framework. *SoftwareX*, 3–4, 12 2015. doi:10.1016/j.softx.2015.10.004.

[6] P. Benard, G. Balarac, V. Moureau, C. Dobrzynski, G. Lartigue, and Y. D'Angelo. Mesh adaptation for large-eddy simulations in complex geometries. *International Journal for Numerical Methods in Fluids*, 81(12):719–740, 2016. doi:https://doi.org/10.1002/fld.4204. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4204`.

[7] A. L. Braun and A. M. Awruch. Aerodynamic and aeroelastic analyses on the CAARC standard tall building model using numerical simulation. *Computers and Structures*, 87(9-10):564–581, may 2009. ISSN

00457949. doi:10.1016/j.compstruc.2009.02.002. URL http://dx.doi.org/10.1016/j.compstruc.2009.02.002.

[8] V. Cima, S. Böhm, J. Martinovič, J. Dvorskỳ, K. Janurová, T. V. Aa, T. J. Ashby, and V. Chupakhin. Hyperloom: A platform for defining and executing scientific pipelines in distributed environments. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*, pages 1–6. ACM, 2018.

[9] L. Cirrottola and A. Froehly. Parallel unstructured mesh adaptation using iterative remeshing and repartitioning. Research Report RR-9307, INRIA Bordeaux, équipe CARDAMOM, Nov. 2019. URL https://hal.inria.fr/hal-02386837.

[10] P. Dadvand, R. Rossi, and E. Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of computational methods in engineering*, 17(3):253–297, 2010.

[11] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers and Fluids*, 80(1):301–309, 2013. ISSN 00457930. doi:10.1016/j.compfluid.2012.02.004.

[12] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358 – 378, 2014. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2014.01.005. URL http://www.sciencedirect.com/science/article/pii/S0021999114000266.

[13] V. M. Ferrándiz, P. Bucher, R. Rossi, jcotela, J. Maria, R. Zorrilla, M. A. Celigueta, G. Casas, C. Roig, A. C. Velázquez, P. Dadvand, S. Latorre, J. I. González, I. de Pouplana, miguelmaso, M. Núñez, F. Arrufat, dbaumgaertner, B. Chandra, A. Ghantasala, armingeiser, S. Warnakulasuriya, lluís, J. Gárate, MFusseder, Pablo, AFranci, L. Gracia, thomas, K. B. Sautter, and R. Tosi. KratosMultiphysics/Kratos: Release 8.1, 2020. URL https://doi.org/10.5281/zenodo.3234644.

[14] V. M. Ferrándiz, R. Tosi, I. de Pouplana, R. Zorrilla, L. Gracia, S. Warnakulasuriya, M. Núñez, B. Chandra, A. Ghantasala, P. Bucher, jcotela, AFranci, F. Arrufat, K. B. Sautter, S. Latorre, jgonzalezusua, A. Geiser, P. Dadvand, miguelmaso, dbaumgaertner, M. A. Celigueta, RahulKN, swenczowski, BasselSaridar, C. Roig, M. Zidan, and G. Casas. KratosMultiphysics/Examples: Kratos Examples 8.1, Nov. 2020. URL https://doi.org/10.5281/zenodo.4293799.

[15] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia. ServiceSs: An Interoperable Programming Framework for the Cloud. *Journal of Grid Computing*, 12(1):67–91, 2014. ISSN 15707873. doi:10.1007/s10723-013-9272-5.

[16] V. Makarashvili, E. Merzari, A. Obabko, A. Siegel, and P. Fischer. A performance analysis of ensemble averaging for high fidelity turbulence simulations at the

strong scaling limit. *Computer Physics Communications*, 2017. ISSN 00104655. doi:10.1016/j.cpc.2017.05.023.

[17] J. Mann. Wind field simulation. *Probabilistic Engineering Mechanics*, 13(4):269–282, 1998. ISSN 02668920. doi:10.1016/s0266-8920(97)00036-2.

[18] E. Tejedor, Y. Becerra, G. Alomar, A. Queralt, R. M. Badia, J. Torres, T. Cortes, and J. Labarta. PyCOMPSs: Parallel computational workflows in Python. *International Journal of High Performance Computing Applications*, 31(1):66–82, 2017. ISSN 17412846. doi:10.1177/1094342015594678.

[19] R. Tosi, R. Amela, R. Badia, M. Núñez, and R. Rossi. D1.2 First release of the softwares. Technical report, BSC, CIMNE, 2019.