ALYA: MULTIPHYSICS ENGINEERING SIMULATION TOWARDS EXASCALE

MARIANO VÁZQUEZ[*], GUILLAUME HOUZEAUX[†], SEID KORIC[‡], ANTONI ARTIGUES[†], JAZMIN AGUADO-SIERRA[†], RUTH ARíS[†], DANIEL MIRA[†], HADRIEN CALMET[†], FERNANDO CUCCHIETTI[†], HERBERT OWEN[†], AHMED TAHA[‡], EVAN DERING BURNESS[‡], JOSÉ MARíA CELA[†], AND MATEO VALERO[†]

Abstract. Alya is a multi-physics simulation code developed at Barcelona Supercomputing Center (BSC). From its inception Alya code is designed using advanced High Performance Computing programming techniques to solve coupled problems on supercomputers efficiently. The target domain is engineering, with all its particular features: complex geometries and unstructured meshes, coupled multi-physics with exotic coupling schemes and physical models, ill-posed problems, flexibility needs for rapidly including new models, etc. Since its beginnings in 2004, Alya has scaled well in an increasing number of processors when solving single-physics problems such as fluid mechanics, solid mechanics, acoustics, etc. Over time, we have made a concerted effort to maintain and even improve scalability for multi-physics problems. This poses challenges on multiple fronts, including: numerical models, parallel implementation, physical coupling models, algorithms and solution schemes, meshing process, etc. In this paper, we introduce Alya's main features and focus particularly on its solvers. We present Alya's performance up to 100.000 processors in Blue Waters, the NCSA supercomputer with selected multi-physics tests that are representative of the engineering world. The tests are incompressible flow in a human respiratory system, low Mach combustion problem in a kiln furnace, and coupled electro-mechanical contraction of the heart. We show scalability plots for all cases and discuss all aspects of such simulations, including solver convergence.

Key words. Multi-physics coupling, Parallelisation, Computational Mechanics

1. Introduction.

Across a range of engineering fields, the use of computational models is pervasive in the whole design and manufacturing process. In complex systems, High Performance Computing (HPC) plays an essential role in simulation and modelling. Researchers and manufacturing teams depend on HPC to create safe cars and energy-efficient aircraft as well as effective communication systems and efficient supply chain models. Availability of advanced HPC technologies has also fundamentally altered the investigative paradigm in the field of biomechanics. But paradoxically, for many engineers and researchers, the existing hardware and software cannot be used to solve their problems. There are many reasons why this happens, but we focus here in only two. On one hand, current HPC systems lack the computational power, network bandwidth and data storage needed for solving tomorrow's real-world engineering challenges. On the other hand, while emerging peta-scale computing is already a strategic enabler of large-scale simulations in many scientific areas (such as astronomy, biology and chemistry), even the most powerful hardware will fail to deliver on its full potential unless matched with simulation software designed specifically for such environments.

Several papers describe the effort of performing large-scale simulations on supercomputers, covering key areas: molecular dynamics [26], mantle convection in solid earth dynamics [3], massive N-body simulations [36], seismic wave propagation [25], weather prediction [1] or fundamentals of turbulence on channels using the vortex method [37]. A similar list can be obtained from the 2014 ACM Gordon Bell Prize in

[*]Barcelona Supercomputing Center BSC-CNS, Campus Nord UPC, Barcelona, Spain and IIIA-CSIC, Bellaterra, Spain (mariano.vazquez@bsc.es)

[†]Barcelona Supercomputing Center BSC-CNS, Campus Nord UPC, Barcelona, Spain

[‡]National Center for Supercomputing Applications-NCSA University of Illinois at Urbana-Champaign, USA (koric@illinois.edu)

High Performance Computing finalists. All these papers address very specific problems using particular algorithms adapted to both the problem and the architecture. The majority of these codes use explicit time schemes, while a few of them use implicit ones with optimal (multigrid) solvers. In most of these papers massive parallelism is achieved for single-physics problems, in relatively simple geometries and on Cartesian meshes.

In this paper, we present three engineering-like cases of very different character: incompressible flow in the respiratory system, the turbulent reactive flow in a rotary kiln and non-linear solid mechanics coupled with electro-physiology of the human heart. In all cases, we use the same simulation tool: Alya, the BSC's in-house software. This paper describes the solution strategy and shows the performance on supercomputers up to 100K processors.

As a matter of fact, most references mentioned above show code performance in a larger core counts. So in order to fairly measure the novelty of this work, we must consider the following:

- The three examples presented here show many of the potentially complex aspects of a real-world engineering problem.
- Two of these problems are real multi-physics one, such as combustion or cardiac electro-mechanics.
- All of them involve both implicit and explicit schemes and we analyze the solver convergence properties all the way up to 100.000 processors.
- Two of them have complex geometries: the respiratory system and a complete cardiac bi-ventricular geometry.
- Meshes are unstructured, including different element types: prisms in boundary layers, tetrahedral, etc.
- Meshes are very large, up to more than four billion elements (4×10^9). We also describe our strategy for producing such meshes.
- Alya's algebraic solvers are programmed in-house, with no external libraries, seamlessly integrated to the solution strategies.
- The problems involve fluid mechanics, non-linear solid mechanics, excitable media, species and chemical reactions.
- All problems are run on the same supercomputer, which is made of general purpose hardware.

As a matter of fact and until now, **engineering**, **one-hundred thousand cores** and **supercomputer** were three concepts hardly found in the same sentence, unless in a negative connotation. This paper represents an effort to turn the negative sense into a positive one, leading the way toward Exascale computing in multiphysics engineering simulations.

This paper addresses the performance of Alya on supercomputers, running on up to 100.000 processors in Blue Waters, the sustained peta-scale system [31] hosted at the University of Illinois' National Center for Supercomputing Applications (NCSA). Blue Waters consists of traditional Cray XE6 compute nodes and accelerated XK7 compute nodes in a single "Gemini" interconnect fabric. Only XE6 nodes where used for this work, with each node containing two AMD "Interlagos" processors and a total of 16 floating point cores (NCSA, USA). Performance is measured through scalability when simulating coupled multi-physics problems in complex geometries coming from different domains.

2. Alya general view. Alya (see for instance [30, 15, 12, 5, 17]) is a simulation code developed at Barcelona Supercomputing Center (BSC-CNS) since 2004, whose main architects are authors GH and MV. Alya is not a born-sequential simulation code which was parallelized afterwards. Instead, it was designed from scratch as a multi-physics parallel code. It's main features are the following:

- It solves discretized partial differential equations (PDEs), preferring variational methods (particularly Finite Elements).
- Space discretisation is based on unstructured meshes, with several types of elements, such as hexaedra, tetraedra, prisms, pyramids... linear, quadratic...
- Both explicit and implicit time advance schemes are programmed.
- Depending on the case, staggered or monolithic schemes are programmed. However, staggered schemes with coupling iterations are preferred for large multi-physics problems.
- Parallelisation is based on mesh partitioning (for instance using Metis [24]) and MPI tasks, which is specially well-suited for distributed memory machines. On top of that, some heavy weight loops are parallelized using OpenMP threads. Both layers can be used at the same time in a hybrid scheme.
- Alya sparse linear algebra solvers are specifically developed, with a tight integration with the overall parallelisation scheme. There are no third-parties solver libraries required.
- Alya includes some geometrical tools which operate on the meshes for smoothing, domain decomposition or mesh sub-division. In particularly, the latter is a key tool for large-scale simulations [13].

Alya is organized in a modular way: *kernel*, *services* and *modules*, which can be separately compiled and linked. Each *module* represents a single set of Partial Differential Equations (PDE) for a given physical model. To solve a coupled multi-physics problem, all the required modules must be active and interacting following a well-defined workflow. Alya's *kernel* controls the run: it contains the solvers, the input-output workflow and everything related to the mesh and geometry. The kernel and the modules enable a given Physical problem to be completely solved The *services* are supplementary tools, notably the parallelisation service or the HDF5 writer. Kernel, modules and services have well-defined interfaces and connection points.

2.1. Computational Mechanics Equations. Let us establish the theoretical setup and very briefly summarize the transition from continuous problem formulation to a discrete one in Alya. Generally speaking, Alya deals with Computational Mechanics problems that can be modelled through conservation laws expressed as a set of partial differential equations:

$$\partial_t^* \Phi^\alpha = \partial_{x_i} F_i^\alpha = \partial_{x_i} C_i^\alpha + \partial_{x_i} K_i^\alpha$$

where Φ^α labels the unknown for the equation α of the set. F_i^α is the compact notation of the fluxes for each of the equations, being divided in two terms, C_i^α and K_i^α, for convenience. Pleae note that the temporal derivative $\partial_t^* \Phi^\alpha$ is starred to note that it can be of first (like in fluid flows or excitable media) or second order (like in solid mechanics or unsteady turbulenf flows). Latin subindices label the space dimensions of domain Ω. To these equations, boundary and initial conditions must be added depending on the problem under study.

The variational form is obtained by projection on a space W with its usual properties, where $\Psi \in W$ is the test function or the characteristic function depending on the context (finite elements, finite volumes, collocation, etc.). After integrating-by-parts some of the fluxes (say the K ones), we obtain

$$\partial_t^* \int \Psi^\alpha \Phi^\alpha d\Omega = \int \Psi^\alpha \partial_{x_i} C_i^\alpha d\Omega - \int \partial_{x_i} \Psi^\alpha K_i^\alpha d\Omega + \int \Psi^\alpha K_i^\alpha n_i d\partial\Omega. \tag{2.1}$$

Boundary conditions on the fluxes K are imposed through the last term, being $\partial\Omega$ the domain boundary and n_i its exterior normal vector. The interpolation space where the variational form solution is to be found is (in practice) the same as W, explicitly time independent. For that reason, the time derivative has been taken out of the space integral, with, for the sake of simplicity, the additional hypothesis that Ω is not changing with time. In any case, if the domain is varying in time some additional terms are computed, as in the case of Arbitrarian Lagrangian-Eulerian schemes. These equations govern problems in fluid mechanics, solid mechanics, chemical reactions, quantum mechanics, heat transfer, etc. for each multi-physics component of the problem.

The modelling equations contain all the "physics" of a given problem, and discretized in time and space in a certain way, they are programmed in what we called above a *module*. Therefore, the module's main task is to compute the elementary matrix and righ-hand-side of its corresponding set of equations, including all the numerical subtleties, boundary conditions, material models and so on. These matrices are assembled in a global matrix and a global right-hand-side vector, creating the algebraic system.

In Alya, the preferred method to discretize in space the weak form Equation 2.1 is the finite element method. When additional numerical stabilisation is required, we usually follow the so-called Variational MultiScale method (see for instance [14] for incompressible or [20] compressible flows). Time is discretized using finite differences to obtain either explicit or implicit schemes of different orders.

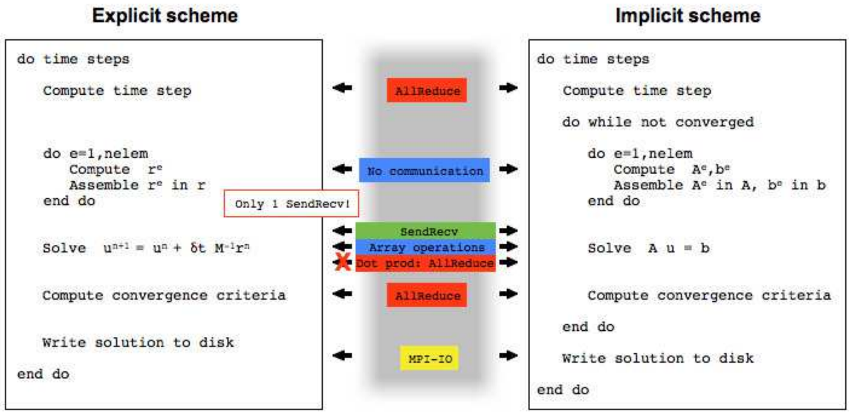
Time and space discretisation of the weak form leads to an algebraic system:

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{2.2}$$

These algebraic systems can be very large. In Alya, we prefer to solve them using iterative methods which are very well suited for parallel programming. According to the problem solved, we follow a wide range of iterative strategies. On one hand, the explicit schemes, which can be viewed as the most simple iterative method with a unique simple (Richardson) iteration per time step. On the other hand, iterative schemes such as GMRES, BiCGSTAB, CG [27] or Deflated CG [28, 19] are chosen depending on the case. Both explicit and implicit schemes are illustrated in Figure 1. Apart from the time loop, a linearisation loop may be necessary for non-linear problems when using implicit schemes.

2.1.1. Multiphysics and Coupling schemes. In the 2011 report by W. Gropp et al. *Multiphysics simulations: challenges and opportunities* [7], the authors profile a definition of multiphysics from a computational mechanics' point of view:

> Semantically, a multiphysics system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically conservation or constitutive laws. A major classification in such systems is whether



```
do time steps
   Compute time step

   do e=1,nelem
      Compute  $r^e$
      Assemble $r^e$ in r
   end do

   Solve  $u^{n+1} = u^n + \delta t\ M^{-1} r^n$

   Compute convergence criteria

   Write solution to disk
end do
```

```
do time steps
   Compute time step
   do while not converged
      do e=1,nelem
         Compute  $A^e, b^e$
         Assemble $A^e$ in A, $b^e$ in b
      end do

      Solve  A u = b

      Compute convergence criteria
   end do
   Write solution to disk
end do
```

FIG. 1. *Typical explicit and implicit schemes, with its basic MPI communication strategy.*

the coupling occurs in the bulk (e.g., through source terms or constitutive relations that are active in the overlapping domains of the individual components) or whether it occurs over an idealized interface that is lower dimensional or a narrow buffer zone (e.g., through boundary conditions that transmit fluxes, pressures, or displacements).

In Alya, multiphysics problems are defined by the *major classification* of the quote: multigrid, contact problems, multi-code coupling, overset meshes or adjoint-based optimisation, all of them lying within the definition. The wide context allows us to study general solutions, considering especially both the coupling algorithms and the parallel programming issues.

In this paper, we focus on the kind of multiphysics where all happens in the same domain and domain discretisation. In these cases, coupling is physical and achieved through coupling Alya modules. The main features are:

- All problems are solved in the same mesh, so no interpolation is required.
- The total number of degrees of freedom is equal to the number-of-nodes *times* the total number-of-variables that defines the problem. Each module covers a set of variables. For instance, when incompressible flow is coupled with thermal transport, the unknowns pressure, velocity (module 1) and temperature (module 2) are sufficient to define the problem.
- As the preferred scheme is the staggered one, the full problem matrix can be decomposed into blocks corresponding to each module solved or "physics" modelled. At each time step, the modules are advanced one after the other and, if required, coupling iterations can be performed to increase accuracy and/or robustness. When more than two modules are coupled, the iterative scheme can be indeed very complex, including sub-time-stepping, block grouping and relaxation. This allows using different time iteration schemes for each physic although on the same mesh, providing that they are all synchronized. For instance, if problem A uses a time step ten times smaller than problem B, for each problem B time step, 10 problem A time steps are run.
- Scalability is measured based on the total CPU-time used per time step. Therefore, if one of the modules is not properly scaling, it will strongly de-

grade the overall scalability.

2.2. Parallelisation layer. The parallelisation paradigm in Alya is a sub-structuring method, using a Master-Worker interaction model between the CPUs. Sub-structuring methods consist essentially in distributing the work among the Workers, leaving the Master in charge of general tasks like I/O. Most of the iterative solvers available in Alya are classical solvers like GMRES or Conjugate gradient, and their convergence do not depend on the number of CPUs when used in parallel. Preconditioners using coarse space corrections (Multigrid, Deflated Conjugate gradient) are implemented independently of the number of CPUs as well, so mesh partition has no effect in solver's convergence. This is not a restriction but a deliberate decision. The fact is that if a set of solver parameters is tuned to achieve convergence on a given number of CPUs, one would like to obtain the same convergence using more CPUs without any change in these parameters. This is a crucial point when considering industrial-realm complex simulations where the user usually does not necessarily have time to try different sets of parameters depending on the number of available CPUs. Things are much worse when considering multiphysics, where each sub-problem is solved with its own convergence strategy. Nevertheless, domain decomposition methods like Additive Restricted Schwarz (RAS), Block LU (one block per subdomain), Schur complement solvers, as well as subdomain dependent preconditioners like linelet [29] are also available in Alya. These solvers and preconditioenrs are generally used whenever classical solvers appear not to be robust enough.

2.2.1. Master-Worker strategy. All the details on the parallelisation of Alya can be found in papers such as [15, 11, 19]. We give here the general idea of the Master-Worker strategy. In the cases shown here, the Master reads the mesh and performs the partition of the element graph with METIS [24], an automatic mesh partitioner which balances the number of elements while minimising the subdomain boundary/interface surfaces, that is the communications. However, when initial meshes are very large, a parallel HDF5 format is preferred, so each Worker reads its corresponding part. For the pure MPI strategy, each MPI task will be in charge of each subdomain, which are the Workers. The Workers build the local matrices ($\mathbf{A}_i$) and right-hand side ($\mathbf{b}_i$), and are in charge of the resulting system solution in parallel. In the assembly stage, very few communications are needed between Workers and the scalability only depends essentially on the load balancing. Basically only few `MPI_AllReduce` are required to compute the solution's residual, critical time step, etc.

2.2.2. Communication types and scheduling. In the iterative solvers, two main types of communications are usually needed.

- Global communications via `MPI_AllReduce`, which are used to compute residual norms, time steps and scalar products involved in algebraic solvers;
- Point-to-point communications via `MPI_ISend` and `MPI_IRecv`, which are used in algebraic solvers when sparse matrix-vector (SMV) products are needed.

We mentioned earlier that the parallelisation of Alya is based on a sub-structuring approach in which most of the solvers and preconditioners are implemented independently of the number of subdomains. Therefore the parallel solution is, up to round off errors, the same as the sequential one at any moment because mesh partition is only used for distributing work without changing the sequential algorithm. Figure 1 shows these two types of communications in explicit and implicit schemes. The element loop consists of the local (to each subdomain) matrix and RHS assemblies and does not involve communication. Therefore, the parallel performance of the ex-

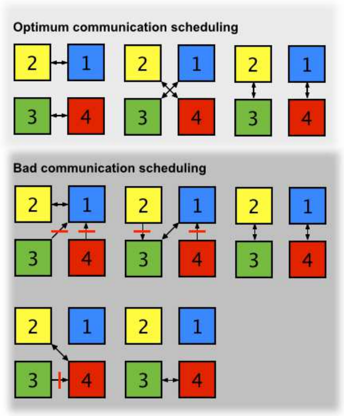

Optimum communication scheduling
Bad communication scheduling

FIG. 2. *Left: Scheduling strategy on a simple example. Top, optimum and communication scheduling in 3 steps. Bottom, Bad communication scheduling in 5 steps.*

plicit scheme is expected to be dominated by the load balance. Another key issue of communication is scheduling [2]. In Alya, a relatively simple but efficient scheme is used, based on coloring and build upon the adjacency graph of each subdomain. Colors group adjacencies that has no common communications which then can be scheduled in non-overlapping stages. Figure 2 shows the kind of problem that can arise when data transfer is not properly scheduled. In this case, four subdomains have to exchange data with all the others. The optimum scheduling is shown on the top part of the figure. On the bottom part, no scheduling is used and subdomains try to exchange their boundary data in a lexical order. In the first communcation step, subdomains 3 and 4 cannot send their data to subdomain 1, as this one is being exchanging data with subdomain 2.

2.2.3. Data structure. A specific data structure for distributed memory parallelisation has been used and can be briefly explained through a simple example illustrated in Figure 3. It shows an example of mesh partitioning into four subdomains (top left) and its corresponding node numbering (top right). In each Worker, interior nodes are first ordered. Then, boundary nodes (grey) are divided into *own boundary nodes* and *others bounday nodes*. The tag *own boundary nodes* cannot be repeated in more than one subdomain and are obtained by partitioning the subdomain boundary with METIS partitioner. The *own boundary node* definition is useful when scalar products are needed to avoid repeating the contribution of boundary nodal values. The nodes involved in scalar products are shown in Figure 3 (right) with a $\times$ sign, being 13 nodes in this particular case (bottom left). Finally, the nodes involved in the `MPI_ISend` and `MPI_IRecv` after a SMV are shown (bottom right).

2.2.4. Sparse matrix-vector product. Using the data structure and the scheduling introduced earlier, the Sparse Matrix Vector (SMV) product is computed together with a non-blocking send-receive as follows:

1. Perform local SMV product $\mathbf{y}_i = \mathbf{A}_i\mathbf{x}_i$ on boundary nodes;
2. Exchange $\mathbf{y}_i$ with neighbors using non-blocking `MPI_ISend` and `MPI_IRecv` according to the communication scheduling;
3. Perform local SMV product $\mathbf{y}_i = \mathbf{A}_i\mathbf{x}_i$ on interior nodes;
4. Synchronize the solution updates with `MPI_WaitAll`.

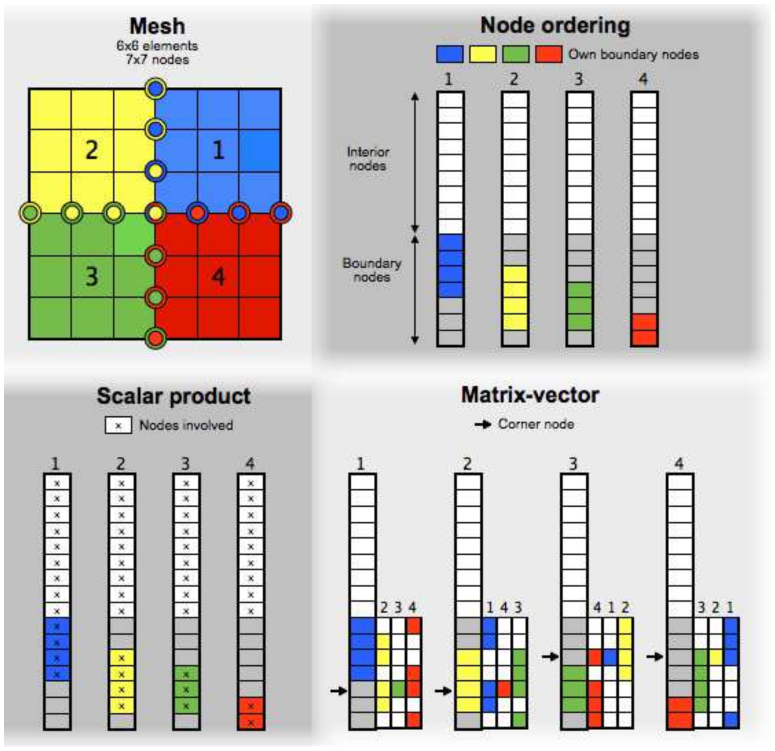


FIG. 3. *Organisation of data in Alya and types of communications.*

3. Multiphysics simulation examples. In this paper we analyse Alya's parallel behaviour through three different examples:

- The human respiratory system: transient incompressible flow.
- The kiln furnace: transient multi-species reacting flow using a low-Mach approximation of the Navier-Stokes equations.
- The electro-mechanical cardiac model: transient non-linear solid mechanics and excitable media.

In all cases we start with meshes in the range of a few million elements, which are progressively subdivided in parallel using the algorithm described in [13], in order to produce sufficiently large meshes to feed a large count of cores. As noted also in other works [23, 16], an automatic local mesh refinement tool should be a key feature in any engineering simulation tool. Once a good large mesh is generated, this tool can provide finer ones effortlessly, preserving the general structure of the original one. Keep in mind that with just one cycle, the elements count can be multiplied by eight. Before analysing the parallel performance, we proceed to briefly describe each of the examples and their associated numerical strategies. It is worth to know that two out of three cases represent the largest problem solved so far in its respective domain: combustion in kiln furnaces and electro-mechanical coupling in cardiac modelling.

3.1. The respiratory system - Implicit incompressible Navier-Stokes (Example 1). Computational simulation enables the mechanics of respiratory airflow to be explored in detail, with considerable potential benefits for health protection. Resolving the complex time-dependent flow in the large airways poses a severe challenge. Various compromises are generally made, such as restricting the portion of the airways considered and approximating the flow conditions or physics. In this example, the unsteady flow in a subject-specific model of the domain that extends from the face to the third branch of the bronchopulmonary tree is simulated.

The whole airway geometry was defined from a single subject, identified via retrospective examination of CT images obtained from clinical records at St Marys Hospital, Paddington, United Kingdom. Consent was obtained to use this data as the basis for airway segmentation and reconstruction. Segmentation of the airways was performed using the Amira package (TGS Europe) and required some manual intervention, particularly in the nasal airways. There, the fine bone structure challenges the resolution typical of data acquired under routine clinical protocols, but the fidelity of the reconstructed data was carefully checked by ENT surgeons. Translation of the coarse segmentation into a smooth surface was performed using in-house, curvature adapted smoothing procedures. Mesh generation was accomplished in stages, using the Gambit and TGrid packages (Ansys Ltd.). This work was done in collaboraiton with D. Doorly and A. Bates from Imperial College London (UK).

The solution of this problem involves the incompressible Navier-Stokes equations. The time discretisation is based on a second order Backwards Finite Differences (BFD) scheme and the linearisation is carried out using the Picard method. The space discretisation and stabilisation is based on the variational multiscale method (VMS) and is extensively described in [14]. At each time step and linearisation iteration, the following system is solved:

$$\begin{bmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{bmatrix} \tag{3.1}$$

where $\mathbf{u}$ and $\mathbf{p}$ are velocity and pressure nodal unknowns. The simultaneous solution of the complete system is usually referred to as the “monolithic” scheme. In order to avoid complex preconditioners to account for the velocity-pressure coupling involved in the monolithic system solution, an algebraic fractional alternative, introduced and described in [11] is used. This scheme enables to segregate the solutions of the velocity and pressure at the algebraic level, by solving the pressure Schur complement using an iterative method (herein the so-called Orthomin(1)).

This strategy offers two main advantages. Firstly, with respect to the monolithic scheme, one shot of the method involves the separated solution of a non-symmetric system for the momentum equation and a symmetric system for the pressure (a Laplacian) which accounts for the continuity equation. Therefore, specific solution schemes are used for each sub-system. The momentum equations usually converge very well, even with a simple diagonal preconditioner, solved with a GMRES or a BiCGSTAB. The continuity equation is solved with the Deflated Conjugate Gardient solver (DCG) [19], together with a linelet preconditioner when anisotropic boundary layers [29] are present. Secondly, with respect to classical fractional step methods, no fractional errors are introduced, guaranteeing that the solution converges to that of the monolithic case. What is important to note here is that the solution strategy which consists of solving the pressure Schur complement instead of attacking directly the monolithic scheme, should be understood as part of the algebraic solution strategy of the Navier-Stokes system, and not a fancy trick to get away from this scheme.

Figure 5 shows the strong scalability and efficiency for the respiratory system problem, for a 345M elements hybrid mesh (tetrahedra, prisms, pyramids). Scalability is measured comparing the CPU time taken to solve one simulation time step

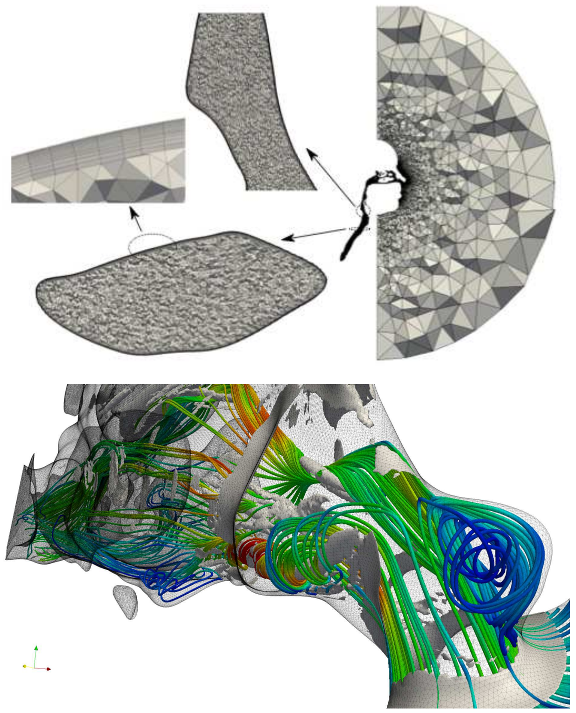


FIG. 4. *Respiratory system. Initial mesh (top) and close up view with streamlines.*

in an increasing number of processors. Efficiency higher than 0.8 is sustained on up to 24K processors and it starts to degrade due to the higher ration of communications/computing time if higher number of processors is employed. The bars plot at the botton gives the average-elements-per-core figure. This is very useful to establish a sweet spot, which depending on the problem physics and size, sets the number of processors you need to maintain a high efficiency. If we choose 0.80 as the limit, for this case the sweet spot is around 15.000 elements per core.

We have also performed a study of the efficiency of the deflated CG (so-called DCG), for a 550M element mesh. The DCG involves the solution of a coarse problem, using a direct solver, to accelerate the convergence by providing a mechanism to damping out the low frequency errors. In order to keep the number of iterations of the DCG solver constant when refining the mesh, one can increase the number of groups, as shown in Figure 6. It shows the decrease in number of iterations as a function of the number of groups which is the size of the coarse problem solved by the DCG. By multiplying the number of groups by four (from 500 to 2000, i.e. making finer the coarse problem), the number of iterations required to achieve the convergence criterion is reduced by a factor 1.7. The second plot shows the relative efficiencies of the solver as a function of the number of groups and number of CPUs. The DCG solver involves an reduction operation of the size of the number of groups, and therefore it is expected that its efficiency decreases drastically with the numbers of groups and CPUs [19]. This is confirmed by observing Figure 6. For example, on 10240 CPUs, the CPU time is decreased by a factor 1.4 instead of 1.7 in number of iterations. On 2560 CPUS, this factor is 1.66, which is almost optimum.

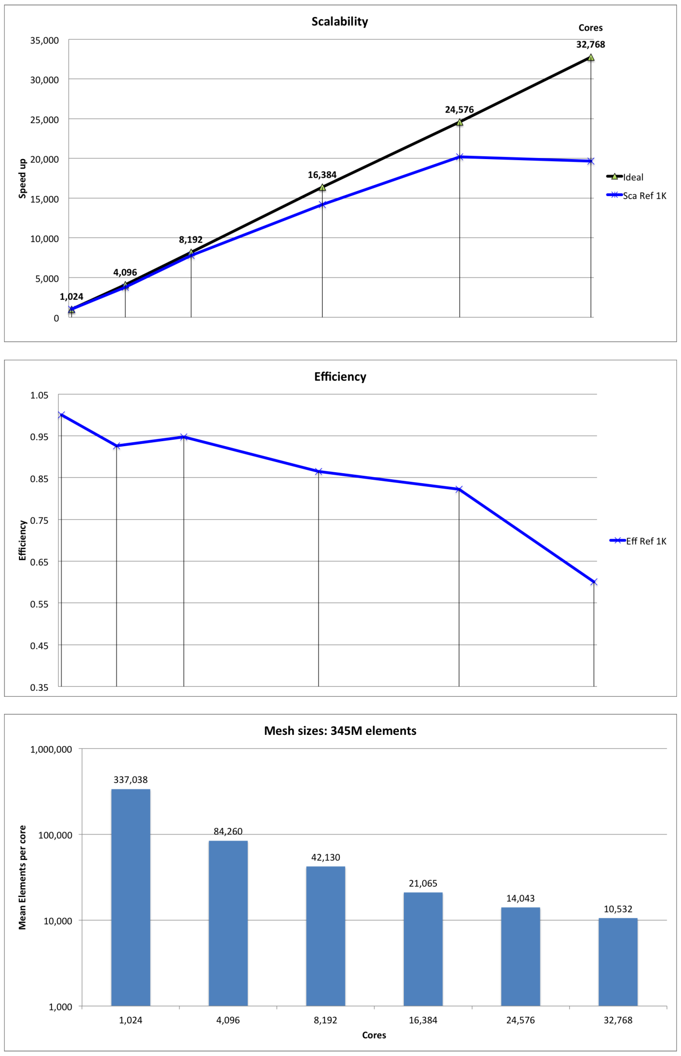


FIG. 5. *Respiratory system. Scalability, efficiency and average-elements-per-core.*

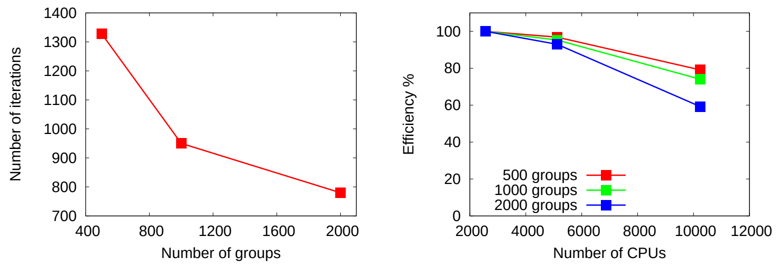


FIG. 6. *Respiratory system. Deflated Conjugate Gradient behaviour: number of iterations (left) and relative efficiency.*

3.2. The kiln furnace - Implicit incompressible Navier-Stokes, chemical reactions and combustion (Example 2). At the heart of the cement production process lies the kiln, a tilted rotary oven where raw materials are heated up to reaction temperatures to form small pellets called clinker, which is then ground to make cement powder [21]. In addition to reducing its raw material consumption, improving the efficiency of cement kiln is a main concern of the industry, as the kiln is the main consumer of energy in the production process.

The kiln rotates at a fixed frequency of about 5 rpm. Its length ranges between 50 and 180 meters, and its diameter between 2 and 4 meters. On the high part of the kiln the raw material is fed in, sometimes as a dry powder and others as a wet sludge, where it begins the process of clinkerisation. On the lower part of the kiln a large burner ejects fuel, typically pulverised coal or waste material. The burner has a primary air injector, and secondary and tertiary injectors that add swirl motion to the flow acting as a flame stabilisation mechanism, which can be up to 10 meters long. The walls of the kiln are a mixture of refractory bricks and metal. Furthermore, cement stuck to the walls around the lower third of the oven forms a coating shell critical for operation of the kiln.

In this example, the gaseous phase of a rotary kiln is simulated using large-eddy simulation (LES). The numerical scheme is based on a staggered algorithm that solves the Navier-Stokes equations at the low-Mach limit, the enthalpy transport equation expressed in terms of temperature and the transport and reaction of the chemical species. In this case, we consider six chemical species to represent the oxidation of methane.

The flow equations are solved using a second order Backwards Differences Scheme (BDF) with a Newton-Raphson linearisation method. The momentum and continuity equations are solved with unsymmetric and symmetric iterative solvers respectively. For the momentum equations, the GMRES is considered to be the best choice while the Conjugate Gradient (CG) or Deflated CG are the choices for the continuity equation [19, 15]. The GMRES solver is also used to solve the enthalpy transport and the species mass fractions. A block Gauss-Seidel iterative method is employed to solve the species mass fraction independently until the targeted convergence. Each single species equation is solved with the GMRES as well.

In this problem, [13], four different levels of mesh subdivision have been considered, referred to here as $h = 1$, $1/2$, $1/4$, $1/8$, from the coarsest to the finest, respectively. The numbers of elements are $8.25M$, $66.0M$, $528M$ and $4.22B$, respectively. The time step size is computed as a multiple of the critical time step. Figure 8 shows the average time step values of the first ten steps, computed for the four meshes.

Figure 9 shows the strong scalability and efficiency for the kiln furnace simulation. The plots show the total scalability, measured summing up the CPU times for all the Physical problems solved, namely low Mach, temperature and chemical reactions. In this example we show the results for two meshes: 528M (called DIV2) and 4.22B elements (called DIV3). For DIV2 (labelled "DIV2 Ref 1K") the scalability is measured all the way from 1024 up to 100K cores, with the sweet spot around 16K average elements per core. Beyond that point, efficiency falls below 0.80. For DIV3, i.e. the largest mesh, we run the last three points of the plot, 32768, 65536 and 100000 cores (labelled "DIV3 Ref 32K"). In this case, we use the CPU time for 32768 as the scalability normalising value. Finally, and in order to be fair with the comparison, we have added a third curve: the scalability and efficiency plots for DIV2 also normalised with 32768 instead of 1024 (labelled "DIV2 Ref 32K"). As expected,

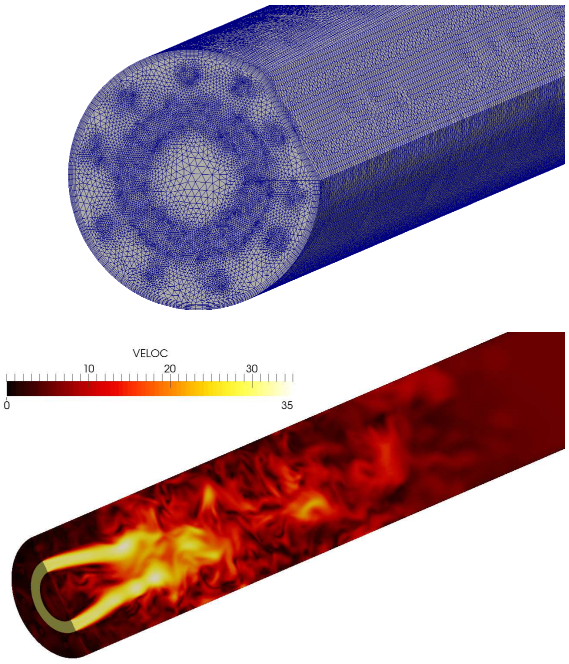


FIG. 7. *Kiln furnace. Original mesh close-up (top) and cut with velocity contours.*

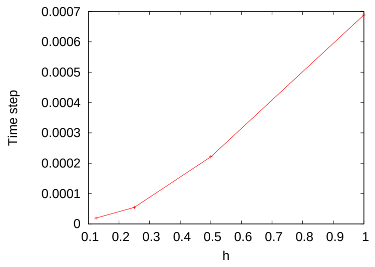


FIG. 8. *Kiln furnace. Time step values for the four meshes.*

"DIV2 Ref 32K" is very close to a translation upwards of "DIV2 Ref 1K". On the other hand, "DIV3 Ref 32K" presents a much better scalability and efficiency, with a sustained large efficiency up to 100.000 cores.

Apart from the scalability, we present some convergence results of the solvers of the momentum (GMRES) and continuity equations (DCG). In the present case, the number of DCG groups in maintained constant at a rather small value of 200 (i.e. for the meshes sizes considered in this example). Figure 10 compares the converges of the first iterations for the momentum and continuity equations, using the four meshes. We observe that the momentum equations converge quite rapidly and similarly for the four meshes. This is because the time step decreases with the mesh size, increasing in this way the diagonal terms of the momentum equations, and leaving their condition numbers almost unchanged. This is not the case of the DCG, which convergence degrades with the mesh size. However, we observe that even with a very small number of groups, the method still converges. In Figure 11 we plotted the rates of convergence

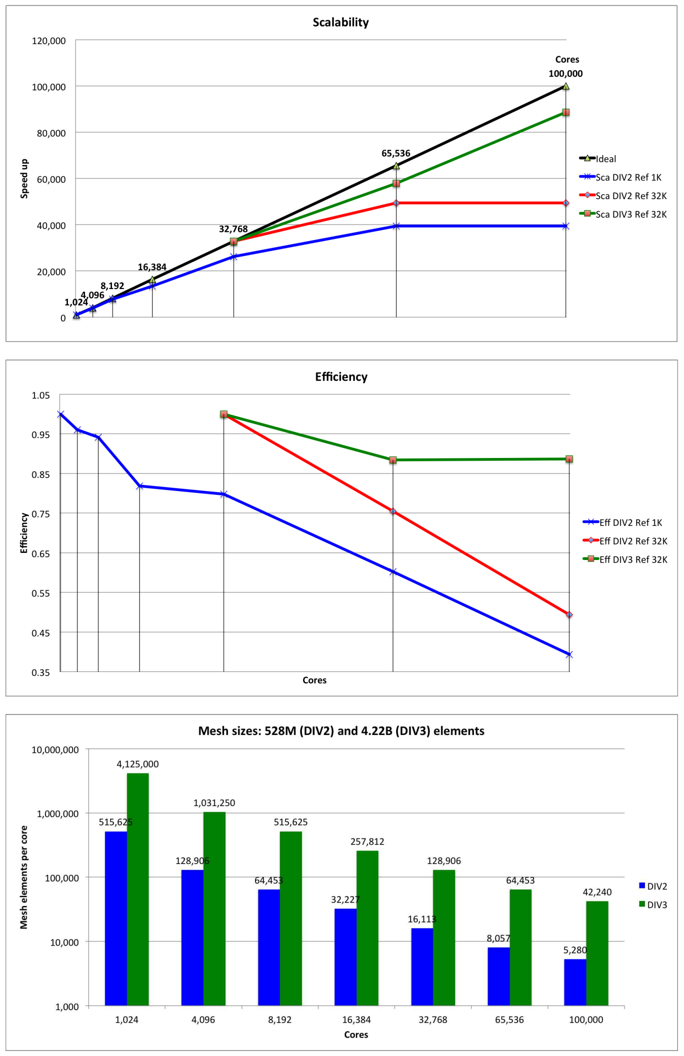


FIG. 9. *Kiln furnace. Scalability, efficiency and average-elements-per-core.*

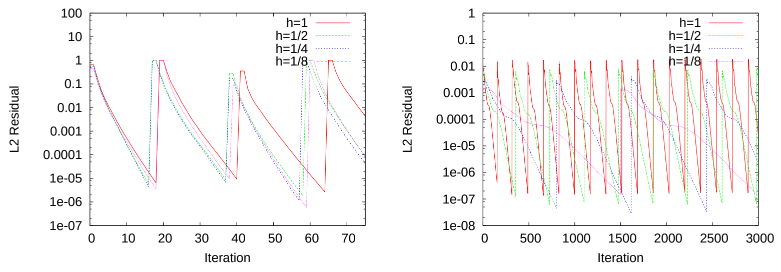


FIG. 10. *Kiln furnace. Convergence of the GMRES solver for the momentum equation (Left) and the DCG solver for the continuity equation.*

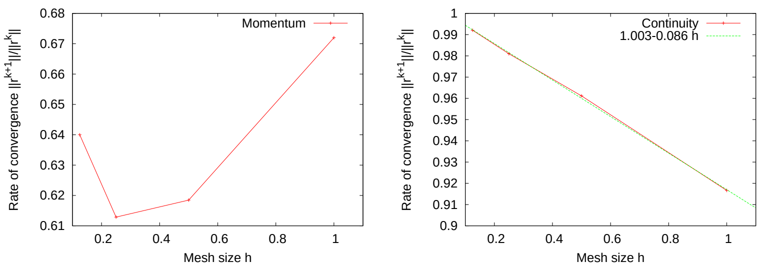


FIG. 11. *Kiln furnace. Rate of convergence of the momentum (Left) and continuity (Right) equations.*

of the GMRES and DCG solvers. These figures can be useful to predict the number of iterations required to achieve a given residual reduction according to the mesh size. In addition, we plotted an approximate linear fit to the rate of convergence of the DCG for this particular case. In the case of the momentum equation, convergence rate degradation is not uniform. From experience, we can assume that this is due to the subdivision mesh scheme we are using, in which after 3 or 4 subdivision cycles can somewhat deteriorate mesh quality.

3.3. The electro-mechanical cardiac model - Explicit non-linear solid mechanics and excitable media (Example 3). The Alya Cardiac Computational Model is explained in papers such as [17, 34, 33]. Simulating a heart beat is a complex, multiphysics and multiscale problem, where many scales are coupled together covering different orders of magnitude: from descriptions of electrical propagation, cells arrangement into a spatial description and up to the geometry of the cardiac chambers [18].

Mathematical models of the heart have been developed during the last decades [32] ranging from the molecular point of view to the anatomical level of the organ. At the organ-level, the cardiac computational model requires the solution of the electrical component as a non-linear reaction-diffusion system, i.e. an excitable media model; and the mechanical component, which produces the deformation using a coupling scheme to link both problems together. Physiologically, the electrical activation of the heart triggers the mechanical contraction via the concentration of Calcium. Mechanical models representing the active deformation of the tissue are based on protein interaction like actin-myosin and events in cardiac myofilaments on single cells, or phenomenological electrical propagation relationships that mimic the action potential. The choice of detail of the mathematical models depend on the specific application.

The electrical propagation is modelled as a reaction-diffusion equation. The diffusion term is locally anisotropic due to the fiber-like complex structrure of the cardiac muscle. The reaction term is the non-linear ion channel behaviour of the cardiac cells. Fibers are defined as a nodal field, which arise from either mathematical modelling, after histological sectioning and measurement of their orientations; or from Diffusion Tensor Magnetic Resonance Imaging, assuming that muscular fiber orientation correlates well with water diffusion direction. The reaction-diffusion system produces a sharp depolarisation advancing front. In this example, a model published in 1969 by FitzHugh and Nagumo is used. The second Physical problem is the muscular pumping action. From the mechanical point of view, the myocardium is here considered as compressible. The material is non-linear hyper-elastic, with anisotropic behaviour ruled by the fiber structure. In this work, we use a transversally isotropic material based on [10] and presented in [17].

Let us briefly describe the mechanical problem, solved in the same discretisation of the electical activation. The dynamical mechanical equations are written in a total-Lagrangian formulation. The Cauchy stress $\boldsymbol{\sigma} = J^{-1}\boldsymbol{P}\boldsymbol{F}^T$, related to the first Piola-Kirchoff P_{iJ} and the deformation gradient $F_{iJ} = \dfrac{\partial x_i}{\partial X_J}$ and its Jacobian J, allows to define the material model. Stress is developed in two parts: active and passive:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{pas} + \sigma_{act}(\lambda, [Ca^{2+}])\boldsymbol{f} \otimes \boldsymbol{f} \tag{3.2}$$

The passive part is governed by a transverse isotropic exponential strain energy function $W(b)$ that relates the Cauchy stress σ to the right Cauchy-Green deformation b. The passive stress is then

$$\begin{aligned} J\boldsymbol{\sigma}_{pas} = (a\ e^{b(I_1-3)} - a)\boldsymbol{b} + 2a_f(I_4-1)e^{b_f(I_4-1)^2}\boldsymbol{f} \otimes \boldsymbol{f} \\ + K(J-1)\boldsymbol{I} \end{aligned} \tag{3.3}$$

The strain invariant I_1 represents the non-collagenous material while strain invariant I_4 represents the stiffness of the muscle fibers, and a, b, a_f, b_f are parameters to be determined experimentally. K sets the compressibility. Vector $\boldsymbol{f}$ defines the fiber direction.

Now, we briefly describe electro-mechanical coupling, which depends on ionic concentration in the tissue, specially the calcium. Depending on the model, ion concentrations (Ca^{2+}, Na^{+}, K^{+}...) in the cellular membrane can be computed as function of the electrical activation (and, eventually, the mechanical contraction). In the simplest model, which is used here, electro-mechanical coupling is modelled as follows. The mechanical deformation is the result of the active tension generated by the myocytes as accounted for in 3.2. The model assumes that the active stress is produced only in the direction of the fiber and depends only on the calcium concentration of the cardiac cell [22]:

$$\sigma_{act} = \alpha\ \frac{[Ca^{2+}]^n}{[Ca^{2+}]^n + C_{50}^n}\sigma_{max}(1 + \beta(\lambda_f - 1)). \tag{3.4}$$

In this equation, C_{50}^n, σ_{max}, λ_f and β are model parameters, which are fitted to ensure the proper propagation speed, coupling force, etc.

We have also introduced a parameter $0 < \alpha < 1$ to calibrate the amount of active stress and measure its sensitivity.

In order to capture all the required time scales, small time steps are needed. Therefore, in cardiac mechanics simulations explicit schemes for time integration are preferred. Figure 12 shows a snapshot on the electromechanical propagation, closing up on the mesh. The original mesh is made of a bit more than 6M elements. As in the other two examples, the original mesh goes through two and three subdivision cycles following [13], reaching 427 millions and 3.4 billions tetrahedra respectively.

Figure 13 shows the strong scalability and efficiency for the cardiac electromechanical model. Again, the plots show the total scalability, measured summing up the CPU times for the Physical problems solved, namely electrophysiology and solid mechanics. As in the kiln example, we show here the results for two meshes: 427M

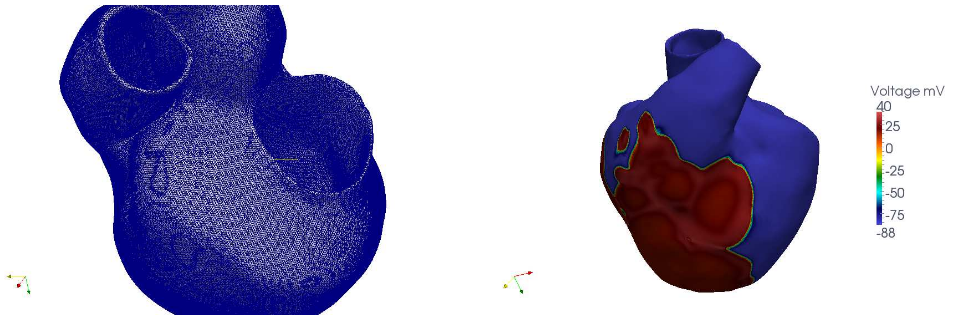


FIG. 12. *Cardiac electro-mechanical model. Initial mesh (left) and electrophysiology activation potential snapshot.*

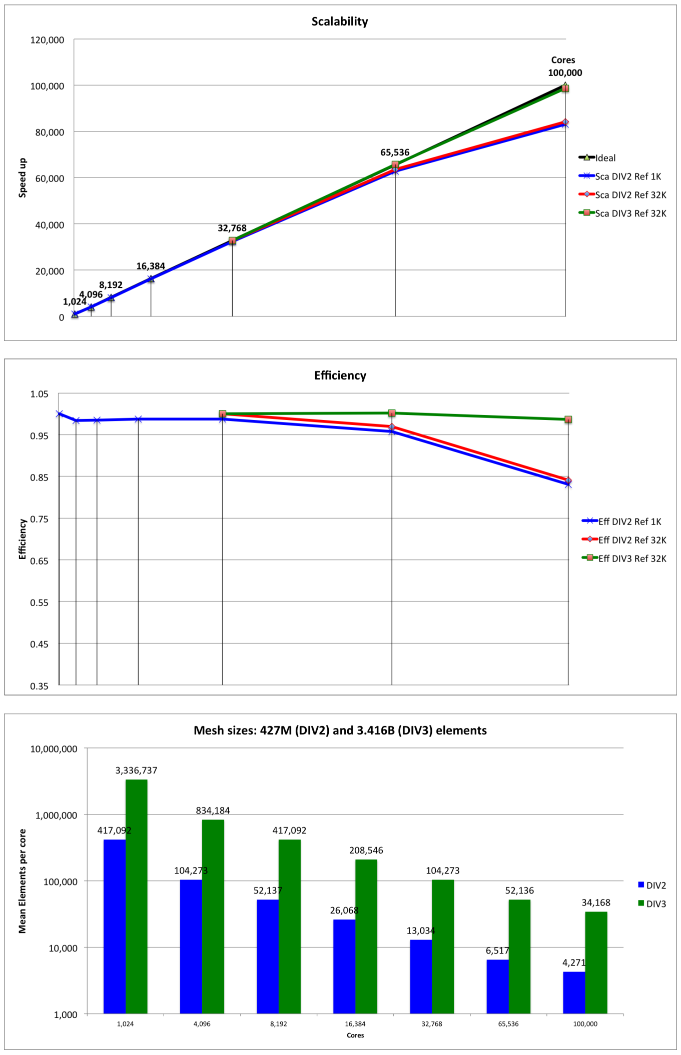


FIG. 13. *Cardiac electro-mechanical model. Scalability, efficiency and average-elements-per-core.*

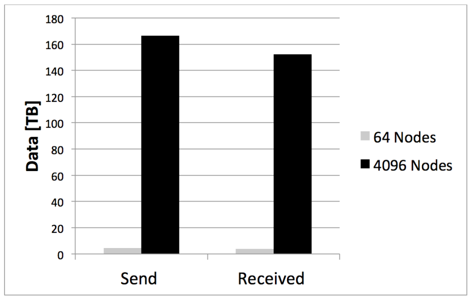


FIG. 14. *Cardiac electro-mechanical model. Data traffic in Blue Water's Gemini interconnects.*

(called DIV2) and 3.416B elements (called DIV3). Using the same approach as above, for DIV2 (labelled "DIV2 Ref 1K") the scalability is measured all the way from 1024 up to 100K cores. In this case, the sweet spot is lower than 4K elements per core. The reason is that for the explicit schemes communication needs per time step are much lower. As in the kiln example, for DIV3, i.e. the largest mesh, we run the last three points of the plot, 32768, 65536 and 100000 cores, using as the scalability normalising value the CPU time obtained for 32768 (labelled "DIV3 Ref 32K"). We have added the scalability and efficiency plots for DIV2, but now normalising with 32768 instead of 1024 (labelled "DIV2 Ref 32K"). Again and as expected, "DIV2 Ref 32K" is very close to a translation upwards of "DIV2 Ref 1K". On the other hand, "DIV3 Ref 32K" presents a much better scalability and efficiency, with a sustained large efficiency up to 100.000 cores. A communication measure is shown in Figure 14, which is the total measured data traffic to and from Gemini interconnects on Blue Waters when solving this problem. A sharp increase in the interconnect traffic by over 30 times is observed going from 64 (1024 cores) to 4096 nodes (65,536 cores).

4. Conclusions and Future Lines.

This article aims to show that engineering codes, if properly coded, optimised and ported, can take a full advantage of a peta-scale architecture, such as Blue Waters, and opening the way towards exascale computing. This demonstrates the feasibility of efficiently solving extreme size multiphysics simulations for the engineering realm by enabling the creation of large, high-fidelity models. These models will yield accurate and detailed insights into the complex behavior of engineering and biological processes. The paper prepares the ground pointing at what to expect when this kind of coupled multi-physics simulations achieves exa-scale. Some of the issues will be similar, but as a larger scale, such as meshing, solver convergence or data analysis, aspects on which we can advance with confidence thanks to the results shown. However, we are aware of the new realm we are facing to and the difficulties ahead.

In this paper, we have presented the simulation strategy of Alya, a multiphysics solver designed to run efficiently on tens of thousands of processors, and especially well-suited for engineering simulations. The NCSA's sustained peta-scale system of Blue Waters has shown its potential with large-scale Alya runs maintaining a high parallel efficiency on 100,000 cores in multiphysics cases. The chosen examples cover the largest possible range of features such a code should have. We solved coupled multi-physics incompressible fluid mechanics, combustion and thermal flow, non-linear solid mechanics and excitable media. We simulated meshes up to billions of elements. We used both explicit and implicit schemes, showing scalability plots for both cases and analysing solver convergence for the implicit ones. We used non-structured hybrid meshes combined with a mesh sub-division strategy. Finally, we have assessed the examples in terms of scalability and parallel efficiency, with special care on the solvers.

Such large-scale problems represent a completely unexplored territory, revealing new issues. Solution strategies must be adapted to take advantage of supercomputers. In the case of the Navier-Stokes equations, an algebraic split strategy has allowed the use of relatively classical iterative solvers with very good convergence and parallel performances for very large unstructured and hybrid meshes. But sufficient load is necessary to keep parallel efficiencly as high as possible. The efficiency obtained on some numerical examples gives us some lower bound estimation, depending on the physics and numerical schemes.

Scalability is the first step. Next, solver convergence must be further and deeper analysed. We show in this study to what extent Alya strategy is appropriate, but there are plenty of issues still to be treated. One key problem is postprocessing. For problems of this size, tools such as HDF5 are very important, but their behaviour on these grounds must be assessed. In the case of multi-physics coupling with non-overlapping subdomains (contact problems, fluid-structure interactions, heat transfer, etc.) parallel implementation presents an added difficulty: point-to-point communication. Preliminary results are very encouraging.

Considering accelerators, let us remark that in recent years, substantial efforts were undertaken to adapt computational sparse methods for evolving GPU systems. We are testing GPU-based solver and mathematical libraries with Alya such as CuBLAS [4] and Paralution [6] on XK7 nodes of Blue Waters. We also plan to test a massively parallel direct solver library WSMP [8] as a preconditioner for the iterative solvers in Alaya. WSMP has shown enough scalability and robustness to perform with multi-million equation problem size on many thousands of cores [9]. Besides, Alya has been tested in Intel Xeon Phi systems with sustained scalability specially using the MPI parallelisation paradigm and virtually no effort in porting [35]. Further research in these two lines will be carried out and reported.

Acknowledgment. The authors would like to thank the following fellow researchers and institutions:

- The Private Sector Program at NCSA and the Blue Waters sustained-petascale computing project-supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois.
- Denis Doorly and Alister Bates (Imperial College London, UK), collaborators of the airways study. Part of this work was financed by European PRACE Type B/C projects.
- The heart geometry was provided by Dr. A. Berruezo (Hospital Clnic de Barcelona) in collaboration with R. Sebastian (UVEG) and O. Camara (UPF), partially financed through project TIN2011-28067 from MINECO, Spain.
- Part of the cardiac model development was financed by the grant SEV-2011-00067 of Severo Ochoa Program, awarded by the Spanish Government.
- Part of the kiln model development was financed by the European Commission in the framework of the FP7 Collaborative project "Advanced Technologies for the Production of Cement and Clean Aggregates from Construction and Demolition Waste (C2CA)", Grant Agreement No 265189.

REFERENCES

[1] T. Aoki and T. Shimokawabe, *Large-scale numerical weather prediction on gpu supercomputer*, in GPU Solutions to Multi-scale Problems in Science and Engineering, D. A. Yuen, L. Wang, X. Chi, L. Johnsson, W. Ge, and Y. Shi, eds., Lecture Notes in Earth System Sciences, Springer Berlin Heidelberg, 2013, pp. 261–270.

[2] P. Brucker, *Scheduling Algorithms*, Springer, fourth ed., 2003.

[3] C. Burstedde, G. Stadler, L. Alisic, L. Wilcox, E. Tan, M. Gurnis, and O. Ghattas, *Large-scale adaptive mantle convection simulation*, Geophysical Journal International, 192 (2013), pp. 889–906.

[4] C. G. B. L. by NVIDIA. November 2013, *http://docs.nvidia.com/cuda/cublas/index.html*.

[5] B. Eguzkitza, G. Houzeaux, R. Aubry, H. Owen, and M. Vázquez, *A parallel coupling strategy for the chimera and domain decomposition methods in computational mechanics*, Computers & Fluids, (2013).

[6] P. L. for Iterative Sparse Methods on CPU and G. N. 2013, *http://www.paralution.com*.

[7] W. Gropp and J. Magerlein, *Multiphysics simulations: challenges and opportunities*, Technical Report ANL/MCS-TM-321, Argonne National Laboratory, 2011.

[8] A. Gupta, *Wsmp : Watson sparse matrix package (part-i: Direct solution of symmetric sparse systems)*, Technical Report RC 21866, IBM T. J. Watson Research Center. Yorkton Heights, NY, 2013.

[9] A. Gupta, S. Koric, and T. George, *Sparse linear solvers on massively parallel machines*, in ACM/IEEE Conference on High Performance Computing SC 2009, Portland, Oregon, USA November 14-20, 2009.

[10] G. A. Holzapfel and R. W. Ogden, *Constitutive modelling of passive myocardium: a structurally based framework for material characterization*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, (2009).

[11] G. Houzeaux, R. Aubry, and M. Vázquez, *Extension of fractional step techniques for incompressible flows: The preconditioned orthomin(1) for the pressure schur complement*, Computers & Fluids, 44 (2011), pp. 297–313.

[12] G. Houzeaux, R. de la Cruz, H. Owen, and M. Vázquez, *Parallel uniform mesh multiplication applied to a navier-stokes solver*, Computers & Fluids, In Press (2013).

[13] G. Houzeaux, R. de la Cruz, H. Owen, and M. Vázquez, *Parallel uniform mesh multiplication applied to a navier-stokes solver*, Computers and Fluids, 80 (2013), pp. 142–151.

[14] G. Houzeaux and J. Principe, *A variational subgrid scale model for transient incompressible flows*, Int. J. Comp. Fluid Dyn., 22 (2008), pp. 135–152.

[15] G. Houzeaux, M. Vázquez, R. Aubry, and J. Cela, *A massively parallel fractional step solver for incompressible flows*, J. Comput. Phys., 228 (2009), pp. 6316–6332.

[16] J.Hoffman, J.Jansson, R. de Abreu, C.Degirmenci, N.Jansson, K.Mller, M.Nazarov, and J. Sphler, *Unicorn: parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction for deforming domains and complex geometry*, Computers and Fluids, 80 (2013), pp. 310–319.

[17] P. Lafortune, R. Arís, M. Vázquez, and G. Houzeaux, *Coupled electromechanical model of the heart: Parallel finite element formulation*, International Journal for Numerical Methods in Biomedical Engineering, 28 (2012), pp. 72–86.

[18] I. LeGrice, P. Hunter, A. Young, and B. Smaill, *The architecture of the heart: a data based model*, Phil. Tans. R. Soc. Lond., 359 (2001), pp. 1217–1232.

[19] R. Löhner, F. Mut, J. Cebral, R. Aubry, and G. Houzeaux, *Deflated preconditioned conjugate gradient solvers for the pressure-poisson equation: Extensions and improvements*, Int. J. Numer. Meth. Engn., 87 (2011), pp. 2–14.

[20] S. Marras, M. Moragues, M. Vázquez, O. Jorba, and G. Houzeaux, *A variational multiscale stabilized finite element method for the solution of the euler equations of nonhydrostatic stratified flows*, Journal of Computational Physics, 236 (2013), pp. 380 – 407.

[21] K. S. Mujumdar and V. V. Ranade, *Simulation of rotary cement kilns using a one-dimensional model*, Chem. Eng. Res. Des., 84 (2006), pp. 165–177.

[22] S. A. Niederer, P. J. Hunter, and N. P. Smith, *A quantitative analysis of cardiac myocyte relaxation: A simulation study*, Biophysical Journal, 90 (2006), pp. 1697–1722.

[23] N.Jansson, J.Jansson, and J.Hoffman, *Framework for massively parallel adaptive finite element computational fluid dynamics on tetrahedral meshes*, SIAM J. Sci. Comput., 34 (2012), pp. C24–C41.

[24] M. F. of Multilevel Partitioning Algorithms, *http://glaros.dtc.umn.edu/gkhome/views/metis*.

[25] T. Okamoto, H. Takenaka, T. Nakamura, and T. Aoki, *Accelerating large-scale simulation of seismic wave propagation by multi-gpus and three-dimensional domain decomposition*, in GPU Solutions to Multi-scale Problems in Science and Engineering, D. A. Yuen, L. Wang,

X. Chi, L. Johnsson, W. Ge, and Y. Shi, eds., Lecture Notes in Earth System Sciences, Springer Berlin Heidelberg, 2013, pp. 375–389.

[26] S. Pronk, S. Pall, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. Shirts, J. Smith, P. Kasson, D. van der Spoel, B. Hess, and E. Lindahl, *Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit*, Bioinformatics, (2013).

[27] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[28] Y. Saad, J. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926.

[29] O. Soto, R. Löhner, and F. Camelli, *A linelet preconditioner for incompressible flow solvers*, Int. J. Num. Meth. Heat Fluid Flow, 13 (2003), pp. 133–147.

[30] A. System, *http://www.bsc.es/alya*.

[31] B. W. S. P. C. System, *https://bluewaters.ncsa.illinois.edu*.

[32] N. A. Trayanova and J. J. Rice, *Cardiac electromechanical models: from cell to organ*, Frontiers in Computational Physiology and Medicine, (2011).

[33] M. Vázquez, R. Aris, J. Aguado-Sierra, G. Houzeaux, A. Santiago, M. Lpez, P. Crdoba, M. Rivero, and J. Cajas, *Alya red ccm: Hpc-based cardiac computational modelling*, in Selected Topics of Computational and Experimental Fluid Mechanics, J. Klapp, G. Ruiz Chavarria, A. Medina Ovando, A. López Villa, and L. D. G. Sigalotti, eds., Environmental Science and Engineering, Springer International Publishing, 2015, pp. 189–207.

[34] M. Vázquez, R. Arís, G. Houzeaux, R. Aubry, P. Villar, J. Garcia-Barnós, D. Gil, and F. Carreras, *A massively parallel computational electrophysiology model of the heart*, International Journal for Numerical Methods in Biomedical Engineering, 27 (2011), pp. 1911–1929.

[35] M. Vázquez, G. Houzeaux, F. Rubio, and C. Simarro, *Alya multiphysics simulations on intels xeon phi accelerators*, in High Performance Computing, G. Hernández, C. J. Barrios Hernández, G. Diaz, C. Garcia Garino, S. Nesmachnow, T. Pérez-Acle, M. Storti, and M. Vázquez, eds., vol. 485 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2014, pp. 248–254.

[36] M. Winkel, R. Speck, H. Hbner, L. Arnold, R. Krause, and P. Gibbon, *A massively parallel, multi-disciplinary barneshut tree code for extreme-scale n-body simulations*, Computer Physics Communications, 183 (2012), pp. 880 – 889.

[37] R. Yokota, L. Barba, T. Narumi, and K. Yasuoka, *Petascale turbulence simulation using a highly parallel fast multipole method on {GPUs}*, Computer Physics Communications, 184 (2013), pp. 445 – 455.