CSMA 2026
17ème Colloque National en Calcul des Structures
18-22 Mai 2026, Presqu'île de Giens (Var)

MANTA: an industrial-strength open-source high performance explicit and implicit multi-physics solver

Olivier Jamond[2], Nicolas Lelong[2], Thomas Helfer[1], Benoit Prabel[2], Adrien Jaccon[2], Antoine Motte[2], Christopher Nahed[2], Pascal Bouda[2], Louis Schuler[3], Mustapha Ellouze[3], Thomas Douillet-Grellier[3]

[1] *CEA, DES/IRESNE/DEC/SESC/LMCP, Département d'Études des Combustibles, Cadarache, France*
[2] *Université Paris-Saclay, CEA, Département de Modélisation des Systèmes et Structures, 91191, Gif-sur-Yvette, France*
[3] *EDF R&D, ERMES, 91120 Palaiseau, France*

Résumé — MANTA (Mechanical Numerical Toolbox for advanced Application) is an open-source effort from the French Alternative Energies and Atomic Energy Commission (CEA) and EDF (Électricité De France) to develop a multiphysics solver for quasi-static and fast-transient simulations of fluids and solids. MANTA aims to replace the 40 years-old Cast3M and Europlexus solvers and provide larger physical modeling abilities using up to date technologies.
Mots clés — finite-elements, finite-volume, HPC, structural mechanics, compressible fluid mechanics, generic PDE solver.

Introduction

MANTA (Mechanical Numerical Toolbox for advanced Application) is an open-source effort from the French Alternative Energies and Atomic Energy Commission (CEA) and EDF (Électricité De France). to develop a multiphysics solver for quasi-static and fast-transient simulations in fluids and solids [**?**]. MANTA aims to replace the 40 years-old Cast3M and EUROPLEXUS solvers and provide larger physical modeling abilities using up to date technologies.

The project has been designed to meet the following objectives :

- quality assurance, robustness and reliability compatible with safety-critical studies in the nuclear industry,
- high performance computing,
- maintainability over decades,
- ease of use for mechanical engineers or researchers,
- extensibility for rapid prototyping of new physical models and boundary conditions,
- generic and flexible to be used by researchers in the field of numerical methods,
- clean and simple Application Programming Interface (API) in C++ and python for coupling with external codes or integration with domain specific numerical platforms.

MANTA targets two main kinds of users :

→ The mechanical engineers or researchers which exploit the output of numerical simulations to design or analyse physical systems of interest. In view of such a user, MANTA provides a so-called end-user layer which offers a clean and easy API (both in C++ and python). Most numerical details are hidden by default. Also, a very important point is that its API is meant to be very stable in time.

→ The researcher in the field of numerical methods which would like to implement and test various algorithms. The MANTA so-called core-layer provides a generic and flexible way to implement a new unstructured-mesh-based numerical method dealing with a given set of Partial Differential Equations (PDE).

This paper is organized in two parts. Section 1 illustrates some abilities of the current version of MANTA in solid and fluid mechanics. Section 2 provides an overview of MANTA's architecture.

1 Overview of MANTA abilities

The Figure 1 gives an overview of some MANTA's available functionalities. A few applications are detailed thereafter in order to highlight the modeling abilities and the numerical performances of MANTA.

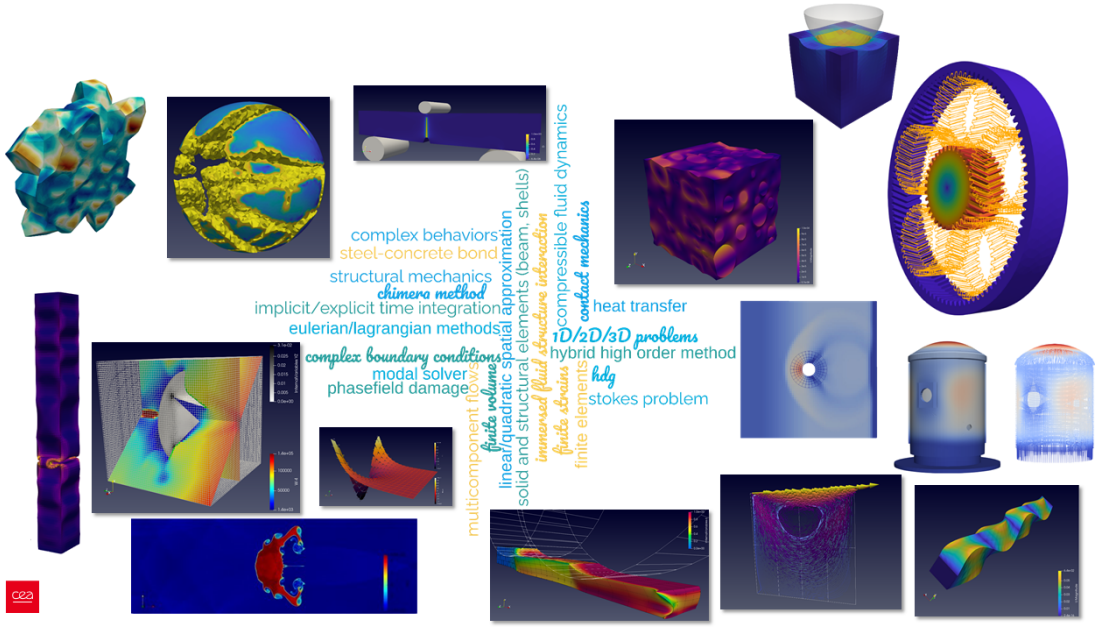

FIGURE 1. *Some illustrations of MANTA's functionalities*

1.1 Solid mechanics

MANTA gathers both implicit and explicit mechanics for deformable structures and their interactions (mainly with compressible fluids). A wide range of applications are concerned from the quasi-static to the fast transient dynamics, embracing also seismic and vibrations domains.

1.1.1 Representative Volume Element of Combustible Mixed Oxides for Nuclear Applications

This simulation represents an RVE of MOx (Mixed Oxide) material under uniform macroscopic deformation. The mesh used is a periodic mesh containing 50 spheres/inclusions, representing a volume fraction of 17 % [†1].

The applied strain tensor is defined as follows

$$\varepsilon = \begin{pmatrix} -t\,\dot{\varepsilon}_0/2 & 0 & 0 \\ 0 & -t\,\dot{\varepsilon}_0/2 & 0 \\ 0 & 0 & t\,\dot{\varepsilon}_0 \end{pmatrix}$$

with a strain rate $\dot{\varepsilon}_0 = 0.012$ and t being the time.

The inclusions are assumed linear elastic. The matrix obeys an elasto-viscoplastic law. The material parameters are summarized in the following table :

TABLE 1. *Material properties*

Material	Young Modulus (Pa)	Poisson Ratio	Stress Threshold (Pa)	Norton Exponent (n)
Matrix	8.182e9	0.364	100.0e6	3.333333
Inclusions	16.364e9	0.364	-	-

The Figure 2 illustrates this simulation, involving approximately 135 million Degrees Of Freedom (DOF), conducted over a total simulation time of 5 seconds across 80 time steps ($\Delta t = 0.0625s$).

[†1] The mesh was generated using the open-source software MEROPE in combination with GMSH mesh generator.

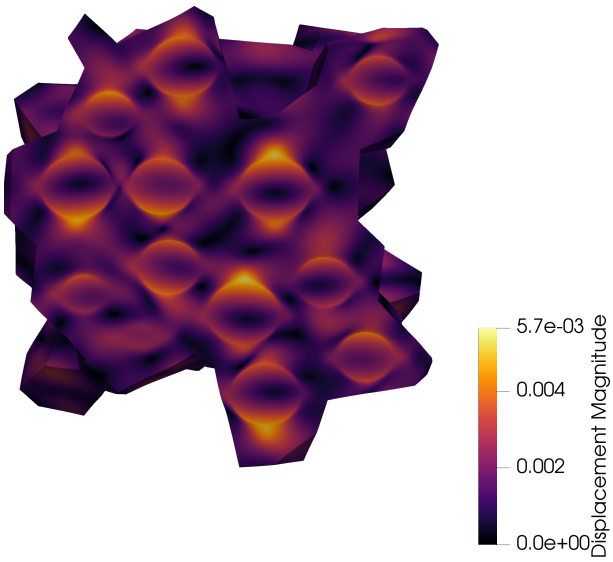

FIGURE 2. *Visualization of a MOX with 17% inclusions, using an elasto-viscoplastic behavior law with color representation based on the magnitude of displacement*

A Newton algorithm is applied during each time step, converging within 2 to 4 Newton iterations for a tolerance 10^{-6}. The solver used with `PETSc` is `GMRES` and the preconditioner is `HypreBoomerAMG`. This simulation was performed using 4,096 `MPI` processes on the CCRT/Irene HPC platform, resulting in an execution time of 10 hours and 33 minutes.

1.1.2 Brittle fracture with phasefield approaches

A phase-field model for fracture has been implemented within `MANTA`. For the moment, a very straightforward implementation based on Miehe's approach [**?**] has been used : a phase-field model solved in a staggered fashion where irreversibility is driven by the introduction of a non-decreasing History field. Additionally, in order to consider unilateral effects in the case of closing cracks, multiple splits of the mechanical energy term have been tested within `MANTA`, taking advantage of the existing phase-field behaviours in the `MFront` library.

Considering the flexibility of `MANTA`'s framework, namely the fact that it can treat explicit or implicit problems with similar tools, means that several kinds of fracture simulation cases can already be performed. On one hand, one can consider the case of brittle fracture in a quasi-static framework, neglecting inertial effects, and thus solving for the displacement and damage successively, within an implicit resolution scheme. On the other hand, one can study dynamic fracture cases using an explicit staggered resolution scheme that is also setup within `MANTA`.

Furthermore, several extensions of this popular approach are also available within `MANTA`. For instance, one can use fatigue extension of the phase-field model as put forward in Alessi et al. [**?**] and Carrara et al. [**?**], effectively providing a platform to study fatigue crack nucleation and propagation. Moreover, multiple dynamic phase-field formulations were implemented to provide the users with a choice in the explicit time integration scheme. For instance, we can cite the regularization of the dynamic formulation as proposed in Kamensky et al. [**?**] that enables to treat the coupled phase-field problem in a fully explicit manner.

Multiple usual benchmarks of the phase-field literature were reproduced in `MANTA`, and notably this Charpy-like test. A notched beam set on two sliding contact support is impacted in the middle yielding the illustrated damage field. This simulation is based on a 67 million cells mesh, solved in multiple parallelized configurations in order to show the scalability performance of the code.

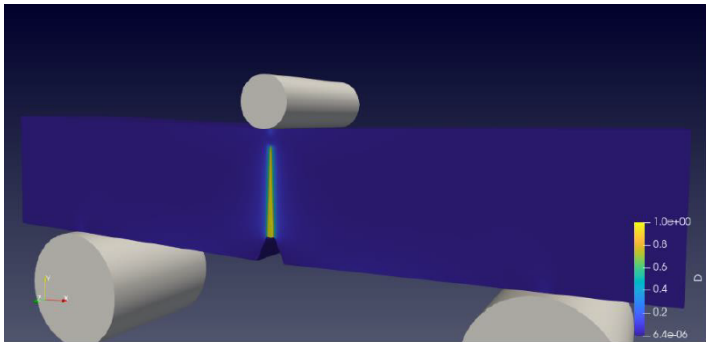

FIGURE 3. *Charpy specimen fracture using an explicit phasefield approach*

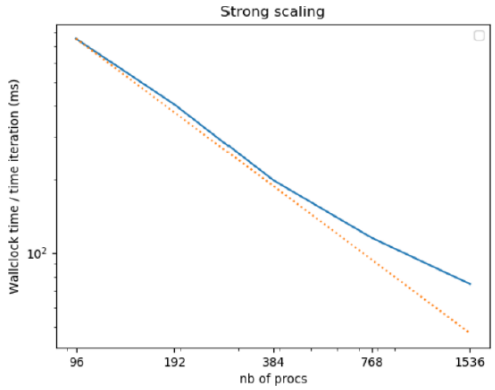

FIGURE 4. *Strong scaling for the Charpy specimen calculation*

1.2 Fluid mechanics

1.2.1 Overview

Fluid mechanic applications in MANTA mainly cover compressible flows, as Europlexus does. For incompressible flows, the CEA develops the TrioCFD software. MANTA is able to simulate both fast transient computations for accidental conditions using explicit time schemes, and nominal quasi-steady states with implicit time schemes. If desired, MANTA is able to chain the computation of an initial steady-state flow and a transient regime, by perturbation of the steady-state. In most cases, the flow is considered as inviscid and is mathematically represented by the Euler conservation equations. The spatial discretization of the Euler equations is presently implemented with the Finite Volumes Method. The conservative property of this method allows for an accurate representation of the velocity and amplitude of shock waves.

Industrial applications for the CEA gather many nominal or accidental situations in nuclear power plants. They also include a broad range of safety-related applications. Fluid flows are often computed to evaluate the loadings applied on the structures (pressure waves, vibrations, drag forces,...) and many cases imply couplings to handle fluid-structure interaction.

We can cite, for example :

- Solid explosions (e.g. TNT) in air or water
- Combustion models for reactive flows (H_2 explosion)
- Pipe modeling (pipe whip, water hammer,...)
- Loss Of Coolant Accident : rarefaction wave propagation after pipe break in the primary circuit of a Pressurized Water Reactor
- Tank perforation
- Prompt bursts in Molten Salt Reactors
- Welding

The following sections show two examples, one steady-state and one transient, showing various functionalities available in the fluid modeling layer of MANTA

1.2.2 Steady-state computation with an implicit time scheme

This 2D simulation represents a Mach 3 supersonic flow of an inviscid perfect gas, with a heat capacity ratio $\gamma = 1.4$, over a circular bump. The x-axis of the reference frame is parallel to the top boundary. All quantities are dimensionless. The lower and upper boundaries are slip walls. The left boundary condition is a supersonic inflow, where all variables are prescribed :

$$\begin{pmatrix} \rho \\ u_x \\ u_y \\ p \end{pmatrix} = \begin{pmatrix} 1.4 \\ 3 \\ 0 \\ 1 \end{pmatrix}$$

The initial condition is a uniform state over the domain equal to the supersonic inflow state. The mesh is an unstructured grid of 140,000 quadrangular cells. The space scheme consists of a HLLC Riemann solver combined with a constant reconstruction per cell an limiter. The time scheme is an implicit first-order Backward Euler scheme. Each implicit time step is solved using a monolithic Newton-Raphson iterative algorithm. To construct the Jacobian matrix of the residual, the HLLC Riemann solver derivative, with respect to the left and right states of each local Riemann problem, is computed using finite differences with a delta equal to 10^{-7}. It corresponds to the square root of the double floating point precision used in the computation.

The convergence of the simulation is monitored by computing the L_∞ norm of the relative residual, with respect to the initial residual. The norm of the relative residual is converged below 10^{-14}. Each time step is solved by performing only one iteration of a Newton-Raphson algorithm. The corresponding linear system is solved using the MUMPS direct solver. The CFL is increased from 1 to 10^6, by multiplying it by 1.5 after each time step. Steady-state is reached in 20 iterations. The steady-state computed solution is represented on Figure 5, and the convergence history is represented on Figure 6.

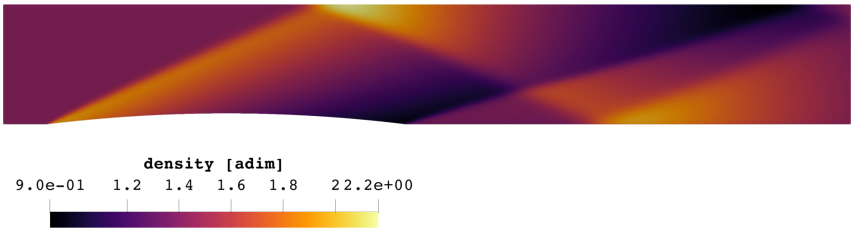

FIGURE 5. *Visualization of a 2D Mach 3 supersonic flow of a perfect gas with a heat capacity ratio* $\gamma = 1.4$ *over a bump in a channel with color representation based on the dimensionless density*

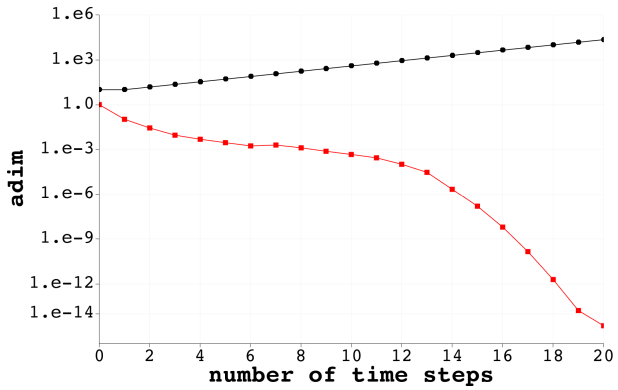

FIGURE 6. *Convergence history of a 2D Mach 3 supersonic flow of a perfect gas with a heat capacity ratio* $\gamma = 1.4$ *over a bump in a channel with the dimensionless* L_∞ *norm of the relative residual in red (squares), and the CFL number in black (disks)*

1.2.3 Transient simulation with an explicit time scheme

This simulation represents a Richtmyer-Metschkov Instability (Figure 7), where a shock interacts with an interface between fluids of different densities. The shape of the resulting interface and the entire flow field show a very non-conventional behavior. The test is initially proposed by Saurel, Petitpas & Berry [**?**] where the phenomenon is computed with a six-equations model, considering the cavitation effects. The left part of the computational domain is filled with pure water while the right part with pure gas. They are initially separated by a curved interface. It is a portion of circle with a 0.6 m radius centered at x = 1.2 m, y = 0.5 m. The physical domain is 3 m long and 1 m high. The unstructured mesh contains 360 000 cells. Both water and gas have an initial velocity of – 200 m/s. Top, bottom and left boundaries are treated as solid walls. Fluids are modeled with the stiffened gas equation of state. The initial parameters for the liquid and gas are :

- Water : $\rho_1 = 1000\ \mathrm{kg/m^3}, P_{\infty 1} = 6.10^8\ \mathrm{bar}, \gamma_1 = 4.4$
- Gas : $\rho_2 = 100\ \mathrm{kg/m^3}, P_{\infty 2} = 0\ \mathrm{bar}, \gamma_2 = 1.8$

In the models presented in the work the cavitation effects are not considered. Therefore, we need to impose the minimum pressure for the liquid.

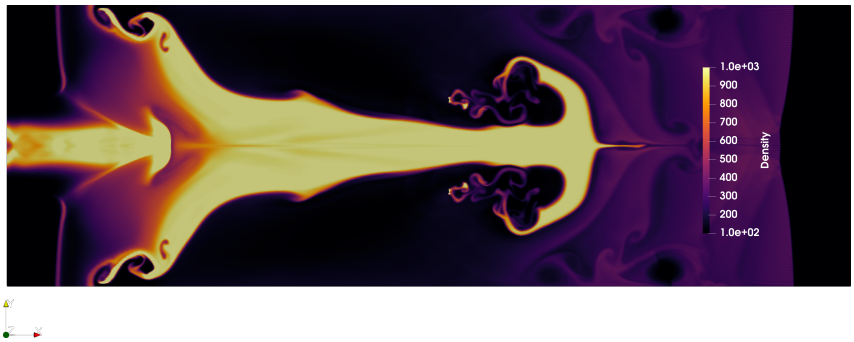

FIGURE 7. *Visualization of a 2D Richtmyer-Metschkov Instability : Density at t=7.9* ms

This case features are :

- Euler equations
- Explicit time integration, CFL = 0.5
- Finite Volume Method, 2nd order in space and time (MUSCL-Hancock)
- multi-component flow with the 5-equations (one pressure / one velocity) model [**?**]
- HLLC Riemann solver

2 Overview of MANTA architecture

2.1 Application Programming Interface (API)

`MANTA` targets different kinds of users and so provides several APIs, tailored for each kind. The two main levels of APIs are called the “end-user” one and the “expert” one. The “expert” API is available only in C++. It offers the maximum genericity and fine tuning possibility, but it does not guarantee to be stable in time (it may evolve quite often). It mainly targets students or researchers in the field of numerical methods, or development teams wanting to build their application on top of `MANTA`’s core functionalities.

The “end-user” API, available in `C++` and `python`, targets engineers or researchers interested in simulating physical systems of interest, domain-specific platforms, and code coupling through the `ICoCo` interface of the `Salome` platform. This API is meant to be extremely stable in time to address industrial applications. It focuses on the physics to be treated. To do so, the names of manipulated functions and objects use an expressive semantic field, regarding physical numerical simulation : coupling scheme, model, loadings, etc. Most numerical details are hidden by default.

The user can seamlessly use all material properties, mechanical behaviours and models generated by the `MFront` code generator [**?**].

More than an API, the “end-user layer” provides some advanced algorithms to ease the onset of coupled problems, and capacities to allow custom extensions.

Dependencies algorithm

Inspired by the `licos` fuel performance code [**?**], each actor of a computation expresses its dependencies that must be met for the computation to be well posed. For instance, inside a mechanical model, a mechanical behaviour may require the temperature and the Young's modulus. The temperature can be provided by a thermal model, a loading or an external solver. The key point is that the mechanical behaviour would not know how the temperature is computed and that the link between the behaviour and the temperature is automatically resolved by `MANTA`. Dependencies resolution is an iterative algorithm : a property may provide the Young's modulus but can require the temperature and the porosity to do so.

This "dependencies management algorithm" makes `MANTA`'s very versatile and minimises user's scripts or input files.

Plugins

The end-user layer can be extended using `C++` plugins to describe additional phenomena or boundary conditions. This system gives direct access to the `MANTA` core-layer described below.

2.2 Core architecture

2.2.1 Objectives

✓ Industrial applications

`MANTA` targets industrial applications and so is able to simulate numerical models having all the complexity of industrial physical systems. To do so it provides a very high level of flexibility. Within a single calculation it can handle multiple zones, multiple physical models or PDEs, multiple material and dimensions, Eulerian/Lagrangian/ALE descriptions, ...

The main restrictions of `MANTA` is that it only deals with mesh-based numerical methods. It is not designed to handle Lattice Boltzmann Methods, or some particles based-methods for example.

This high level of flexibility required to address complex industrial numerical models distinguish `MANTA` from other frameworks for solving PDEs more oriented toward research works, such as `FEniCS` or `MFEM`.

✓ HPC

One of `MANTA`'s first objectives is to be able to run in a HPC context. At this time, it focus mainly on distributed memory parallelism. The parallel scalability for both computation times and memory consumption are of great concern in the development of the code. In particular, regarding memory consumption, every data structures is "local" and grows as the size of the sub-domain. There are ongoing works on the portability of performances, mainly toward GPUs.

A large part of the responsibility for parallel scalability is delegated to third parties, in particular the distributed linear solver package which is `PETSc` at this time.

✓ Easy maintainability and evolvability

`MANTA` is designed for long-term operation. It is developed with a high care about factorization and modularity of the code.

Besides the modularity of its own source code, `MANTA` makes modular usage of its external third parties, in order to preserve its sovereignty on the long term. Here modularity refers to the code's ability to replace one external component with another equivalent for a given functionality. This agility ensures the code's longevity, through the easy renewal of external components, and the facility to to evolve towards more recent tools, without being inextricably bound to one of them.

This has two consequences for software design :

- External components and libraries must only be called up via well-defined interfaces. These should not be diffuse, but restricted to a few dedicated source files.
- The functionalities of external components should be used in an "atomic" way (as elementary as possible). In the event of replacing an external component, this makes it much easier to find another

component with equivalent functionality. For example, PETSc offers a complete development framework for a simulation software, but MANTA has chosen not to use this global framework. Only functions required for distributed linear system solving are directly called.

Factoring consists in imposing that any succession of commands making up part of an algorithm to appear only once in the code. Ultimately, this means aiming for a high ratio between the number of functionalities and the number of lines. This implies :

- Factorization of numerical methods. A generic framework allowing to deal with a priori any mesh-based method through the programming of some entry points has been devised. It is referred to as the pipeline, and described below. Every new functionality should be developed as much as possible within the context of this pipeline, through a specific implementation of some of the entry points. Furthermore, when developing new functionalities, a generalization effort is needed to extract elementary methods that can be generalized to be compatible with other functionalities, in other contexts.
- Source code factorization. This point is linked to good development practice. It aims to the best compromise between performance, factorization and code readability.

2.2.2 The "pipeline"

MANTA core functioning is structured around a generic algorithm to assemble distributed linear systems from spatial integration (and solve them) which exhibits some entry points. These entry points have to be implemented specifically to deal with the numerical method and PDEs at stakes. This single generic execution channel called the pipeline in MANTA's jargon. The core-layer also provides a wide range of generic tools and services (shape functions, mesh operations, computational geometry, ...) to ease the implementation of the entry points.

→ The linear system

The pipeline aims at assembling block linear systems of the form :

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}_1^T & \mathbf{C}_2^T & \cdots \\ \mathbf{C}_1 & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda_1 \\ \lambda_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{D}_1 \\ \mathbf{D}_2 \\ \vdots \end{bmatrix}$$

where each block of the LHS and the RHS is assembled through a unified generic procedure which offers some entry point to specify the numerical method and physical model considered.

→ LHS and RHS assembling

The generic "mathematical framework" related to the pipeline consists in the assembling and resolution of sparse linear systems constructed by integration of dense matrix integrands on entities of a mesh. Broadly, the numerical method consider should involve sparse matrices that can be constructed the following way :

$$\mathbf{M} = \sum_i \mathcal{A}_i \int_{E_i} \mathbf{m}(\underline{x}) \underline{\mathrm{x}}$$

where E_i are the mesh entities on which the matrix valued function $\mathbf{m}$ is integrated. $\mathcal{A}_i$ is an assembling operator which maps indices of rows/columns local to the entity i to global indices of rows/columns in $\mathbf{M}$.

In practice, the integral is approximated using quadrature rules a mapping to reference cells.

$$\mathbf{M} = \sum_i \mathcal{A}_i \sum_j w_j \mathbf{m}(\underline{\xi}_j) |\det(\underline{\phi}_i(\underline{\xi}_j))| \text{ , where } (\underline{x} \in E_i) = \underline{\phi}_i(\underline{\xi})$$

where $\underline{\xi}_j$ and w_j are the location and weight related to the considered reference entity, and $\underline{\phi}_i : \underline{\xi} \rightarrow \underline{x} \in E_i$ is the mapping from the reference entity and the mesh entity E_i.

Broadly, the pipeline computes this "formula" in a generic way (there is a single implementation of this formula in the source code which is used by all the numerical methods), and each numerical method just defines its own operator $\mathcal{A}$ and function $\mathbf{m}$.

One major interest of this is that the two sums, which turn into loops in the implementation, are located in the generic part of the source code. Then all the numerical methods which are implemented within this framework take advantages from the parallelization of these loops.

→ Linear system solving

Once assembled, a distributed linear system as above has to be solved. If there actually are some matrices and vectors $\{\mathbf{C}_i, \mathbf{D}_i\}$, then the system is a saddle-point type, which may be tricky to solve efficiently.

MANTA provides several methods to eliminate, when it is possible, some Lagrange multipliers unknowns : A and B are modified to remove a given set $\{\mathbf{C}_i, \mathbf{D}_i, \lambda_i\}$ such that the solution X is preserved or approximated. All the sets $\{\mathbf{C}_i, \mathbf{D}_i\}$ which are not eliminated are gathered into a unique set $\{\mathbf{C}, \mathbf{D}\}$ to have a system of the form :

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{D} \end{bmatrix}$$

The remaining system, being of saddle point type or not, is then transferred to the distributed linear system external solver (PETSc).

→ Automatic parallelism

An important point in the design of MANTA is what we call "automatic parallelism". This means that one can code a new feature in the pipeline framework (almost) as if it would in a sequential code, and it just works in parallel.

The only thing related to parallelism that a developer have to handle occurs when dealing with a numerical method for which the computation of $\mathbf{m}$ requires some data related to another mesh entity than the current one. This is the case for example for a numerical method involving a stencil (such as the finite volume method), or a method involving interaction between entities related to geometrical criteria, such as contact mechanics or fluid-structure interactions with immersed boundaries approaches. In such cases, the developer has to inform the pipeline how many layers of ghost cells (we talk about "topological ghost entities") is required in case of stencils, or how to compute which entities are interacting with the local ones in case of geometrical interactions (we talk about "geometrical ghost entities").

Once declared, the ghost entities will be copied from their native subdomain to the subdomains requesting them. The important point is that there are transferred with all data related to them, and recursively for the entities of lower dimension composing them (the field attached to them, the "named zone" to which they belong, ...). This allow the developer to be sure that every entity in the subdomain, ghost or not, carries the same data as in a sequential run. The drawback of this approach is that it prevent from specific optimization for a given numerical method on the amount of data transferred between the subdomains.

2.2.3 The "services"

Alongside the pipeline, MANTA provides a large set of built-in tools to help programming the entry points of the pipeline when developing a new feature.

Here are some of them :

- mesh handling
- dense matrix algebra
- handling of "configurations" to handle Lagrangian, Eulerian and ALE descriptions
- "field" data structures
- computational geometry toolbox
- discrete differential operators for finite elements
- monomial basis for non-conforming discretization methods

- gradient reconstruction and limitation tools for finite volumes
- MFront behaviors support
- ...

Conclusions and future works

MANTA is still young. A lot of generic and more applicative functionalities have to be developed. Numerous developments are underway to be rapidly integrated. For example :

- save/restart
- adaptive hierarchical refinement (AMR) and refinement criteria,
- friction contact modeling,
- crack propagation modeling (element erosion, debonding, remeshing, etc.)
- "micromorph" approaches to structural damage,
- fluid models more representative of explosion physics,
- anti-hourglass to enable the use of under-integrated elements,

On the performance side, the efficient handling of "complex" problems involving several tens of billions of degrees of freedom on hundreds of thousands of computing cores of future exascale machines represents an ambitious challenge that we hope to meet with the MANTA code by 2030. The following 3 axes will be addressed :

1. Handling complex boundary conditions when solving distributed linear systems. Specific methods will have to be developed and implemented to efficiently handle large sparse systems with iterative solvers. Work will also be carried out on the definition of efficient preconditioners for this type of problem.
2. Dynamic load balancing for distributed memory parallelism. This involves regularly revising the domain decomposition during computation, to ensure that the load is distributed evenly between the computational units. One of the main difficulties for the targeted computations lies in the fact that the optimal sub-domain decomposition may be different for each computational stage when solving a time step. Strategies to deal with this difficulty will be developed. AMR (Adaptive Mesh Refinement) also makes dynamic balancing particularly difficult on very large computational platforms.
3. Performance portability, especially on GPUs. In order to address this issue of GPUs within the more general framework of performance portability, we plan to use tools such as Kokkos (https ://kokkos.org/about/). A great deal of work will be required to adapt and optimize the software architecture and data structures of MANTA core layer in order to achieve maximum performance on GPU architectures. The question of processing of constitutive laws on GPUs will be a central focus of this axis, as this is a major cost center in terms of execution time - the MFront [**?**] library is used to process for processing these laws.

In order to illustrate MANTA's capabilities and meet the needs of its future users, the developments described above will be accompanied by the implementation of target calculations in various fields of application (e.g. thermomechanical calculations of various nuclear components, interaction vibrations, etc.). applications (e.g. thermomechanical calculations of various nuclear components, vibrations with fluids, deformation of fuel assemblies, wear, explosion, earthquake response with Soil-Structure Interaction, multi-scale modeling of composites, etc.).

Acknowledgements

This research was conducted in the framework of the MECANUM project, which was supported financially by the CEA (Commissariat à l'Énergie Atomique et aux Énergies Alternatives) and EDF (Électricité De France).