Computational Simulations of 3D fluid-dynamic Processes in High Performance Computing Environment

L. Carracciuolo[1], D. Casaburi[2], L. D'Amore[2], G. D'Avino[3], P.L. Maffettone[3] and A. Murli[2]

[1]Consiglio Nazionale delle Ricerche

[2]Dipartimento di Matematica ed Applicazioni
Universitá di Napoli "Federico II"

[3]Dipartimento di Ingegneria Chimica
Universitá di Napoli "Federico II"

***Abstract*—Within the S.Co.P.E. italian national projects, a multidisciplinary collaboration between computational and chemical engineering scientists was established with the aim of developing three dimensional simulations of fluido-dynamic processes. The computing environment relies on PETSc (Portable, Extensible Toolkit for Scientific Computation [1]) components integrated with the TFEM (Toolkit for Finite Element Method [3]) software toolkit, for discretization and solution of the partial differential equation-based model describing the physical processes. We discuss first computational efforts, experiments, numerical and performance results on the basis of selected test cases.**

***Index Terms*—parallel computing, fluid dynamics simulations, PETSc.**

I. Introduction

IN many technological applications, solid fillers of different shape, size and chemistry are added to fluids in order to implement functional changes which result in a reduction of process costs or, more frequently, in an improvement of mechanical performance of the final product.

The presence of solid fillers in fluids, which often exhibit themselves a complex rheological behavior, involves other phenomena which, from an applicative point of view have a strong impact on the reliability and the quality of the final product. For instance, the tendency of particles to aggregate and to form ordered structures leads to different filler distribution within the fluid: such behavior should be avoided when the aim of the process is to produce composite materials of high toughness and strength. Therefore, to understand the onset and the evolution of such phenomena is of primary importance for the investigation and construction of materials satisfying the application requirements.

To this aim a multidisciplinary collaboration between computational and chemical engineering scientists in the contest of the S.Co.P.E. Italian national projects was established in order to carry on computational simulations of such 3D fluydo-dynamics processes on high end computing infrastructures. The results of this work can be used as the starting point of future applications in industrial processes such as flow intubation and injection molding.

Scientific simulation is an integrated process. It is also an iterative process, in which computational scientists traverse several times a hierarchy of issues in modeling, discretization, solution, code implementation, visualization and interpretation of results, and comparison to expectations (theories, experiments, or other simulations). They do this to develop a reliable *scientific instrument* for simulation, namely a marriage of a code with a computing and user environment.

One loop over the computational process is related to validation of the model and verification that the simulation process is correctly describing the model. The other loop over the process of simulation is related to algorithm and performance tuning. A simulation that yields high-fidelity results is of little use if it is too expensive to run or if it cannot be scaled up to the resolutions, simulation times, or ensemble sizes required to describe the real-world phenomena of interest. With some oversimplification, algorithm tuning poses the question, *Are we getting enough science per operation?*

During or after the validation of a simulation code, attention must be paid to the performance of the code in the target computing environment. Efforts to control discretization and solution error in a simulation may grossly increase the number of operations required to execute a simulation, far out of proportion to their benefit. Often adaptive strategies must be adopted for the simulation to be execution-worthy on an expensive massively parallel computer, even though these strategies may require highly complex implementation.

It is clear that a scientific simulation depends upon numerous components, all of which have to work.

In this paper we describe first computational efforts towards the development of software tools needed to the computational simulation of fluido-dynamic applications to harness the computational power of high performance computing (HPC) environments.

The computing environment that supports the simulation software relies on PETSc (Portable, Extensible Toolkit for Scientific Computation [1]) components integrated with the TFEM (Toolkit for Finite Element Method [3]) software toolkit, employed to discretize and solve numerical problems deriving from partial differential equation-models.

In particular, we discuss the integration of TFEM modules inside the PETSc parallel data structures, the introduction of the parallelism inside the discretization step, the validation of numerical results and performance gain on the basis of a test case [2].

II. From the mathematical model to the design of the numerical algorithm

A. The mathematical model

The problem that we consider consists of a single sphere in a sheared viscoelastic fluid. Assuming incompressibility, negligible inertia, and buoyancy free conditions the governing equations are the continuity (mass balance) and momentum balance equations, plus a constitutive equation depending on the nature of the suspending liquid:

$$\nabla \cdot \boldsymbol{v} = 0 \tag{1}$$

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0} \tag{2}$$

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\eta_s \boldsymbol{D} + \boldsymbol{\tau} \tag{3}$$

where $\boldsymbol{v}$ is the velocity, $\boldsymbol{\sigma}$ is the stress tensor, p is the pressure, $\boldsymbol{I}$ the unity tensor, $\boldsymbol{D} = (\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^T)/2$ is the rate-of-deformation tensor, η_s is the solvent viscosity and $\boldsymbol{\tau}$ is the constitutive extra stress.

The polymer stress, $\boldsymbol{\tau}$, is given written as:

$$\boldsymbol{\tau} = \frac{\eta_p}{\lambda}(\boldsymbol{c} - \boldsymbol{I}) \tag{4}$$

where $\boldsymbol{c}$ is the ‘conformation tensor’, η_p is the polymer viscosity, and λ is the relaxation time.

We will model the viscoelastic fluid with the Giesekus constitutive equation (for $\boldsymbol{c}$):

$$\lambda \overset{\nabla}{\boldsymbol{c}} + \boldsymbol{c} - \boldsymbol{I} + \alpha(\boldsymbol{c} - \boldsymbol{I})^2 = \mathbf{0} \tag{5}$$

where α is the so-called mobility parameter that modulates the shear thinning behavior. The symbol ($^\nabla$) denotes the upper-convected time derivative, defined as:

$$\overset{\nabla}{\boldsymbol{c}} \equiv \frac{\partial \boldsymbol{c}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{c} - (\nabla \boldsymbol{v})^T \cdot \boldsymbol{c} - \boldsymbol{c} \cdot \nabla \boldsymbol{v} \quad (6)$$

A sphere of radius R is located between two plates moving along the x-direction in different sense with an imposed shear rate $\dot{\gamma}$. Shear flow conditions are imposed on the external fluid boundaries.

The sphere is located at the center of the gap and symmetry implies that it can only rotate. We assume that the sphere is torque-free, or "freely rotating", i.e. its rotation is only due to the motion of the surrounding fluid. Thus, the torque-free boundary condition at $r = R$ is:

$$\int_{\partial S_p} \boldsymbol{r} \times \boldsymbol{\sigma} \cdot \boldsymbol{n}\, dA = \boldsymbol{0} \quad (7)$$

where ∂S_p is the sphere surface, $\boldsymbol{n}$ is the normal at the sphere surface, $\boldsymbol{r}$ the position vector from the sphere center and the integral of the local torque $\boldsymbol{r} \times \boldsymbol{\sigma} \cdot \boldsymbol{n}\, dA$ spans the sphere surface. Due to the symmetry of the imposed shearing flow, only the vorticity component z of Eq. (7) is relevant, the other two components being identically zero. In fact, the velocity at the sphere surface is known, and can be written as:

$$\boldsymbol{v}(R, t) = \omega(t) \boldsymbol{e}_z \times \boldsymbol{R} \quad (8)$$

$\boldsymbol{e}_z$ being the unit vector along z. The particle angular velocity, $\omega(t)$, is, however, unknown and it depends on the nature of the fluid.

B. Weak form

The governing equations are solved by the finite element method on a cell containing a single sphere at the center of the cell. A fictitious domain is used [4] together a rigid-ring description of the particle [5], since inertia is neglected. A log-conformation representation for the conformation tensor has been used: the original equation for the conformation tensor $\boldsymbol{c}$, Eq. (5), is transformed to an equivalent equation for $\boldsymbol{s} = \log(\boldsymbol{c})$, leading to a substantial improvement of stability (see [7] for details). To further improve the numerical stability a DEVSS-G formulation is implemented [8] for the momentum balance and the SUPG technique [6] is used for the constitutive relation.

Under these assumptions, a weak form can be stated as follows: For $t > 0$, find $\boldsymbol{v} \in U, p \in P, \boldsymbol{s} \in S, \boldsymbol{G} \in H, \boldsymbol{\omega} \in \Re, \boldsymbol{\lambda} \in L^2(\partial S_p)$ such that:

$$\int_V 2\eta_s \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v})\, dV - \int_V \nabla \cdot \boldsymbol{u}\, p\, dV + \int_V \eta_p (\nabla \boldsymbol{u})^T : \nabla \boldsymbol{v}\, dV - \int_V \eta_p (\nabla \boldsymbol{u})^T : \boldsymbol{G}^T\, dV + \langle \boldsymbol{u} - (\boldsymbol{\chi} \times \boldsymbol{r}), \boldsymbol{\lambda} \rangle_{\partial S_p} = -\int_V \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\tau}\, dV, \quad (9)$$

$$\int_V q\, \nabla \cdot \boldsymbol{v}\, dV = 0, \quad (10)$$

$$\int_V \boldsymbol{H} : \boldsymbol{G}\, dV - \int_V \boldsymbol{H} : (\nabla \boldsymbol{v})^T\, dV = \boldsymbol{0}, \quad (11)$$

$$\int_V (\boldsymbol{S} + \tau \boldsymbol{v} \cdot \nabla \boldsymbol{S}) : \left(\frac{\partial \boldsymbol{s}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{s} - \boldsymbol{g}(\boldsymbol{G}, \boldsymbol{s}) \right) dV = \boldsymbol{0}, \quad (12)$$

$$\langle \boldsymbol{\mu}, \boldsymbol{v} - (\boldsymbol{\omega} \times \boldsymbol{r}) \rangle_{\partial S_p} = \boldsymbol{0}, \quad (13)$$

$$\boldsymbol{s} = \boldsymbol{s}_0 \quad \text{at } t = 0, \quad \text{in } V, \quad (14)$$

for all $\boldsymbol{u} \in U$, $q \in P$, $\boldsymbol{S} \in S$, $\boldsymbol{H} \in H$, $\boldsymbol{\chi} \in \Re$ and $\boldsymbol{\mu} \in L^2(\partial S_p)$, where U, P, S, G are suitable functional spaces. The τ parameter in Eq. (12) is given by $\tau = \beta h / 2U_e$, where β is a dimensionless constant, h is a typical size of the element and U_e is a characteristic velocity for the element. Finally, we take the initial value of $\boldsymbol{s}_0 = \boldsymbol{0}$, corresponding to zero initial stress.

Notice that the angular velocity, $\boldsymbol{\omega}$, is treated as an additional unknown and is included in the weak form of momentum equation. Only the z-component of $\boldsymbol{\omega}$ is set different to zero since, for the symmetry of the problem, the sphere can rotate around the vorticity axis only. The torque-free condition is imposed through the Lagrange multipliers, $\boldsymbol{\lambda}$.

C. Numerical model and algorithm

For the discretization of the weak form, we use hexahedron elements and a continuous biquadratic interpolation (Q_2) for the velocity $\boldsymbol{v}$, bilinear continuous interpolation (Q_1) for the pressure p, linear continuous interpolation (Q_1) for velocity gradient $\boldsymbol{G}$ and bilinear continuous interpolation (Q_1) for the log-conformation tensor $\boldsymbol{s}$. We discretize the boundary of the particle using the weak constraints. Then, discretization of Eq. (13) gives to:

$$\langle \boldsymbol{\mu}, \boldsymbol{v} - (\boldsymbol{\omega} \times \boldsymbol{r}) \rangle_{\partial S_p} = \int_{\partial S_p} \boldsymbol{\mu} \cdot [\boldsymbol{v} - (\boldsymbol{\omega} \times \boldsymbol{r})] \, dA \quad (15)$$

Regarding the time discretization, initially, the viscoelastic polymer stress is set to zero in the whole domain. Then we solve the equations (9)-(11) and the constraint equation (13) in order to get the distribution of the fluid velocity and the angular velocity of the particle, at the initial time step. At every time step, we perform the following procedure:

Step 1. The log-conformation tensor at the next time step, $\boldsymbol{s}^{n+1}$, is evaluated by integrating the constitutive equation (12). A semi-implicit first-order Euler scheme is used:

$$\frac{\boldsymbol{s}^{n+1}}{\triangle t} + \boldsymbol{v}^n \cdot \nabla \boldsymbol{s}^{n+1} = \frac{\boldsymbol{s}^n}{\triangle t} + \boldsymbol{g}(\boldsymbol{G}^n, \boldsymbol{s}^n) \quad (16)$$

where $\boldsymbol{g}$ is the term which appears in the evolution equation of $\boldsymbol{s}$.

Step 2. The remaining unknowns $(\boldsymbol{v}, p, \boldsymbol{G}, \boldsymbol{\omega})^{n+1}$ as well as the Lagrange multipliers $(\boldsymbol{\lambda})$ can be found by solving the Eqs. (9)-(11) and (13) using the particle configuration and the polymer stress evaluated in the previous two steps:

$$\begin{aligned} &\int_\Omega 2\eta_s \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}^{n+1}) \, dA - \\ &\int_\Omega \nabla \cdot \boldsymbol{u} \, p^{n+1} \, dA + \int_\Omega \eta_p (\nabla \boldsymbol{u})^T : \nabla \boldsymbol{v}^{n+1} \, dA \\ &\quad - \int_\Omega \eta_p (\nabla \boldsymbol{u})^T : (\boldsymbol{G}^{n+1})^T \, dA + \\ &\quad \langle \boldsymbol{u} - (\boldsymbol{\chi} \times \boldsymbol{r}), \boldsymbol{\lambda}^{n+1} \rangle_{\partial S_p} \\ &\quad\quad = - \int_\Omega \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\tau}^{n+1} \, dA, \quad (17) \end{aligned}$$

$$\int_\Omega q \nabla \cdot \boldsymbol{v}^{n+1} \, dA = 0, \quad (18)$$

$$\int_\Omega \boldsymbol{H} : \boldsymbol{G}^{n+1} \, dA - \int_\Omega \boldsymbol{H} : (\nabla \boldsymbol{v}^{n+1})^T \, dA = 0, \quad (19)$$

$$\langle \boldsymbol{\mu}, \boldsymbol{v}^{n+1} - (\boldsymbol{\omega}^{n+1} \times \boldsymbol{r}) \rangle_{\partial S_p} = 0 \quad (20)$$

The Eq. (16) leads to an unsymmetric linear system whereas in Step 2 a sparse linear symmetric system needs to be solved.

Algorithm 1 shows how **Step 1** and **Step 2** organize themself to discretize and solve the problem: in particular **Step 1** is performed as described at points 6 and 7 while **Step 2** is performed as described at 8 and 9.

Algorithm 1 Algorithm outline

```
 1: Define problem
 2: Compute $A_0$ and $b_0$
 3: $x_0 \leftarrow A_0^{-1} b_0$
 4: $x_0^c \leftarrow$ Some initial condition
 5: for $t \leftarrow 1, num\_time\_steps$ do
 6:     Compute $A_t^c$ and $b_t^c$ from $x_{t-1}$ and $x_{t-1}^c$
 7:     $x_t^c \leftarrow (A_t^c)^{-1} b_t^c$
 8:     Compute $A_t$ and $b_t$ from $x_t^c$
 9:     $x_t \leftarrow A_t^{-1} b_t$
10: end for
```

III. The computational environment and the parallelization details

The target computing environment we consider to implement Algorithm 1 is a parallel and distributed memory model where we hide, within the PETSc objects, the details of internode communications.

PETSc is a suite of data structures and routines for the scalable parallel solution of scientific applications modeled by partial differential equations. PETSc is flexible: its modules, that can be used in application codes written in Fortran, C and C++, are developed by means of object-oriented programming techniques [1].

The PETSc hierarchical structure, shown in figure 1 [1], allows the choice of problem depending levels of abstraction; it relies on standard basic computational (BLAS, LAPACK) and communication (MPI) kernels, and provides mechanisms needed within parallel application codes, such as parallel matrix and vector assembly routines that allow the overlap of communication and computation, as well as tools for the parallel management of discretization grids. Looking at figure 1 from bottom-up, i.e. towards a growing level of abstraction, Krylov subspace methods and scalable parallel preconditioners are located, as the *"floor"* upon which linear and non-linear systems solvers rely.

The suite of solvers and preconditioners can be enriched interfacing PETSc with several packages, such as, *BlockSolve95*, *SuperLU*, *SuperLU_Dist*, *Hypre*, *MUMPS*.

We integrate TFEM (Toolkit for Finite Element Method [3]) within the PETSc computing environment TFEM. TFEM is the sequential software already employed for the discretization and the solution of this problem. TFEM is modular and its modules, that can be used in application codes written in Fortran90/95, are developed by means of object-oriented programming techniques. It consists of:

- a kernel, which includes object type (i.e. `mesh`, `problem`, `sysmatrix` and `sysvector` objects) definition/construction and tool for Gaussian integration on elements of different shape;
- some *"add-on"*s including tools performing: data visualization, the interface to external linear equation system solvers (PARDISO, MUMPS, etc.), the interface to external software for graphs reordering (METIS), the definition of details related to the discretization of some models (Stokes model, Viscoelastic model, etc.).

Implementation of Algorithm 1 on the target HPC environment means the use of TFEM at steps 1, 2, 6, 8 and of PETSc at steps 3, 7, 9. Schematically, we design the following algorithm:

Algorithm 2 Parallel Algorithm using TFEM+PETSc

```
1: TFEM: Define problem
2: TFEM: Compute $A_0$ and $b_0$
3: PETSc: $x_0 \leftarrow A_0^{-1} b_0$
4: $x_0^c \leftarrow$ Some initial condition
5: for $t \leftarrow 1, num\_time\_steps$ do
6:     TFEM: Compute $A_t^c$ and $b_t^c$ from $x_{t-1}$ and $x_{t-1}^c$
7:     PETSc: $x_t^c \leftarrow (A_t^c)^{-1} b_t^c$
8:     TFEM: Compute $A_t$ and $b_t$ from $x_t^c$
9:     PETSc: $x_t \leftarrow A_t^{-1} b_t$
10: end for
```

To this aim, it is required:

1) to modify TFEM modules to replace original data structure used in `sysmatrix` and `sysvector` objects by the PETSc parallel objects
2) to introduce the parallelism during the definition of such objects using *"row-block"* distribution (points 2, 6, 8 of the Algorithm 1),
3) to identify the PETSc computational solvers to use (points 3, 7, 9 of the Algorithm 1).

IV. Preliminary results and Conclusions

[1] `http://www.mcs.anl.gov/petsc/petsc-as/features/diagram.html`

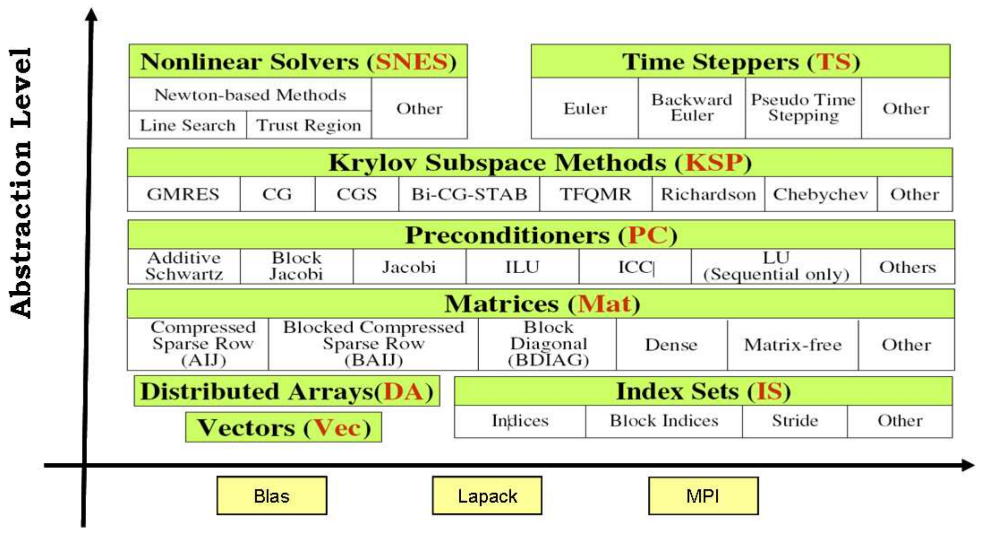


Fig. 1. Hierarchical structure of PETSc

WE show performance results related to the execution time and speed-up obtained on a test case.

Discretization of the problem described in section II is performed using a mesh of $15 \times 15 \times 15$ elements while the particle is discretized by using 128 elements (see figure 2 for mesh representation). This discretization leads to a sparse and symmetric matrix A_0 (see figure 3-(a) for sparsity pattern) with a sparsity pattern of $.03\%$ respet to $O\left(10^{12}\right)$ entries.

This problem has a medium-large level of complexity (the storage of double precision floating point numbers not-zero entries of A_0 requires more than $4GB$ of memory).

We adopt the *row-block data distribution*, which is the standard data distribution adopted in PETSc. In figure 3-(b) we show the *block-row distribution* of A_0 on 4 processors. As expected, such data distribution does not take into account the sparsity pattern of data and it may lead to workload unbalance.

Tests are carried out on a multi processor system with the following configuration [2]:

Hardware configuration: a blade type system composed by:

- 16 blade Dell PowerEdge M600 with,
  - two quad core Intel Xeon E5410@2.33GHz processors,
  - 16 Gb of RAM,
  - A Mellanox Technologies MT25418 Infiniband card,
- A Cisco M SFS7000 switch for internal Infiniband connectivity,

Software configuration: a cluster for parallel application based on:

- Scientific Linux 4.6 operating system,
- Mellanox driver and software for Infini-

[2]http://scopedma-wn.dma.unina.it

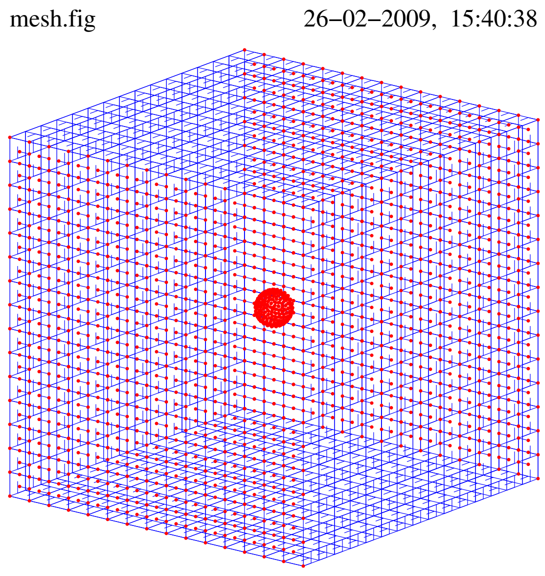
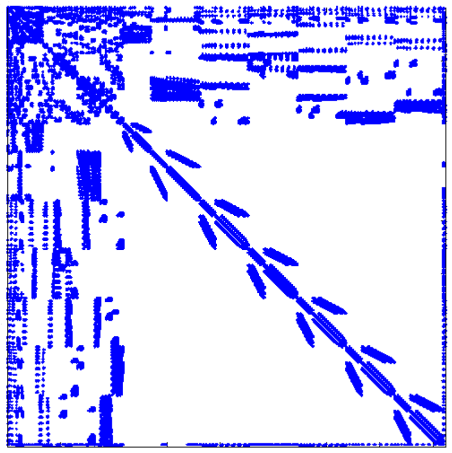
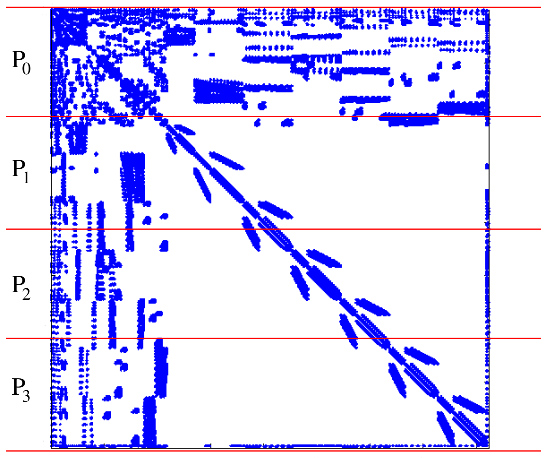


Fig. 2. Representation of discretization mesh



(a)



(b)

Fig. 3. Sparsity pattern of matrix A_0 (a), representation of *block-row* type distribution of A_0 (b)

band connectivity,

- SCoPE Toolkit 1.0 [9](PETSc, ScaLapack, etc.), Intel compilers and libraries.

In figure 4 we show the execution time (a) and speed-up (b) versus the processor number. It is worth to underline that these preliminary experiments focus on the discretization step only. More precisely, we monitor the performance gain obtained introducing concurrency during the discretization step only. As a consequence, the execution time and the speedup refer to point 2 only and not to the entire execution of Algorithm 2.

We appreciate a significative improvement with respect to the mono processor software. Observe that the sequential version of TFEM was not able to perform this computation. As expected, as the processor number grows, the speedup line highlights the quite limited obtained performance gain, because of the limited amount of parallelism that we have introduced. Moreover the data distribution does not take into account the sparsity pattern of the matrix. Of course, a more suitable data distribution will improve computational workload.

V. Future work

THe primary intent of this work has been the exploitation of high performance computing resources in order to reduce the overall computational cost required for the simulation of 3D fluido-dynamic processes. We are currently working on the management of parallel objects (matrices and vectors) in TFEM *"parallel"* version, and on the selection of the most efficient linear solver (preconditioned Krylov subspace methods, based on B-Jacobi, ASM, or AMG preconditioners, or LU-based direct methods and Multifrontal methods). Finally, we will be devoted to the visualization and interpretation of results, and comparison to expectations (theories, experiments, or other simulations).

References

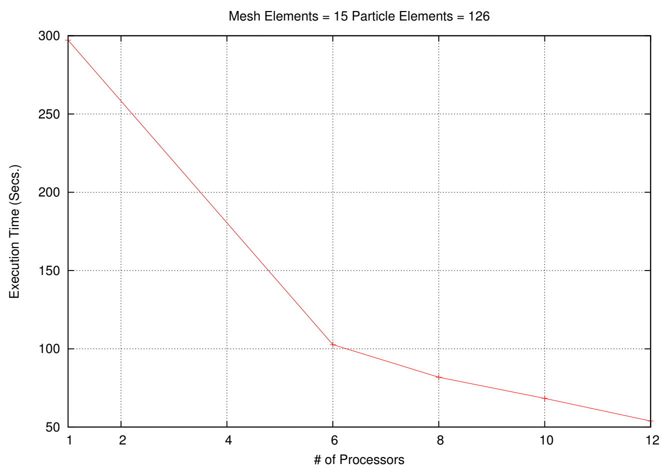
[1] S. Balay, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. Curfman McInnes, B. Smith, H. Zhang, *PETSc Users Manual*, Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[2] G. D'Avino, M.A. Hulsen, F. Snijkers, J. Vermant, F. Greco, P.L. Maffettone, *Rotation of a sphere in a viscoelastic liquid subjected to shear flow. Part I: Numerical results*, J. Rheol. 52(6), 2008.

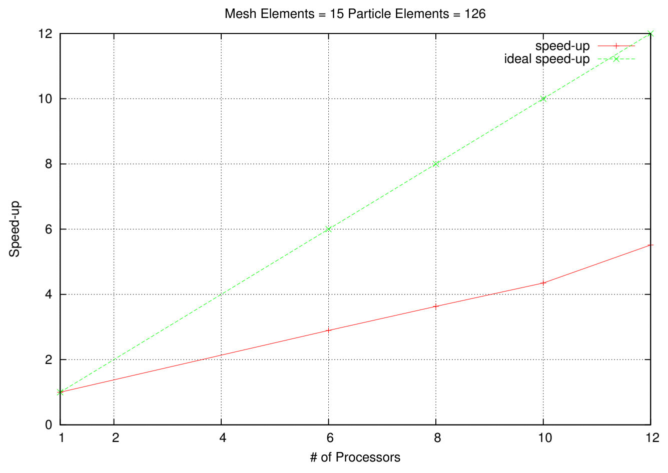
[3] M.A. Hulsen, *TFEM: A toolkit for the finite element method*, Userguide, 2009.

[4] R. Glowinski, T.-W. Pan, T.I. Hesla, D.D. Joseph, *A distributed Lagrangian multipliers/fictitious domain method for particulate flows*, Int. J. Multiphase Flow 25, 1999.

[5] W.R. Hwang, M.A. Hulsen, H.E.H. Meijer, *Direct simulation of particle suspensions in sliding bi-periodic frames*, J. Comput. Phys. 194, 2004.



(a)



(b)

Fig. 4. Preliminary performance results: execution time (a) and speed-up (b) versus the processors number

[6] A.N. Brooks, T.J.R. Hughes, *Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comp. Meth. Appl. Mech. Eng., 32, 1982.

[7] M.A. Hulsen, R. Fattal, R. Kupferman, *Flow of viscoelastic fluids past a cylinder at high Deborah number: stabilized simulations using matrix logarithms*, J. Non-Newtonian Fluid Mech., 127, 2005.

[8] R. Guénette, M. Fortin, *A new mixed finite element method for computing viscoelastic flows*, J. Non-Newtonian Fluid Mech., 60, 1995.

[9] G.B. Barone, D Bottalico, V Boccia, L Carracciuolo, S Pardi, M Scognamiglio, F Serio, *Middleware Applicativo: lo S.Co.P.E. Toolkit*, Proceedings of Italian National Conference of e-Science, 2008.