FFT-based Poisson Solver for large scale numerical simulations of incompressible flows

R. Borrell*, O. Lehmkuhl*, F.X. Trias**, G. Oyarzún** and A. Oliva**
Corresponding author: ricard@termofluids.com, cttc@cttc.upc.edu

* Termo Fluids, S.L., C/ Magí Colet 8, 08204 Sabadell, Spain.
** Heat and Mass Transfer Technological Center, Technical University of Catalonia, Spain.

Abstract: In the context of time-accurate numerical simulation of incompressible flows, a Poisson equation needs to be solved at least once per time-step to project the velocity field onto a divergence-free space. Due to the non-local nature of its solution, this elliptic system is one of the most time consuming and difficult to parallelise parts of the code. In this paper, a parallel direct Poisson solver restricted to problems with one uniform periodic direction is presented. It is a combination of a Direct Schur-complement based Decomposition (DSD) and a Fourier diagonalisation. The latter decomposes the original system into a set of mutually independent 2D systems which are solved by means of the DSD algorithm. Since no restrictions are imposed in the non-periodic directions, the overall algorithm is well-suited for solving problems discretised on extruded 2D unstructured meshes. A new overall parallelisation strategy with respect to our earlier works is presented. This has allowed us to solve discrete Poisson equations with up to 10^9 grid points in less than half a second, using up to 8192 CPU cores of the MareNostrum Supercomputer.

Keywords: Parallel Poisson solver, FFT, Schur Complement, DNS, Unstructured meshes

1 Introduction

Time-accurate DNS/LES simulations generally demand a large amount of time-steps (for DNS applications it can reach $\sim 10^6$). If the mesh does not change during the simulation, the Poisson system also remains constant and is solved repeatedly with different right-hand-side terms. In this situation, a pre-processing phase with relatively large computing demands can be accepted. Another usual feature in DNS/LES applications is to have at least one statistically homogeneous or isotropic direction in the flow. For this direction(s), periodic boundary conditions and an uniform mesh are suitable, making the Fourier diagonalisation [1] applicable. In this work, we restrict ourselves to problems with only one periodic homogeneous direction, examples of this kind of configuration are the flow around a cylinder, airfoil sections or a differential heated cavity, and also the Rayleigh Benard or the flow around a sphere, where the FFT-diagonalisation is applied to the azimuthal direction.

To solve the 2D systems obtained with the diagonalisation, a Direct Schur-complement based algorithm is preferred. The robustness of this direct solver makes its performance irrespective of the condition number or the specific application, and only dependent on the sparsity pattern of the matrix.

Moreover, it is very fast in the solution phase. In [2] we successfully compare it with different Preconditioned Conjugate Gradient methods. As a main drawback, the DSD requires a computationally demanding pre-processing phase and large memory resources. Nevertheless, as mentioned above, this pre-processing cost becomes almost negligible for the time-accurate applications here considered. And, since the set of systems are 2D, the memory requirements remain still feasible for the range of mesh sizes required for DNS/LES application.

With regard to the overall parallelisation strategy, some important improvements have been introduced with respect to our previous works [3, 4]. Where the same partition was used for both the physical and spectral spaces. This was a convenient approach for relatively low numbers of CPUs, but eventually limited the number of partition elements in the periodic direction. Here, in order to optimise the different parts of the algorithm, the partitions of both spaces are chosen independently. This has produced a significant expansion on the range of efficient scalability of the overall solver: maximal tests with 128 partition elements in the periodic direction, engaging 8192 CPU cores in total, show that the scalability is not exhausted yet.

2 Poisson solver

In an operator-based formulation, the finite volume spatial discretisation of the Navier-Stokes equations reads

$$\Omega\frac{du_h}{dt}+C(u_h)u_h+Du_h+\Omega Gp_h=0_h, \qquad Mu_h=0_h, \tag{1}$$

where u_h and p_h are the velocity and pressure fields, Ω is a diagonal matrix with the size of the control volumes, $C(u_h)$ and D are the convective and diffusive operators and, finally, M and G are the divergence and gradient operators, respectively. Arranging the pressure-velocity coupling by means of a classical fractional step projection method [5], leads to a Poisson equation, with matrix $\mathsf{L}=-M\Omega^{-1}M^*$, to be solved on each time step in order to find the pressure-correction field:

$$\mathsf{L}p_h^{n+1}=b_h^n \qquad n=1,....,N_t, \tag{2}$$

where b_h^n depends on u_h^n and some of the previous velocity fields; the Laplacian operator, L, is by construction symmetric and negative-definite and N_t is the total number of time-steps. Since the mesh does not change, L remains constant during all the simulation. Thus, any preprocessing phase is reduced N_t times.

The dimension of the Poisson system is $N=N_{2d}\times N_{per}$, where N_{2d} and N_{per} are the size of the 2D component of the mesh and of the extrusion direction, respectively. The uniformly and periodicity, permit to decompose the system, by means of a FFT-based diagonalisation [1], into N_{per} uncoupled *two-dimensional* systems:

$$\widehat{\mathsf{L}}_k\hat{x}_k^{2d}=\hat{b}_k^{2d} \qquad k=0,...,N_{per}-1, \tag{3}$$

where each system, hereafter denoted as *frequency system*, corresponds to a frequency in the Fourier space. These systems are solved by means of a direct Schur-complement based method [6, 7, 8, 9], described shortly in the next paragraphs.

For each 2D system in (3), the set of unknowns, here named $\mathcal{Y}$, is partitioned into P_{2d} subsets, $\{\mathcal{Y}_0,...,\mathcal{Y}_{P_{2d}-1}\}$, becoming $\mathcal{Y}_k$ the local unknowns of process k. To decouple the system, it is needed an interface subset, $\mathcal{S}\subset\mathcal{Y}$, fulfilling the following property:

$$\{i \in \mathcal{Y}_k \cap \mathcal{S}^c,\ j \in \mathcal{Y}_l \cap \mathcal{S}^c \text{ and } k \neq l\} \Longrightarrow \{l_{ij} = l_{ji} = 0\}. \tag{4}$$

where $\mathcal{S}^c$ is the complement of $\mathcal{S}$ in $\mathcal{Y}$.This is, two *local* non-*interface* variables of different processes cannot be directly coupled by the system. The subsets $\mathcal{S}_k := \mathcal{Y}_k \cap \mathcal{S}$ and $\mathcal{U}_k := \mathcal{Y}_k \cap \mathcal{S}^c$, are here named the *local interface* and *local inner* unknowns of process k, respectively. Then, labeling the unknowns in the order $\mathcal{U}_0, \ldots \mathcal{U}_{P_{2d}-1}, \mathcal{S}$, and performing a block Gaussian elimination, the block structure of the system becomes

$$\begin{pmatrix} B & E \\ 0 & \tilde{C} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ \tilde{g} \end{pmatrix}, \tag{5}$$

where

$$B = \bigoplus_{i=0}^{P_{2d}-1} B_i, \tag{6}$$

is a block diagonal matrix. The sub-blocks B_i are the couplings between the i'th *local inner* unknowns. E are the couplings between the *inner* and *interface* unknowns, $F = E^T$ are the coupling between the *interface* and the *inner* unknowns. C are the linear couplings between the *interface* variables and $\tilde{C} = C - FB^{-1}E$ is the Schur Complement (SC) matrix that results from the Gaussian elimination. $\tilde{g} = g - FB^{-1}f$ its r.h.s. term. With this structure, once the interface variables are evaluated, each subdomain can be solved independently. Note that, although the *inner* systems with matrices B_i are solved twice, first to find $\tilde{g}$ and then solve the inner unknowns, they are mutually independent and can be solved simultaneously. This is the main concept of the SC techniques: to separate, by means of a common distributed *interface*, a subset of the *local* unknowns of each process and solve them independently.

The determination and load balancing of the interface is solved by means of an in-house algorithm (see [2] for more details). The set of *inner* systems are solved by means of a sparse Cholesky factorisation [10]. The parallel solution of the *interface* system is carried out by means of an explicit evaluation of $\tilde{C}^{-1}$.

The parallelisation of the overall (FFT+DSD) solver is based on a geometric domain decomposition into P subdomains, one for each parallel process. Standard MPI is used to implement the algorithm. The partition of the initial mesh $\mathcal{M}$ is carried out by dividing its *two-dimensional*, $\mathcal{M}_{2d}$, and periodic, $\mathcal{M}_{per}$, components into P_{2d} and P_{per} parts respectively, being $P = P_{2d} \cdot P_{per}$. This is referred as a $P_{2d} \times P_{per}$-partition. Different partitions are employed for the FFT-based change-of-basis (from *physical* to *spectral* space and vice versa) and for the solution of the *frequency systems*. The former operation is performed without partitioning the mesh in the periodic direction, a $P \times 1$-partition is used. On the other hand, when solving the *frequency systems* , the number of processes employed, P_{2d}, must be in the range of linear scalability of the DSD algorithm. Despite the additional transmissions of data between these two partitions, this strategy benefits the scalability of the overall algorithm. This reads:

FFT+DSD algorithm:

1. Evaluate $\hat{b} = (\mathsf{I}_{N_{2d}} \otimes \mathsf{F}^*_{N_{per}})b$ on the $P \times 1$-partition.
2. Redistribute $\hat{b}$ from the $P \times 1$- to the $P_{2d} \times P_{per}$-partition MPI_Alltoall.
3. Solve the the *frequency systems*, $\hat{\mathsf{L}}_k \hat{x}^{2d}_k = \hat{b}^{2d}_k$, on the $P_{2d} \times P_{per}$-partition.
4. Redistribute $\hat{x}$ from the $P_{2d} \times P_{per}$- to the $P \times 1$-partition, MPI_Alltoall.
5. Evaluate $x = (\mathsf{I}_{N_{2d}} \otimes \mathsf{F}_{N_{per}})\hat{x}$ on the $P \times 1$-partition.

Where $\mathsf{F}_{N_{per}}$ and $\mathsf{F}^*_{N_{per}}$ are the N_{per}-dimensional Fourier transform ant its inverse/adjoint.

3 Numerical experiment

All the numerical tests presented in this paper have been carried out on the MareNostrum supercomputer of the Barcelona Supercomputing Center (BSC). To avoid dispersion, results have been obtained after averaging over several time-steps.

Firstly, the DSD solver is tested separately. Speed-up results starting from 4 up to 100 CPUs are displayed in Figure 1 for 2D meshes ranging between 250,000 (m250m) and 2,000,000 (m2M) nodes. At first sight, we can observe that all the meshes of different sizes show the same qualitative behavior. Nevertheless, as expected, the speed-up improves with the size of the problem.

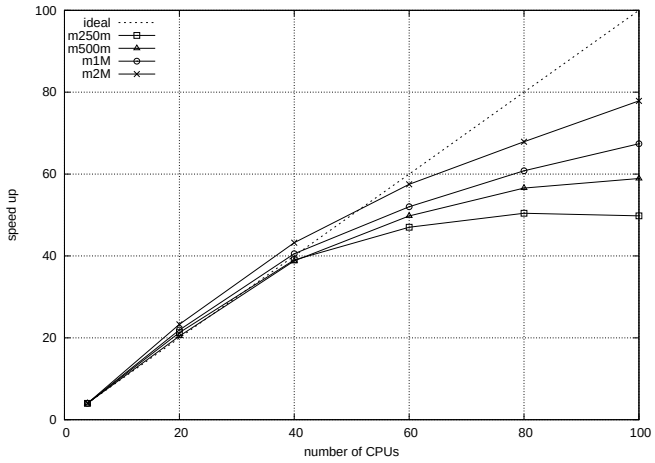

Figure 1: *Strong* speed-up of DSD solution phase measured in meshes of different sizes.

In Figure 2 (left), the weak scalability when the mesh grows in the periodic direction is displayed. The meshes for this test have been generated using m2M as 2D component, reaching up to 1024 million grid points. P_{2d} is fixed equal to 64, which is in the limit of the linear speedup region of the DSD algorithm for this mesh (see Figure 1). Therefore, the parallel efficiency of the DSD component will be close to one. Initially N_{per} and P_{per} are 8 and 1, respectively, thus the load per CPU is approximately $125,000$ nodes. Then P_{per} and N_{per} are increased 128 times, while the solution time only grows 1.5 times. Therefore, the size of the problem varies from 8 to 1024 million nodes, and the wall clock time spent in the solution from 0.27 to 0.42 seconds. The setup time is less than 30 minutes in all cases. In practice, for time-accurate simulations on such a meshes, this time is almost negligible compared with the expected accumulated solution time.

Finally, in Figure 2 (right), the strong speedup for the overall algorithm is studied on a 512 million nodes mesh, generated from the extrusion of m2M ($N_{per} = 256$). Initially $P = 2048$ ($P_{2d} = 32$, $P_{per} = 64$), when doubling P_{2d} and P_{per} (up to 8192 CPUs) the parallel efficiencies obtained are 0.84 and 0.87, respectively. The walk clock time spent in the solution varies from 0.69 to 0.24 seconds.

4 Concluding remarks

A parallel direct algorithm for the solution of the Poisson equation arising in incompressible flows with one periodic direction has been presented. It is a combination of a Direct Schur-complement based Decomposition (DSD) and a Fourier diagonalisation. The scalability and efficiency of the proposed method have been shown by performing several numerical experiments on the MareNostrum Supercomputer. Scalability tests using up to 8192 parallel processes with up to 10^9 million nodes meshes have demonstrated the algorithm capability on solving large-scale problems with a very short time.

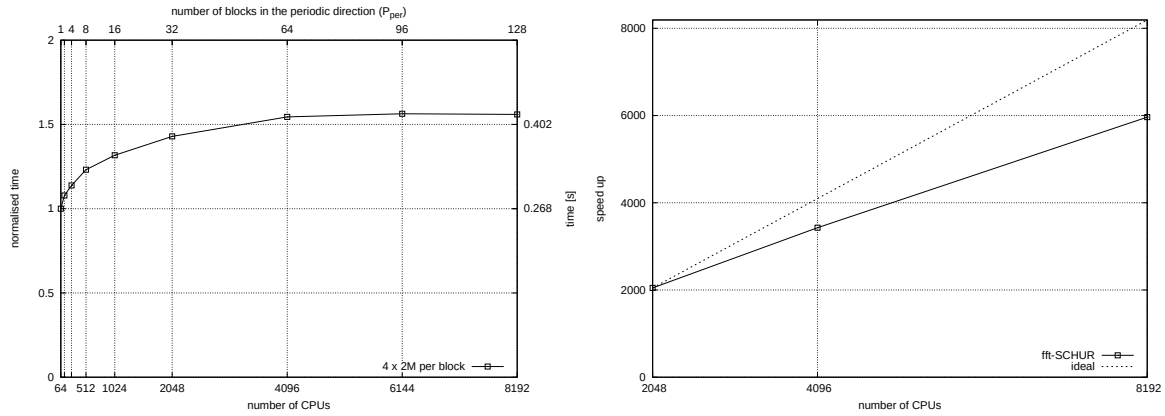

Figure 2: Left: *Weak* speed-up in the periodic direction for meshes generated by the extrusion of the mesh m2M, the load per CPU is kept around 125000 nodes. The size of the problem (and the wall-clock time) varies from 8 million (0.27s) to 1024 million (0.42s), respectively. Right: *Strong* speedup of the overall algorithm for a 512 million nodes mesh generated from the extrusion of m2M. On the first step P_{2d} is doubled from 32 to 64 and on the second P_{per} is doubled form 64 to 128. The walk clock time spent in the solution varies from 0.69 to 0.24 seconds.

Acknowledgements

This work has been financially supported by the Ministerio de Ciencia e Innovación of Spain (Contracts/ Grants No. ENE2010-17801 and No. ENE2009-09496), and Termo Fluids S.L. Calculations have been performed on the MareNostrum supercomputer at the Barcelona supercomputing center. The authors thankfully acknowledge these institutions.

References

[1] R. M. Gray, Toeplitz and Circulant Matrices: A review, Foundations and Trends in Communications and Information Theory 2 (2006) 155–239.

[2] R. Borrell, O. Lehmkuhl, F. X. Trias, A. Oliva, Parallel direct Poisson solver for discretisations with one Fourier diagonalisable direction, Journal of Computational Physics. 230 (2011) 4723–4741.

[3] A. Gorobets, F. X. Trias, M. Soria, A. Oliva, A scalable parallel Poisson solver for three-dimensional problems with one periodic direction, Computers & Fluids 39 (2010) 525–538.

[4] R. Borrell, O. Lehmkuhl, M. Soria, A. Oliva, Schur Complement Methods for the solution of Poisson equation with unstructured meshes, in: Parallel Computational Fluid Dynamics, Elsevier, Antayla, Turkey, 2007.

[5] A. J. Chorin, Numerical Solution of the Navier-Stokes Equations, Journal of Computational Physics 22 (1968) 745–762.

[6] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2003.

[7] S. Kocak, H. U. Akay, Parallel Schur complement method for large-scale systems on distributed memory computers, Applied Mathematical Modelling 25 (2001) 873–886.

[8] N. Rakowsky, The Schur Complement Method as a Fast Parallel Solver for Elliptic Partial Differential Equations in Oceanography, Numerical Linear Algebra with Applications 6 (1999) 497–510.

[9] M. Soria, C. D. Pérez-Segarra, A. Oliva, A Direct Parallel Algorithm for the Efficient Solution of the Pressure-Correction Equation of Incompressible Flow Problems Using Loosely Coupled Computers, Numerical Heat Transfer, Part B 41 (2002) 117–138.

[10] I. S. Duff, A. M. Erisman, J. K. Reid, Direct Methods for Sparse Matrices, Oxford University Press, 1989.