International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:3, 2007

Strip Decomposition Parallelization of Fast Direct Poisson Solver on a 3D Cartesian Staggered Grid

Minh Vuong Pham, Frédéric Plourde, Son Doan Kim

***Abstract*—**A strip domain decomposition parallel algorithm for fast direct Poisson solver is presented on a 3D Cartesian staggered grid. The parallel algorithm follows the principles of sequential algorithm for fast direct Poisson solver. Both Dirichlet and Neumann boundary conditions are addressed. Several test cases are likewise addressed in order to shed light on accuracy and efficiency in the strip domain parallelization algorithm. Actually the current implementation shows a very high efficiency when dealing with a large grid mesh up to 3.6×10^9 under massive parallel approach, which explicitly demonstrates that the proposed algorithm is ready for massive parallel computing.

***Keywords*—**Strip-decomposition, parallelization, fast direct poisson solver.

I. Introduction

THE Poisson equation is one of the most basic equations in scientific calculation in many different fields of interest such as, for instance, acoustics, electromagnetism and fluid dynamics, just to quote a few. In fluid dynamics, among the flows that are of importance and interest to mankind, the category of low Mach-number or incompressible flow is by far the largest. As recently put forward by Löhner et al. [1], incompressible solvers have recently been enhanced in a variety of ways such as sub-stepping for advection, implicit treatment of convective terms and linelet preconditioning flow the pressure-Poisson equation. The combined effect of these improvements leads to speed-up factors of the order of 1 to 10; In addition, massive parallel approach is a way to significantly reduce time simulation for academic and industrial problems. However, even if numerous solvers have already been addressed in the past in view of solving the pressure-Poisson equation, solver efficiency drastically falls with parallel computing. This is naturally a key problem because pressure-Poisson equation solving requires significant CPU time.

A standard procedure for solving the Poisson equation by direct method requires $O(N^6)$ operations for N^3 grid points in a 3D problem, which makes such a direct method prohibitive for a large-scale physical problem. Fast direct Poisson solver development began in 1965 when Hockney [2] used Fourier analysis and fast Fourier transform (FFT) [3], which leads to computation speed up to $O(N\log N)$ on a $N\times N$ grid. Such a method is often used [4, 5, 6] for 3D Poisson problems [7, 8, 9] since it costs only about $O(N^3\log N)$ for a $N\times N\times N$ grid points. With large-scale problems, particularly incompressible 3D Navier-Stokes equations, large mesh point grids are carried out. Therefore, parallelization of Poisson solver is required and the latter must be as fast as possible. Thus, several methods for parallel fast Poisson algorithms have been developed and one should first distinguish two main groups of parallel techniques dependent on parallel machine architecture. The first one was proposed according to parallel vectorial machine [10, 11]. Thus, solution of Poisson equation is carried out by distributing various computational segments from the direct solver. That is to say, parallel algorithm was developed for both fast Fourier transform and solution of tridiagonal systems [12]. The second group considered a massive parallel philosophy [13, 14, 15, 16] in which the computational domain is split into multi-domains [17]. Each computational sub-domain is then performed on one single processor and data transfer between processors is naturally requested. Several techniques of domain decomposition were presented for 2D Poisson solver [13, 18] using FFT parallelization on multiprocessors [19, 20]. Unfortunately, such development is rather complex to achieve and, in addition, its cost in data transfer is significantly high.

However, fast direct Poisson solver on a massively parallel machine was first introduced by Swarztrauber and Sweet [21] in which FFT parallelization is avoided. Each processor should perform the same amount of data and the latter are then dynamically redistributed between the whole sub-domains. This approach is called the transposed method, a technique that is generally preferred on account of several advantages. First, and foremost, it requires less communication between processors than other techniques. A second reason to prefer the transposed method is its simplicity of use. The method is based only on already existing efficient algorithms and no specific and complex development is required to address parallel FFT. This technique has successfully been used for the solving of a 2D cylindrical Poisson solver problem [22] but to author's knowledge, transposed method has been rarely

Minh Vuong PHAM, LET – ENSMA, UMR CNRS 6608, 1, avenue Clément Ader, BP 40109, 86960 Chasseneuil Futuroscope Cedex, France (e-mail: phammg@ensma.fr).

Frédéric PLOURDE, LET – ENSMA, UMR CNRS 6608, 1, avenue Clément Ader, BP 40109, 86960 Chasseneuil Futuroscope Cedex, France (corresponding author to provide phone: +33549498119; fax: +33549498130; e-mail: plourde@let.ensma.fr).

Son DOAN KIM, LET – ENSMA, UMR CNRS 6608, 1, avenue Clément Ader, BP 40109, 86960 Chasseneuil Futuroscope Cedex, France (e-mail: doan@let.ensma.fr).

applied to tri-dimensional problems.

Direct methods are often used ensuring stability in the computing process, which is often requested especially when Poisson equation must be solved every time-step. Then, fast algorithms are generally built to take advantage on FFT algorithm. However, FFT algorithms bring their own drawbacks. Spatial discretization must be constant in the spectral directions, i.e. which render complex geometry in fluid mechanics for instance not easy to target. However, a new trend in fluid mechanics solver is actually under development. Nowadays, there is interest in investigating a novel grid generation concept, the Immersed Boundary Method, as an improved methodology to boundary fitted grids. With such algorithms, the grid does not coincide any longer with the geometry being solved. Since the grid does not fit the surface geometry, this type of grid used is often chosen for computational efficiency instead of geometry, which allows the use of simple orthogonal grids. Furthermore the grid generation complexity and time is greatly reduced as the complex geometry only needs to be mapped onto the underlying orthogonal grids. Yang and Balaras [23], Balaras [24], Fadlum et al. [25], just to mention a few, successfully approaches carried out with various algorithms to address complex flow simulations. Ranges of application are naturally very broad. A parallel version of such method is easy to develop even if the major difficulties are to solve Poisson equation in an efficient and fast way.

The aim of this paper is to propose a very fast parallel way for solving 3D Poisson solver. As far as the author's knows, only 2D Poisson equations were addressed with a transposed method. In addition, in a massively parallel implementation, one should develop a very efficient and clever way of parallelization in order to provide efficient computing. As we will see later, strip domain decomposition will be found to be sufficiently efficient when solving the Poisson equation. Therefore, we will also analyze the efficiency of direct fast Poisson solver when dealing with a massive parallel environment. First, section 2 briefly describes and rapidly recalls the mathematical preliminaries of Fast Fourier Transform. Sequential implementations as well as parallel algorithm based on strip decomposition are described. Naturally, the key point concerns its efficiency. Therefore, two test cases are shown with close attention being paid to errors and efficiency of the parallel implementation. Data transfer may be considered as a major drawback, and efficiency will be tested on massive parallel simulations with mesh grid points up to 3.6×10^9 points so as to shed light on the potential of the proposed algorithm.

II. Preliminary analysis

Mathematical formulation for fast direct solver of the Poisson equation is first recalled; we arbitrarily chose to focus on a staggered grid arrangement in the Cartesian coordinates. However, application of the proposed algorithm to co-located grids is straightforward. The Poisson equation in (x, y, z) system is:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=f(x,y,z) \quad \text{for } (x,y,z)\in\Omega \tag{1}$$

where $\Omega=(0,L_x)\times(0,L_y)\times(0,L_z)$ is the computational domain. Boundary conditions applied on each limit face may be either the Dirichlet or the Neumann boundary condition. For the Dirichlet boundary condition:

$$u(x,y,z)=\phi(x,y,z) \quad \text{for } (x,y,z)\in\partial\Omega \tag{2}$$

where $\phi(x,y,z)$ are the values of solution at $\partial\Omega$, i.e. boundary condition. For the Neumann boundary condition:

$$\frac{\partial u(x,y,z)}{\partial x_i}=\phi(x,y,z) \tag{3}$$

for $x_i=(x,y,z)$ and $(x,y,z)\in\partial\Omega$

In order to discretize the Poisson equation, the domain Ω is covered with a regular grid mesh $(\Delta x,\Delta y,\Delta z)$ corresponding to (x,y,z) directions with (L,M,N) points in each direction. Each grid point (x_i,y_j,z_k) is obtained by $x_i=(i-0.5)\Delta x$, $y_j=(j-0.5)\Delta y$, $z_k=(k-0.5)\Delta z$ for $i=1:L, j=1:M, k=1:N$. Replacing the derivatives by second-order central-difference approximation, the interior points of Poisson's equation can be written as:

$$\frac{u_{i+1,j,k}-2u_{i,j,k}+u_{i-1,j,k}}{\Delta x^2}+\frac{u_{i,j+1,k}-2u_{i,j,k}+u_{i,j-1,k}}{\Delta y^2}+\frac{u_{i,j,k+1}-2u_{i,j,k}+u_{i,j,k-1}}{\Delta z^2}=f(i,j,k) \tag{4}$$

while the boundary values $u_{i,j,k}$ are specified at the interface of Ω.

The direct Poisson solver was performed by a direct Fourier transform. Note that a direct transform method requires $O(N^2)$ arithmetic operations for N points i.e. such a method is exceedingly expensive in terms of CPU time. However, a very clever and well-known algorithm, the Fast Fourier transform (FFT) algorithm [3] requires $O(N\log N)$ arithmetic operations to directly solve the Poisson equation. According to the kind of boundary condition applied, sine or cosine transforms are addressed. The fast Fourier-sine transform is used for the Dirichlet boundary condition and the fast Fourier-cosine transform is required for the Neumann boundary condition. On a 3D staggered grid, these fast Fourier transforms can be written as:

In z direction:

$$\hat{u}_n=\sqrt{\frac{2}{N}}\sum_{k=1}^{N}u_k\sin\left(\frac{\pi n(k-1/2)}{N}\right) \tag{5}$$

$\Rightarrow$ Dirichlet boundary condition

$$\hat{u}_n=\sqrt{\frac{2}{N}}\sum_{k=1}^{N}u_k\cos\left(\frac{\pi (n-1)(k-1/2)}{N}\right) \tag{6}$$

$\Rightarrow$ Neumann boundary condition

The resulting equation, after applying fast Fourier transform in *z* direction, is written as:

$$\frac{\hat{u}_{i+1,j,n}-2\hat{u}_{i,j,n}+\hat{u}_{i-1,j,n}}{\Delta x^2}+\frac{\hat{u}_{i,j+1,n}-2\hat{u}_{i,j,n}+\hat{u}_{i,j-1,n}}{\Delta y^2}+\lambda_z\hat{u}_{i,j,n}=\hat{f}(i,j,n) \quad (7)$$

where:

$$\lambda_z=\frac{1}{\Delta z^2}\left(2\cos\left(\frac{k\pi}{N}\right)-2\right) \quad (8)$$

$\Rightarrow$ Dirichlet boundary condition

$$\lambda_z=\frac{1}{\Delta z^2}\left(2\cos\left(\frac{(k-1)\pi}{N}\right)-2\right) \quad (9)$$

$\Rightarrow$ Neumann boundary condition

In *y* direction, the same Fourier transforms are applied for *M* points. The resulting equations after fast Fourier transform in *y* and *z* directions are rewritten as:

$$\frac{\hat{u}_{i+1,m,n}-2\hat{u}_{i,m,n}+\hat{u}_{i-1,m,n}}{\Delta x^2}+(\lambda_y+\lambda_z)\hat{u}_{i,m,n}=\hat{f}(i,m,n) \quad (10)$$

where:

$$\lambda_y=\frac{1}{\Delta y^2}\left(2\cos\left(\frac{j\pi}{M}\right)-2\right) \quad (11)$$

$\Rightarrow$ Dirichlet boundary condition

$$\lambda_y=\frac{1}{\Delta y^2}\left(2\cos\left(\frac{(j-1)\pi}{M}\right)-2\right) \quad (12)$$

$\Rightarrow$ Neumann boundary condition

The final tridiagonal systems can be written as:

$$\frac{1}{\Delta x^2}\hat{u}_{i+1,m,n}-\left(\frac{2}{\Delta x^2}-\lambda_y-\lambda_z\right)\hat{u}_{i,m,n}+\frac{1}{\Delta x^2}\hat{u}_{i-1,m,n}=\hat{f}(i,m,n) \quad (13)$$

The systems contain $L\times M\times N$ periodic tridiagonal equations and the system of equations can be solved by using a direct method in order to determine the solutions in Fourier's space. Following that, solutions in the physical space can be reconstructed via a double inverse Fourier transform in the *y* and *z* directions. For example, the formulations of inverse Fourier - sine and Fourier - cosine in *z* direction are written as:

$$u_k=\sqrt{\frac{2}{N}}\sum_{n=1}^{N}\hat{u}_n\sin\left(\frac{\pi n(k-1/2)}{N}\right) \quad (14)$$

$\Rightarrow$ Dirichlet boundary condition

$$u_k=\sqrt{\frac{2}{N}}\left(\frac{1}{2}\hat{u}_1+\sum_{n=2}^{N}\hat{u}_n\cos\left(\frac{\pi(n-1)(k-1/2)}{N}\right)\right) \quad (15)$$

$\Rightarrow$ Neumann boundary condition

In this section, homogeneous boundary conditions were addressed whereas all the non-homogeneous boundary conditions could be found in [5] and through these mathematical preliminaries, sequential and mainly parallel algorithm is presented in the following sections.

III. Sequential and Parallel Algorithm

The sequential algorithm of fast direct Poisson solver for 3D Cartesian staggered grid problem is straightforward to the mathematical preliminaries presented above. For a $L\times M\times N$ grid mesh, the main tasks for solving the Poisson equation are listed:

- Compute fast Fourier transform in *z* direction for *LM* sets of data $f(i,j,k)$ for $k=1,N$ following different boundary conditions.
- Compute fast Fourier transform in *y* direction for *LN* sets of data $\hat{f}(i,j,n)$ for $j=1,M$ following different boundary conditions.
- Solve *LMN* periodic tridiagonal systems using the direct method.
- Compute the inverse fast Fourier transform in *y* direction for *LN* sets of data $\hat{f}(i,m,n)$ for $m=1,M$.
- Compute the inverse fast Fourier transform in *z* direction for *LM* sets of data $\hat{f}(i,j,n)$ for $n=1,N$.

The sequential algorithm recalled just above is a well-known fast algorithm to access the solution of the Poisson equation. However, insofar as a parallel computation is required, a clever strategy must be selected. To provide a fast and efficient algorithm, a strip domain decomposition is proposed. The computational domain is decomposed into several sub-domains, which are distributed on each processor. The parallelization for fast Fourier transform is thereby avoided for the proposed technique through the use of crossing data transfers between every sub-domain and processors. Strip domain decomposition also offers additional benefits. First, for a specified grid mesh, the number of total operations on all sub-domains for a parallel algorithm is similar to the one necessary for sequential algorithm, which

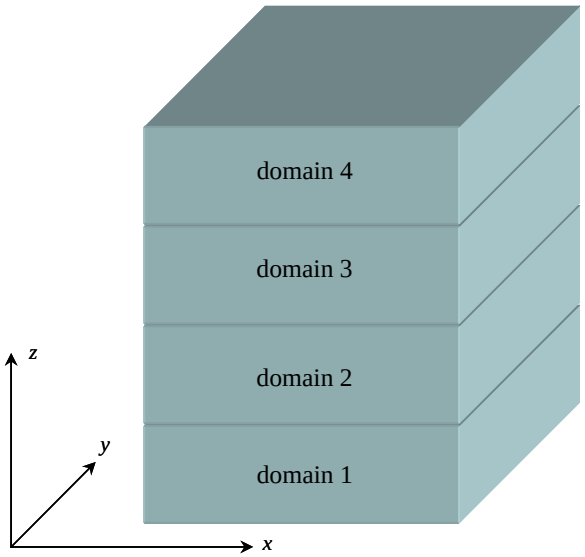


Fig. 1 Schematic of strip domain decomposition in *z* direction on a 3D Cartesian grid for fast direct Poisson solver parallel algorithm.

means that total computational times of all processors used are exactly equal to those of a single processor. Second, the proposed algorithm can use almost the same subroutines as the sequential algorithm. Finally, for a large grid mesh and a large number of processors, the communication time between the processors is minimized.

As mentioned in the introduction, our goal is to achieve efficient parallel algorithm for solving the Poisson equation on different architectures. However, with advances in parallel computing, massive parallel approach with domain decomposition is more and more widely used. The choice of strip domain decomposition method allows one to avoid solving fast Fourier transforms with parallel algorithms; in fact, strip decomposition is performed in order to weaken as much as possible the effect of data transfer on algorithm efficiency. The computational domains divided into *P* sub-domains in the *z* direction as shown in Figure 1, where a 4-sub-domain decomposition is given as an example.

The data in the *z* direction is divided into *P* sub-domains, whose contents *N/P* points in the *z* direction. Then, each sub-domain contains data associated with the grid points $f(i, j, k)$ with $(i = 1:L,\ j = 1:M, k = pN/P + 1:(p+1)N/P)$, in which *p* corresponds to the index of the considered sub-domain. To avoid parallelization of Fast Fourier Transform on multi-domains, one needs to transfer necessary data. In order to minimize time transfer, each strip is also divided (virtually) into *P* sub-domains corresponding to the *P* processors in use. Cross-data transfers between the *P* sub-domains and *P* processors are then necessary and are schematically shown in Figure 2. As soon as data are transferred, each processor contains the *LM/P* data of the $f(i, j, k)$ right-hand side term

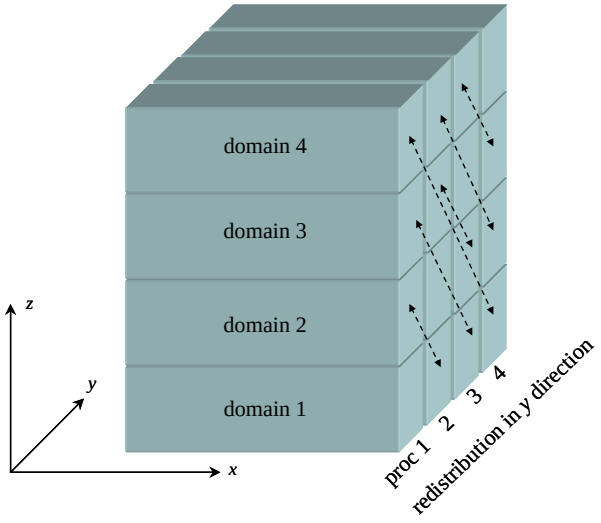


Fig. 2 Schematic of strip domain decomposition in *z* direction, the redistribution of data in *y* direction for several processors and crossing transfer between sub-domain decomposition and processor data distribution, for a 3D Cartesian grid for the parallel algorithm fast direct Poisson solver.

with $(i = 1:L,\ j = pM/P + 1:(p+1)M/P, k = 1:N)$ for processor *p*. In order to optimize computational time, FFT on each processor are computed simultaneously. At this point, the discretized equation given in (7) corresponds to a 2D Cartesian staggered grid, where each solution of *LM* points in the *xy* plan corresponds to a Fourier mode in the z direction

To optimize the solver of *N* systems of the 2D Poisson equation, the computational data are redistributed on the *P* processors. Thus, each processor evaluates the solution of one sub-domain decomposition. To do so, it is essential to carry out a cross-data transfer between the processors and the sub-domains in order to obtain the complete data after fast Fourier transform in the *z* direction for each sub-domain. After that, each sub-domain contains *N/P* sets of the *LM* coefficients of the complete fast Fourier transform. Therefore, each processor evaluates *N/P* systems of 2D Poisson equations. The FFT in the *y* direction is carried out for each 2D Poisson equation in order to obtain the final tridiagonal systems (13). Finally, the tridiagonal systems are solved for each 2D Poisson equation. The different steps of the proposed algorithm can be summarized as follows:

- Each processor performs *(P-1)* sets of LMN/P^2 cross-data transfer between the strip sub-domain and data divided by *P* processors in *y* direction.
- A set of *LM/P* fast Fourier transforms, which contains *N* points in *z* direction, is computed on each processor.
- Each processor carries out *(P-1)* sets of LMN/P^2 cross-data transfer between the data divided by *P* processors in the *y* direction and the strip sub-domain.
- Each processor solves a set of *N/P* systems of the 2D Poisson equation (*LM* points) by using the fast direct solver of the 2D staggered grid Poisson equation. Each 2D Poisson system corresponds to a mode of Fourier transform. Compute the *LN/P* fast Fourier transforms of the *M* points in *y* direction for each processor. Solve *LMN/P* tridiagonal systems per processor to obtain the solution of problem in the Fourier space. Compute the *LN/P* inverse fast Fourier transforms of the *M* points in *y* direction from the solution in the Fourier space for each processor.
- Each processor performs *(P-1)* sets of LMN/P^2 crossing data transfer between the strip sub-domain and the data divided by *P* processors in the *y* direction.
- A set of *LM/P* inverse fast Fourier transforms, which contains *N* points in the *z* direction, is computed on each processor.
- Each processor carries out *(P-1)* sets of LMN/P^2 crossing data transfer between the data divided by *P* processors in the *y* direction and the strip sub-domain.

Finally, the solution of the problem is obtained in the physical space. Therefore, each sub-domain Ω, corresponding to the processor *p*, contains the solution associated with the grid points $u(i, j, k)$ with $(i = 1:L,\ j = 1:M, k = pN/P + 1:(p+1)N/P)$. Such

algorithm is regarded as an attractive algorithm because, despite domain decomposition, the Poisson equation is based on a direct FFT solver. The major drawback is certainly linked to the amount of data transfer but to shed light on the efficiency of the proposed algorithm, numerical tests are required.

IV. Numerical tests

The main goal of the several numerical test cases shown hereinafter is to highlight accuracy and efficiency of the proposed strip domain decomposition parallelization algorithm for fast direct Poisson solver. To achieve portability, the MPI library [26] is used for inter-communication between processors. For accuracy, relative errors in norms $\|\bullet\|_\infty$ will be shown by computing the difference between the numerical and the exact solutions related to the exact solution and accuracy of the method depends on the number of discretized points in each direction.

As mentioned in the introduction, performance of the proposed algorithm is a key point for the physical problems addressed. Thus, estimation of operations for the algorithm is essential [21, 22]. Many tests are carried out in this section by using a significant number of processors on a MPI library to estimate the time required for several grid point resolutions.

A. Accuracy of fast direct Poisson solve

In order to estimate accuracy of the fast direct Poisson solver, two problems were solved and results obtained were directly compared with the exact solution. In addition, both

TABLE I
Relative errors in norms $\|\bullet\|_\infty$ and running times for Problem 1 with regard to the number of grid points and processors for two tests of boundary conditions

	Grids				Processors	Times	Errors	
Case	*L*	*M*	*N*	$L\times M\times N$	*P*	*t(s)*	Dirichlet	Neumann
1	32	32	32	3.3×10^4	1	0.0	2.61×10^{-4}	2.41×10^{-3}
2	64	64	64	2.6×10^5	1	0.04	6.52×10^{-5}	6.09×10^{-4}
3	128	128	128	2.1×10^6	1	0.51	1.63×10^{-5}	1.53×10^{-4}
4	256	256	256	1.7×10^7	1	6.18	4.08×10^{-6}	3.84×10^{-5}
5	512	512	512	1.3×10^8	16	6.05	1.02×10^{-6}	9.42×10^{-6}
6	1024	1024	1024	1.0×10^9	64	22.6	2.55×10^{-7}	2.35×10^{-6}
7	1280	1280	1280	2.1×10^9	64	41.0	1.63×10^{-7}	1.51×10^{-6}
8	1536	1536	1536	3.6×10^9	256	24.0	1.11×10^{-7}	1.03×10^{-6}

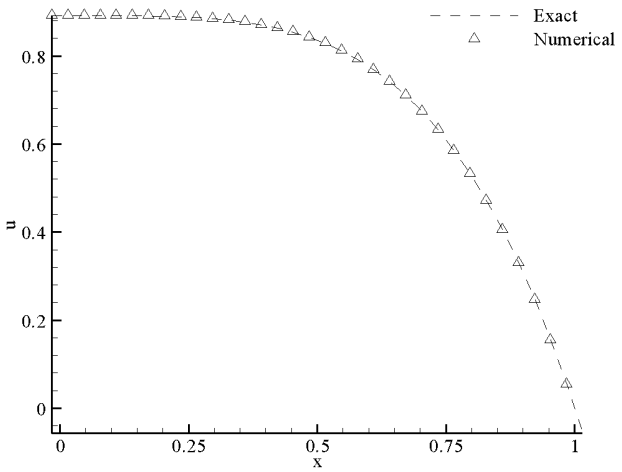


Fig. 3 Solution of Poisson's equation for a polynomial problem in a 3D Cartesian staggered grid with Dirichlet boundary conditions in *x,y,z* direction. The solution is plotted versus *x* at $y = 0.5$ and $z = 0.5$. Dotted curve shows the exact solution, whereas the open triangles correspond to the numerical solution for $32\times32\times32$ grid points.

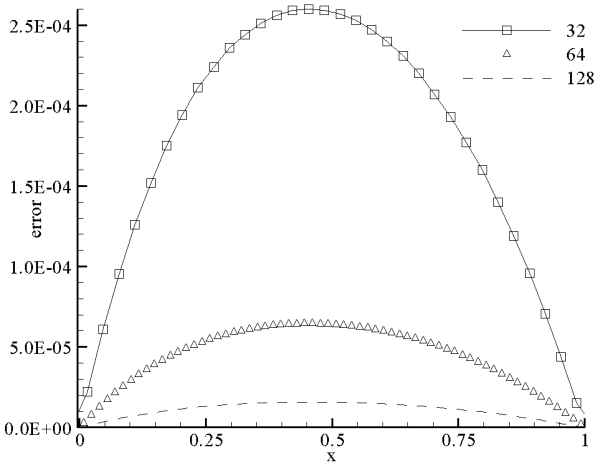


Fig. 4 Errors of Poisson's equation for a polynomial problem in a 3D Cartesian grid with Dirichlet boundary conditions in *x,y,z* direction. The error is plotted versus *x* at $y = 0.5$ and $z = 0.5$ for several grid points.

Dirichlet and Neumann boundary conditions were addressed. The numerical results were carried out by using double precision on an IBM-P4 64 bit cluster. The test cases concern a four-order polynomial distribution as well as a sine function.

Let us now use the technique discussed above to solve the Poisson equation for a four-order polynomial in a three-dimensional domain over a staggered grid. The exact solution is given by:

$$u(x,y,z) = (1-x^4)(1-y^4)(1-z^4) \tag{16}$$

on a cubic region $\Omega = (0,1)\times(0,1)\times(0,1)$. The right-hand side term of equation (4) is then given by:

$$\begin{aligned} f(x,y,z) = &-12x^2(1-y^4)(1-z^4) \\ &-12y^2(1-x^4)(1-z^4) - 12z^2(1-x^4)(1-y^4) \end{aligned} \tag{17}$$

Table 1 gives details of the several mesh grids tested on both Dirichlet and Neumann conditions. The first four cases were carried out on one single processor, i.e. with the sequential algorithm. On the contrary, as soon as grid points exceed $256\times256\times256$, simulations were carried out on several processors due to the high memory capacity required. The largest test case involved 256 processors with 3.6 billion points in the mesh grid and each sub-domain contained $1536\times1536\times6$ grid points. Under these conditions, the required computational time remained acceptable. Note that error levels listed in Table 1 concern maximum level reached in the whole sub-domains. As expected, errors were not linked to sequential or parallel algorithm but rather to grid resolution and the kind of boundary condition. Actually, errors were always higher with Dirichlet conditions than with Neumann conditions. Furthermore, these errors decrease drastically with regard to the grid point number increase in each direction.

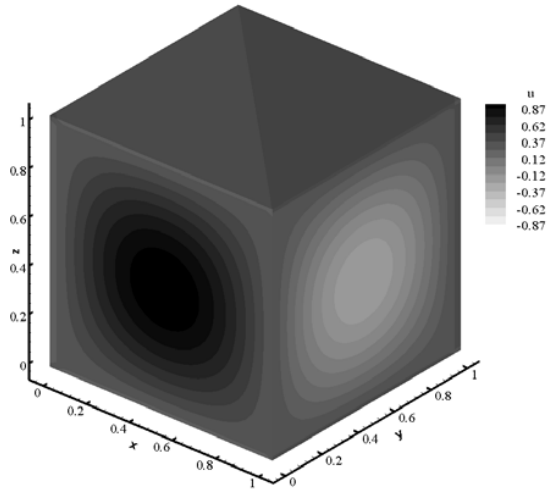
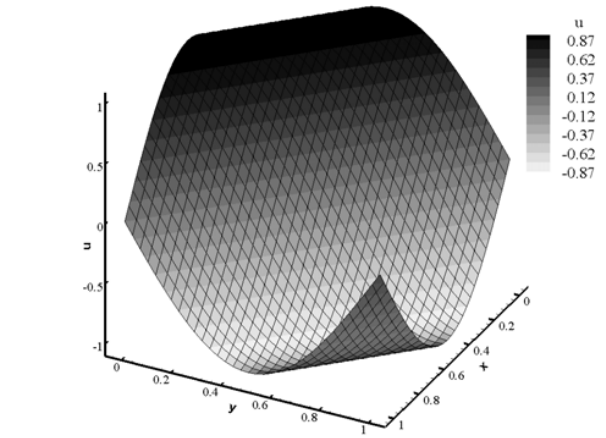


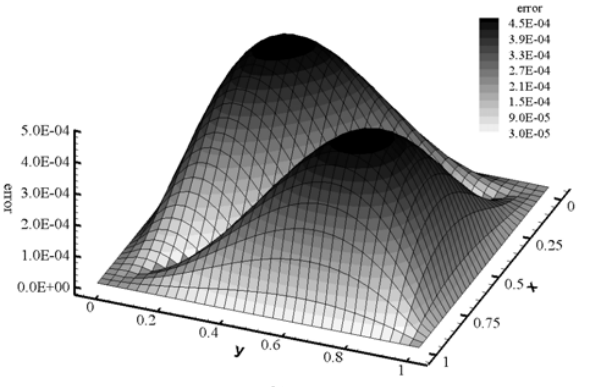
Fig. 5 Contour of Poisson's equation solution for a trigonometric case on a 3D Cartesian grid with Dirichlet boundary conditions in *x,y,z* direction. The numerical solution is plotted versus *x,y,z* for $32\times32\times32$ grid points.

Figure 3 allows one to compare numerical solutions, obtained by fast direct Poisson solver with $32\times32\times32$ grid points with the exact solution at $y = 0.5$ and $z = 0.5$. Note that, even for a small number of grid points, the numerical solution fits relatively well with the exact solution. In order to follow the error distribution, Figure 4 presents the error variation as a function of x at $y = 0.5$ and $z = 0.5$ for different grid resolutions. In this figure, the maximum error is obtained in the center of the computational domain. The latter corresponds to the maximum error shown in Table 1. Furthermore, the errors decrease rapidly with regard to the grid mesh resolutions.

Let us perform a similar analysis involving trigonometric function of sine [27] given by:



(a)



(b)

Fig. 6 Contour of numerical solution (a) and error (b) of Poisson's equation for the trigonometric case on a 3D Cartesian staggered grid with the Dirichlet boundary conditions in *x,y,z* direction. The numerical solution is plotted in a plan *xy* at $z = 0.5$ for $32\times32\times32$ grid points.

$$u(x,y,z)=\sin\left(\pi(x+y)\right)\sin\left(\pi z\right) \quad (18)$$

on a cubic region $\Omega=(0,1)\times(0,1)\times(0,1)$. The source term of Poisson's equation (4) is given by:

$$f(x,y,z)=-3\pi^2\sin\left(\pi(x+y)\right)\sin\left(\pi z\right) \quad (19)$$

Table 2 lists the errors obtained from fast direct Poisson solver applied on the sine problem with regard to several numbers of grid resolutions. It is necessary to mention that the errors are of the same order of magnitude as those obtained in Problem 1. The three-dimensional solution was shown in Figure 5 for $32\times32\times32$ grid points. In order to clearly observe variation on solution and its errors, Figure 6 plots the numerical solution (a) and the error (b) in a *xy* plan at *z* = 0.5. Maximum error levels were found to remain at a low level underlining the accuracy of the algorithm.

TABLE II
RELATIVE ERRORS IN NORMS $\|\bullet\|_\infty$ AND RUNNING TIMES FOR PROBLEM 2 WITH REGARD TO THE NUMBER OF GRID POINTS AND PROCESSORS FOR TWO TESTS OF BOUNDARY CONDITIONS

	Grids				Processors	Times	Errors	
Case	*L*	*M*	*N*	$L\times M\times N$	*P*	*t(s)*	Dirichlet	Neumann
1	32	32	32	3.3×10^4	1	0.0	4.70×10^{-4}	3.34×10^{-3}
2	64	64	64	2.6×10^5	1	0.04	1.19×10^{-4}	8.34×10^{-4}
3	128	128	128	2.1×10^6	1	0.52	2.99×10^{-5}	2.08×10^{-4}
4	256	256	256	1.7×10^7	1	6.22	7.48×10^{-6}	5.21×10^{-5}
5	512	512	512	1.3×10^8	16	6.06	1.84×10^{-6}	1.31×10^{-5}
6	1024	1024	1024	1.0×10^9	64	22.4	4.59×10^{-7}	3.26×10^{-6}
7	1280	1280	1280	2.1×10^9	64	40.9	2.94×10^{-7}	2.09×10^{-6}
8	1536	1536	1536	3.6×10^9	256	24.1	2.00×10^{-7}	1.42×10^{-6}

B. Cost of the algorithm

The cost of the proposed algorithm is naturally of great importance and the number of operations must be estimated. In addition, the time required for sequential and parallel algorithm of fast direct Poisson solver needs to be closely studied. Finally, an estimation of asymptotic speedup and efficiency of strip decomposition parallelization algorithm and its performance will be presented.

First of all, in order to estimate the number of operations and time required for the proposed algorithm, a standard communication model for distributed memory computers may be assumed. To normalize the time counting procedure, t_1 is supposed to correspond to the computational time required for fast Fourier transform of one operation, t_2 is the computational time for the solver of tridiagonal system of one operation, t_3 is the time for communication between inter-processors for a double precision number, the message startup time for parallel communication is considered as negligible. The number of operations and the computational time while using a single processor can be estimated as follows:

- *LM* fast Fourier transforms for *N* points in *z* direction costs $t_1 LMN\log N$.
- *LN* fast Fourier transforms for *M* points in *y* direction costs $t_1 LNM\log M$.
- The solution of *LMN* tridiagonal systems costs $t_2 LMN$.
- *LN* inverse fast Fourier transforms for *M* points in *y* direction costs $t_1 LNM\log M$.
- *LM* inverse fast Fourier transforms for *N* points in *z* direction costs $t_1 LMN\log N$.

Therefore, total time required for solving Poisson equation on a single processor with a fast direct solver is:

$$T_s=2\left(LMN\log N+LNM\log M\right)t_1+\left(LMN\right)t_2 \quad (20)$$

While using the strip parallel algorithm, the number of operations and the computational time can be estimated as:

- Each processor performs *(P-1)* sets of LMN/P^2 crossing communication in order to obtain the total right-hand side data for computing on each processor fast Fourier transform. This task costs $t_3(P-1)LMN/P^2$.
- Each processor carries out a set of *LM/P* fast Fourier transforms for *N* points in *z* direction. This work costs $t_1 LM/PN\log N$.
- Each processor performs *(P-1)* sets of LMN/P^2 crossing communication to obtain the total Fourier coefficients on one sub-domain. This task costs $t_3(P-1)LMN/P^2$.
- Each processor carries out a set of *LN/P* fast Fourier transforms for *M* points in *y* direction. This work costs $t_1 LN/PM\log M$.
- The solution of LMN/P tridiagonal systems for each processor costs $t_2 LMN/P$.
- A set of *LN/P* inverse fast Fourier transforms for *M* points in y direction on each processor costs $t_1 LN/PM\log M$.
- *(P-1)* sets of LMN/P^2 crossing communication to

obtain the total coefficients for the inverse fast Fourier transform for a single processor is performed. This task costs $t_3(P-1)LMN/P^2$.

- A set of *LM/P* inverse fast Fourier transforms for *N* points in *z* direction on each processor costs $t_1 LM/PN \log N$ operations.
- *(P-1)* sets of LMN/P^2 crossing communication to obtain the final solution of each sub-domain are performed. This task costs $t_3(P-1)LMN/P^2$.

Thus, the total time used for the computation of strip parallel algorithm of fast direct Poisson solver is equal to:

$$T_p = 2\left(LM/PN \log N + LN/PM \log M\right)t_1 + \left(LMN/P\right)t_2 + 4\left((P-1)LMN/P^2\right)t_3 \quad (21)$$

Finally, speed-up for the parallel algorithm can be obtained by the asymptotic estimation as:

$$S = \frac{T_s}{T_p} = P\frac{2\left(LMN \log N + LNM \log M\right)t_1 + \left(LMN\right)t_2}{2\left(LMN \log N + LNM \log M\right)t_1 + \left(LMN\right)t_2 + 4\left((P-1)LMN/P\right)t_3} \quad (22)$$

and the corresponding E efficiency coefficient can be estimated as:

$$E = \frac{S}{P} = \frac{2\left(LMN \log N + LNM \log M\right)t_1 + \left(LMN\right)t_2}{2\left(LMN \log N + LNM \log M\right)t_1 + \left(LMN\right)t_2 + 4\left((P-1)LMN/P\right)t_3} \quad (23)$$

Therefore, it is obvious that the efficiency *E* is lower than unity and that the latter decreases while increasing the number of processors involved. According to the number of operation analyses required for the fast direct Poisson solver with strip decomposition parallelization, the number of operations for the two *y* and *z* Fourier directions are $O(M \log M)$ and $O(N \log N)$ respectively when the direct solver of tri-diagonal system in *x*-direction takes about $O(L)$ operations. Thus, for a given problem involving *L*, *M* and *N* of different orders of magnitude, a clever choice is to select the direction of the highest point resolution for the tri-diagonal system resolution, the two other directions being the Fourier's direction. Note that the number of points in the two Fourier directions must be divided by the proposed processor number.

C. Efficiency

In order to estimate the efficiency of the proposed algorithm, several tests were carried out using Problem 1 with Dirichlet boundary conditions. Details on the grid mesh are given in Table 3. Case M1 corresponds to the largest test case performed on a single processor while case M5 is the largest grid mesh points performed on 256 processors. A so large number of processors was necessary to estimate the performance of the proposed algorithm (from 1 to 256 processors). Figure 7 presents running times while allocating 1, 2, 4, 8, 16, 32, 64, 128, 256 processors for several grids given in Table 3. It is obvious that the time variation is approximately linear in the logarithmic scale, i.e. the algorithm offers a good performance for parallelization. Moreover, it is important to emphasize that the slope of variation time remained constant whatever the number of grid points considered.

TABLE III
LIST OF GRIDS USED FOR TESTING PERFORMANCE AND EFFICIENCY OF THE PROPOSED ALGORITHM

Case	*L*	*M*	*N*	$L \times M \times N$
M1	256	512	512	6.7×10^7
M2	512	512	512	1.3×10^8
M3	1024	1024	1024	1.0×10^9
M4	1280	1280	1280	2.1×10^9
M5	1536	1536	1536	3.6×10^9

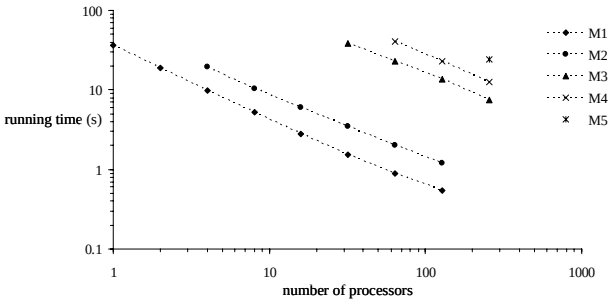


Fig. 7 Running times change with regard to the number of processors used for a polynomial problem on a 3D Cartesian grid with Dirichlet boundary conditions in *x,y,z* direction. Running times are plotted in a *log-log* scale for several grids.

Moreover, in order to evaluate the cost of parallelization of the proposed Poisson solver, speedup and efficiency were estimated. Speedup is estimated by comparing running time for a single processor against that obtained in a multiple processor architecture. Table 4 presents running times for parallel algorithm using up to 128 processors for case M1 grid. *S* Speedup factor is defined as the ratio of running time on a single processor using the sequential algorithm with time taken to solve the problem using *P* processors. For example, in Table 4, the speedup reaches about 66 for a 128 processor test.

Based on running times and speedup factor for M1 test

TABLE IV
SPEED-UPS AND EFFICIENCIES OF STRIP DECOMPOSITION PARALLELIZATION ALGORITHM OF FAST DIRECT POISSON SOLVER FOR TEST GRID M1

Number of processors	Running time (*s*)	Communication time (*s*)	Speedup *S*	Efficiency
1	36.40	0.00	1.0	1.00
2	19.00	0.80	1.9	0.96
4	9.80	0.70	3.7	0.93
8	5.20	0.65	7.0	0.88
16	2.80	0.53	13.0	0.81
32	1.53	0.39	23.8	0.74
64	0.88	0.31	41.4	0.65
128	0.55	0.27	66.2	0.52

case, it is useful to stress that the present algorithm offers a very high degree of efficiency. First, FFT allows one to minimize the number of operations. Second, the choice of strip decomposition appears to be a clever way to achieve sub-domain decomposition. In order to estimate the performance of the proposed algorithm, efficiency is defined as $E = S/P$, i.e. which corresponds to the ratio between speedup and number of processors. Table 4 also shows efficiencies of the proposed algorithm for case M1 using up to 128 processors. It is obvious that the highest efficiency is obtained for $P = 1$ in the case of sequential simulation while efficiency decreases with the increase of number of processors considered. Speedup and efficiency were estimated for grid M1 since it is the only one that can be carried out by one single processor. However, one may ask whether or not the degree of implementation varied with the increase of the number of grid points. Unfortunately, previous definitions of speedup and efficiency are not estimated for other grid cases since the computational problem cannot be carried out by a mono-processor. Actually, the cost of the multi-processor running time used for solving the problem includes computational time and inter-communication time between each processor. Following this idea, efficiency E can be defined as the ratio between computational time and total running time. Such a definition was applied to this case while total computational time of direct Poisson solver for a specified grid is independent of the number of processors. Figure 8 presents efficiency (with the last definition) of the proposed algorithm with regard to the increase of number of processors for several test grids. It is obvious that the order of magnitude in efficiency is similar to that obtained for grid M1. For a given test grid, efficiency decreases with the increase of the number

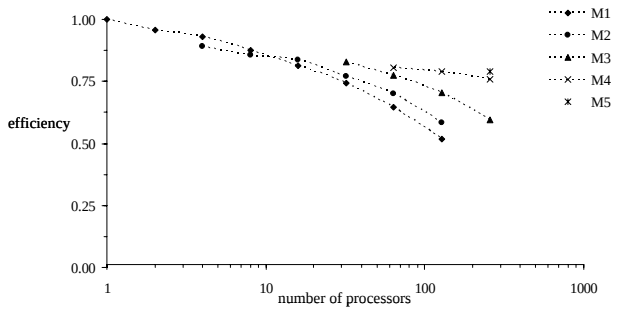


Fig. 8 Efficiencies of strip domain decomposition parallelization variation for fast direct Poisson solver with regard to the number of processors. Efficiencies were obtained from the tests of Poisson's equation for a polynomial problem on a 3D Cartesian grid with Dirichlet boundary conditions in x,y,z direction for several grids.

of processors. However, efficiency increases significantly with regard to the increase of the number of grid points. For example, when using 128 processor, efficiency of the algorithm for M1 grid is about 0.52 whereas for grid M5 this efficiency is about 0.80. Actually, it is important to note that the proposed algorithm is highly efficient, particularly when each sub-domain is significantly loaded in terms of grid points. As clearly observed in the results shown, it is indeed true that a compromise needs to be found between amounts of data to transfer with regard to grid points contained per sub-domain.

V. Conclusion

A parallel algorithm based on strip decomposition for 3D Cartesian staggered grid fast direct Poisson solver was investigated. The algorithm was developed for the purpose of solving the Poisson equation on a multi-processor architecture machine. Two types of boundary conditions were applied for the test cases and several grid resolutions. Several numbers of grid points were performed to estimate errors between the numerical and exact solutions. As expected, these errors drastically decrease in relation to the increase of the grid resolutions.

Moreover, performance of parallelization of the proposed algorithm was particularly well-focused. In our procedure, the number of operations was estimated. Running time decreases exponentially with the increase of the number of processors. In addition, several grids were tested to estimate the efficiency of the algorithm. By using the MPI library on an IBM-P4 multi-processor machine, the Poisson equation was solved with several grid resolution and various processor numbers, the largest involving $P = 256$ processors with 3.6×10^9 grid points. The CPU time was equal to 24 seconds so that efficiency was about 0.80. The proposed algorithm offered a high efficiency for large grid meshes and the communication times were found smaller than 20 percent of total running times. As a result, one may conclude that the efficiency of the proposed method when dealing with Poisson solver is quite significant for large-scale physical problems as well. Finally, based on this proposed parallel algorithm, it is possible to develop such techniques of strip decomposition parallelization for other coordinate grid meshes as well as 3D cylindrical problems.

Acknowledgment

Computations were carried out at the Institut de Développement et des Ressources en Informatique Scientifique (IDRIS), the computational center of the Centre National de la Recherche Scientifique (CNRS). The authors wish to warmly thank the head of IDRIS department, V. Alessandrini, for his constant willingness to provide needed support.

References

[1] R. Löhner, C. Yang, J, J. Cebral, F. Camelli, O. Soto and J. Waltz, "Improving the speed and accuracy of projection-type incompressible flow solvers", *Comput. Methods Appl. Mech. Engrg*, vol. 195, 2006, pp. 3087-3109.

[2] R. W. Hockney, "A fast direct solution of Poisson equation using Fourier analysis", *J. Assoc. Comput. Mach.*, vol. 8, 1965, pp. 95-113.

[3] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Math. Comput.*, vol. 19, 1965, pp. 297-301.
[4] A. McKenney, L. Greengard and A. Mayo, "A fast Poisson solver for complex geometries", *J. Comput. Phys.*, vol. 118, 1995, pp. 348-355.
[5] U. Schumann and R. A. Sweet, "Fast Fourier transforms for direct solution of Poisson's equation with staggered boundary conditions", *J. Comput. Phys.*, vol. 75, 1988, pp. 123-127.
[6] C. Temperton, "On the FACR(l) algorithm for the discrete Poisson equation", *J. Comput. Phys.*, vol. 34, 1980, pp. 314-329.
[7] E. Braverman, M. Israeli and A. Averbuch, "A fast solver for a 3D Helmholtz equation", *SIAM J. Sci. Comput.*, vol. 20(6), 1999, pp. 2237-2260.
[8] G. H. Golub, L. C. Huang, H. Simon and W. P. Tang, "A fast Poisson solver for the finite difference solution of the incompressible Navier-Stokes enquations", *SIAM J. Sci. Comput.*, vol. 19(5), 1998, pp. 1606-1624.
[9] J. C. Adams and P. N. Swarztrauber, "SPHEREPACK 3.0: A model development facility", *Monthly Weather Review*, vol. 127, 1999, pp. 1872-1878.
[10] P. N. Swarztrauber and R. A. Sweet, "Vector and parallel methods for the direct solution of Poisson's equation", *J. Comput. and Appl. Math.*, vol. 27, 1989, pp. 241-163.
[11] P. N. Swarztrauber, "The vector multiprocessor", *Int. J. High Speed Comput.*, vol. 11, 2000, pp. 1-18.
[12] U. Schumann and M. Strietzel, "Parallel solution of Tridiagonal systems for the Poisson equation", *J. Sci. Comput.*, vol. 10, 1995, pp. 181-190.
[13] T. F. Chan and D. C. Resasco, "A domain-decomposed fast Poisson solver on a rectangle", *SIAM J. Sci. Stat. Comput.*, vol. 8, 1987, pp. 27-42.
[14] T. Hoshino, Y. Sato and Y. Asamoto, "Parallel Poisson solver FAGECR-implementation and performance evaluation on PAX computer", *J. Info. Proc.*, vol. 12(1), 1988, pp. 20-26.
[15] S. Ghanemi, "A domain decomposition method for Helmholtz scattering problems", *Ninth. Int. conf. Dom. Demcomp. Meth.*, 1998, pp. 105-112.
[16] J. -Y. Lee and K. Jeong, "A parallel Poisson solver using the fast multipole method on networks of workstations", *Comput. Math. Appl.*, vol. 36, 1998, pp. 47-61.
[17] P. Grandclément, S. Bonazzola, E. Gourgoulhon and J. -A. Marck, "A multidomain spectral method for scalar and vectorial Poisson equations with noncompact sources", *J. Comput. Phys.*, 170, 2001, pp. 231-260.
[18] M. Israeli, L. Vozovoi and A. Averbuch, "Parallelizing implicit algorithm fo time-dependent problems by parabolic domain decomposition", *J. Sci. Comput,*. Vol. 8(2), 1993, pp. 151-166.
[19] W. Briggs, L. Hart, R. A. Sweet and A. O'Gallagher, "Multiprocessor FFT methods", *SIAM J. Sci. Stat. Comput.*, vol. 8, 1987, pp. 27.
[20] P. N. Swarztrauber, "Multiprocessor FFTs", *Parallel Computing*, vol. 5, 1987, pp. 197-210.
[21] P. N. Swarztrauber and R. A. Sweet, "The Fourier and cyclic reduction methods for solving Poisson's equation", In: *Handbook of Fluid Dynamics and Fluid Machinery*, J. A. Schetz and A. E. Fuhs, eds., John Wiley and Sons, New York, NY, 1996.
[22] L. Borges and P. Daripa, "A fast parallel algorithm for the Poisson equation on a disk", *J. Comput. Phys.*, vol. 169, 2001, pp. 151-192.
[23] J. Yang and E. Balaras, "An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries", *J. Comput. Phys.*, vol. 215(1), 2006, pp. 12-40.
[24] E. Balaras, "Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations", *Computers & Fluids*, vol. 33(3), 2004, pp. 375-404.
[25] E. A. Fadlun, R. Verzicco, P. Orlandi and J. Mohd-Yusof, "Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations", *J. Comput. Phys.*, vol. 161(1), 2000, pp. 35-60.
[26] P. Pacheco, "Parallel programming with MPI", *Morgan Kaufmann, San Francisco, CA*, 1997.
[27] G. Sköllermo, "A Fourier method for the numerical solution of Poisson's equation", *Math. Comput.*, vol. 29, 1975, pp. 697.