Chapter 3

Linear solvers

3.1 Features of equation systems

In previous chapter, the algebraic equation systems arising from both Navier-Stokes and mass equations were derived. It is to be stressed that each of these systems, in spite of their coupling, is solved separately with the SIMPLE-like algorithm [32, 35]. The Navier-Stokes system of equations solves the velocities $\vec{V} = \{u, v, w\}$ with the assumption of the known pressure P^* and temperature T fields. Then, the pressure correction system of equations solves P' with the guessed velocity field just evaluated in the previous step. Once the velocities and pressures are corrected, it is the turn of the energy equation, where the temperature T is obtained. Following this iterative procedure it is possible to uncouple the equation systems and focus efforts on each of these systems.

Algorithms [8, 39, 9] which consider directly the coupling between velocity and pressure fields within a single system of equations are not considered in this work, but most of the ideas that will be exposed throughout this chapter can be extended to the coupled systems.

It is well known that Navier-Stokes equations, once discretized and linearized, give a set of algebraic equation systems with non symmetric coefficients in their matrices. Meanwhile the continuity equation leads to a linear system of equations with symmetric coefficient matrix. This fact should be in mind when a solver is used which one suits for symmetric matrices or suits for non symmetric matrices. It is to be mentioned that a symmetric matrix can be considered as a special case of non symmetric matrix from the solver point of view. Thus a solver which treats with non symmetric matrices deals also with the symmetric matrices. Nevertheless, although such kind of solvers are robust and feasible for any linear system of equations, the efficiency in symmetric matrices decreases. For instance, for symmetric matrices the CG solver is faster than the robust BiCGSTAB solver, which performs double number of operations per iteration. Therefore, it is suggested to use a solver for non symmetric matrices and another one for symmetric matrices instead of a general and robust solver which is less efficient for symmetric matrices. Further guidelines on suitable solvers for symmetric and non symmetric matrices may be found in [40].

In addition, any algebraic system includes the set of equations derived from the discretization of the boundary and initial conditions. Typical boundary conditions are non-slip conditions, free-slip conditions, convective conditions and periodic conditions for Navier-Stokes equations (see details in chapter 2). And the pressure gradient is fixed to a value

in the pressure correction equation . The initial conditions give the starting point of all variables at time $\tau = 0$. They all close the problem in order to supply only one solution at each successive time step $\tau + \Delta\tau$ (i.e. the linear system with the boundary conditions and initial conditions included has a non singular matrix).

Furthermore, some of these conditions, say strong conditions, enhance the convergence and offer stability to any solver. While the non-slip and initial conditions shall consider strong sets of equations, the free-slip, fixed gradients, convective and periodic boundary conditions are considered weak sets of equations. For instance, the pressure correction system of equations plus zero gradient in boundaries appear in problems with bounded domains. This system of equations is weakly closed. Since the matrix turns into singular, there are an infinite number of shifted solutions. Luckily, if one point is fixed to a constant value during all the procedure it is enough to get a single solution. Fixing a point means replacing one of the equations of the system in order to obtain a non singular matrix. By doing so, a shifted solution is obtained which matches with the pressure correction system of equations.

Another example is a repeated flow pattern in some directions which is treated with periodicity in these directions. So in these cases all boundary conditions are weak, and it is necessary to fix a value in one point.

A simple example with either double periodicity or free gradient in a two dimensional case shows an infinitive number of shifted solutions if no additional information (i.e. fixed point) is given. The partial differential equation and the boundary conditions are

$$\frac{\partial^2 \phi}{\partial x_i x_i} = b, \; i = \{1, 2\}, \; x_i \in [0, 2\pi]$$

$$b = cos(x_1) sin(x_2)$$

See Fig. 3.1 for different shifted solutions ϕ which match the partial differential equation.

As mentioned before, equation systems with strong or weak sets of boundary conditions have different behaviours when attempts are made to solve them. In addition, these systems are affected by other important factors: the stability and sensitivity of the problem of perturbations. Since neither problem data nor computer arithmetic is exact the solution will not be exact. How small changes in the system of equations affects the final solution is a critical point in the discretization and the design of a robust or a stable solver.

From the point of view of the discretization process, stability is ensured if Scarborough's criterion [32] is satisfied. Or in other words, the coefficients of the matrix A are diagonally dominant. Then, once the discretization is done, the sensitivity is dealt in consideration to the solver. Let $Ax = b$ be a given system of equations, the sensitivity of this system is expressed mathematically as

$$||A + \delta A||_2 ||x + \delta x||_2 = ||b + \delta b||_2$$

$Ax = b$ is a well-conditioned system if for small changes or perturbations in the system (i.e. the matrix $||\delta A||_2$ or the right hand side $||\delta x||_2$) the changes in the solution $||\delta x||_2$ are equal or smaller than the perturbations.

$$||\delta A||_2, ||\delta b||_2 \geq ||\delta x||_2$$

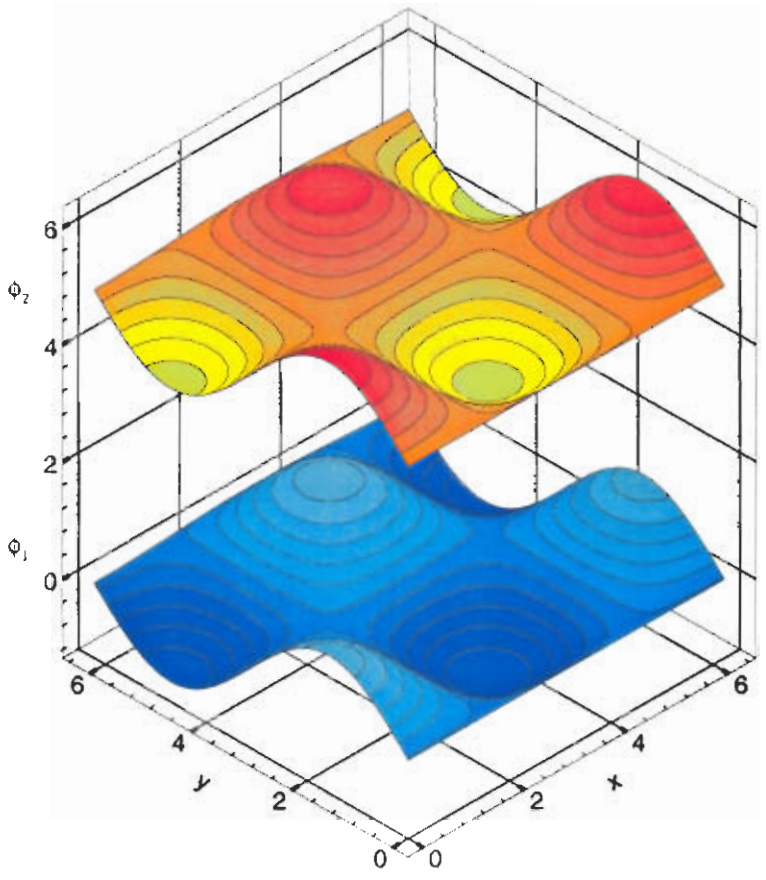

Figure 3.1: Two shifted solutions of the 2D problem with periodic boundary conditions.

Conversely, if small perturbations produce large changes in the solution, then it is said that the system is ill-conditioned.

$$||\delta A||_2, ||\delta b||_2 < ||\delta x||_2$$

It is easy to imagine what it means when solving a linearized system with an iterative procedure. Coefficients of matrix and right hand side term are updated at each iteration in function of values of an approximate solution. Then the system is solved again for a new iteration obtaining a different solution from the previous one and so on. For an ill-conditioned system, the differences between successive solutions increase at each iteration, i.e. the solution diverge.

This fact appears in both Navier-Stokes and pressure correction equation systems. From Navier Stokes system the convective term is linearized at each iteration of the global procedure (see SIMPLE-like algorithm in previous chapter). For high Reynold's numbers, this term dominate over the rest of the terms in the set of equations. Similar situation happens in the pressure correction system of equations for an incompressible fluid. Although the coefficients of the matrix remain constant during the overall algorithm the right hand side

becomes very sensitive to small variations of the divergence of mass.

Since these equation systems are very sensitive to changes in solutions one simple way to prevent the divergence is based on the underrelaxation of solutions.

$$x_{new}^{(k+1)} = \alpha x_{old}^{(k+1)} + (1-\alpha)x^{(k)}$$

Where α is the relaxation parameter, a scalar value within the range $[0, 1]$.

Another more complicate way [41] but effective is based on the measure of how well or ill-conditioned is the matrix problem. That is to the condition number of the matrix $\kappa(A)$. A high value of the condition number points out an ill-conditioned matrix, while a low value indicates a well-conditioned matrix. Therefore, if one changes the problem for another with the same solution, say $\bar{A}x = \bar{b}$, but in addition, it has a reduced condition number

$$\kappa(\bar{A}) < \kappa(A)$$

then this problem would be easier to solve. Details of such technique (the so called preconditioning) are explained later on in this chapter.

3.1.1 Sparse matrix formats

The equation systems derived from CFD problems have been described mathematically. Now is time to have a look, from the computational point of view. How to handle and solve such equation systems is the principal argument of this work.

Since these equation systems contain several thousands (even millions) of unknowns in three dimensional cases with only a few set of unknowns linked per equation, clearly there are large matrices which many of the coefficients are zero. For practical reasons, a matrix is considered sparse if there is an advantage in exploiting the zeros by saving enormous computational storage and time of operations [14] (some operations are avoided where zero values are involved).

Due to the structured grid of points over the domain, they were ordered with a lexicographic or *natural* order. Such organization of unknowns that is *ijk* or any permutation of index like *jki*, *kij*,... gives a matrix with a constant set of diagonals. A number of diagonals depends on the formulation (schemes of discretization) and the dimension of the problem (two or three dimensional). For two dimensional problems, there arises *5-points* and *9-points* formulations while for three dimensional problems there are *7-points* and *19-points* formulations [42]. The distribution of non zero coefficients (within some example of matrices) is shown in Figs. 3.2 and 3.3. Therefore it seems convenient and enough to store the sets of non zero coefficients by diagonals.

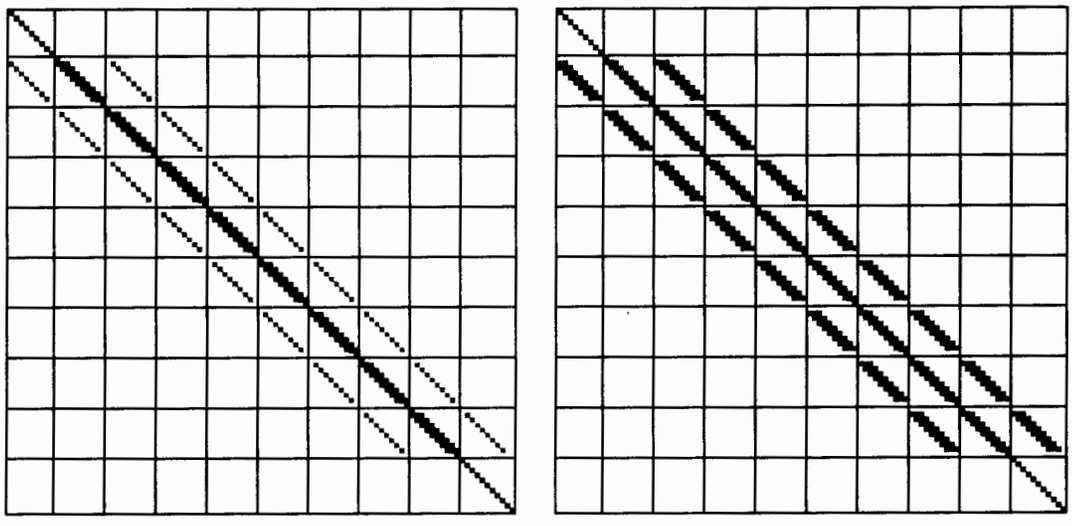

Figure 3.2: 2D-5PF (left) and 2D-9PF (right) for a 10×10 case. Each of these squares has 10×10 entries.

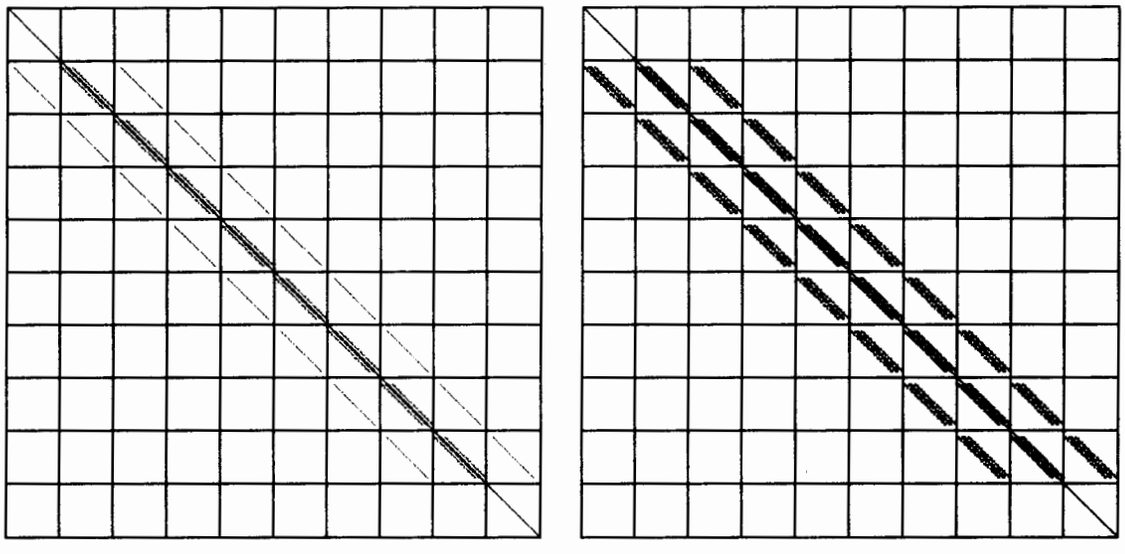

Figure 3.3: 3D-7PF (left) and 3D-19PF (right) for a $10 \times 10 \times 10$ case. Each of these squares has 100×100 entries.

Another option which exploits such kind of sparsity is the band format. It includes all coefficients even zero entries contained between both sides of the main diagonal and fitted in a narrowed band. Although the band width storage format is more general or flexible than the diagonal storage format due to the possibility of modification of some zero entries

by non zero values, it needs to store more values. See Fig. 3.4 for a draft of the band matrix storage format. Furthermore, a system with lexicographic ordering (i.e. the *ijk*

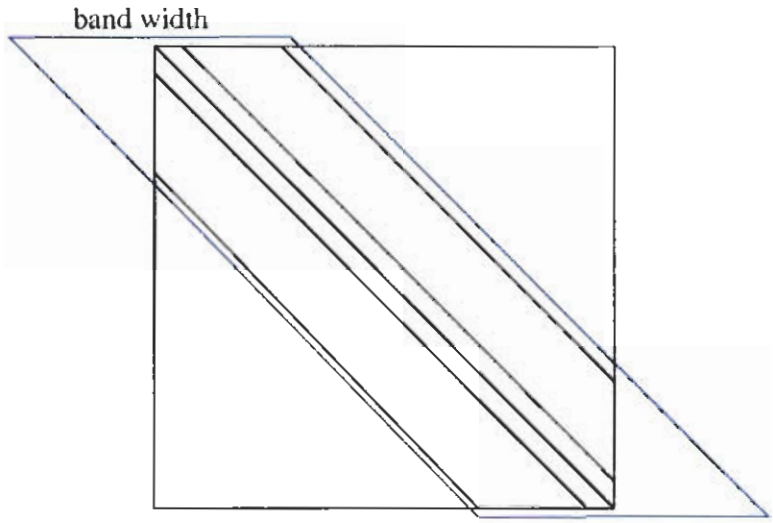

Figure 3.4: The draft of the 2D-5PF case with a band width painted in blue. It evolves the five diagonals.

order) and periodicity in k direction leads to a wider band width than the ordered in *ikj* or *kij*. So in this situation, any of these last orderings is suggested consequently leading to a reduction of the band width. This problem of reordering can be seen in Figs. 3.5, 3.6, 3.7 for a $10 \times 10 \times 10$ case with 7-point formulation.

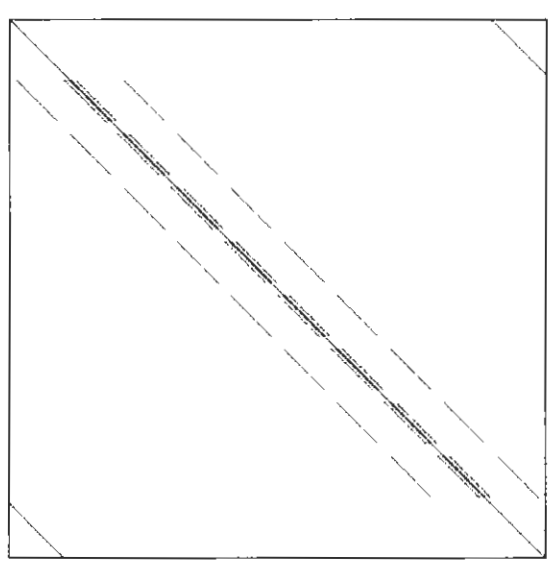

Figure 3.5: The *ijk* ordering for a $10 \times 10 \times 10$ case with *7-point* formulation.

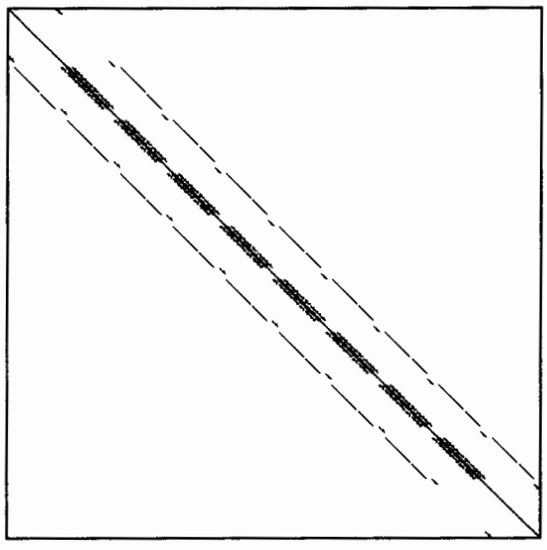

Figure 3.6: The *ikj* ordering for a $10 \times 10 \times 10$ case with *7-point* formulation.

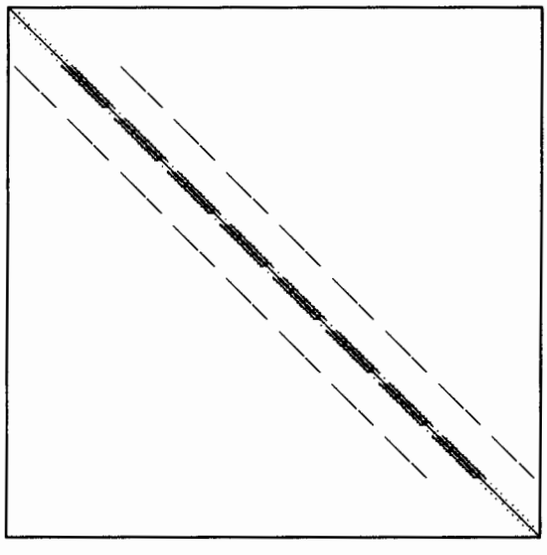

Figure 3.7: The *kij* ordering for a $10 \times 10 \times 10$ case with *7-point* formulation.

For periodicity in k direction the *ijk* ordering has the widest band width. This fact will be considered in the solvers or preconditioners which handle the system in band matrix storage format like complete LU decompositions [14, 43].

3.2 Solving equation systems

Nowadays a full description and even enumeration of all solvers present in the scientific literature would be impossible. However this section is firstly aimed to present a review of the most efficient solvers for CFD equation systems (i.e. the state of the art in CFD solvers). Although the reader interested in general implementations of each solver will find further details in the references, this section provides a compact view of them. And secondly, it will be illustrated not only the advantages and disadvantages of the implementations and efficiencies but also how to link them in order to obtain a more powerful solver by means of the efficiencies linked.

3.2.1 LU solver

Solvers based on factorizations are widely used in narrow banded equation systems [14]. The factorization of a non singular matrix A in lower (L) and upper (U) matrices leads in a direct procedure for the evaluation of the inverse so there by leads to a direct evaluation of the solution for a given right hand side. Once given b and factorized A in L and U, the solution is obtained in a two step process (see Alg. 3): a forward substitution followed by a backward substitution.

Algorithm 3 Complete LU factorization: LU

evaluate *complete LU factorization of* A
$Ax = (LU)x = L(Ux) = Ly = b$

solve $Ly = b$ *by forward substitution*
$y = L^{-1}b$

solve $Ux = y$ *by backward substitution*
$x = U^{-1}y$

Sparse matrices, symmetric or non symmetric, are stored in band matrix formats: A, L and U. As mentioned in cases with periodicity, it is better to reorder the system of equations before factorization in order to reduce the band width and definitely to save in storage requirements and computations. For instance, the well known Cholesky factorization [14] suits for symmetric matrices where only the half part of the band matrix is needed for the factorization and storage. The Crout's factorization [43] suits well for non symmetric matrices but a partial pivoting by rows is recommended in order to reduce roundoff errors produced in the inexact arithmetic of the machine.

Although this solver supplies directly the solution within the machine accuracy, it is only feasible for relatively small size systems. This solver comes to fall into large equation systems due to the requirements of memory and number of floating point operations. However, for the pressure correction system of equations where the matrix remains constant along the iterative algorithm of coupling and only the right hand side changes, it may be feasible to compute and store just once such amount of data.

3.2.2 ILU solver

The incomplete LU factorization, also so called ILU factorization, overcomes the problems of the complete LU factorization. It computes and stores the 'main' values of L and U matrices from the factorization of such large matrix size. Since the ILU gives an approximation to the matrix, it yields in a iterative method.

The several incomplete factorizations [10, 11, 12], are generalized in the concept of fill-in or ILU(fill-in) and are stored in the band matrix format or the set of diagonals. Furthermore, a threshold [26] can also be introduced to achieve a significant reduction of coefficients in cases with a wide band width. These versions lead to different degrees of sparsity, approximation and finally in more or less efficient iterative algorithms. The ILU factorization proposed by Stone [10], and Zedan [8] for non symmetric matrices is presented here (see Alg. 4).

Algorithm 4 Incomplete LU factorization: ILU

evaluate *incomplete LU factorization of* A

$Ax = (M - N)x = (LU - N)x = b$

$Mx^{(k+1)} = Mx^{(k)} - (Ax^{(k)} - b) = Mx^{(k)} + r^{(k)}$, *such that* $||M|| >> ||N||$

$M(x^{(k+1)} - x^{(k)}) = Md^{(k)} = LUd^{(k)} = r^{(k)}$

$LUd^{(k)} = L(Ud^{(k)}) = Ly^{(k)} = r^{(k)}$

set *a guess*

$k = 0$, $x^{(k)}$

$r^{(k)} = b - Ax^{(k)}$

while $(||r^{(k)}||_2 \geq \epsilon)$ ***do***

$r^{(k)} = b - Ax^{(k)}$

solve $Ly^{(k)} = r^{(k)}$ *by forward substitution*

$y^{(k)} = L^{-1}r^{(k)}$

solve $Ud^{(k)} = y^{(k)}$ *by backward substitution*

$d^{(k)} = U^{-1}y^{(k)}$

update *solution*

$x^{(k+1)} = x^{(k)} + d^{(k)}$

end while

The evaluation of coefficients of the ILU factorization shall be described. Matrices L and U are selected, such that the product of these two matrix, say M, gives a good approximation to the matrix A. A first option is the ILU(0) (i.e., the L and U matrices have the same fill-in that the lower and the upper submatrices of A).

In order to fix ideas, an example for the 3D case with a *19-point* formulation is given [11]. Setting an *ijk* ordering of the unknowns, each diagonal of the matrix A and its factorization matrices L and U are represented in Figs. 3.8, 3.9 respectively.

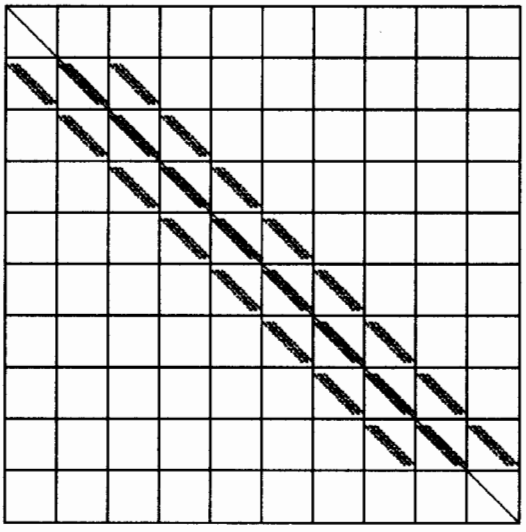

Figure 3.8: Matrix A with *19-point* formulation for a $10 \times 10 \times 10$ size. Each of these squares has 100×100 entries.

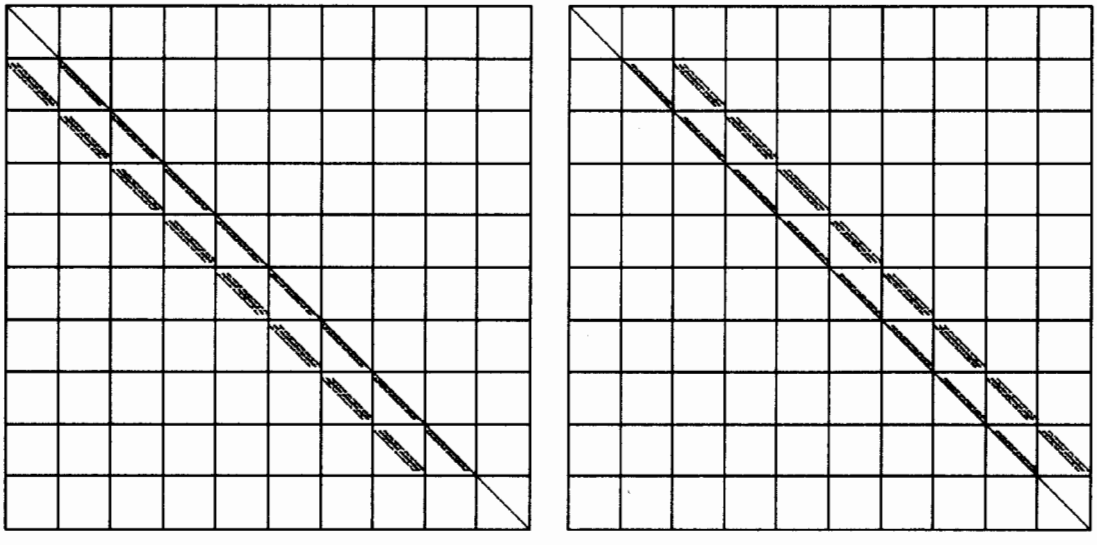

Figure 3.9: (Left) Lower matrix for a $10 \times 10 \times 10$ size.(Right) Upper matrix for a $10 \times 10 \times 10$ size.

After the product of L by U it can be seen that, in addition to the 19 entries in the original matrix A, there are 24 additional coefficients in matrix M. Despite of the additional entries, the equations to be used to determine the coefficients of L and U requires that the 19 coefficients in M remains unchanged from A. In Fig. 3.10, the original entries with points filled in black and half part of the additional entries filled in blue are pictured.

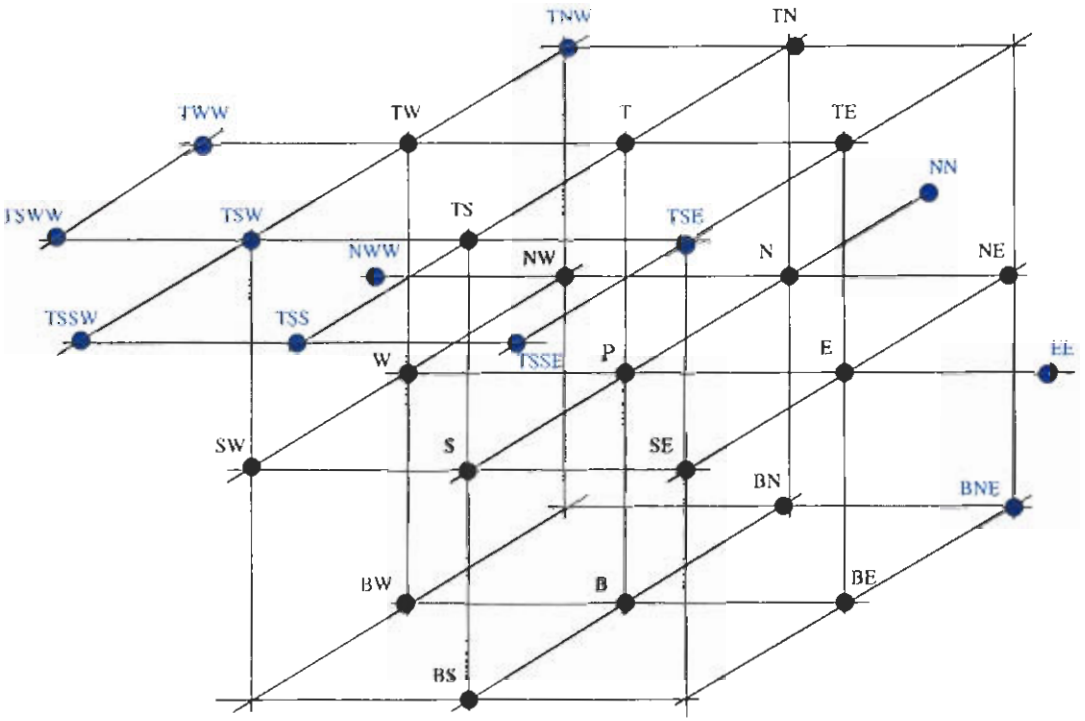

Figure 3.10: Computational molecule with 19 point formulation. Original entries are represented with black filled points. The half part of the additional entries are filled in blue.

The additional 24 entries can be partially canceled through the use of Taylor series expansions and considering points from the 19 entries. For instance,

$$NN = \alpha(-P + 2N)$$

$$TWW = \alpha(-2P + 2W + T)$$

$$BNE = \alpha(-2P + E + N + B)$$

$$TSWW = \alpha(-3P + 2W + S + T)$$

The implementation details can be found in Stone [10] and Zedan [11].

For periodic boundary conditions, it has been seen in previous section that the matrix A contains few additional entries. These entries match well for the additional entries of M, so there is no need to cancel these coefficients. Therefore an ILU for the 3D case is built with or without periodicity just cancelling the additional entries from the original matrix.

The degree of cancellation is controlled by the cancellation parameter α within a range of $[0 : 1)$. Some numerical studies [12, 13] show a good performance for $\alpha = 0.9$ but it is less stable than $\alpha = 0.5$.

Another kind of factorization, SIS [13] (see Alg. 5) is implemented in order to reduce the number of operations, therefore the time of computation per iteration is reduced. Having a look into this algorithm, the computation of the residual vector is unnecessary.

Algorithm 5 Strongly Implicit Solver: SIS

evaluate *incomplete LU factorization of A*
$Ax = (M - N)x = (LU - N)x = b$
$Mx^{(k+1)} = Nx^{(k)} + b$, *with* $||M|| >> ||N||$
$Mx^{(k+1)} = LUx^{(k+1)} = c^{(k)}$
$LUx^{(k)} = L(Ux^{(k+1)}) = Ly^{(k)} = c^{(k)}$

set *a guess*
$k = 0$, $x^{(k)}$
$r^{(k)} = b - Ax^{(k)}$

while $(||r^{(k)}||_2 \geq \epsilon)$ ***do***

evaluate *new right hand side*
$c^{(k)} = Nx^{(k)} + b$
solve $Ly^{(k)} = c^{(k)}$ *by forward substitution*
$y^{(k)} = L^{-1}c^{(k)}$
solve $Ux^{(k+1)} = y^{(k)}$ *by backward substitution*
$x^{(k+1)} = U^{-1}y^{(k)}$

end while

For 2D cases, the number of new entries present in the N matrix is less than the number of original entries present in the A matrix. Therefore the reduction of the number of operations when performing the right hand side term $c^{(k)}$ instead of the residual $r^{(k)}$ is clear. However, there is no advantage for the 3D case. The 24 additional entries are greater than the 19 entries needed for the evaluation of the residual.

If the additional N entries are used the cancellation parameter is avoided so it is more robust than SIP and MSIP. Finally, the periodic boundary conditions are treated implicitly like in SIP or MSIP.

A variable ILU decomposition [26] can be also useful for some matrix whose bandwidth is not well defined. In such cases a threshold parameter is used in order to neglect some coefficients, thus decreasing time of computation and saving storage in memory. It is specially used in high order wavelet matrix transformations (see the section of multiresolution analysis with wavelets). Following this method a robust ILU is defined without regarding the real entries of the matrix.

3.3 Krylov solvers

In this section some Krylov solvers [40, 25] shall be described. Although the Krylov solvers are in theory direct solvers, the inexact arithmetic of that operations leads to iterative solvers. The derivation of Krylov solvers starts from the idea of finding out the solution for x of the linear system $Ax = b$ by means of orthogonal directions of descent in a minimization functional problem $\phi(x)$ like

$$\phi(x) = \frac{1}{2} < x, Ax > - < x, b >$$

Let us assume that $x = (x_1, x_2)$, then the minimal of the function $\phi(x)$ (see Fig. 3.11 for descent directions) which is also the solution to the linear system is found.

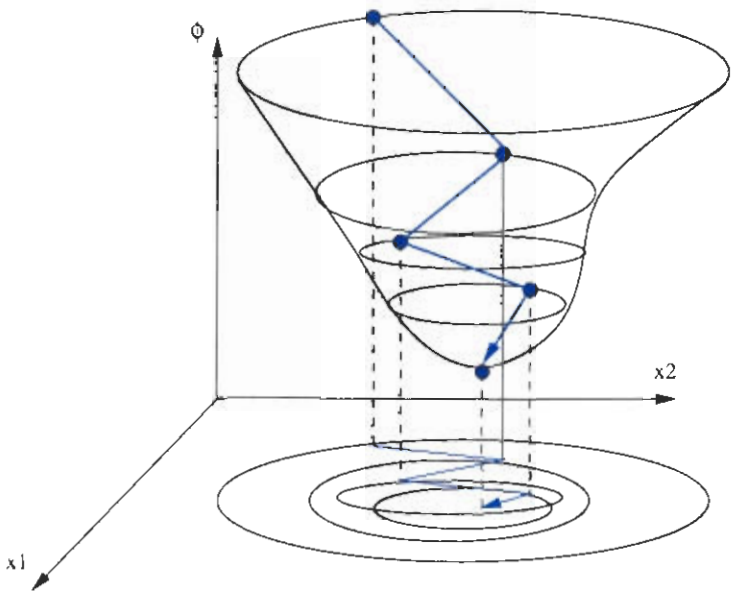

Figure 3.11: Steepest descent directions over the surface of the function ϕ until they arrive to the minimal of the function.

With the initial guess $x^{(0)}$ and the initial residual $r^{(0)}$, the Krylov solver computes in the k-th iteration the optimal correction $p^{(k)}$ over the k-th Krylov subspace associated with A and $r^{(0)}$ $K^k(A, r^{(0)})$

$$K^k(A, r^{(0)}) \equiv span\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, ..., A^{k-1}r^{(0)}\}$$

in the sense that the 2-norm of the residual $r^{(k)} = b - A(x^{(0)} + p^{(k)})$ is minimized. The relation between the minimal residual correction, $p^{(k)}$, and the orthogonality of the new residual $r^{(k)}$ to the shifted Krylov space $AK^k(A, r^{(0)})$

$$AK^k(A, r^{(0)}) \equiv span\{Ar^{(0)}, A^2r^{(0)}, ..., A^{k-1}r^{(0)}\}$$

is given by the following theorem.

Theorem The vector $p^{(k)} \in K^k(A, r^{(0)})$ satisfies

$$p^{(k)} = min||b - A(x^{(0)} + p)||_2, \quad p \in K^k(A, r^{(0)})$$

if and only if

$$r^0 - Ap^k \perp AK^k(A, r^{(0)})$$

□

The Conjugate Gradieut solver (CG), the BiConjugate Gradient STABilized solver (BiCGSTAB), and the Generalized Minimal RESidual solver (GMRES) have been widely used in CFD problems [44]. Features of Krylov solvers and guidelines about their implementations have been summarized.

3.3.1 CG solver

The Conjugate Gradient [40] (see Alg.6) is an effective solver for symmetric positive definite matrices. It is based on the steepest descent solver but it is improved using conjugate directions $p^{(k)}$. Each new direction is orthogonal to the set of previous directions. Moreover it is scaled by the factor α in order to update the solution minimizing the residual norm.

Algorithm 6 Conjugate Gradient: CG

set a guess
$k = 0$, $x^{(k)}$
$r^{(k)} = b - Ax^{(k)}$

while $(||r^{(k)}||_2 \geq \epsilon)$ ***do***
$\rho^{(k)} = < r^{(k)}, r^{(k)} >$

evaluate *new direction and scaled factor*
if $(k = 0)$
$p^{(k+1)} = r^{(k)}$
else
$\beta = \dfrac{\rho^{(k)}}{\rho^{(k-1)}}$
$p^{(k+1)} = r^{(k)} + \beta p^{(k)}$
end if

evaluate *new direction and scaled factor*
$q^{(k+1)} = Ap^{(k+1)}$
$\alpha = \dfrac{\rho^{(k)}}{< p^{(k+1)}, q^{(k+1)} >}$

update *solution and residual*
$x^{(k+1)} = x^{(k)} + \alpha p^{(k+1)}$
$r^{(k+1)} = r^{(k)} - \alpha q^{(k+1)}$

$k = k + 1$

end while

In absence of roundoff errors the series of conjugate directions and series of residuals are so called A-orthogonal. Then the following theorem is written.

Theorem. Let A be a symmetric positive definite matrix and $p^{(0)}, ..., p^{(N-1)}$ are A-orthogonal, for any $x^{(0)} \in \mathbb{R}^N$ given the algorithm converges to the exact solution in less or equal than N iterations. □

This theorem guarantees not only the convergence of the algorithm but also in exact arithmetic, it would be a direct solver with N steps.

3.3.2 BiCGSTAB solver

The conjugate gradient method is not suitable for non symmetric matrices because the series of residual vectors loose its orthogonality. Thus a bi orthogonal process [45] is done in order

to improve the conjugate directions. For BiCGSTAB solver (see Alg. 7), the conjugate directions and the residuals are updated in two steps. In this case two parameters α and β are needed to scale the conjugate directions and residuals.

Algorithm 7 Bi Conjugate Gradient STABilized: BiCGSTAB

set *a guess*
$k = 0$, $x^{(k)}$
$r^{(k)} = b - Ax^{(k)}$

while $(||r^{(k)}||_2 \geq \epsilon)$ ***do***
$\rho^{(k)} =< r^{(0)}, r^{(k)} >$
evaluate *new direction and scaled factor*
if $(k = 0)$
$p^{(k+1)} = r^{(k)}$
else
$\beta = \frac{\rho^{(k)}}{\rho^{(k-1)}} \frac{\alpha}{\omega}$
$p^{(k+1)} = r^{(k)} + \beta(p^{(k)} - \omega q^{(k)})$
end if
evaluate *new direction and scaled factor*
$q^{(k+1)} = Ap^{(k+1)}$
$\alpha = \frac{\rho^{(k)}}{< r^{(0)}, q^{(k+1)} >}$
$s^{(k+1)} = r^{(k)} - \alpha q^{(k)}$
check *direction*
if $(||s^{(k+1)}||_2 < \epsilon)$
update *solution*
$x^{(k+1)} = x^{(k)} + \alpha p^{(k+1)}$
break while
end if
evaluate *new direction and scaled factor*
$t^{(k+1)} = As^{(k+1)}$
$\omega = \frac{< t^{(k+1)}, s^{(k+1)} >}{< t^{(k+1)}, t^{(k+1)} >}$
update *solution and residual*
$x^{(k+1)} = x^{(k)} + \alpha p^{(k+1)} + \omega s^{(k+1)}$
$r^{(k+1)} = s^{(k)} - \omega t^{(k+1)}$
$k = k + 1$
end while

BiCGSTAB solver can also be used in symmetric positive definite matrices instead of CG solver. However, the convergence rate is equal to CG but at twice the cost per iteration.

3.3.3 GMRESR solver

Like BiCGSTAB solver, the Generalized Minimal RESidual solver [46] and the restarted version GMRESR [47] deals with the non symmetric matrices. The orthogonalization process is based on a modified version of the Gram-Schmidt orthogonalization [43]. Each direction is found in order to minimize the following problem.

$$p^{(k)} = min||b - A(x^{(0)+p})||_2 = min||r^{(0)} - Ap||_2,\ p \in K^k(A, r^{(0)})$$

Let us consider V_k the orthogonal basis for the Krylov's subspace $K^k(A, r^{(0)})$ and H_k the upper $k \times k$ Hessenberg matrix generated from the orthogonalization process. The direction $p^{(k)}$ can be computed as

$$p^{(k)} = V_k y^{(k)},\ y^k = H_k^{-1}||r^{(0)}||_2 e_1$$

Where e_1 is the unit vector $e_1 = (1, 0, ..., 0)^T$. Substituting the last expression in the minimization problem the below is found

$$x^{(k)} = x^0 + V_k y^{(k)} \leftarrow min||\ ||r^{(0)}||_2 e_1 - H_k y^{(k)}\ ||_2$$

The solution of the minimization problem is done by a QR factorization with matrix transformations: the Given's rotations.

$$Q_k H_k = R_k$$

Introducing the QR factorization in the 2-norm of the minimization problem

$$Q_k(||r^{(0)}||_2 e_1 - H_k y^{(k)}) = Q_k||r^{(0)}||_2 e_1 - R_k y^{(k)}$$

The minimization problem is reduced to solve the following triangular matrix system

$$R_k y^{(k)} = Q_k||r^{(0)}||_2 e_1 = g_k$$

Once evaluated $y^{(k)}$ the found direction $p^{(k)}$ can be updated easily.

Like CG and BiCGSTAB, GMRES converges in no more than N steps so it could be considered a direct solver. However, the storage of search directions is limited by the available RAM of the computer. Therefore, a restarted version must be used leading to an iterative solver, i.e. the named GMRESR(m).

Choosing m of the searched directions, the solution must be updated and directions cleared. This procedure is repeated until convergence is achieved. If m is too small, GMRESR(m) may be slow to converge or even may be stagnated. A large than necessary value of m will have good convergence rate per iteration but at highly computation cost (time and storage). Although the range of the restart parameter is fixed between 5 and 50, it is said to be problem dependent.

Moreover, the inexact arithmetic produces loses of orthogonality in the search of new directions so a orthogonal check and a reorthogonalization process [48] is added in order to improve from the robustness point of view the algorithm (see details in Alg. 8).

Algorithm 8 Generalized Minimal RESidual Restarted: GMRESR(m)

set a guess
$k = 0,\ x^{(k)}$
$r^{(k)} = b - Ax^{(k)}$
$\beta = ||r^{(k)}||_2,\ q_1 = \dfrac{r^{(k)}}{\beta}$
$g = g(1:k+1) = \beta(1, 0, \ldots, 0)^T$
while $(||r^{(k)}||_2 \geq \epsilon)$ **do**
orthogonalization by modified Gramm Schmidt
$k = k + 1$
$v_k = v,\ v = Aq_k$
evaluate Hessenberg matrix
for $(i = 1$ *to* $k)$
$H_{i,k} =< q_i, v_k >$
$v_k = v_k - H_{i,k}q_i$
end for
$H_{k+1,k} = ||v_k||_2$
check orthogonality, $\delta = 10^{-3}$
if $(||v||_2 + \delta||v_k||_2 = ||v||_2)$
for $(i = 1$ *to* $k)$
$\mu =< q_i, v_k >$
$H_{i,k} = H_{i,k} - \mu$
$v_k = v_k - \mu q_i$
end for
end if
$q_{k+1} = \dfrac{v_k}{H_{k+1,k}}$
evaluate QR factorization by Given's rotations
if $(k > 1)$
$H_{*,k} = Q_{k-1}H_{*,k}$
end if
$\nu = \sqrt{H_{k,k}^2 + H_{k+1,k}^2}$
$c_k = \dfrac{H_{k,k}}{\nu},\ s_k = \dfrac{H_{k+1,k}}{\nu}$
$H_{k,k} = c_k H_{k,k} - s_k H_{k+1,k},\ H_{k+1,k} = 0$
$g = G_k g,\ ||r^{(k)}||_2 = |g_{k+1}|$
restart at m-th vector v
if $(k = m)$
solve $Ry = w$ *where* $R =$*upper triangular* $(H)_{k\times k},\ w = g(1:k)$
$y^{(k)} = R^{-1}w$
update solution
$x^{(k)} = Qy^{(k)}$
set $k = 0$
end if
end while

3.4 Preconditioners

In the section 3.1 related with well and ill-conditioned systems it was mentioned that the convergence rate of the iterative methods depends on the condition number $\kappa(A)$, or in other words, on the spectral properties of the coefficient matrix A. Hence the linear system is transformed into one that has the same solution but has better condition number.

Let us call M the non singular preconditioning matrix operating over the linear system $Ax = b$. The matrix A is approximate to the matrix M in such a way that

$$M^{-1}Ax = M^{-1}b$$

is easiest to solve than the original system, thus $\kappa(M^{-1}A) < \kappa(A)$, in spite of the additional computational cost and storage. In this case, the preconditioner is applied to the right of both sides of the linear system. There are two equivalent expressions, the so called left preconditioner and the symmetric preconditioner.

$$AM^{-1}Mx = b$$

$$M^{-\frac{1}{2}}AM^{-\frac{1}{2}}M^{\frac{1}{2}}x = M^{-\frac{1}{2}}b$$

Preconditioners can be applied explicitly, implicitly and both. However, an explicit left preconditioner or a symmetric preconditioner can be implemented in a preprocess and post-process steps. The implementation of each preconditioner into a general solver is outlined in Algs. 9, 10.

Algorithm 9 Left preconditioner

precondition *the problem*
$Ax = b$, M
$\bar{A} = M^{-1}A$, $\bar{b} = M^{-1}b$

solve *the preconditioned problem*
solve $\bar{A}x = \bar{b}$

Algorithm 10 Symmetric preconditioner

precondition *the problem*
$Ax = b$, $M^{\frac{1}{2}}, M^{-\frac{1}{2}}$
$\bar{A} = M^{-\frac{1}{2}}AM^{\frac{1}{2}}$, $\bar{x} = M^{-\frac{1}{2}}x$, $\bar{b} = M^{-\frac{1}{2}}b$

solve *the preconditioned problem*
solve $\bar{A}\bar{x} = \bar{b}$
solve $M^{-\frac{1}{2}}x = \bar{x}$

The left preconditioner can be considered as a scaling preprocess. Usually, the main diagonal of matrix A, say D_A serves as M preconditioner. Therefore, the storage is very reduced and the inverse M^{-1} can be computed directly by the inverse of all coefficients of D_A.

$$\bar{A} = M^{-1}A = D_A^{-1}A$$

The right preconditioner is usually implemented implicitly in the algorithm. In the scientific literature we found many kinds of preconditioners for Krylov's solvers for sparse matrix based on scaling [40], incomplete LU factorizations [49], polynomial preconditioners [27] and sparse approximate inverses [29] (SPAI).

A diagonal preconditioner was explained also as an explicit left preconditioner. It is also called point jacobi preconditioner, and it is usually considered thought as a scaling technique instead of a preconditioner. Due to the low efficiency, more powerful preconditioners were looked for such as those based on ILU factorizations or on SPAI.

The preconditioned Krylov's solvers can be implemented by adding an intermediate step into the search direction procedure. For instance, BiCGSTAB and GMRESR algorithms are rewritten (see Algs. 11, 12 respectively) including this intermediate step where it is needed.

3.4.1 Factorizations and SPAI

Solvers based on ILU factorizations can be used directly as implicit preconditioners in Krylov's solvers. Our work has been focused on the incomplete factorization due to easy implementation in fixed sparse matrix patterns, low computational cost and low storage requirements. For instance, if our system has a symmetric coefficient matrix, the CG solver has to be used with a symmetric incomplete factorization preconditioner like the Incomplete Cholesky factorization thus leading in the well known ICCG solver [50]. For non symmetric coefficient matrix, the ILU can be implemented into the BiCGSTAB or into the GMRESR.

On the other hand, the Sparse Approximate Inverse preconditioner is given as an alternative for the common preconditioners pointed above. It is known that a good preconditioner M must have similar spectral properties to A. Moreover the inverse of the preconditioner M^{-1} must also have similar properties to A^{-1}. Therefore we try to directly find the inverse of the preconditioner by an approximation to the inverse of the matrix A. The inverse is approximated by rows or columns in the dependence whether it is used as a left or a right preconditioner.

One possible implementations is given by Grote [28]. The main idea of SPAI consists of guessing the entries of the inverse matrix and finding by minimization in the Frobenius norm each row or column. The experience in such kind of matrix give us a guideline to define quickly this shape in band. However the results provided by different authors [51] show a problem dependent convergence behaviour.

In the following sections two acceleration techniques for any solver are provided. The first one is based on the algebraic multigrid AMG [16, 18] and the second one on the multiresolution analysis MRA [20] with wavelets [23]. Both accelerators have similar properties and convergence rates.

Algorithm 11 Preconditioned BiCGSTAB

set *a guess*
$k = 0$, $x^{(k)}$
$r^{(k)} = b - Ax^{(k)}$

while $(||r^{(k)}||_2 \geq \epsilon)$ ***do***
$\rho^{(k)} =< r^{(0)}, r^{(k)} >$
evaluate *new direction and scaled factor*
if $(k = 0)$
$p^{(k+1)} = r^{(k)}$
else
$\beta = \frac{\rho^{(k)}}{\rho^{(k-1)}} \frac{\alpha}{\omega}$
$p^{(k+1)} = r^{(k)} + \beta(p^{(k)} - \omega q^{(k)})$
end if
evaluate *new direction and scaled factor*
solve *the preconditioned problem*
$M\bar{p} = p^{(k+1)}$
$q^{(k+1)} = A\bar{p}$
$\alpha = \frac{\rho^{(k)}}{< r^{(0)}, q^{(k+1)} >}$
$s^{(k+1)} = r^{(k)} - \alpha q^{(k)}$
check *direction*
if$(||s^{(k+1)}||_2 < \epsilon)$
update *solution*
$x^{(k+1)} = x^{(k)} + \alpha p^{(k+1)}$
break while
end if
evaluate *new direction and scaled factor*
solve *the preconditioned problem*
$M\bar{s} = s^{(k+1)}$
$t^{(k+1)} = A\bar{s}$
$\omega = \frac{< t^{(k+1)}, s^{(k+1)} >}{< t^{(k+1)}, t^{(k+1)} >}$
update *solution and residual*
$x^{(k+1)} = x^{(k)} + \alpha p^{(k+1)} + \omega s^{(k+1)}$
$r^{(k+1)} = s^{(k)} - \omega t^{(k+1)}$
$k = k + 1$
end while

Algorithm 12 Preconditioned GMRESR(m)

set a guess
$k = 0$, $x^{(k)}$
solve the preconditioned problem
$Mr^{(k)} = b - Ax^{(k)}$
$\beta = ||r^{(k)}||_2$, $q_1 = \dfrac{r^{(k)}}{\beta}$
$g = g(1:k+1) = \beta(1, 0, \ldots, 0)^T$
while $(||r^{(k)}||_2 \geq \epsilon)$ **do**
orthogonalization by modified Gramm Schmidt
$k = k + 1$
solve the preconditioned problem
$Mv = Aq_k$
$v_k = v$
evaluate Hessenberg matrix
for $(i = 1$ *to* $k)$
$H_{i,k} =< q_i, v_k >$
$v_k = v_k - H_{i,k}q_i$
end for
$H_{k+1,k} = ||v_k||_2$
check orthogonality, $\delta = 10^{-3}$
if $(||v||_2 + \delta||v_k||_2 = ||v||_2)$
for $(i = 1$ *to* $k)$
$\mu =< q_i, v_k >$
$H_{i,k} = H_{i,k} - \mu$
$v_k = v_k - \mu q_i$
end for
end if
$q_{k+1} = \dfrac{v_k}{H_{k+1,k}}$
evaluate QR factorization by Given's rotations
if $(k > 1)$
$H_{*,k} = Q_{k-1}H_{*,k}$
end if
$\nu = \sqrt{H_{k,k}^2 + H_{k+1,k}^2}$
$c_k = \dfrac{H_{k,k}}{\nu}$, $s_k = \dfrac{H_{k+1,k}}{\nu}$
$H_{k,k} = c_k H_{k,k} - s_k H_{k+1,k}$, $H_{k+1,k} = 0$
$g = G_k g$, $||r^{(k)}||_2 = |g_{k+1}|$
restart at m-th vector v
if $(k = m)$
solve $Ry = w$ *where* $R =$*upper triangular* $(H)_{k\times k}$, $w = g(1:k)$
$y^{(k)} = R^{-1}w$
update *solution*
$x^{(k)} = Qy^{(k)}$
set $k = 0$
end if
end while

3.5 Algebraic Multigrid algorithm

There are numerous publications [15], tutorials,... about algebraic multigrid (AMG) and here one refers mainly to Huttchinson [52] and Wesseling [16]. AMG is based on a hierarchy of linear systems $A_l x_l = b_l$ constructed directly from the original linear system $Ax = b$ at fine grid $l = 1$ to a maximal level $l = l_{max}$ by means of transfer operators. The first system $A_1 x_1 = b_1$ is solved roughly and the error $e_1 = x - x_1$ is erased by adding succesive corrections x_l from the different levels $l = 2, 3.., l_{max}$. Solving each system, different frequency range of corrections are dealt with.

For instance, a two level multigrid scheme begins solving the first system $A_1 x_1 = b_1$ within a certain number ν_1 of iterations. The measure of the difference from the numerical solution x_1 to the exact solution x in this first level $l = 1$ is the error $e_1 = x - x_1$.

If the frequency components of this error are analyzed at using the discrete Fourier transform (DFT), the solver smoothes well the high frequency components and keeps the lowest frequency components nearly untouched. For this reason it is said that the solver acts as smoother of low frequency components of the error. This iterative process is called the pre-smoothing step. An additional system may be found in order to compute this error numerically and then correct the numerical solution.

$$A_1 e_1 = A_1(x - x_1) = b_1 - A_1 x_1 = r_1$$

Unfortunately, the last linear system is computationally expensive as the first system. Furthermore, this error solution contains low frequencies which cannot be solved efficiently by the iterative solver at fine grid. Therefore a transformation of this system into another one is carried out in order to convert the low frequency components into higher frequency components. This transformation is done with the so called restriction R_1^2 and prolongation P_2^1 operators [16].

$$A_2 = R_1^2 A_1 P_2^1$$

$$b_2 = R_1^2 r_1$$

$$A_2 x_2 = b_2$$

Having a look at such transformation a reduced system of equations is obtained, which seems to be derived from a coarsest problem. Hence, the transformation receives the name of coarse grid approximation [16] (CGA) of the original problem. Then, a post-smoothing step is performed within ν_2 iterations in order to obtain an approximation to the error e_1 but in the coarse level $l = 2$, let us say x_2.

This approximation to the error is also called the correction x_2 and it must be transferred and added to the solution x_1 at fine grid. The prolongation operator transfers the vector from the coarsest level $l = 2$ to the finest level $l = 1$.

$$x_1 = x_1 + P_2^1 x_2$$

Finally, this algorithm is repeated until a desired error threshold ϵ is achieved or a maximum number of iterations it_{max} is done. The algebraic multigrid (see Alg. 13), the prediction (see Alg. 14) and the correction (see Alg. 15) algorithms are written as follows:

Algorithm 13 Algebraic Multi Grid: AMG

Fix parameters of multigrid cycle
$l_{max}, it_{max}, \nu_1, \nu_2, \epsilon$

for$(l = 1$ *to* $l_{max} - 1)$
evaluate matrix coeff. by Coarse Grid Approximation
$A_{l+1} = R_l^{l+1} A_l P_{l+1}^l$
end do

set a guess in finest level
$l = 1,\ k = 0,\ x_l^{(k)}$

while $(||r||_2 \geq \epsilon$ or $k \leq it_{max})$
smooth with ν_1 *iterations*
solve $A_l x_l = b_l$

if$(l < l_{max})$
go to the coarser level $l + 1$
predict x_{l+1}
add the correction to the level l
correct x_l

end if

$k = k + 1$
end while

Algorithm 14 Prediction

evaluate right hand side of coarser level $l + 1$
$r_l^{(\nu_1)} = b_l - A_l x_l^{(\nu_1)}$
$b_{l+1} = R_l^{l+1} r_l^{(\nu_1)}$

set a guess at coarser level
$l = l + 1,\ k = 0, x_l^{(k)} = 0$
smooth with ν_1 *iterations*
solve $A_l x_l = b_l$

if$(l < l_{max})$
go to the coarser level $l + 1$
predict x_{l+1}
add the correction to the level l
correct x_l

end if

Algorithm 15 Correction

add *the correction to the level* l

$x_l^{(\nu_1)} = x_l^{(\nu_1)} + P_{l+1}^l x_{l+1}^{(\nu_1)}$

smooth *with* ν_2 *iterations*

solve $A_l x_l = b_l$

3.5.1 Transfer operators

These operators transfer information from one level to the closest level. The restriction operator does it from the fine level l to the coarse level $l+1$ while the prediction transfer operator does it in an opposite way, (i.e., from the coarse level to the fine level).

In order to give an easy explanation to such operators, a suitable set of two dimensional piece wise constant [16] restriction and prolongation operators has been chosen and applied to a two dimensional problem with *5-point* formulation.

Let G_l and G_{l+1} be a fine and coarse grid respectively defined over the domain Ω (see Fig. 3.12).

$$l = \{1, 2, \ldots, l_{max}\}$$

$$G_l = \{(x_1, x_2) \in \Omega,\ \Omega : [0, L_1] \times [0, L_2]\}$$

$$x_i(j) = \{0, \ldots, (j-1)h_i, \ldots, L_i\},\ j = 1, \ldots, \frac{n_i}{2^{l-1}},\ h_i = \frac{L_i}{\frac{n_i}{2^{l-1}} - 1},\ i = \{1, 2\}$$

Let (x_1, x_2) be a point in G_{l+1} associated with a $(i, j)_{l+1}$ index point. Then, there is a set of index points in G_l which are placed at same position that G_{l+1}

$$G_{l+1} = \{(2i, 2j)_l, (2i+1, 2j)_l, (2i, 2j+1)_l, (2i+1, 2j+1)_l$$

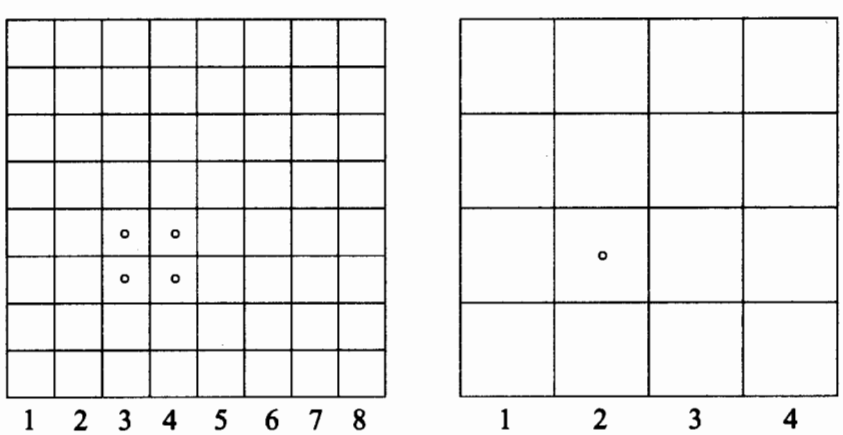

Figure 3.12: Example of two related grids: $G_1 : 8 \times 8$ and $G_2 : 4 \times 4$. G_2 has double size-mesh of G_1.

For a better comprehension, the stencil notation gives a simply representation in two dimensions of these operators.

$$x_l : G_l \to \mathbf{R},\ x_{l+1} : G_{l+1} \to \mathbf{R}$$

$$R_l^{l+1} : G_l \to G_{l+1}$$

$$R_{l+1}^{l+1} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$x_{l+1} = R_l^{l+1} x_l$$

$$x(i,j)_{l+1} = x(2i,2j)_l + x(2i+1,2j)_l + x(2i,2j+1)_l + x(2i+1,2j+1)_l$$

In this example (see Fig. 3.13), the restriction operator R performs a summation over blocks of four grid points weighted by a unit factor.

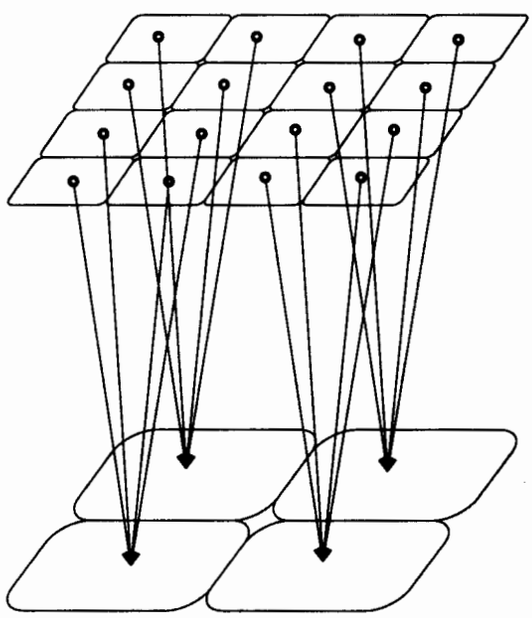

Figure 3.13: Restriction process over blocks of four grid points.

If natural ordering is considered by applying the restriction operator to the example with fine level $l = 1$, and coarse level $l = 2$ leads to the matrix form

$$x_1 = (x(1,1)_1, x(2,1)_1, \ldots, x(7,8)_1, x(8,8)_1)^T_{64\times 1}$$

$$x_2 = (x(1,1)_2, x(2,1)_2, \ldots, x(3,4)_2, x(4,4)_2)^T_{16\times 1}$$

$$x_2 = R_1^2 x_1 = R_{1,y}^2 R_{1,x}^2 x_1$$

$$R^2_{1,x} = \begin{bmatrix} \begin{bmatrix} 11 & & \\ & \cdots & \\ & & 11 \end{bmatrix}_{4\times 8} & & \\ & \ddots & \\ & & \begin{bmatrix} 11 & & \\ & \cdots & \\ & & 11 \end{bmatrix}_{4\times 8} \end{bmatrix}_{32\times 64}$$

$$R^2_{1,y} = \begin{bmatrix} \begin{bmatrix} 1 & & 1 \\ & \cdots & \\ 1 & & 1 \end{bmatrix}_{4\times 8} & & \\ & \ddots & \\ & & \begin{bmatrix} 1 & & 1 \\ & \cdots & \\ 1 & & 1 \end{bmatrix}_{4\times 8} \end{bmatrix}_{16\times 32}$$

In general, weight factors in restriction operator must satisfy certain rule [16]:

$$\sum_{i,j} R_{i,j} = \left(\frac{h_{l+1}}{h_l}\right)^{\alpha}$$

Where α varies from inside to boundary points and depends of boundary conditions implemented. Here let us assume $\alpha = 2$ in the whole domain. Then $\sum_{i,j} R_{i,j} = 4$ everywhere.

Prolongation operator P (see Fig. 3.14) is expressed in matrix form as the transpose of the restriction operator R. $P^l_{l+1} = R^{l+1,T}_l$.

$$P^l_{l+1} : G_{l+1} \rightarrow G_l$$

$$P^l_{l+1} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$x_l = P^l_{l+1} x_{l+1}$$

$$x(2i, 2j)_l = x(2i+1, 2j)_l = x(2i, 2j+1)_l = x(2i+1, 2j+1)_l = x(i, j)_{l+1}$$

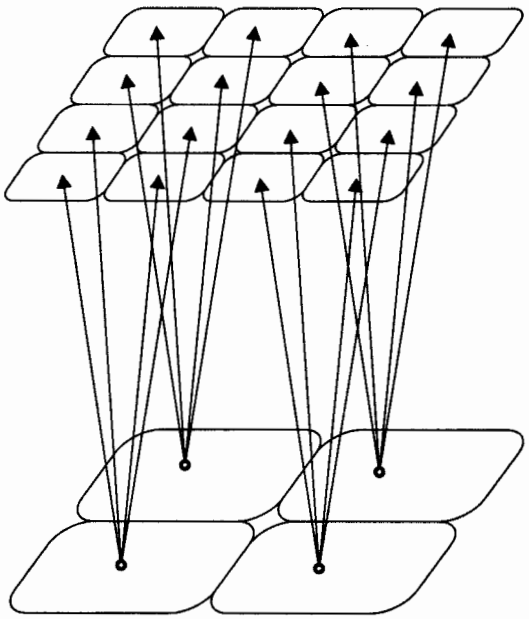

Figure 3.14: Prolongation process to blocks of four grid points.

Furthermore the product of both operators leads into a scaled identity matrix.

$$P_{l+1}^{l} R_{l}^{l+1} = \alpha I$$

In this example with piece wise constant operators, $\alpha = 4$. However, it is possible to find a pair of transfer operators were the product is equal to the identity. In such case, there is an important feature between transfer operators

$$P_{l+1}^{l} = R_{l}^{l+1,T} = R_{l}^{l+1,-1}$$

This feature points out the ability to use a single matrix and its transpose like a prolongation or a restriction operator. This idea will be recovered and exploited in multiresolution analysis by wavelets [21].

Finally, the coarse grid approximation is evaluated by

$$A_{l+1} = R_{l,y}^{l+1} R_{l,x}^{l+1} A_l P_{l+1,y}^{l} P_{l+1,x}^{l}$$

Seeing the example, each 2×2 set of equations on fine grid G_l is transformed in to one equation on coarse grid G_{l+1} with the same stencil.

$$A_l = \begin{bmatrix} & A_l^n & \\ A_l^w & A_l^p & A_l^e \\ & A_l^s & \end{bmatrix}$$

$$A_{l+1} = \begin{bmatrix} & A_{l+1}^n & \\ A_{l+1}^w & A_{l+1}^p & A_{l+1}^e \\ & A_{l+1}^s & \end{bmatrix}$$

Where

$$A^s_{l+1}(i,j) = A^s_l(2i,2j) + A^s_l(2i+1,2j)$$

$$A^w_{l+1}(i,j) = A^w_l(2i,2j) + A^w_l(2i,2j+1)$$

$$A^e_{l+1}(i,j) = A^e_l(2i+1,2j) + A^e_l(2i+1,2j+1)$$

$$A^n_{l+1}(i,j) = A^n_l(2i,2j+1) + A^n_l(2i+1,2j+1)$$

$$\begin{aligned} A^p_{l+1}(i,j) = {} & A^p_l(2i,2j) + A^p_l(2i+1,2j) + \\ & A^p_l(2i,2j+1) + A^p_l(2i+1,2j+1) + \\ & A^s_l(2i,2j+1) + A^s_l(2i+1,2j+1) + \\ & A^w_l(2i+1,2j) + A^w_l(2i+1,2j+1) + \\ & A^e_l(2i,2j) + A^e_l(2i,2j+1) + \\ & A^n_l(2i,2j) + A^n_l(2i+1,2j) \end{aligned}$$

Similar expressions can be obtained for the *9-point* formulation, the *7-point* formulation and the *19-point* formulation. Instead of a transfer operator acting over a 2×2 set of grid points for a *5-point* formulation, the extension can be derived at the *9-point* formulation with 3×3 set of grid points. In three dimensional cases, for the *7-point* formulation there is $2 \times 2 \times 2$ set of grid points and for the *19-point* formulation $3 \times 3 \times 3$ set of grid points.

However, for three dimensional cases the stencil notation does not helps to the representation of these operators and only an algebraic expression can be given as

$$A_{l+1} = R^{l+1}_{l,x} R^{l+1}_{l,y} R^{l+1}_{l,x} A_l P^l_{l+1,z} P^l_{l+1,y} P^l_{l+1,x}$$

A more detailed description of transfer operators is exposed in Zeeuw [17].

3.6 Multiresolution Analysis with wavelets

The basic idea behind wavelet multiresolution analysis [20] (MRA) is to represent a function in terms of basis of functions called wavelets [23] having discrete scales and locations. In other words, wavelet analysis can be viewed as a multilevel or multiresolution representation of a function, where each level of resolution consists of basis of functions having the same scale but located at different positions.

Therefore, a function (continuous or sampled), a vector or a matrix can be split by wavelet analysis into low and high frequency parts. Both resulting parts have about half the dimension of the original one. Again the part containing the low frequency components can be decomposed in the same way. This procedure predestines the wavelets to serve as a restriction and prolongation operators [22] in the usual multigrid method.

3.6.1 Multilevel representation of a function

In order to develop a multilevel representation of a function $f(x)$ in $L^2(\mathbb{R})$ a sequence of embedded subspaces V_i is looked for

$$\{0\} \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \cdots \subset L^2(\mathbb{R})$$

with the following properties:

1. $\bigcup_{l \in Z} V_l$ is dense in $L^2(\mathbb{R})$.
2. $\bigcap_{l \in Z} V_l = \{0\}$.
3. The embedded subspaces are related by a scaling law

$$f(x) \in V_l \Longleftrightarrow f(2x) \in V_{l-1}$$

4. Each subspace is spanned by integer translates of a single function $g(x)$ such that

$$f(x) \in V_0 \Longleftrightarrow f(x+1) \in V_0$$

Since the space V_1 lies within the space V_0, we can express any function in V_1 in terms of the basis of functions of V_0. In particular,

$$\phi(x) = \sum_{i=-\infty}^{\infty} a_i \phi(2x - i), \; i \in \mathbb{Z}$$

where i is a finite number and a_i is a square summation sequence. This equation is called dilation equation or scaling relation.

If we define

$$\phi_{l,i} = 2^{l/2} \phi(2^l - i)$$

then $\phi_{l,i}$, $i \in \mathbb{Z}$ forms a basis for the space V_l. The dilation parameter l shall be referred to as the scale.

In general, a function may be approximated by the projection $P_l f$ onto the space V_l:

$$P_l f = \sum_{i=-\infty}^{\infty} c_{l,i} \phi_{l,i}(x), \; i \in \mathbb{Z}$$

and in fact $P_l f$ approaches f when $l \to \infty$.

Let us now define a new subspace W_{l+1} such that it is the orthogonal complement of V_{l+1} in V_l,

$$V_l = V_{l+1} \oplus W_{l+1}, \; V_{l+1} \perp W_{l+1}$$

where $\oplus$ represents a direct sum. Let us introduce a wavelet function ψ such that $\psi(x - i)$ form a basis for the subspace W_0. Then

$$\psi_{l,i} = 2^{\frac{l}{2}} \psi(2^l x - i)$$

It is a basis for W_l. If in addition, the $\{\psi(x-i),\ i \in \mathbb{Z}\}$ forms an orthonormal set of functions, then it follows that $\{\psi_{l,i},\ l,i \in \mathbb{Z}\}$ forms an orthonormal basis for $L^2(\mathbb{R})$.

Let us denote the projection of f on W_l as $Q_l f$. Then we have

$$P_l f = P_{l+1} f + Q_{l+1} f$$

This means that $Q_l f$ represents the detail that needs to be added to get from one level of approximation to the next finer level of approximation.

Furthermore, since the space W_1 is contained in the space V_0, the wavelet function can be expressed in terms of the scaling function at the next higher scale,

$$\psi(x) = \sum_{i=-\infty}^{\infty} b_i \psi(2x-i),\ i \in \mathbb{Z}$$

There is a relation between the so called filter coefficients a_i and b_i

$$b_i = (-1)^i a_{N-1-i}$$

The derivation [53] of filter coefficients requires the solution of an N coefficient system where the approximation of order $p = \frac{N}{2}$ of any function f is involved.

3.6.2 Multiresolution decomposition and reconstruction

Multiresolution decomposition [54] takes the values $c_{l,i}$ of a function f vector x_l or a matrix A_l at level l and decomposes them into

1. The values $c_{l+1,i}$ of the approximation, $P_{l+1} f$ (low frequency components) at next coarser level $l+1$.

2. The values $d_{l+1,i}$ of the detail component, $Q_{l+1} f = P_l f - P_{l+1} f$ (high frequency components) at next coarser level $l+1$.

Consider a function f. Let $P_l f$ denote the projection of f onto the subspace V_l and $Q_l f$ denote the projection onto the subspace W_l. Thus,

$$P_l f = \sum_{i=-\infty}^{\infty} c_{l,i} \phi_{l,i}(x),\ c_{l,i} = < f, \phi_{l,i} >$$

$$Q_l f = \sum_{i=-\infty}^{\infty} d_{l,i} \psi_{l,i}(x),\ d_{l,i} = < f, \psi_{l,i} >$$

Since W_{l+1} is the orthogonal complement of V_{l+1} in V_l

$$P_{l+1} f = P_l f - Q_{l+1} f$$

Substituting this in

$$c_{l+1,i} = < P_{l+1} f, \phi_{l+1,i} >$$

leads to the following result:

$$c_{l+1,i} = \frac{1}{\sqrt{2}} \sum_{j=-\infty}^{\infty} c_{l,j} a_{j-2i}$$

Similarly, it can be shown that

$$d_{l+1,i} = \frac{1}{\sqrt{2}} \sum_{j=-\infty}^{\infty} d_{l,j} a_{N-1-j+2i}$$

Multiresolution reconstruction [54] uses coefficients $c_{l+1,i}$ and $d_{l+1,i}$ at level $l+1$ to reconstruct the coefficients $c_{l,i}$ at next finer level l.

Since W_{l+1} is the orthogonal complement of V_{l+1} in V_l,

$$P_l f = P_{l+1} f + Q_{l+1} f$$

Substituting this in

$$c_{l,i} = < P_l f, \phi_{l,i} >$$

leads to

$$c_{l,i} = \frac{1}{\sqrt{2}} \sum_{j=-\infty}^{\infty} c_{l+1,j} a_{i-2j} + \frac{1}{\sqrt{2}} \sum_{j=-\infty}^{\infty} d_{l+1,j} (-1)^i a_{N-1-i+2j}$$

3.6.3 Mallat's transform and inverse transform

The Mallat's transform [54] provides a simple means of transforming data from one level of resolution, l, to the next coarser level of resolution $l+1$. The inverse Mallat's transform is a transformation from the coarser level $l+1$ to the finer level l.

The Mallat's transform algorithm implements the decomposition process as follows. Consider a string of data $c_{l,i}$ of finite and even length n which represents the approximation , $P_l f$, to a function. For convenience, suppose that this data is periodic with period $n >> N$. Then the matrix form of multiresolution decomposition equation is

$$\begin{bmatrix} c_{l+1,0} \\ \times \\ c_{l+1,1} \\ \times \\ c_{l+1,2} \\ \times \\ \cdots \\ \cdots \\ c_{l+1,\frac{n}{2}-1} \\ \times \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} & \cdots & 0 \\ 0 & a_0 & a_1 & \cdots & a_{N-2} & \cdots & 0 \\ 0 & 0 & a_0 & \cdots & a_{N-3} & \cdots & 0 \\ 0 & 0 & 0 & \cdots & a_{N-4} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & a_{N-1} \\ a_{N-1} & 0 & 0 & \cdots & \cdots & \cdots & a_{N-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_2 & a_3 & a_4 & \cdots & 0 & \cdots & a_1 \\ a_1 & a_2 & a_3 & \cdots & 0 & \cdots & a_0 \end{bmatrix} \begin{bmatrix} c_{l,0} \\ c_{l,1} \\ c_{l,2} \\ c_{l,3} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ c_{l,n-2} \\ c_{l,n-1} \end{bmatrix}$$

in which $\times$ represents information of no value. The effect of the periodicity is simply a wrap around of the coefficients at the bottom left corner of the matrix.

A similar process gives the coefficients, $d_{l+1,i}$ of the detail which is lost in reducing the resolution of the data. The matrix form is

$$\begin{bmatrix} d_{l+1,0} \\ \times \\ d_{l+1,1} \\ \times \\ d_{l+1,2} \\ \times \\ \cdots \\ \cdots \\ d_{l+1,\frac{n}{2}-1} \\ \times \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} a_{N-1} & -a_{N-2} & \cdots & -a_0 & \cdots & 0 \\ 0 & a_{N-1} & \cdots & a_1 & \cdots & 0 \\ 0 & 0 & \cdots & -a_2 & \cdots & 0 \\ 0 & 0 & \cdots & a_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & -a_0 \\ -a_0 & 0 & \cdots & \cdots & \cdots & a_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N-3} & -a_{N-4} & \cdots & 0 & \cdots & -a_{N-2} \\ -a_{N-2} & a_{N-3} & \cdots & 0 & \cdots & a_{N-1} \end{bmatrix} \begin{bmatrix} c_{l,0} \\ c_{l,1} \\ c_{l,2} \\ c_{l,3} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ c_{l,n-2} \\ c_{l,n-1} \end{bmatrix}$$

The inverse Mallat transform implements the reconstruction process. The matrix form is

$$\begin{bmatrix} c_{l,0} \\ c_{l,1} \\ c_{l,2} \\ c_{l,3} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ c_{l,n-2} \\ c_{l,n-1} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} a_0 & 0 & 0 & \cdots & a_{N-1} & \cdots & a_1 \\ a_1 & a_0 & 0 & \cdots & 0 & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & 0 & \cdots & a_3 \\ a_3 & a_2 & a_1 & \cdots & 0 & \cdots & a_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N-2} & a_{N-3} & a_{N-4} & \cdots & \cdots & \cdots & a_{N-1} \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{N-3} & \cdots & 0 \\ 0 & 0 & 0 & \cdots & a_{N-2} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} c_{l+1,0} \\ 0 \\ c_{l+1,1} \\ 0 \\ c_{l+1,2} \\ 0 \\ \cdots \\ \cdots \\ c_{l+1,\frac{n}{2}-1} \\ 0 \end{bmatrix}$$

$$+\frac{1}{\sqrt{2}} \begin{bmatrix} a_{N-1} & 0 & 0 & \cdots & -a_0 & \cdots & -a_{N-2} \\ -a_{N-2} & a_{N-1} & 0 & \cdots & a_1 & \cdots & a_{N-3} \\ a_{N-3} & -a_{N-2} & a_{N-1} & \cdots & -a_2 & \cdots & -a_{N-4} \\ -a_{N-4} & a_{N-3} & -a_{N-2} & \cdots & 0 & \cdots & a_{N-5} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_1 & -a_2 & a_3 & \cdots & \cdots & \cdots & -a_0 \\ -a_0 & a_1 & -a_2 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -a_2 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & a_1 & \cdots & a_{N-1} \end{bmatrix} \begin{bmatrix} d_{l+1,0} \\ 0 \\ d_{l+1,1} \\ 0 \\ d_{l+1,2} \\ 0 \\ \cdots \\ \cdots \\ d_{l+1,\frac{n}{2}-1} \\ 0 \end{bmatrix}$$

3.6.4 Wavelet transfer operators

Let us denote the Mallat transformations H, G of a vector $x \in \mathbb{R}^n$,with n even.

$$H, G : \mathbb{R}^n \to \mathbb{R}^{\frac{n}{2}}$$

$$x_l = c_{l,i}, i = 0, \cdots, n-1$$

$$Hx_l = c_{l+1,i}, i = 0, \cdots, \frac{n-1}{2}$$

$$Gx_l = d_{l+1,i}, i = 0, \cdots, \frac{n-1}{2}$$

The Mallat transformations satisfy

$$H^T H + G^T G = I$$

$$HH^T + GG^T = I$$

$$GH^T = HG^T = 0$$

We use I to denote the identity matrix of appropriate size. Therefore, H may be interpreted as an averaging or smoothing operator whose smoothing effect increases with N. Conversely, G can be viewed as a difference approximation operator.

Following the philosophy of the multigrid methods, we first apply to the l-system $A_l x_l = b_l$ the iterative solver as pre-smoother ν_1 times with the initial guess $x_l^{(0)}$. An approximate solution $x_l^{(\nu_1)}$ is obtained. In general, the smoother only reduces error components in the direction of eigenvectors corresponding to large eigenvalues of A_l [22]. But in the presence of strong anisotropies, say in x-direction, the error $e_l^{(\nu_1)} = x - x_l^{(\nu_1)}$ may contain not only a part which is a low frequency in both the x and y-directions, but also a part which is a high frequency in x direction. Consequently, we approximate the error $e_l^{(\nu_1)}$ in both spaces $Vx_{l+1} \otimes Vy_{l+1}$ and $Wx_{l+1} \otimes Vy_{l+1}$ instead of only in $Vx_{l+1} \otimes Vy_{l+1}$ as in a standard multigrid procedure. Where $\otimes$ denotes the tensor product of the spaces, operators and vectors. Then, the resulting coarse grid correction for the two level multigrid is

$$x_l = x_l^{(\nu_1)} + [(H_x^T \otimes H_y^T) A_{l+1}^{-1} (H_x \otimes H_y) + (G_x^T \otimes H_y^T) A_{l+1}^{-1} (G_x \otimes H_y)](b_l - A_l x_l^{(\nu_1)})$$

It may be viewed as an additive subspace correction with respect to the orthogonal subspaces $Vx_{l+1} \otimes Vy_{l+1}$ and $Wx_{l+1} \otimes Vy_{l+1}$.

One has to remark that the transition from A_l to A_{l+1} increases the band width moderately as far as N becomes not too large. This fact must be kept in mind in order to obtain a constant band width if an incomplete LU factorization with fixed number of off diagonals is used for all levels.

3.6.5 The Haar's wavelet transfer operator

The Haar basis [53] is an orthogonal basis which has been known since 1910. It is also the earliest known example of wavelet basis, and perhaps one of the simplest. Furthermore, Haar wavelets supplies transfer operators without increasing the bandwidth of matrix. In this case $N = 2$ and the Mallat's transformation is quite close to piece wise constant restriction and prolongation operators described in the algebraic multigrid section.

$$a_0 = 1, \; a_1 = 1$$

$$H_x = \frac{1}{\sqrt{2}} R_x$$

$$H_y = \frac{1}{\sqrt{2}} R_y$$

$$H^2_{1,x} = \frac{1}{\sqrt{2}} \begin{bmatrix} \begin{bmatrix} 11 & & \\ & \cdots & \\ & & 11 \end{bmatrix}_{4\times 8} & & \\ & \ddots & \\ & & \begin{bmatrix} 11 & & \\ & \cdots & \\ & & 11 \end{bmatrix}_{4\times 8} \end{bmatrix}_{32\times 64}$$

$$H^2_{1,y} = \frac{1}{\sqrt{2}} \begin{bmatrix} \begin{bmatrix} 1 & & 1 \\ & \cdots & \\ & 1 & 1 \end{bmatrix}_{4\times 8} & & \\ & \ddots & \\ & & \begin{bmatrix} 1 & & 1 \\ & \cdots & \\ & 1 & 1 \end{bmatrix}_{4\times 8} \end{bmatrix}_{16\times 32}$$

Finally, the transformation of A_l into A_{l+1} for a 5-point formulation matrix leads

$$A_{l+1} = H_y H_x A_l H_y^t H_x^t$$

$$\begin{aligned}
A^s_{l+1}(i,j) &= \frac{1}{4}(A^s_l(2i,2j) + A^s_l(2i+1,2j)) \\
A^w_{l+1}(i,j) &= \frac{1}{4}(A^w_l(2i,2j) + A^w_l(2i,2j+1)) \\
A^e_{l+1}(i,j) &= \frac{1}{4}(A^e_l(2i+1,2j) + A^e_l(2i+1,2j+1)) \\
A^n_{l+1}(i,j) &= \frac{1}{4}(A^n_l(2i,2j+1) + A^n_l(2i+1,2j+1)) \\
A^p_{l+1}(i,j) &= \tfrac{1}{4}\ (A^p_l(2i,2j) + A^p_l(2i+1,2j) + \\
&\quad A^p_l(2i,2j+1) + A^p_l(2i+1,2j+1) + \\
&\quad A^s_l(2i,2j+1) + A^s_l(2i+1,2j+1) + \\
&\quad A^w_l(2i+1,2j) + A^w_l(2i+1,2j+1) + \\
&\quad A^e_l(2i,2j) + A^e_l(2i,2j+1) + \\
&\quad A^n_l(2i,2j) + A^n_l(2i+1,2j))
\end{aligned}$$

For a 7-point formulation in a three dimensional case the Haar wavelet is already used

$$A_{l+1} = H_z H_y H_x A_l H_z^T H_y^T H_x^T$$

For a 9-point formulation in a two dimensional case or a 19-point formulation in a three dimensional case, it is better to use a wavelet with $N > 2$. For example the Daubechies wavelet [53] of fourth order $N = 4$. However, as it was pointed out above, the pattern of the resulting matrix changes to a more dense pattern. The number of diagonal entries increases slightly at each level.

3.7 Comparison between AMG and MRA

We have seen that AMG and MRA share the same algorithm. Both of them have different grids or levels of resolution and the corrections or the details to the finest grid solution are added. Furthermore, the algebraic expression of the coarse grid approximation is quite similar. However, the transfer operators are implemented in different ways. For AMG, the restriction and prolongation operators act in blocks well defined (i.e. 2×2, 3×3, $2 \times 2 \times 2$ and $3 \times 3 \times 3$). Hence it is necessary to know the grid size and the distribution of grid points. For MRA based on wavelets, the transfer operators are applied directly as they were defined: products between matrices and vectors. This fact gives to the wavelet multiresolution analysis a great generalization to any sparse matrix pattern. Moreover, just changing the order of a wavelet, say Daubechies with orders 4, 6, 12 and 20, different coarse grid approximations can be more easily obtained than AMG does by blocking in predefined sets of grid points.

Finally this generalization can be extended even to non structured grids in order to have a non structured multigrid [55].

3.8 Stopping criteria

We have already described some families of iterative solvers based on ILU's and Krylov's subspaces. In the algorithm of each iterative solver, there is a common step, the stopping criterion. It decides when the algorithm has achieved the right solution within a certain error. Since the error in such linear systems is never known, the 2-norm of the residual vector $||r||_2$ has to be used.

$$||r||_2 = \sqrt{\sum_{i,j,k} r_{i,j,k}^2}$$

Thus, let k be the k-th iteration, if the $||r^{(k)}||_2$ is small enough, say smaller than a given threshold ϵ, the iterative solver can be stopped and get the solution of $x^{(k)}$.

Most solvers perform at least once the computation of the residual vector so it is easy to implement the stopping criterion based on its 2-norm. Others like GMRESR performs this computation directly without needing the residual vector. And others perform an approximation to the residual like the SIS [13]. In any case, the 2-norm of the residual vector leads to a good criterion of convergence to the solution.

However, this criterion must be generalized for all kind of problems. Different coefficient matrices and right hand sides can have solutions with large values and generate large residual values. Conversely, solutions with small values generate small residual values. Therefore a relative residual norm $\rho^{(k)}$ should be used instead of the residual norm $||r^{(k)}||$. For instance the residual norm is divided by the factor $||b||_2$ which features the problem.

$$\rho^{(k)} = \frac{||r^{(k)}||_2}{||b||_2}, \quad ||b||_2 \neq 0$$

Another stopping criterion is based on a fixed number of iterations ν. The solver stops when the counter of iterations achieve this value. That happens, for example, in inner loops of the multigrid in the pre-smoothing and the post-smoothing steps. In this case, the global

stopping criterion is based on the residual norm of the finest level. And the fixed number of iterations can be used as local criterion. For example, the preconditioning step is included in the krylov's solver and since it has to be with low computational cost, it performs only a fixed small number ν of iterations.

Since the number of iterations does not change, we say that the criterion is static. In other cases, the local number of iterations is dynamic and limited by a relative threshold ϵ_r and a maximum number of iterations it_{max}. The relative threshold ϵ_r is usually fixed in the range $70\% - 90\%$ of the residual norm $||r^{(0)}||_2$ at the guessed solution or before performing the local iterations.

$$\epsilon_r = \alpha||r^{(0)}||_2, \ \alpha \in [0.7, 0.9]$$

Finally, there is another dynamic criterion based on the convergence rate of the local iterative solver. The convergence rate β^k for the k-th iteration is expressed as

$$\beta^{(k)} = \frac{r^{(k-1)}}{r^k}$$

Representing $\beta^{(k)}$ versus the iteration number stagnation can be detected in the iterative procedure and exit the algorithm. This dynamic criterion is used in the decision of change to a coarse grid or a coarse multiresolution within the multigrid algorithm.

These criteria are embedded in the multigrid algorithm 16 with a preconditioned solver 17.

3.9 Sequential performance of solvers

The scope of this section is to provide a comparison between the different solvers built in the solver layer for a given model problem. The comparison is based on two issues: the time spent in the resolution until a defined residual criterion and the amount of memory needed to store the vectors and matrices involved in the computation of the solution. On the other hand, the model problem is designed to show the stability and convergence behaviour of each solver. The main idea underlying on this test is to find out a robust solver of a wide range of CFD problems.

3.9.1 CFD model problem

We are interested on the behaviour of solvers for convection and diffusion equations under the boundary conditions that produce ill conditioned linear systems. For this purpose, a two-dimensional partial differential equation with different boundary conditions has been designed:

$$\frac{\partial u\phi}{\partial x} + \frac{\partial v\phi}{\partial y} - \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f$$

$$(x, y) \in \Omega : [0, 2] \times [0, 2]$$

Where the velocity field $\vec{V} = \{u, v\}$ in function of the Reynolds number satisfies the mass conservation equation.

$$u = x^2(1 - 2y)Re, \quad v = 2x(y^2 - y)Re$$

Algorithm 16 AMG + solver + preconditioner + criteria

if$(l = 1)$
  fix parameters of multigrid cycle
    $l_{max}, it_{max}, \nu_1, \nu_2, \epsilon, \alpha, \beta$
  for$(l = 1$ *to* $l_{max} - 1)$
    evaluate *matrix coeff. by Coarse Grid Approximation*
      $A_{l+1} = R_l^{l+1} A_l P_{l+1}^l$
    $\epsilon_r = \epsilon$
  end do
else
  evaluate *right hand side of coarse level* $l + 1$
    $r_l = b_l - A_l x_l$
    $b_{l+1} = R_l^{l+1} r_l$
    $l = l + 1$
  $\epsilon_r = \alpha\epsilon$
end if
set *a guess in level* l
  $k = 0,\ x_l^{(k)}$
global static iteration
while$(\rho^{(k)} \geq \epsilon$ *and* $k \leq it_{max})$
    solve $A_l x_l = b_l$ *with preconditioner*
    if $(l < l_{max})$
      go *to the coarser level* $l + 1$
        call *AMG at l+1*
      add *the correction to the level* l
        $x_l = x_l + P_{l+1}^l x_{l+1}$
      solve $A_l x_l = b_l$ *with preconditioner*
    end if
    $k = k + 1;$
end while

Algorithm 17 Preconditioned solver

local dynamic iteration
while$(\frac{||r^{(k-1)}||_2}{||r^{(k)}||_2} \geq \beta$ *and* $k < \nu_1)$
  solve $A_l x_l = b_l$
    local static iteration
    for $(k = 1$ *to* $k = \nu_2)$
      solve $M_l \bar{p}_l = p_l$
    end for
end while

Then, the convective and diffusive terms are discretized to obtain the coefficient matrix A with a 5-point formulation.

In order to satisfy all boundary conditions (i.e. Dirichlet, Neumann and periodic conditions), a single solution ϕ is set to this problem.

$$\phi = cos(\pi x) + cos(\pi y) + cos(3\pi x) + cos(3\pi y)$$

Finally, the right hand side b is evaluated from the product of the matrix A by the vector ϕ instead of the integration of f at each volume $\mathcal{V}$.

$$b = f\mathcal{V} = A\phi$$

If the solution ϕ is known, it is possible to evaluate and analyze at each iteration the numerical error $e^{(k)}$ at any iteration k. Therefore a comparison of the convergence rates between solvers may be done. In this sense the Discrete Fourier Transform of the error help us to explain the spectral properties of the different solvers.

3.9.2 Sequential performance

For the given problem, the performance of each solver in a single process $np = 1$ is done. Since no communications are needed the convergence behaviour of each solver gives an idea of how good may it be solving the problem in a np parallel environment.

Solvers are compared from two points of view: the convergence behaviour and the memory resources required. A list of solvers is given in table 3.1.

Solver		Implementation features
0	band LU	Crout's implementation + partial pivoting
1	Gauss-Seidel	not overrelaxed
2	ILU	MSIP implementation, $\alpha = 0.5$
3	BiCGSTAB	preconditioned with MSIP($\alpha = 0.5$)
4	GMRESR	restart=10, preconditioned with MSIP($\alpha = 0.5$)
5	AMG	10 levels, smoother MSIP($\alpha = 0.5$)
6	MRA	10 levels, smoother MSIP($\alpha = 0.5$)

Table 3.1: List of the solvers used in this test.

Each solver performs iterations until the normalized residual norm reaches the accuracy $\epsilon = 10^{-4}$.

$$\frac{||r^{(k)}||_2}{||b||_2} < \epsilon$$

A battery of cases for different Reynolds numbers and different square grid sizes $I \times I$ are tested following Alg. 18. The results of the test, executed in the PC cluster (see appendix), are grouped by the Reynolds number and represented with the pair of axis: size problem(x) - time of computation(y). For Dirichlet boundary conditions, the results for three Reynolds $(0, 10^2, 10^4)$ are given in Figs. 3.15, 3.16 and 3.17.

Algorithm 18 Sequential performance of solvers

for ($solver = 0$ *to* $solver = 6$)
  for ($Re = 0$ *to* $Re = 10000$, $Re = Re * 100$)
    for ($I = 32$ *to* $I = 512$, $I = I * 2$)
      generate *the problem* $Ax = b$
      fix *accuracy of the solution to* $\epsilon = 10^{-4}$
      initialize *unknown* $x = 0$
      t0=start_wall_clock(gettimeofday)
        solve $Ax = b$ *such that* $\frac{\|r^2\|}{\|b^2\|} < \epsilon$
      t1=stop_wall_clock(gettimeofday)
      tcomp=t1-t0
      write *throughput: tcomp and memory resources*
    end for
  end for
end for

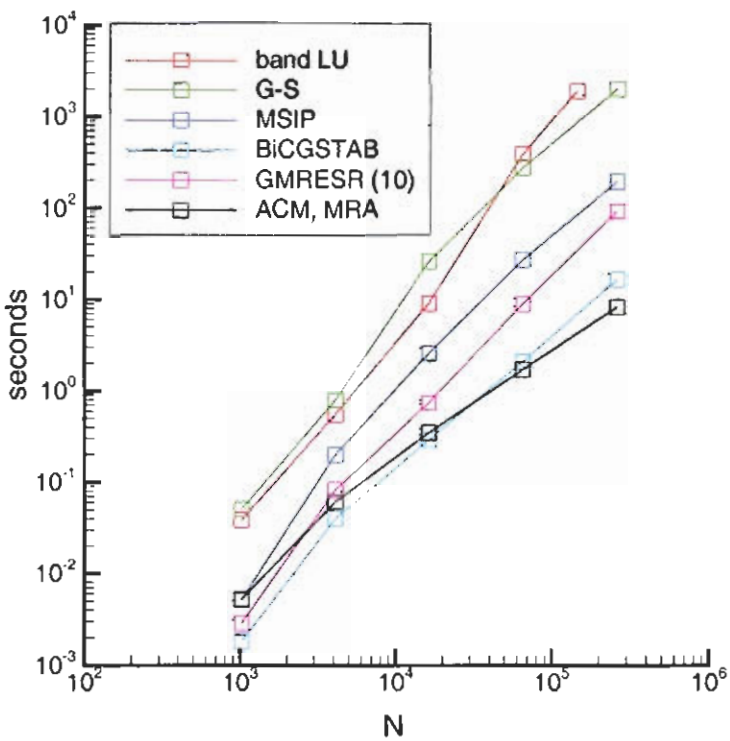

Figure 3.15: Time of computation at Re=0 for different size problems.

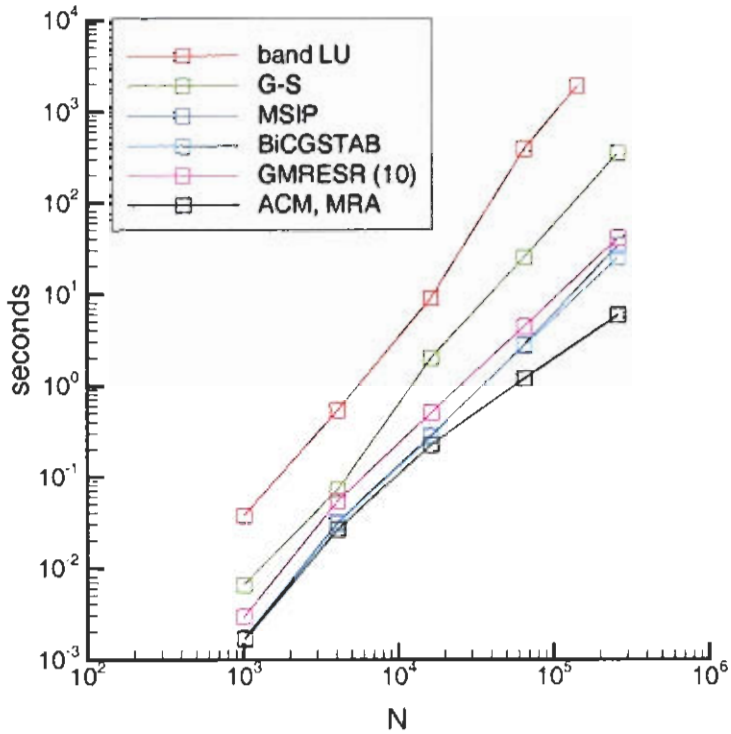

Figure 3.16: Time of computation at Re=100 for different size problems.

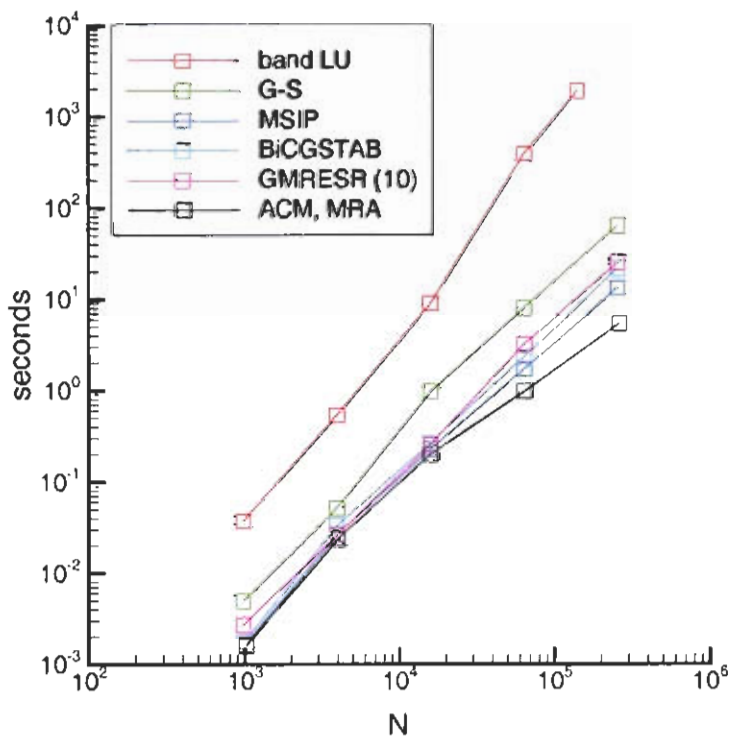

Figure 3.17: Time of computation at Re=10000 for different size problems.

Although full results have not been shown due to the large amount of tested cases, the above results show that ACM and MRA are the best solvers for all Reynolds numbers and problem sizes. Furthermore, the difference of time of computation among solvers grows with the size problem, being the worst case for the pure diffusion problem, (i.e. $Re = 0$) where the matrix is symmetric. For large Reynolds uumbers the matrix becomes non symmetric and the solvers work better. Since the pure diffusion problem is the hardest case, we reduce the number of cases here in after to such case and for any boundary condition.

For Neumann boundary conditions and for the pure diffusion case, (i.e. $Re = 0$), the same sequential performance test is repeated. Since the Neumann bouudary conditions lead to a singular matrix, a point at the center of the domain is fixed in order to set a single solution. The results represented in Fig. 3.18 are quite similar to the obtained for Dirichlet boundary conditions.

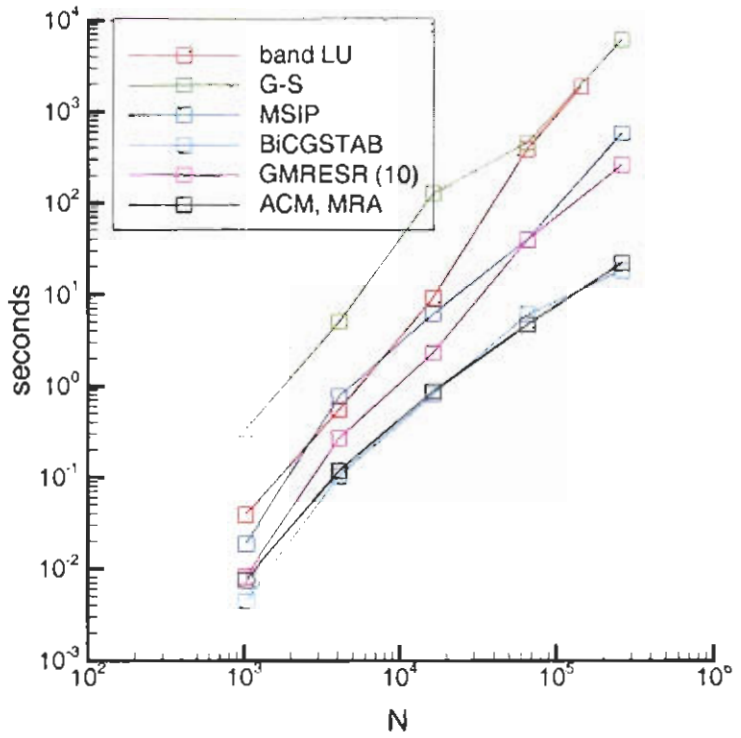

Figure 3.18: Time of computation at Re=0 for different size problems.

Fig. 3.18 shows that although BiCGSTAB reported good results the best convergence behaviour is with ACM and MRA solvers.

The memory requirements for each solver at each size are represented in Fig. 3.19.

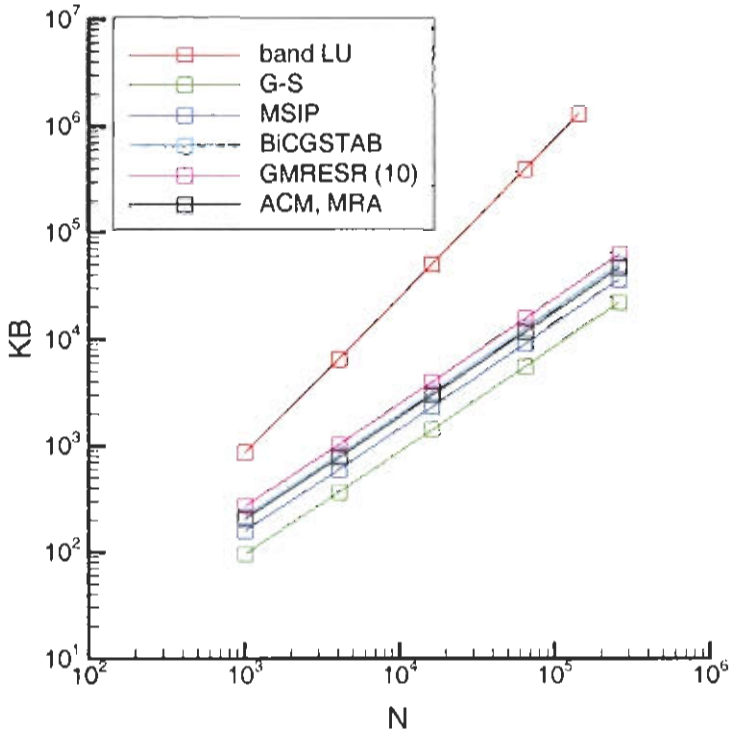

Figure 3.19: Memory requirements in Kbytes for different size problems.

It is worth noting that GMRES with a restart of 10 requires more memory resources than the rest of solvers. A higher value of the restart parameter is also tested but it increases largely the memory without a reduction of the time of computation. Therefore, ACM and MRA are also better than BiCGSTAB and GMRESR from the cost-memory point of view.

3.10 Nomenclature

A	discretization matrix
a	coeff. in A
	filter coeff.
b	right hand side, filter coeff.
CV	control volum
c	filter coeff.
d	auxiliar vector
e	unit vector
G	grid
g	similar to d
H	Hessenberg matrix, Haar wavelet
I	unknowns per direction
ILU	incomplete LU decomposition
i	index coordinate
K	Krylov space
L	lower matrix from LU
LU	lower upper decomposition
l	coeff. of matrix L
M	auxiliar matrix for LU
N	number of unknowns, similar to M
P	prediction, projection
p	similar to d
Q	matrix from QR, projection
QR	QR factorization
q	similar to d
R	restriction, similar to Q
Re	Reynolds number
r	residual
s	similar to d
t	similar to d
U	upper matrix from LU
u	coeff. of matrix U,
V	Krylov base, subspace
v	similar to d
W	wavelet, subspace
w	similar to d
x	unknown
y	similar to d
z	similar to d
$\{u, v, w\}$	fluid flow velocity components
$\{x, y, z\}$	cartesian coordinates

Greek symbols

α	relaxation parameter, scalar value
β	scalar value
δ	general perturbation
κ	condition number
ϵ	precission
μ	similar to β
ν	fixed number of iterations
Ω	domain
ω	similar to β
ϕ	general variable, scaling function
ψ	wavelet function
ρ	similar to β

Other symbols

$5-PF$	five point formulation
$7-PF$	seven point formulation
$9-PF$	nine point formulation
$19-PF$	nineteen point formulation
$\mathcal{V}$	volume of the CV
$<,>$	inner product
$\|\|.\|\|_2$	Euclidean norm
$\oplus$	tensor product

Subscripts

NGB	general neighbour grid point
$*,$	for all elements of a row
$,*$	for all elements of a column
l	level, go down to the level l

Superscripts

(k)	k-th iteration
-1	inverse
T	transpose
$\frac{1}{2}$	square root
$-$	modified value
l	go up to the level l