Proceedings of the 11th Brazilian Congress of Thermal Sciences and Engineering – ENCIT 2006
Braz. Soc. of Mechanical Sciences and Engineering – ABCM, Curitiba, Brazil, Dec. 5-8, 2006

Paper CIT06-0433

COMPARISON OF POISSON SOLVERS IN A TRANSIENT 2D FLUID FLOW PROBLEM WITH DIRICHLET BOUNDARY CONDITIONS

Leandro F. Souza
ICMC – USP, Av. Trabalhador São-carlense, 400 - Cx. Postal: 668 - CEP: 13560-970 - São Carlos - SP
lefraso@icmc.usp.br

Cassio M. Oishi
ICMC – USP, Av. Trabalhador São-carlense, 400 - Cx. Postal: 668 - CEP: 13560-970 - São Carlos - SP
oishi@lcad.icmc.usp.br

José A. Cuminato
ICMC – USP, Av. Trabalhador São-carlense, 400 - Cx. Postal: 668 - CEP: 13560-970 - São Carlos - SP
jacumina@icmc.usp.br

Aristeu da Silveira Neto
UFU, Av. Engenheiro Diniz, 1178 - Cx. Postal: 593 - CEP: 38.400-902 - Uberlândia - MG
aristeus@mecanica.ufu.br

Abstract. *The numerical simulation of incompressible fluid flow often requires a Poisson solver subroutine. For the majority of these Computational Fluid Dynamics codes, this subroutine is one of the greatest CPU time consumer. In the present paper 6 different Poisson solver subroutines are presented and compared, namely: point SOR (Successive Over-Relaxation), line SOR, conjugate gradient method, Full Approximation Storage (FAS) multigrid solver with: line SOR, conjugate gradient method and a mixed (line SOR – conjugate gradient). All of the above methods use second order finite difference discretization in space. A Cartesian grid is used for each test case. The distance between consecutive points is constant in both directions. A performance test of the Poisson subroutines is carried out using a problem with analytical solution. An evaluation of the Poisson subroutines in a 2D incompressible flow simulation is done using a driven cavity flow as benchmark. In these cases, the equations are written in vorticity-stream function formulation. A fourth order Runge-Kutta scheme is used as time integrator. Different CFL numbers are used with each Poisson solver. The results show that the multigrid solvers are the most efficient for the tests performed here.*

keywords: *Poisson equation, Vorticity-Stream function formulation, Multigrid methods, incompressible flow.*

1. Introduction

Incompressible flow phenomena arise in a number of applications in science and engineering. Almost all numerical solutions of incompressible fluid flows requires a Poisson solver subroutine. Therefore, the study of efficient iterative methods and multigrid schemes for solving the Poisson equation are very important in Computational Fluid Dynamic (CFD).

The numerical solution of the Poisson equation can be obtained by *iterative methods* or *direct methods*. In the present study only *iterative methods* are evaluated. The term *iterative method* refers to a wide range of techniques that use successive approximations to obtain more accurate solutions of linear systems at each time step. In this work we will study two classes of iterative methods: stationary and non-stationary methods.

Stationary methods, known as Basic iterative methods, that are older but simpler to understand and implement, and usually not very efficient. In this paper, the Basic iterative method considered is the Successive Over-Relaxation. The Successive Over-Relaxation can be derived from the Gauss-Seidel method by introducing an relaxation parameter ω. For the optimal choice of ω, Successive Over-Relaxation may converge faster than Gauss-Seidel by an order of magnitude. An elementary introduction is given in Wesseling, 2001; Saad, 2003.

Non-stationary methods, known as Krylov subspace methods are relatively recent; their analysis are usually harder to understand, but they can be highly effective. The Conjugate Gradient method was adopted here as a type of Krylov subspace method. The conjugate gradient method derives its name from the fact that it generates a sequence of conjugate (or orthogonal) vectors. These vectors are the residuals of the iterations. They are also the gradients of a quadratic functional, the minimization of which is equivalent for solving the linear system. The conjugate gradient converges to the correct solution when the matrix is symmetric positive definite, and is very efficient, since storage for only a limited number of vectors is required. Various authors have presented the theory and implementations of conjugate gradient (Golub and Van Loan, 1989; Baker, 1991; Wesseling, 2001; Saad, 2003).

Convergence acceleration techniques linked with iterative methods have recently been developed and have led to a dramatic improvement in convergence rates Hirsch, 1988. The multigrid method works as an acceleration technique and is the most effective and general iterative technique known today. This method has its origin in the properties of conventional iterative techniques, in particular, their asymptotic slow convergence because of the poor damping of high frequency errors (short wavelengths). The basic ideas of a multigrid method follows the lines:

- apply one or more sweeps of an iterative method with good smoothing properties of the higher-frequency components;
- transfer the problem to a coarse grid;
- transfer the corrections obtained to the fine grid in order to generate a new approximation of the solution.

In the current work an analysis of Poisson solvers subroutines was done. In the first test case the subroutines were evaluated in a simple test where the analytical solution is known. The performance of the Poisson solvers subroutines in a numerical simulation of a 2D incompressible flow was also analyzed. In these cases the Navier-Stokes equation were written in vorticity-stream function formulation. The benchmark test adopted was a driven cavity flow. Three different mesh sizes were tested, and for each mesh many CFL numbers were imposed. These tests enable a evaluation of the performance of the Poisson solver subroutines in a 2D flow simulation.

2. Formulation

For the numerical solution, the Navier-Stokes equations were written in vorticity-stream function formulation. The vorticity in the spanwise direction, denoted by ω, is:

$$\omega = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}. \tag{1}$$

and the stream function ψ is defined such that:

$$u = \frac{\partial \psi}{\partial y}, \tag{2}$$

and

$$v = \frac{\partial \psi}{\partial x} \tag{3}$$

The vorticity transport equation can be obtained by applying a rotational of the momentum equation is:

$$\frac{\partial \omega}{\partial t} = -u\frac{\partial \omega}{\partial x} - v\frac{\partial \omega}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right). \tag{4}$$

The continuity equation is:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{5}$$

Using the definitions of stream function Eqs. (2) and (3) and applying them in the vorticity definition Eq. (1) a Poisson-type equation for the ψ can be derived:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \omega. \tag{6}$$

Equations (2), (3), (4) and (6) were solved numerically by the schemes described below.
The solution was marched in time according to the following steps:

1. Impose initial conditions for u, v, ω and ψ compatible with each other;
2. Calculate the vorticity from the vorticity transport equation (4), for time $t + dt$;
3. Calculate ψ from the Poisson equation (6);

4. Calculate v velocity from Eq. (3);
5. Calculate u velocity from Eq. (2);
6. Calculate the vorticity generation at the wall for the new velocity distribution;
7. return to the second step until the desired integration time is reached.

3. Numerical Method

The time derivatives in the vorticity transport equations were discretized with a classical 4^{th} order Runge-Kutta integration scheme Ferziger and Peric, 1997. The spatial derivatives were calculated using a 2^{nd} order finite difference schemes. The vorticity updating at the wall, for all boundaries but the upper boundary, was done using the formula:

$$\omega_{wall} = \frac{-85\psi_{wall} + 108\psi_{wall\pm1} - 27\psi_{wall\pm2} + 4\psi_{wall\pm3}}{18(\Delta)^2} \tag{7}$$

The vorticity at the upper boundary was calculated by:

$$\omega_{jmax} = \frac{11u_{jmax} - 18u_{jmax-1} + 9u_{jmax-2} - 2u_{jmax-3}}{6\Delta y} \tag{8}$$

In the present work, six different Poisson solver subroutines were evaluated. The details of each one are given below.

3.1. Poisson Solvers

The Poisson solvers tested in the current work are classified as iterative methods. The algebraic systems that arise from the discretization are sparse and can be very large (depending on the number of points in both directions). Therefore iterative methods are more efficient and demand far less storage than direct methods. The methods implemented can be classified as:

3.1.1. Basic iterative methods

These methods were very popular in the 60's. The first iterative methods for solving large linear system were based on relaxation of the coordinates. Beginning with a given approximate solution, they modify the components of the approximation, one or a few at a time and in a certain order, until convergence is reached. One of the main difficulties with these methods is finding an optimal relaxation factor for general matrices. However when combined with more efficient methods they can be quite successful. Moreover, there are some application areas where variations of these methods are still quite popular Wesseling, 2001.

3.1.2. Krylov subspace methods

The system to be solved can be written as $Ax = b$, where A is a large and sparse matrix. These techniques are based on projection processes, both orthogonal and oblique, onto Krylov subspace, which are subspaces spanned by vectors of the form $p(A)v$, where p is a polynomial. In short, these techniques approximate $A^{-1}b$ by $p(A)v$, where p is a "good" polynomial Saad, 2003. These methods are widely used because their rate of convergence can be estimated. The attractiveness of those methods, for large sparse systems, is that they reference A only through the multiplication by a vector. The gradient (or steepest descent) and the conjugate gradient methods can be applied to the solution of $Ax = b$ only when A is symmetric and positive definite.

3.1.3. Multigrid methods

The convergence of Krylov's subspace methods for solving systems arising from discretized partial differential equations tends to slow down considerably as these systems become larger. In contrast the multigrid methods are, in theory, independent of the mesh size. One significant difference with the Krylov subspace approach is that multigrid methods were initially designed specifically for the solution of the discretized elliptic Partial Differential Equations. The multigrid methods were later extended to nonlinear systems, and other type of equations. The discovery of multigrid methods Brandt, 1977 is the most significant development in numerical analysis in the last 30 years, and is having a strong impact on Computational Fluid Dynamics (CFD) Wesseling, 2001. A review of the application of multigrid in CFD is showed by Wesseling, 2001. The main advantage of these methods is their very fast convergence rate, however they may require implementations that are specific to the physical problem at hand. Three multigrid methods were tested and their main differences are the solver used in each grid. A Full Approximation Scheme (FAS) was adopted, and a V cycle with 4 meshes were used in all the methods described bellow.

There are many multigrid methods that solve elliptic problems (Briggs, 1987; Gupta *et al.*, 1997; Spitaleri, 2000; Zhang, 1996; Zhang, 1997). However, the multigrid idea can also be applied to nonlinear problems. The adopted algorithm in this work is the FAS applied in a V cycle with 4 meshes (Stüben and Trottenberg, 1981). The Poisson equation, Eq. (6) can be written in the form:

$$\nabla^2 f = st, \tag{9}$$

where ∇^2 is the Laplacian operator, f is the unknown solution, and st is the source term. This equation is solved in each grid, using an iterative procedure.

The multigrid scheme works in the following way, taking h as the distance between two consecutive points in the finest mesh, 2 iterations is performed in this mesh (h):

$$\nabla^2 f_h = st_h. \tag{10}$$

The residual is calculated next from (res_h):

$$res_h = st_h - \nabla^2 f_h. \tag{11}$$

To define the variables in the next grid (coarser), a operation called restriction is used:

$$f_h^i \quad \Rightarrow \quad f_{2h}^0 \qquad (SI), \tag{12}$$

$$res_h^i \quad \Rightarrow \quad res_{2h}^0 \qquad (FW), \tag{13}$$

where SI stands for Straight Injection and FW stands for Full Weight. More details of the restriction operation can be found in Press *et al.*, 1997.

Then, the source term for the second mesh is calculated:

$$st_{2h} = res_{2h}^0 + \nabla^2 f_{2h}^0. \tag{14}$$

Here again 2 iterations are done in this mesh ($2h$):

$$\nabla^2 f_{2h} = st_{2h}. \tag{15}$$

After that, the residual is calculated from(res_{2h}):

$$res_{2h} = st_{2h} - \nabla^2 f_{2h}, \tag{16}$$

and this procedure is repeated until the coarsest mesh ($8h$) is reached, where 40 iterations are performed:

$$\nabla^2 f_{8h} = st_{8h}. \tag{17}$$

The way back to the finest mesh starts with the calculation of the correction on the coarsest mesh:

$$corr_{8h} = f_{8h}^i - f_{8h}^0. \tag{18}$$

This correction is interpolated, using a bi-linear interpolation to the grid $8h$ to the next grid $4h$:

$$corr_{8h} \Rightarrow corr_{4h}. \tag{19}$$

After that, the function is corrected, using new values:

$$f_{4h} = f_{4h}^i + corr_{4h}. \tag{20}$$

One single iteration is performed in this mesh ($4h$):

$$\nabla^2 f_{4h} = st_{4h}, \tag{21}$$

Then the correction in this mesh is also calculated, and this procedure is repeated until the finest mesh is reached:

$$\nabla^2 f_h = st_h. \tag{22}$$

The number of V cycles used to obtain the solution of the Poisson equation depends on the number of points in the mesh and on the minimum residual allowed on it. Figure 1 illustrates the above multigrid procedure.

Three different iterative schemes were used with the multigrid method and are described below.

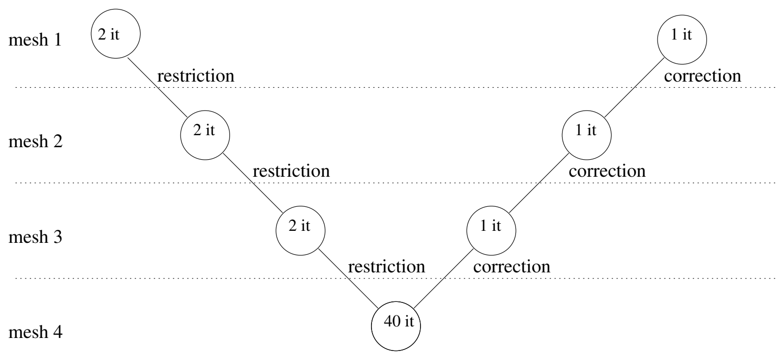

Figure 1: V Cycle.

- Multigrid with LSOR (FAS-LSOR)

  This method uses a Line Successive Over-Relaxation (LSOR) as the iterative procedure in each mesh. Special attention should be given to the relaxation factor used in this procedure, a value between 1 and 2 can be used when the method goes from the fine to the coarse grid but this factor must be 1 when the method goes from coarse to the fine grid, otherwise the smoothing of the high frequencies, main idea of this scheme, does not work.

- Multigrid with Conjugate Gradient (FAS-CG)

  This method uses the Conjugate Gradient (CG) iterative procedure in each mesh. The main advantage is that no relaxation factor is needed in this scheme.

- Multigrid with LSOR and CG (FAS-mix)

  Several tests were performed changing the iterative procedures in the multigrid scheme. The best CPU time results were obtained when Line Successive Over-Relaxation (LSOR) was adopted in the 3 first meshes and a Conjugate Gradient (CG) was used in the coarsest mesh.

4. Results

This section is divided in two, the first one shown the comparison of the Poisson solvers for a simple test case; and the second part shows the results of a 2 dimensional incompressible flow simulation where the Poisson solvers were evaluated. The results were obtained using Atlon AMD 64 3200+ computers. The processing time obtained in each simulation is shown in seconds.

4.1. Poisson equation case

Initially, the efficiency of each solver is tested for the Poisson equation (6) where the minimum and maximum values of x and y were 0 and 1, respectively. The boundary conditions and the source term for this test case are given by:

$$\begin{array}{rcl}
\text{Left boundary} & \Rightarrow & \psi(0,y) = sin(y) \\
\text{Right boundary} & \Rightarrow & \psi(1,y) = cos(1)sin(y) \\
\text{Lower boundary} & \Rightarrow & \psi(x,0) = 0.0 \\
\text{Upper boundary} & \Rightarrow & \psi(x,1) = cos(x)sin(1) \\
\text{Source term} & \Rightarrow & \omega(x,y) = -2cos(x)sin(y)
\end{array}$$

Observing the results in Tab. 1, and comparing the PSOR, LSOR and CG methods, one can note that the CG method is faster than the other PSOR and LSOR methods when we have a large number of points. Comparing the results obtained by the multigrid methods, the FAS-MIX method is the faster in almost all cases. The FAS-CG method is the slowest method, when comparing with the other multigrid methods, but is faster than the PSOR, LSOR and CG methods in the most tested cases. The result obtained with the multigrid method with CG solver demands a construction of a complex system in each mesh, and this causes an increase in processing time.

Table 1: CPU time (seconds) for test case.

Mesh	PSOR	LSOR	CG	FAS-LSOR	FAS-CG	FAS-MIX
65×65	0.02299	0.04999	0.04799	0.02599	0.03499	0.02599
65×129	0.14497	0.52791	0.17597	0.05598	0.16397	0.05598
65×257	1.06381	4.38333	0.85687	0.11700	0.89385	0.11596
65×513	9.60253	38.0681	4.19140	0.25	1.23681	0.24218
129×65	0.14597	0.10198	0.17497	0.05698	0.15797	0.05499
129×129	0.48791	1.03584	0.51791	0.12197	0.20797	0.12098
129×257	2.68560	8.86767	2.32775	0.25891	1.01672	0.25195
129×513	20.2465	75.6035	9.66210	0.54199	11.7773	0.52441
257×65	1.06584	0.20295	0.86486	0.24996	0.86788	0.24295
257×129	2.61560	2.08966	2.48657	0.28198	1.25488	0.27795
257×257	8.88867	18.0651	6.02392	0.58789	1.24169	0.57812
257×513	45.0468	146.985	21.5058	1.22851	18.1875	1.18359
513×65	8.41259	0.40405	4.58837	1.97070	1.32690	1.91577
513×129	18.3930	4.27832	8.84277	1.14892	7.58691	1.14794
513×257	43.5468	35.9785	19.8515	1.99609	8.56445	1.22656
513×513	139.898	291.054	43.7070	4.12500	5.38281	2.49609

4.2. Navier-Stokes equation cases

In order to verify the efficiency of the Poisson subroutine solvers in a transient simulation, many CFL numbers were imposed. The simulations were stopped when:

$$\omega_{i,j}^{n+1} - \omega_{i,j}^{n} \leq 1 \times 10^{-5}. \tag{23}$$

This criteria was imposed in all point (i, j) of the domain. The CPU times to obtain this criteria are shown bellow, for each Poisson subroutine solvers. The graphics are plotted in log x log scales, with CFL number in the x direction and CPU time in the y direction. The CFL numbers are plotted in a reverse axis direction.

Figure 2 shows the results obtained for a mesh of 65 x 65 points. These results using FAS-MIX and FAS-LSOR subroutines showed the lower CPU times for any CFL number. The FAS-MIX results were slightly better than the FAS-LSOR results in all tested cases. The use of FAS-CG, PSOR and LSOR subroutines in the numerical code showed similar results. The CPU times obtained with PSOR subroutine were better than the FAS-CG and LSOR. The CG subroutine results required the larger CPU times, if compared to the others. These could be explained by the fact that the solver was not preconditioned. It was verified that the CG subroutine demands more CPU time to obtain a solution, if compared to the other methods, when this solution is close to the initial condition.

The CPU time results obtained with a mesh of 129 x 129 points were very similar to the results with a mesh of 65 x 65 points as can be seen in Fig. 3. The only difference is that, with CFL number lower than 5×10^{-2}, the use of FAS-CG subroutine consumed more CPU time than LSOR and PSOR methods.

Table 2 shows the CPU time results for a mesh with 257 x 257 points. It can be seen that the results with FAS-MIX subroutine were the fastest than the other tested methods. The simulations with CFL = 0.01, using PSOR, LSOR and CG subroutines, were not performed because of the long time required. In these cases, the use of FAS-CG method showed best results if compared to PSOR, LSOR and CG results. The CPU time results using multigrid methods were lower than the other iterative methods. As the number of points increases this becomes more visible, as expected.

Table 2: CPU time results (seconds) versus CFL number – 257 x 257 points.

CFL Number	FAS-LSOR	FAS-MIX	FAS-CG	PSOR	LSOR	CG
0.1	4244.456	3824.622	22057.372	83874.687	90526.093	116068.031
0.05	6211.566	5964.343	34671.631	128348.285	137669.625	195534.25
0.01	16248.661	16076.160	87336.236	-	-	-

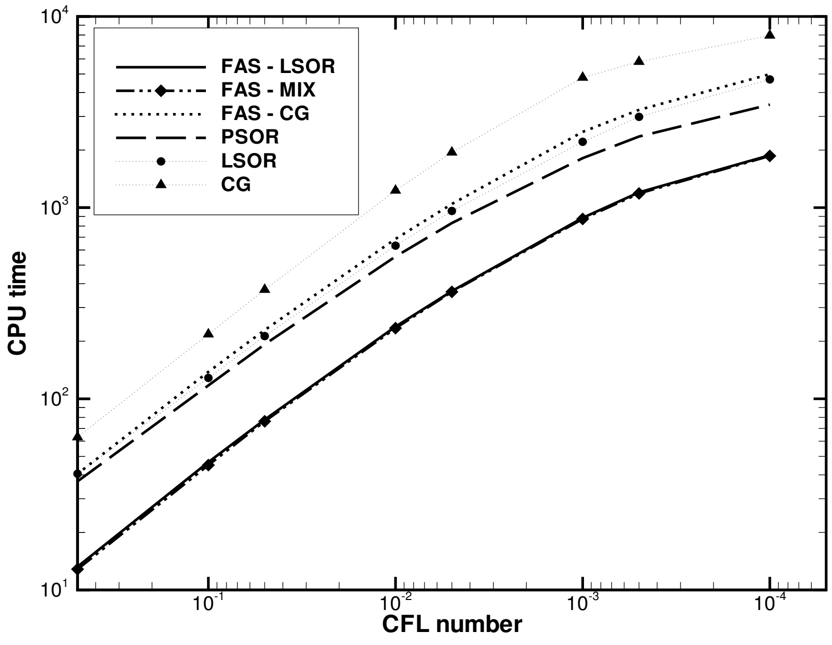

Figure 2: CPU time results (seconds) versus CFL number – 65 x 65 points.

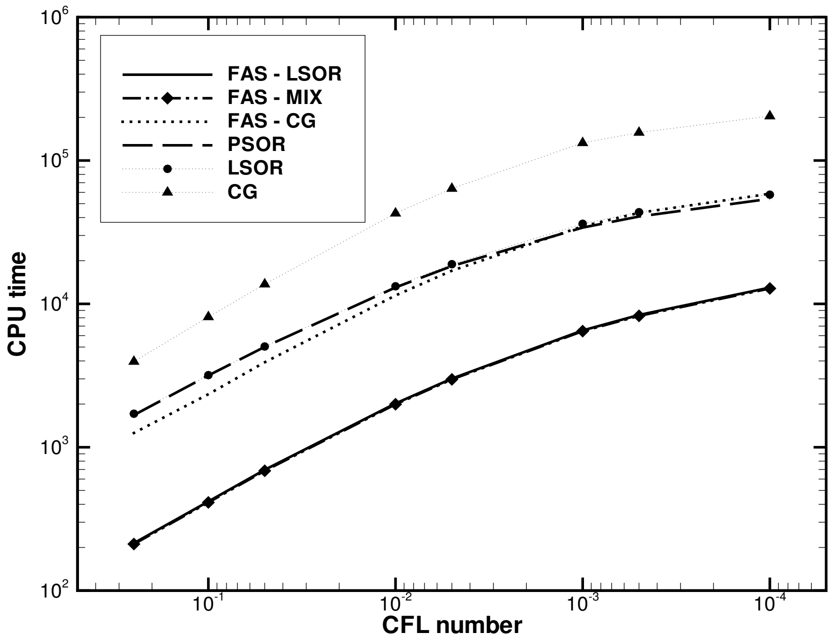

Figure 3: CPU time results (seconds) versus CFL number – 129 x 129 points.

5. Conclusions

In the present work, a study of the Poisson solver subroutine was carried out. Six different Poisson solver subroutines were studied and compared, namely: point Successive Over-Relaxation – PSOR, line Successive Over-Relaxation – LSOR, Conjugate Gradient method – CG, Full Approximation Storage (FAS) multigrid solver with: line SOR, conjugate gradient method and a mixed (line SOR – conjugate gradient). These solvers were tested with a Poisson equation with analytical solution and with a 2D incompressible flow simulation. A driven cavity flow was adopted as benchmark. Different CFL numbers were used with each Poisson solver. The proposed FAS-MIX method, that uses a mixture of basic methods and Krylov subspace methods showed the best results. The main conclusion of this work is that multigrid methods, as a convergence acceleration technique, should be used if one wants to develop a high performance code for solving Poisson equation.

6. Acknowledgments

The authors acknowledge the financial support given by FAPESP under grants 03/12612-9 and 04/07507-4.

7. References

Baker, L., 1991, "More C tools for scientist and engenieers", McGraw-Hill.

Brandt, A., 1977, Multilevel adaptative solutions to boundary values problems, "Mathematics of Computation", Vol. 31, pp. 333–390.

Briggs, W., 1987, "A Multigrid Tutorial", SIAM, Lancaster Press.

Ferziger, J. H. and Peric, M., 1997, "Computational Methods for Fluid Dynamics", Springer-Verlag Berlin Heidelberg New York.

Golub, G. H. and Van Loan, C. F., 1989, "Matrix Computations", Johns Hopkins University Pres.

Gupta, M. M., Kouatchou, J., and Zhang, J., 1997, Comparison of second- and Fourth-Order Discretizations for Multigrid Poisson Solvers, "J. Computational Physics", Vol. **132**, pp. 226–232.

Hirsch, C., 1988, "Numerical Computation of Internal and External Flows", Wiley & Sons.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1997, "Numerical Recipes in Fortran 77", Cambridge University Press.

Saad, Y., 2003, "Iterative Methods for Sparse Linear Systems", SIAM.

Spitaleri, R. M., 2000, Full-FAS Multigrid Grid Generation Algorithms, "Applied Numerical Math.", Vol. **32**, pp. 483–494.

Stüben, K. and Trottenberg, U., 1981, "Nonlinear multigrid methods, the full approximation scheme", chapter 5, pp. 58–71, Lecture Notes in Mathematics. Köln-Porz.

Wesseling, P., 2001, "Principles of Computational Fluid Dynamics", Springer-Verlag.

Zhang, J., 1996, Acceleration of Five-Point Red-Black Gauss-Seidel in Multigrid for Poisson Equation, "Applied Math. and Comp.", Vol. **80**, pp. 73–93.

Zhang, J., 1997, Residual Scaling Techniques in Multigrid, I: Equivalence Proof, "Applied Math. and Comp.", Vol. **80**, pp. 283–303.