INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN BIOMEDICAL ENGINEERING
Int. J. Numer. Meth. Biomed. Engng. 2010; **26**:73–85
Published online 16 March 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm.1235

Fast numerical solutions of patient-specific blood flows in 3D arterial systems

Fernando Mut*,†, Romain Aubry, Rainald Löhner and Juan R. Cebral

Center for Computational Fluid Dynamics, George Mason University, Fairfax, VA 22030-4444, U.S.A.

SUMMARY

The study of hemodynamics in arterial models constructed from patient-specific medical images requires the solution of the incompressible flow equations in geometries characterized by complex branching tubular structures. The main challenge with this kind of geometries is that the convergence rate of the pressure Poisson solver is dominated by the graph depth of the computational grid. This paper presents a deflated preconditioned conjugate gradients (DPCG) algorithm for accelerating the pressure Poisson solver. A subspace deflation technique is used to approximate the lowest eigenvalues along the tubular domains. This methodology was tested with an idealized cylindrical model and three patient-specific models of cerebral arteries and aneurysms constructed from medical images. For these cases, the number of iterations decreased by up to a factor of 16, while the total CPU time was reduced by up to 4 times when compared with the standard PCG solver. Copyright © 2009 John Wiley & Sons, Ltd.

Received 31 July 2008; Revised 12 December 2008; Accepted 28 January 2009

KEY WORDS: hemodynamics; cerebral aneurysms; deflated conjugate gradients; cerebral arteries; computational fluid dynamics

1. INTRODUCTION

Detailed patient-specific hemodynamic information is important for better understanding vascular diseases and to improve and personalize their treatment. Image-based computational fluid dynamics models have been used in a wide variety of applications [1]. The geometry of these models is typically characterized by a complex network of interconnected tubular structures. These geometries make the numerical solution of the flow equations computationally expensive. This is one of the reasons why these methods have not yet been introduced into the routine clinical practice. The objective of this work was to improve the performance of incompressible flow solvers for patient-specific hemodynamics calculations, making them more practical for clinical purposes.

Several numerical schemes have been used to solve the incompressible Navier–Stokes equations [2–4]. Many of them are based on so-called projection techniques, where the advancement of the flowfield in time is split into the following three substeps [5]:

(a) *advective–diffusive* prediction: $\mathbf{v}^n \rightarrow \tilde{\mathbf{v}}$

$$\left[\frac{\rho}{\Delta t} - \nabla\mu\nabla\right](\tilde{\mathbf{v}} - \mathbf{v}^n) + \rho\mathbf{v}^n \cdot \nabla\mathbf{v}^n + \nabla p^n = \nabla\mu\nabla\mathbf{v}^n \tag{1}$$

*Correspondence to: Fernando Mut, Center for Computational Fluid Dynamics, George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: fmut@gmu.edu

Contract/grant sponsor: Philips Medical Systems

(b) *pressure* correction: $p^n \to p^{n+1}$

$$\nabla \cdot \mathbf{v}^{n+1} = 0$$

$$\rho \frac{\mathbf{v}^{n+1} - \tilde{\mathbf{v}}}{\Delta t} + \nabla(p^{n+1} - p^n) = 0$$

which, by taking the divergence, results in

$$\nabla^2(p^{n+1} - p^n) = \rho \frac{\nabla \cdot \tilde{\mathbf{v}}}{\Delta t} \tag{2}$$

(c) *velocity* correction: $\tilde{\mathbf{v}} \to \mathbf{v}^{n+1}$

$$\mathbf{v}^{n+1} = \tilde{\mathbf{v}} - \Delta t \nabla(p^{n+1} - p^n) \tag{3}$$

Stabilization is achieved by a consistant edge-based discretization of the numerical fluxes in space that lead to a fourth-order damping for the divergence constraint as described in [5]. The discretization of the so-called pressure Poisson equation (2) leads to a very large but symmetric system of linear equations. Therefore, a system of linear equations of the form $Ax = b$ needs to be solved at each time step. The solution of this system is typically carried out using a preconditioned conjugate gradients (PCG) technique. The classical a priori bound for the error of the conjugate gradients method is given by [6]

$$\|x_* - x_k\|_A \leqslant 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^k \|x_* - x_1\|_A \tag{4}$$

where x_* is the exact solution, x_k is the approximate solution at the kth step, κ is the condition number of the matrix A and $\|\cdot\|_A = (A\cdot, \cdot)$ is the energy norm. It is known that the condition number becomes very large for tubular or elongated domains. This is due to the lowest eigenmodes of the Laplace operator that point in the longest direction of the domain (smallest eigenvalues). However, as described in [7], the convergence speeds up as soon as the lowest eigenvalues are 'discovered' by the CG process, giving rise to a condition number based on the active i.e. the nondiscovered eigenvalues. This suggests that removing the lowest eigenmodes from the spectrum of A would improve convergence as compared with the classical conjugate gradients process. That is what deflated conjugate gradients (DCG) algorithms try to achieve.

The first paper to consider a deflation method for the CG solver is perhaps due to Nicolaides [8]. The main idea was to remove certain components of the initial residual that may impede convergence. It also introduced the idea of subdomain deflation, where the slow varying components of the solution are approximated by a coarser discretization of the domain. However, no numerical experiments were reported. The subdomain deflation approach was successfully applied to the bending of a cantilever beam and to the stationary Stokes problem, as reported in Mansfield [9]. Deflation was also used for an augmented conjugate gradients [10] where the Krylov subspace generated by a previous system is recycled for further solves in subsequent systems. An approximation of the eigenvectors is used to augment the space of subsequent systems to deflate the lowest eigenmodes [11]. In this case, the eigenvectors are computed explicitly and not approximated through a coarser discretization of the domain. In this paper, we present the application of the DPCG method to incompressible flows in tubular domains. A subspace deflation technique is used to approximate the lowest eigenmodes (along the vessels) in order to accelerate the convergence.

2. METHODOLOGY

2.1. *Deflated conjugate gradients* (*DCG*)

The DCG technique starts by selecting a subspace $\mathbb{W} \in \mathbb{R}^n$, which is called the *deflation subspace*. Recall that the standard CG algorithm generates a sequence of search directions that are A-orthogonal (or *conjugate*) to all previous search directions. This motivates the following step that consists of writing the solution space as a direct sum of $\mathbb{W}$ and its A-orthogonal complement

DOI: 10.1002/cnm

($\mathbb{N} = \mathbb{W} \oplus \mathbb{W}^{\perp_A}$). The initial error is then projected onto $\mathbb{W}$ and a CG-type algorithm is applied on $\mathbb{W}^{\perp_A}$ in order to eliminate the remaining components of the error.

Given an initial guess x_0, the initial error projection step is carried out by writing

$$e_0 = W c_0 + e_1 \tag{5}$$

where $e_0 = x_* - x_0$ is the initial error, W is a matrix whose columns are a basis of $\mathbb{W}$ and c_0 is an m-dimensional vector ($m = \dim(\mathbb{W})$). The vector c_0 can be computed by enforcing e_1 to be in $\mathbb{W}^{\perp_A}$, i.e. A-orthogonal to $\mathbb{W}$, which yields

$$W^{\mathrm{T}} A W c_0 = W^{\mathrm{T}} r_0 \tag{6}$$

where $A e_0 = r_0$ is the initial residual. The initial guess is then updated as follows:

$$x_1 = x_0 + W c_0 \tag{7}$$

The remaining components of the error are eliminated by applying a CG algorithm whose search space is restricted to $\mathbb{W}^{\perp_A}$. Recall that from the standard CG algorithm, each subsequent search direction d_{k+1} is computed from the current residual and the previous search direction [6]

$$d_{k+1} = r_{k+1} + \beta_k d_k, \quad \beta_k = \frac{r_{k+1}^{\mathrm{T}} r_{k+1}}{r_k^{\mathrm{T}} r_k} \tag{8}$$

where the parameter β_k is obtained by enforcing the conjugacy between the subsequent search directions ($(d_{k+1}, d_k)_A = 0$). The deflated CG adds one more conjugacy condition to enforce the search directions to be in $\mathbb{W}^{\perp_A}$

$$d_{k+1} = r_{k+1} + \beta_k d_k - W c_{k+1} \tag{9}$$

where the parameter β_k is obtained as before and the m-dimensional vector c_k is computed by enforcing the search directions to be A-orthogonal with $\mathbb{W}$, which yields:

$$W^{\mathrm{T}} A W c_{k+1} = W^{\mathrm{T}} A r_{k+1} \tag{10}$$

Preconditioning can be easily included in the DCG algorithm by simply replacing the usual Euclidean inner product with the M-inner product, where M is an SPD preconditioner matrix [6].

The deflated preconditioned CG algorithm reads as follows:

Algorithm 1 Deflated Preconditioned Conjugate Gradients (DPCG)

Given A, b, W, M and an initial guess x_0
Compute $r_0 := b - A x_0$
Solve $W^{\mathrm{T}} A W c_0 = W^{\mathrm{T}} r_0$
Set $x_1 := x_0 + W c_0$
Compute $r_1 := b - A x_1$
Solve $M z_1 = r_1$
Solve $W^{\mathrm{T}} A W c_1 = W^{\mathrm{T}} A z_1$
Set $d_1 := z_1 - W c_1$
do until convergence
  $\alpha_k := (r_k, z_k)/(d_k, A d_k)$
  $x_{k+1} := x_k + \alpha_k d_k$
  $r_{k+1} := r_k - \alpha_k A d_k$
  $M z_{k+1} = r_{k+1}$
  $\beta_k := (r_{k+1}, z_{k+1})/(r_k, z_k)$
  $W^{\mathrm{T}} A W c_{k+1} = W^{\mathrm{T}} A z_{k+1}$
  $d_{k+1} := z_{k+1} + \beta_k d_k - W c_{k+1}$
end do

DOI: 10.1002/cnm

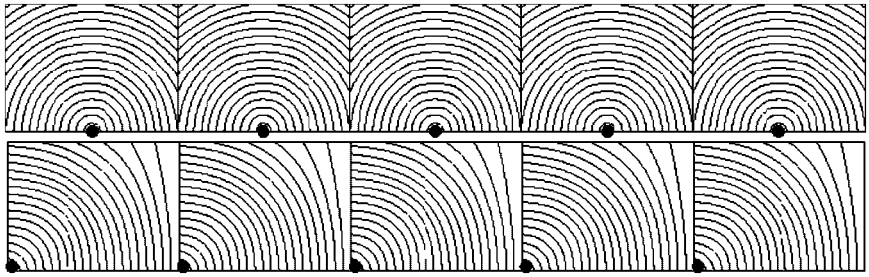

Figure 1. Schematics of the SA (top) and the ALM (bottom) applied to a 2D pipe.

2.2. Subdomain deflation

The DCG technique requires the definition of a deflation subspace $\mathbb{W}$. For matrices arising from a discretization of the domain (e.g. finite elements), the subdomain deflation approach described in [8] defines a deflation subspace based on a coarser discretization of the domain. This technique attempts to spatially approximate the smallest eigenmodes of the underlying system so that they can be removed from the search space of the conjugate gradients method.

As the lowest eigenmodes of the Laplace operator in elongated or tubular domains are oriented along the domain longest direction, a good strategy for approximating these eigenmodes would be a coarser discretization of the domain along this longest direction. One of the simplest ways of defining a coarser discretization from a given unstructured mesh is to agglomerate the nodes of the mesh into subdomains. Two alternative methods have been developed following this strategy:

- *Seedpoints Alternative* (*SA*): For this (manual) technique, the user defines an arbitrary set of points, called *seedpoints*. Given a mesh, the closest mesh points to the seedpoints are found, and a region number is assigned accordingly. Points not assigned to any region are then added one layer at a time until all points have been assigned a region number.
- *Advancing Layers Method* (*ALM*): Starting from a prescribed point, neighboring points are added one layer at a time, until a specified number of points per region is exceeded. The last set of points added is then used as a starting point for the next group. The procedure is repeated until all points have been assigned a region number.

Figure 1 shows a schematics of the SA (top) and the ALM (bottom) applied to a 2D pipe. The contours in this figure represent layers of elements being added to each deflation group. For the SA the deflation groups are grown in parallel starting from the seed points (black dots), while for the ALM the deflation groups are build sequentially from left to right, where the start points (black dots) are selected automatically from the last layer of the last built group.

The last step is to define an interpolant polynomial over the deflation regions. The simplest choice for this is a constant function. Using this approach, each column of the matrix W will represent one deflation subdomain. For a given column, a unity value will be assigned if a point belongs to that region, and zero otherwise.

$$W_{ij} = \begin{cases} 1, & p_i \in g_j \\ 0, & p_i \notin g_j \end{cases} \tag{11}$$

3. EXAMPLES

The DPCG was tested on an idealized pipe flow model and three patient-specific models of cerebral arteries and aneurysms are constructed from medical images using a previously developed methodology [12]. For all the cases a simple diagonal preconditioned was used. It was also verified that the results obtained using the deflated PCG solver coincide with those obtained by the standard PCG solver.

DOI: 10.1002/cnm

For the hemodynamic simulations, the blood was considered as a Newtonian fluid, which is modeled by the *unsteady* incompressible Navier–Stokes equations. Pulsatile flow boundary conditions were computed using a Womersley model with measured flow rates of a normal subject as described in [12]. The simulations were performed using implicit time stepping, solving a pseudo-steady problem at each time step [13]. Within each pseudo-time step the advective terms were integrated implicitly using 5 LU-SGS passes (local Courant number $C=5.0$), followed by the pressure projection. The time step was set to $\Delta t=0.01$ s. The material properties of the fluid were taken to be $\rho=1.0\,\mathrm{g/cm^3}$ and $\mu=0.04$ Poise. Two cardiac cycles were computed with 100 time steps per cycle.

The aim of these examples was the comparison of the number of iterations of the pressure Poisson solver with the speed of the flow calculations. An Intel Xeon processor (E5345/2.33 MHz) computer with 16 GB of RAM was used in all the examples.

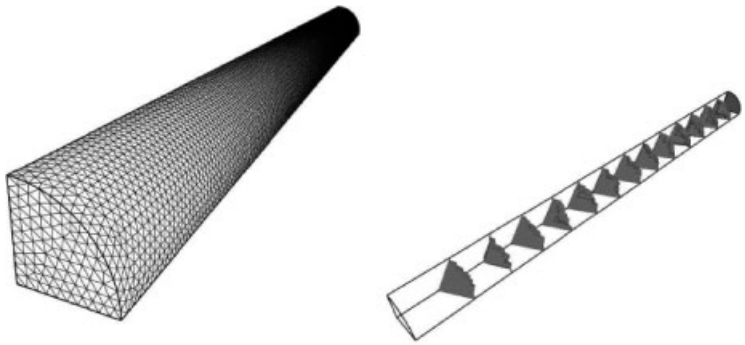

Figure 2. Pipe Flow: surface mesh and deflation domain boundaries for $l=20$.

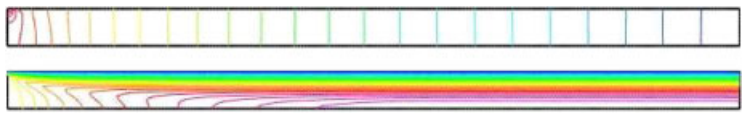

Figure 3. Pipe Flow: pressure and Abs(velocity) for plane $z=0$.

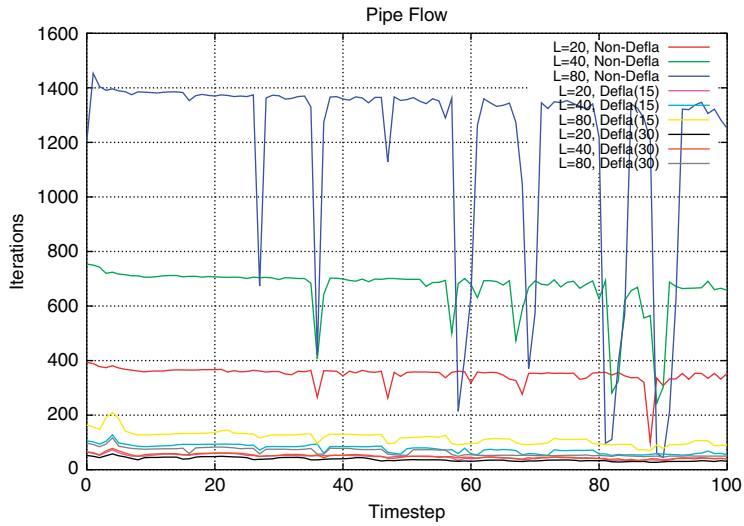

Figure 4. Pipe Flow: number of iterations required for the PCG solver at each time step.

DOI: 10.1002/cnm

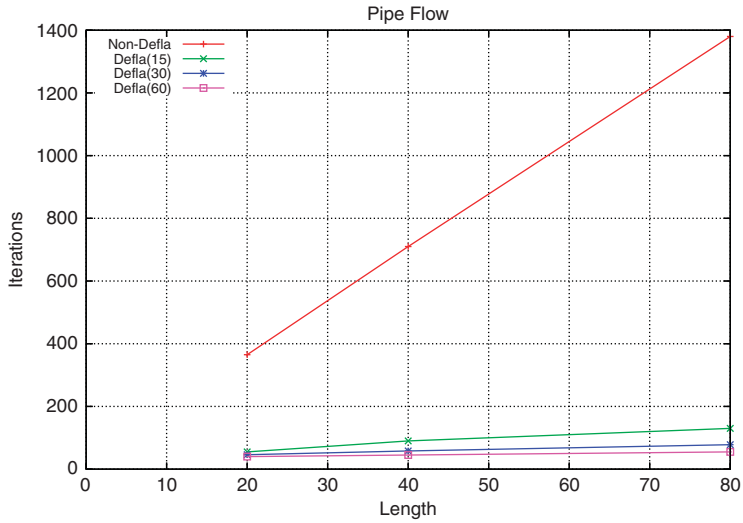

Figure 5. Pipe Flow: average number of iterations required for the PCG solver as a function of the pipe length.

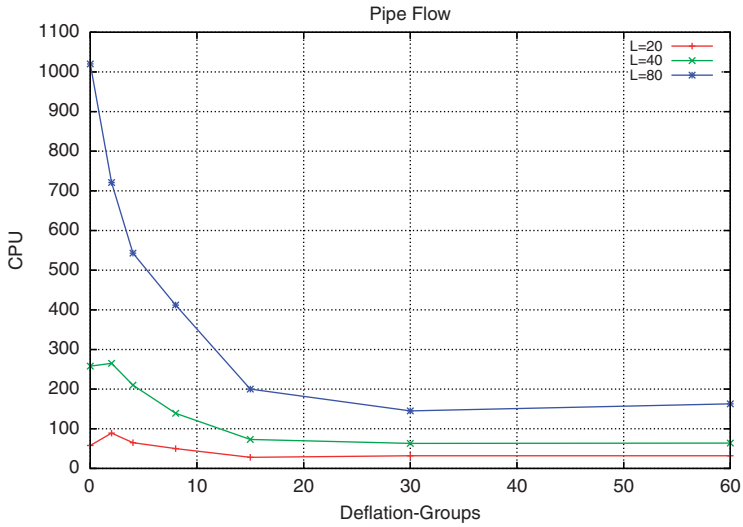

Figure 6. Pipe Flow: total CPU time required for 100 time steps.

3.1. Pipe flow

The first example is the classic Poiseuille pipe flow, a steady flow of a viscous Newtonian fluid in a straight circular domain. A uniform velocity profile is prescribed at the inflow, while a constant pressure is prescribed at the outflow. As the pressure field has to establish itself along the pipe, the number of iterations required increases with the graph depth of the finite element mesh. The physical dimensions and parameters were set as follows:

- pipe radius: $r=1$,
- pipe length: $l=20$, 40, 80,
- density: $\rho=1$,
- inflow velocity: $v=1$,
- viscosity: $\mu=0.01$.

The element size was set to $h=0.1$, implying approximately a graph depth of 200, 400 and 800 for the cases considered. This resulted in grids of 129, 260 and 516 Kels, respectively. All cases

DOI: 10.1002/cnm

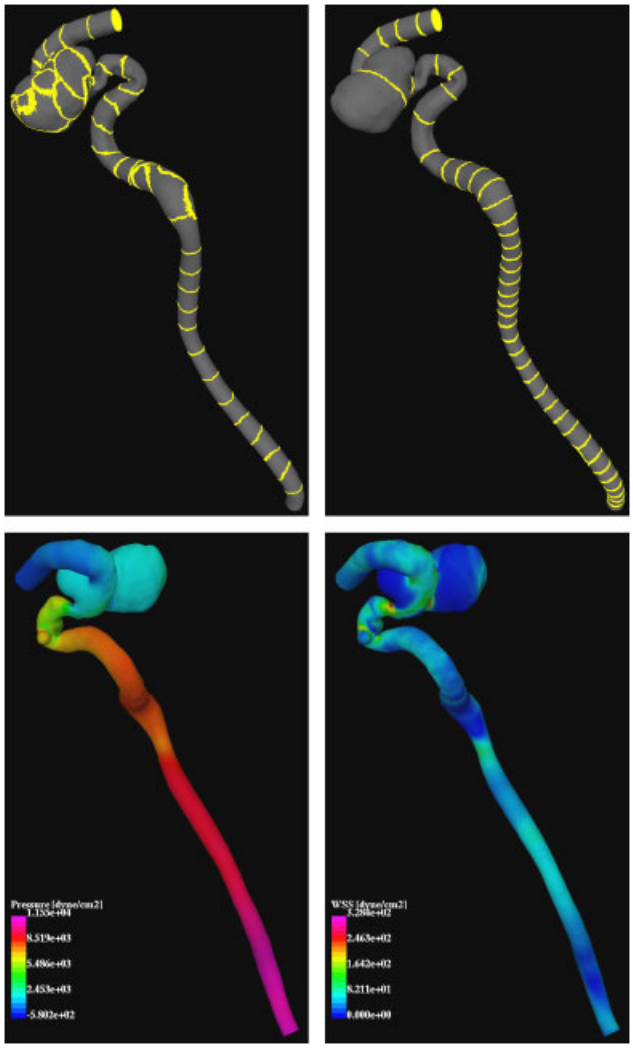

Figure 7. ICA Aneurysm. Top: deflation boundary subdomains for 50 groups using ALM (left) and 45 groups using the SA (right). Bottom: pressure (left) and wall shear stress distribution (right) at peak systole.

were run for 100 time steps using explicit time stepping. The number of groups was chosen to be 15, 30 and 60. The deflation subdomains were generated by the advancing layers technique, starting from the exit. Figures 2 and 3 show the surface mesh, deflation subdomain boundaries for 15 groups, pressure and absolute value of the velocity for $l = 20$. As expected, the velocity profile at the outlet is parabolic, and the pressure field is linear along the cylinder axis after an initial development distance.

Figure 4 shows the number of iterations required for the PCG solver. The sudden 'dips' in the number of iterations at some time steps are due to the fact that a projective prediction of pressure increments with 2 Krylov vectors [14] was used. Note the dramatic decrease in the number of iterations achieved by the deflated PCG solver. This decrease may also be seen in Figure 5, which depicts the average number of iterations for the first 20 steps for the different options chosen.

While the number of iterations increases linearly with the pipe length for the conventional PCG, the performance of the deflated PCG solver seems to be insensitive to the pipe length and the number of groups chosen. Figure 6 shows the total CPU time required for the simulation, highlighting the importance of a fast pressure Poisson solver. Note that for the case $l = 80$, the deflated PCG case performs seven times faster.

DOI: 10.1002/cnm

3.2. Internal carotid artery aneurysm

The second example is a patient-specific model of a cerebral aneurysm located at the internal carotid artery (ICA). This case is particularly interesting as it was possible to reconstruct the entire ICA from the carotid bifurcation to the carotid terminus. The vascular model was reconstructed from 3DRA images.

The volume mesh obtained for this model had 646 Kpts and 3.6 Mels. The deflation groups were generated using both methods. For the SA 45 points were manually selected from the surface model. For the ALM 20, 50, 100 and 150 groups were automatically generated. Figure 7 shows the overall domain, as well as the group boundaries for the 45 (top left) and 50 (top right) groups cases.

The pressure was prescribed (homogeneous bc) at the outflow boundary at the ICA terminal (top part), while a time-dependent velocity profile was prescribed at the inflow boundary at the origin of the ICA (bottom part). No-slip boundary conditions were applied at the vessel wall.

Figure 7 depicts the pressure drop (bottom left) and the wall shear stress distribution (bottom right) at the peak systole (the inflow rate peak) of the second cardiac cycle. As it was expected, the pressure gradient is aligned with the parent vessel, while the pressure is almost constant inside the aneurysm.

Figure 8 shows the average number of iterations per time step (top) for the pressure Poisson solver and the total CPU time (bottom) needed for completing the 200 time steps. In this case, the

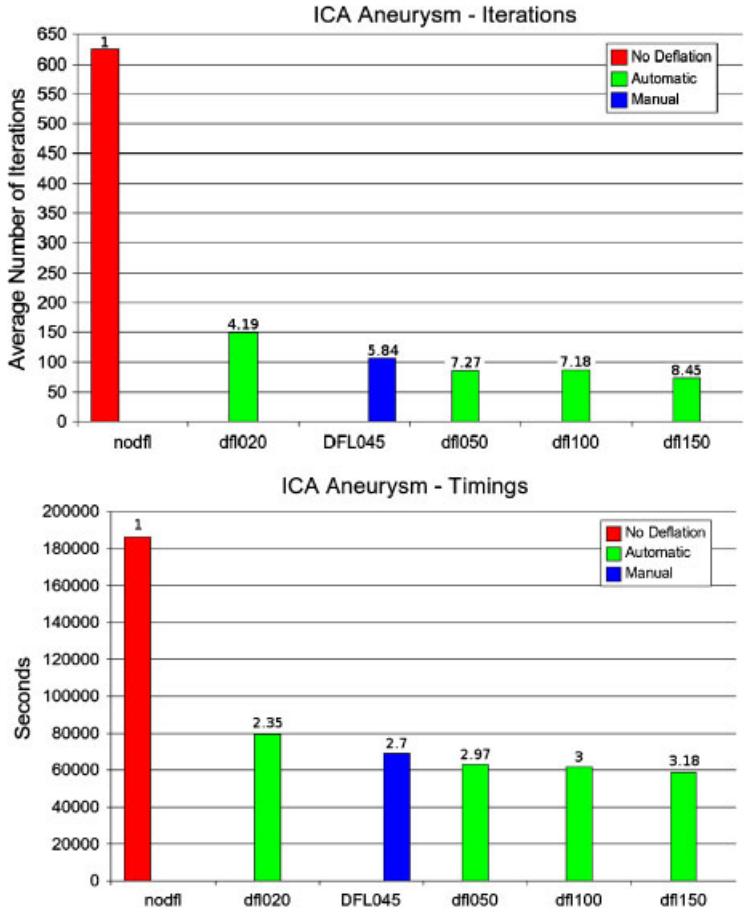

Figure 8. ICA aneurysm: average number of iterations (top) per time step for the pressure Poisson solver and total CPU time (bottom).

DOI: 10.1002/cnm

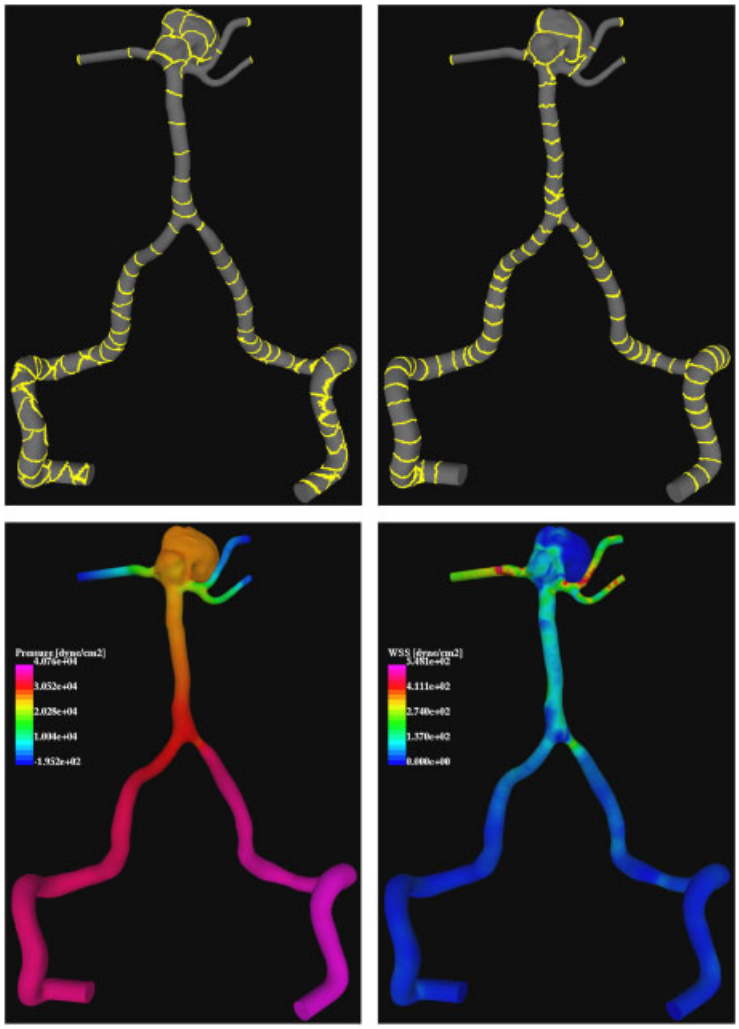

Figure 9. BT aneurysm. Top: deflation boundary subdomains for 100 groups using the ALM (left) and 90 groups using the SA (right). Bottom: pressure (left) and wall shear stress distribution (right) at peak systole.

number of iterations was reduced by up to 8.5 times while the CPU time was reduced by up to 3.2 times. This result corresponds to the case with 150 groups using the ALM.

3.3. Basilar tip aneurysm

The third example is a patient-specific model of a cerebral aneurysm located at the tip of the basilar artery constructed, as in the previous example, from 3DRA images. The model included in the upstream direction both left and right vertebral arteries. The volume mesh generated for this model had 535 Kpts and 2.9 Mels. The deflation subdomains were generated using both methods. For the SA, 90 points were manually selected from the surface model. For the ALM, 20, 50, 100 and 150 groups were automatically generated. Figure 9 shows the overall domain, as well as the deflation group boundaries for the 90 and 100 groups cases.

The pressure was prescribed (homogeneous bc) at the three outflows (top part), while a time-dependent velocity profile was prescribed at the two inflows (bottom part). No-slip boundary conditions were applied at the vessel wall.

DOI: 10.1002/cnm

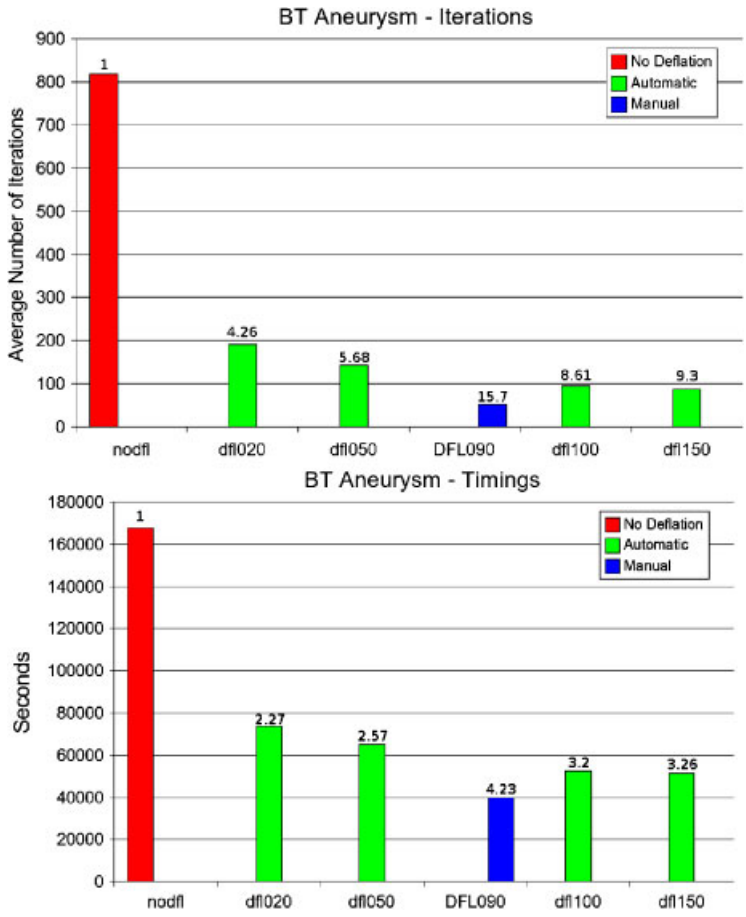

Figure 10. BT aneurysm: average number of iterations (top) per time step for the pressure Poisson solver and total CPU time (bottom).

Figure 9 depicts the pressure (left) and the wall shear stress distribution (right) at the peak systole of the second cardiac cycle. As it was expected, the pressure changes along the vessels, while it is almost constant inside the aneurysm.

Figure 10 shows the average number of iterations per time step (top) for the pressure Poisson solver and the total CPU time (bottom) required to compute 200 time steps. The highest reduction in the number of iterations was about 16 times while the total CPU time was reduced by up to a factor of 4.2 times. Notice that in this case, the best result was obtained with 90 groups using the SA.

3.4. Circle of Willis

The last example is a subject-specific model of the circle of Willis of a normal volunteer constructed from magnetic resonance images. The volume mesh generated for this model had approximately 836 Kpts and 4.6 Mels. The deflation groups were generated using both methods. For the SA 131 points were manually selected from the surface of the model, while for the ALM 20, 50, 100 and 150 groups were automatically generated. Figure 11 shows the overall domain, as well as the deflation subdomain boundaries for the 131 and 150 groups cases.

The pressure was prescribed (homogeneous bc) at the six outflows, while a time-dependent velocity profile was prescribed at the three inflows (bottom part). No-slip boundary conditions were applied at the vessel wall.

DOI: 10.1002/cnm

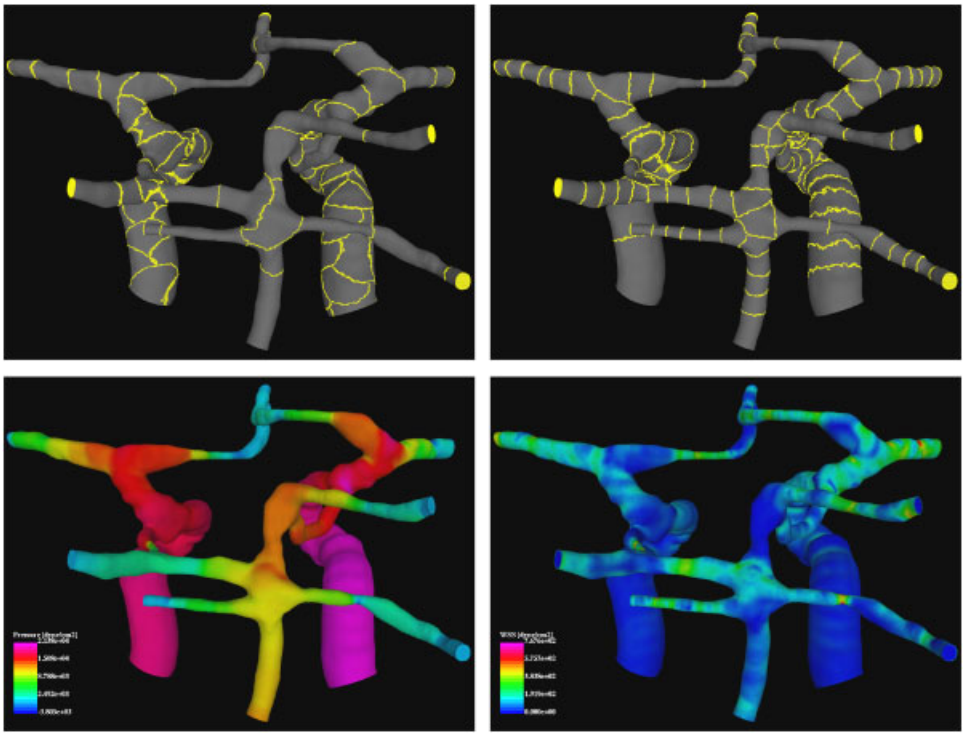

Figure 11. Circle of Willis. Top: deflation boundary subdomains for 100 groups using the ALM (left) and 131 groups using the SA (right). Bottom: pressure (left) and wall shear stress distribution (right) at peak systole.

Figure 11 depicts the pressure (left) and the wall shear stress distribution (right) at the peak systole of the second cardiac cycle. As it was expected, the pressure changes slowly on the large vessels when compared with the small vessels.

Figure 12 shows the average number of iterations per time step (top) for the pressure Poisson solver and the total CPU time (bottom). The highest reduction in the number of iterations was about 10 times while the total CPU time was reduced by up to a factor of 3.9 times. Notice that as in the previous case, the best result was obtained using the SA.

4. CONCLUSIONS

A deflated preconditioned conjugate gradients (DPCG) technique has been developed for the pressure Poisson equation within an incompressible flow solver. A simple idealized pipe flow model and three patient-specific image-based blood flow computations were carried out to assess the performance of the deflated PCG solver. The number of iterations was significantly reduced (up to a factor of 16) while speedups of about 4 times were obtained in the solution of the incompressible flow equations.

Automatic (advancing layers) and manual (seedpoints) methods for constructing the deflation subdomains in elongated or tubular domains have been developed. These methods resulted in similar performances. However, in some cases, the manual method outperformed the automatic one, possibly because it produced more regular groups with more uniform pressure distributions.

In conclusion, this methodology has improved substantially the performance of the incompressible flow solver especially for vascular domains, making larger and more complex hemodynamic

DOI: 10.1002/cnm

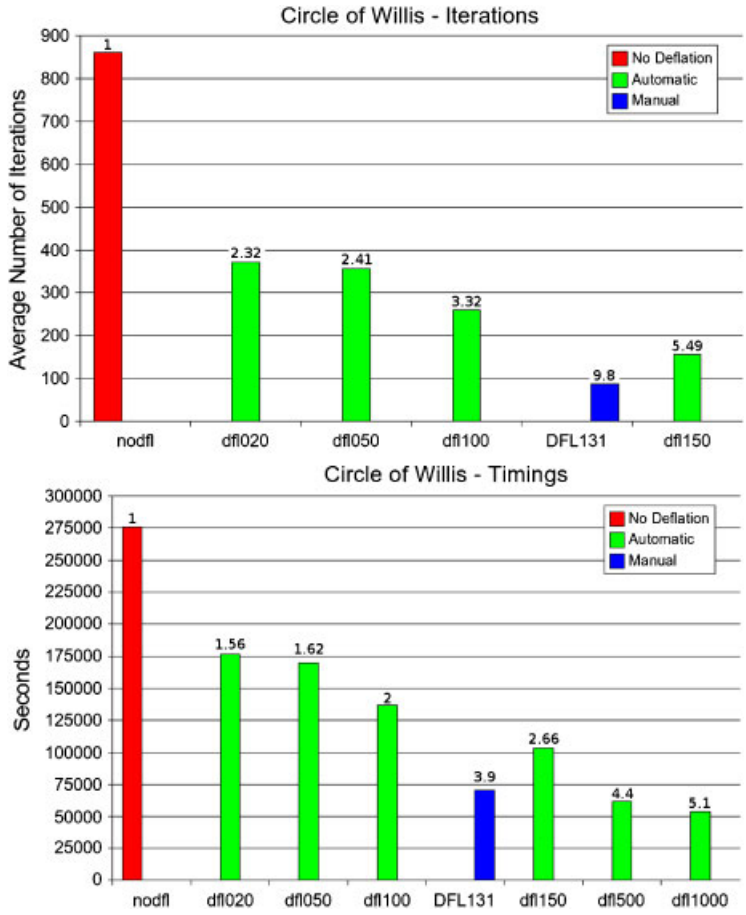

Figure 12. Circle of Willis: average number of iterations (top) per time step for the pressure Poisson solver and total CPU time (bottom).

models practical. Although the methodology was developed in the context of blood flow simulations, it is general and can be used for any incompressible flow calculation.

ACKNOWLEDGEMENTS

We thank Philips Medical Systems for financial support.

REFERENCES

1. Steinman DA, Taylor CA. Flow imaging and computing: large artery hemodynamics. *Annals of Biomedical Engineering* 2005; **33**(12):1704–1709.
2. Thomaset F. *Implementation of Finite Element Methods for Navier–Stokes Equations*. Springer: Berlin, 1981.
3. Gunsburger MD, Nicolaides R. *Incompressible Computational Fluid Dynamics*: *Trends and Advances*. Cambridge University Press: Cambridge, 1993.
4. Hafez MM. *Numerical Simulation of Incompressible Flows*. World Scientific: Singapore, 2003.
5. Lohner R. *Applied CFD Techniques*: *An Introduction Based on Finite Element Methods* (2nd edn). Wiley: New York, 2008.
6. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, 2003.
7. van der Sluis A, van der Vorst HA. The rate of convergence of conjugate gradients. *Numerische Mathematik* 1986; **48**:543–560.
8. Nicolaides RA. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis* 1987; **24**(2):355–365.
9. Mansfield L. On the use of deflation to improve the convergence of the conjugate gradient iteration. *Communications in Applied Numerical Methods* 1988; **4**:151–156.

DOI: 10.1002/cnm

10. Erhel J, Burrage K, Pohl B. Restarted gmres preconditioned by deflation. *Journal of Computational and Applied Mathematics* 1996; **69**:303–318.
11. Saad Y, Yeung M, Erhel J, Guyomarc'h F. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing* 2000; **21**(5):1909–1926.
12. Cebral JR, Castro MA, Appanaboyina S, Putman CM, Millán D, Frangi AF. Efficient pipeline for image-based patient-specific analysis of cerebral aneurysm hemodynamics: technique and sensitivity. *IEEE Transactions in Medical Imaging* 2005; **24**(1):457–467.
13. Löhner R, Yang C, Cebral JR, Camelli F, Soto O, Waltz J. Improving the speed and accuracy of projection-type incompressible flow solvers. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(23–24): 3087–3109.
14. Löhner R. Projective prediction of pressure increments. *Communications in Numerical Methods in Engineering* 2005; **21**(4):201–207.

DOI: 10.1002/cnm