V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal, 14–17 June 2010

ACCELERATION OF CFD COMPUTATIONS THROUGH A SUBSPACE DECOMPOSITION METHOD

George Pashos*, Nikolaos Cheimarios, Eleni D. Koronaki and Andreas G. Boudouvis

* School of Chemical Engineering, National Technical University of Athens
Athens 15780, Greece
e-mail: gpashos@chemeng.ntua.gr
e-mail: nixeimar@chemeng.ntua.gr
e-mail: ekor@mail.ntua.gr
e-mail: boudouvi@chemeng.ntua.gr

Key words: GMRES, RPM, FLUENT, acceleration, lid-driven cavity

Abstract. *A computational shell-like method is proposed, based on the Recursive Projection Method (RPM) in order to accelerate large-scale CFD computations. The method is wrapped around a home-made Galerkin/finite element code and also the commercial finite volume code FLUENT. For both cases, it does not require any intervention on the CFD codes. The RPM decomposes the solution subspace into two subspaces, the one that contains the critical eigenspace and its orthogonal complement. The former subspace contains those eigenvalues that are about to cross the unit circle thus signaling instabilities associated with change of stability of the computed solutions or with failures of the numerical scheme that slow down the convergence process; it is treated separately, which results in acceleration of the numerical scheme, whether it is FLUENT or the finite element code. The latter subspace contains the well-behaved eigenvalues that are closer to the center of the unit circle and pose no 'threat' to the numerical stability of the scheme; this subspace requires no special treatment. A welcome by-product of the proposed method is the extraction of useful information regarding the critical eigenspace that can trace the flow instabilities. This is particularly interesting in the case of the commercial package since eigenvalues and eigenvectors are not provided by FLUENT. The case study is the 2D lid-driven cavity, a standard benchmark problem for CFD codes that has recently rekindled the scientific interest on the aspect of tracing its instabilities. The RPM manages to accelerate the convergence process of FLUENT, i.e. it reduces the number of total iterations required to converge. The RPM wrapped around the finite element code, besides providing acceleration, enables convergence on an otherwise stagnant scheme (Newton/GMRES). The computations were performed on a 16-node computational cluster.*

1 INTRODUCTION

In recent years, the development of CFD codes is unfolding in multiple levels in order to accommodate the need to simulate transport phenomena in complex geometries, multiple scales, turbulent and supersonic flows and several simultaneous chemical reactions. Each one of these levels are more relevant to one industry than another, leading thus inevitably to a plethora of codes: some are general-scope commercial packages and others are "legacy" codes developed over the years in a particular industry or university focusing on a particular type of application. Whichever the case, these codes represent a serious investment in time and money from the viewpoint of developers and also end users who dedicate resources to purchase and train in a particular software. Nevertheless, the particular needs of an end user may come to include additional features such as acceleration or nonlinear features that are not readily available by the computational package. It is often the case that the user has limited or non-existent access to the source code in order to make the desired alterations and more importantly this could prove highly inefficient. It therefore becomes clear that a framework should exist that enables the expansion of the current capabilities of a given legacy code without any intervention in the code itself.

This is essentially the scope of this work, which focuses on accelerating legacy codes and also extracting useful nonlinear information. The legacy codes in question include a "homemade" finite element code for the Navier-Stokes equations and also a wide-spread and established computational package, FLUENT. The test problem was the same in both cases, the well known lid-driven cavity problem which has recently reignited interest because of its often ambivalent solution space.

The so-called Recursive Projection Method (henceforth RPM) is "wrapped" around the CFD codes as a computational shell in order to accelerate, or even enable convergence in some cases as will be discussed in subsequent sections. The RPM was introduced by G. Shroff and H. Keller [1] for the stabilization of time-steppers. In this work it is implemented on steady state codes which are reformulated in order to assume the form

$$U^{(k+1)} = F(U^{(k)}, \lambda) \tag{1}$$

Here λ is a parameter of the physical system under investigation and $U^{(k+1)} \in \mathbf{R}^N$ is the N-dimensional vector that results from a discretization in space at the k+1 iteration of the unknown quantities. $F: \mathbf{R}^N \times \mathbf{R} \rightarrow \mathbf{R}^N$ is a function of $U^{(k)}$ and it computes an updated approximation of the solution. The convergence rate of the fixed point iteration (1) deteriorates when a few eigenvalues of the Jacobian matrix $F_U \in \mathbf{R}^{N \times N}$ approach the limits of the unit circle. The RPM requires consecutive iterates of F in order to approximate this "dangerous" eigenspace, separate it from the "healthy" part of the spectrum that is close to the origin and treat it with Newton's method using a low dimensional Jacobian.

The RPM, which will be discussed in a subsequent section and in greater detail in [1] has been used in conjunction with fixed-point iterative procedures resulting from time integrators in order to accelerate stable and even mildly unstable schemes [2]. In the thesis of H. Von Sosen [3] the RPM is applied to DAEs in incompressible flow computations. P. Love [4] applied the RPM to Kolmogorov and Taylor-vortex flows. The RPM has also been used successfully for detection of stable and unstable periodic orbits of large-scale dynamical systems [5] as well as of periodically forced systems [6]. Furthermore, a RPM-based method was proposed, in order to perform 'coarse' stability analysis using microscopic evolution rules directly [7, 8, 9]. Recently the RPM has been used to accelerate a Newton/GMRES(m) scheme [10] and this philosophy will also be

demonstrated in this work in comparison to the application of the method around the commercial CFD package FLUENT.

Finally it is worth noting that in the current version of FLUENT (ANSYS 12) the RPM is implemented as an option for the stabilization of the solution of the linear system in the Multigrid solver. Nevertheless the proposed approach treats FLUENT as a "black box" and does not intervene in any internal solver; in fact it may be implemented in conjunction with the RPM in the Multigrid solver. In this work the RPM is "wrapped" around a parallel implementation of FLUENT for increased acceleration.

2 THE RECURSIVE PROJECTION METHOD

The solution of parameter-dependent, nonlinear steady state problems often leads to recursive fixed-point procedures of the form of Eq. (1). In this work, the role of $F \in \mathbf{R}^N$ is assumed by two different solvers: in the first case, it is the result of the Newton iteration that yields the solution approximation of the nonlinear system at the (k+1)th iteration; in this case F is provided by a home-made code of the Galerkin/finite element discretization of the Navier-Stokes equation for the lid-driven cavity problem. In the second case the same physical problem is solved with the finite volume code FLUENT and F is also an updated approximation of the solution. The convergence properties of iteration (1) are determined by the spectrum of the Jacobian matrix $F_U \in \mathbf{R}^{N\times N}$ and in particular the usually few eigenvalues that reach and eventually cross the limits of the unit circle. Due to the large number of degrees of freedom, N, it is not efficient to compute and store F_U but with the help of the RPM it is possible to approximate the part of the spectrum that is responsible for the deterioration of the convergence rate. Here we discuss briefly how this is done. The method is described in detail by Shroff and Keller [1] and in a family of related 'Newton–Picard' methods in several theses (e.g. [3]). Along the same line is earlier work by Jarausch and Mackens [11] who propose a so-called 'adaptive condensation' method for symmetric systems.

Consider the invariant subspace **P**, that corresponds to the, usually few, eigendirections in which the linearized map is slowly contracting or even slowly expanding. Let **Q** be its orthogonal complement. By P and Q we also denote the orthogonal projectors of $\mathbf{R}^N$ on **P** and **Q** respectively. The solution U is decomposed into p and q such that U = p +q, where p and q are the projections of F (U,λ) onto P and Q , respectively. Under certain assumptions, the RPM stabilizes fixed-point iterative procedures such as (1) by first computing an approximation of **P** and consequently of **Q**. The projection p is then computed by performing a Newton's method step on P; q is determined by the projection of F (U,λ) on Q. Subsequently, the Picard iteration acts on the sum, U = p +q. The fixed-point iteration is stable when all the eigenvalues of the matrix F_U (U,λ) ≡ ∂ F /∂U, lie in the unit disk. The stabilized iterative procedure takes the form of Algorithm 1.

Algorithm 1

(i) $p^{(0)} = PU^{(0)}$, $q^{(0)} = Q\,U^{(0)}$

(ii) Do until convergence:

$$p^{(k+1)} = p^{(k)} + (I - P\,F_U(U^{(k)},\lambda)P)^{-1}(P\,F_U(U^{(k)},\lambda) - p^{(k)}) \text{ (Newton step)}$$
$$q^{(k+1)} = QF(U^{(k)},\lambda) \text{ (fixed-point iteration).}$$

(iii) $U^{(kfinal)} = p^{(kfinal)} + q^{(kfinal)}$.

In practice after a prescribed number of outer — inexact Newton, or FLUENT — iterations is performed and convergence is deemed too slow, the RPM intervenes; every k_{max} iterations the RPM uses the last $n + 1$ iterates in order to compute, using the modified Gram–Schmidt algorithm, an orthonormal basis of the subspace that is spanned by the sequence of n vectors

$$\{\Delta q^{(i)}\}\equiv\{(q^{(kmax)}-q^{(kmax-1)}),\ldots,(q^{(kmax-n+1)}-q^{(kmax-n)})\},\ i=k_{max}-n+1,\ldots,k_{max} \quad (2)$$

It is shown in the literature [4,26] that the above vectors asymptotically lie in the dominant eigenspace of the matrix QF_UQ which is the eigenspace of F_U that corresponds to the remaining dominant eigenvalues (i.e. the eigenvalues that are not yet included in the subspace **P**). Let l_Z be the number of eigenvalues close to the unit circle and the dimension of **P**. If $Z \in \mathbf{R}^{N\times l_Z}$ is an orthonormal basis of the invariant subspace **P**, then with the above method it is updated with new vectors.

We will briefly discuss how this is achieved (for more information cf. [1]). For $n = 2$, the set of difference vectors, $\{\Delta q^{(kmax)},\Delta q^{(kmax-1)}\}$ is formed from the three last iterates of q. The modified Gram–Schmidt procedure [13] is applied to construct a QR-factorization

$$D\equiv\{\Delta q^{(kmax)},\Delta q^{(kmax-1)}\}=\hat{D}T \quad (3)$$

with $T \in \mathbf{R}^{n\times n}$ upper triangular and $\hat{D} \in \mathbf{R}^{N\times n}$ orthogonal. If $T_{11}>>T_{22}$ then one new vector ($n_{def} = 1$) is added to the basis Z; otherwise, it is possible that a pair of complex eigenvalues approach the limit of stability and two new vectors ($n_{def} = 2$) are added in Z. The effectiveness of the RPM depends greatly on the assumption that the majority of eigenvalues of F_U are clustered together whereas only a few eigenvalues (typically no more that 10 at worst) increase in magnitude and approach the unit circle. In this work the criterion, $T_{11} > 10^3 T_{22}$ is used that indicates the relative magnitude of the two eigenvalues captured by the method and is deemed an adequate measure of the separation of scales assumed by the RPM method. The factor 10^3 depends on the spectrum of F_U, but it has been known to work adequately for a broad range of applications [2]. In practice, the Jacobian of the Newton step performed on the reduced-size subspace **P**, can be written

$$H \equiv Z^T F_U Z, \in \mathbf{R}^{l_Z\times l_Z} \quad (4)$$

The matrix-vector product F_UZ, is computed with differencing

$$F_U Z_i = \frac{F(U+\varepsilon Z_i,\lambda)-F(U,\lambda)}{\varepsilon}, i=1,2,\ldots,l_Z \quad (5)$$

where l_Z is the number of columns of the basis Z. As a result, for each matrix–vector computation only an extra function estimation, i.e. the calculation of $F(U+\varepsilon Z_i,\lambda)$ is required and not the creation of the full Jacobian (the extra function estimation initiates a full cycle of inner iterations). Avoiding, therefore, the construction of large matrices and the solution of large eigenvalue problems, it is possible to draw conclusions for the stability of F by solving the small simple eigenvalue problem of the matrix H (cf. [1,2]), the spectrum of which approximates the dominant spectrum of F_U.

3 THE LID-DRIVEN CAVITY PROBLEM

The test application in this work is the lid-driven cavity problem which is described by the Navier-Stokes equations combined with the continuity equation

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \frac{1}{Re} \nabla \cdot (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \tag{6}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{7}$$

where $\mathbf{u}$ is the fluid velocity vector in two dimensions, $\mathbf{u} = (u, v)$, P is the static pressure and Re is the Reynolds number. Eq. (6) is the dimensionless form of the Navier–Stokes in stress-divergence form with $Re = \rho u_0 L/\mu$, which is more suitable for computations in the large Re regime [12]. Eqs. (6)–(7) are posed in the unit square domain, $\Omega = [0, 1] \times [0, 1]$, with no-slip Dirichlet boundary conditions on the boundaries $\Gamma_1, \Gamma_2, \Gamma_3$, $\mathbf{u}|_{\Gamma 1,\Gamma 2,\Gamma 3} = \mathbf{0}$ and a Dirichlet moving boundary condition on Γ_0, $\mathbf{u}|_{\Gamma 0} = (0, 1)$; $\Gamma_0 = [0] \times [0, 1]$, $\Gamma_1 = [0, 1] \times [1]$, $\Gamma_2 = [1] \times [0, 1]$, $\Gamma_3 = [0, 1] \times [0]$.

4 THE NEWTON/GMRES SCHEME

The boundary value problem (6)–(7) along with the boundary conditions is treated with the Galerkin/finite element method and the domain Ω is tessellated into a uniform grid of quadrilateral Q_2Q_1 elements, i.e. each velocity component is approximated by continuous 2nd order piecewise polynomials and the pressure by continuous 1st order piecewise polynomials in each spatial direction. A 40×40 mesh is used that results in N = 14,803. The Galerkin/finite element weak formulation of (6)–(7) in steady state and for Dirichlet-only boundary conditions casts

$$\int_\Omega \left[\phi \left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) + \frac{1}{Re} \left(2 \frac{\partial \phi}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \phi}{\partial y} \frac{\partial u}{\partial y} + \frac{\partial \phi}{\partial y} \frac{\partial v}{\partial x} \right) - P \frac{\partial \phi}{\partial x} \right] d\Omega = 0 \tag{8}$$

$$\int_\Omega \left[\phi \left(u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) + \frac{1}{Re} \left(\frac{\partial \phi}{\partial x} \frac{\partial v}{\partial x} + 2 \frac{\partial \phi}{\partial y} \frac{\partial v}{\partial y} + \frac{\partial \phi}{\partial x} \frac{\partial u}{\partial y} \right) - P \frac{\partial \phi}{\partial y} \right] d\Omega = 0 \tag{9}$$

$$\int_\Omega \psi \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) d\Omega = 0 \tag{10}$$

where ϕ and ψ is a set of piecewise biquadratic basis functions and piecewise bilinear basis functions, respectively.

The above set of nonlinear algebraic equations, can be written compactly

$$R(U) = 0 \tag{11}$$

It is solved iteratively for the vector of unknowns $U \in \mathbf{R}^N$, by Newton's method, which updates at each step, k, the approximation of the solution. The linear system that yields the correction for the new approximation reads

$$J^{(k)} \delta U^{(k)} = -R^{(k)}, \; J \in \mathbf{R}^{N \times N}, R \in \mathbf{R}^N, \delta U \in \mathbf{R}^N \tag{12}$$

$R^{(k)}$ is the value of the residual at the k^{th} iteration, $\delta U^{(k)}$ is the correction of the new approximation of the solution and $J^{(k)} \equiv \partial R^{(k)}/\partial U^{(k)}$ is the Jacobian matrix. As N is usually large, the linear system (12) is solved with an iterative, Krylov-type solver, namely the restarted GMRES, or GMRES(m). In this context, m is maximum dimension of the Krylov basis allowed to be built before the method restarts again.

The new solution approximation, $U^{(k+1)}$, is then computed

$$U^{(k+1)} = U^{(k)} + \delta U^{(k)} \equiv F(U^{(k)}) \tag{13}$$

Since the Newton iteration is an one-step method, that is, the new solution approximation is computed using information from the previous step only, it may be considered as a scheme similar to (1) as illustrated in Eq. (13).

The Newton/GMRES(m) scheme compactly described by Eq. (13) is known to stagnate even for low Reynolds numbers [10, 13]. Several values for the parameter l_{final}=m×restarts are selected and the convergence process is depicted in Fig. 1. Here the Reynolds number is very low, namely Re=10 and the total number of degrees of freedom N=14,803. The convergence rate is generally steep until the residual of the Newton iteration reaches 10^{-4} but subsequently the convergence rate greatly deteriorates. Even for l_{final}=150 with m=30 and restarts=5, the residual drops but not at a satisfactory rate.

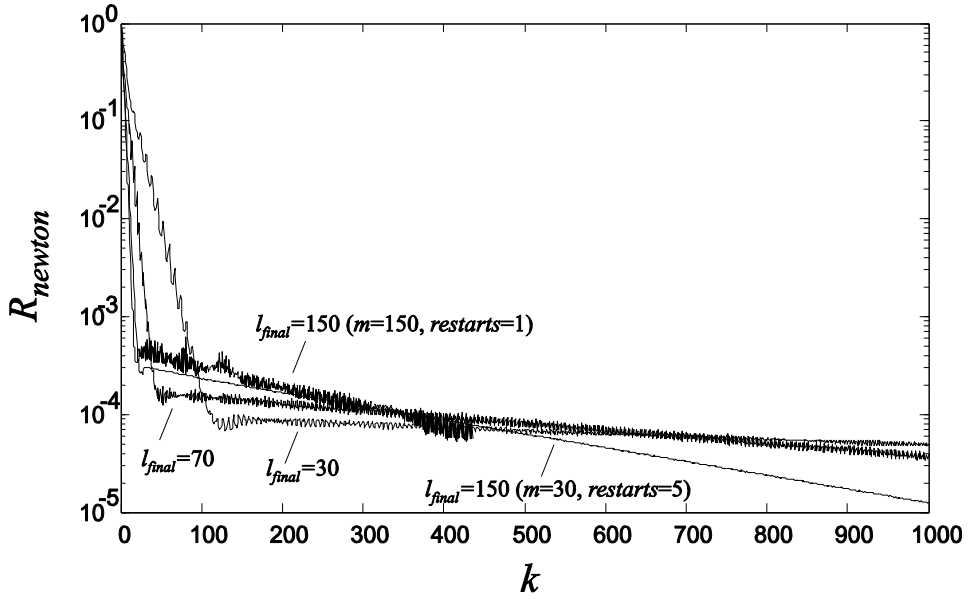

Figure 1: Convergence rates of the Newton iteration of the lid-driven cavity problem with the stand-alone Newton/GMRES; k is the number of Newton iterations; different values of input parameters are: l_{final} = 30 (m = 30, restarts = 1), l_{final} = 70 (m = 70, restarts = 1), l_{final} = 150 (m = 150, restarts = 1), l_{final} = 150 (m = 30, restarts = 5); N = 14803; Re = 10; $U^{(0)}$ = 0.

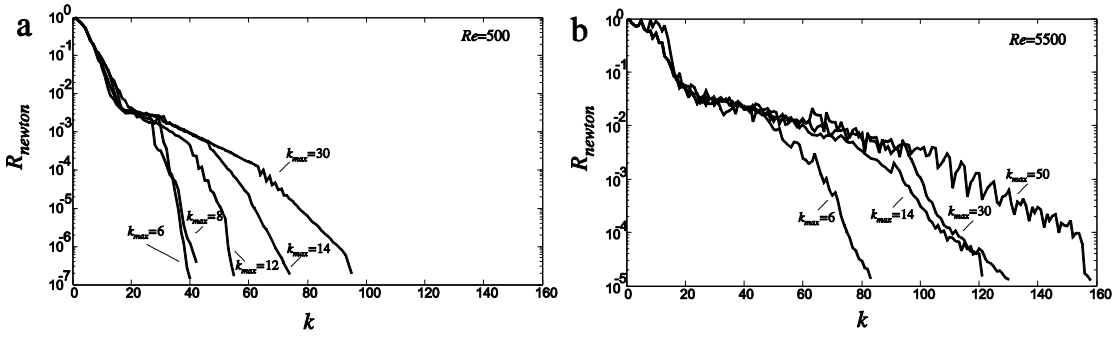

Figure 2: Convergence rates of the outer iteration of the lid-driven cavity problem with the RPM-assisted Newton/GMRES for different values of k_{max}; l_{final} = 250 (m = 50, restarts = 5); $tol_{newton} = 10^{-6}$; N = 14803. (a) Re = 500; $U^{(0)}$ = 0; (b) Re = 5500; $U^{(0)}$(Re = 5500) = $U^{(kfinal)}$(Re = 5000).

In this work (and in more detail in [10]), the RPM is wrapped around the Newton/GMRES scheme resulting from the discretization of the Navier-Stokes

equations in the lid-driven cavity problem. The RPM-assisted Newton/GMRES, with zero initial guess [see Fig. 2 (a)], avoids stagnation even at large values of Re (Re = 5500) [see Fig. 2 (b)]. The parameter that affects the convergence rate is the frequency k_{max} with which Z is updated. By reducing it, the total number of outer iterations, k_{final}, is also reduced as shown in Fig. 2(a) and Figure 2(b). However, frequent updates will also require evaluations of the derivative $F_U Z$ which requires that F is computed an extra l_Z times. In turn this initiates l_Z full cycles of inner iterations. Note that the basis Z, which is initially null, is forced to be expanded by one column each time the maximum number of iterations, k_{max}, is reached. This potentially lowers the quality of the updated basis but keeps the extra F calculations at a minimum.

5 THE CFD CODE FLUENT

The computational fluid dynamics (CFD) problem defined in Section 3 [Eqs. (6)-(7)] is also solved numerically with the commercial CFD code FLUENT [14] where the finite volume method (FVM) [15] is used. The continuity and the components of the momentum equation in steady state for an incompressible fluid, can be written in a general form in terms of a variable Φ as

$$\nabla \cdot (\Phi \mathbf{u}) = \frac{1}{\rho} \nabla \cdot (\Gamma \nabla \Phi) \tag{14}$$

where ρ is the density of the fluid, $\mathbf{u}$ is the velocity vector and Γ the diffusion coefficient. The computational field is discretized in elementary control volumes. For the CFD problem concerned here a 120x120 mesh was used. The key step in the FVM is the integration of Eq. (14) over each elemental control volume (CV). The latter yileds

$$\int_{CV} \nabla \cdot (\Phi \mathbf{u}) dV = \frac{1}{\rho} \int_{CV} \nabla \cdot (\Gamma \nabla \Phi) dV \tag{15}$$

For the solution of Eq. (15) FLUENT's pressure-based solver was used along with the "coupled algorithm" [16]. The "coupled algorithm" solves the momentum and continuity equations simultaneously. The full implicit coupling is achieved through an implicit discretization of pressure gradient terms in the momentum equations, and an implicit discretization of the face mass flux [16]. The "algebraic"' multigrid scheme is used for the solution of the coupled, discretized equations [16].

The RPM, programmed in MATLAB in this work, is used as a computational shell "wrapped around" FLUENT. The communication of the RPM and FLUENT is performed by User Defined Functions (UDFs) [16]. The information can be transferred directly to FLUENT by using libraries, written in C/C++ or Fortran, namely *dlls* on windows operating systems or *so* on linux operating systems.

The computations were performed on a 16-node distributed memory cluster, called Pegasus. Each node of Pegasus has two Intel Xeon processors at 3 GHz and 2 Gb of RAM. The nodes are interconnected with two networks: Myrinet and Gigabit Ethernet. Administration is performed via the Gigabit Ethernet network and the Myrinet network is used only by Message Passing Interface (MPI). MPI is a library specification for the exchange of computational data between the processors, proposed as a standard by a broadly based committee of vendors, implementors and users [17, 18]. MPI can be used from both distributed and shared memory computers. The parallel solver of FLUENT, namely CORTEX [16], uses MPI. Pegasus runs under the freeware operating system Linux for clusters, Rocks 4.1.

6 THE RPM-ASSISTED IMPLEMENTATION OF FLUENT

6.1 Acceleration

The principal goal of combining the RPM with FLUENT is making the overall solution procedure more efficient, i.e. reduce the number of iterations and the solution time. This indeed is achieved with the proposed approach. In this implementation the function evaluation, F(U) of (1), is the result of n_{FLUENT}=80 iterations of the selected FLUENT solver with initial guess U. Therefore, each fixed-point, or outer iteration refers to this prefixed number of FLUENT or, inner iterations. The stand-alone FLUENT code is compared to the RPM-assisted implementation over a range of Reynolds numbers. In each case the initial guess, $U^{(0)}$, is the solution at Re number which is smaller by 1000 that the one sought. The achieved acceleration both in terms of number of function evaluations and speedup, i.e. time of stand-alone iteration over time of RPM-assisted iteration, is summarized in Table 1. The function evaluations performed with the RPM-assisted code, include the extra evaluations needed in order to build and update the basis. In each case the basis required by the RPM consists of 2 basis vectors. The overall speedup ranges from 1.22 to 1.37 leading to an average 25 % reduction in CPU time.

Reynolds number	Function evaluations, stand-alone FLUENT	time (sec), stand-alone FLUENT	Function evaluations with RPM	time (sec), RPM-FLUENT	Speedup
4000	11	315.8	8	229.7	1.37
5000	11	315.8	9	257.8	1.22
6000	14	373.1	10	286.4	1.30
7000	15	401.7	11	315.1	1.27

Table 1: Effect of RPM on the total number of function evaluations and CPU time over a range of Re numbers.

Reynolds number	Function evaluations, F(U)	time (sec)	speedup
4000	8	229.7	1.37
5000	4	117.7	2.68
6000	4	117.7	3.17
7000	5	145.7	2.76

Table 2: RPM-assisted parameter continuation; time and speedup in relation to stand-alone FLUENT implementation.

The effect of the RPM becomes more important in the context of parameter continuation as demonstrated in Table 2. Here, the RPM basis is built once for the first parameter value, i.e. Re=4000, and it is used for the subsequent Re numbers as well. As long as the low-dimension matrix $H=Z^T F_U Z$, contains an adequate approximation the dominant spectrum of F_U, the basis Z, need not be reevaluated. The computational tools for evaluating the "quality" of the spectrum of the matrix H are hardwired into the RPM and the user need not make any interventions. In this implementation the speedup reaches 3.1, which means that for a single Re number the iteration convergence is 70% quicker than the stand-alone FLUENT iteration.

#CPUs	FLUENT time (sec)	RPM-FLUENT time (sec)
1	401.7	315.1
5	114.7	90.0

Table 3: Execution time for stand-alone FLUENT and the RPM-assisted FLUENT using the serial and the parallel solver of FLUENT (CORTEX).

Since the RPM treats the FLUENT iterations as "black box", it goes without saying that it can be transparently applied in conjunction with the parallel solver CORTEX built into the FLUENT package. It is worth noting that the RPM processes may be entirely serial regardless of the iteration F. Here the stand-alone FLUENT code and the RPM-assisted iteration are solved in 1 and 5 CPUs. Table 3 summarizes the execution time in each case and it becomes clear that the effect of the RPM is beneficial even for the parallel implementation of FLUENT reducing the solution time by more than 20 %.

6.2 Nonlinear features

A valuable byproduct of the RPM-accelerated procedure is the extraction of dominant eigenvectors of F(U) from the eigenvectors of the reduced Jacobian matrix H. Specifically, the eigenvectors, $y_i \in \mathbf{R}^N$, of F_U are defined as $y_i=Zv_i$, where $v_i \in \mathbf{R}^{l_Z}$ are the eigenvectors of H and $Z \in \mathbf{R}^{N\times l_Z}$ is the basis computed in the course of the RPM procedure. In principle, though, the interesting part of the spectrum in terms of nonlinear analysis, is the dominant spectrum of the Jacobian of the discretized physical problem, J, which is absolutely unavailable. Nevertheless, when the iteration F is stable, it is a fair assumption that the dominant spectrum of F_U contains the dominant spectrum of the Jacobian of the discretized physical problem.

The distinction between the dominant eigenmodes of the iteration and those of the physical problem has been demonstrated in detail in [2] for systems of partial differential equations and systems of partial differential and algebraic equations. In the current implementation, where we are dealing with a "black box" type of code, we can verify that assumption by comparing the eigenvectors y_i with the ones derived from perturbation analysis of the transient Navier-Stokes. Let $(\bar{p},\bar{\mathbf{u}})$ be a perturbation that is added to the steady solution of the continuity equation and the transient Navier-Stokes that gives

$$p = p_s + e^{\mu t}\bar{p}\,,\ \mathbf{u} = \mathbf{u}_s + e^{\mu t}\bar{\mathbf{u}} \tag{16}$$

where $(p_s,\mathbf{u}_s)$ are the steady solutions. Eqs. (16) are replaced in the continuity equation and the transient Navier-Stokes that yields

$$-\frac{1}{Re}\nabla\cdot\left(\nabla\bar{\mathbf{u}}+(\nabla\bar{\mathbf{u}})^T\right)+(\mathbf{u}_s\cdot\nabla)\bar{\mathbf{u}}+(\bar{\mathbf{u}}\cdot\nabla)\mathbf{u}_s+\nabla\bar{p}=-\mu\bar{\mathbf{u}} \tag{17}$$

$$\nabla\cdot\bar{\mathbf{u}}=0 \tag{18}$$

Eqs. (17)-(18) are linear and are treated with the GFEM method that yields the generalized eigenvalue problem

$$J\begin{bmatrix}\bar{p}\\ \bar{\mathbf{u}}\end{bmatrix}=-\mu\begin{bmatrix}0 & 0\\ 0 & M\end{bmatrix}\begin{bmatrix}\bar{p}\\ \bar{\mathbf{u}}\end{bmatrix} \tag{19}$$

where J is the Jacobian matrix as defined in paragraph 4 and M is the mass matrix defined as $M = \int_{\Omega} \varphi\varphi^T d\Omega$. The solution $(p_s, \mathbf{u}_s)$ is stable when the perturbation that is applied goes to zero with time, i.e. the real parts of the eigenvalues, μ, are negative, otherwise the solution is unstable. Working on a 200×200 mesh for Re=5000, only a handful of eigenvalues and eigenvectors are computed using ARPACK routines [19] (for more results and detailed analysis cf. [20]). The computed eigenmodes are the ones closest to the origin along the imaginary axis and are shown in Fig. 3 and Fig. 4.

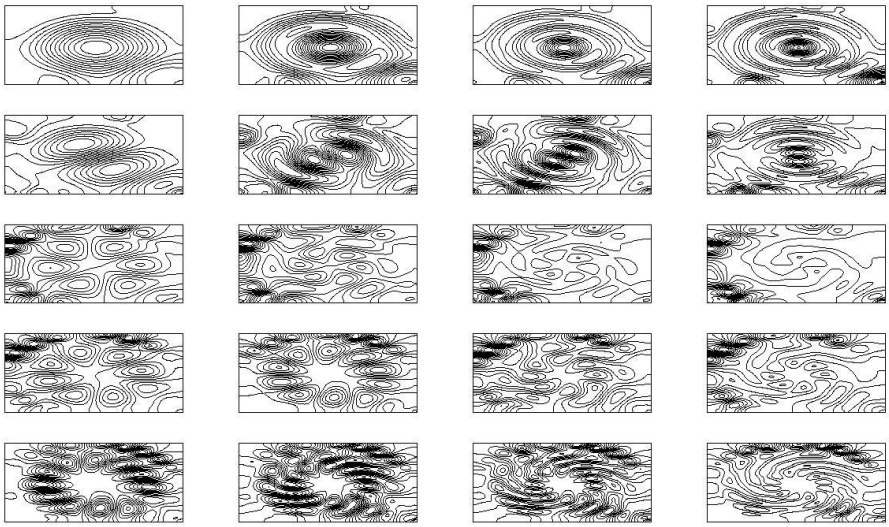

Figure 3: Pressure contours of the real parts of the pressure eigevector for Re=5000

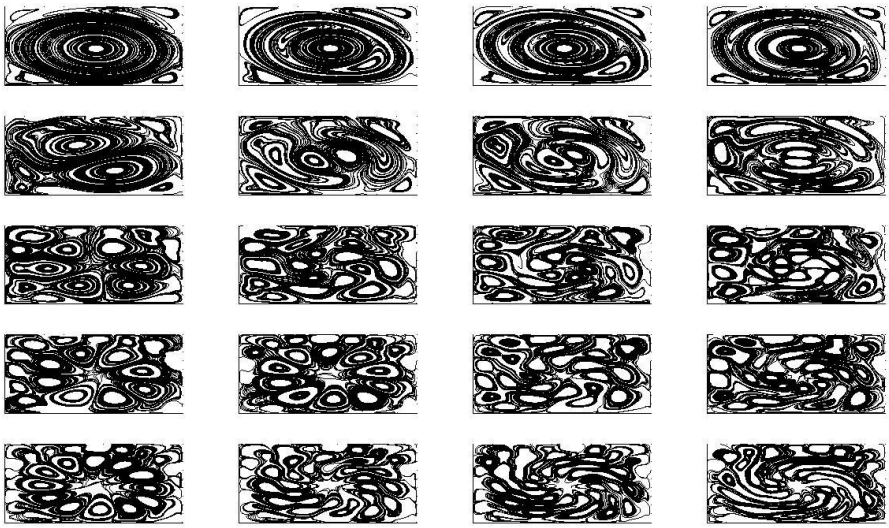

Figure 4: Streamlines created from the real parts of the velocity component of the eigenvectors for Re=5000

The eigenvectors derived by the RPM-assisted iteration, shown in Figures 5 and 6 are in agreement with the first and third eigenvector of the first row in Figures 3 and 4 and they correspond to eigenvalues that are μ_1= -0,012+1,412e^{-16}i and μ_2=-0,083-

$1{,}639e^{-15}i$. The corresponding dominant eigenvalues of F_U are 0.296 and 0.0099 respectively which lie far away from the limits of the unit circle. This indicates that the iteration F "remaps" the Jacobian J into the matrix F_U with a more favorable spectrum signifying that the iteration F is numerically very stable.

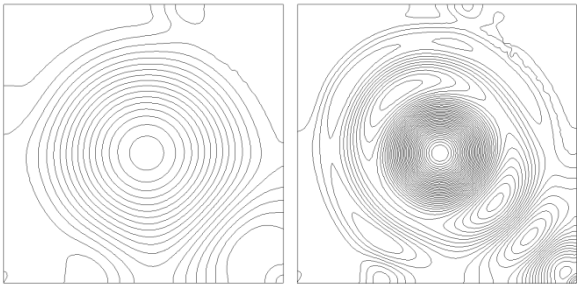

Figure 5: Pressure contours from RPM-derived eigenvectors; Re=5000

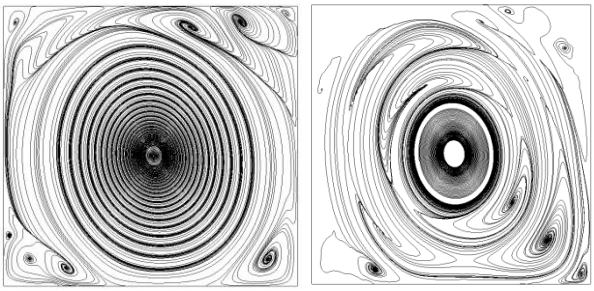

Figure 6: Streamlines created from the RPM-derived eigenvectors; Re=5000

7 CONCLUSIONS

The Recursive Projection method is successfully applied in conjunction with a legacy finite element code and also a "black box" commercial package, FLUENT. The RPM enabled convergence on a Newton/GMRES(m) implementation of the incompressible Navier-Stokes equations for the lid-driven cavity problem. The stand-alone code results in stagnation for various parameters of GMRES(m) yet combined with the RPM, convergence is quick and seamless over a range of parameter values for the method, such as k_{max}. Wrapped around FLUENT, significant acceleration is observed especially in the context of parameter continuation since the method retains useful information for subsequent parameter values. Therefore it economizes on basis building calculations which are the most computationally expensive part of the RPM. The method can be transparently combined with the parallel FLUENT solver CORTEX. A useful by-product of the RPM-FLUENT implementation, is the extraction of dominant eigenmodes of the discretized physical problem which are otherwise unavailable from FLUENT. The derived eigenvectors result from the solution of a low-dimensional eigevalue problem for the reduced Jacobian H which is computed in the course of iteration.

ACKNOWLEDGMENTS

This work is part of a ΠΕΝΕΔ 2003 project, cofinanced 80% of public expenditure through EC - European Social Fund, 20% of public expenditure through Ministry of Development - General Secretariat for Research and Technology and through the

private sector, under measure 8.3 of OPERATIONAL PROGRAM 'COMPETITIVENESS' in the 3rd Community Support Program. The work was also partially supported by the State Scholarships Foundation, through a fellowship to N.C.

REFERENCES

[1] G.M. Shroff, and H.B. Keller, Stabilization of unstable procedures - the recursive projection method. *SIAM J. Num. Anal.* **30**, pp. 1099-1120 (1993).
[2] E. Koronaki, A. Boudouvis, I. Kevrekidis, Enabling stability analysis of tubular reactor models using PDE/PDAE integrators. *Comput. Chem. Eng.* **27**, pp. 951–964 (2003).
[3] H. Von Sosen, I. Folds and bifurcations in the solutions of semi-explicit differential-algebraic equations. II. The recursive projection method applied to differential-algebraic equations and incompressible fluid mechanics, *PhD thesis*, California Institute of Technology (1994).
[4] P. Love, Bifurcations in Kolmogorov and Taylor-vortex flows, *PhD thesis*, California Institute of Technology (1999).
[5] K. Lust, D. Roose, A. Spence, A. Champneys, An adaptive Newton–Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.* **19**, pp. 1188–1209 (1998).
[6] C. Theodoropoulos, N. Bozinis, C. Siettos, C. Pantelides, I. Kevrekidis, A stability/bifurcation framework for process design, Presented at *Annual Meeting of the American Institute of Chemical Engineers*, AIChE, (2001).
[7] C. Gear, I. Kevrekidis, C. Theodoropoulos, 'Coarse' integration/bifurcation analysis via microscopic simulators: Micro-Galerkin methods. *Comput. Chem. Eng.* **26,** pp. 941–963 (2002).
[8] J. Moller, O. Runborg, P. Kevrekidis, K. Lust, I. Kevrekidis, Equation-free, effective computation for discrete systems: A time stepper based approach. *Int. J. Bifurcat. Chaos* **15,** pp. 975–996 (2005).
[9] C. Theodoropoulos, Y. Qian, I. Kevrekidis, "Coarse" stability and bifurcation analysis using time-steppers: A reaction–diffusion example, In Proceedings of *the National Academy of Sciences of the United States of America* **97,** pp. 9840–9843 (2000).
[10] G. Pashos, E.D. Koronaki, A.N. Spyropoulos, A.G. Boudouvis, Accelerating an inexact Newton/GMRES scheme by subspace decomposition. *Appl. Num. Math.*, doi:10.1016/j.apnum.2009.08.003 (2009).
[11] H. Jarausch, W. Mackens, Solving large nonlinear systems of equations by an adaptive condensation process. *Numer. Mathem.* **50,** pp. 633–653 (1987).
[12] P. Gresho, R. Sani, Incompressible Flow and the Finite Element Method: Advection–Diffusion and Isothermal Laminar Flow, *John Wiley and Sons* (1998).
[13] A. Yeckel, J. Derby, Parallel computation of incompressible flows in materials processing: Numerical experiments in diagonal preconditioning. *Parallel Comput.* **23,** pp. 1379–1400 (1997).
[14] http://www.fluent.com
[15] H. K. Versteeg and W. Malalasekera, An introduction to computational fluid dynamics. The finite volume method, 2nd Edition, *Prentice Hall* (2006)
[16] FLUENT, FLUENT Documentation, User Guide, Version 6.3. (2006)
[17] http://www-unix.mcs.anl.gov/mpi/
[18] G.E. Karniadakis and R.M. Kirby II, Parallel Scientific Computing in C++ and MPI, *Cambridge University Press* (2003).

[19] R.B. Lehouqc, D.C. Sorensen and C. Yang, Arpack user's Guide, solution of Large-Scale Eigenvalue Problems with implicitly Restarted Arnoldi Methods, *SIAM* (1998).
[20] G. Pashos, Large-Scale, nonlinear computations with parallel iterative methods-applications in the analysis of transport phenomena, *PhD thesis*, National Technical University of Athens (2009).