Commun. Comput. Phys.
doi: 10.4208/cicp.301109.080410s

Vol. **9**, No. 3, pp. 607-626
March 2011

Extrapolation-Based Acceleration of Iterative Solvers: Application to Simulation of 3D Flows

Leopold Grinberg[*] and George Em Karniadakis

Division of Applied Mathematics, Brown University, Providence 02912, USA.

Received 30 November 2009; Accepted (in revised version) 8 April 2010

Available online 17 September 2010

To the memory of David Gottlieb

Abstract. We investigate the effectiveness of two extrapolation-based methods aiming to approximate the initial state required by an iterative solver in simulations of unsteady flow problems. The methods lead to about a ten-fold reduction in the iteration count while requiring only negligible computational overhead. They are particularly suitable for parallel computing since they are based almost exclusively on data stored locally on each processor. Performance has been evaluated in simulations of turbulent flow in a stenosed carotid artery and also in laminar flow in a very large domain containing the human intracranial arterial tree.

AMS subject classifications: 52B10, 65D18, 68U05, 68U07

Key words: High-order methods, POD, CFD, low-energy preconditioner.

1 Introduction

Discretization of first-order (in time) PDEs

$$\frac{\partial \mathbf{u}}{\partial t} = f_a(t,\mathbf{u},\mathbf{x}) \Rightarrow \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = f_d(t^{n+1},\mathbf{u}^{n+1},\mathbf{u}^n,\mathbf{x})$$

often requires the iterative solution of a linear system of equations of the form $\mathbf{A}\mathbf{u}^{n+1}=\mathbf{b}$; here f_a is an analytic function and f_d is its discrete counterpart.

In general, the performance of the iterative solver is determined by: (1) the condition number of the (preconditioned) operator; (2) the number of floating points operations required at each iteration (e.g., sparsity of the operator); (3) degree of parallelization, i.e., minimization of the sequential part of an algorithm with respect to the work performed

[*]Corresponding author. *Email addresses:* `lgrinb@dam.brown.edu` (L. Grinberg), `gk@dam.brown.edu` (G. E. Karniadakis)

in parallel; and (4) the proximity of the initial state (guess) to the exact solution. The first two factors are related to the properties of the operator and also to the discretization method employed. The third factor is greatly affected by the choice of the numerical scheme for iterative solution and its implementation. While in the last few decades significant effort has been put in developing effective preconditioning techniques that minimize the condition number of the matrix $\mathbf{A}$ and maximize its sparsity, very little work has been published on attempts to obtain a good initial state to initialize the iterative solver. It is common practice to use the solution from a previous time step $\mathbf{u}^n$ as an initial state, however, an *approximate* solution for $\mathbf{u}^{n+1}$ can be obtained using various techniques. Depending on the method and the smoothness of $\mathbf{u}(t)$, the distance between approximate solution (hereafter denoted by $||\mathbf{u}_{ap}-\mathbf{u}^{n+1}||$) can be significantly smaller than $||\mathbf{u}^n-\mathbf{u}^{n+1}||$, hence leading to a substantial reduction in the number of iterations. This, in turn, lowers the CPU-time and enhances the overall efficiency of a solver, hence it is crucial that obtaining $\mathbf{u}_{ap}$ will require *minimal computational effort*.

Markovinović and Jansen [1] employed Proper Orthogonal Decomposition (POD) to accelerate convergence of iterative solvers, and tested it in simulations of two-phase flow through heterogeneous porous media. Specifically, a two-step projection method was proposed: in the first substep, solutions computed at previous time steps were projected onto a subspace spanned by a low number of global POD modes. Subsequently, an approximate solution $\mathbf{u}_{ap}$ was obtained by *solving* the governing equations in the reduced space and projecting the result back to the original, high-dimensional space. This procedure was completed by solving the original system of equations using the $\mathbf{u}_{ap}$ (instead of $\mathbf{u}^n$) as an improved initial state, leading to 67% reduction in the computing time. The key idea of this method is based on the observation that computing the approximate solution (initial state for the iterative solver) using the reduced basis model is less computationally demanding than performing a number of iterations required to advance the solution to the same state. In another study, Tromeur-Dervout and Vassilevski [2] also implemented POD to predict the solution field $\mathbf{u}^{n+1}$ using a fully-implicit scheme. A low-dimensional solution based on POD was used to provide a better initial state to an inexact Newton back-tracking method, and a two-fold reduction in the computing time was reported.

A different use of POD was attempted in [3], where a Galerkin-free methodology was implemented for simulating unsteady flow past a circular cylinder. Extrapolation of POD modes was pursued over *large* time steps and the predictions were used as initial conditions to a Navier-Stokes solver running over much shorter time steps. A related study but targeting the right-hand-side (RHS) of the linear system was pursued in [4] where it was proposed to reduce the iteration count in solution of a linear system of equations $\mathbf{A}\mathbf{u}^{n+1}=\mathbf{b}^{n+1}$ by using information contained in successive RHS vectors $\mathbf{b}^{n+1-k}$, $k=1,\cdots,L$ and corresponding solutions $\mathbf{u}^{n+1-k}$, $k=1,\cdots,L$. This method is based on solving a modified system $\mathbf{A}\tilde{\mathbf{u}}=\tilde{\mathbf{b}}$, where $\tilde{\mathbf{b}}$ is computed by the Gram-Schmidt orthogonalization of the successive RHS vectors. The solution $\mathbf{u}^{n+1}$ is obtained by adding a superposition of the solutions $\mathbf{u}^{n+1-k}$, $k=1,\cdots,L$ to $\tilde{\mathbf{u}}$. In [4] several variations of this method have been proposed; while the computational complexity of the methods varied, all lead to significant reduction in the iteration count. This method also adds a certain computational overhead as well as blocking collective and non-blocking point-to-point communications.

In this paper we investigate the performance of extrapolation-based acceleration techniques for the spectral/hp element method. We test the POD-based extrapolation and also an extrapolation performed in the *modal* space (space of the amplitudes of the high-order polynomials approximating the solution). The later is more efficient than the approach in [1, 2] and it also involves significantly less memory and inter-processor communications. We test performance using an embarrassingly parallel diagonal preconditioner and also a scalable and very effective parallel Low-Energy Basis preconditioned (LEBP) [7, 8] for the iterative solution of Helmholtz and Poisson problems with Conjugate Gradient method (PCG). Numerical results are reported on simulations of arterial flows performed on CRAY XT4 and CRAY XT5.

2 Methods

In this section we first overview the spectral/hp element method implemented in the parallel code $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$. Second, we present two techniques employed in the current study for calculating the approximate solution $\mathbf{u}_{ap}$. The two methods are based on extrapolation performed in modal space. The first method (denoted as Method 1) requires computing the POD modes and reconstructing the flow field variables using extrapolated values of the temporal POD modes. The second method (denoted as Method 2) is employed in conjunction with the spectral/hp element discretization. The spatial discretization is based on C^0 Galerkin projection of the solution onto a space spanned by mixed-order Jacobi polynomials. The extrapolation method is applied in the modal space and involves directly the time-dependent amplitudes of the polynomial expansion.

The two acceleration techniques are integrated in the numerical scheme for solving the incompressible Navier-Stokes equations. Specifically, we apply the extrapolation techniques to predict a more accurate initial state for Helmholtz and Poisson equations for the velocity and the pressure, respectively. Velocity is a dynamic variable and it is a smooth function in time, hence a good prediction of $\mathbf{u}^{n+1}$ can be obtained via extrapolation methods. Pressure is not a state variable but a constraint required to ensure the incompressibility of the velocity field, and hence it is difficult to obtain a good initial state for the pressure solver.

2.1 Numerical methods in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$

Here we provide only a brief overview of the discretization method and the numerical scheme employed, for a complete review we refer to [5, 6]. The computational domain employed in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ consists of tetrahedra, hexahedra, prisms, pyramids or a combination of these. Within each element the solution is approximated in terms of hierarchical, mixed order, semi-orthogonal Jacobi polynomial expansions [5]:

$$u^\delta(\mathbf{x}) = \sum_{i=1}^{N_{dof}} \hat{u}_i \Phi_i(\mathbf{x}) = \sum_{e=1}^{N_{el}} \sum_{i=1}^{dim(\mathcal{V}^\delta)} \hat{u}_i^e \phi_i^e(\mathbf{x_e}(\xi)), \tag{2.1}$$

where N_{dof} is a total number of degrees of freedom and $\phi_i(\mathbf{x}(\xi))$ are polynomials defined in a space $(\mathcal{V}^\delta)$ of order P, which when pieced together under the mapping $\mathbf{x}(\xi)$ make a C^0 continuous (global) expansion $\Phi_i(\mathbf{x})$. The superscript δ emphasizes that we use a finite (truncated) space. To simplify the notation we will omit the superscript δ in the rest of the manuscript. The expansion is hierarchical in a sense that the modes are separated into vertex (linear term), edge, face and bubble (interior) modes. In Fig. 1 we provide an illustration of the domain decomposition and polynomial basis employed in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$.

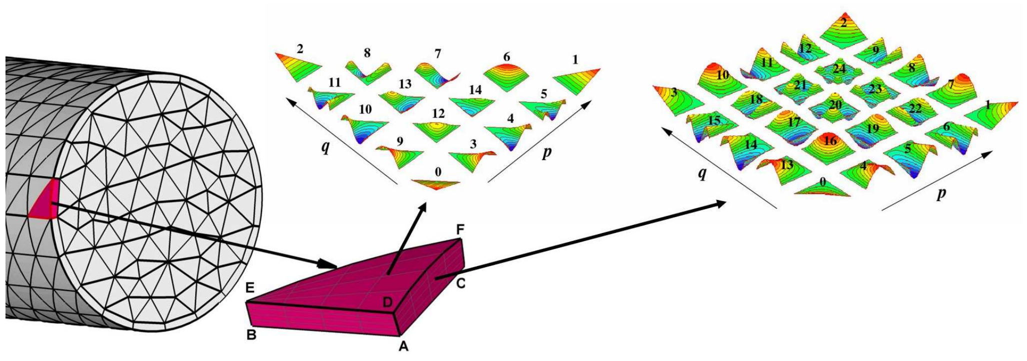


Figure 1: Illustration of the unstructured surface grid and the polynomial basis employed in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$. The solution domain is decomposed into nonoverlaping elements; within each element the solution is approximated by vertex, edge, face and (in 3D) interior modes. The shape functions associated with the vertex, edge and face modes for fourth-order polynomial expansion ($P=4$) defined on triangular and quadrilateral faces are shown in color.

The polynomial expansion basis within each element is decomposed into interior and boundary modes (vertex, edge and face) to help construct a global C^0-continuous field. The interior modes have zero support on the elemental boundaries, thus the boundary and interior degrees of freedom can be numerically decoupled through a technique known as *substructuring*, where the Schur complement of the boundary system is constructed. The boundary degrees of freedom, corresponding to adjacent elements, are coupled due to the requirement of C^0-continuity.

In the current study we consider 3D unsteady incompressible flow in a rigid domain Ω in R^3; the flow is described by the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot(\nabla \mathbf{u}) = -\nabla p + \nu \nabla^2 \mathbf{u}, \qquad \nabla \cdot \mathbf{u} = 0, \tag{2.2}$$

where $\mathbf{u} = [u,v,w]$ is the velocity vector, p is the pressure, t is time and ν is the kinematic viscosity. In order to solve Eq. (2.2) numerically we decouple the velocity and the pressure fields by applying a high-order time-splitting scheme [12]. First, the provisional field

$\mathbf{u}^*$ is computed in physical space (on a grid of quadrature points) using formula (2.3a). Second, we apply Galerkin projection to obtain the weak formulations for the pressure (2.3b) and the velocity (2.3c) fields:

$$\mathbf{u}^* = \sum_{k=0}^{Je-1} \alpha_k \mathbf{u}^{n-k} - \Delta t \Big(\sum_{k=0}^{Je-1} \beta_k (\mathbf{nl})^{n-k} + \mathbf{f} \Big), \quad \mathbf{nl} = \mathbf{u} \cdot (\nabla \mathbf{u}), \tag{2.3a}$$

$$\mathbf{L}\hat{p} = -\frac{1}{\Delta t}(\nabla \cdot \mathbf{u}^*, \phi) + \Big(\frac{\partial p}{\partial n}, \phi \Big), \tag{2.3b}$$

$$\mathbf{H}\hat{\mathbf{u}}^{n+1} = \frac{1}{\gamma_0}(\mathbf{u}^* - \Delta t \nabla p, \phi) + \frac{\Delta t \nu}{\gamma_0} \Big(\frac{\partial \mathbf{u}}{\partial n}, \phi \Big)^{n+1}. \tag{2.3c}$$

Here γ_0, α_k are coefficients of backward differentiation formula, β_k are the integration coefficients as suggested in [12], Je is the order of the time discretization scheme, $\mathbf{H} = \mathbf{M} - (\Delta t \nu / \gamma_0)\mathbf{L}$, and $\mathbf{M}$ and $\mathbf{L}$ are the mass and stiffness matrices, respectively.

The linear systems (2.3b) and (2.3c) are solved iteratively. For example, the solution vector $\hat{\mathbf{u}}$ and the appropriate forcing term $\mathbf{f}$ are decomposed into contributions associated with the boundary ($\hat{\mathbf{u}}_\mathbf{b}$) and interior ($\hat{\mathbf{u}}_\mathbf{i}$):

$$\begin{bmatrix} \mathbf{H_{bb}} & \mathbf{H_{bi}} \\ \mathbf{H_{ib}} & \mathbf{H_{ii}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_\mathbf{b} \\ \hat{\mathbf{u}}_\mathbf{i} \end{bmatrix} = \begin{bmatrix} \mathbf{f_b} \\ \mathbf{f_i} \end{bmatrix}. \tag{2.4}$$

Due to non-overlapping support of the interior modes, the contributions of $\mathbf{H_{ii}}$ to $\mathbf{H}$ are decoupled from each other. It is therefore more efficient to construct the Schur complement

$$\mathbf{S} = \mathbf{H_{bb}} - \mathbf{H_{bi}}[\mathbf{H_{ii}}]^{-1}\mathbf{H_{ib}}$$

and solve first for the boundary degrees of freedom, which when known can be used to recover the interior degrees of freedom. In the discretization method we implement here the global Schur complement operator can be constructed from the local (element-wise) operators, however, global assembly is not necessary if the linear problem for the boundary degrees of freedom is solved iteratively as it is explained in detail in [5]. The solution for the boundary degrees of freedom is obtained iteratively by PCG. The efficiency of the iterative solver is strongly affected not only by the preconditioner but also by the proximity of the initial state $\hat{\mathbf{u}}_\mathbf{ap}$ to the solution $\hat{\mathbf{u}}_\mathbf{b}$. To obtain better initial states we employ the methods described in the next sections.

2.2 Method 1: POD-based extrapolation

POD is a very effective method for identifying an energetically dominant set of eigenmodes in an evolving system. For a comprehensive review on the POD method we refer the reader to [9–11]. Here we provide a brief overview of the method. For a set of data $\mathbf{u}(t,\mathbf{x})$, represented as a function of physical space $\mathbf{x}$ and time t, POD determines a set of orthogonal basis functions of space $\Psi_q(\mathbf{x})$ and temporal modes $a_q(t)$:

$$\mathbf{u}(t,\mathbf{x}) = \sum_{q=1}^{Q} a_q(t)\Psi_q(\mathbf{x}). \tag{2.5}$$

We employ the method of snapshots [9] to compute the POD modes in a time interval $T=[t-(Q-1)\Delta t,\ t]$. The inner product between every pair of velocity fields (snapshots)

$$C(t,t') = \int_\Omega \mathbf{u}(t,\mathbf{x})\mathbf{u}(t',\mathbf{x})d\mathbf{x}$$

is the temporal auto-correlation covariance matrix $\mathbf{C}$ used as the kernel. The temporal modes $a_q(t)$ are the eigenvectors of the $\mathbf{C}$ matrix and are calculated by solving an eigenvalue problem of the form:

$$\int_T C(t,t')a_q(t')dt' = \lambda_q a_q(t).$$

Using orthogonality, the POD spatial modes are calculated by

$$\Psi_q(\mathbf{x}) = \int a_q(t)\mathbf{u}(t,\mathbf{x})dt.$$

The eigenvalue of a single mode represents its contribution to the total kinetic energy of the field which is proportional to the sum over all eigenvalues. Therefore, the eigenspectrum of the decomposition can be regarded as the primary indicator of the importance of each individual mode from the energetic point of view. The modes with the lowest numbers are the most energetic whereas the high modes may contribute little to the field $\mathbf{u}(t,\mathbf{x})$, and hence they may be ignored.

In order to approximate the solution at time step t^{n+1} the following procedure is employed:

Step 1: At every time step t^n the correlation matrix $\mathbf{C}$ is constructed from the solution fields $\mathbf{u}^{n-k}$, $k=0,1,\cdots,Q-1$:

$$C_{i,j} = \int_\Omega \mathbf{u}(t^{n+1-i},\mathbf{x})\mathbf{u}(t^{n+1-j},\mathbf{x})d\mathbf{x}, \qquad i,j=1,\cdots,Q.$$

It is not necessary to recompute $\mathbf{C}$ for *all* indices $i,j=1,\cdots,Q$ at every time step, but only the inner products $\big(\mathbf{u}(t^n,\mathbf{x}),\mathbf{u}(t^{n+1-j},\mathbf{x})\big)$, $j=1,\cdots,Q$, since the inner products of the fields computed at previous times steps do not change. The correlation matrix is computed in parallel and one global summation of a vector of size Q is required. We also considered a variation of the POD of a vector field, where the velocity components are treated as independent (uncorrelated) scalar fields, i.e, the correlation matrix and the temporal modes for each component of the velocity vector are computed. The computational complexity of such method is only slightly higher: (a) global summation of a vector of size Qd is required, and (b) the eigenvalues and eigenvectors of d correlation matrices (instead of one) should be computed; here d is the spatial dimension of a problem. To distinguish the two approaches, we denote the first one by $\mathbf{u}_{ap} = POD_1(\mathbf{u})$ and the second one by $\mathbf{u}_{ap} = POD_2(\mathbf{u})$; most of the results reported here are for POD_1.

Step 2: The eigenvalues and eigenvectors ($a_q(t)$) of $\mathbf{C}$ are calculated at each time step, then the spatial modes $\Psi_q(\mathbf{x})$, $q=1,\cdots,Q_R$ are computed. Typically $Q_R < Q$, which leads to computational savings. The parameter $1 \le Q_R \le Q$ is adjusted at each time step using the following criteria:

(a) The value of $1 \le Q_R \le Q$ is then chosen such that

$$\sum_{k=Q_R+1}^{Q} \lambda_k < \epsilon, \quad 0 < \epsilon \ll 1,$$

i.e., the first Q_R modes contribute $100\% - \epsilon$ of the kinetic energy $E_{1,Q} = \sum_{k=1}^{Q} \lambda_q$.

(b) Alternatively, threshold criteria can be applied, such that the Q_R parameter satisfies $\lambda_q < \epsilon$, $\forall q > Q_R$.

The choice of ϵ is very important for effectiveness of the method. Although, POD is basically a compression technique which might be employed to filter-out the high frequency components associated with small energy, we have found it most efficient to retain the low energy components by setting ϵ to very low values of order $\mathcal{O}(10^{-14} - 10^{-15})$. We also note that here we use Q_R as a global parameter; generally speaking, it is possible to adjust Q_R for different regions of the computational domain based on some local features of the solution. For example, reducing Q_R in the regions of high gradients may have some smearing effect.

Step 3: The values of $a_q(t^{n+1})$ are computed using the formula

$$a_q(t^{n+1}) = \sum_{k=0}^{Q_R-1} \beta_k a_q(t^{n-k}), \tag{2.6}$$

where β_k are extrapolation coefficients.

Step 4: The approximate solution is computed from

$$\mathbf{u}_{ap} = \sum_{q=1}^{Q_R} a_q(t^{n+1}) \Psi_q(\mathbf{x}). \tag{2.7}$$

Depending on the spatial discretization method employed, an additional step might be required to complete the procedure: If the linear system of equations is formulated in the modal space, e.g., the Eq. (2.4), transformation of $\mathbf{u}_{ap}$ from physical space to modal is required. This transformation is typically the most computationally intensive step of the POD-based extrapolation.

Computational complexity: The POD-based extrapolation technique requires $2\times Q\times Nqp\times sizeof(double)$ bytes storage per field-half of it to store the solutions at previous time steps and another half to store spatial modes. The memory requirement for the correlation matrix $\mathbf{C}$ is $Q\times Q$ and the same memory is required to store the temporal modes; since Q is typically of order $\mathcal{O}(1)$, the memory requirements are not substantial. At each time step the following operations are required:

(i) Q values of $\mathbf{C}$ are computed via numerical integration:

$$C_{i,1}=C_{1,i}=\sum_{j=1}^{Nqp}\left[u(t^n,\mathbf{x}_j)u(t^{n+1-i},\mathbf{x}_j)w_j\right],\quad i=1,\cdots,Q,$$

here w_j are the integration weights; computing $C_{1,i}$, $i=1,\cdots,Q$, requires one *dvmul* operation[†] and Q vector-vector dot products.

(ii) The eigenvalues and eigenvectors of $\mathbf{C}$ are computed by *dsyev* function[‡]. Considering the small size of $\mathbf{C}$, the eigenvalue decomposition can be performed sequentially.

(iii) Q_R spatial modes are computed by performing one *dscal* and Q_R-1 *daxpy* operations for each field.

(iv) The field $\mathbf{u}_{ap}$ is computed from the spatial and temporal POD modes; this operation holds the same computational complexity as (iii).

(v) Transformation of a solution into modal space, which is performed by solving a projection problem. The projection problem can be solved locally, within each element, or globally. From the computing standpoint, the local projection is advantageous since it involves only one matrix-vector multiply per element in addition to computing inner products of the approximated solution with the shape functions Φ. Approximate solution for the interior modes is not required, which reduces the computational effort. However, the local projection does not guarantee the C^0 continuity, which can be achieved by averaging degrees of freedom shared by the adjacent elements.

2.3 Method 2: Spectral-based extrapolation

In this section we review a simple but effective method to obtain an approximation $\mathbf{u}_{ap}\approx\mathbf{u}^{n+1}$. Compared to the other methods, the spectral-based extrapolation has the least computational complexity and memory requirements. Moreover, the method is local, i.e., approximation of a degree of freedom i does not require information on the degree of freedom j, hence the method is embarrassingly parallel. It is also general and can be applied in conjunction with other methods, e.g., finite differences or finite elements.

The approximate solution at the time step t^{n+1} is computed using

$$\mathbf{u}_{ap}(\mathbf{x})=EXT(\mathbf{u})\equiv\sum_{k=0}^{N-1}\beta_k\mathbf{u}^{n-k}(\mathbf{x}),\tag{2.8}$$

[†]function *dvmul* computes $z[i]=x[i]\cdot y[i]$.

[‡]function *dsyev* computes all eigenvalues and, optionally, eigenvectors of a real symmetric matrix.

where β_k are extrapolation coefficients.

The formula (2.8) can be applied to extrapolate solution in the physical as well as in the modal space (2.1). The numerical algorithm implemented in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ requires transforming velocity values from the modal to physical space in order to compute the non-linear term. Hence, choosing to perform extrapolation of solution in physical space will not require additional computationally expensive operation. However, it is still advantageous to perform the extrapolation using modal values $\hat{\mathbf{u}}_b(t)$ due to several reasons:

• Consider that the linear system we solve iteratively is formulated for modal Galerkin bases, then extrapolation performed in the physical space would require solving the projection problem $\mathbf{M}\hat{\mathbf{u}}_{ap} = (\mathbf{u}_{ap},\mathbf{\Phi})$, which demands considerable computational effort. To reduce the computational cost it is possible to solve a local projection problem (elementwise) directly using the pre-computed inverse of the mass matrix $(\mathbf{M}^{\mathbf{e}})^{-1}$. The local projection may destroy the C^0 continuity, hence additional computational effort might be needed if C^0 continuity of $\hat{\mathbf{u}}_{ap}$ is required.

• The number of modal degrees of freedom is typically lower than the number of quadrature points needed for projection operations; consequently extrapolation in the modal space requires less floating point operations and storage.

• Solution of a linear system arising in spectral element discretization is typically performed using the Schur decomposition technique, which decouples the boundary and interior modes. The interior modes are often computed using a direct solver, whereas, the boundary modes are computed iteratively; thus, extrapolation of the interior modes is not required.

In this study, the extrapolation operator is applied to the modal coefficients, hence naturally preserving the C^0 continuity. We note that we employ the extrapolation not to compute the velocity field $\mathbf{u}^{n+1}$, but to obtain an initial state for the iterative solver. Hence, the accuracy and stability of the numerical scheme are not affected by methods for computing $\mathbf{u}_{ap}$.

Computational complexity: The aforementioned extrapolation technique requires storing solutions from the previous N time steps. The required storage per field is then $N \times Nqp \times sizeof(double)$ bytes, where Nqp is the total number of quadrature points if the extrapolation is performed in physical space. Alternatively, if extrapolation is performed in the modal space, Nqp is the number of *global* boundary degrees of freedom. Additional storage for $\mathbf{u}_{ap}$ is not required since the data can be written instead of solution field from the time step t^{n-N+1}. At each time step one *dscal* and $(N-1)$ *daxpy*[§] calls to BLAS library are required. Transformation of a solution into modal space (if needed) requires N_{el} direct solves of a linear system $\mathbf{M}^e\hat{\mathbf{u}}_{ap} = (\mathbf{u}_{ap},\phi)$ which can be performed using precomputed LU factorization or inverse of a symmetric operator $\mathbf{M}^e$; note that only solution for boundary modes is required. If $\mathbf{u}_{ap}$ is a vector field in R^d then one system with d right-hand-sides should be solved, which is more efficient then solving a series of single RHS systems.

[§]*dscal* function computes $y[i] = ax[i]$; *daxpy* function computes $y[i] = ax[i] + y[i]$.

3 Results

To investigate numerically the effectiveness of the extrapolation methods we considered the following cases: (a) turbulent flow in a stenosed carotid artery, and (b) flow in the intracranial arterial tree, consisting of a very large number of spectral elements. Before we proceed further we provide some definitions and clarify the notation. In this section $\mathbf{x}^k$ denotes a solution vector at iteration k and the vector $\mathbf{b}$ denotes a suitable forcing term. Other quantities are defined as:

- $||a|| = \left(\sum_{i=1}^{N} a_i^2\right)^{\frac{1}{2}}$;
- $r^k = ||\mathbf{A}\mathbf{x}^k - \mathbf{b}||$ — residual, at iteration k;
- $r^k_{Ns} = \frac{r^k}{Ns}$ — residual normalized by the length of vector $\mathbf{x}$ (number of unknowns);
- $r^k_s = \frac{1}{||\mathbf{x}^0||}||\mathbf{A}\mathbf{x}^k - \mathbf{b}||$ — residual scaled by the $||\cdot||$ norm of the initial state vector.

3.1 Accuracy verification

The accuracy of the solver has been verified by simulating unsteady flow in a pipe, forced by a periodically varying pressure gradient. The numerical solution was compared to the analytical solution (Womersley velocity profile). In Table 1, we show the error computed for u and w (streamwise) velocity components; as expected, comparable accuracy is achieved for all choices of the initial state provided to the iterative solver.

Table 1: Simulations of unsteady flow in a pipe: L_∞-error for u (w-streamwise) velocity components. Initial state ($\mathbf{x}^0$) for conjugate gradient solver is provided by: a) $\mathbf{x}^0 = \mathbf{u}^n$, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ with $N=4$, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ with $Q=4$. $\Delta t = 2.0E-4$, stopping criterion for conjugate gradient solver: $r^k_s < TOL\_CG = 1.0E-12$.

P	$\mathbf{x}^0 = \mathbf{u}^n$	$\mathbf{x}^0 = EXT(\mathbf{u})$	$\mathbf{x}^0 = POD_1(\mathbf{u})$
4	1.0E-3 (4.3E-3)	1.0E-3 (3.5E-3)	1.0E-3 (3.5E-3)
6	1.8E-5 (2.1E-4)	2.0E-5 (5.2E-5)	2.0E-5 (5.2E-5)
8	9.9E-7 (7.3E-6)	8.5E-7 (3.2E-6)	8.5E-7 (3.2E-6)

3.2 Turbulent flow simulations in stenosed carotid artery

In this section we consider simulations of flow in a domain of stenosed carotid artery, shown in Fig. 2(right). The narrowing of the internal carotid artery creates a strong jet-flow, and due to curvature and geometric asymmetry the jet adheres to arterial wall and becomes unstable when the flow exceeds a certain velocity limit. In Fig. 2(left) we present typical velocity data, recorded downstream of the stenosis; the high-frequency oscillations signify the onset of turbulence.

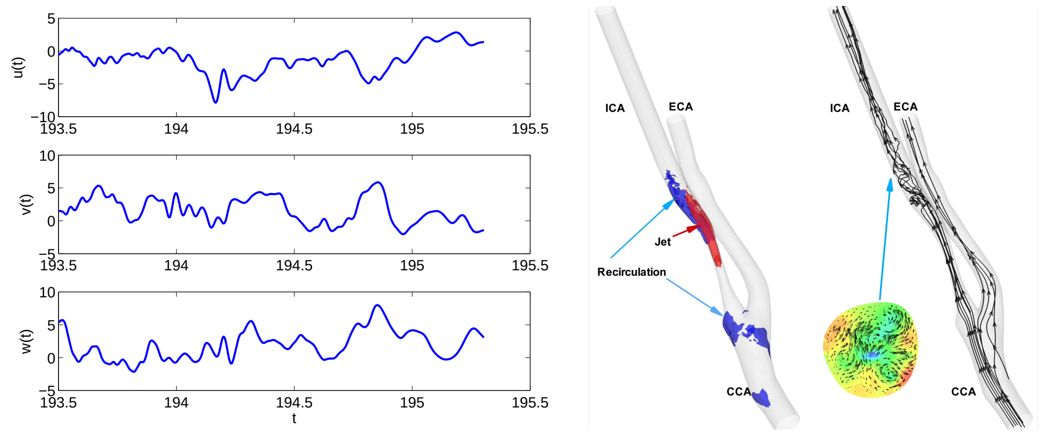


Figure 2: (in color) Unsteady flow simulations in stenosed carotid artery. Left: Non-dimensional velocity monitored downstream stenosis. Right: a high-speed region-red iso-surfaces, back-flow regions-blue iso-surfaces; instantaneous path-lines of swirling flow and cross-stream secondary flows. ICA= Internal Carotid Artery; ECA=External Carotid Artery; CCA=Common Carotid Artery.

Table 2: Turbulent flow simulations in stenosed carotid artery: performance of iterative solver with different initial states. r^0_{Ns}, Nit, T_e-average (over time) initial residual (normalized by the number of unknowns), number of iterations (Nit) and CPU-time (in seconds) required for extrapolation at each time step. Data are averaged over time steps 100 to 4,000; preconditioner: low energy; $\Delta t = 5.0E-5$, $P = 6,10$, $Nel = 22,441$.

	$\mathbf{x}^0 = \mathbf{u}^n$		$\mathbf{x}^0 = EXT(\mathbf{u}), \mathbf{N}=3$			$\mathbf{x}^0 = POD_1(\mathbf{u}), \mathbf{Q}=3$		
P	r^0_{Ns}	Nit	r^0_{Ns}	Nit	T_e	r^0_{Ns}	Nit	T_e
6	2.8E-4	24	1.6E-8	9.1	1.3E-3	1.6E-8	9.1	3.6E-2
10	1.7E-5	19.2	1.9E-9	7.1	1.7E-3	1.9E-9	7.1	1.1E-1

In Table 2, we present results of the two acceleration techniques; both acceleration techniques show comparable performance. The significant reduction in the initial residual r^0_{Ns} is a result of accurate approximation of the initial state. Reduction of the iteration count in simulation with higher resolution ($P=10$) is intriguing and deserves some comments: The results of Table 2 correspond to solution of the Helmholtz equation using preconditioned (with LEBP) conjugate gradient method. We recall that the Helmholtz operator is constructed as a linear combination of the L_2 and H_1 operators, i.e., $\mathbf{H} = \mathbf{M} - (\Delta t \nu / \gamma_0)\mathbf{L}$. Results presented in Table 2 have been computed using very small size of Δt, which makes the L_2 operator ($\mathbf{M}$) dominant over the H_1 operator ($\mathbf{L}$). The excellent p- and h-scaling in solving H_1 problem with the LEBP has been presented and analyzed in depth in [7]; however, the properties of the projection operator $\mathbf{M}$, preconditioned with the LEBP, have not yet been studied. To verify that the reduction in the iteration count is a result of preconditioning we compared the performance of our solver using the LEBP and the diagonal preconditioner, and setting the same initial states and stopping criterion for the PCG; the results are presented in Fig. 3. In the case of diagonal preconditioning the number of iterations grows with P (as expected), while employing

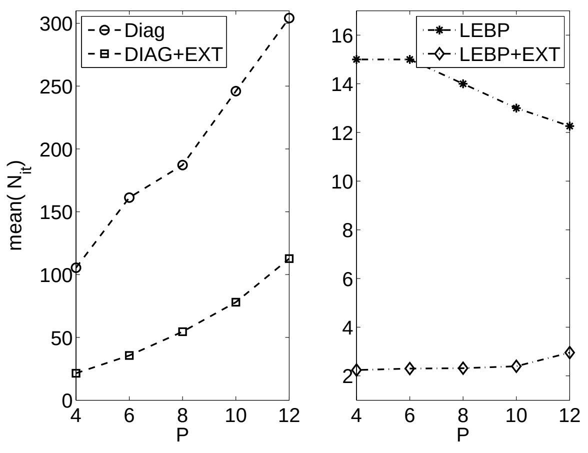


Figure 3: Unsteady flow simulations in stenosed carotid artery: number of PCG iterations required for the Helmholtz solver for the w-component (streamwise) of the velocity field. The iteration count has been averaged over time steps 150 to 450. Solution is obtained with the diagonal (left) and LEBP (right) and also with and without extrapolation of the initial state ($N=4$ and $N=0$). Stopping criteria for PCG solver: $r_s^k < TOL\_CG = 1.0E-8$.

the LEBP lead to large reduction in the iteration count. Similar behavior was also observed in simulations of a pulsatile flow in a pipe used for the accuracy verification in the previous section. In simulation without preconditioning, P-refinement increased the iteration count. More comprehensive analysis of preconditioning with the LEBP operator $\mathbf{M}$ is out of the scope of this paper and is left for future research.

The LEBP requires considerable computational effort: blocking collective and non-blocking point-to-point communications and also matrix-vector multiplications. In contrast, the diagonal preconditioner requires no interprocessor communications and negligible computational effort, hence it might be feasible to use the less effective but embarrassingly parallel preconditioner in conjunction with the methods providing an accurate initial solution for the PCG in order to obtain the lowest computational time. To this end, in Table 3 we compare the performance of the PCG solver with the LEBP and diagonal preconditioner and two choices of initial state; clearly, the choice of the LEBP is advantageous. Although, the last observation is valid for spectral element discretization, this conclusion cannot be generalized for other numerical schemes and preconditioning strategies.

Table 3: Turbulent flow simulations in stenosed carotid artery: average CPU-time required by three Helmholtz solves (for the three velocity components together) for two choices of initial state: a) $\mathbf{x}^0=\mathbf{u}^n$, and b) $\mathbf{x}^0=EXT(\mathbf{u})$ with $N=4$; and two preconditioners: LEBP and diagonal; $\Delta t=5.0E-5$, $P=6$, $Nel=22{,}441$.

	CPU-time (sec)	
preconditioner	$\mathbf{x}^0=\mathbf{u}^n$	$\mathbf{x}^0=EXT(\mathbf{u})$
low energy	0.188	0.037
diagonal	1.09	0.25

It is reasonable to assume that higher order extrapolation may lead to more accurate prediction of the initial state, as long as the extrapolated field is sufficiently smooth. Such assumption can be easily justified by estimating the truncation error of the polynomial extrapolation, which is

$$error \approx c(\Delta t)^N \frac{\partial^N \mathbf{u}}{\partial t^N}.$$

In the case of uniform time-stepping, i.e., $t^n - t^{n-1} = \text{const}$, we have $c=1$. In Table 4 we present results obtained with different orders of extrapolation ($N,Q=3$, $N,Q=4$ and $N,Q=5$). About two orders of magnitude improvement in the accuracy of initial state $\mathbf{x}^0 = EXT(\mathbf{u})$ is observed, and the reduced number of iterations reflects this improvement. However, fifth-order accurate extrapolation ($N=5$) does not lead to lower initial residual, which can be attributed to the numerical error of order TOL_CG present in solutions $\mathbf{u}^{n-k}$, $k=0,\cdots,N-1$ and the possible Runge effect related to high-order interpolation on equidistant grid. In the case of $\mathbf{x}^0 = POD_1(\mathbf{u})$ and $Q=4$ the improvement is not significant. Moreover, with higher POD expansion order ($Q=5$) the performance gets even worse. The relatively poor performance of the POD-based extrapolation is due to the ill-conditioned correlation matrix. Large condition number results in error in computation of orthogonal temporal and spatial POD modes, and, consequently, relatively high extrapolation error. The eigenspectrum of the correlation matrix will be discussed in more detail in the next section.

Table 4: Turbulent flow simulations in stenosed carotid artery: initial state is computed by high-order extrapolation. r^0_{Ns}, Nit, T_e-average (over time) normalized initial residual, number of iterations and CPU-time (in seconds) required for extrapolation at each time step. Data are averaged over time steps 100 to 4,000, preconditioner: low energy; $\Delta t = 5.0E-5$, $P=6$, $Nel=22{,}441$.

	$\mathbf{x}^0 = EXT(\mathbf{u})$			$\mathbf{x}^0 = POD_1(\mathbf{u})$		
	r^0_{Ns}	Nit	T_e	r^0_{Ns}	Nit	T_e
N,Q=3	1.6E-8	9.1	1.3E-3	1.6E-8	9.1	3.6E-2
N,Q=4	2.75E-10	3.7	1.3E-3	1.16E-8	6.75	3.9E-2
N,Q=5	1.94E-10	3.7	1.3E-3	3.33E-8	9.2	5.5E-2

In Tables 2 and 4 we showed data averaged over 4,000 time steps and over the three components of the velocity field. Here, in Fig. 4 we compare the number of iterations required at every time step for each of the three velocity components (u,v,w), an up to eight-fold reduction in iteration count is observed over 40,000 time steps. The reduction is due to improved initial state, which minimizes the initial residual r^0_{Ns}, also shown in Fig. 4(right). In Table 5 we summarize the number of iterations and initial residual (averaged over 40,000 time steps) for three choices of initial state; a slightly higher number of iterations is observed when the initial state is approximated using POD. A more detailed discussion on the last observation follows in the next section.

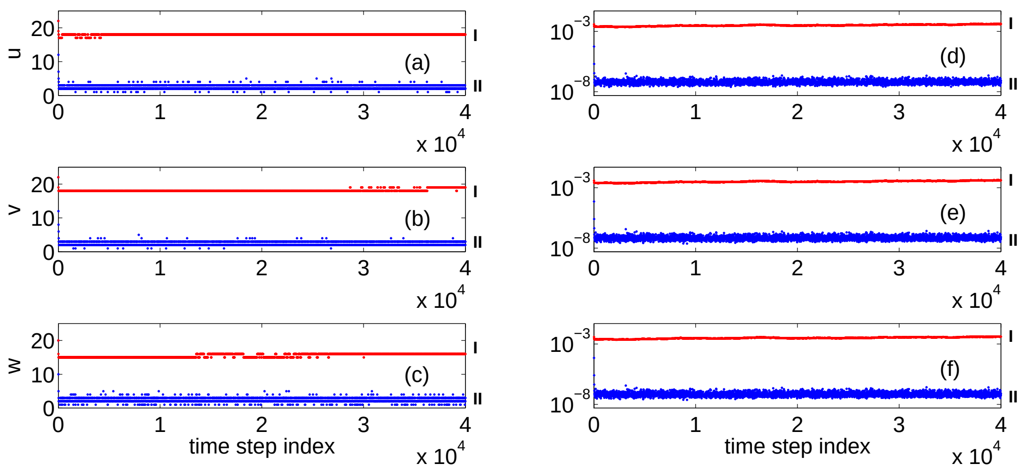


Figure 4: Turbulent flow simulations in stenosed carotid artery: (a-c)-number of iterations required by iterative Helmholtz solver for three velocity components. (d-f)-initial residual r^0_{Ns}, curve marked by **I** (**II**) corresponds to $\mathbf{u}_{ap}=\mathbf{u}^n$ ($\mathbf{u}_{ap}=EXT(\mathbf{u})$, $N=4$). $\Delta t=5.0E-5$, $P=6$, $Nel=22{,}441$.

Table 5: Turbulent flow simulations in stenosed carotid artery: average number of iterations (Nit) and average initial residual $\bar{r}^0_N$ for three choices of initial state: a) $\mathbf{x}^0=\mathbf{u}^n$, b) $\mathbf{x}^0=EXT(\mathbf{u})$ with $N=4$, and c) $\mathbf{x}^0=POD_1(\mathbf{u})$ with $N=4$. Data is averaged over time steps 100 to 40,000, preconditioner: low energy; $\Delta t=5.0E-5$, $P=6$, $Nel=22{,}441$.

	$\mathbf{x}^0=\mathbf{u}^n$		$\mathbf{x}^0=EXT(\mathbf{u})$		$\mathbf{x}^0=POD_1(\mathbf{u})$	
	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$
u	17.97	2.60E-3	2.31	6.80E-8	3.33	1.99E-7
v	18.12	3.20E-3	2.42	7.56E-8	3.55	2.54E-7
w	15.53	5.46e-3	2.36	7.65E-8	2.48	6.62E-8

3.3 Simulations of flow in the Circle of Willis

In this section we consider simulations of flow in a complex network of major brain arteries including the Circle of Willis (CoW), which is reconstructed from MRI images of a human brain; the computational domain is presented in Fig. 5(right). The simulations are started from initial solution $\mathbf{u}(t=0)=0$, and a steady velocity profile is imposed at three inlets of the domain. The simulation time considered here is significantly lower than the time required to establish a steady state solution. The goal of this section is to show that the aforementioned techniques aiming to provide a good initial state for iterative solver can be successfully applied to simulations of a flow in very large computational domains. The size of the computational domain of CoW compared to the domain of carotid artery is very large; specifically, CoW consists of 22 arteries and is discretized into $Nel=162{,}909$ tetrahedral spectral elements.

In Table 6 we compare the number of iterations required by the Helmholtz solver with respect to three choices of initial state. Similarly to the turbulent flow simulations,

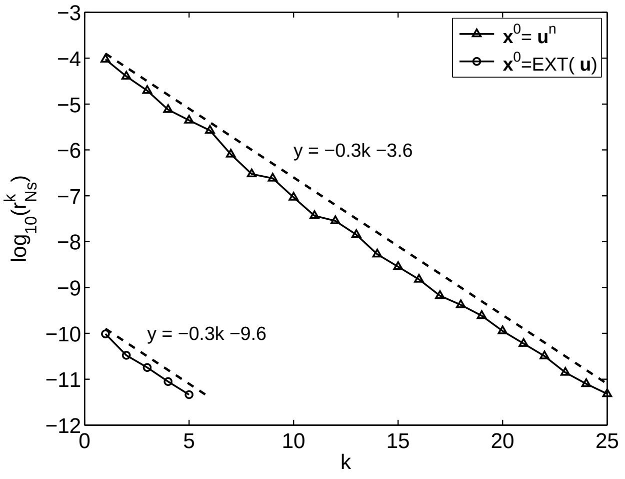
Table 6: Flow simulations in Circle of Willis: performance of iterative solver with different initial states. Average number of iterations (Nit) and average initial residual $\bar{r}^0_{Ns}$ for three choices of initial state: a) $\mathbf{x}^0 = \mathbf{u}^n$, b) $\mathbf{x}^0 = EXT(\mathbf{u})$ with $N=4$, and c) $\mathbf{x}^0 = POD_1(\mathbf{u})$ with $Q=4$. Data is averaged over time steps 300 to 3,000. $Nel = 162{,}909$, $\Delta t = 5.0E-4$, preconditioner: low energy. Stopping criterion for PCG: $r^k < TOL\_CG = 1E-5$ ($r^k_{Ns} \approx 7.9E-12$).

	$\mathbf{x}^0=\mathbf{u}^n$		$\mathbf{x}^0=EXT(\mathbf{u})$		$\mathbf{x}^0=POD(\mathbf{u})$	
	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$	Nit	$\bar{r}^0_{Ns}$
P=3	24.60	2.99E-4	3.70	3.34E-10	6.81	9.70E-9
P=4	22.15	6.01E-5	3.15	1.06E-10	5.81	2.05E-9
P=5	20.85	1.77E-5	3.27	7.22E-11	4.94	5.90E-10
P=6	18.80	8.20E-6	2.67	4.50E-11	4.06	2.87E-10

Table 7: Flow simulations in Circle of Willis: performance of POD-based accelerator. U_{it},V_{it} and W_{it}-number of iterations required for solution of Helmholtz equations for velocity at times steps 2991 to 3000. Q_R-number of POD modes used for velocity field reconstruction. λ_i, i=1,2,3,4-eigenvalues of correlation matrix $\mathbf{C}$. Q=4, P=4, Nel=162,909, Δt=5.0$E-4$, preconditioner: low energy. Stopping criterion for PCG: r^k<TOL_CG=1$E-5$ ($r^k_{Ns}\approx 7.9E-12$).

U_{it}	V_{it}	W_{it}	Q_R	λ_4	λ_3	λ_2	λ_1
7	6	7	3	-8.03e-14	3.37e-10	9.75e-05	1.38e+04
7	7	7	3	-8.96e-13	3.35e-10	9.74e-05	1.38e+04
4	4	4	**4**	**1.01e-12**	3.37e-10	9.73e-05	1.38e+04
7	6	7	3	-1.26e-12	3.38e-10	9.72e-05	1.38e+04
8	8	8	3	-6.90e-13	3.36e-10	9.72e-05	1.38e+04
7	7	7	3	-7.13e-13	3.39e-10	9.71e-05	1.38e+04
4	3	4	**4**	**1.57e-12**	3.39e-10	9.70e-05	1.38e+04
7	6	7	3	-1.86e-12	3.40e-10	9.70e-05	1.38e+04
4	3	4	**4**	**4.05e-13**	3.39e-10	9.69e-05	1.38e+04
3	3	3	**4**	**1.41e-13**	3.39e-10	9.68e-05	1.38e+04

in simulations with LEBP the P-refinement results in lower iteration count. The effect of improved initial state is noticeable for both choices of the extrapolation techniques; however, the performance of Method 2 is superior. To understand the poorer performance of the POD-based extrapolation we monitored the eigenspectra of the correlation matrix $\mathbf{C}$ at every time step (see Table 7). The very fast decay in the eigenspectra signifies that the correlation length between velocity fields is significantly greater then the size of a time step, which results in an ill-conditioned correlation matrix. In simulations where $\partial\mathbf{u}/\partial t \to 0$, the correlation length asymptotically approaches infinity. The correlation matrix $\mathbf{C}$ is symmetric and positive definite, hence the only source for the negative sign of λ_4 (see Table 7) is the numerical error. Since the number of modes used for reconstruction of the velocity field (Q_R, see the forth column in Table 7) is computed dynamically, the negative λ_4 will restrict the value of $Q_R < 4$. We can observe that the number of iterations is correlated with the Q_R parameter, and for $Q_R = 4$, the number of iteration is comparable to what is achieved with $\mathbf{x}^0 = EXT(\mathbf{u})$.



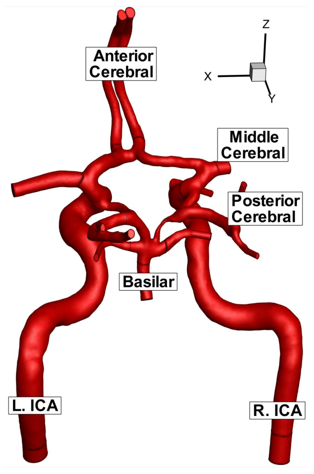


Figure 5: Flow simulations in Circle of Willis. Left: effect of improved initial state and convergence of residual (r^k_{Ns}) in solution with preconditioned conjugate gradient solver. The dash lines correspond to linear least square approximation for the convergence rate. $N=4$, $P=4$, $Nel=162{,}909$, $\Delta t=5.0E-4$, preconditioner: low energy. Right: geometry of the Circle of Willis. Stopping criterion for PCG: $r^k < TOL\_CG = 1E-5$ ($r^k_{Ns} \approx 7.9E-12$).

In Fig. 5 the convergence of residual is presented. The data corresponds to time step 500; the presented residuals correspond to the u- component of the velocity field; the results for v- and w- components as well as for other time steps are similar. We observe that accurate prediction of the initial state for PCG results in significantly faster convergence. Another observation is that the rate of convergence of the residual is similar for both choices of the initial state and can be approximated by $\log_{10}(r^k_{Ns}) \propto -0.3k$. In simulations of turbulent flow in the stenosed carotid artery we have observed similar exponential convergence rate with exponent -0.3. In simulations with the diagonal preconditioner the convergence rate of the Helmholtz solver was not uniform and oscillated around $\log_{10}(r^k_{Ns}) \propto -0.021k$.

3.4 Acceleration of Poisson solver for the pressure

In order to accelerate convergence of the Poisson solver for the pressure, extrapolation of the pressure field from the solutions at the previous time step has been applied. In Fig. 6 we compare the number of iterations required by the Poisson solver and plot the convergence of the residual. Computing an initial state for the pressure solver by extrapolation indeed reduces the number of conjugate gradient iterations. In many numerical experiments we observed that improving the initial state for the Poisson solver by polynomial extrapolation with $N=2$, 3 reduced the number of iteration by approximately 50%, however, for higher order extrapolation ($N \geq 4$) the iteration count increased. An additional observation, which is quite noticeable from Fig. 6, is the different rate of the residual convergence; such behavior has not been observed in solving the Helmholtz problem.

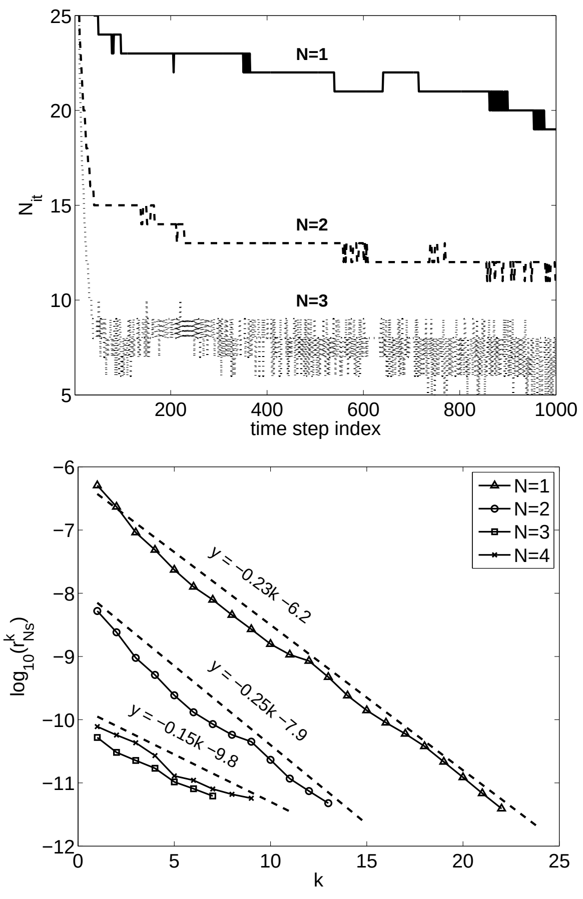


Figure 6: Flow simulations in Circle of Willis: effect of improved initial state for the pressure and convergence of residual (r^k_{Ns}) in solution with preconditioned conjugate gradient solver. Top: number of iterations required by Poisson solver for the pressure during the first 1000 time steps. Bottom: convergence of residual at time step 500. The dash lines depict the linear least square approximation for the convergence rate. $P=4$, $Nel=162{,}909$, $\Delta t=5.0E-4$, preconditioner: low energy.

3.5 Comparison of "Method 2" to method proposed in [4]

Here we consider unsteady simulation in the domain of the stenosed carotid artery and compare the efficiency and accuracy of the solution obtained with the spectral-based extrapolation (Method 2) and with the method proposed in [4]. The simulations are performed with $\Delta t=0.00005$ and $P=8$. To estimate the solution accuracy we compare the velocity flux at the inlet to flux at the outlets of the domain; larger deviation reflects the error in enforcing the incompressibility constraint. The results are summarized in Table 8.

The results show that increasing the number of the RHS, L, vectors employed by the projection method have an adverse effect on the accuracy. We note that in the solution with lower number of degrees of freedom (e.g., $P\le 6$) the difference between $|Q_{in}|-|Q_{out}|$ increased slower. Unlike the spectral extrapolation method, where approximation

Table 8: Turbulent flow simulations in stenosed carotid artery: performance of iterative solver with different approximation methods for the initial state. L-number of right-hand-sides. Q_{in}-flow rate at the inlet ($Q_{in}=62.04$), Q_{out}-sum of the flow rates at the two outlets, P_{it}, U_{it},V_{it} and W_{it}-average number of iterations required for solution of Poisson and Helmholtz equations for pressure and velocity at time steps 100 to 500. $P=8$, $\Delta t=5.0E-4$.

L	$\lvert Q_{in}\rvert - \lvert Q_{out}\rvert$	P_{it}	U_{it}	V_{it}	W_{it}
successive RHS method					
0	0.0104	19.5	16	17	14
4	0.0107	11.8	7.6	8.2	6.1
8	0.0114	10	4.9	5.2	3.9
16	0.016	9.3	3.3	3.5	2.5
32	0.489	8.8	3.2	3.3	2.5
64	unstable				
spectral extrapolation method					
$N_u(N_p)$	$\lvert Q_{in}\rvert - \lvert Q_{out}\rvert$	P_{it}	U_{it}	V_{it}	W_{it}
4(2)	0.0104	9.4	2.3	2.4	2.3
4(3)	0.0104	9.7	2.3	2.3	2.3

of the degree of freedom $(\mathbf{u}_{ap})_i^{n+1}$ depends on $(\mathbf{u}_{ap})_i^{n-k}$, $k=0,\cdots,N-1$ only, the projection method is a global method: the approximation of degree of freedom $(\mathbf{u}_{ap})_i^{n+1}$ depends on $(\mathbf{u}_{ap})_j^{n-k}$, $k=0,\cdots,L-1$, $j=0,\cdots,N_{dof}-1$, hence it is more vulnerable to accumulation of the numerical error. Moreover, the accuracy of the approximation is adversely affected by the numerical errors in the Gram-Schmidt orthogonalization. It is expected (and also consistent with our results) that these numerical errors will grow with L and N_{dof}.

In simulations with relatively low number of the RHS vectors the accuracy is not degraded, however, due to frequent restarts required by the projection method, the average number of iterations remains high. We want to emphasize that in simulations of an unsteady pipe flow in a small domain the projection method was more efficient than the spectral extrapolation in approximating the orthogonal to the main flow velocity components. The non-zero values of these velocity components are due to numerical error only, which can be amplified by the spectral extrapolation.

Combinations of the technique proposed in [4] with the spectral- and POD-based extrapolation are possible. Such combinations reduce the peak number of iterations which is present due to the restart of the Gram-Schmidt orthogonalization, and allow the use of lower number of the right-hand-sides. In some tests we have performed such combination leads to a slightly better overall performance than using either of the methods, but in other tests it leads to slightly worse overall performance.

4 Limitations

The two extrapolation-based predictors of the numerical solution at time step t^{n+1} have several limitations, namely:

• High-order extrapolation (typically with $N>4$) may increase the iteration count. As we have already mentioned, such behavior is due to the high Lebesgue constant associated with interpolation on a uniform grid (constant Δt).

• Application of the extrapolation techniques for problems converging to steady state, may also increase the iteration count. This is due to amplification (by extrapolation) of the truncation error of order $\mathcal{O}(TOL\_CG)$. Such error is the result of terminating the PCG solver before the residual reaches $r^k=0$.

• Application of $\mathbf{u}_{ap}=POD_1(\mathbf{u})$ operator in problems where the flow is unidirectional typically increases the number of PCG iterations for velocity components orthogonal to the flow direction. For example, consider a 3D simulation of a pulsatile flow in a tube, where the exact solution is $\mathbf{u}=[0,0,w(t,\mathbf{x})]$. According to $\mathbf{u}_{ap}=POD_1(\mathbf{u})$, only one correlation matrix is constructed, and, consequently, the number of the POD modes used for velocity reconstruction (Q_R) will be the same for all velocity components. We have observed that this will reduce the number of iterations required to compute the w-velocity, however, the number of iterations to compute the u- and v- components may increase. In fact, the non-zero solution for u- and v- is due to numerical error only, hence the extrapolation of these velocity components leads to the error amplification. The alternative POD-based method, denoted earlier as POD_2 has some advantages, since it allows reconstruction of the velocity vector components using separate sets of the temporal and spatial modes. We have observed that due to dynamic evaluation of the Q_R parameter the number of POD modes employed for reconstruction of the streamwise (w-) velocity is typically three to four, whereas the number of modes for reconstruction of the other two velocity components is one.

5 Conclusions

In this paper we have tested two extrapolation-based methods for accurate prediction of the initial state for the iterative solver. Both methods lead to significant reduction in the iteration count. Their performance has been studied in conjunction with the high-order spectral element spatial discretization, however, these techniques are general and can be employed with other discretization methods. For example, the proposed techniques have been successfully applied in simulation of a propagation of electrical signal in the heart, specifically in solution of the so-called bi-domain problem [13], where finite-element discretization and variable time step were employed. Over 50% savings in computational time has been reported, and it was observed that the best overall performance was achieved using Method 2 presented in this paper with the extrapolation performed in the physical space.

Research on application of methods aiming to accurately predict the solution at time step t^{n+1} should be extended to fully-implicit Navier-Stokes and other mechanics solvers for non-linear problems where a large time step can be adopted. The accuracy of the initial guess in such problems is crucial for the fast convergence of the iterative solver and it would be interesting to investigate if POD-based extrapolation is a more effective in that case unlike the cases we examined here corresponding to relatively *small* time steps.

Acknowledgments

This paper is dedicated to our teacher and colleague Professor David Gottlieb whose simple and clear thinking has been an inspiration in our own work. This work was supported by the NSF under grants OCI-0636336 and NSF OCI-0904288. Supercomputer resources were provided by the National Institute for Computational Sciences at University of Tennessee.

References

[1] R. Markovinović and J. D. Jansen, Accelerating iterative solution methods using reduced-order models as solution predictors, Int. J. Numer. Meth. Engng., 68 (2006), 525–541.

[2] D. Tromeur-Dervout and Y. Vassilevski, POD acceleration of fully implicit solver for unsteady nonlinear flows and its application on grid architecture, Adv. Eng. Software., 38 (2007), 301–311.

[3] S. Sirisup, G. E. Karniadakis, D. Xiu and I. G. Kevrekidis, Equation-free/Galerkin-free POD-assisted computation of incompressible flows, J. Comput. Phys., 2007 (2005), 568–587.

[4] P. Fischer, Projection techniques for iterative solution of $Ax=b$ with successive right-hand sides, Comp. Meth. Appl. Mech., 163 (1998), 193–204.

[5] G. E. Karniadakis and S. J. Sherwin, Spectral/hp Element Methods for CFD, Second Edition, Oxford University Press, Oxford, 2005.

[6] C. G. Canuto, M. Y. Hussaini, A. M. Quarteroni and T. A. Zang, Spectral Methods Evolution to Complex Geometries and Applications to Fluid Dynamics, Springer-Verlag, Berlin, Heidelberg, 2007.

[7] S. Sherwin and M. Casarin, Low-energy basis preconditioning for elliptic substructured solvers based on unstructured spectral/hp element discretization, J. Comput. Phys., 171 (2001), 394–417.

[8] L. Grinberg, D. Pekurovsky, S. J. Sherwin and G. E. Karniadakis, Parallel performance of a low energy basis preconditioner for spectral/hp elements, Parallel. Comput., 35 (2009), 284–304.

[9] L. Sirovich, Turbulence and dynamics of coherent structures: I-III, Quart. Appl. Math., 45 (1987), 561–590.

[10] J. L. Lumley, The structure of inhomogeneous turbulent flow, in Atmospheric Turbulence and Radio Wave Propagation, (ed. A. M. Yaglom and V. I. Tatarski), Nauka, Moscow, 160–178, 1967.

[11] G. Berkooz, P. Holmes and J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Ann. Rev. Fluid. Mech., 25 (1993), 539–575.

[12] G. E. Karniadakis, M. Israeli and S. A. Orszag, High-order splitting methods for the incompressible Navier-Stokes equations, J. Comput. Phys., 97 (1991), 414–443.

[13] L. F. Pavarino, Dipartimento di Matematica, Universitá degli Studi di Milano, private communication.