ICAS 2004 24th INTERNATIONAL CONGRESS OF THE AERONAUTICAL SCIENCES

# EVALUATION OF THE RECURSIVE PROJECTION METHOD FOR EFFICIENT UNSTEADY TURBULENT CFD SIMULATIONS

**Stefan Görtz, Joakim Möller**
**Royal Institute of Technology (KTH), 100 44 Stockholm, Sweden**

**Keywords:** *Recursive Projection Method, CFD, dual time stepping, convergence, unsteady flow, buffet.*

## Abstract

*The Recursive Projection Method (RPM) has been implemented into an unstructured CFD code to improve the efficiency of dual time stepping for unsteady turbulent CFD simulations. RPM is a combined implicit-explicit method that enhances convergence. It can easily be implemented into existing codes and the solver's existing acceleration techniques can be used without change. The method has been evaluated by computing the periodic self-induced shock oscillations over an 18% thick biconvex airfoil at 0° angle of attack, a Mach number of 0.76 and a Reynolds number of 11 million. On average, RPM accelerated the convergence of the inner loop of dual time stepping to a predefined convergence criterion by a factor of about 2.5.*

## 1 Introduction

Despite the phenomenal growth of computational fluid dynamics (CFD) in the last three decades, significant barriers still exist for routine computations of high Reynolds number, highly separated and unsteady flows [1]. Even with the continuous and steady growth in computational power realized during this period, the simulation of three-dimensional unsteady flows is still, for most cases, a computationally expensive task. Advances in more efficient algorithms for time-dependent analysis are therefore urgently needed.

The work presented here offers a means to enhance the performance of existing codes to make unsteady simulations more affordable for practical applications. A commonly used approach to solving the time-accurate Navier-Stokes equations is dual time stepping [2], where a residual equation is solved iteratively by explicit time stepping to reach a steady-state in the inner loop of every physical (outer) time step. In many cases, the inner loop is computationally too expensive because of the large number of iterations required to converge the inner loop to steady state.

One candidate for accelerating the inner loop convergence is the Recursive Projection Method (RPM). RPM was initially developed by Schroff and Keller [3, 4] for bifurcation analysis. The method has also been applied to accelerate steady-state iteration processes. It's main advantage is that it can easily be implemented on top of the native flow solver's inner iteration loop. The solver's existing acceleration techniques can be used without change. To RPM, the flow solver is simply a fixed-point scheme. By monitoring its convergence, a low-dimensional subspace associated with the dominant eigenvalues of the Jacobian of the fixed-point scheme is identified. RPM eliminates the negative influence of these eigenvalues on the fixed-point iterations by combining the original iterations scheme with a Newton iteration in the dominant subspace. The method sidesteps, therefore, the construction and solution of a large system of equations.

RPM has previously been successfully used with a structured grid CFD code for steady-state simulations [5], accelerating convergence by a factor of up to four. More recently, RPM has been implemented into a time-accurate CFD code for unstructured grids. Convergence of the inner-loop of dual time stepping could be accelerated by a factor of about two for laminar unsteady computations of the flow around a circular cylinder at a Reynolds number of 100 [6]. Here, were are more interested in applied aerdynamics problems, that is turbulent unsteady flow at higher Reynolds numbers (over $10^6$).

The paper begins with a brief description of the used flow solver, followed by a detailed presentation of RPM and its implementation into the flow solver. The efficiency of RPM for unsteady turbulent simulations is then evaluated by computing the onset of buffet over an 18% thick biconvex airfoil.

## 2 Flow Solver

As a typical current generation aeronautics CFD code, `Edge` was chosen as the flow solver for this project. `Edge` [7, 8, 9] is a three-dimensional compressible flow solver for unstructured hybrid grids of arbitrary elements, being jointly developed by the Swedish Defence Research Agency (FOI/FFA) and others, among them KTH.

The parallel flow solver is based on a node-centered finite volume scheme. For steady flows, the equations are integrated toward steady state with an explicit multi-stage Runge-Kutta scheme. To accelerate convergence, residual smoothing and a multi-grid technique can be employed. Low Mach-number preconditioning is also available.

Several different turbulence models are available. Here we use the two-equation $k-\omega$ model by Wilcox [10] combined with the explicit algebraic Reyniolds stress model (EARSM) by Wallin and Johansson [11]. The EARSM is a fully self-consistent approximation of a Reynolds stress transport model in the weak-equilibrium limit. It also behaves reasonably well in non-equilibrium flows. Moreover, the model has correct near-wall asymptotic behavior for all individual Reynolds stresses with a near-wall damping function formulated without the use of the wall skin friction (or $y^+$). This model has been shown to give results that in many aspects are very similar to those of full Reynolds stress transport models but at a computational effort that is comparable to that of standard two-equation models.

Time-accurate calculations are done either by Runge-Kutta time marching with a global time step or by implicit time marching with explicit sub-iterations (dual time stepping [2]).

### 2.1 Dual Time Stepping

For explicit time-accurate solutions, the above mentioned convergence-accelerating techniques cannot be used. Usually, explicit time stepping becomes impractical because, due to stability restrictions, the explicit time step has to be significantly smaller than the physical time scales involved.

Dual time stepping [2] is one way around this problem in that it offers higher-order time accuracy while allowing the use of efficient convergence procedures, at the cost of an additional iteration loop. The starting point is the Navier-Stokes equations in semi-discrete form:

$$\frac{du}{dt} + \mathrm{R}(u) = 0 \tag{1}$$

where $u$ is the state vector of conserved variables, and R is the net flux through the cell divided by the cell volume.

A second-order accurate implicit backward difference formula (BDF) for the time derivative in Eq. (1) yields a fully discrete approximation:

$$\frac{3}{2\Delta t}u^{n+1} - \frac{4}{2\Delta t}u^{n} + \frac{1}{2\Delta t}u^{n-1} + \mathrm{R}\left(u^{n+1}\right) = 0 \tag{2}$$

where $n$, $n-1$ and $n+1$ indicate the present, previous and future time level, respectively. Because of the presence of $\mathrm{R}\left(u^{n+1}\right)$, Eq. (2) is a nonlinear system of coupled equations, which has to be solved iteratively at each time step.

The idea behind dual time stepping is to transform Eq. (2) into a steady-state problem by adding a set of pseudo-time derivatives to the lefthand side,

$$\frac{dw}{d\tau} + \left[\frac{3w - 4u^n + u^{n-1}}{2\Delta t} + \mathrm{R}(w)\right] = 0 \qquad (3)$$

and to advance $w$ in pseudo-time $\tau$ by some suitable method until convergence is reached. Here, we employ a first-order accurate three-stage Runge-Kutta scheme with coefficients $\alpha_1 = 2/3$, $\alpha_2 = 2/3$ and $\alpha_3 = 1$. The advantage of this scheme is its low memory requirements and good smoothing properties. Existing acceleration techniques like local time stepping, multi grid or implicit schemes can be used, though with the slightly modified residual of Eq. (3). On convergence, the pseudo-temporal terms vanish, $dw/d\tau \to 0$, and "steady state" in pseudo time is approached. The original BDF is recovered and the solution is truly time-accurate.

The advantage of this method is that if the inner iterations are fully converged,

$$u^{n+1} = \lim_{\tau\to\infty}(w(\tau)) \qquad (4)$$

the full non-linear BDF is solved, giving an efficient A-stable scheme, which allows large time steps $\Delta t$. Note, however, the time step has to be small enough to resolve all physical time scales relevant to the flow problem.

In practice, the number of inner iterations is either fixed, or is controlled by some convergence criterion. The scheme becomes very expensive if a large number of inner iterations is required. Here, the idea is to reduce the number of inner iterations using RPM.

## 3 Recursive Projection Method

Consider a system of ordinary differential equations (ODEs)

$$\frac{du}{dt} = f(u),\; u \in \mathbb{R}^N. \qquad (5)$$

This system is typically the result of the discretization of a system of partial differential equations (PDEs). Note that many such systems do not give a system of ODEs but rather result in a differential-algebraic system. In this case, it is necessary to first eliminate the algebraic constraints and transform the system to a ODE system. In [12] it is shown how to do this for the Navier-Stokes equations for incompressible flow.

We start from a fixed-point iteration scheme of the type

$$u^{n+1} = F(u^n),\; u^n \in \mathbb{R}^N,\; F:\mathbb{R}^N \to \mathbb{R}^N, \qquad (6)$$

corresponding to Eq. (5) and assume that the sequence converges to the steady-state solution

$$u^n \to u^*,\; u^* = F(u^*) \qquad (7)$$

The ultimate rate of convergence is determined by the dominant eigenvalues of the Jacobian

$$J = F_u \equiv \frac{\partial F}{\partial u}(u) \qquad (8)$$

evaluated at the solution $u^*$. If all eigenvalues lie strictly within the unit circle, the scheme is asymptotically convergent in a neighborhood of $u^*$ and the linear asymptotic convergence factor is the modulus of the largest eigenvalue of $J$. Should any of the eigenvalues lie outside the unit circle, convergence is lost.

Following [3, 13, 14, 15, 16], we suppose a small number $k$ of the eigenvalues $\lambda_i$, $i = 1,\ldots,N$ of $J(u^*)$ lie outside the disk

$$K_\delta = \{|z| \le 1-\delta\}, \delta > 0 \qquad (9)$$

$$|\lambda_1| \ge \ldots \ge |\lambda_k| > 1-\delta \ge |\lambda_{k+1}| \ge \ldots \ge |\lambda_N|$$

From an orthogonal basis $V_p$ of the invariant subspace of $J$ belonging to $\{\lambda_i\}$, $i = 1,\ldots,k$, a projection $P$ is created which is used to split the solution in two parts, $p$ and $q$.

$$\begin{aligned} u^n &= Pu^n + Qu^n = p^n + q^n \\ P &= V_p V_p{}^T \\ Q &= V_q V_q{}^T = (I - V_p V_p{}^T) \end{aligned} \qquad (10)$$

In general, the basis $V_q$ and its subspace $Q\mathbb{R}^N$ are not invariant. By applying these projections to Eq. (6), the algorithm is split into two parts.

$$\begin{aligned} q^{n+1} &= QF(p^n + q^n) \\ p^{n+1} &= PF(p^n + q^n) \\ u^{n+1} &= p^{n+1} + q^{n+1} \end{aligned} \qquad (11)$$

The $Q$-part of this iteration scheme converges if $p^n$ is fixed. However, for fixed $q^n$, the $P$-part will still have very poor converge properties. RPM replaces the $P$-part of the iteration scheme with an implicit equation

$$p^{n+1} = PF(p^{n+1} + q^n) \tag{12}$$

and applies one Newton step.

$$\begin{aligned} q^{n+1} &= F(u^n) - V_p(V_p^T F(u^n)) \\ \Delta p^n &= V_p(I - V_p^T J V_p)^{-1} V_p^T (F(u^n) - p^n) \\ p^{n+1} &= p^n + \Delta p^n \\ u^{n+1} &= q^{n+1} + p^{n+1}. \end{aligned} \tag{13}$$

The convergence of this scheme is limited by the convergence behavior of the fixed-point iteration for the $Q$-projection. Assuming that most of the CPU-time is spent on computing $F(u^n)$, RPM can be implemented economically. The only major computations specific to RPM are the evaluation of the projected Jacobian matrix $V_p^T J V_p$ when $J$ is unavailable (which is usually the case). The projected matrix can be approximated by finite-differencing in the direction of the basis vectors $V_p = [v_1 \dots v_k]$

$$\begin{aligned} Jv_i &= \frac{1}{\varepsilon}\left(F(u^k + \varepsilon v_i) - F(u^k)\right) + E_i(\varepsilon), \\ E_i(\varepsilon) &= O(\varepsilon) + O(\frac{\varepsilon_F}{\varepsilon}), \;\; i = 1, \dots, k \end{aligned} \tag{14}$$

and requires $k$ function evaluations. The truncation error is $O(\varepsilon)$ and $\varepsilon_F$ is the error due to finite precision in $F$

As long as $k \ll N$, the Newton step is not computationally intensive, and the fixed-point iteration $q^{n+1} = QF(u^n)$ becomes fast. It converges, as does the overall RPM algorithm, at a rate proportional to $|\lambda_{k+1}| < 1 - \delta$. Also, the accuracy of the computation is enhanced by eliminating a substantial part of the residual in the Newton step. These two effects provides substantial convergence acceleration.

## 3.1 Basis Construction

By monitoring the convergence of the $Q$-part, we can identify the dominant eigenspace, [15, 16, 5, 3]. By a Taylor expansion of $\Delta q^n = q^{n+1} - q^n$, we derive the following relation.

$$\Delta q^n = QJQ\Delta q^{n-1} + QJP\Delta p^{n-1} + O(\Delta p^2, \Delta q^2) \tag{15}$$

If $P$ is an accurate projection of an invariant subspace of $J$, $QJP$=0. The vectors $\{\Delta q^{n-k_s+1}, \dots, \Delta q^n\}$ form an approximative $k_s$-dimensional Krylov space for $QJQ$,

$$\begin{aligned} K &= [\Delta q^n \; \Delta q^{n-1} \; \dots \; \Delta q^{n-k_s+1}] = \\ &= [QJQ\Delta q^{n-1} \; QJQ\Delta q^{n-2} \; \dots QJQ\Delta q^{n-k_s}] \end{aligned} \tag{16}$$

## 3.2 Increasing the Basis

The following ideas are used to identify the basis. Via a QR-factorization with column pivoting, such that $|r_{ii}|, \quad i = 1, \dots, k$, is decreasing, an orthogonal basis $V$ that spans $K$ is retrieved. We introduce the Krylov Acceptance ratio $k_a$ as a criteria for adding vectors to the basis. For the largest $i$ satisfying

$$\left| \frac{r_{i,i}}{r_{i+1,i+1}} \right| > k_a, \tag{17}$$

the first $i$ columns of $V$ should be added to the basis $V_p$. After orthogonalizing new vectors against the old basis $V_p$, the new basis $V_p$ is retrieved.

$$V_p = \text{orth}([V_p, v_1, \dots, v_i]) \tag{18}$$

The rate and accuracy with which the basis is identified depends on the starting vector and the spectrum. If $\Delta q^{n-k_s}$ is not rich in the desired eigenvectors, or if the gaps between the eigenvalues of the sought eigenspace and the rest is small, we expect a slow retrieval of the basis.

The $k_a$-value is problem dependent, in [14] $k_a = 25$, [3] $k_a = 1000$, [5] $k_a = O(100)$. A low value of $k_a$ will enable an early extraction of the basis, at price of poor accuracy.

Assume that the basis $V_p$ satisfies the relation $V_p R_p = X_p + S_p$, where $X_p \in \mathbb{C}^{N \times k}$ is the eigenvector-matrix associated with the dominant eigenspace and $S_p$ represents the error in the approximation. Then for the Newton-part of the

RPM-algorithm to perform well, the following condition must be fulfilled,

$$\|R_p^{-1}\|_2 \|S_p\Lambda_p - JS_p\|_2 < \min_i |1-\lambda_i|, \quad (19)$$

where $JX_p = X_p\Lambda_p$, $\Lambda_p \in \mathbb{C}^{k\times k}$, see [5] for details.

## 3.3 Decreasing the Size of the Basis

During the course of iterations the extracted basis can become unnecessarily large.

- The extracted basis contains directions for which $\lambda(H_p) < 1-\delta$, $H_p = V_p^T JV_p$ i.e. either these eigenvalues have returned to the disk $K_\delta$ due to non-linear effects or too many vectors were added to the basis due to a too low $k_a$ value.
- The accuracy of the basis is too low.

In either case these directions should be deflated from the basis. We now describe the procedure. Given the basis $V_p$ and the matrix vector product $W_p \equiv JV_p$,

$$W_p = V_pH_p + R_p \quad (20)$$

where $R_p$ is the residual. From the eigenvalue decomposition of $H_p$, $H_pZ = Z\Psi$, $\Psi = \mathrm{diag}(\psi_i)$, we can obtain information about the accuracy of the eigenspace by using that

$$\begin{aligned} \|W_pz_i - V_pH_pz_i\|_2 &= \|W_pz_i - V_pz_i\psi_i\|_2 = \\ \|J\widetilde{x}_i - \widetilde{x}_i\psi_i\|_2 &= \|R_pz_i\|_2 \end{aligned} \quad (21)$$

where $\widetilde{x}_i = V_pz_i$, and $(\widetilde{x}_i, \psi_i)$ forms a Ritz-pair. Let the permutation matrix $P \in \mathbb{R}^{k\times l}$, sort and extract the $l$ eigenvalues according to either our accuracy or magnitude requirements

$$\left(ZP, P^T\Psi P\right) \to (Z_l, \Psi_l) \quad (22)$$

To avoid complex arithmetic we use that a complex conjugated eigenpair $(z_{Re} \pm iz_{Im}, \theta \pm i\mu)$, can be expressed in real arithmetic as

$$H_p\left[z_{Re}\; z_{Im}\right] = \left[z_{Re}\; z_{Im}\right]\begin{bmatrix} \theta & \mu \\ -\mu & \theta \end{bmatrix} \quad (23)$$

By the above relation, we can transform $\Psi_l$ to a block-diagonal matrix

$$H_p\widetilde{Z}_l = \widetilde{Z}_l\widetilde{\Psi}_l \quad (24)$$

Via a QR-factorization of $\widetilde{Z}_l = Q_zR_z$,

$$W_pQ_z = V_pQ_zR_z\widetilde{\Psi}_lR_z^{-1} + R_pQ_z \quad (25)$$

the updated basis and projected Jacobian is finally computed by

$$\begin{aligned} V_p &= V_pQ_z \\ H_p &= R_z\widetilde{\Psi}_lR_z^{-1}. \end{aligned} \quad (26)$$

## 3.4 Time-Dependent RPM

Since the inner loop in the dual time-stepping consists of a steady-state calculation, the techniques described in previous chapter can applied. Discretizing the time derivative in Eq. (3), yields a fixed-point iteration, where $m$ is associated with the fictitious time $\tau$,

$$\begin{gathered} w^{m+1} = F\left(w^m, t^{n+1}\right) \\ w^m \in \mathbb{R}^N, \quad F: \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N. \end{gathered} \quad (27)$$

As the sequence converges,

$$\begin{gathered} w^m \to u^{n+1} \\ u^{n+1} = F\left(u^{n+1}, t^{n+1}\right). \end{gathered} \quad (28)$$

In order for RPM to be an effective accelerator of the fixed-point iteration, the dominant eigenspace should be small and identified fast.

Many problems suffer from clustering of eigenvalues close to the stability limit, with no clear gap to between $\lambda_k$ and $\lambda_{k+1}$, hence the identification can become cumbersome. In a steady-state simulation, one can allow spending quite a large number of iterations to obtain an accurate basis, since the tolerance requirements are still reached several times faster using RPM, see [5].

In dual time stepping, typically the inner loop tolerance is set to be a fraction of the accuracy requirements of the outer loop. Hence in order for RPM to be an effective accelerator, the identification must be fast. To quickly obtain a basis, a low $k_a$ value must be used, compared to [5]. The computed $V_p^T JV_p$ will not be very accurate, but as long as Eq. (19) is fulfilled, we still expect RPM to converge, but at a reduced rate.

### 3.4.1 Basis Handling in Time-Dependent Problems

Assume that we have computed the solution $u^n$ by RPM. At the next step $n+1$, we have two possibilities,

**1. Restart**: Before start, delete the basis $V^n$: RPM is always switched on with a fresh basis and the basis-size will be limited. As stated earlier, a downside of this approach is that it takes more time before RPM can be switched on.

**2. Saving the basis**: Take $V^n$ to approximate the new basis $V^{n+1}$, which potentially saves iterations, since a candidate for the basis is already available. The accuracy will depend on the problem and the outer time-step.

By re-using the bases, the size of the basis will grow, but with no necessary an improvement of the convergence rate since accuracy of the basis is degraded over time therefore it is necessary to introduce deflation to limit the basis size and costs.

Performance could be greatly enhanced if a number iteration steps without RPM was performed. Initially non-linear effects are strong and the matrix-vector product $J(w)V$ can differ allot from $J(w^*)V$. By a number of pre-RPM steps, the non-linear effects are damped out and RPM is more likely to perform well, see [6].

A remedy for the inaccuracy could be some matrix-free update techniques, e.g. Krylov methods. They also suffer from convergence problems when no clear separation between $\lambda_k$ and $\lambda_{k+1}$ exists. Furthermore, the extra function evaluations do not advance the solution closer to the steady state. In our numerical experiments, we also found that the extra gain in accuracy could not compensate for the extra costs, compared to using the restart approach.

### 3.4.2 A Combined Update/Restart Approach to Basis Handling

Assume that the solution $u^n$, a basis $V^n$ of size $k_n$ and its accuracy estimate $\omega_V$, see Eq. (29), is available from step $n$. At step $n+1$:

1. Perform $r$ pre-RPM (smoothing) steps.
2. At the cost of $k_n$ function evaluations, update $V_p^T J V_p$ and check basis accuracy by Eq. (21). If the accuracy estimate is violated, $\|R_p z_i\|_2 > \omega_V$ for some $i \in [1, k_n]$, then that column of $V_p$ is removed. Furthermore since the basis has been corrupted, the entire basis will be discarded at the end of step $n+1$, resulting in a restart at step $n+2$.
3. In the end of step $n+1$, when the solution has converged, we discard any basis vector associated with an $|\lambda_i| < \beta$, $i = 1, \dots, k_{n+1}$, see section 3.3, in order to limit the growth of the basis size. We used a $\beta = 0.9$ in our calculations.
4. Finally we computed the accuracy estimate of the overall basis $V^{n+1}$, $\omega_V$

$$\omega_V \equiv \left( \min_i \|R_p z_i\|_2 \right)^{-1} \tag{29}$$

For numerical experiments, see section 5.5.

## 4 Implementation Issues

RPM treats the flow solver as a "black-box". It sends a vector $u$ into the flow solver, which first updates the boundary conditions and then computes and returns $F(u)$ by doing a specified number of iterations of the native scheme (called function evaluations in the following). Note that in order for the flow solver to be viewed as a fixed-point scheme, $F(u)$ must not depend on previous iterations. For example this places a restriction on the boundary conditions that can be used. The number of function evaluations each RPM iteration is controlled by the parameter NSTEP.

A `Matlab` implementation of RPM was linked to `Edge` by an interface. First, `Edge` writes the state vector to a file in the native `Edge` FFA data format. This file is then read into `Matlab` by the RPM process running parallel to `Edge`, using the FFA-`Matlab` Toolbox [17]. After being processed and updated by the RPM process, the state vector is written to file and read by the waiting (idle) Edge process. The read-write process is coordinated using a flag file that can be read by both Edge and RPM.

### 4.1 Increased Robustness by Scaling Variables

In [5], it was found that scaling the variables increased the robustness of RPM. In laminar cases, the pressure component is in general several orders of magnitudes larger than the velocity- and density-components. Here, $\Omega$ is the dominent component. The resulting projections are very skewed and sensitive to inaccuracy. This problem could be removed by scaling the components to be O(1).

Here, the scaling factors are determined at the beginning of each new outer time-step. If $w_n^* = (\rho, \mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{p}, \mathbf{k}, \omega)^T$ is the converged solution at step $n$, then the scaling factors become

$$\begin{aligned} \rho_0 &= \max(\rho, 1) \\ u_0 &= \max(|\mathbf{u}|, |\mathbf{v}|, |\mathbf{w}|, 1) \\ p_0 &= \max(\mathbf{p}, 1) \\ k_0 &= \max(\mathbf{k}, 1) \\ \omega_0 &= \max(\omega, 1) \end{aligned} \tag{30}$$

and the scaled initial solution at step $n+1$ is

$$\begin{aligned} \tilde{w}_{n+1}^0 &= S^{-1} w_n^* \\ S &= \mathrm{diag}(\rho_0, u_0, p_0, k_0, \omega_0) \end{aligned} \tag{31}$$

Note that only RPM and not Edge, the flow solver, is aware of the scaling. Assume that a function evaluation in Edge is given by the solution operator $\mathcal{F}, \mathcal{F} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$, then one function evaluation in RPM is given by

$$\tilde{w}^{m+1} = S(\mathcal{F}(S^{-1}\tilde{w}^m)). \tag{32}$$

## 5 Application: Flow around Biconvex Airfoil

We consider the periodic self-excited turbulent flow around an 18% thick circular-arc airfoil in free flow at $M_\infty = 0.76$, $\alpha = 0°$ and $Re = 11 \times 10^6$. At these conditions, the transonic flow results in a periodic, 180° out-of-phase motion of the shocks over the upper and lower surface of the airfoil. The unsteadiness is driven by the interaction between the shocks, the boundary layer and the vortex shedding in the wake. Experimental data [18, 19, 20] as well as previous computations [18, 19, 21, 22] are available for comparison.

The demanding nature of this flow problem was illustrated by Wang *et al* [22], who performed a convergence investigation using the structured EURANUS code [23]. As many as 260 multi grid iterations were required in the inner loop of dual time stepping for each of the 50 outer time steps per period to converge the residual by one order of magnitude. When the number of outer time steps per period was increased to 350, still as many as 60 inner iterations were required to fulfill the convergence criterion. It is anticipated that RPM can significantly reduce this number and make such simulations more affordable.

### 5.1 Numerical Grid and Boundary Conditions

A 2D hybrid grid with 12,000 nodes was generated using the commercial grid generation software ICEM CFD, see Fig. 1. The circumference

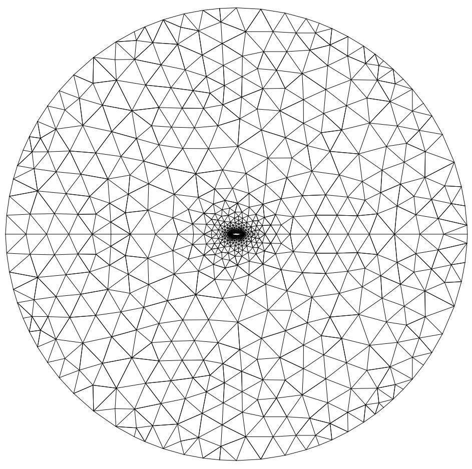

**Fig. 1** Hybrid Grid for Biconvex Airfoil

of the airfoil is discretized with 216 grid points. There are 50 layers of quads, compare Fig. 2. The spacing of the first grid point normal to the solid wall is $2.0 \times 10{-}6$ m, giving $y^+ < 1$. Away from the wall, the spacing increases by a ratio of 1.2. The circular outer boundary is located 25 root chord lengths away from the airfoil. The boundary condition on the airfoil is a solid wall. Characteristic variable freestream conditions are used on the outer boundary.

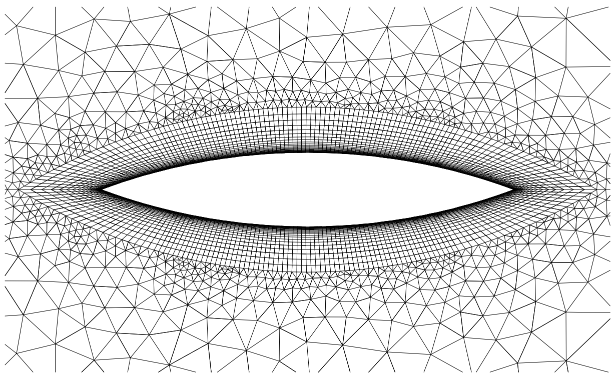

**Fig. 2** Close-up of Numerical Grid

The far-field boundary condition with vortex correction in 2D used in [6] was not used here because it is based on information from the previous iteration, rendering Edge a non-fixed-point solver. By comparing the two boundary conditions it was found, however, that this had a negligible influence on the accuracy of the extracted basis and no impact on the final (converged) solution.

## 5.2 Numerical Parameters

The unsteady flow was calculated with `Edge` using dual time stepping and the EARSM turbulence model. Previous computational results [22] for the same geometry and flow conditions demonstrated that the EARSM predicts the unsteadiness due to strong shock-boundary layer interaction better than both algebraic turbulence models and two-equation models based on Boussinesq's hypothesis. Fully turbulent flow was assumed. A first order upwind scheme were used to discretize the turbulence equations.

The outer (physical) time step was set to $\Delta t = 0.0001$ s ($\Delta t^* = \Delta t/(cU_\infty) = 0.026$), corresponding to about 250 time steps per period of oscillation. The (inner-loop) CFL number was set to 0.7. Three multi-grid levels and residual smoothing were used in the inner loop.

The unsteady numerical simulation was initialized by a steady-state solution. A total of 2,000 outer time steps, corresponding to 0.2 s of simulation time, were computed. For every outer time step, the absolute residual of the inner loop,

$$\text{res}_{\max} = \max_i |\text{res}(\rho_i)| \tag{33}$$

where $i$ is the node number, was decreased by two-and-a-half orders of magnitude, $\log_{10}(\text{res}_{\max_1}/\text{res}_{\max_n}) = -2.5$. For this particular flow problems, this was found to be suffcient to converge the aerodynamic coefficients within the inner loop. The number of inner iterations was not limited.

The simulation without RPM took about 180 h of wall-time on one Pentium Xeon CPU.

## 5.3 Flow-Field Results

The Navier-Stokes code reproduced the time-dependent aspects of the onset of buffet. Figure 3 shows instantaneous Mach number contours at different times. Coalescence of the near-vertical contours over the latter half of the airfoil indicates the formation of a shock-wave. It forms near the trailing edge just above a region of trailing-edge separation. Its strength increases as the local velocity ahead of the shock increases. The increased strength gives rise to shock-induced separation, and the shock wave and separated region begin to move forward. The local surface velocities upstream of the shock continue to increase and stabilize in a maximum velocity distribution. As the shock continues forward into a region of locally lower velocities, it diminishes in strength and vanishes as the separation point reverts to the trailing edge to complete the cycle. Meanwhile, the identical process is occurring on the lower surface 180° out of phase.

This periodic phenomenon causes oscillations in the aerodynamic forces. The time histories of the lift coefficient $C_L$ and the drag coefficient $C_D$ are shown in Fig. 4. The time histories are identical for both the simulation with and

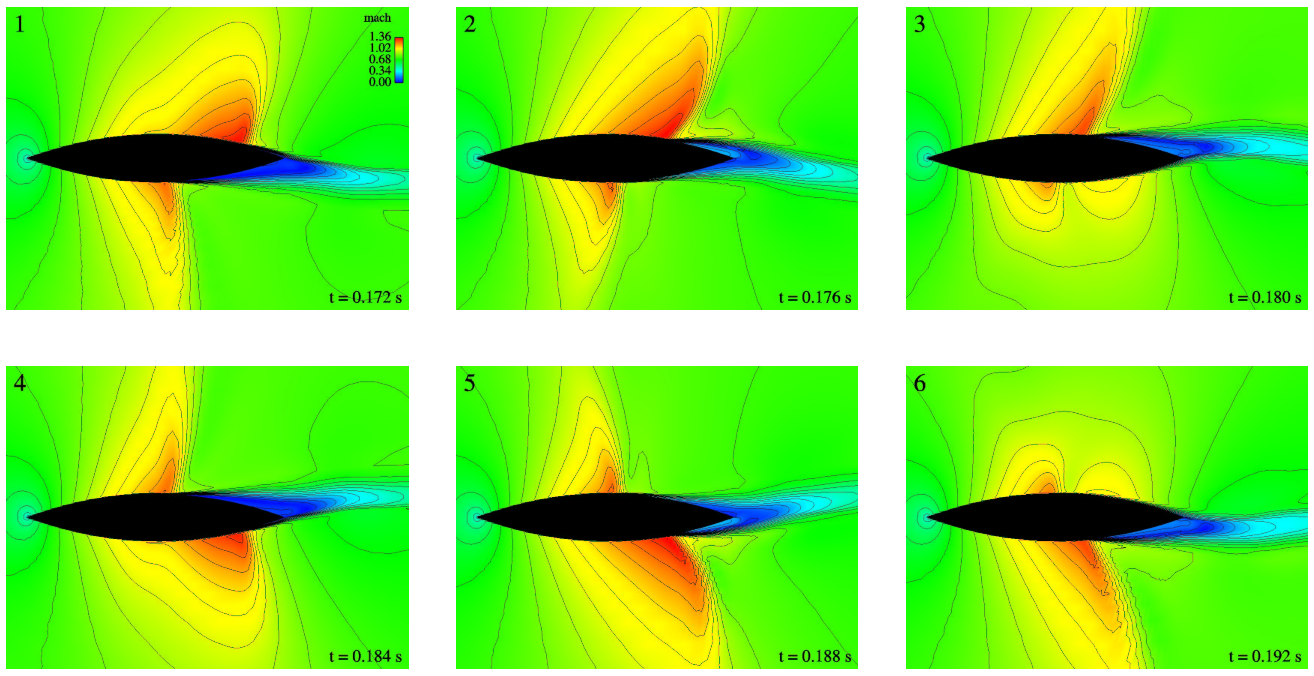

**Fig. 3** Instantaneous Mach Number Contours in the Flow Field about the Circular Arc Airfoil, showing Oscillatory Separation; $M_\infty = 0.76$, $Re = 11 \times 10^6$

without RPM, because the same convergence criterion was applied in the inner loop.

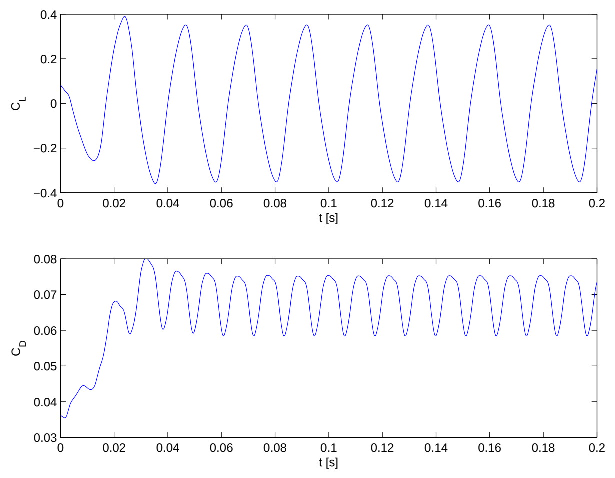

**Fig. 4** Lift and Drag Coefficient versus Time; $M_\infty = 0.76$, $Re = 11 \times 10^6$, $\Delta t = 0.0001$ s

After the transients have decayed after about 0.04 s, the lift coefficient oscillates almost sinusoidally around a zero mean value with an amplitude of about 0.35. The drag coefficient scillies with twice that frequency around a mean value of 0.067 and an amplitude of 0.008. The computed reduced frequency of the lift coefficient is $k = \pi f c / u_\infty = 0.549$. The experimental value is $k_{\text{exp}} = 0.49$ [20].

The difference in the computed frequency is due to the first-order upwind discretization of the turbulence equations. Note that the computed reduced frequency was $k = 0.485$ when we used a second-order discretization. This is within 1% of the experimental value! However, the second-order upwind discritization changed the dynamics and the spectrum of the problem in such a way that RPM could not identify a basis. We also experimented with a second-order central scheme, with the same result. This demonstrates the complicated nature of this problem.

Since we are mainly interested in the convergence properties of the native iteration scheme with and without RPM, rather than the dynamics of this particular flow problem, we only present results obtained with the first-order upwind discretization in the following.

## 5.4 RPM with Restart: Renewing the Basis Every Outer Time Step

In all cases presented below, the absolute residual of the inner loop was converged until the criterion $\log_{10}(\mathrm{res}_{\mathrm{max}_1}/\mathrm{res}_{\mathrm{max}_n}) = -2.5$ was fulfilled. The size of the Krylov space was set to $k_s = 10$. Every $k_i = 10$ RPM iterations, RPM seeked to identify or increase the basis $V$. The value of ε in Eq. (14) was set to $10^{-8}$, apart from the first cases, where we used $10^{-6}$. However, the influence of ε on the accuracy of the basis, and thus the convergence rate, was found to be neglible.

First, we experimented with a Krylov acceptance ratio of $k_a = 20$ and two function evaluations (flow solver iterations) each RPM iteration, NSTEP=2. The corresponding number of inner-loop function evaluations (iterations) are presented as a function of the number of outer time steps in Fig. 5. Only the first 1,000 time steps are shown for the sake of clarity. The dashed red and solid blue lines correspond to computations with and without RPM, respectively.

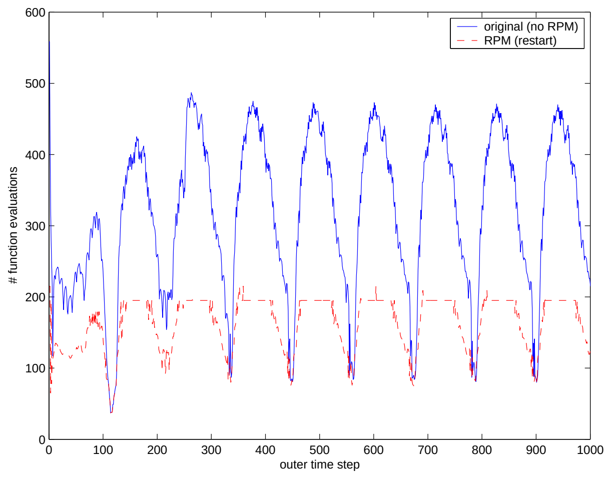

**Fig. 5** Number of Inner-Loop Function Evaluations with and without RPM versus Number of Outer Time Steps; $k_a = 20$, NSTEP=2

The periodic behavior of the flow is reflected by the time history of convergence of the inner-loop. The results for the simulation without RPM indicate that portions of the buffet cycle are more difficult to resolve in time, resulting in up to 490 inner iterations needed to fulfill the inner-loop convergence criterion. Other portions of the cycle are resolved with as little as 80 inner iterations. The average number of inner iterations over the entire length of the simulation is 324.

RPM is triggered when many inner-loop iterations are required. It reduces the maximum number of inner-loop function evaluations to about 195 or less. This correspond to a maximum inner-loop convergence acceleration by a factor of 2.5. RPM does not provide any benefit when 100 or less inner iterations are sufficient for convergence. The average number of inner iterations is 164, corresponding to an average speed-up ratio is 1.98.

Typically, RPM extracted two eigenvalues after 50 RPM iterations (corresponding to 104 inner iterations). During the portions of the buffet cycle that are more difficult to resolve in time, RPM usually found another three eigenvalues after 90 RPM iterations (184 inner iterations), resulting in a basis size of $p = 5$. Thus, the condition $p \ll N$ was well fulfilled.

### *5.4.1 Influence of RPM Parameters on Inner-Loop Convergence*

In a next step, we investigated the influence of the Krylov acceptance ratio, $k_a$, and the number of function evaluations each RPM iteration, NSTEP, on the performance of RPM. In Fig. 6 we compare the results for .discussed above (red dashed line) with those obtained for $k_a = 50$ and NSTEP=1 (green solid line with circles). For the sake of clarity, only the first 400 outer times steps are presented here.

It can be seen that during the transient part of the simulation, it pays off to use a larger Krylov acceptence ratio together with one function evaluation each RPM iteration. The pay off is less after the transients have decayed. On average, 10% less inner iteration were required with this parameter settings compared to the case with $k_a = 20$ and NSTEP=2. The average speed-up ratio is 2.15.

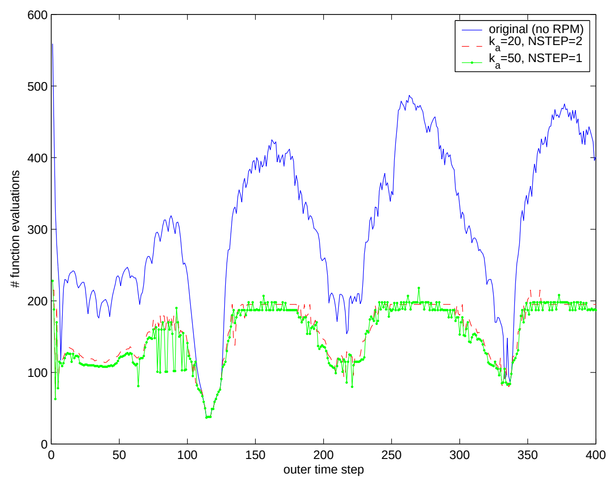

**Fig. 6** Influence of Parameters $k_a$ and NSTEP on Number of Inner-Loop Function Evaluations

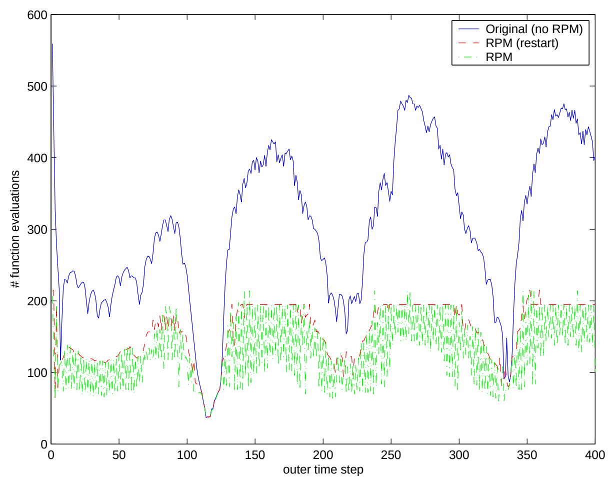

**Fig. 7** Comparison between Restart and Restart/Update Approach; $k_a = 20$, NSTEP=2

### 5.5 Saving the Basis: Reusing the Basis from a Previous Outer Time Step

In Fig. 7 we compare the restart approach with the combined restart/update approach described in section 3.4.2. As before, for the sake of clarity, only the first 400 outer times steps are presented here. The blue solid line represents the number of inner iterations versus the number of outer time steps without RPM. The red dashed and green dot-dashed lines denote the total number of function evaluations with the restart approach and the combined restart/update approach, respectively. The RPM results were obtained for a Krylov acceptance ratio of $k_a = 20$ and NSTEP=2 function evaluations each RPM step. We used 20 pre-RPM (smoothing) steps in the combined restart/update approach.

The numerical experiments substantiated that there is a small benefit to be gained from reusing an old basis. For some outer time steps the inner loop converged up to four times faster with RPM than without RPM! The average speed-up was by a factor of 2.45. However, the basis was usually kept for only one outer time step before it violated the accuracy criterion and was rejected.

The results of all three numrical experiments are summerized in Table 1 together with the relevant RPM parameters.

## 6 Conclusions

The Recursive Projection Method (RPM) has been implemented into a CFD code for unstructured grids to accelerate the inner-loop convergence of dual time stepping. The modified code was used to compute the unsteady turbulent buffet flow around an 18%-thick biconvex airfoil at transonic speed. On average, the inner loop of the modified code converged two times faster to a predefined convergence criterion than the original code. An average acceleration by a factor of 2.5 could be achieved by reusing and updating information that RPM extracted at previous outer

| | case 1 | case 2 | case 3 |
|---|---|---|---|
| $\varepsilon$ | $10^{-6}$ | $10^{-8}$ | $10^{-8}$ |
| $k_s$ | 10 | 10 | 10 |
| $k_i$ | 10 | 10 | 10 |
| $k_a$ | 20 | 50 | 20 |
| NSTEP | 2 | 1 | 2 |
| smoothing steps | - | - | 20 |
| aver. # inner iter. | 164 | 151 | 131 |
| aver. speed-up | 1.98 | 2.15 | 2.45 |

**Table 1** Summary of RPM Parameters, Average Number of Inner Iterations and Average Speed-Up

time steps. The additional computational cost that RPM incurs was shown to be neglible. In summary, for unsteady turbulent computations, the overall efficiency of the code was more than doubled by using RPM.

## 7 Acknowledgements

We would like to thank Jesper Oppelstrup and Arthur Rizzi, KTH, for supervising this work; Michael Hanke, KTH, for valuable comments; Jonathan Smith, Swedish Defence Research Agency (FOI), for writing the `Edge-Matlab` interface and for fruitful discussions on `Matlab` programming and the FFA data format; Peter Eliasson and Stefan Wallin, FOI, for helping with `Edge`. Financial support from NADA/PSCI is gratefully acknowledged.

## References

[1] V. N. Vatsa and B. A. Singer. Evaluation of a Second-Order Accurate Navier-Stokes Code for Detached Eddy Simulation Past a Circular Cylinder. *AIAA Paper 2003-4085*, 2003.

[2] A. Jamason. Time dependent calculations using multigrid, with application to unsteady flows past airfoils and wings. *AIAA Paper 91-1596*, 1991.

[3] G. M. Schroff and B. Keller. Stabilization of Unstable Procedures: The Recursive Projection Method. *SIAM J. Numer. Anal.*, 30(4):1099–1120, August 1993.

[4] H. B. Keller. RPM - A Remedy For Instability. In D. Estep and S. Tavener, editors, *Collected Lectures on the Preservation of Stability under Discretization*, Proceedings in Applied Mathematics 109, pages 185–196. SIAM, 2002.

[5] J. Möller. *Studies of Recursive Projection Methods for Convergence Acceleration of Steady State Calculations*. Licentiate thesis, Royal Institute of Technology (KTH), Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden, 2001. TRITA-NA-0119.

[6] S. Görtz and J. Möller. Recursive Projection Method for Efficient Unsteady CFD Simulations. *ECCOMAS 2004*, 2004.

[7] P. Eliasson. EDGE - A Navier-Stokes Solver for Unstructured Grids. Technical Report FOI-R-0298-SE, Swedish Defence Research Agency (FOI), Division of Aeronautics (FFA), Stockholm, Sweden, December 2001.

[8] P. Eliasson. EDGE, a Navier-Stokes solver for unstructured grids. In *Proc. To Finite Volumes for Complex Applications III*, pages 527–534, 2002.

[9] Swedish Defence Research Agency (FOI), Aeronautics Div. (FFA). Edge 3.1 Installation and User Guide. http://www.edge.foi.se, October 2003. accessed 03/2004.

[10] D. C. Wilcox. Reassessment of the Scale Determining Equation for Advances Turbulence Models. *AIAA Journal*, 6(11), 1988.

[11] S. Wallin and A. Johansson. A Complete Explicit Algebraic Reynolds Stress Model for Incompressible and Compressible Turbulent Flows. *Journal of Fluid Mechanics*, 403:89–132, 2000.

[12] B. Keller and H. von Sosen. New Methods in CFD: DAE and RPM. In *Proceedings of the First Asian CFD conference*, pages 17–30, Hong Kong, 1995.

[13] B. D. Davidson. Large-Scale Continuation and numerical Bifurcation for Partial Differential Equation. *SIAM J. Numer. Anal.*, 34:2008–2027, 1997.

[14] M. Dorobantu, K. Lust, C. Jacobson, and A. Khibnik. The Recursive Projection Method for Acceleration of Iterative Methods. Technical report, United Technologies Research Center, 1999.

[15] P. Love. *Bifurcation in Kolmogorov and Taylor-Vortex Flows*. Phd thesis, California Institute of Technology, Pasadena, California, 1999.

[16] K. Lust, D. Roose, A. Spence, and A. R. Champneys. An Adaptive Newton-Picard Algorithm with Subspace Iteration for Computing Periodic Solutions. *SIAM J. Sci. Comp.*, 19:1188–1209, 1998.

[17] Swedish Defence Research Agency (FOI), Aeronautics Div. (FFA). FFA Matlab Toolbox. http://www.edge.foi.se/matlab_tools/index.html, 2003. accessed 03/2004.

[18] L. L. Levy Jr. Experimental and Computational Steady and Unsteady Transonic FLows about a Thick Airfoil. *AIAA Journal*, 16(6):564–572,

June 1978.

[19] H. L. Seegmiller, J. G. Marvin, and L. L. Levy Jr. Steady and Unsteady Transonic Flow. *AIAA Journal*, 16(12):1262–1270, December 1978.

[20] J. McDevitt. Supercritical Flow about a Thick Cicrular-Arc Airfoil. NASA TM-78549, 1979.

[21] C. L. Rumsey amd M. D. Sanetrik, R. T. Biedron, N. D. Melson, and E. B. Parlette. Efficiency and Accuracy of Time-accurate Turbluent Navier-Stokes Computations. *AIAA Paper 95-1835-CP*, 1995.

[22] D. Wang, S. Wallin, M. Berggren, and P. Eliasson. A computational study of unsteady turbulent buffet aerodynamics. *AIAA Paper 2000-2657*, 2000.

[23] A. Rizzi, P. Eliasson, I. Lindblad, C. Hirsch, C. Lacor, and S. Häuser. The Engineering of Multiblock/Multigrid Software for Navier-Stokes Flows on Structured Meshes. *Computers & Fluids*, 22:341–367, 1993.