Development of an efficient bifurcation tracking method

S.J. Huntley[a,*], D.P. Jones[a], A.L. Gaitonde[a]

[a]*Aerospace Engineering Department, University of Bristol, Queens Building, University Walk, BS8 1TR, United Kingdom*

Abstract

The buffet onset boundary is associated with a change in stability of the flow solution. The buffet onset boundary has been computed for a NACA 0012 aerofoil using a modified bifurcation tracking method. The algorithm combines the projection aspect of the Recursive Projection Method with two different direct bifurcation tracking methods. This considerably reduces the cost of the bifurcation tracking methods by solving for the bifurcation point on a small subspace of the system containing only the least stable dynamics. The method has been extended to allow the boundary to be computed as geometrical parameters, camber and thickness, are changed. This enables rapid evaluation of the effect of different aerofoil designs on the boundary. Results are presented comparing the full bifurcation tracking methods to their projected equivalent and show that although the projected methods lack the numerical accuracy of the full counterpart, the trends of the boundaries agree well.

Keywords: buffet, bifurcation, continuation, flow stability, Navier-Stokes

1. Introduction

The behaviour of flow that is inherently non-linear in nature is dependent on the values of certain parameters. At critical parameter values this can result in a change in stability, known as a bifurcation point. Identification of bifurcation points is necessary to understand these non-linearities and tracking the path of the bifurcation point as it varies with two parameters allows the behavioural response of the system under a range of different

[*]Corresponding author

Email address: samantha.huntley@bristol.ac.uk (S.J. Huntley) conditions to be determined. Algorithms to compute a bifurcation point can be categorised as one of two types: an indirect method or a direct method.

In the literature, the most common approach is to use an indirect method. This is usually the cheapest way of finding a bifurcation point. An indirect method typically involves applying a test function at each equilibrium point calculated by a continuation method. The test function must be defined so that it has a value of zero at the bifurcation point and is continuous in an interval of parameter values around this point [1, 2]. It is also beneficial to define a test function that invokes a change of sign as a bifurcation point is passed [3]. As an example, a test function could be the maximum value of the real parts of all the eigenvalues of the Jacobian [4, 5].

One alternative to using a test function to indirectly detect a bifurcation is to perform a time-accurate simulation at each equilibrium within the continuation method. This is done by applying a small perturbation to the steady solution. If the solution is unstable then this perturbation will cause the system to diverge. There are a number of drawbacks to this approach as the divergence can be slow when close to the bifurcation, which results in long run times and the perturbation size must be controlled to ensure it is not so large that the system converges onto a different, nearby stable solution [6].

The other category of algorithms used to compute a bifurcation are termed direct methods. Indirect methods are useful for monitoring the presence of a bifurcation and can give an approximation to the bifurcation point, however when it is important to locate the bifurcation accurately it is necessary to use a direct method as the chances of computing a bifurcation point exactly during the continuation method is small [3]. With direct methods, a bifurcation is computed by defining a system whose solution is a bifurcation point. This results in a more accurate solution than the indirect methods achieve but has a higher computational cost. The defining system will be dependent on the type of bifurcation to be computed and on how the underlying system is discretised in time. Most systems come under one of two categories: the "augmented system" and the test function.

The augmented system works by adding equations that define the bifurcation of interest to the base or flow equations [4]. For a Hopf bifurcation, which is the focus of this work, there are four additional equations corresponding to the complex eigenvalue equation (which is two equations using real-valued variables); an equation to fix the phase of the eigenvector and one to fix the amplitude [7]. There are several existing algorithms to solve the augmented system for a Hopf bifurcation such as the method of Griewank and Reddien [8], Roose and Hlavacek [9] and Poliashenko and Aidun [5]. A direct method involving test functions utilizes the same test functions as for the indirect method but in a different way. In a direct method they are used in combination with a bordered matrix [4] to create a system, whose solution is a bifurcation. This type of system is also known as a minimally augmented systems [3]. Both of these direct methods use an iterative solver and as such need an accurate initial value. This involves finding an initial guess to the critical eigenvalue, which can be difficult as usually the evolution of eigenvalues with a parameter is not linear [10]. We compute an initial guess for the direct bifurcation methods by employing a continuation method until a bifurcation has been detected.

Direct methods increase the size of the system by adjoining extra equations onto the mean flow equations. For systems that are already large, this increase means that solving the full system in a direct manner would be infeasible. Bordering algorithms can be employed to reduce the size of the solve required such as is implemented in the LOCA package [7]. For a Hopf bifurcation, this results in the minimum solve being twice the size of the Jacobian of the flow variables, although the matrix is sparse. This means that it is possible to take advantage of existing direct solvers designed for sparse systems such as the sparse unsymmetric multifrontal method called Umfpack [11, 12, 13, 14]. Furthermore, using a bordering algorithm means that matrix-vector products can be used to reduce the storage requirements rather than constructing the full Jacobian and Hessian matrices.

Studies concerned with the stability of transonic flows associated with aircraft aerodynamics mainly use a method known as global-stability analysis. Examples of this in the literature include the studies by Crouch *et al.* [15, 16, 17], where global stability analysis is used to investigate the shock-buffet problem. Global stability analysis requires linearization around a basic state, the choice of which can be complicated [18]. Furthermore, computations involving the linearized Navier-Stokes equations are expensive and investigations into changes in design variables cannot be easily performed using global stability analysis unlike for direct bifurcation tracking methods. Therefore the use of a computationally efficient direct bifurcation tracking method would be a desirable alternative. The application of direct bifurcation tracking methods to large systems in aerodynamics has mainly been limited to enclosed or internal flows. However, a few studies do exist, primarily concerned with bifurcation tracking of transonic flows of aeroelastic systems such as those by Badcock *et al.* [19, 20, 21, 22]. These studies focused on the computation of the stability limit of aeroelastic systems. They used analytical Jacobians and performed simulations on systems that were smaller in size than those used in this work.

In this work both time-independent systems of the form

$$\mathbf{R}^*(\mathbf{Y}) = 0, \tag{1}$$

where $\mathbf{R}$ is the residual and $\mathbf{Y}$ is the vector of unknowns, and time-discrete systems given by

$$\mathbf{Y}^{\nu+1} = \mathbf{F}(\mathbf{Y}^{\nu}), \tag{2}$$

where $\mathbf{F}$ is the fixed point iteration function and ν is the iteration counter, are investigated. This means that two different continuation methods are required to compute the initial guess for the bifurcation tracking methods depending on whether the system is discretised in time or not. For the time-independent case the continuation method used is the one described by Wales *et al.* [23]. For the time-discrete system, the Recursive Projection Method [24] is used to compute the solution branch as a parameter is varied. The RPM has been used to accelerate the convergence of steady state simulations [25, 26] and to perform coarse stability analysis using microscopic evolution rules [27, 28]. However, the application of the RPM as a means to facilitate a continuation method for large systems is limited. Its application as a stabilisation method for large systems is typically limited to single solutions, such as its use by Campobasso and Giles [29], whilst its use in a continuation method has been applied to small systems only.

This paper presents a new, computationally inexpensive method to directly locate and track bifurcation points. This is accomplished by combining aspects of the Recursive Projection Method with existing direct bifurcation tracking methods. Initially, the Recursive Projection Method has been tested as a continuation method for cases involving a NACA 0012 aerofoil displaying shock oscillations. The Recursive Projection Method allows continuation to be performed with a large time-discrete RANS system. This extends the applicability of the underlying time integration scheme and improves convergence rate.

For the bifurcation tracking methods, results are presented using the Spalart-Allmaras turbulence model and a number of different aerofoils that are known to exhibit the flow behaviour of interest. The results show that the new, inexpensive bifurcation tracking methods developed in this work offer a low-cost way of understanding the dependency of bifurcation points as two parameters are varied.

2. CFD Model

The CFD model forms the basis for the continuation method and subsequently the bifurcation tracking methods implemented in this work. This section describes the formulation of the Reynolds Averaged Navier-Stokes (RANS) equations along with the Spalart-Allmaras turbulence model for the Reynolds stresses that was used to model the CFD. This follows the method used in [30] which has been verified for similar problems to those of interest in this work. However, as the primary focus of this work is to compare the computationally inexpensive bifurcation tracking methods to their standard counterpart, the accuracy of the computed stability boundary is not of major concern and, as such, a detailed validation of the solver and numerical setup was not necessary.

2.1. Governing equations

The governing equations are the two-dimensional RANS equations. These are written in conservative form as

$$\frac{\partial \mathbf{Y}}{\partial t} + \frac{\partial \mathbf{C}}{\partial x} + \frac{\partial \mathbf{D}}{\partial y} = 0 \tag{3}$$

where $\mathbf{Y}$ is the vector of unknowns, t is time, $\mathbf{C}$ and $\mathbf{D}$ are flux vectors and x and y are the Cartesian coordinates.

The vector of unknowns, $\mathbf{Y}$, is given by

$$\mathbf{Y} = (\rho, \rho u, \rho v, \rho E)^T . \tag{4}$$

Here ρ is the density, u and v are the velocity components in the x and y direction respectively and E is the total energy calculated by

$$E = e + 1/2(u^2 + v^2) \tag{5}$$

where e is the internal energy.

The flux vectors $\mathbf{C}$ and $\mathbf{D}$ are made up of inviscid and viscid components.

$$\mathbf{C} = \mathbf{C^I} - \mathbf{C^V}, \;\; \mathbf{D} = \mathbf{D^I} - \mathbf{D^V}. \tag{6}$$

The inviscid flux vectors are given by

$$\mathbf{C^I} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho v u \\ (\rho E + p)\, u + p \end{bmatrix}, \quad \mathbf{D^I} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (\rho E + p)\, v + p \end{bmatrix} \tag{7}$$

and the viscous flux vectors are given by

$$\mathbf{C^V} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{bmatrix}, \quad \mathbf{D^V} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} - q_y \end{bmatrix}. \tag{8}$$

In the preceding equations p is the pressure and is related to the internal energy by $p = (\gamma - 1)\left(\rho E - \frac{\rho(u^2+v^2)}{2}\right)$. The components of the stress tensor are given by

$$\tau_{xx} = \frac{2}{3Re}(\mu + \mu_t)\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right),$$

$$\tau_{yy} = \frac{2}{3Re}(\mu + \mu_t)\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right),$$

and

$$\tau_{xy} = \tau_{yx} = \frac{1}{Re}(\mu + \mu_t)\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right).$$

The heat flux vectors are given by

$$q_x = -\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_T}\right)\frac{1}{[\gamma - 1]M_\infty^2 Re}\frac{\partial T}{\partial x},$$

and

$$q_y = -\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_T}\right)\frac{1}{[\gamma - 1]M_\infty^2 Re}\frac{\partial T}{\partial y}.$$

Re is the Reynolds number, T is the temperature, M_∞ is the freestream Mach number, Pr and Pr_T are the laminar and turbulent Prandtl number, respectively, and μ and μ_t are the laminar and turbulent eddy viscosity. The laminar eddy viscosity is calculated using Sutherland's law

$$\mu = T^{\frac{2}{3}}\left(\frac{1 + S}{T + S}\right) \tag{9}$$

where $S = \frac{110}{T_\infty}$ and T_∞ is the freestream temperature. The turbulent eddy viscosity is given by the Spalart-Allmaras tubrulent model described below.

2.2. Spalart-Allmaras turbulence model

The Spalart-Allmaras one-equation turbulence model has been used to compute the turbulent eddy viscosity. The turbulent viscosity is described by the scalar transport equation, which is given as

$$\frac{\partial \nu}{\partial t}+U_i\cdot\nabla\nu-\frac{1}{\sigma Re}(\nabla\cdot((\tilde{\nu}+\nu)\nabla\nu)+c_{b2}\left|\nabla\nu\right|^2)-c_{b1}\tilde{\omega}(\nu)\nu+c_{w1}f_w(\nu)\left(\frac{\nu}{d}\right)^2=0. \tag{10}$$

where $\tilde{\nu}$ is the molecular viscosity. This equation solves for the variable ν, which is related to the turbulent eddy viscosity by $\nu_T = \nu f_{v1}(\nu/\tilde{\nu})$. Other relevant equations are:

$$\tilde{\omega}(\nu)=\omega+\frac{\nu}{\kappa^2 d^2}f_{v2}(\nu/\tilde{\nu}),\quad f_{v2}(x)=1-\frac{x}{1+xf_{v1}(x)},\quad f_{v1}(x)=\frac{x^3}{x^3+c_{v1}^3},\quad x=\frac{\nu}{\tilde{\nu}},$$

$$\omega=\left|\left|\nabla\times U_i\right|\right|^2,\quad f_w(g)=g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6},\quad g(r)=r+c_{w2}(r^6-r),\quad r(\nu)=\frac{\nu}{\tilde{\omega}\kappa^2 d^2}$$

The constants [31] are given by:

$$c_{b1}=0.1355,\ c_{b2}=0.622,\ \sigma=2/3,\ c_{w1}=\frac{c_{b1}}{\kappa^2}+\frac{1+c_{b2}}{\sigma},\ c_{w2}=0.3,\ c_{w3}=2,\ c_{v1}=7.1 \tag{11}$$

The Spalart-Allmaras model is a widely used turbulence model and has proved to provide good results for separated flows [32] and drag predictions [33] whilst being robust with respect to the grid spacing.

2.3. Solution method

The RANS equations are spatially discretised using a cell-centred finite volume scheme. A central difference scheme is used to approximate the convective flux of the mean flow equations. Scalar numerical dissipation is included to stabilize the scheme by suppressing odd-even decoupling of the cell-centered scheme and oscillations near shocks. The dissipation used is the one proposed by Jameson [34]. A modification to the scaling factor has been employed to prevent excessive dissipation as suggested by Swanson and Turkel [35].

A finite volume scheme is used to discretise the Spalart-Allmaras equation and the turbulence model convective flux is approximated using an upwind scheme, which prevents the flowfield from becoming negative [36]. The source terms in the Spalart-Allmaras equation are evaluated using the mid-point rule.

Combining the RANS equations and the turbulence equations gives the system

$$\frac{d\mathbf{Y}}{dt} + \mathbf{R} = 0 \tag{12}$$

where $\mathbf{Y}$ is the vector of unknowns, which now includes both flow variables for the RANS equations and turbulence model variable. The residual, $\mathbf{R}$, has also been modified to include the turbulence terms.

Applying equation (12) to steady flows reduces it to a system of algebraic equations given by

$$\mathbf{R} = 0. \tag{13}$$

This non-linear equation, (13), can then be solved directly using an iterative solver to give a steady-state solution. However, a steady-state solution can also be computed by applying a time-integration technique to the unsteady formulation, (12). These two methods of computing a steady solution produce two systems which are different in a dynamical sense. Following the convention used in dynamical systems, the first is termed a time-discrete system given by a temporal discretisation of the non-linear equation (13), which is known as a time-independent system. As this work focuses on tracking bifurcations of both types of systems, the distinction is made clear although it has been shown that the results of the two systems should agree [37]. For the computation of flow solutions Newton's method is used for the time-independent system, whilst an explicit Runge-Kutta method is used for the time-discrete system.

3. Continuation methods

In order to compute a starting solution for the bifurcation tracking algorithms, a continuation method was employed. This computed the system equilibria as a parameter, λ, was varied. At each equilibrium the corresponding stability information was also computed. This process was continued until a bifurcation had been detected. The formulation of the continuation method is different for the time-independent and time-discrete systems and both are described separately below.

However, for both methods an arclength continuation method is applied using a predictor-corrector algorithm. The solution branch is found by calculating an initial solution, $(\mathbf{Y}^0, \lambda^0)$, then changing the value of the continuation parameter, λ, by a small amount and finding the solution at this new parameter value. Arclength continuation is used in order to allow the continuation to continue through a fold bifurcation. The solution curve is parameterized in terms of its arclength, which ensures that a singular point is not encountered. The arclength is given as the distance between two consecutive solution points along the curve.

3.1. Time-independent continuation method

For the time-independent system, the predictor-corrector algorithm works by computing an approximate solution in the predictor stage and then iterating on this value using Newton's method in the corrector stage to find the system equilibrium point converged to acceptable accuracy.

In this work a secant predictor is used. This requires the first two solutions on the branch, $(\mathbf{Y}^0, \lambda^0)$ and $(\mathbf{Y}^1, \lambda^1)$, to be generated. An approximate solution at the next parameter value is then generated using

$$\bar{\mathbf{X}}^{n+1} = \mathbf{X}^n + h\mathbf{t}^n \tag{14}$$

where $\mathbf{X} = (\mathbf{Y}^n, \lambda^n)$ and

$$\mathbf{t} = \frac{\mathbf{X}^n - \mathbf{X}^{n-1}}{\|\mathbf{X}^n - \mathbf{X}^{n-1}\|}. \tag{15}$$

This value is then used as the starting point for the Newton solver, which iterates on this value to find the actual solution at the current parameter value. In arclength continuation the parameter is allowed to vary which means there is one extra unknown, which requires an extra equation. This extra equation is the arclength equation used to parametrise the curve.

$$g(\mathbf{Y}, \lambda, \varsigma) = \left[\frac{\partial \mathbf{Y}}{\partial \varsigma}\right] \Delta\mathbf{Y} + \left[\frac{\partial \lambda}{\partial \varsigma}\right] \Delta\lambda - \Delta\varsigma = 0, \tag{16}$$

This process works by restricting the Newton updates to a hyperplane normal to the direction taken for the predictor. The complete augmented system which must be is solved is

$$\begin{bmatrix} \mathbf{J} & \frac{\partial \mathbf{R}}{\partial \lambda} \\ \frac{\partial g}{\partial \mathbf{Y}} & \frac{\partial g}{\partial \lambda} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{Y} \\ \Delta\lambda \end{bmatrix}^n = \begin{bmatrix} -\mathbf{R} \\ -g \end{bmatrix}^n, \tag{17}$$

where $\mathbf{J} = \frac{\partial \mathbf{R}}{\partial \mathbf{Y}}$ is the Jacobian of the flow variables. The system, Equation (17), was solved using a bordered solver known as block elimination mixed (BEM) [38] bordering algorithm. This requires only solves involving the sparse Jacobian, which were accomplished using Umfpack [11, 12, 13, 14], a direct solver designed for sparse systems.

The Jacobian was calculated numerically using first order differencing of the residual as in the work of Wales [6].

3.2. Time-discrete continuation method using the Recursive Projection Method

A continuation method computes the equilibrium solution of a dynamical system. In order to implement the arclength continuation method described in Section 3.1, an iterative solver, such as Newton's method, must be available. However, it is common practice in CFD to perform fixed point iterations such that

$$\frac{d\mathbf{Y}}{d\tau} = \mathbf{R}(\mathbf{Y}, \lambda) \tag{18}$$

is solved using some form of time-stepping in τ, such as the Runge-Kutta method [34]. Solving this system for a range of parameter values would only be possible if the solution was guaranteed to be stable at every point. This is typically not the case, especially in this work as we are concerned with bifurcation points which indicate a change in stability. For unstable solutions, time-integration methods such as the Runge-Kutta method do not converge. In these cases they may oscillate about a point or even diverge [24]. If the result is an oscillatory solution then a common but costly way of computing a steady solution is to average out the oscillatory values over a certain number of time-steps.

A method that overcomes the convergence issues displayed by time-integration techniques in the unstable parameter range is the Recursive Projection Method (RPM) [24]. The RPM works as it is a small number of eigenvalues leaving the unit disk that cause the convergence problems [24]. Shroff and Keller therefore suggest that this part of the system can be extracted and solved using Newton's method whilst the large remainder of the system continues to be solved using the time-integration method. Using this technique enables fixed point iterations to converge onto unstable solutions and therefore be used within a continuation method. This occurs with minimal additional computational cost.

The RPM separates the unstable part of the system from the stable part by identifying a subspace that corresponds to the unstable eigenvalues and spans their eigenvectors. This unstable subspace is denoted by P and its orthogonal complement, the stable subspace, is Q. The solution vector $\mathbf{Y}$ is then projected onto each of the subspaces giving $\mathbf{p} = P\mathbf{Y}$ and $\mathbf{q} = Q\mathbf{Y}$ and the total solution vector is a sum of the two parts,

$$\mathbf{Y} = \mathbf{p} + \mathbf{q}. \tag{19}$$

The two subspaces are given by, $P = \mathbf{Z}\mathbf{Z}^T$ and $Q = \mathbf{I} - \mathbf{Z}\mathbf{Z}^T$, where $\mathbf{Z}$ is an orthonormal basis for P. The unstable part, $P\mathbf{Y}$, is solved using Newton's method and the stable part, $Q\mathbf{Y}$, is solved using the original time-stepping method.

3.2.1. Increasing the basis.

Computing the orthonormal basis, $\mathbf{Z}$, is perhaps the most important aspect of the RPM as it dictates the effectiveness of the method. In order to know when to increase the basis, the past iterates of the stable solution part are monitored as it is known that when the convergence rate of this part of the solution degrades, at least one eigenvalue must be approaching the unit disk [24]. The past iterates of the stable part of the solution form a second order approximation to a power iteration for the stable Jacobian. In other words, the past iterates will form the dominant eigenspace for the stable part of the solution and can therefore be used to not only dictate when to increase the basis but also be used to create the basis itself. This is achieved by collecting the last $k_v + 1$ iterates of the stable solution part (which will initially be the entire solution), every ν_{max} iterations, into a matrix, $\{\Delta\mathbf{q}\}$.

$$\begin{gathered}\{\Delta\mathbf{q}^i\} = \{(\mathbf{q}^{\nu_{max}} - \mathbf{q}^{\nu_{max}-1}), (\mathbf{q}^{\nu_{max}-1} - \mathbf{q}^{\nu_{max}-2}), \dots, (\mathbf{q}^{\nu_{max}-k_v+1} - \mathbf{q}^{\nu_{max}-k_v})\} \\ for \quad i = \nu_{max} - k_v + 1, \dots, \nu_{max}\end{gathered} \tag{20}$$

A modified Gram-Schmidt procedure is then performed on this matrix to construct a QR-factorisation. If

$$\mathbf{D} = \{\Delta\mathbf{q}^{\nu_{max}-k_v+1}, \dots, \Delta\mathbf{q}^{\nu_{max}}\} \tag{21}$$

then performing a QR-factorisation gives

$$\mathbf{D} = \hat{\mathbf{D}}\mathbf{T} \tag{22}$$

where $\mathbf{T}$ is a $k_v \times k_v$ upper triangular matrix and $\hat{\mathbf{D}}$ is orthogonal. A test is then performed on the matrix $\mathbf{T}$. For the largest j satisfying

$$\left|\frac{T_{j,j}}{T_{j+1,j+1}}\right| \geq \mathtt{KA}, \quad j = 1, \dots, k_v - 1 \tag{23}$$

where `KA` is the Krylov Acceptance ratio and defines how dominant the eigenvalues must be for the eigenvectors to be added to the basis. The values for ν_{max}, k_v and `KA` are user-defined and problem dependent. The first j vectors of $\hat{\mathbf{D}}$ are then added to the basis $\mathbf{Z}$.

3.2.2. Maintaining the basis.

When this technique is used within a continuation method, occasionally the basis must be re-orthogonalised as the dominant eigenspace of the Jacobian of the time-integration method, $\mathbf{G}$, changes, causing the basis to become inaccurate. Re-orthogonalisation is achieved by completing one step of an orthogonal power iteration [39] on the columns of $\mathbf{Z}$, which is achieved by performing a Gram-Schmidt orthogonalisation on $\mathbf{GZ}$. This step requires minimal extra work; in particular, no additional function evaluations are needed because an approximation to the quantity $\mathbf{GZ}$ is required for executing the stabilised iteration [24].

3.2.3. Decreasing the basis.

Sometimes, during the continuation process, it is necessary to reduce the size of the basis as some eigenvalues may move back towards the origin of the disk. This is a necessary step as it is not sufficient for the basis to contain an invariant subspace but the basis must span an invariant subspace [24]. Therefore, at the end of each continuation step the eigenvalues and eigenvectors of $\mathbf{H} = \mathbf{Z}^T\mathbf{G}^*\mathbf{Z}$ are computed, where $\mathbf{G}^* = \frac{\partial \mathbf{F}(\mathbf{Y}^*)}{\partial \mathbf{Y}}$. The matrix, $\mathbf{H}$, corresponds to the projection of $\mathbf{G}^*$ on the unstable subspace and the eigenvalues of $\mathbf{H}$ are therefore a subset of those of $\mathbf{G}^*$. This is relatively inexpensive operation to perform as $\mathbf{H}$ is only an $m \times m$ matrix, where m is the size of the basis $\mathbf{Z}$. A new basis is then formed corresponding to the eigenvectors, $\mathbf{V}$, that relate to the eigenvalues that are outside the disk K_δ of radius $(1-\delta)$. The basis is then updated by replacing $\mathbf{Z}$ with the Gram-Schmidt orthogonalisation of $\mathbf{ZV}$.

3.2.4. Practical implementation.

In order to reduce the computational cost of the method, coordinate variables, $\mathbf{z}$, are used for the representation of $\mathbf{p}$ in the basis $\mathbf{Z}$ [24]. This gives

$$\mathbf{z} \equiv \mathbf{Z}^T\mathbf{p} \tag{24}$$

The Newton iteration for $\mathbf{p}$ can then be written as

$$\mathbf{z}^{\nu+1} = \mathbf{z}^{\nu} + \left(\mathbf{I} - \mathbf{Z}^T\mathbf{G}\mathbf{Z}\right)^{-1}\left(\mathbf{Z}^T\mathbf{F}(\mathbf{Y}^{\nu},\lambda) - \mathbf{z}^{\nu}\right). \tag{25}$$

This allows the Newton step to be performed on a low-dimensional subspace of size $m \times m$. To avoid large computational costs and ensure that the RPM can be used with a "blackbox" timestepper, the matrix $\mathbf{GZ}$ is computed using finite differences. The jth row is given by

$$\{\mathbf{GZ}_i\}_j \approx \frac{\mathbf{F}_j\left(\mathbf{Y}_j + \epsilon_j \mathbf{Z}_{i,j}, \lambda\right) - \mathbf{F}_j\left(\mathbf{Y}_j, \lambda\right)}{\epsilon_j} \quad i = 1, 2, \ldots, m; \;\; j = 1, 2, \ldots, N \tag{26}$$

Here ϵ is a small, user-defined number and is case specific. In this work a value of 1×10^{-5} is used.

The derivative of $\mathbf{F}$ with respect to the continuation parameter, λ denoted as $\mathbf{F}_\lambda$ must also be computed in order to perform arclength continuation. This is also computed using finite differencing and is given by

$$\mathbf{F}_\lambda \approx \frac{\mathbf{F}(\mathbf{Y}, \lambda + \epsilon_\lambda) - \mathbf{F}(\mathbf{Y}, \lambda)}{\epsilon_\lambda} \tag{27}$$

where ϵ_λ is a small perturbation and is taken to be 1×10^{-7} in this work. As Newton's method is only applied to the unstable part of the system and this is small, bordering algorithms such as that used for the time-independent case are not necessary. This minimises the computational complexity of the method. The process for one arclength continuation step using the RPM is given in Algorithm 1.

The knowledge obtained from the application of RPM to enable continuation on a large RANS system around an aerofoil in conjunction with the bifurcation tracking algorithms described in Section 4 was used to help develop the bifurcation tracking methods presented in Sections 5 and 6.

4. Bifurcation tracking

In this work the continuation methods have been used primarily as a way of calculating an initial guess for the bifurcation tracking methods. The first unstable solution computed by the continuation method is used as the initial guess for the first bifurcation point. Once this first bifurcation point has been computed then a second parameter, γ, can be freed and the bifurcation can be tracked as both γ and the original continuation parameter, λ, vary. This enables the problem's stability boundary to be traced in two-parameter space.

Algorithm 1 Recursive Projection Method - Arclength Continuation

```
Define $\{\mathbf{Y}^{-1}, \mathbf{Y}^0, \lambda^{-1}\lambda^0, tol, \nu_{max}\}$
$Z = [\,]$
Secant Predictor: $(\bar{\mathbf{Y}}, \bar{\lambda})$
Runge-Kutta: $\mathbf{F}(\bar{\mathbf{Y}}, \lambda)$
$\nu = 0$
while $\|\mathbf{Y} - \mathbf{F}\|_2 > tol$ do
    $\mathbf{z} \leftarrow \mathbf{Z}^T\mathbf{Y}; \zeta \leftarrow \mathbf{Z}^T\mathbf{F};\ \mathbf{q} \leftarrow \mathbf{Y} - \mathbf{Z}\mathbf{z};$
    $\frac{\partial \mathbf{Y}}{\partial \varsigma} \leftarrow \mathbf{Z}^T \frac{(\bar{\mathbf{Y}} - \mathbf{Y}^0)}{\Delta\varsigma};\ \frac{\partial \lambda}{\partial \varsigma} \leftarrow \frac{(\bar{\lambda} - \lambda^0)}{\Delta\varsigma};$
    $\mathbf{z}_\lambda \leftarrow \mathbf{Z}^T\mathbf{F}_\lambda;\ \mathbf{q} \leftarrow \mathbf{F} - \mathbf{Z}\zeta;$
    $\Delta\mathbf{Y} \leftarrow (\mathbf{z} - \mathbf{Z}^T\bar{Y});\ \Delta\lambda \leftarrow \lambda - \bar{\lambda};$
    $g(\mathbf{Y}, \lambda, \varsigma) \leftarrow \left[\frac{\partial \mathbf{Y}}{\partial \varsigma}\right]\Delta\mathbf{Y} + \left[\frac{\partial \lambda}{\partial \varsigma}\right]\Delta\lambda - \Delta\varsigma$
    $\begin{pmatrix} \delta\mathbf{z} \\ \delta\lambda \end{pmatrix} \leftarrow \begin{bmatrix} \mathbf{I}_m - \mathbf{H} & -\mathbf{z}_\lambda \\ \frac{\partial \mathbf{Y}}{\partial \varsigma} & \frac{\partial \lambda}{\partial \varsigma} \end{bmatrix}^{-1} \begin{bmatrix} \zeta - \mathbf{z} \\ -g \end{bmatrix}$
    $\mathbf{q} \leftarrow \mathbf{Y} - \mathbf{Z}\zeta + (\mathbf{F}_\lambda - \mathbf{Z}\mathbf{z}_\lambda)\delta\lambda;\ \mathbf{Y} \leftarrow \mathbf{Z}\mathbf{z} + \mathbf{q};\ \lambda \leftarrow \lambda + \delta\lambda;$
    Runge-Kutta: $\mathbf{F}(\mathbf{Y}, \lambda)$
    $\nu = \nu + 1$
    if $\nu > \nu_{max}$ then
        Update $\mathbf{Z}$
        compute $\mathbf{H} \leftarrow \mathbf{Z}^T[\mathbf{G}\mathbf{Z}]$
        $\nu = 0$
    end if
end while
```

Two different methods were used to firstly locate and then to track a bifurcation. In this work the type of bifurcation encountered was always of the Hopf type, therefore the bifurcation tracking methods employed were set up to locate and track a Hopf bifurcation, although for both methods an alternative formulation for tracking other simple bifurcations is possible.

4.1. Augmented system

Firstly, the augmented system was considered. This requires the solution to a system that defines the bifurcation of interest. For a Hopf bifurcation the augmented system requires four additional equations to be solved. These correspond to the complex eigenvalue equation, which when expressed in terms of real-valued variables can be split into two equations, and the two scalar equations which fix the phase and amplitude of the eigenvector. This set of equations can be written as [7]

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{J}\mathbf{v}_1 + \omega\mathbf{v}_2 \\ \mathbf{J}\mathbf{v}_2 - \omega\mathbf{v}_1 \\ \phi \cdot \mathbf{v}_1 - 1 \\ \phi \cdot \mathbf{v}_2 \end{bmatrix} = 0. \tag{28}$$

This system has a size of $3N + 2$ and solves for the five unknowns: the flow solution vector, $\mathbf{Y}$, the real and imaginary parts of the complex eigenvector, $\mathbf{v}_1$ and $\mathbf{v}_2$, the eigenvalue, ω, and the first parameter, λ. The first equation in system (28) specifies that the solution must be on the flow solution branch, the second and third ensure that the point has a purely imaginary eigenvalue and the final two fix the phase and amplitude of the eigenvectors. The last two equations can be formulated differently and have been done so in other papers [8, 5, 40]. Newton's method is applied to this system to give

$$\begin{bmatrix} \mathbf{J} & \mathbf{0} & \mathbf{0} & 0 & \frac{\partial \mathbf{R}}{\partial \lambda} \\ \frac{\partial \mathbf{J}\mathbf{v}_1}{\partial \mathbf{Y}} & \mathbf{J} & \omega\mathbf{I} & \mathbf{v}_2 & \frac{\partial \mathbf{J}\mathbf{v}_1}{\partial \lambda} \\ \frac{\partial \mathbf{J}\mathbf{v}_2}{\partial \mathbf{Y}} & -\omega\mathbf{I} & \mathbf{J} & -\mathbf{v}_1 & \frac{\partial \mathbf{J}\mathbf{v}_2}{\partial \lambda} \\ \mathbf{0} & \phi^T & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \phi^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{Y} \\ \Delta\mathbf{v}_1 \\ \Delta\mathbf{v}_2 \\ \Delta\omega \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \mathbf{R} \\ \mathbf{J}\mathbf{v}_1 + \omega\mathbf{v}_2 \\ \mathbf{J}\mathbf{v}_2 - \omega\mathbf{v}_1 \\ \phi \cdot \mathbf{v}_1 - 1 \\ \phi \cdot \mathbf{v}_2 \end{bmatrix} \tag{29}$$

The underlying vector of flow equations is of size N and the continuation method used to produce the initial guess also involved solves of an $N \times N$ system. Therefore, increasing the system to a size of $3N + 2$ can be prohibitive. However, following the method used by LOCA [7] the computational expense of this method can be reduced by using a bordering algorithm. Employing this algorithm requires a maximum solve of size $2N \times 2N$. One iteration of the algorithm is given by:

$$\mathbf{Ja} = -\mathbf{R}, \tag{30}$$

$$\mathbf{Jb} = -\frac{\partial \mathbf{R}}{\partial \lambda} \tag{31}$$

$$\begin{bmatrix} \mathbf{J} & \omega \mathbf{I} \\ -\omega \mathbf{I} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_2 \\ -\mathbf{v}_1 \end{bmatrix} \tag{32}$$

$$\begin{bmatrix} \mathbf{J} & \omega \mathbf{I} \\ -\omega \mathbf{I} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{f} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathbf{Jv}_1}{\partial \mathbf{Y}}\mathbf{a} \\ -\frac{\partial \mathbf{Jv}_2}{\partial \mathbf{Y}}\mathbf{a} \end{bmatrix} \tag{33}$$

$$\begin{bmatrix} \mathbf{J} & \omega \mathbf{I} \\ -\omega \mathbf{I} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathbf{Jv}_1}{\partial \mathbf{Y}}\mathbf{b} - \frac{\partial \mathbf{Jv}_1}{\partial \lambda} \\ -\frac{\partial \mathbf{Jv}_2}{\partial \mathbf{Y}}\mathbf{b} - \frac{\partial \mathbf{Jv}_2}{\partial \lambda} \end{bmatrix} \tag{34}$$

$$\Delta\lambda = \frac{(\phi \cdot \mathbf{c})(\phi \cdot \mathbf{f}) - (\phi \cdot \mathbf{e})(\phi \cdot \mathbf{d}) + (\phi \cdot \mathbf{d})}{(\phi \cdot \mathbf{d})(\phi \cdot \mathbf{g}) - (\phi \cdot \mathbf{c})(\phi \cdot \mathbf{h})} \tag{35}$$

$$\Delta\omega = \frac{(\phi \cdot \mathbf{h})\Delta\lambda + (\phi \cdot \mathbf{f})}{\phi \cdot \mathbf{d}} \tag{36}$$

$$\Delta \mathbf{Y} = \mathbf{a} + \Delta\lambda \mathbf{b} \tag{37}$$

$$\Delta \mathbf{v}_1 = \mathbf{e} + \Delta\lambda \mathbf{g} - \Delta\omega \mathbf{c} - \mathbf{v}_1 \tag{38}$$

$$\Delta \mathbf{v}_2 = \mathbf{f} + \Delta\lambda \mathbf{h} - \Delta\omega \mathbf{d} - \mathbf{v}_2 \tag{39}$$

The most expensive part of this algorithm is the three solves involving the $2N \times 2N$ matrix, $\begin{bmatrix} \mathbf{J} & \omega \mathbf{I} \\ -\omega \mathbf{I} & \mathbf{J} \end{bmatrix}$ however the algorithm also requires two solves involving $\mathbf{J}$, an $N \times N$ matrix. These two matrices are sparse so it is possible to employ the Umfpack solver as used for the continuation method. This also has the advantage that only the first solve with each matrix is a significant cost due to its use of symbolic factorisation methods. The choice of the vector ϕ is relatively free and in this work ϕ was chosen to be a vector of ones for simplicity. The derivatives that appear on the right-hand sides of equation (33) and equation (34) are calculated using second order differences. Equations (40)-(42) show the derivatives involving $\mathbf{v}_1$. The derivatives involving $\mathbf{v}_2$ are calculated in a similar manner.

$$\frac{\partial \mathbf{J}\mathbf{v}_1}{\partial \mathbf{Y}}\mathbf{a} = \frac{\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1+\epsilon_1\mathbf{a},\lambda)-\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1-\epsilon_1\mathbf{a},\lambda)-\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1+\epsilon_1\mathbf{a},\lambda)+\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1-\epsilon_1\mathbf{a},\lambda)}{4\sigma_1\epsilon_1} \tag{40}$$

$$\frac{\partial \mathbf{J}\mathbf{v}_1}{\partial \mathbf{Y}}\mathbf{b} = \frac{\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1+\epsilon_2\mathbf{b},\lambda)-\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1-\epsilon_2\mathbf{b},\lambda)-\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1+\epsilon_2\mathbf{b},\lambda)+\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1-\epsilon_2\mathbf{b},\lambda)}{4\sigma_1\epsilon_2} \tag{41}$$

$$\frac{\partial \mathbf{J}}{\partial \lambda}\mathbf{v}_1 = \frac{\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1,\lambda+\epsilon_3)-\mathbf{R}(\mathbf{Y}+\sigma_1\mathbf{v}_1,\lambda-\epsilon_3)-\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1,\lambda+\epsilon_3)+\mathbf{R}(\mathbf{Y}-\sigma_1\mathbf{v}_1,\lambda-\epsilon_3)}{4\sigma_1\epsilon_3} \tag{42}$$

Here ϵ_1, ϵ_2, ϵ_3, and σ_1 are small perturbations. The size of these perturbations effects the efficiency and robustness of the algorithm. They are calculated using the following equations as they were found to perform well in the work of Salinger *et al.* [7]

$$\epsilon_1 = \delta\left(\frac{\|\mathbf{Y}\|}{\|\mathbf{a}\|}+\delta\right), \tag{43}$$

$$\epsilon_2 = \delta\left(\frac{\|\mathbf{Y}\|}{\|\mathbf{b}\|}+\delta\right), \tag{44}$$

$$\epsilon_3 = \delta\left(|\lambda|+\delta\right), \tag{45}$$

$$\sigma_1 = \delta\left(\frac{\|\mathbf{Y}\|}{\|\mathbf{v}_1\|}+\delta\right). \tag{46}$$

The value of δ can be chosen by the user to be any small number. A value of 1×10^{-6} proved to work well in this work.

4.2. Test function

An alternative way of locating a bifurcation is by using a test function. Test functions are continuous around the bifurcation point and have the condition that at a bifurcation point the solution is zero. Many different test functions exist which all have the same result but varying methods [38]. The most common test function for Hopf bifurcations works by noting that at a Hopf bifurcation the matrix $\mathbf{J}^2 + \omega^2\mathbf{I}$ has rank defect 2. This method was considered initially, however although the matrix $\mathbf{J}^2 + \omega^2\mathbf{I}$ has the same dimensions as $\mathbf{J}$ it is a lot more dense, which slows down the computation considerably. Therefore an alternative method was implemented that works on the basis that at a Hopf bifurcation point the matrix $\mathbf{J} - i\omega\mathbf{I}$ is singular. This matrix is then appended with elements to make it non-singular. This gives a bordered matrix of

$$\mathbf{M} = \begin{bmatrix} \mathbf{J} - i\omega\mathbf{I} & \mathbf{W} \\ \mathbf{V}^T & 0 \end{bmatrix} \tag{47}$$

Here $\mathbf{V}$ is the right complex eigenvector, where $\mathbf{V} = \mathbf{v}_1 + i\mathbf{v}_2$ and $\mathbf{W}$ is the left complex eigenvector, $\mathbf{W} = \mathbf{w}_1 + i\mathbf{w}_2$. This system can be split into real and imaginary components. This increases the size of the system from $N+1$ to $2N+2$ but allows a real-valued solver to be applied to the system. The bordered matrix written in terms of real-valued variables is

$$\mathbf{M} = \begin{bmatrix} \mathbf{J} & \omega\mathbf{I} & \mathbf{w}_1 & -\mathbf{w}_2 \\ -\omega\mathbf{I} & \mathbf{J} & \mathbf{w}_2 & \mathbf{w}_1 \\ \mathbf{v}_1^T & -\mathbf{v}_2^T & 0 & 0 \\ \mathbf{v}_2^T & \mathbf{v}_1^T & 0 & 0 \end{bmatrix} \tag{48}$$

This matrix is then used to define the test function, which in this case has two components (h_1 and h_2), by

$$\begin{bmatrix} \mathbf{J} & \omega\mathbf{I} & \mathbf{w}_1 & -\mathbf{w}_2 \\ -\omega\mathbf{I} & \mathbf{J} & \mathbf{w}_2 & \mathbf{w}_1 \\ \mathbf{v}_1^T & -\mathbf{v}_2^T & 0 & 0 \\ \mathbf{v}_2^T & \mathbf{v}_1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \\ 0 \end{bmatrix}. \tag{49}$$

In order for Equation (48) to have rank defect 1 it must be true that $h_1 = h_2 = 0$. This means the defining system $\mathbf{s} = (\mathbf{Y}, \lambda, \omega)$ is then given by

$$\begin{bmatrix} \mathbf{R} = 0 \\ h_1 = 0 \\ h_2 = 0 \end{bmatrix}. \tag{50}$$

A single Newton iteration then solves the system

$$\begin{bmatrix} \mathbf{J} & \frac{\partial \mathbf{R}}{\partial \lambda} & \mathbf{0} \\ \frac{\partial h_1}{\partial \mathbf{Y}} & \frac{\partial h_1}{\partial \lambda} & \frac{\partial h_1}{\partial \omega} \\ \frac{\partial h_2}{\partial \mathbf{Y}} & \frac{\partial h_2}{\partial \lambda} & \frac{\partial h_2}{\partial \omega} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{Y} \\ \Delta \lambda \\ \Delta \omega \end{bmatrix} = - \begin{bmatrix} \mathbf{R} \\ h_1 \\ h_2 \end{bmatrix} \tag{51}$$

The partial derivatives of h_1 and h_2 with respect to $\mathbf{s}$ can be computed efficiently by considering the solution of transpose of the system

$$\begin{bmatrix} \mathbf{q}_1^T & \mathbf{q}_2^T & g_1 & g_2 \\ \mathbf{q}_3^T & \mathbf{q}_4^T & g_3 & g_4 \end{bmatrix} \mathbf{M} = \begin{bmatrix} \mathbf{0}^T & \mathbf{0}^T & 1 & 0 \\ \mathbf{0}^T & \mathbf{0}^T & 0 & 1 \end{bmatrix} \tag{52}$$

By differentiating Equation (49) and substituting Equation (52) into the result, the derivatives for $\mathbf{Y}$, λ, and ω are shown to be:

$$h_{1\mathbf{Y}} = -\mathbf{q}_1^T \mathbf{J}_\mathbf{Y} \mathbf{p}_1 - \mathbf{q}_2^T \mathbf{J}_\mathbf{Y} \mathbf{p}_2 \tag{53}$$

$$h_{2\mathbf{Y}} = -\mathbf{q}_3^T \mathbf{J}_\mathbf{Y} \mathbf{p}_1 - \mathbf{q}_4^T \mathbf{J}_\mathbf{Y} \mathbf{p}_2 \tag{54}$$

$$h_{1\lambda} = -\mathbf{q}_1^T \mathbf{J}_\lambda \mathbf{p}_1 - \mathbf{q}_2^T \mathbf{J}_\lambda \mathbf{p}_2 \tag{55}$$

$$h_{2\lambda} = -\mathbf{q}_3^T \mathbf{J}_\lambda \mathbf{p}_1 - \mathbf{q}_4^T \mathbf{J}_\lambda \mathbf{p}_2 \tag{56}$$

$$h_{1\omega} = -\mathbf{q}_1^T \mathbf{p}_2 + \mathbf{q}_2^T \mathbf{p}_1 \tag{57}$$

$$h_{2\omega} = -\mathbf{q}_3^T \mathbf{p}_2 + \mathbf{q}_4^T \mathbf{p}_1 \tag{58}$$

where $\mathbf{J}_\mathbf{Y}$ is the Hessian matrix and $\mathbf{J}_\lambda$ is the derivative of the Jacobian matrix, $\mathbf{J}$, with respect to the parameter, λ. For a two-dimensional system the Hessian matrix is a large three-dimensional system and its computation is expensive. However, in the current implementation the Hessian is only required when post-multiplied by two vectors, as in equations (61) and (62) which allows it to be computed using a finite difference formulation such as that given in (40). The system was solved by a block elimination algorithm that requires only two solves of the matrix $\mathbf{J}$ as shown below.

$$\mathbf{J}\mathbf{a} = -\mathbf{R} \tag{59}$$

$$\mathbf{J}\mathbf{b} = \frac{\partial \mathbf{R}}{\partial \lambda} \tag{60}$$

$$\Delta\lambda = \frac{\left(\frac{h_{1\omega}}{h_{2\omega}}h_2 - h_1 - h_{1\mathbf{Y}}\mathbf{a} + \frac{h_{1\omega}}{h_{2\omega}}h_{2\mathbf{Y}}\mathbf{a}\right)}{\left(h_{1\mathbf{Y}}\mathbf{b} + h_{1\lambda} - \frac{h_{1\omega}}{h_{2\omega}}\left(h_{2\mathbf{Y}}\mathbf{b} + h_{2\lambda}\right)\right)} \tag{61}$$

$$\Delta\omega = \frac{-h_2 - h_{2\mathbf{Y}}\mathbf{a} - \left(h_{2\mathbf{Y}}\mathbf{b} + h_{2\lambda}\right)\Delta\lambda}{h_{2\omega}} \tag{62}$$

$$\Delta\mathbf{Y} = \mathbf{a} + \mathbf{b}\Delta\lambda \tag{63}$$

5. Bifurcation tracking using projected methods

A drawback of the bifurcation tracking methods, described in Section 4, is that they are computationally very expensive. Taking the Recursive Projection Method developed in Section 3.2 and applying it to one of the bifurcation tracking methods described in Section 4 results in a new and innovative bifurcation tracker that is very inexpensive. Assuming initial conditions that are close to a bifurcation point, the unstable subspace found by the RPM should be sufficiently accurate that the bifurcation point is contained within that subspace. Then by solving the equations that define a bifurcation on this subspace means solving a relatively small system. The greatest restriction of this method is the accuracy of the basis extracted via the RPM.

As the system being solved is the fixed point iteration, (12) and not the linear system, (13), this means the equations that define the bifurcation are different. As for the original bifurcation trackers, only the Hopf bifurcation is discussed here but the technique is equally applicable to all types of bifurcations.

5.1. Augmented system

The equations that define a Hopf bifurcation for a fixed point iteration of the form

$$\mathbf{Y}^{\nu+1} = \mathbf{F}\left(\mathbf{Y}^{\nu}, \lambda\right) \tag{64}$$

are given by [41]

$$\mathbf{F}\left(\mathbf{Y}, \lambda, \mu\right) - \mathbf{Y} = 0, \tag{65}$$

$$\mathbf{G}\mathbf{v} - e^{i\beta}\mathbf{v} = 0, \tag{66}$$

$$\mathbf{v}\phi - 1 = 0, \tag{67}$$

where $\mathbf{v} = \mathbf{v}_1 + i\mathbf{v}_2$ is the eigenvector of the time-discrete system that corresponds to the eigenvalue crossing the unit disk, $\mathbf{G}$ is the Jacobian matrix of the time-discrete system, β is the imaginary part of the eigenvalue that is crossing the unit disk and ϕ is a constant vector. If these equations are projected onto the unstable subspace found through RPM and within which the bifurcation should lie, the equations become

$$\zeta - \mathbf{z} = 0, \tag{68}$$

$$\mathbf{H}\tilde{\mathbf{v}} - e^{i\beta}\tilde{\mathbf{v}} = 0, \tag{69}$$

$$\tilde{\mathbf{v}}\Phi - 1 = 0. \tag{70}$$

where $\tilde{\mathbf{v}}$ is the projection of the eigenvector $\mathbf{v}$ onto the unstable subspace and is given by $\tilde{\mathbf{v}} = \mathbf{Z}^T\mathbf{v}$ and ζ, $\mathbf{z}$ and $\mathbf{H}$ are as given in Section 3.2. Writing this system in real form to allow existing solver methods to be used the equations to be solved become

$$\zeta - \mathbf{z} = 0, \tag{71}$$

$$\mathbf{H}\tilde{\mathbf{v}}_1 - \tilde{\mathbf{v}}_1\cos(\beta) + \tilde{\mathbf{v}}_2\sin(\beta), \tag{72}$$

$$\mathbf{H}\tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1\sin(\beta) - \tilde{\mathbf{v}}_2\cos(\beta), \tag{73}$$

$$\Phi \cdot \tilde{\mathbf{v}}_1 - 1 = 0, \tag{74}$$

$$\Phi \cdot \tilde{\mathbf{v}}_2 = 0. \tag{75}$$

As can be seen these equations have a similar form to those of the time-independent system. However, the size of this system is $(3m+2) \times (3m+2)$, where once again m is the size of the basis extracted by RPM and $m << N$.

In this work these equations are solved using Newton's method, which gives

$$\begin{bmatrix} \mathbf{H}-\mathbf{I} & \mathbf{0} & \mathbf{0} & 0 & \frac{\partial\zeta}{\partial\lambda}-\frac{\partial\mathbf{z}}{\partial\lambda} \\ \frac{\partial\mathbf{H}\tilde{\mathbf{v}}_1}{\partial\mathbf{z}} & \mathbf{H}-\cos(\beta) & \sin(\beta)\mathbf{I} & \tilde{\mathbf{v}}_1\sin(\beta)+\tilde{\mathbf{v}}_2\cos(\beta) & \frac{\partial\mathbf{H}\tilde{\mathbf{v}}_1}{\partial\lambda} \\ \frac{\partial\mathbf{H}\tilde{\mathbf{v}}_2}{\partial\mathbf{z}} & -\sin(\beta)\mathbf{I} & \mathbf{H}-\cos(\beta) & \tilde{\mathbf{v}}_2\sin(\beta)-\tilde{\mathbf{v}}_1\cos(\beta) & \frac{\partial\mathbf{H}\tilde{\mathbf{v}}_2}{\partial\lambda} \\ \mathbf{0} & \Phi^T & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \Phi^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z} \\ \Delta\tilde{\mathbf{v}}_1 \\ \Delta\tilde{\mathbf{v}}_2 \\ \Delta\beta \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \zeta-\mathbf{z} \\ \mathbf{H}\tilde{\mathbf{v}}_1-\tilde{\mathbf{v}}_1\cos(\beta)+\tilde{\mathbf{v}}_2\sin(\beta) \\ \mathbf{H}\tilde{\mathbf{v}}_2-\tilde{\mathbf{v}}_1\sin(\beta)-\tilde{\mathbf{v}}_2\cos(\beta) \\ \phi\cdot\tilde{\mathbf{v}}_1-1 \\ \phi\cdot\tilde{\mathbf{v}}_2 \end{bmatrix} \quad (76)$$

At each iteration the system (76) is solved in exactly the same way as the original augmented system described in Section 4.1. The reason for this being that the bordering algorithm makes the evaluation of the partial derivatives easier and allows the use of easily implementable finite differences to achieve this.

5.2. Test function

For the time-discrete system, the same test function was applied as used in the time-independent system, detailed in Section 4.2, with the transformation to a time-discrete system made. Although it would now be computationally viable to use the square of the Jacobian in a test function, the complex form was preferred to allow more meaningful comparison. The complex test function for a time-independent system works on the basis that the matrix $\mathbf{J}-i\omega\mathbf{I}_N$ is singular, for a time-discrete system this becomes $\mathbf{G}-\mathbf{I}_N\cos(\beta)-i\mathbf{I}_N\sin(\beta)$. Then following the same method as for the time-independent system but projecting the system onto the unstable basis, the bordered matrix, $\mathbf{M}$, can be written as

$$\mathbf{M}=\begin{bmatrix} \mathbf{H}-\mathbf{I}_m\cos(\beta)-i\mathbf{I}_m\sin(\beta) & \tilde{\mathbf{w}} \\ \tilde{\mathbf{v}}^T & 0 \end{bmatrix} \quad (77)$$

Where $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{w}}$ are the right and left complex eigenvectors of $\mathbf{H}$, respectively and can be written as $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}_1 + i\tilde{\mathbf{v}}_2$ and $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_1 + i\tilde{\mathbf{w}}_2$. In terms of real-valued variables the bordered matrix becomes

$$\mathbf{M} = \begin{bmatrix} \mathbf{H} - \mathbf{I}_m \cos(\beta) & \mathbf{I}_m \sin(\beta) & \tilde{\mathbf{w}}_1 & -\tilde{\mathbf{w}}_2 \\ -\mathbf{I}_m \sin(\beta) & \mathbf{H} - \mathbf{I}_m \cos(\beta) & \tilde{\mathbf{w}}_2 & \tilde{\mathbf{w}}_1 \\ \tilde{\mathbf{v}}_1^T & -\tilde{\mathbf{v}}_2^T & 0 & 0 \\ \tilde{\mathbf{v}}_2^T & \tilde{\mathbf{v}}_1^T & 0 & 0 \end{bmatrix} \tag{78}$$

and the bordered system for the test functions, h_1 and h_2 becomes

$$\begin{bmatrix} \mathbf{H} - \mathbf{I}_m \cos(\beta) & \mathbf{I}_m \sin(\beta) & \tilde{\mathbf{w}}_1 & -\tilde{\mathbf{w}}_2 \\ -\mathbf{I}_m \sin(\beta) & \mathbf{H} - \mathbf{I}_m \cos(\beta) & \tilde{\mathbf{w}}_2 & \tilde{\mathbf{w}}_1 \\ \tilde{\mathbf{v}}_1^T & -\tilde{\mathbf{v}}_2^T & 0 & 0 \\ \tilde{\mathbf{v}}_2^T & \tilde{\mathbf{v}}_1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_m \\ \mathbf{0}_m \\ 1 \\ 0 \end{bmatrix}. \tag{79}$$

It is still the case that in order for equation (78) to have rank defect 2 it must be true that $h_1 = h_2 = 0$. This means the defining system after projection is then given by

$$\begin{bmatrix} \zeta - \mathbf{z} = 0 \\ h_1 = 0 \\ h_2 = 0 \end{bmatrix}. \tag{80}$$

A single Newton iteration then solves the system

$$\begin{bmatrix} \mathbf{H} - \mathbf{I}_m & \frac{\partial \zeta}{\partial \lambda} - \frac{\partial \mathbf{z}}{\partial \lambda} & \mathbf{0} \\ \frac{\partial h_1}{\partial \mathbf{z}} & \frac{\partial h_1}{\partial \lambda} & \frac{\partial h_1}{\partial \beta} \\ \frac{\partial h_2}{\partial \mathbf{z}} & \frac{\partial h_2}{\partial \lambda} & \frac{\partial h_2}{\partial \beta} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{z} \\ \Delta \lambda \\ \Delta \beta \end{bmatrix} = - \begin{bmatrix} \zeta - \mathbf{z} \\ h_1 \\ h_2 \end{bmatrix} \tag{81}$$

This is solved using the bordered algorithm approach used for the time-independent case as it makes the partial derivatives easier to calculate using the same finite difference formulation.

6. Projection methods for time-independent case

Taking the principles used in the previous section for the time-discrete case it is possible to develop a computationally inexpensive method for tracking a bifurcation on large time-independent dynamical systems. By developing a way of extracting a basis, either the augmented system or the test function equations could be projected onto this basis in order to reduce its size. The process of forming a projected augmented system is presented here, the process of projection for the test function initially presented in Section 4.2 follows similarly.

6.1. Fully projected augmented system

The concepts developed in the Recursive Projection Method can also be applied to the original time-independent augmented system. The primary principle of RPM is that a basis is formed that spans the unstable subspace. In the original RPM this basis is created by performing a QR decomposition on the past iterates. The past iterates were used as an approximation to the dominant eigenvectors, which were unavailable. In the original time-independent continuation method an approximation to the dominant eigenvectors is available from ARPACK. The eigenvectors computed using ARPACK, once an unstable flow solution has been found, can be used to form the basis, by performing a QR decomposition on them. The original time-independent augmented equations, can then be projected onto this basis in the same way the time-discrete bifurcation equations were projected. This reduces the system from a size of $(3N+2)\times(3N+2)$ to $(3m+2)\times(3m+2)$, where m is the size of the basis and results in a novel bifurcation tracking method that has a low computational cost.

If all of the equations are projected onto this basis then the set of equations to be solved becomes

$$\mathbf{Z}^T\mathbf{R} = 0, \tag{82}$$

$$\mathbf{Z}^T\mathbf{J}\mathbf{Z}\mathbf{Z}^T\mathbf{v}_1 + \omega\mathbf{Z}^T\mathbf{v}_2 = 0, \tag{83}$$

$$\mathbf{Z}^T\mathbf{J}\mathbf{Z}\mathbf{Z}^T\mathbf{v}_2 - \omega\mathbf{Z}^T\mathbf{v}_1 = 0, \tag{84}$$

$$\phi\cdot\mathbf{Z}^T\mathbf{v}_1 - 1 = 0, \tag{85}$$

$$\phi\cdot\mathbf{Z}^T\mathbf{v}_2 = 0. \tag{86}$$

Setting $\tilde{\mathbf{Y}} = \mathbf{Z}^T\mathbf{Y}$, $\tilde{\mathbf{R}} = \mathbf{Z}^T\mathbf{R}$, $\tilde{\mathbf{J}} = \mathbf{Z}^T\mathbf{J}\mathbf{Z}$, $\tilde{\mathbf{v}}_1 = \mathbf{Z}^T\mathbf{v}_1$ and $\tilde{\mathbf{v}}_2 = \mathbf{Z}^T\mathbf{v}_2$, the equations can be written as

$$\tilde{\mathbf{R}} = 0, \tag{87}$$

$$\tilde{\mathbf{J}}\tilde{\mathbf{v}}_1 + \omega\tilde{\mathbf{v}}_2 = 0, \tag{88}$$

$$\tilde{\mathbf{J}}\tilde{\mathbf{v}}_2 - \omega\tilde{\mathbf{v}}_1 = 0, \tag{89}$$

$$\phi \cdot \tilde{\mathbf{v}}_1 - 1 = 0, \tag{90}$$

$$\phi \cdot \tilde{\mathbf{v}}_2 = 0. \tag{91}$$

Following the same process as described in Section 4 and applying Newton's method to this set of equations gives

$$\begin{bmatrix} \tilde{\mathbf{J}} & \mathbf{0} & \mathbf{0} & 0 & \dfrac{\partial \tilde{\mathbf{R}}}{\partial \lambda} \\ \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \tilde{\mathbf{Y}}} & \tilde{\mathbf{J}} & \omega \mathbf{I}_m & \tilde{\mathbf{v}}_2 & \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \lambda} \\ \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \tilde{\mathbf{Y}}} & -\omega \mathbf{I}_m & \tilde{\mathbf{J}} & -\tilde{\mathbf{v}}_1 & \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \lambda} \\ \mathbf{0} & \phi^T & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \phi^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta \tilde{\mathbf{Y}} \\ \Delta \tilde{\mathbf{v}}_1 \\ \Delta \tilde{\mathbf{v}}_2 \\ \Delta \omega \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \tilde{\mathbf{R}} \\ \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1 + \omega \tilde{\mathbf{v}}_2 \\ \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2 - \omega \tilde{\mathbf{v}}_1 \\ \phi \cdot \tilde{\mathbf{v}}_1 - 1 \\ \phi \cdot \tilde{\mathbf{v}}_2 \end{bmatrix}. \tag{92}$$

Then applying the bordering algorithm to solve this matrix system, not in this case to reduce the computational burden, but to allow differencing techniques to be used to compute the Hessian terms gives the following algorithm defined by equations (93) to (102).

$$\tilde{\mathbf{J}}\tilde{\mathbf{a}} = -\tilde{\mathbf{R}}, \tag{93}$$

$$\tilde{\mathbf{J}}\tilde{\mathbf{b}} = -\frac{\partial \tilde{\mathbf{R}}}{\partial \lambda}, \tag{94}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega \mathbf{I}_m \\ -\omega \mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{c}} \\ \tilde{\mathbf{d}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_2 \\ -\tilde{\mathbf{v}}_1 \end{bmatrix}, \tag{95}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega \mathbf{I}_m \\ -\omega \mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{e}} \\ \tilde{\mathbf{f}} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \tilde{\mathbf{Y}}}\tilde{\mathbf{a}} \\ -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \tilde{\mathbf{Y}}}\tilde{\mathbf{a}} \end{bmatrix}, \tag{96}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega \mathbf{I}_m \\ -\omega \mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{g}} \\ \tilde{\mathbf{h}} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \tilde{\mathbf{Y}}}\tilde{\mathbf{b}} - \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \lambda} \\ -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \tilde{\mathbf{Y}}}\tilde{\mathbf{b}} - \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \lambda} \end{bmatrix}, \tag{97}$$

$$\Delta\lambda = \frac{(\phi\cdot\tilde{\mathbf{c}})(\phi\cdot\tilde{\mathbf{f}}) - (\phi\cdot\tilde{\mathbf{e}})(\phi\cdot\tilde{\mathbf{d}}) + (\phi\cdot\tilde{\mathbf{d}})}{(\phi\cdot\tilde{\mathbf{d}})(\phi\cdot\tilde{\mathbf{g}}) - (\phi\cdot\tilde{\mathbf{c}})(\phi\cdot\tilde{\mathbf{h}})}, \tag{98}$$

$$\Delta\omega = \frac{(\phi\cdot\tilde{\mathbf{h}})\Delta\lambda + (\phi\cdot\tilde{\mathbf{f}})}{\phi\cdot\tilde{\mathbf{d}}}, \tag{99}$$

$$\Delta\tilde{\mathbf{Y}} = \tilde{\mathbf{a}} + \Delta\lambda\tilde{\mathbf{b}}, \tag{100}$$

$$\Delta\tilde{\mathbf{v}}_1 = \tilde{\mathbf{e}} + \Delta\lambda\tilde{\mathbf{g}} - \Delta\omega\tilde{\mathbf{c}} - \tilde{\mathbf{v}}_1, \tag{101}$$

$$\Delta\tilde{\mathbf{v}}_2 = \tilde{\mathbf{f}} + \Delta\lambda\tilde{\mathbf{h}} - \Delta\omega\tilde{\mathbf{d}} - \tilde{\mathbf{v}}_2. \tag{102}$$

The algorithm requires the computation of quantities such as

$$\frac{\partial\tilde{\mathbf{J}}}{\partial\tilde{\mathbf{Y}}}\tilde{\mathbf{v}}_1\tilde{\mathbf{a}}. \tag{103}$$

It is not clear how quantities such as this would be computed and initial investigations using this for the RANS system proved unsuccessful with $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ proving to be the cause of the problem as the inaccuracies in the basis meant that the values computed for $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ were not accurate enough for convergence to be achieved. Also preliminary investigations with a simple test case showed that as the number of unknowns increased, using this method with a small basis provided less accurate results even though the eigenvalues computed using the projected Jacobian continued to match those of the full Jacobian. Therefore, a compromise between computational efficiency and accuracy was sought and is discussed below.

6.2. Partially projected augmented system

The compromise to this situation arose by way of projecting all but the first equation onto the basis. This was decided upon based on the fact that the projected Jacobian remained fairly accurate, therefore it was assumed that the eigenvalue equations were correct and the problems were caused by the projection of the flow equations onto the unstable basis. This resulted in a new bifurcation tracking method that was computationally less expensive than the original, unprojected methods but with a similar robustness and accuracy level.

The equation set to be solved in this case is

$$\mathbf{R} = 0, \tag{104}$$

$$\tilde{\mathbf{J}}\tilde{\mathbf{v}}_1 + \omega\tilde{\mathbf{v}}_2 = 0, \tag{105}$$

$$\tilde{\mathbf{J}}\tilde{\mathbf{v}}_2 - \omega\tilde{\mathbf{v}}_1 = 0, \tag{106}$$

$$\phi \cdot \tilde{\mathbf{v}}_1 - 1 = 0, \tag{107}$$

$$\phi \cdot \tilde{\mathbf{v}}_2 = 0. \tag{108}$$

and Newton's method for this system is

$$\begin{bmatrix} \mathbf{J} & \mathbf{0} & \mathbf{0} & 0 & \dfrac{\partial \mathbf{R}}{\partial \lambda} \\ \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \mathbf{Y}} & \tilde{\mathbf{J}} & \omega\mathbf{I}_m & \tilde{\mathbf{v}}_2 & \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \lambda} \\ \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \mathbf{Y}} & -\omega\mathbf{I}_m & \tilde{\mathbf{J}} & -\tilde{\mathbf{v}}_1 & \dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \lambda} \\ \mathbf{0} & \phi^T & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \phi^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\tilde{\mathbf{Y}} \\ \Delta\tilde{\mathbf{v}}_1 \\ \Delta\tilde{\mathbf{v}}_2 \\ \Delta\omega \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \tilde{\mathbf{R}} \\ \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1 + \omega\tilde{\mathbf{v}}_2 \\ \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2 - \omega\tilde{\mathbf{v}}_1 \\ \phi \cdot \tilde{\mathbf{v}}_1 - 1 \\ \phi \cdot \tilde{\mathbf{v}}_2 \end{bmatrix}. \tag{109}$$

The algorithm then changes as the derivatives with respect to the projected flow-variable vector, $\mathbf{Z}^T\mathbf{Y}$, are no longer necessary, instead these terms become derivatives with respect to the original flow variable vector, $\mathbf{Y}$. This can be achieved using the existing differencing techniques. The resulting bordering algorithm is defined as follows.

$$\mathbf{J}\mathbf{a} = -\mathbf{R}, \tag{110}$$

$$\mathbf{J}\mathbf{b} = -\frac{\partial \mathbf{R}}{\partial \lambda}, \tag{111}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega\mathbf{I}_m \\ -\omega\mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{c}} \\ \tilde{\mathbf{d}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_2 \\ -\tilde{\mathbf{v}}_1 \end{bmatrix}, \tag{112}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega\mathbf{I}_m \\ -\omega\mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{e}} \\ \tilde{\mathbf{f}} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_1}{\partial \mathbf{Y}}\mathbf{a} \\ -\dfrac{\partial \tilde{\mathbf{J}}\tilde{\mathbf{v}}_2}{\partial \mathbf{Y}}\mathbf{a} \end{bmatrix}, \tag{113}$$

$$\begin{bmatrix} \tilde{\mathbf{J}} & \omega \mathbf{I}_m \\ -\omega \mathbf{I}_m & \tilde{\mathbf{J}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{g}} \\ \tilde{\mathbf{h}} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \tilde{\mathbf{J}} \tilde{\mathbf{v}}_1}{\partial \mathbf{Y}} \mathbf{b} - \frac{\partial \tilde{\mathbf{J}} \tilde{\mathbf{v}}_1}{\partial \lambda} \\ -\frac{\partial \tilde{\mathbf{J}} \tilde{\mathbf{v}}_2}{\partial \mathbf{Y}} \mathbf{b} - \frac{\partial \tilde{\mathbf{J}} \tilde{\mathbf{v}}_2}{\partial \lambda} \end{bmatrix}, \tag{114}$$

$$\Delta\lambda = \frac{(\phi \cdot \tilde{\mathbf{c}})(\phi \cdot \tilde{\mathbf{f}}) - (\phi \cdot \tilde{\mathbf{e}})(\phi \cdot \tilde{\mathbf{d}}) + (\phi \cdot \tilde{\mathbf{d}})}{(\phi \cdot \tilde{\mathbf{d}})(\phi \cdot \tilde{\mathbf{g}}) - (\phi \cdot \tilde{\mathbf{c}})(\phi \cdot \tilde{\mathbf{h}})}, \tag{115}$$

$$\Delta\omega = \frac{(\phi \cdot \tilde{\mathbf{h}})\Delta\lambda + (\phi \cdot \tilde{\mathbf{f}})}{\phi \cdot \tilde{\mathbf{d}}}, \tag{116}$$

$$\Delta\mathbf{Y} = \mathbf{a} + \Delta\lambda \mathbf{b}, \tag{117}$$

$$\Delta\tilde{\mathbf{v}}_1 = \tilde{\mathbf{e}} + \Delta\lambda\tilde{\mathbf{g}} - \Delta\omega\tilde{\mathbf{c}} - \tilde{\mathbf{v}}_1, \tag{118}$$

$$\Delta\tilde{\mathbf{v}}_2 = \tilde{\mathbf{f}} + \Delta\lambda\tilde{\mathbf{h}} - \Delta\omega\tilde{\mathbf{d}} - \tilde{\mathbf{v}}_2. \tag{119}$$

The partial derivative then becomes

$$\frac{\partial \tilde{\mathbf{J}}}{\partial \mathbf{Y}} \tilde{\mathbf{v}}_1 \mathbf{a} = \mathbf{Z}^T \frac{\partial \mathbf{J}}{\partial \mathbf{Y}} \left(\mathbf{Z}\tilde{\mathbf{v}}_1\right) \mathbf{a}. \tag{120}$$

which is more straightforward to compute, as differencing can be used. The system has had to be increased to a size of $(N + 2m + 2) \times (N + 2m + 2)$ to allow the current solution method to be employed however this is still much smaller than the original system and the whole solution process using the algorithm given in the equations above requires only two solves of the $N \times N$ Jacobian matrix $\mathbf{J}$.

7. Results

7.1. Validation of the Recursive Projection Method

The ability of the Recursive Projection Method to allow time integration methods to converge onto unstable solutions is shown here. This is something that the usual explicit CFD solver would not usually be able to do thus increasing its range of applicability and allowing its use within a continuation method. As the RPM works by extracting the part of the solution that is causing convergence difficulties it has the greatest effect in the unstable parameter range and also in the stable parameter range that is close to the bifurcation point. The theory behind the RPM was presented in Section 3.2.

Shown in Figure 1 is the comparison of the convergence history of the four-stage, second order Runge-Kutta method both on its own and supplemented with the RPM applied to a NACA 0012 aerofoil at an angle of attack of $4.7°$, a Mach number of 0.7 and a Reynolds number of 1×10^7. These parameter values give a solution which is just stable as shown by the eigenvalues in Figure 3. For this case the inputs used for RPM were: $\nu_{max} = 50$, $\texttt{KA} = 1000$ and $k_v = 10$. The Jacobian was calculated every ν_{max} iterations and the basis was re-orthogonalised at these points also, irrespective of whether any new vectors were added. As can be seen, the effect of the Recursive Projection method in this case is to increase the convergence rate by extracting the part of the solution that is hindering the convergence. On convergence of the solution the total size of the basis is four, $m = 4$. This was done in four separate extractions, meaning that the basis was increased by one vector on four separate occasions. This can be seen by the four sudden drops in the residual, although the second time this occurs, at 1800 iterations, the drop is immediately reversed. The reason for an upward spike in the residual at the time of basis extraction is usually caused by an inaccurate basis, meaning the vector extracted did not correspond to the least stable part of the solution. This suggests the value used for KA in equation (23) is too small as this parameter controls the accuracy required for acceptance of a vector into the basis. However, care must be taken not to set it too high as the criteria may never be met and no vectors would be added. Whilst this would not prevent convergence in this case, as the solution is stable and the time integration method converges anyway, for unstable solutions this becomes important as evidenced in the next test case. There are also several other inputs which are user dependent and can affect the effectiveness of the method such as ν_{max} which indicates how often new vectors are sought and k_v which is the number of past iterates used to search for new basis vectors and as such limits the maximum number of new vectors that can be extracted. Although the basis was only increased by a single vector each time, by the time the solution is converged calculation of the eigenvalues of the solution projected onto the basis indicates two complex pairs of eigenvalues.

Although the RPM enables convergence acceleration, this is not its primary use and the comparison is made here with a non-preconditioned solver without convergence acceleration. Shown in Figure 2, is the comparison of the convergence history between the Runge-Kutta method with and without

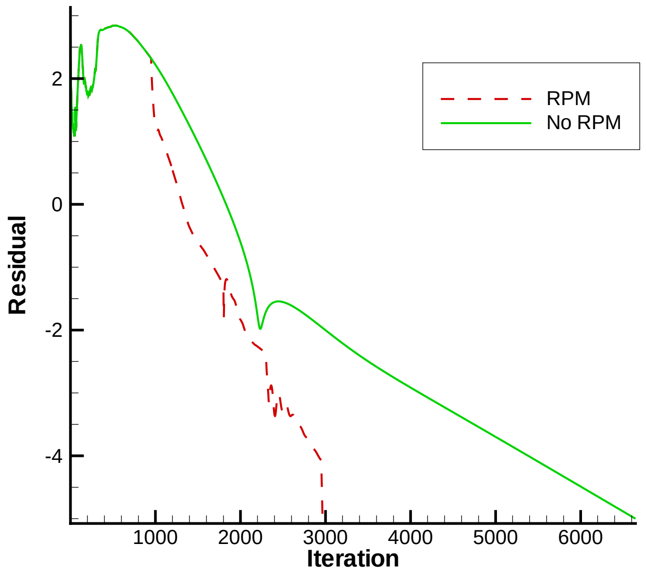

Figure 1: Comparison of convergence history for a NACA 0012 aerofoil at an $\alpha = 4.7°$, $M = 0.7$, $Re = 1 \times 10^7$ with and without RPM.

RPM applied to a NACA 0012 aerofoil at an angle of attack of 7.0°, a Mach number of 0.7 and a Reynolds number of 1×10^7. This is a point well into the unstable regime as shown by the eigenvalues in Figure 3. As can be seen the underlying time integration technique does not converge on its own but through the addition of the RPM convergence can be achieved. The values for the RPM inputs used were $\nu_{max} = 50$ and $\texttt{KA} = 100$. Again, the Jacobian was calculated every ν_{max} iterations and the basis was re-orthogonalised at these points also, irrespective of whether any new vectors were added. The reason for lowering the value of `KA`, which is the criterion which dictates when a new vector is added to the basis is to allow a vector to be extracted before the residual stalls, as after this point the vectors do not become any more accurate as the Jacobian is not increasing in accuracy. This is a key parameter in many cases as it must be set to a value that allows the Jacobian to converge far enough that the vectors are not so inaccurate the cause a massive rapid divergence but must also be low enough to allow a vector to be found before the convergence stalls. The final size of the basis on convergence of the solution was six, $m = 6$. This was achieved through three separate extractions. Firstly, two vectors were extracted after 2000 iterations, then a further three vectors were added to the basis at 3500 iterations and one final vector was added at 5000 iterations. The low value of `KA` needed to start the RPM is the reason for the large oscillations when the final vector is added, as its addition renders the basis inaccurate and it is only after a certain number of re-orthogonalisations and further convergence that the oscillations are overcome.

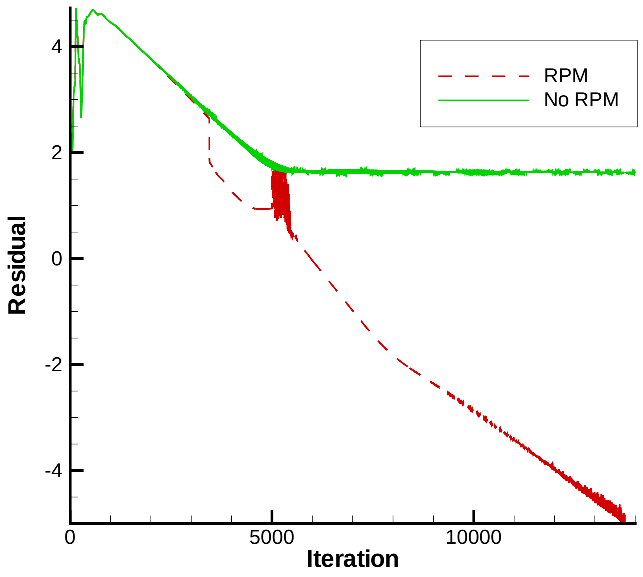

Figure 2: Comparison of convergence history for a NACA 0012 aerofoil at an $\alpha = 7.0°$, $M = 0.7$, $Re = 10.0 \times 10^6$ with and without RPM.

Figure 4 shows an excerpt from the continuation run for this test case. In this case the angle of attack was used as the continuation parameter, $\lambda = \alpha$. Figure 4 shows the number of iterations needed to converge at each step during the continuation run both with and without RPM and also the lift coefficient for each angle of attack. Without RPM it is not possible to trace the complete continuation curve as there are intervals where the unmodified

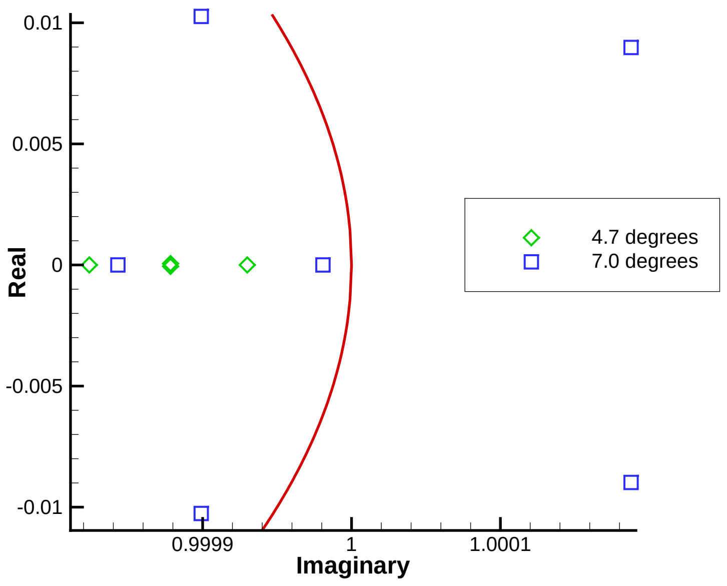

Figure 3: Largest magnitude eigenvalues computed by RPM for NACA 0012 aerofoil at $\alpha = 4.7°$ and $\alpha = 7.0°$ with $M = 0.7$, $Re = 10.0 \times 10^6$.

Runge-Kutta method does not converge. The RPM inputs for this case were $\nu_{max} = 50$, $\texttt{KA} = 1000$ and $k_v = 10$. It is possible to use a larger value for KA in this case as the final basis and solution is reused from the previous continuation step meaning that there is no need to set a lower value of KA to extract a basis. This also explains the greater number of iterations needed on the first step as in both cases (with and without RPM) the solution is started from a constant initial flow field and for the RPM case there is no basis. There is a noticeable increase in the number of iterations needed to converge on the parameter values around the bifurcation point, which occurs between 4.92° and 4.93°. This is expected as this is the region of parameter values where there will be a complex pair of eigenvalues with a magnitude close to or greater than unity.

Although not shown here the above case and cases involving different flow conditions around the same aerofoil were computed for different KA, ν_{max} and k_v values. It was found that the only requirement for k_v was that it should

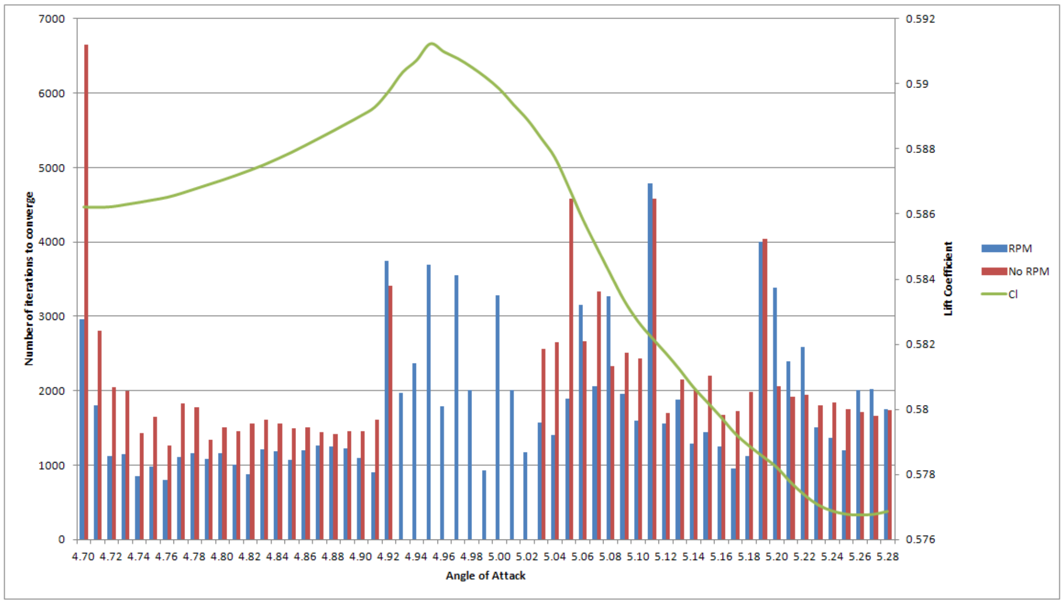

Figure 4: Comparison of convergence history for a NACA 0012 aerofoil for continuation in angle of attack with and without RPM.

be a value larger than 2 to ensure that a complex pair of eigenvectors was not missed, however, at no point was more than two eigenvectors extracted at any one time. The results presented in this section suggest that the basis extracted by the RPM is a good representation of the subspace containing the least stable dynamics. This provides confidence that the bifurcation can be located by solving the equations on this subspace alone. However, if, as expected, it is necessary for the subspace to represent the least stable part of the eigenspectrum as opposed to the unstable part alone in order to allow movement of the solution as the bifurcation is tracked; then it may be necessary to manipulate the value of `KA` during the continuation process to ensure that extra vectors are added to the basis. This does not appear to affect the accuracy of the final basis.

7.2. Bifurcation tracking of shock induced oscillations

The onset of shock induced oscillations is associated with a Hopf bifurcation and, as such, continuation methods can be used to identify the parameter values where this unsteady flow regime begins. The original continuation methods described in section 3.1 can be used to calculate rough values where this bifurcation occurs and the accuracy of the bifurcation point will depend on the step length used as it is unlikely that the step will give the exact parameter value where the bifurcation occurs. Furthermore, computing the stability boundary for a range of second parameters would require a continuation run to be performed for a selection of second parameter values. This method was used in [6]. Before this, the previous computational evaluation of stability used standard time integration and inspection of time histories to decide on the stability. Through the use of bifurcation tracking methods the stability boundary can be computed directly, giving the exact bifurcation point without the need for multiple continuation runs.

Calculations were carried out for the flow past a NACA 0012 aerofoil using the Spalart-Allmaras turbulence model. The angle of attack was used as the first continuation parameter, $\lambda = \alpha$, and Mach number was used as the second continuation parameter, $\gamma = M$ with a Reynolds number of 1×10^7. Figure 5 shows the predicted boundary of the flow stability in terms of angle of attack and Mach number. The required converged solution tolerance was set at 10^{-8} for the time-independent methods and 10^{-4} for the time-discrete methods.

All six methods show fairly good agreement and whilst the time-independent methods fail to find any solutions in certain areas, the time-discrete methods do not suffer from this failure. The time-discrete methods and the projected time-independent methods do begin to diverge from the solutions found using the full time-independent methods as the continuation progresses. The most likely reason for this is the current inability to update the basis accurately coupled with the previous solutions not being converged sufficiently. The inability to accurately update the basis in the current implementation means that the as the solution evolves at the next parameter values the basis will not be an accurate representation of the unstable subspace. This has the knock-on effect of preventing the subsequent solution from converging as much, and as the previous solution is used as an initial guess for the next bifurcation point, means that convergence onto the next solution is more difficult. This results in the irregular boundary that is seen for the projected methods. Convergence issues also result in the bifurcation tracking methods involving a test function producing an irregular boundary.

The experimental buffet boundary is also shown in Figure 5 as computed by McDevitt and Okuno [42]. Clearly, this shows that the buffet boundary computed in this work does not match the experimental results. This will be

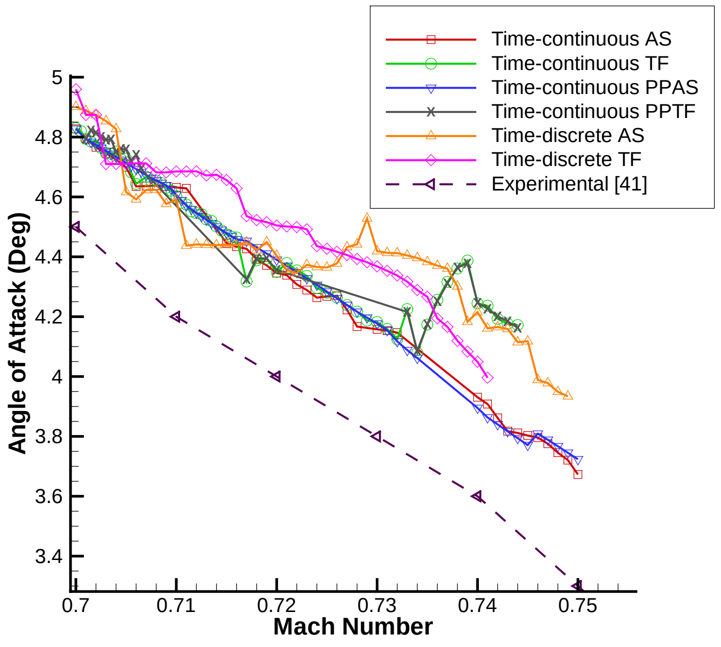

Figure 5: Shock induced oscillations for a NACA 0012 aerofoil $Re = 1 \times 10^7$. AS: Augmented System, TF: Test Function, PPAS: Partially Projected Augmented System, PPTF: Partially Projected Test Function

due to the underlying solver and mesh. However, as previously stated, the primary goal of this work is the development of a computationally efficient bifurcation tracking method. Therefore, the main concern is the accuracy of the projected methods with respect to the original, unprojected, methods and not with respect to the experimental results.

The CPU time taken for all the methods applied to this test case are given in Table 1. Again, the cost of the augmented system and test functions for a given projection level are similar, as would be expected given the size of the system is the same. The partially-projected methods show a similar decrease in computational cost as for bifurcation tracking in Reynolds number and angle of attack however the fully-projected methods cost half as much as they did for that case. This is probably due to the fact the gradient of the stability boundary is more constant for this case meaning the basis does not change greatly between any two bifurcation points.

Method	CPU Time (s)	% time of Time-independent AS
Time-independent AS	3.06×10^5	100
Time-independent TF	3.19×10^5	104.1
Time-independent PPAS	1.37×10^4	4.5
Time-independent PPTF	1.39×10^4	4.5
Time-discrete AS	300	0.10
Time-discrete TF	277	0.09

Table 1: Time taken for the bifurcation tracking methods for shock induced oscillations.

The following results present bifurcation tracking of the flutter stability boundary with a shape parameter. This reflects the idea of how a change in design could move the unwanted bifurcation outside the operating range.

7.3. Bifurcation tracking in camber and angle of attack

The partially-projected bifurcation tracking methods was not implemented for the test cases involving bifurcation tracking with shape parameters because for the time-independent methods ARPACK, which was used to compute the basis, had difficulty converging. This meant that although it found the critical eigenpair to sufficient accuracy more vectors, which are needed to perform bifurcation tracking were unobtainable.

This section compares the performance of the bifurcation tracking methods at mapping the stability boundary in a parameter space defined by both angle of attack and camber. The results were carried out for the flow past a

NACA 4-digit aerofoil with a thickness of 12% chord, with the initial aerofoil being a NACA 0012. The Spalart-Allmaras turbulence model was used and the partially-projected time-independent methods were not applied to this case.

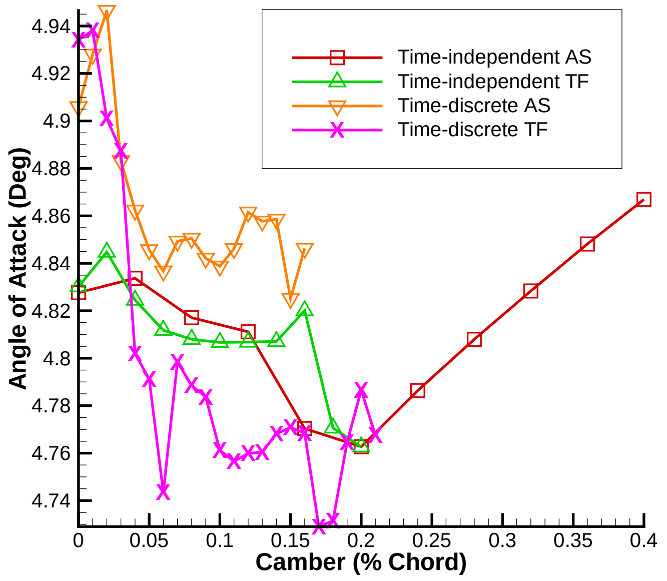

Figure 6: Stability limit of camber and angle of attack for a NACA 4-digit aerofoil with a thickness of 12% of the chord at $M = 0.7$ and $Re = 1 \times 10^7$.

The first continuation parameter used was the angle of attack whilst the second continuation parameter was camber. The tracking was performed for a range of cambers between 0% chord and 0.4% chord with the maximum camber location being kept constant at 40% chord. It was found that some of the methods required a smaller step size in order to progress with tracking the bifurcation. The two time-discrete methods used an initial step size of 0.01% chord whilst the time-independent augmented system method used a step size of 0.04% chord and the time-independent test function method used a step size of 0.02% chord The Mach number was fixed at 0.7 and the Reynolds number was fixed at 1×10^7. The required converged solution tolerance was set at 10^{-8} for the time-independent methods and 10^{-4} for the time-discrete methods.

The results, shown in Figure 6, show less of an agreement in value between the time-independent and time-discrete methods compared with the previous test cases but the general trend of the time-discrete methods agree with the trend of the time-independent methods. Tracking a bifurcation in camber proved difficult for all the methods with no method being able to track the bifurcation beyond a camber of 0.4% chord. This figure highlights the fact that the initial solutions for the time-independent and time-discrete systems do not match. Although this is true for all the test cases presented so far, it is more apparent for this case due to the scale of the axes. The initial solutions are always the same where the starting conditions define a flow with a Mach number of 0.7 and a Reynolds number of 1×10^7 around a NACA 0012 aerofoil.

7.4. Bifurcation tracking in thickness and angle of attack

This section compares the performance of the bifurcation tracking methods at mapping the stability boundary in a parameter space defined by both angle of attack and thickness. The results were carried out for the flow past a symmetric NACA 4-digit aerofoil, again starting with the NACA 0012 aerofoil. For this case the Spalart-Allmaras turbulence model was implemented and the partially-projected time-independent methods were not applied.

The first continuation parameter used was the angle of attack whilst the second continuation parameter was thickness. The tracking was performed for a range of thicknesses between 12% chord and 15% chord with a constant step size for all methods of 0.1% chord. The Mach number was fixed at 0.7 and the Reynolds number was fixed at 1×10^7. The required converged solution tolerance was set at 10^{-8} for the time-independent methods and 10^{-4} for the time-discrete methods.

All four tracking methods were initialised by performing continuation in angle of attack at a thickness of 12% chord, giving the NACA 0012 aerofoil, using the Newton arclength continuation method for the time-independent systems and the Recursive Projection Method for the time-discrete systems. The continuation was performed until the eigenvalue solver had detected that a bifurcation had occurred. Again, the results, shown in Figure 7, show less of an agreement between the time-independent and time-discrete methods compared with the first two test cases but the general trend of the time-discrete methods agree with the trend of the time-independent methods.

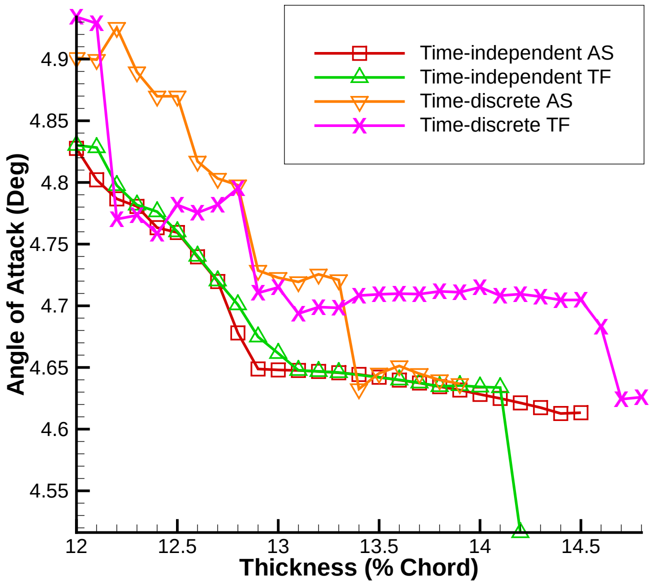

Figure 7: Stability limit of thickness and angle of attack for a NACA 4-digit symmetric aerofoil at $M = 0.7$ and $Re = 1 \times 10^7$.

7.5. Bifurcation tracking in thickness and Mach number

This section compares the performance of the full time-independent and fully-projected time-discrete bifurcation tracking methods at mapping the stability boundary in a parameter space defined by both Mach number and thickness. The results were carried out using the Spalart-Allmaras turbulence model for the flow past a symmetric NACA 4-digit aerofoil with the initial aerofoil being a NACA 0012. The first continuation parameter used was the Mach number whilst the second continuation parameter was thickness. The tracking was performed for a range of thicknesses between 12% chord and 15% chord. It was necessary to use a smaller step size for the time-discrete methods in order for the tracking to progress beyond the first point. Therefore a step size of 0.1% chord has been used for the time-discrete methods whilst a step size of 0.25% chord has been used for the time-independent methods. The angle of attack was fixed at 4.8° and the Reynolds number was fixed at 1×10^7. The required converged solution tolerance was set at 10^{-8}

for the time-independent methods and 10^{-4} for the time-discrete methods.

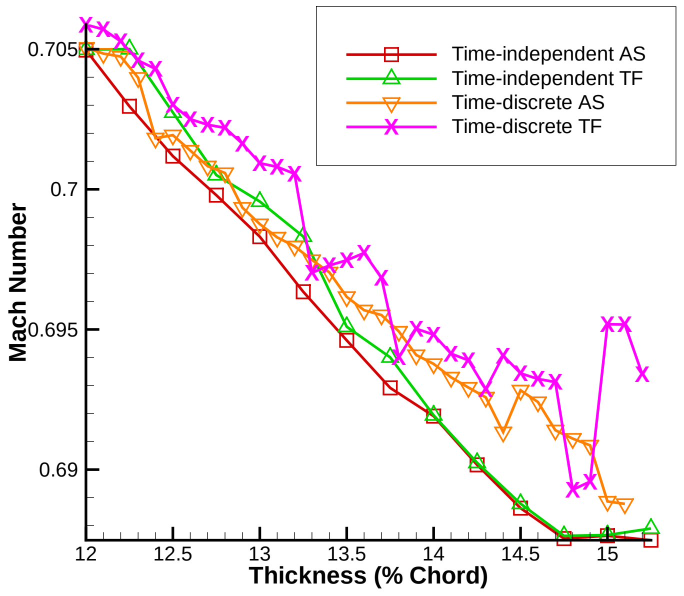

Figure 8: Stability limit of thickness and Mach number for a NACA 4-digit symmetric aerofoil at an angle of attack of $4.8°$ and $Re = 1 \times 10^7$.

As before, all four tracking methods were initialised by performing continuation in Mach number at a thickness of 12% chord, giving the NACA 0012 aerofoil, using the Newton arclength continuation method for the time-independent systems and the Recursive Projection method for the time-discrete systems. The tracking methods were implemented once the continuation method had detected that a bifurcation had occurred, which was achieved by computing the eigenvalues. The results, shown in Figure 8, show a good agreement between the time-independent and time-discrete methods especially at the start of the tracking methods. However, as continuation in the second parameter progresses, the results begin to diverge more as the basis for the time-discrete methods is not updated accurately enough. The overall trend of all the methods agree.

8. Conclusions

This work has presented the development of some computationally inexpensive bifurcation tracking methods and compared them to existing direct bifurcation tracking methods. These bifurcation tracking methods were developed with varying degrees of computational cost and accuracy. The full methods were applied to the time-independent RANS system and followed the stability limit that corresponded to the initial bifurcation from the continuation methods. The projected methods were applied to the RANS time-independent and time-discrete systems. These were computationally cheap but suffered from a degradation in the achievable accuracy. Despite this they proved successful in predicting the general trend of the stability limit. All of the direct methods solve a bifurcation system and locate exactly the bifurcation point for the given problem conditions but are limited, for a given implementation, to a single type of bifurcation, which in this case is a Hopf bifurcation although similar systems can be implemented that follow other types of bifurcation, such as a fold bifurcation.

Acknowledgements

This work was supported by funding from the Engineering and Physical Sciences Research Council (ESPRC) and Airbus UK.

[1] A. Jepson, Numerical hopf bifurcation, Ph.D. thesis, California Institute of Technology (1981).

[2] W. Govaerts, Numerical bifurcation analysis for odes, Journal of Computational and Applied Mathematics 125 (2000) 57–68.

[3] R. Seydel, Practical bifurcation and stability analysis: from equilibrium to chaos, Springer, 1994.

[4] Y. Kuznetsov, Elements of applied bifurcation theory, Springer New York, 1998.

[5] M. Poliashenko, C. K. Aidun, A direct method for computation of simple bifurcations, J. Comput. Physics (1995) 246–260.

[6] C. Wales, Continuation methods for high reynolds number compressible flows, Ph.D. thesis, University Of Bristol (2010).

[7] A. Salinger, E. Burroughs, R. Pawlowski, E. Phipps, L. Romero, Bifurcation tracking algorithms and software for large scale applications, Int. J. Bifurcation and Chaos 15 (2005) 1015–1032.

[8] A. Griewank, G. Reddien, The calculation of hopf points by a direct method, IMA J. Numerical Analysis 3 (1983) 295–303.

[9] D. Roose, V. Hlavacek, A direct method for the computation of hopf bifurcation points, SIAM J. Applied Mathematics 45 (1985) 879–894.

[10] A. Hussein, K. Chen, On efficient methods for detecting hopf bifurcation with applications to power system instability prediction, International Journal of Bifurcation and Chaos 13 (5) (2003) 1247–1262.

[11] T. Davis, A column pre-ordering strategy for the unsymmetric-pattern multifrontal method, ACM Trans. Math. Softw. 30 (2) (2004) 165–195.

[12] T. Davis, Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method, ACM Trans. Math. Softw. 30 (2) (2004) 196–199.

[13] T. Davis, I. Duff, A combined unifrontal/multifrontal method for unsymmetric sparse matrices, ACM Trans. Math. Softw. 25 (1) (1999) 1–20.

[14] T. Davis, I. Duff, An unsymmetric-pattern multifrontal method for sparse lu factorization, SIAM Journal on Matrix Analysis and Applications 18 (1) (1997) 140–158.

[15] J. Crouch, A. Garbaruk, D. Magidov, Predicting the onset of flow unsteadiness based on global stability, Journal of Computational Physics 224 (2) (2007) 924–940.

[16] J. Crouch, A. Garbaruk, D. Magidov, A. Travin, Origin of transonic buffet on aerofoils, J. Fluid Mech. 628 (2009) 357–369.

[17] J. Crouch, A. Garbaruk, D. Magidov, L. Jacquin, Global strucutre of buffeting flow on transonic airfoils, in: IUTAM Symposium on Unsteady Separated Flows and their Control, Vol. 14 of IUTAM Bookseries, Springer, Dordrecht, 2009, pp. 297–306.

[18] M. Juniper, A. Hanifi, V. Theofilis, Modal stability theory, Appl. Mech. Rev 66 (2).

[19] K. Badcock, M. Woodgate, B. Richards, Hopf bifurcation calculations for a symmetric airfoil in transonic flow, AIAA Journal 42 (5) (2004) 883–892.

[20] K. Badcock, M. Woodgate, B. Richards, Direct aeroelastic bifurcation analysis of a symmetric wing based on euler equations, Journal of Aircraft 42 (3) (2005) 731–737.

[21] M. Woodgate, K. Badcock, Fast prediction of transonic aeroelastic stability and limit cycles, AIAA Journal 45 (2007) 1370–1381.

[22] K. Badcock, M. A. Woodgate, Bifurcation prediction of large-order aeroelastic models, AIAA Journal 48 (6) (2010) 1037–1046.

[23] C. Wales, A. L. Gaitonde, D. P. Jones, D. Avitabile, A. R. Champneys, Numerical continuation of high reynolds number external flows, International Journal for Numerical Methods in Fluids 68 (2012) 135–159.

[24] G. Shroff, H. Keller, Stabilization of unstable procedures: the recursive projection method, SIAM Journal on numerical analysis (1993) 1099–1120.

[25] J. Moller, Studies of recursive projection methods for convergence acceleration of steady state calculations, Ph.D. thesis, Royal Institute of Technology (KTH) (2001).

[26] S. Gortz, J. Moller, Recursive projection method for efficient unsteady cfd simulations, European Congress on Comp. Meth. in App. Sci. and Eng.

[27] J. Moller, O. Runborg, P. Kevrekidis, K. Lust, I. Kevrekidis, Equation-free effective computation for discrete systems: A time stepper based approach., Interntional Journal of Bifurcation and Chaos 15 (2005) 975–996.

[28] C. Gear, I. Kevrekidis, C. Theodoropoulos, 'Coarse' integration/bifurcation analysis via microscopic simulators: Micro-Galerkin methods., Computers and Chemical Engineering 26 (2002) 941–963.

[29] M. Campobasso, M. Giles, Stabilization of linear flow solver for turbomachinery aeraeroelastic using recursive projection method, AIAA Journal 42 (9) (2004) 1765–1774.

[30] C. Wales, A. Gaitonde, D. Jones, An initial study of the flow around an aerofoil at high reynolds nnumber using continuation, International Journal of Bifurcation and Chaos 22.

[31] P. Spalart, S. Allmaras, A one equation turbulence model for aerodynamic flows, Recherche Aerospatiale French edition (1994) 5–5.

[32] C. Rumsey, S. Ying, Prediction of high lift: review of present CFD capability, Progress in Aerospace Sciences 38 (2) (2002) 145–180.

[33] J. Vassberg, E. Tinoco, M. Mani, O. Brodersen, B. Eisfeld, R. Wahls, J. Morrison, T. Zickuhr, K. Laflin, D. Mavriplis, Abridged summary of the third AIAA computational fluid dynamics drag prediction workshop, Journal of aircraft 45 (3) (2008) 781.

[34] A. Jameson, W. Schmidt, E. Turkel, Numerical solution of the Euler equations by finite volume methods using runge-kutta time-stepping schemes, AIAA paper 81-1259.

[35] R. Swanson, E. Turkel, On central-difference and upwind schemes, Journal of computational physics 101 (2) (1992) 292–306.

[36] L. Kral, Recent experience with different turbulence models applied to the calculation of flow over aircraft components, Progress in Aerospace Sciences 34 (7-8) (1998) 481–541.

[37] N. Christodoulou, Discrete hopf bifurcation for rungekutta methods, Applied Mathematics and Computation 206 (2008) 346–356.

[38] W. Govaerts, Stable solvers and block elimination for bordered systems., SIAM J. Matrix Anal. Appl. 12 (1991) 459,483.

[39] G. Golub, C. V. Loan, Matrix Computations, Johns Hopkins University Press, 1989.

[40] P. Wriggers, J. Simo, A general procedure for the direct computation of turning and bifurcation points, Int. J. for Numerical Methods in Engineering 30 (1990) 155–176.

[41] E. Doedel, H. Keller, J. Kernevez, Numerical analysis and control of bifurcation problems (i) bifurcation in finite dimensions, Int 1 (1991) 493–520.

[42] J. McDevitt, A. Okuno, Static and dynamic oressure measurements on a nacanaca airfoil in the ames high reynolds number facility, Tech. Rep. NASA-TP-2485, NASA (1985).