Science and Technology Facilities Council

A Review of Solution Continuation Techniques: Methods and Software

Technical Report 2068625-TN-07
Deliverable 4.1

Niall Bootland* Tyrone Rees* Sue Thorne†

September 2023

1 Introduction

In this report, we consider methodologies for continuation of solutions to nonlinear equations as we vary a problem parameter $\lambda \in \mathbb{R}$. These techniques are the focus of numerical bifurcation analysis, which looks to examine how solutions change with λ and analyse properties, such as stability, through numerical computations. In this work, we try to pick out the key approaches most relevant to modern computing for potentially large-scale application codes and give a basic understanding of the types of behaviours that one may observe.

We acknowledge primary references that have helped inform the writing of this report, the first being lecture notes by Patrick Farrell on bifurcation analysis and, in particular, deflation techniques [23]. Secondly, Hannes Uecker gives a good, open access and relatively short survey of key ideas in (classical) continuation methods and bifurcations in nonlinear PDEs in [62], with experiments focused on the `pde2path` software in MATLAB; a more comprehensive textbook by the same author is [61] and was often consulted. Thirdly, a series of lectures found in [41] by Herbert Keller, one of the pioneers of the field. Finally, in terms of relevant physical applications, we note that a wide-ranging review of numerical bifurcation methods applied to fluid dynamics is found in [19], which highlights their use in various complex real-world fluid flow problems; see also the more recent survey [59]. Other relevant textbooks include [1, 55, 18, 42].

We consider solving the nonlinear problem

$$F(u, \lambda) = 0, \tag{1}$$

for an unknown u and parameter λ. We can think of F as being a mapping $F\colon \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$, which will ultimately be our case after discretisation, but the problem can also be formulated at the infinite-dimensional level as a mapping between Banach spaces (see, e.g., [61]). Note that (1) is a generic form of problem and includes finding fixed points, such as for $G(u, \lambda) = u$ wherein

$$F(u, \lambda) := G(u, \lambda) - u = 0,$$

and solving partial differential equations (PDEs), for instance the (stationary) Swift–Hohenberg equation

$$F(u, \lambda) := \lambda u - (1 + \nabla^2)^2 u + u^2 - u^3 = 0,$$

with suitable boundary conditions, which is related to the study of thermal fluctuations within a fluid near the Rayleigh–Bénard convective instability [57]. A key feature of (1) being nonlinear is that it may have *multiple solutions*, or perhaps no solutions, and the number of solutions will, in general, depend on

*Scientific Computing Department, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK.
†Hartree Centre, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK. Email contact: `sue.thorne@stfc.ac.uk`

the parameter λ. Moreover, a naive method to find a solution of (1) may not result in the desired, or physically relevant, solution. Exploration of possible solutions as λ varies is therefore of key importance; this is addressed by numerical bifurcation analysis.

The key theoretical tool that justifies the concept of solution continuation and associated algorithms is the implicit function theorem (IFT). This provides local uniqueness of solution branches and their smoothness under certain conditions. While the IFT can be expressed in terms of Banach spaces, we detail a simpler finite dimensional version and minimise terminology to focus on the key points; a more rigorous statement is given in, for example, [62, Theorem 2.1]. Suppose that we have a mapping $F\colon \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ with $F(u, \lambda)$ continuously differentiable in u with derivatives given by the Jacobian matrix $J \in \mathbb{R}^{N\times N}$ (which we define later in (4)). Further, suppose we know a point (u_0, λ_0) such that $F(u_0, \lambda_0) = 0$, the Jacobian J is nonsingular here, and F is continuous in some neighbourhood of the point. Then there exists a neighbourhood around (u_0, λ_0) where a unique solution branch of (1) exists, given by $(H(\lambda), \lambda)$ for some continuous function $H(\lambda)$. Further, the smoothness (differentiability) of F transfers to give the same smoothness of H, and if F is analytic (and so has a convergent Taylor expansion) then so is H. Note that a key ingredient is that the Jacobian J is nonsingular, and hence invertible, and we will be interested in what happens when this is not the case. We further remark that the invertibility of J is sufficient but not necessary for a unique resolution $(H(\lambda), \lambda)$ in some edge cases. However, it is required to obtain appropriate smoothness.

Given a solution u_0, for parameter value λ_0, the goal of *(local) solution continuation* is to trace out the one-dimensional manifold (branch) of "nearby" solutions connected to a (regular) solution (u_0, λ_0). In its simplest form, the task is to determine a solution of

$$F(u, \lambda_0 + \delta\lambda) = 0,$$

for some small perturbation $\delta\lambda$ in the parameter space. That is, we wish to find the solution to a nearby problem, where the parameter has incremented, based on a previous solution u_0. So long as the parameter has not changed too much, and assuming such a solution exists (as given by the IFT), our knowledge of existing solutions helps us to traverse through parameter space in order to reach solutions which may otherwise be hard to find.

One may also be interested in the harder problem of mapping all (or at least a subset of) solutions to (1) over a given parameter range through *global solution continuation*, namely tracing out all solution branches of a *bifurcation diagram* (see, e.g., Figure 1). While local continuation is typically able to traverse along a single branch containing u_0, a bifurcation diagram can include several distinct branches which intersect at bifurcation points (locally there may be multiple nearby solutions) or branches which are entirely disconnected from each other. These issues pose additional challenges for solution continuation techniques.

A simple example of a bifurcation diagram is given in Figure 1 for the case $F(u, \lambda) = \lambda u - u^3 = 0$. When $\lambda < 0$ we have only one real solution, namely $u = 0$, while for $\lambda > 0$ we have two additional solutions $u = \pm\sqrt{\lambda}$. The point $u = 0$, $\lambda = 0$ is a bifurcation point and this situation represents the canonical form of a (supercritical) pitchfork bifurcation. Note that the derivative of F at $u = 0$ and $\lambda = 0$ is zero, namely

$$\frac{dF}{du}(u = 0, \lambda = 0) = \lambda - 3u^2|_{u=0,\lambda=0} = 0,$$

which is always the case at a bifurcation point and reveals (cf. the implicit function theorem outlined above) the non-uniqueness of solutions in a neighbourhood around such a point.

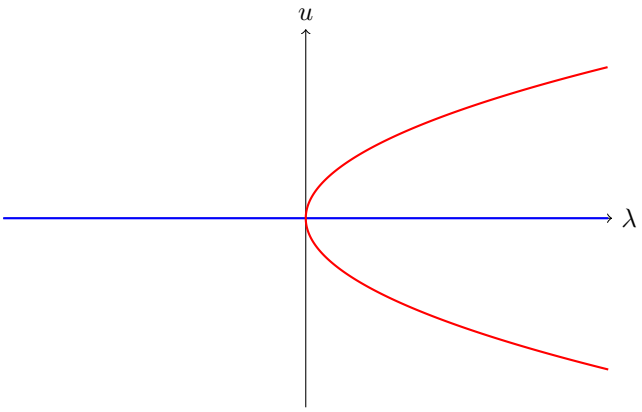

Figure 1: The bifurcation diagram for $F(u, \lambda) = \lambda u - u^3$. The blue line corresponds to the branch of solutions $u = 0$ while the red line corresponds to the branch of solutions $u = \pm\sqrt{\lambda}$. The point $u = 0$, $\lambda = 0$ is a bifurcation point and represents a (supercritical) pitchfork bifurcation.

In order to apply local solution continuation, most approaches use the past known solution to compute a *predictor*, which aims to approximately guess a new solution lying further along the branch, before employing a *corrector*, which takes this initial guess and tries to solve for a true solution of $F(u, \lambda) = 0$ nearby, possibly through use of an augmented system. Typically the corrector used is Newton's method, however, a cheaper alternative such as the chord method can also be used if derivatives are expensive or otherwise difficult to obtain.

We briefly review Newton's method to solve the nonlinear problem (1). Suppose we either fix λ and let $f(u) = F(u, \lambda)$, or we otherwise allow λ to vary and combine with the variable(s) u, so that we must solve

$$f(u) = 0. \tag{2}$$

In general, the system in (2) will correspond to a vector equation for a vector of unknowns u. In particular, when solving PDE problems where we must first discretise, u may represent values on a mesh. To be more precise, even if the original system is infinite dimensional, we suppose it has been discretised[1] so that $u \in \mathbb{R}^N$ and $f\colon \mathbb{R}^N \to \mathbb{R}^N$ for some finite dimension $N \geq 1$. Note that if the problem stems from discretisation of a PDE, then N can be very large. Newton's method to solve (2) starts with a given initial guess $u^{(0)}$ and iterates[2] with

$$u^{(k+1)} = u^{(k)} - J_f(u^{(k)})^{-1} f(u^{(k)}), \tag{3}$$

where J_f is the Jacobian matrix, namely the matrix of all first-order partial derivatives

$$J_f := \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial u_1} & \cdots & \frac{\partial f_N}{\partial u_N} \end{pmatrix}. \tag{4}$$

Note that the Newton step (3) is simply solving a linearisation of the problem at $u^{(k)}$ (in particular, tangent linearisation). In practice, we do not invert $J_f(u^{(k)})$ but rather solve the linear system

$$J_f(u^{(k)})\delta u^{(k+1)} = -f(u^{(k)}), \tag{5}$$

for the update $\delta u^{(k+1)} = u^{(k+1)} - u^{(k)}$. The method can be terminated when $\delta u^{(k+1)}$ becomes sufficiently small. If we swapped the Jacobian for a fixed matrix, say $J_f(u^{(0)})$, then (3) would become a chord

[1] It is not necessary to discretise, Newton's method can be applied to find the roots of a functional defined in a Banach space (where it is known as the Newton–Kantorovich method), in this case we require the existence of the Fréchet derivative of f. To keep the exposition simple, we avoid delving into the powerful tools of functional analysis required for the infinite dimensional problem.

[2] We will use a superindex to denote Newton iteration, with a general iteration index k, and reserve subindices for the continuation process, with a general iteration index n.

method. Approximating the Jacobian gives rise to other quasi-Newton methods, for example a finite difference approximation leads to the secant method in one dimension, while in higher dimensions one might employ Broyden's method. We note that Newton's method (and others) will not always converge and typically one must start with a good initial guess that is not too far from the desired solution in order to converge. The classical conditions under which Newton's method will converge are prescribed in the Newton–Kantorovich Theorem (see [53] and references within for related results). We note that there are alternatives to using Newton-like methods to solve nonlinear equations; for instance, one can use a nonlinear multigrid method such as the so-called full approximation scheme (FAS) [38].

Once a predictor–corrector pair has been chosen, local continuation is used to trace solution branches from a set of initial solutions but may fail at certain points. This may be due the non-existence of solutions if λ is fixed, for instance no solutions of $F(u, \lambda) = \lambda - u^2 = 0$ exist once $\lambda < 0$ ($\lambda = 0$ corresponds to a fold point and the bifurcation diagram would consist of only the red curve in Figure 1). Alternatively, the Jacobian may become singular (so $J_f(u^{(k)})$ cannot be inverted in (3)), which corresponds to a bifurcation point where multiple solutions exist in any surrounding neighbourhood. In practice, one needs to be careful when the Jacobian becomes nearly singular, in a small vicinity of a bifurcation point, and not only at the point itself. However, generically the local continuation method may simply skip over bifurcation points unnoticed without further checking, this leads to the necessity of branch switching. It is important to note that, in practice, we compute a sequence of solutions, that is points along a branch, and it is not always immediately clear which branch they may belong to. Further, since solutions are computed numerically, we note that some structures within bifurcation diagrams are unstable to perturbations, such as pitchfork bifurcations as demonstrated in Figure 2, and so one may need to take care in the interpretation of branches.

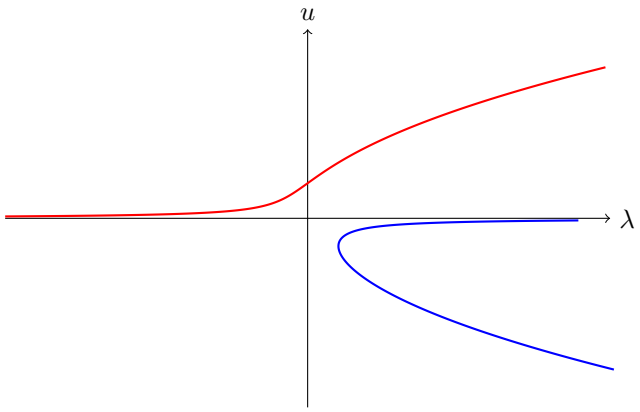

Figure 2: The bifurcation diagram for $F(u, \lambda) = \lambda u - u^3 + \delta$ for small perturbation $\delta > 0$. The blue line corresponds to the branch of solutions $u < 0$ while the red line corresponds to the branch of solutions $u > 0$. We note that the pitchfork bifurcation at $u = 0$, $\lambda = 0$ when $\delta = 0$ is unstable to the perturbation δ, breaking the symmetry and introducing a fold point as well as two non-intersecting branches instead.

An alternative approach, which avoids the need for connectedness of branches, is to modify the nonlinear function $F(u, \lambda)$ so as to avoid converging to already known solutions in the hope that we can find new solutions. This is done in *deflated continuation* [25, 26], where a deflation operator based on known solutions is applied to F and the modified system solved, typically with Newton's method. This can be especially effective for computing disconnected bifurcation diagrams but also avoids potential issues with switching of connected branches. Nonetheless, it is important to note that even deflation cannot guarantee that we find all possible solutions of (1), even for a fixed λ, in the general case. This is because convergence of Newton's method depends on the quality of the initial guess(es). While adaptions to Newton's method have been proposed to try to attain global convergence (see, e.g., [51, 52, 15] but note that no attempt can actually achieve this), such methods are typically slower and for large, complicated problems it may not be feasible to expend the computational effort to combine such approaches with deflation and find all solutions. Deflation can also be viewed as an alternative to branch switching, such that by removing solutions on known branches close to a bifurcation we can hope to converge to solutions on new branches emanating from the bifurcation point.

We note that techniques other than those of predictor–corrector form exist for solving the nonlinear problem (1). One of the main examples, which we mention as it is the basis of the software package `ManLab`, is the asymptotic numerical method (ANM) [35, 36]. This method uses a high-order Taylor series as a predictor, for which it is then claimed in [35] that no corrector is needed in general. A drawback of this method is that the nonlinear system must be recast in a polynomial form with a quadratic non-linearity, although later evolutions of software implementing this method leverage automatic differentiation to reduce the burden for the user. Furthermore, there is a smoothness assumption that allows the solution branch to be well-approximated by a high-order Taylor series, which may not be the case for more complex problems of practical interest. We do not discuss this type of approach further here, and focus on the more popular predictor–corrector methods.

Before continuing, we briefly discuss the issue of obtaining Jacobian matrices J, as in (4), which are typically required by the corrector. Broadly there are two classes of Jacobians that can be computed, *analytical* Jacobians and *numerical* Jacobians. Analytical Jacobians form a way to express the exact derivatives, this may simply be deriving the formulas by hand or by using a symbolic computation tool to provide such expressions for you, which are then hard coded. These are good for relatively simple problems or when you have only a few key fixed problems and the expressions can be checked but the approach may be unreliable or time consuming if such expressions need to calculated regularly. An alternative is to use automatic differentiation software [33, 45] to compute Jacobians. This will add an overhead to the code runtime and, further, needs careful implementation (unless using an existing tried-and-tested package) but benefits from being black-box and not requiring the user to specify the Jacobian (which may be error prone, especially for new applications). If using software such as FEniCS or Firedrake, which utilise unified form language (UFL), a much more efficient and automatic tool than standard algorithmic differentiation is provided by pyadjoint [50] (or its predecessor dolfin-adjoint [24]). This is the current state-of-the-art in modern software for problems posed in a variational framework.

The other option is to use a numerical approximation of the Jacobian, typically via finite differences. This is done by approximating each directional derivative, aligned with $u_1, \ldots, u_N$, and involves repeated evaluation of the function f. One must exploit sparsity of J for large-scale problems and try to minimise the number of calls to evaluate f. One way to do this is by employing a colouring approach [29, 5], but this may not always be particularly efficient. Within finite elements, a more efficient implementation can be developed by performing the finite differences at the local element level before assembling the approximate Jacobian. Note that for numerical Jacobians, we typically end up with a quasi-Newton method as the corrector, while for analytical Jacobians we can harness the faster convergence of Newton's method.

While it is possible that λ could be a vector, for instance being two scalar parameters that are changed simultaneously (as in the case of the unfolding of the pitchfork bifurcation in Figure 2), we shall not consider this case here. Further, we do not address semismooth problems that arise from inequality constraints; see [27] for details. We also discuss relatively little about time-stepping methods to find equilibrium solutions and their potential for use in bifurcation analysis, more details of which can be found in [60, 19].

The structure of the remainder of this report is as follows. In Section 2, we discuss the now well-established methods for continuation of connection solutions branches. This is complemented in Section 3 by discussion of deflation techniques to find and continue potentially disconnected branches. We then explore ideas from time-dependent problems and provide a dynamical systems point of view in Section 4, introducing the elementary bifurcations of steady states and, further, consideration of periodic orbits, which are related to bifurcations in maps. In Section 5, we then outline the special role that symmetry plays and discuss necessary considerations that should be taken into account. Software is considered in Section 6, focusing on those available to tackle more large-scale problems stemming from PDEs, before we summarise our conclusions in Section 7.

2 Continuation of connected solution branches

We first discuss the more classical style of continuation methods, which aim to trace a branch of solutions from a known initial solution u_0 or, analogously, several initial solutions. To begin we discuss options for the predictor which provides a new initial guess to continue the branch before discussing formulations which solve an augmented system based on a predictor–corrector pairing. We then discuss selecting the parameter step size, along with finding bifurcation points and branch switching at such points.

2.1 Predictors

The simplest, and most naive, predictor for the nearby solution along a branch at parameter value $\lambda_0+\delta\lambda$ is given by *natural continuation*, which chooses the known previous solution u_0 at λ_0. That is, we make a first-order approximation (Taylor expansion)

$$u(\lambda_0 + \delta\lambda) \approx u(\lambda_0), \tag{6}$$

where, by definition, $u(\lambda_0) = u_0$. The benefit of this approach is that it is essentially free and requires no work to compute the predictor. However, this may not be a particularly good approximation if the parameter step size $\delta\lambda$ is large and, further, may require the corrector (say, Newton's method) to take an unnecessarily large number of iterations to converge. Additionally, failure will occur at or beyond fold points where the solution branch curves back on itself and no (nearby) solutions exist as the parameter is increased (or decreased if $\delta\lambda < 0$).

Considering the (local) branch of solutions depending on the parameter λ as a curve $u(\lambda)$ then, as with Newton's method, a better predictor may be envisaged by considering the tangent to the curve. Equivalently, we can make a second-order Taylor approximation

$$u(\lambda_0 + \delta\lambda) \approx u(\lambda_0) + \frac{\partial u}{\partial \lambda}(\lambda_0)\delta\lambda. \tag{7}$$

However, we do not know the solution branch $u(\lambda)$ to immediately calculate the derivative so we must do some work to compute it. We are aiming to find u such that $F(u, \lambda) = 0$ and so taking the total derivative with respect to λ on both sides we obtain

$$\frac{\mathrm{d}F}{\mathrm{d}\lambda} = \frac{\partial F}{\partial u}\frac{\partial u}{\partial \lambda} + \frac{\partial F}{\partial \lambda} = 0,$$

along the curve $u(\lambda)$ defining the branch of solutions. This can be rearranged and evaluated at (u_0, λ_0) to yield $\frac{\partial u}{\partial \lambda}(\lambda_0)$ by solving the system

$$J_f(u_0)\frac{\partial u}{\partial \lambda}(\lambda_0) = -\frac{\partial F}{\partial \lambda}(u_0, \lambda_0), \tag{8}$$

where J_f is the Jacobian of $f(u) = F(u, \lambda_0)$ with λ_0 fixed. Assuming that F is not a black-box, so that its dependence on λ is explicit, we can compute the derivative on the right-hand side of (8) and then solving for $\frac{\partial u}{\partial \lambda}(\lambda_0)$ is similar to solving one Newton step; cf. (5). Combining (8) and (7) we arrive at *tangent continuation*.

The cost of finding the tangent vector stems primarily from the solution of (8), which is the same cost as a Newton step, so we would hope the corrector would converge faster (e.g., by two Newton steps) for the work to pay off. Since we are only finding a predictor, we need not use an exact tangent and a cheaper predictor, of almost as good quality, can be derived from secant approximation once we have two solutions on the branch. This can be thought of as a simple finite difference approximation of the tangent at λ_n based on the known previous solutions $u_n = u(\lambda_n)$ and $u_{n-1} = u(\lambda_{n-1})$:

$$\frac{\partial u}{\partial \lambda}(\lambda_n) \approx \frac{u(\lambda_n) - u(\lambda_{n-1})}{\lambda_n - \lambda_{n-1}} = \frac{u_n - u_{n-1}}{\lambda_n - \lambda_{n-1}}.$$

We now approximate the solution at $\lambda_{n+1} = \lambda_n + \delta\lambda$ as

$$u(\lambda_{n+1}) \approx u(\lambda_n) + \frac{\partial u}{\partial \lambda}(\lambda_n)\delta\lambda \approx u_n + \frac{u_n - u_{n-1}}{\lambda_n - \lambda_{n-1}}\delta\lambda, \tag{9}$$

which provides *secant continuation*. If $\delta\lambda$ is fixed as we trace the branch, so that $\lambda_n = \lambda_{n-1} + \delta\lambda$ too, this simply reduces to

$$u_{n+1} \approx 2u_n - u_{n-1}.$$

The secant predictor simply requires evaluation based on two past solutions, rather than a solve with a Jacobian, and so is of comparable cost to natural continuation, albeit with the need to store the last two solutions.

While both tangent and secant continuation can improve the initial guess provided to the corrector, they will still fail at fold points as neither will allow λ to trace back on itself. This is fundamentally a problem of parametrisation by requiring that our branch is given by a function $u(\lambda)$, despite knowing that the underlying problem will likely have multiple solutions for some choices of λ. To avoid this we must also solve for λ as we proceed along the branch, which leads to the ideas of arclength and pseudo-arclength continuation.

It is also possible to consider higher-order approximation to further improve the predictor, including fitting higher order polynomials over multiple past solutions that lie on the solution branch $u(\lambda)$, though one should be careful not to induce more work than is saved in the corrector. It may also be preferable to parametrise, instead, with arclength as we now discuss.

2.2 Pseudo-arclength continuation

A more robust way to parametrise the curve of solutions that forms a branch is by the arclength s on the curve, measured from a point, say (u_0, λ_0), on the curve. Specifically, consider a branch as

$$(u(s), \lambda(s)), \tag{10}$$

which allows λ to vary and thus the ability to trace round (multiple) fold points. Since λ is no longer fixed we must solve for λ as well as u in the continuation process. This requires an extra equation and thus yields an augmented system to solve

$$A(u(s), \lambda(s)) := \begin{pmatrix} F(u(s), \lambda(s)) \\ C(u(s), \lambda(s), s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{11}$$

where $C(u, \lambda, s) = 0$ is a constraint that is used as a condition to define λ, typically implicitly. Note that $C(u, \lambda, s) = \lambda - \lambda_n - \delta\lambda$ would return the choice made by natural, tangent, or secant continuation as described in the previous section, where λ is simply incremented by a fixed amount $\delta\lambda$. The augmented system (11) is solved using the corrector, typically Newton's method.

The purpose of this is to allow us to trace along the curve a given distance, irrespective of how λ changes along the curve. Perhaps the most obvious choice is to let C encode this notion by saying we want to find a point $(u(s), \lambda(s))$ on the solution curve a distance $\delta s = s - s_n$ away from the last known solution $(u_n, \lambda_n) = (u(s_n), \lambda(s_n))$ at $s = s_n$. This is given by *arclength continuation* which chooses

$$C(u(s), \lambda(s), s) = \|u(s) - u_n\|^2 + |\lambda(s) - \lambda_n|^2 - (s - s_n)^2. \tag{12}$$

Here, the norm $\|\cdot\|$ for the solution space should be chosen appropriately. However, the main problem with this approach is that there will be two solutions of (11) in the general case, one tracing forwards in s and one tracing backwards in s, so we may accidentally converge to the wrong one. Further, the expression in (12) adds yet more nonlinearity to the system we want to solve which can make it harder for the corrector to be effective.

The more popular and widely used approach, popularised through [41], is to add a linear constraint based on arclength, which is known as *pseudo-arclength* continuation. Intuitively, this uses a predictor on the tangent to the solution curve $(u(s), \lambda(s))$ at a distance δs away from the last known solution and then constrains to look for new solutions only in the space that is perpendicular to the tangent. A simple diagram is given in Figure 3 for 1D, which can be thought of as going along the tangent and after some small distance δs turning a right-angle towards the solution curve and heading straight for that new point of the curve. This allows us to easily trace around fold points provided δs is small enough (depending on the curvature of the solution branch).

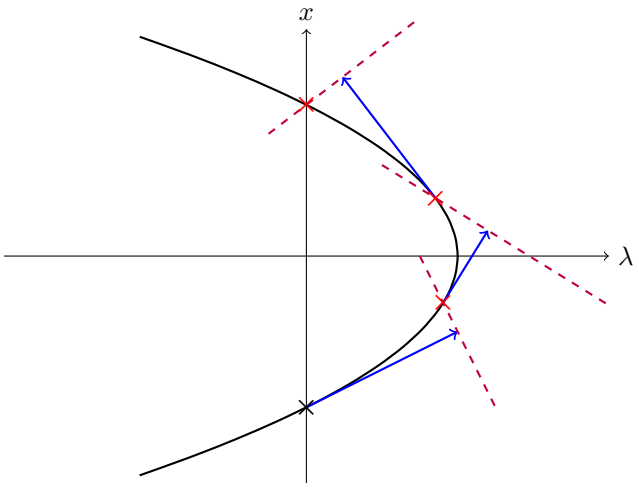

Figure 3: Sketch of applying pseudo-arclength continuation. The solution branch is given in black with a fold point at $x = 0$, tangent vectors are given in blue and the orthogonal hyperplanes are given in dashed purple. The solutions obtained are marked with a red $\times$, with the initial solution marked in black.

Mathematically, the hyperplane orthogonal to the tangent $(\dot{u}_n, \dot{\lambda}_n)$ at the past solution (u_n, λ_n) along a distance δs is given by

$$C(u(s), \lambda(s), s) = \langle u(s) - u_n, \dot{u}_n \rangle + (\lambda(s) - \lambda_n)\dot{\lambda}_n - (s - s_n) = 0. \tag{13}$$

Again, one should take care in using a suitable inner product $\langle \cdot, \cdot \rangle$ so that the first term in (13) is commensurate with the second. When solving PDEs posed on a domain Ω, we stress that the inner product and norm used (here and elsewhere) should be mesh-independent, namely giving the same result for equivalent inputs when represented on different meshes. In a simple scalar problem this might typically correspond to using the $L^2(\Omega)$ inner product and, for example, when considering the discrete formulation on a uniform mesh, where $u \in \mathbb{R}^N$ is a vector of pointwise values in ℓ^2, we should normalise by the mesh spacing h and take

$$\langle u, v \rangle = h^d v^T u,$$

where d is the spatial dimension of the domain Ω. A benefit of pseudo-arclength continuation is that the augmented system (11) is no longer singular at fold points (though some care may be needed when solving at or very close to fold points) and the approach easily allows us to traverse past such points.

Alternate variations for (13) are also possible, such as favouring change in u or λ directions through a weighting parameter $0 < \theta < 1$ and setting

$$C(u(s), \lambda(s), s) = \theta\langle u(s) - u_n, \dot{u}_n \rangle + (1 - \theta)(\lambda(s) - \lambda_n)\dot{\lambda}_n - (s - s_n),$$

which can equally be through of as changing the inner product on the whole space of solution and parameter (u, λ). Further, it is not necessary to compute the exact tangent $(\dot{u}_n, \dot{\lambda}_n)$ and, for example, secant approximation can be made for the predictor akin to (9).

To compute the tangent along the branch (10) at a point (u_n, λ_n) with parameter value s_n, we consider taking the total derivative (with respect to s) of the augmented system (11) with $C(u, \lambda, s)$ defined in (13) to yield

$$\begin{aligned} \frac{\partial F}{\partial u}\frac{\partial u}{\partial s} + \frac{\partial F}{\partial \lambda}\frac{\partial \lambda}{\partial s} &= 0, \\ \left\langle \frac{\partial u}{\partial s}, \dot{u}_n \right\rangle + \dot{\lambda}_n \frac{\partial \lambda}{\partial s} - 1 &= 0. \end{aligned}$$

Evaluating at s_n and noting that, by definition, the tangent vector at this point is

$$\left(\dot{u}_n, \dot{\lambda}_n\right) = \left(\frac{\partial u}{\partial s}(s_n), \frac{\partial \lambda}{\partial s}(s_n)\right),$$

we obtain the following system

$$J_F(u_n, \lambda_n) \begin{pmatrix} \dot{u}_n \\ \dot{\lambda}_n \end{pmatrix} = 0, \tag{14a}$$

$$\|\dot{u}_n\|^2 + |\dot{\lambda}_n|^2 = 1, \tag{14b}$$

where $J_F \colon \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ is the Jacobian of $F(u, \lambda)$, that is the matrix of all first-order partial derivatives (including in λ). We note that (14b) is simply a length normalisation condition on the tangent vector. To solve the system (14) we note that $\dot{\lambda}_n$ is a scalar whose value propagates multiplicatively through to $\dot{u}_0$ in the linear equation (14a), which means we can instead solve for the un-normalised $\hat{u}_n$ via

$$J_f(u_n)\hat{u}_n = -\frac{\partial F}{\partial \lambda}(u_n, \lambda_n), \tag{15}$$

followed by the normalisation condition

$$\|\hat{u}_n \dot{\lambda}_n\|^2 + |\dot{\lambda}_n|^2 = 1 \tag{16}$$

for $\dot{\lambda}_n$ and finally setting $\dot{u}_n = \hat{u}_n \dot{\lambda}_n$. We recall that, in (15), we use the Jacobian of $f(u) = F(u, \lambda_n)$, where λ_n is assumed fixed, which is equivalent to standard tangent solve in (8). Further we can explicitly write the formula for $\dot{\lambda}_n$ from (16) as

$$\dot{\lambda}_n = \frac{\pm 1}{\sqrt{1 + \|\hat{u}_n\|^2}}, \tag{17}$$

where the choice of sign should be made such that the orientation along the branch of solutions is preserved (assumed to be increasing s). One way this can be done is by ensuring the preceding tangent vector, say $(\dot{u}_{n-1}, \dot{\lambda}_{n-1})$, has a positive inner product with the new one, namely that

$$\operatorname{sign} \dot{\lambda}_n = \operatorname{sign}(\langle \dot{u}_{n-1}, \hat{u}_n \rangle + \dot{\lambda}_{n-1}).$$

Since the computation of the tangent carries some non-trivial work, namely the solution of (15), it may be beneficial to use a secant approximation based on the past two points (u_n, λ_n) and (u_{n-1}, λ_{n-1}) to instead yield the constraint

$$C(u(s), \lambda(s), s) = \left\langle u(s) - u_n, \frac{u_n - u_{n-1}}{s_n - s_{n-1}} \right\rangle + (\lambda(s) - \lambda_n)\frac{\lambda_n - \lambda_{n-1}}{s_n - s_{n-1}} - (s - s_n) = 0. \tag{18}$$

One can extend the idea of using previous solutions to produce a predictive model of the solution branch. For instance, suppose we store the past m solutions, then a jth order polynomial in s can be constructed for $j < m$ by a least-squares fit (or polynomial interpolation for $j = m - 1$). This type of approach is taken in [66], which provides further details; adjustments are also described, such as rescaling s and the need to avoid j being too large in case of instability. In general, low-order methods are sufficient when the corrector is not too expensive and tangent or secant predictors are standard in many applications.

2.3 Moore–Penrose continuation

We also note a related approach, called *Moore–Penrose continuation* (see, e.g., [48, 43]), which stems from the use of a Moore–Penrose pseudoinverse for $J_F \colon \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ to solve

$$J_F(u^{(k)}, \lambda^{(k)}) \begin{pmatrix} \delta u^{(k+1)} \\ \delta \lambda^{(k+1)} \end{pmatrix} = -F(u^{(k)}). \tag{19}$$

In this sense, no constraint $C(u, \lambda, s) = 0$ is actively chosen, however, using the pseudoinverse would implicitly apply the constraint that $(\delta u^{(k+1)}, \delta \lambda^{(k+1)})^T$ is (ℓ^2-) orthogonal to any vector in the kernel of J_F, which will be one-dimensional away from bifurcation points. The approach can be motivated by the fact that the solution to (19) is also the solution to minimising the distance from the initial guess (predictor) to the solution branch determined by $F(u, \lambda) = 0$. It can further be interpreted as a variant of pseudo-arclength continuation in which the tangent, and hence hyperplane, varies at each corrector iteration [48].

In practice, the pseudoinverse is too expensive to work with and so the orthogonality constraint is added based on an approximation of the kernel vector. This ultimately adds another augmented system (with the same matrix) owing to the fact that the approximate kernel vector must be computed. Similar considerations to that of pseudo-arclength continuation are required, as will be discussed next. We remark that recent work in [44] has tried to make this approach more robust by adding several control measures and by combining it with deflated continuation to try to ascertain problematic regimes in the continuation steps. Overall, we note that pseudo-arclength continuation appears to be the more popular approach in the literature and available software.

2.4 Solving bordered linear systems

One of the challenges of pseudo-arclength continuation is the need to solve the augmented system (11), with C defined in (13), say. Applying the corrector, typically Newton's method, leads to a *bordered linear system* of the form

$$\mathcal{A}\begin{pmatrix} u \\ \lambda \end{pmatrix} := \begin{pmatrix} J & b \\ c^T & d \end{pmatrix}\begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} v \\ \nu \end{pmatrix}, \tag{20}$$

where $J \in \mathbb{R}^{N\times N}$ is the Jacobian matrix in u (or some approximation thereof) and $b, c \in \mathbb{R}^N$ and $d \in \mathbb{R}$. In the most common case of using Newton's method as the corrector and pseudo-arclength continuation given by (13) with a computed tangent $(\dot{u}_n, \dot{\lambda}_n)$ at the previous solution $(u_n, \lambda_n) = (u(s_n), \lambda(s_n))$, we specifically solve for the Newton updates $\delta u^{(k+1)} = u^{(k+1)} - u^{(k)}$ and $\delta\lambda^{(k+1)} = \lambda^{(k+1)} - \lambda^{(k)}$ via

$$\begin{pmatrix} J_f(u^{(k)}, \lambda^{(k)}) & \partial_\lambda F(u^{(k)}, \lambda^{(k)}) \\ \dot{u}_n^T & \dot{\lambda}_n \end{pmatrix}\begin{pmatrix} \delta u^{(k+1)} \\ \delta\lambda^{(k+1)} \end{pmatrix} = \begin{pmatrix} -F(u^{(k)}, \lambda^{(k)}) \\ -C(u^{(k)}, \lambda^{(k)}, s_n) \end{pmatrix}. \tag{21}$$

Here, as before, J_f denotes the Jacobian of $f(u) = F(u, \lambda^{(k)})$ with the parameter $\lambda^{(k)}$ considered fixed, while we abbreviate $\partial_\lambda F = \frac{\partial F}{\partial \lambda}$. The initial guess for our Newton method here is $u^{(0)} = u_n + \dot{u}_n\delta s$, $\lambda^{(0)} = \lambda_n + \dot{\lambda}_n\delta s$, which is our predictor in the hyperplane given by the constraint (13). It is worth noting that the main block J changes each iteration since it depends on the current iterates (u_n, λ_n).

There are several ways to attack the solution of the bordered linear system (20) at small scale, but this becomes more challenging for large-scale computations typical in PDE applications. The primary method in [41] applies when both J and $\mathcal{A}$ are nonsingular (away from bifurcation points) and is as follows. First solve two $N \times N$ linear systems with J (we assume, as is typical, that we have a solver for J) given by

$$Jy = b, \qquad Jz = v. \tag{22a}$$

We can then form the solution via

$$\lambda = \frac{\nu - c^T z}{d - c^T y}, \qquad u = z - \lambda y. \tag{22b}$$

Note that the denominator in the expression for λ is the Schur complement of J in $\mathcal{A}$, namely $d - c^T J^{-1} b$.

An alternative when $d \neq 0$, which corresponds in (21) to $\dot{\lambda}_n \neq 0$ and so when we are away from a fold point (or potentially another bifurcation point), is to first eliminate λ in (20) as described in [41] to get

$$\lambda = \frac{1}{d}(\nu - c^T u), \tag{23a}$$

and so, by substitution of this expression, the $N \times N$ linear system

$$\left(J - \frac{1}{d}bc^T\right)u = v - \frac{\nu}{d}b. \tag{23b}$$

The system matrix in (23b) is a rank-one perturbation of J and so its inversion can be given through the inversion of J by the Sherman–Morrison formula

$$\left(J - \frac{1}{d}bc^T\right)^{-1} = J^{-1} + \frac{J^{-1}bc^T J^{-1}}{d - c^T J^{-1} b}. \tag{24}$$

Hence, we are required to solve two $N \times N$ linear systems with J, now with right-hand sides b and $v - \frac{\nu}{d}b$. We note that this is very similar to (22), which eliminates u first instead, and again we have the appearance of the Schur complement in the denominator of (24).

Both approaches will lose stability and accuracy when the Schur complement $d - c^T J^{-1} b$ approaches zero, namely precisely when $\mathcal{A}$ is nearly singular. However, the same is true when J becomes nearly singular (close to bifurcation points). Nonetheless, $\mathcal{A}$ will remain nonsingular at fold points where J is singular. If we can detect this situation, then the system (20) can still be solved using the approach in [41, Section 4.11]. Mathematically the assumptions are that the null space of the Jacobian J is of dimension one and that b is not in the range space of J nor c in the range space of J^T. That is, suppose ϕ and ψ are non-trivial solutions of

$$J\phi = 0, \qquad J^T\psi = 0,$$

then we require that

$$\begin{aligned} N(J) &= \operatorname{span}\phi, \\ \psi^T b &\neq 0, \\ c^T\phi &\neq 0. \end{aligned}$$

The approach of [41] is then to solve as

$$\lambda = \frac{\psi^T v}{\psi^T b}, \qquad u = x_p - \frac{c^T x_p}{c^T \phi}\phi + \frac{\nu - d\lambda}{c^T\phi}\phi, \tag{25a}$$

where x_p is any particular solution of

$$Jx_p = v - \lambda b, \tag{25b}$$

which is consistent since $v - \lambda b \in R(J)$ but has infinitely many solutions because J has a non-trivial null space $N(J)$. More details, such as on calculating ϕ and ψ, are given in [41, Section 4.11]. A more careful treatment of the general case where $\mathcal{A}$ has more than one additional row of constraints or J has higher nullity, such as at other bifurcation points, is presented in [40] but note that the methodology requires an LU-factorisation of J with full pivoting; this is typically prohibitive at large scale.

When J is close to being singular, iterative refinement is also possible, however, improvement is only seen in the bordering algorithm (22) when the solver for J is based on factorisation such as an LU- or QR-decomposition [31]. When a more general solver is used, such as an iterative method like preconditioned CG, then a modified block-elimination method given in [31] can be used and performs well provided $\mathcal{A}$ is well conditioned; see also [32]. A slightly more expensive alternative is to use deflated block-elimination [9, 10], having similarities to (25). Another approach is to adapt the Krylov method to ensure iterates satisfy the orthogonality constraint present in the bottom row of (20) exactly [65]. This is done by use of a Householder transformation, and is useful when solving (20) iteratively with a loose tolerance whereupon the constraint may only be enforced very approximately.

2.5 Parameter step size control

One topic not yet discussed is that of selecting an appropriate step size for the parameter continuation, namely $\delta\lambda$ for natural or purely tangent-based methods or δs for pseudo-arclength continuation. A fixed step size is the simplest choice but may miss behaviour of interest or fail to find a solution if chosen too big, yet waste computational resource and slow down the procedure if chosen too small. Ideally then, adaptive step size selection should be employed.

Allgower and Georg discuss the use of adaptive step sizes in [1, Chapter 6]. A common theme for these approaches is to exploit the natural idea of determining a step size based on how well the corrector performed when finding the last solution; if convergence was fast, then we can likely increase the step size, but if convergence was slow, then it may be preferable to reduce the step size for the next continuation step. A simple scenario would be to double the step size if the corrector step was easy and halve it if was difficult. The approach taken in [30] is to base this on the initial contraction rate of the corrector, given by the ratio of the first two steps of the corrector. Upon assuming a suitable asymptotic relationship

(which will depend on the precise corrector used) for small step size, we can obtain an *a posteriori* estimate of the leading order term of the contraction rate. Given a target value for this contraction rate (which will depend on the particulars of the problem and how well you wish to trace the solution curve), this can be used to determine a step size which, based on the observed initial contraction rate from the previous application of the corrector, would yield approximately the desired target and thus informs how to adjust the current step size. Other quantities can also be tracked on assumption that they follow a suitable asymptotic relationship, such as the initial corrector step length or the angle between two consecutive steps. Further details and examples for Newton's method or a chord method can be found in [1, Section 6.1].

Instead of monitoring an additional quantity such as contraction rate, the approach of [14] uses the number of iterations required by the corrector to reach a given tolerance to control the step size, this requires an error model for the corrector iteration. This has the advantage of being more widely applicable as it does not assume asymptotic relationships. Suppose for generality that h is our parameter we wish to control the step size of and denote the predictor (e.g., tangent predictor) as $v_0(h)$. Further, let the corrector steps (e.g., an iteration of Newton's method) be given by application of a map T so that we obtain iterates $v_{i+1}(h) = T(v_i(h))$ where we suppose that the limit $v_\infty(h) = \lim_{i\to\infty} v_i(h)$ exists, for sufficiently small h. We would like to compute a deceleration factor q to update the step parameter as $h \mapsto h/q$ based on a desired number of corrector iterations $\tilde{k}$, the actual number required to find the previous solution k, and the associated iterates $v_i(h)$. We follow the discussion of this in [1, Section 6.2].

To proceed, we make use of an assumption that the corrector essentially operates orthogonally to the predictor and, moreover, that the (modified) error in the solution from terminating the corrector at iteration k, namely

$$\epsilon_k(h) := \gamma\|v_\infty(h) - v_k(h)\|,$$

satisfies an inequality of the form

$$\epsilon_{k+1}(h) \le \psi(\epsilon_k(h)),$$

for some error model function ψ which is monotonic, and where $\gamma > 0$ is a constant independent of h. Two such error models for the Newton corrector are derived in [14] based on Newton–Kantorovich theory, namely

$$\psi(\epsilon) = \frac{\epsilon^2}{3 - 2\epsilon}, \qquad 0 \le \epsilon \le 1,$$
$$\psi(\epsilon) = \frac{\epsilon + \sqrt{10 - \epsilon^2}}{5 - \epsilon^2}\epsilon^2, \qquad 0 \le \epsilon \le 1.$$

Once an appropriate error model ψ is determined, we can then a posteriori evaluate the quotient

$$\omega(h) := \frac{\|v_k(h) - v_{k-1}(h)\|}{\|v_k(h) - v_0(h)\|} \approx \frac{\|v_\infty(h) - v_{k-1}(h)\|}{\|v_\infty(h) - v_0(h)\|} = \frac{\epsilon_{k-1}(h)}{\epsilon_0(h)} \tag{26}$$

and use the estimate $\epsilon_{k-1}(h) \le \psi^{k-1}(\epsilon_0(h))$ to obtain

$$\omega(h) \lessapprox \frac{\psi^{k-1}(\epsilon_0(h))}{\epsilon_0(h)}, \tag{27}$$

from which we can find an estimate for $\epsilon_0(h)$ by solving for equality. We then want to find where the modified error $\epsilon_{\tilde{k}}(h/q)$ after $\tilde{k}$ iterations is small enough to terminate the corrector based on this estimate for $\epsilon_0(h)$. After calculating ω as defined in (26), we do this by determining ϵ such that (27) holds with equality and then $\tilde{\epsilon}$ such that $\psi^{\tilde{k}}(\tilde{\epsilon}) = \psi^k(\epsilon)$ and finally set $q = \sqrt{\epsilon/\tilde{\epsilon}}$. Note that q is typically also restricted so that it can not be too big or small, such as by enforcing that $\frac{1}{2} \le q \le 2$. See [1, Section 6.2] for further justification on this approach, along with examples and modifications.

Other approaches have also been considered in the literature. By noting a connection of step size selection with damping Newton's method, under suitable conditions, [3] suggest damping of the step size until a sufficient decrease criterion is reached. Standard line search techniques can then be used to

determine appropriate damping. The use of higher-order predictors can also be used in tandem with step size choice, as discussed in [1, Section 6.3].

A separate issue occurs when we are close to a bifurcation point where two or more branches intersect in that we may accidentally jump to a different branch in our continuation method. This can also happen if two branches are sufficiently close even if they do not meet, for instance with perturbed (or imperfect) transcritical or pitchfork bifurcations. As a rule, we may at first want to take a long enough step to pass the bifurcation point, rather than taking many small steps attempting to remain on the same branch. We can then aim to locate the bifurcation point afterwards; this will be discussed in the next section.

To try to mitigate branch jumping, [61, Section 3.6.2] suggests using multiple predictors (which can be applied in parallel); this approach is implemented in the MATLAB software `pde2path`. Such multipredictors are given along the tangent but now using p step sizes $i\delta s$, for $i \in 1, \ldots, p$, this allows for long predictors. The idea is heuristic and based on the assumption that the corrector will converge slowly when it is converging to a solution on a different branch, since the corrector will have to change the shape of the approximate solution until it fits with the solution properties on the new branch. Thus the criterion used as to whether to accept or reject the outcome from each predictor is that the residual in the (Newton) corrector has to decrease by at least a factor $\alpha < 1$ at each step (in a suitable norm). The hope is that, while some intermediary predictors near the bifurcation may converge to different branches, a suitably long predictor will pass the bifurcation and converge quickly to the original branch on the other side of the bifurcation point. While there is no guarantees, [61] reports that the idea works remarkably well in practice (within their scope of nonlinear PDE problems); here a factor $\alpha = 10^{-3}$ is typically used but they sometimes require a smaller value of 10^{-6}. Further, if the residuals do not decrease sufficiently, instead of reducing δs, a relaxed condition can be used such that a factor α decrease in the residual norm need only be reached within some fixed number of iterations greater than one. This highlights that no one-size-fits-all approach will unfailingly work and one should be mindful of the particular problem at hand as much as possible.

2.6 Locating bifurcation points

While a continuation method is being employed, it is often of interest to find bifurcation points as we traverse solution branches as these determine where qualitative changes in solution behaviour occur and may represent some physical phenomenon of interest. To detect that a bifurcation has occurred[3], we measure some property as we continue along the branch until it reveals an appropriate change. This usually relates to eigenvalues of the Jacobian J.

A simple test relates to the sign of the determinant of the Jacobian at a point (u, λ), namely

$$\operatorname{sign}\left(\det J(u,\lambda)\right) = \operatorname{sign}\left(\prod_i \mu_i(u,\lambda)\right), \tag{28}$$

where μ_i are the eigenvalues of J, including multiplicities. At a bifurcation point the determinant of the Jacobian is zero, as the Jacobian has a zero eigenvalue. Since for so-called simple bifurcation points a single eigenvalue crosses the origin, we can detect such bifurcations by monitoring (28); this method works when any odd number of eigenvalues cross zero, but cannot detect even numbers. The determinant in (28) can be readily calculated from an LU-decomposition, however, this is only feasible for relatively small problems. In general, one must track eigenvalues with real part close to zero. This can be done at scale by computing a small number of eigenvalues at each continuation step. Note that at fold points an eigenvalue will approach zero but not cross to change sign and so (28) will not detect a fold point, however, this can instead be inferred by tracking whether λ is increasing or decreasing.

Once we have detected that bifurcation has occurred, we may wish to *localise* in order to find the bifurcation point more precisely. A simple technique, given a detection test such as that in (28), is bisection. For instance, if we have two points where the sign of the determinant changes then we can bisect the interval in half to determine a smaller interval which contains the bifurcation point. Since the interval only halves each time, convergence to the bifurcation point is linear and this may be slow if we want high precision.

[3] In general, we may need to exclude certain technicalities. This includes cases where branches intersect tangentially, indeed, in such situations it may not be clear which branch is which.

An alternative to determining the bifurcation point more efficiently is given by noting that a necessary condition for (u, λ) to be a bifurcation point is that the Jacobian $J(u, \lambda)$ is singular. This means that the bifurcation point satisfies the extended (Seydel–Moore–Spence) system

$$F(u, \lambda) = 0, \tag{29a}$$
$$J(u, \lambda)v = 0, \tag{29b}$$
$$\|v\| = 1, \tag{29c}$$

where v is an eigenvector of the Jacobian. The extended system (29) can be solved for u, v and λ using, say, Newton's method. To reduce the nonlinearity from (29c), in practice one can replace this with $\langle c, v\rangle = 1$ for some random unit vector c, which will work with probability one (i.e., so long as c is not orthogonal to v). This approach allows us to directly target the bifurcation point and allows for faster convergence. At fold points the convergence is quadratic, however, the Jacobian associated with the extended system (29) is singular at other bifurcation points and so we can expect slower convergence. While it is possible to construct other extended systems for a variety of bifurcations (see, e.g., [47, Section 3.3]), this is limited in the sense that we need to know what type of bifurcation point we are trying to localise in order to regain faster convergence.

2.7 Branch switching

After finding a bifurcation point where two (or more) branches intersect, often known as a branch point when the intersection is non-tangential, in order to continue along the new branch we must employ branch switching. From the theoretical standpoint this can be determined via Lyapunov–Schmidt reduction; see, e.g., [61, Chapter 2]. Under some assumptions, this states that, if the dimension of the kernel of the Jacobian at the bifurcation point is m, all solution branches of $F = 0$ are related to solutions of an $m \times m$ *algebraic* system (in the infinite-dimensional case it is possible that this algebraic system is not square but for our purposes it always will be).

In the simplest case, at the bifurcation point we have a single real eigenvalue (of no higher multiplicity) of the Jacobian J of $f(u) = F(u, \lambda)$ which crosses through zero. This case is analysed in the Crandall–Rabinowitz theorem which in essence states—under the technical assumptions of both the dimension of the kernel of J and the codimension of the range of J being one, sufficient smoothness of F around the branch point, and a transversality condition—that we do have a branch point and the non-trivial branch (the one we are seeking) is given locally as a continuously differential curve branching off in a direction given, at leading order, by the single kernel vector of J. Furthermore, all solutions of $F(u, \lambda)$ close to the branch point are either on the original trivial branch or the single bifurcating branch. See, e.g., [62, Section 2.2] for a technical and rigorous statement. This gives the theoretical framework for the existence of a bifurcating branch and the associated tangent vector in the simplest case.

While Lyapunov–Schmidt reduction is a powerful theoretical tool, in practice we do not know enough to determine the $m \times m$ algebraic system. Taylor approximation can be used, as exhibited in [61, Section 3.2] (see further in [47, Section 6.7]), but becomes challenging when m is relatively large; see also approaches in [1, Chapter 8] and [41, Section 5.26]. Instead, the expedient approach may simply be to use brute force with many initial guesses to find the desired bifurcating solution branches. One may also try to use deflation techniques to remove known solutions in order to help converge to the bifurcating branches, in this way deflation can be seen as an alternative to traditional branch switching techniques.

The Taylor approximation approach results in a set of algebraic bifurcation equations. At second order we arrive at the so-called quadratic bifurcation equations (QBE) which determine all branches only in the simplest case described above (i.e., giving both branches). To determine branches for (generic) pitchfork bifurcations we end up requiring a third order Taylor expansion and thus a set of cubic bifurcation equations (CBE). There are cases of higher-order indeterminacy which require even more derivatives in the Taylor expansion and this can become challenging and complicated to pursue in general. Nonetheless, it is reported in [61] that the combined use of both QBE and CBE works well in practice, with some occasional fine tuning, for all the PDE problems they consider but note that care should be taken in the case of having continuous symmetries, as we will discuss in Section 5, where preprocessing required.

To see how the Taylor approximation works, following [62, Section 2.3], consider the case where the dimension of kernel of J is m and $\partial_\lambda F$ is in the range of J at the bifurcation point, namely $\dim N(J) = m$ and $\partial_\lambda F \in R(J)$. This means that we have linearly independent spanning sets for the kernel of J and

its adjoint, given by $N(J) = \text{span}\{\phi_1, \ldots, \phi_m\}$ and $N(J^T) = \text{span}\{\psi_1, \ldots, \psi_m\}$ with $\langle \phi_i, \psi_j \rangle = \delta_{i,j}$. Furthermore this gives the existence of a unique $\phi_0 \in N(J)^\perp$ such that $J\phi_0 + \partial_\lambda F = 0$ with $\langle \phi_0, \psi_j \rangle = 0$. We now suppose that we have a smooth solution branch parametrised by s, namely $(u(s), \lambda(s))$, with a bifurcation point at (u_0, λ_0) where $s = s_0$. Since we have $F(u(s), \lambda(s)) = 0$ along the branch we can differentiate in s to obtain, by the chain rule,

$$J\dot{u}(s_0) + \partial_\lambda F \dot{\lambda}(s_0) = 0, \tag{30}$$

where $\dot{u}$ and $\dot{\lambda}$ denote derivates in s. Thus $\dot{u}(s_0)$ can be expressed as an expansion in the vectors ϕ_j for $0 \leq j \leq m$, say

$$\dot{u}(s_0) = \sum_{j=0}^{m} \alpha_j \phi_j, \tag{31}$$

with $\alpha_0 = \dot{\lambda}(s_0)$ and $\alpha_j = \langle \phi_j, \dot{u}(s_0) \rangle$ for $1 \leq j \leq m$. To derive equations that prescribe the coefficients α_j we differentiate (30) again in s to obtain

$$J\ddot{u} + \partial_\lambda F \ddot{\lambda} = -\left(\partial_u J[\dot{u}, \dot{u}] + 2\partial_\lambda J \dot{u}\dot{\lambda} + \partial_{\lambda\lambda} F \dot{\lambda}^2 \right), \tag{32}$$

valid at $s = s_0$, where we use the notation $\partial_u J[\dot{u}, \dot{u}]$ from [62] to denote the tensor contraction

$$\left(\partial_u J[\dot{u}, \dot{u}]\right)_i = \sum_{j,k} \frac{\partial^2 F_i}{\partial u_j \partial u_k} \frac{\partial u_k}{\partial s} \frac{\partial u_j}{\partial s}.$$

Now since the left-hand side of (32) is in the range space $R(J)$, then so must be the right-hand side. Thus by taking inner products with the ψ_i for $1 \leq i \leq m$ we obtain a system of m equations in $m+1$ unknowns but which is homogeneous and allows us to solve for the coefficients of the tangent vector in (31) up to a multiplicative constant. These are the quadratic bifurcations equations (QBE) $B(\alpha_0, \boldsymbol{\alpha}) = \mathbf{0} \in \mathbb{R}^m$, with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$, where

$$\begin{aligned}
B_i(\alpha_0, \boldsymbol{\alpha}) &:= \sum_{j=1}^{m} \sum_{k=1}^{m} a_{ijk} \alpha_j \alpha_k + 2 \sum_{j=1}^{m} b_{ij} \alpha_j \alpha_0 + c_i \alpha_0^2, \qquad 1 \leq i \leq m, \\
a_{ijk} &:= \langle \psi_i, \partial_u J[\phi_j, \phi_k] \rangle, \\
b_{ij} &:= \langle \psi_i, \partial_u J[\phi_j, \phi_0] + \partial_\lambda J \phi_j \rangle, \\
c_i &:= \langle \psi_i, \partial_u J[\phi_0, \phi_0] + \partial_\lambda J \phi_0 + \partial_{\lambda\lambda} F \rangle.
\end{aligned}$$

Note that this simplifies quite substantially in the simplest case of $m = 1$ to a single quadratic in α_0 and α_1. Similar ideas apply to derive the cubic bifurcation equations (CBE) by taking another derivative in s, with some simplification to (32) on the assumption that the QBE is insufficient, but the formulation becomes lengthy and more complex. They can be found in, for instance, [61, Section 3.2] which also gives details on approximating (by finite differences) the higher-order derivatives of the Jacobian that appear in both quadratic and cubic bifurcation equations.

Clearly there is some non-trivial work involved in branch switching and the more prior information about the bifurcation at hand, the better tools we have to determine new branches. Nonetheless, since we must ultimately solve an $m \times m$ system, where m is the typically small dimension of the kernel of the Jacobian, these approaches are able to scale to relatively large problem sizes through approximation of the higher-order derivatives of the Jacobian, since we usually do not need the tangent vectors along new branches with high accuracy. An alternative that we touched upon earlier is to use deflation as a tool not only to find solutions but to naturally aid in branch switching, this alleviates some of the complexity of the algebraic bifurcation equations. We now turn to this alternative perspective.

3 Continuation of disconnected solution branches with deflation

The discussions in previous sections have focused on techniques applicable to continuing connected branches of solutions. It may well be the case that some branches in the bifurcation diagram are not

connected to the initial known solution and, as such, will remain unobtainable. This is one of the main motivations behind the use of deflation, a technique which does not require connectedness of solution branches to find solutions. While deflation is not a new approach, see [8], it was considered to be somewhat unreliable and lack robustness until recent redevelopment [25, 26, 11] where it has proved very successful. We remark that there is no guarantee to find *all* solutions but in practice we can find many more solutions this way, which makes it, in the words of [61], a "robust partial success".

The deflation approach is simple in essence, for a fixed λ we use a suitable initial guess to find a solution (e.g., by Newton's method) and then remove this solution by *deflation* so that we will not find it again. We then search anew from the same initial guess in the hope of finding a new solution (on a different branch) with any such new solution again being deflated away until we cannot find further solutions. We can then either use an alternative initial guess, if available, or terminate and increment λ to repeat the process. Note, each solution at the previous value of λ can now be used to compute an initial guess, for instance via a tangent predictor.

Suppose we fixed λ so that we wish to solve $f(u) := F(u, \lambda) = 0$ and that we already have a known solution u^*. The key properties required by a *deflation operator* $D(u; u^*)$, applied so that we solve

$$g(u) := D(u; u^*)f(u) = 0, \tag{33}$$

are that

$$f(u) = 0 \iff g(u) = 0 \quad \text{for } u \neq u^*, \tag{34a}$$

and

$$\liminf_{u \to u^*} \|g(u)\| > 0. \tag{34b}$$

These conditions are summarised as preserving all solutions of the original problem aside from u^*, (34a), and removing the known solution u^* as a solution to $g(u) = 0$, (34b). When more than one known solutions are required to be deflated away, say $u_1^*, u_2^*, \ldots, u_l^*$, we can simply concatenate the deflation operators to utilise

$$D(u; u_1^*, u_2^*, \ldots, u_l^*) := D(u_1^*)D(u_2^*)\ldots D(u_l^*),$$

or potentially combine all solutions into the form of the deflation operator (see a later example in (37)).

The deflation operator proposed in [25, 26] is given by

$$D(u; u^*) = \left(\frac{1}{\|u - u^*\|^p} + \sigma \right) I, \tag{35}$$

for some power $p \in \mathbb{N}$, shift $\sigma \in \mathbb{R}$, and where I is the identity. Note that this is effectively a scalar function which multiplies f and we subsequently treat it as such. In (35) an appropriate norm should be used; for PDE problems this may be a discretised version of the $L^2(\Omega)$-norm or other problem-relevant norm on Ω. The addition of a non-zero shift σ (typically $\sigma = 1$) ensures we have the correct limits at both u^* and ∞, namely

$$\begin{aligned} D(u; u^*) &\to \infty && \text{as } u \to u^*, \\ D(u; u^*) &\to \sigma \neq 0 && \text{as } u \to \infty. \end{aligned}$$

The latter is important to ensure the function f is not destroyed by sending it to zero away from u^*. For (34b) to hold, we require that p is large enough to counteract the multiplicity of the solution u^*; since this is typically not known in advance, we can keep applying deflation for a fixed p until we have removed the solution up to and including all its multiplicities. In practice, numerical experiments in [26] always use $p = 2$ and $\sigma = 1$.

A desirable property of the deflation approach is that we only require the solution of Newton-like systems (and not extended systems) so if a good approach to solve each step of the Newton method for $f(u) = 0$ is available (e.g., via a preconditioned iterative method for solving with the Jacobian J_f) then this automatically translates into a good approach to solve the deflated systems of the form

$$J_g(u)\delta u_g = -g(u). \tag{36}$$

To see this, consider the Jacobian $J_g(u)$ given by the product rule

$$J_g(u) = D(u; u^*)J_f(u) + g(u)J_D^T(u).$$

The first term is simply a scalar multiple of $J_f(u)$ but the second term is a generally dense addition that may seem problematic, however, this second term represents a rank-one perturbation (we will have $g(u) \in \mathbb{R}^{N\times 1}$ and $J_D^T(u) \in \mathbb{R}^{1\times N}$) and so we can apply the Sherman–Morrison formula, as in (24), to only require to solve systems involving $J_f(u)$. In fact, if one follows through the algebra, we only require a single solve with $J_f(u)$ rather than the usual two and the algorithm to solve (36) can be written as follows:

1. Solve the Newton system $J_f(u)\delta u_f = -f(u)$;
2. Evaluate the scalar $\nu = J_D^T(u)\delta u_f$;
3. Evaluate the scalar
$$\tau = 1 + \frac{D^{-1}\nu}{1 - D^{-1}\nu};$$
4. The solution is computed as $\delta u_g = \tau\delta u_f$.

A strength of the deflation approach to continuation is its simplicity. It is not required to use pseudo-arclength continuation (though this can be used in complement to deflation), tangent prediction is readily available and we are not so worried about jumping branch, all we must do is repeatedly solve Newton systems with the Jacobian of the original function f. This will typically piece together a bifurcation diagram and localisation of bifurcations can be performed afterwards, rather than as a requirement to find new branches. It is an open question as to how best utilise both the more classical continuation approaches in Section 2 and deflation approaches in an optimally complementary way.

We remark that other choices for the deflation operator, of a similar style, can also be used. In particular, [61, Section 3.6.3] proposes a modification when many solutions are already known (for a fixed λ) so that we use

$$D(u; u_1^*, u_2^*, \ldots, u_l^*) := \left(\alpha_1 + \prod_{j=1}^{l} \min(\|u - u_j^*\|_p^p, \alpha_2)\right)^{-1}, \tag{37}$$

with default values $\alpha_1 = 1$ and $\alpha_2 = 1$. Here the use of α_2 in the minimum is due to the fact that the product may otherwise be large even when u is close to some u_j^* as it may well be far from other solutions u_i^*. This choice of deflation operator loses differentiability on null sets around each known solution, depending on α_2, but this appears not to be an issue in practice. In general, there is a balance between wanting to ensure we deflate away known solutions effectively, so that Newton's method does not try to converge to them at any point, and keeping the rest of the function g as nice as possible to enable the best chance of converging to a new solution. This can also be problem dependent, as is the choice for p and σ in (35); an exploration of this is found in [25].

Deflation can also be applied to coupled systems, an example of which is given in [7] for Rayleigh–Bénard convection with no-slip boundary conditions in two dimensions. The steady-state problem for fluid velocity $\mathbf{u}$, pressure p, and temperature T is given in nondimensionalised form as

$$-\mathbf{u}\cdot\nabla\mathbf{u} - \nabla p + \operatorname{Pr}\Delta\mathbf{u} + \operatorname{Pr}\operatorname{Ra}T\hat{\mathbf{z}} = \mathbf{0}, \tag{38a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{38b}$$

$$-\mathbf{u}\cdot\nabla T + \Delta T = 0, \tag{38c}$$

where Pr is the Prandtl number, Ra is the Rayleigh number, and $\hat{\mathbf{z}}$ is the buoyancy direction. Here the Oberbeck–Boussinesq approximation has been made; see [7] for further details on the derivation and simplifications made to obtain (38). The continuation parameter used is Ra and the equations in (38) are collected together to form the nonlinear problem $F(\phi, \operatorname{Ra}) = 0$, where $\phi = (\mathbf{u}, p, T)$. In this case, the deflation operator takes into account $\mathbf{u}$ and T and is given by

$$D((\mathbf{u}, p, T); (\mathbf{u}^*, p^*, T^*)) = \frac{1}{\|\mathbf{u} - \mathbf{u}^*\|_2^2 + \|\nabla(\mathbf{u} - \mathbf{u}^*)\|_2^2 + \|T - T^*\|_2^2} + 1, \tag{39}$$

for known solution $\phi^* = (\mathbf{u}^*, p^*, T^*)$. Note that the pressure p is not present in (39) so as to deflate all solutions which are trivially related to the known solution by the addition of a constant to the pressure. Similar considerations should be made for symmetries in the underlying problem, as detailed in Section 5.1, and considered in examples from the physics of Bose–Einstein condensates in [11, 12, 6]. Deflation has also been considered for computing equilibrium states within cholesteric liquid crystals [22].

4 Dynamical systems

So far we have only considered solutions of (1) in the stationary sense, that is without depending on time. In practice, many problems of interest are time-dependent and steady solutions may bifurcate into time-dependent (or even turbulent) solutions, adding additional complexity into consideration of continuation methods. A typical starting point is an ordinary differential equation (ODE)

$$\frac{\mathrm{d}u}{\mathrm{d}t} = F(u, \lambda), \tag{40}$$

for a vector field F. In this context, in the previous sections we have been considering steady solutions given by (1), which are fixed points of (40) in the sense that solutions do not change over time. A key feature of (40) is that it is *autonomous*, namely F does not depend explicitly on time t (only through the solution u). This means that, from an initial state u_0 at time $t = 0$, the solution in time $u(t; u_0)$ satisfies

$$u(t + \tau; u_0) = u(t, u(\tau; u_0)), \tag{41}$$

which is the key defining property of a *dynamical system*.

With the introduction of time, we also need to consider the concept of stability of fixed points, i.e., solutions of (1). A fixed point is stable if nearby solutions always stay nearby for all forward times, otherwise it is unstable; this is related to eigenvalues of the Jacobian and is used to categorise various bifurcations, such as pitchforks, as subcritical or supercritical depending on the stability properties of the branches. If all nearby solutions tend to the fixed point, then it is asymptotically stable and the fixed point is an attractor. As well as fixed points (steady states), other types of attractors may be encountered in the time-dependent case, such as periodic orbits, invariant tori, and other more exotic structures in higher dimensions. These objects themselves can bifurcate as problem parameters vary and can give rise to other bifurcation behaviour, such as period doubling of periodic orbits. We will discuss some aspects of periodic orbits below.

Many real-world problems are represented not by ODEs such as (40), but by PDEs of the form

$$\frac{\partial u}{\partial t} = F(u, \lambda), \tag{42}$$

where F may now contain spatial differential operators; again we assume such a system is autonomous. Numerical continuation and bifurcation methods for (42) are arguably the same as those for (40) in that we typically discretise first in space to obtain a semidiscrete system of the form (40). The key difference is that now the system (40) is much larger due to the spatial discretisation and so $F\colon \mathbb{R}^{N+1} \to \mathbb{R}^N$ where N is large, in particular such that direct solvers for the Jacobian (such as via an LU-decomposition) are prohibitive. This relates to discussions in Section 2.4 and it should be kept in mind that additional implementation issues arise due to the large size of the dynamical systems when solving PDE problems.

One way to find steady state solutions of (40) is to use some kind of time-stepping technique in the hope of converging to a solution for large time t. This will only be able to find stable solutions but may avoid difficulty in the linear algebra for the types of large systems arising from PDEs. It is further possible to reformulate finding steady states within the wider setting of fixed points of the map

$$v \mapsto u(T; v), \tag{43}$$

where $u(T; v)$ is the solution of time-stepping (40) to time T from initial state v. We can then look for fixed points of the nonlinear map (43) (e.g., by applying Newton's method); see [19, Section 3.1] and also [56]. Fixed points will include steady states but also T-periodic orbits, should any exist, and so it must be checked whether solutions found are true steady states. There is a balance between making T large, which tends to make the nonlinear solve easier, and making T small to reduce the work from

the time-stepper. Hence, T should be chosen to optimise the total computation time, which usually requires some numerical experimenting to determine a good value. Similar ideas can be applied to other attractors, such as periodic orbits and invariant tori, as described in [19, Section 3.1]. We will discuss shortly this further in relation to periodic orbits.

We now consider standard bifurcations that arise from steady states of (40) where λ is a scalar. Each has a *normal form* in which the bifurcation is exhibited in its most fundamental formulation. The first is the so-called *saddle-node bifurcation*, also known as a *fold* in the terminology of previous sections, which has the normal form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda \pm u^2.$$

In the supercritical case, given by the negative sign, the branch given by $u = \sqrt{\lambda}$ is stable while $u = -\sqrt{\lambda}$ is unstable. This is reversed in the subcritical case, given by the positive sign, where the branch $u = \sqrt{-\lambda}$ is unstable and $u = -\sqrt{-\lambda}$ is stable.

A *transcritical bifurcation* corresponds to two branches intersecting at an angle, in the normal form the bifurcation occurs at the intersection of two straight lines as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u \pm u^2.$$

In both the supercritical case (negative sign) and subcritical case (positive sign) the fixed points given by the two branches change stability at the bifurcation point. A *pitchfork bifurcation* takes the form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u \pm u^3.$$

Here in the supercritical case the trivial branch $u = 0$ is stable for $\lambda < 0$ and stability is exchanged with the conjugate solutions $u = \pm\sqrt{\lambda}$ upon bifurcation at $\lambda = 0$, with the trivial branch then becoming unstable for $\lambda > 0$. However, in the subcritical case the solutions $u = \pm\sqrt{-\lambda}$ are unstable, restricting convergence to the stable trivial branch for $\lambda < 0$, which ultimately undergoes a so-called catastrophic loss of stability upon bifurcation, with all solutions (here just the trivial branch) becoming unstable for $\lambda > 0$.

Transcritical and pitchfork bifurcations are both structurally unstable as small perturbations destroy their structure (as in Figure 2 in Section 1), and this means they are non-generic and typically arise due to symmetry of the underlying system. In fact, two parameters are required to obtain all local behaviour (e.g., δ in Figure 2 as well as λ) and so such bifurcations are said to be of *co-dimension* two; see [21]. In contrast, saddle-node (fold) bifurcations are of co-dimension one, as is the final bifurcation of steady states that we now consider.

A *Hopf bifurcation* relates to the birth (or death) of a periodic orbit from a steady state. Note that this is therefore a dynamic bifurcation. The normal form requires two components and is typically given in polar coordinates (r, θ) as

$$\begin{aligned}\frac{\mathrm{d}r}{\mathrm{d}t} &= \lambda r \pm r^3,\\ \frac{\mathrm{d}\theta}{\mathrm{d}t} &= \omega,\end{aligned}$$

for $\omega \neq 0$. This has some resemblance to a pitchfork bifurcation, as can be seen in the r-equation, but instead of a conjugate pair of solutions, we instead have a periodic orbit interacting with a fixed point. In the supercritical case, the trivial solution $r = 0$ again transfers its stability, this time to a periodic orbit at $r = \sqrt{\lambda}$, as λ increases through the bifurcation point at $\lambda = 0$. In fact, the periodic orbit attracts nearby solutions at an exponential rate in time. The generic Hopf bifurcation occurs when two complex conjugate eigenvalues (of the Jacobian) cross the imaginary axis. In the subcritical case, an unstable periodic orbit is present along with the stable trivial branch for $\lambda < 0$, in this case solutions inside the periodic orbit ($r < \sqrt{-\lambda}$) are attracted by the trivial fixed point while outside of the periodic orbit they diverge.

4.1 Periodic orbits

To address stability and potential bifurcation of periodic orbits we can consider Poincaré maps. The idea is to introduce a hyperplane Σ, called a Poincaré section, which intersects the periodic orbit in a transverse manner. This should be constructed so that Σ only intersects the orbit once (not twice, i.e., not "on the way back round" too), for instance corresponding to $r > 0$ and $\theta = 0$ in the normal form of the Hopf bifurcation above. Note that transversality requires that the intersection point u^* of the periodic orbit, say $\mathring{u}(t; u_0)$, and the Poincaré section Σ should be linearly independent from $F(u^*, \lambda)$. More details on parametrising Σ can be found in [54]. We can then define the *Poincaré map* $\Pi\colon \Sigma \to \Sigma$ to be the first intersection of the forward orbit of an initial state $u_0 \in \Sigma$ to (40) with Σ. That is, for $u_0 \in \Sigma$ we have $\Pi(u_0) = u(T_1, u_0)$ where $T_1 > 0$ is the first time the solution $u(t; u_0)$ returns to Σ. A sketch is given in Figure 4. The stability of a periodic orbit $\mathring{u}(t; u^*)$, where $u^* \in \Sigma$ is its intersection with Σ, is determined by the associated stability of the fixed point u^* of the Poincaré map Π.

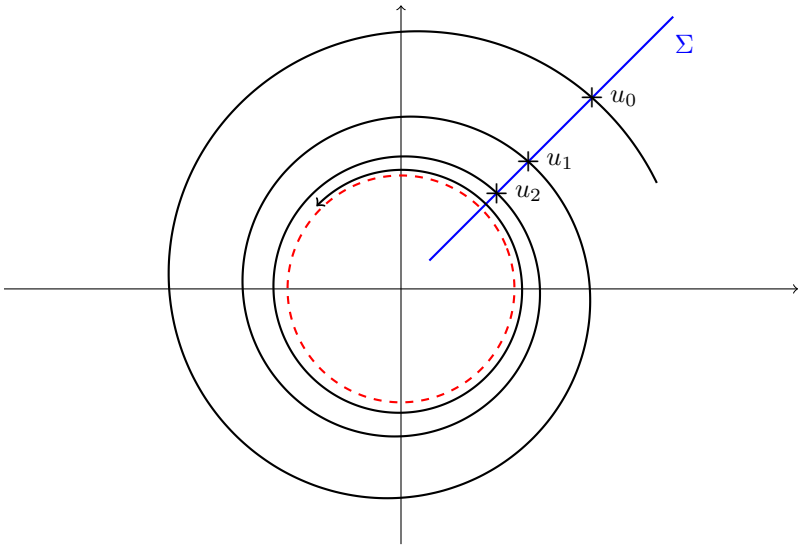

Figure 4: Sketch of a Poincaré map in action with associated Poincaré section Σ in blue. The initial state is u_0 with its orbit $u(t; u_0)$ given by the black spiral, which tends to the periodic orbit $\mathring{u}(t; u^*)$ in dashed red. The Poincaré map Π is such that $\Pi(u_0) = u_1$, $\Pi(u_1) = u_2$, and so on.

Stability depends on the eigenvalues of the linearisation of Π around u^*, in a similar way that stability of fixed points depends on the eigenvalues of the Jacobian. In practice, this relates to a (non-unique) *monodromy matrix* M; see [55, Chapter 7] for much more detail. One way to construct a monodromy matrix is by considering the (in general nonautonomous) matrix valued ODE

$$\frac{\mathrm{d}\Phi}{\mathrm{d}t} = J_f(u^*)\Phi, \qquad \Phi(0) = I,$$

where I is the identity matrix, and defining $M = \Phi(T_1)$. As before, J_f is the Jacobian of $f(u) := F(u, \lambda)$. While the monodromy matrix depends on the phase of the periodic orbit, namely given by the position of Σ, the eigenvalues do not. These eigenvalues of M are called *Floquet multipliers* and we always have a unit multiplier $\gamma = 1$ which relates to the translation invariance of the periodic orbit, that is $t \mapsto \mathring{u}(t + \tau)$ is the same periodic orbit for any τ. The remaining multipliers determine stability, with (asymptotic) stability only when all but the trivial multiplier satisfy $|\gamma| < 1$. If any multiplier has $|\gamma| > 1$, then the periodic orbit is unstable.

We emphasise that Π is a map, namely being discrete rather than continuous like (40). Such discrete maps also present a rich source of bifurcations which, in this case, correspond to bifurcations of the periodic orbits (rather than solely steady states). Bifurcations now occur when non-trivial multipliers cross the unit circle in the complex plane. In one dimensional maps we can only have the non-trivial multiplier cross through $\gamma = 1$ or $\gamma = -1$. The former corresponds to the standard bifurcations previously discussed for steady states, namely saddle-node (fold), transcritical and pitchfork bifurcations (now of

periodic orbits). The case $\gamma = -1$ gives rise to a so-called *period-doubling bifurcation*, where the periodic orbit will double its period (i.e., return time T_1 of the Poincaré map), or equivalently halve its period, depending on whether the parameter λ is increasing or decreasing. This bifurcation in maps is of co-dimension one and we note that it can lead to the onset of chaotic dynamics [16].

In two or more dimensions we can have a pair of complex conjugate eigenvalues cross the unit circle at $e^{\pm i\theta}$ for θ not a multiple of π. This is a *(Poincaré–Andronov–)Hopf bifurcation*. If $\theta/2\pi$ is rational, say p/q, then this results in a periodic orbit with a potentially much larger period related to the denominator q. Otherwise, the orbit becomes dense on an invariant two-dimensional torus, that is if we start on the torus, we remain on the torus and eventually the orbit cover all parts of the torus. Note that, even in the rational case, the orbit will lie on a torus but in this case, due to extra periodicity, will not be dense. Due to such structure this bifurcation is also called a *torus bifurcation*. More generally the bifurcation in the discrete time map is called a *Naimark–Sacker bifurcation*.

More on periodic orbits, calculating the monodromy matrix, and bifurcation behaviour can be found in [55, Chapter 7]. At this point we remark that, in sufficiently high dimensions, other invariant manifolds, such as invariant tori, can bifurcate and that there is a whole zoo of increasingly complex bifurcation behaviour to explore if one is willing to go looking. We briefly consider the concept of global bifurcations in complement to aforementioned examples of local bifurcations.

4.2 Global bifurcations

Other types of bifurcations for periodic orbits to those discussed above also exist, including a so-called blue-sky catastrophe, which relates to infinite period and infinite length periodic orbits [46], thus global behaviour. In general, bifurcation not only refers to local changes in behaviour, as we have so far considered, but also changes in the global behaviour of solutions, which manifests as *global bifurcations*. The simplest example is that of a *homoclinic bifurcation* where a periodic orbit merges with a saddle point. As we approach the bifurcation point the period of the periodic orbit tends to infinity and at bifurcation becomes a homoclinic orbit (starting and ending at the saddle point). After bifurcation the periodic orbit ceases to exist.

In general, global bifurcations are more difficult to find and analyse, as well as to compute, by the nature of being related to global phenomena. Typically a good understanding of the problem at hand is required to investigate the behaviour. We refer to [42] and references therein for a variety of topics related to applied bifurcation theory. We will not consider further issues related to global bifurcation.

5 Symmetries

In this section, we comment on the special roles that symmetry can play and how this may affect branching and bifurcation. We have seen the presence of symmetry, in particular in pitchfork bifurcations (see Figure 1 in Section 1). In the simplest one-dimensional case for $F(x, \lambda) = 0$, the symmetry stems from the transformation $x \mapsto -x$ with $F(-x, \lambda) = -F(x, \lambda)$, which is present in the normal form of the pitchfork $F(x, \lambda) = \lambda x \pm x^3$. This symmetry can be broken by a low order perturbation term, namely for $F(x, \lambda) = \lambda x - x^3 + \delta$ with $\delta > 0$ small (see Figure 2 in Section 1) and is a consequence of the fact that pitchfork bifurcations are of co-dimension two. We note that in numerical computation the symmetry might also be broken by the discretisation if it is nonsymmetric or on a non-uniform mesh. This has consequences for continuation methods designed for connected solutions branches as it may disconnect the branches that otherwise would meet at a pitchfork. If we are able to enforce and maintain symmetry, then connectedness is retained and the pitchfork bifurcation effectively reduces to be of co-dimension one. Similar principles apply in higher dimensions and to other bifurcations that relate to symmetry.

Equivariant branching is the name given to the branching phenomena that occur when a higher-dimensional kernel arises from symmetries of the underlying problem. The natural language to express more complex symmetries is via a symmetry group and the ideas here lead to the key result of the *equivariant branching lemma*, which gives the existence of a branch related by symmetry. The machinery required becomes somewhat technical, involving compact Lie groups, so we refrain from a description here; see, e.g., [61, Section 2.5]. We note that an equivariant Hopf bifurcation theorem also exists [39, Chapter 4]. In essence, symmetries are encoded by the fact that $F(\gamma u, \lambda) = \gamma F(u, \lambda)$ for all γ in a symmetry group Γ, with γu and γF being appropriate actions of the group. For homogenous PDE

problems this can stem from symmetry of the domain, such as through rotation or reflection. For instance, the symmetries for a square domain are given by the dihedral group D_4, stemming from the generators of rotation by 90° and reflection in one axis. See [39, Section 4.3] for an example on the square and [61, Section 2.5] for an example on an equilateral triangle; see also [61, Section 6.5].

Discrete symmetries can lead to a higher multiplicity of branches intersecting at a bifurcation point, which can cause trouble with branch switching and finding all branches that emanate in practice. However, it is also possible to have a continuous symmetry, described by a Lie group of operators that have a (continuous) manifold structure; we follow the discussion in [61, Section 3.5]. Now it becomes possible that, instead of solution branches as (one-dimensional) curves, they may form higher dimensional objects such as a sheet of solutions. Here, non-trivial solutions of $F(u, \lambda)$ can come in continuous families related by the *group orbit*, namely $O_\Gamma(u) = \{\gamma u \,|\, \gamma \in \Gamma\}$. This degenerate situation results in at least one zero eigenvalue of the Jacobian. Since solution families are linked by the continuous symmetry, the objects of interest here should be the group orbits rather than individual solutions; that is we should remove the continuous symmetry from the problem we are trying to solve.

One way to remove the continuous symmetry is to constrain the problem so that, given a solution u^*, we require any new solutions to be transverse to the group orbit of u^*. Given a suitable inner product $\langle \cdot, \cdot \rangle$, this can be achieved through orthogonality, namely a phase condition

$$\langle \partial_\gamma u^*, u - u^* \rangle = 0,$$

where ∂_γ stems from the group action which in turn is dictated by the generators of the group. A simple example comes from the use of periodic boundary conditions in x for a homogeneous problem, in which case we have $\partial_\gamma = \partial_x$. Note that this can also arise in polar, cylindrical, or spherical co-ordinates where periodicity in an angle θ is to be expected. We further remark that while spatial discretisation may break the symmetry this will only be weakly and lead to ill-conditioning and instability of the continuation methodology, which should be avoided. Implementation of a phase condition can be done through use of a Lagrange multiplier η, which is an extra parameter to solve for, and the modified system

$$F(u, \lambda) + \eta \partial_\gamma u = 0.$$

More details on treating continuous symmetries can be found in [4] and examples in `pde2path` are discussed in [61, Section 6.8]. Our primary purpose here is to emphasise that care must be taken when the problem has symmetry, which may also be a *hidden symmetry* which is not immediately evident from the structure, for example, of the PDE or domain.

5.1 Treating symmetries in deflated continuation

When using deflated continuation, instead of deflating away the known solution u^*, in the presence of a continuous symmetry group we must deflate away the whole group orbit $O_\Gamma(u^*)$. To do this the deflation operator should be constructed so that it is invariant to the action of the underlying Lie group. A rich example of this approach is given in [11, 12, 6] by considering the (complex-valued) Gross–Pitaevskii equation, here described in 3D as

$$-\frac{1}{2}\Delta u + \frac{x^2 + y^2 + z^2}{2} u - \lambda u + |u|^2 u = 0,$$

on a domain Ω with suitable boundary conditions (for instance, homogeneous Dirichlet conditions so long as the domain is suitably large). This is a nonlinear Schrödinger equation with a parabolic potential and relates to atomic Bose–Einstein condensates. The PDE here has the continuous symmetry group $SO(2)$ (the special orthogonal group on the plane) corresponding to complex phase shifts

$$u(\mathbf{x}) \mapsto e^{i\theta} u(\mathbf{x}), \quad \text{for } \theta \in \mathbb{R}.$$

In this case, given a solution u^*, we can build a deflation operator invariant to this symmetry as

$$D(u; u^*) = \| |u| - |u^*| \|^{-2} + 1,$$

for a suitable norm (e.g., the H^1-norm in [11]) and where $|u|$ is the amplitude of the complex-valued wavefunction u.

A further symmetry group also exists, namely $SO(3)$ corresponding to fixed-body rotations $R \in \mathbb{R}^{3\times 3}$ in space, that is

$$u(\mathbf{x}) \mapsto u(R\mathbf{x}), \quad \text{for } R^{-1} = R^T \text{ with } \det R = 1.$$

This time an appropriate deflation operator can be given by

$$D(u; u^*) = \| |\hat{u}| - |\hat{u}^*| \|^{-2} + 1,$$

where, in spherical coordinates, $\hat{u}(r, \theta, \psi)$ is the average of u over a shell of radius r; see [12] for a more concrete example in 2D. The practicality of the approach for more complex symmetry groups remains open as, while there are good ideas on what exactly must be computed, the computation itself becomes challenging and difficult to parallelise due to non-local behaviour.

Finally, we reiterate that even with symmetry considerations the deflated continuation approach will not always find all solution branches and, in general, the quality of the initial guess(es) for Newton's method are of key importance; in [11] an example of this is shown and additional knowledge was used to supply initial guesses that revealed new solutions branches that deflated continuation alone missed. Indeed, we highlight one of the conclusions from this work which states that "deflated continuation should not be thought of as a universal method that blindly reveals all solutions to a particular bifurcation problem, but as a useful tool that becomes even more powerful when combined with physical insight". Nonetheless, deflation has proved itself to be an essential part of solving challenging PDE continuation problems when bifurcation results in many solutions branches.

6 Available software

There is a good variety of high quality, robust and well-maintained software packages that are available for numerical continuation. We recommend that, where appropriate, the NEPTUNE project should utilise these instead of re-implementing well known algorithms. We summarise some of the more popular packages in Table 1. A more complete list of software for dynamical systems can be found on the SIAM Dynamical Systems website[4], while a selection of those capable of continuation and bifurcation analysis are highlighted on the `BifurcationKit.jl` website[5].

In Table 1, we have included information about when the software was last updated, its primary language, and what are the terms of its license. As it is important for NEPTUNE that the software is suitable for large-scale problems, and supports deflation, we have identified these packages in the table. We also include details about the main algorithms implemented by the software and any package-specific notes.

Of these packages, we have identified `pde2path`, `BifurcationKit.jl` and `defcon`, which all support large-scale problems and deflation, as the best candidates in the context of NEPTUNE. The `Trilinos` package `LOCA` does not support deflation at the moment, but is otherwise fully featured and well developed, and may also be worth exploring. We note that `LOCA` (along with its companion package `NOX` for the nonlinear systems) aims to be generic and general-purpose, thus making it less straightforward to use to begin with. We directly quote from the review article [19]: "The main obstacle involved in devising such a library is that the performance and robustness often depend largely on the inner linear solver method, and its application-specific data structures and preconditioners. So, a general purpose package must use abstract layers around the linear algebra so the bifurcation analysis algorithms can be written independently of the linear algebra. This makes it harder to write, and to interface to, than might be expected."

One potential issue with the existing software implementations currently available is that they tend to require that the user describes the problem in their own language; for example, `defcon` works with `FEniCS` or `Firedrake`, and `pde2path` only solves PDE systems of a certain (albeit broad) type, using its own discretisation. Note also that `BifurcationKit.jl` does not natively solve PDE systems, but the companion package `GridapBifurcationKit.jl` [63] allows the solution of such problems using the Julia Finite Element packages `Gridap.jl` [2].

[4] `https://dsweb.siam.org/Software`

[5] `https://bifurcationkit.github.io/BifurcationKitDocs.jl/stable/#Other-softwares`

Table 1: Summary of continuation software (ordered by programming language and then alphabetically). Within the algorithms implemented column we assume a regular Newton method by default unless otherwise specified and for pseudo-arclength continuation only list the predictors when they can be outside of the default tangent predictor.

Software	Last Updated	Language	Licence	Large-scale?	Deflation?	Main algorithms implemented	Notes
`COCO` [13]	March 2020	MATLAB	GPL			Pseudo-arclength with Newton/damped Newton	
`ManLab` [34]	August 2019 (v4.1.7)	MATLAB	CeCILL-C			Asymptotic numerical method (ANM)	Requires nonlinear system to be polynomial with quadratic nonlinearity
`MatCont` [17]	January 2023 (v7.4)	MATLAB	None			Pseudo-arclength/Moore–Penrose	
`pde2path` [49]	September 2021 (v3.0)	MATLAB	GPL	✓	✓	Pseudo-arclength with regular/chord Newton; natural/quadratic/multiple predictors also available	PDEs based on OOPDE or FSElib (high order). Only solves problems of a certain PDE class
`BifurcationKit.jl` [64]	June 2023 (v0.2.9)	Julia	MIT	✓	✓	Pseudo-arclength/Moore–Penrose with natural/polynomial/multiple predictors also available as well as the asymptotic numerical method (ANM)	
`defcon` [28]	June 2023	Python	GPL	✓	✓	Pseudo-arclength with tangent/secant predictor	Uses either `Firedrake` or `FEniCS`
`AUTO` [20]	August 2021 (v0.9.3)	Fortran/ Python	None			Pseudo-arclength	
`LOCA` [58]	June 2023 (v14.2.0)	C++	BSD	✓		Pseudo-arclength with natural/tangent/secant/ random predictor	Part of the `Trilinos` package
`oomph` [37]	January 2022	C++	LGPL	✓		Pseudo-arclength	Supports continuation, but mainly a finite element library — continuation methods not well documented

7 Summary and conclusions

In this report, we have given a survey of the state-of-the-art in numerical continuation methods. This is a large field of study, with numerous well-written reference works and mature software packages, to which we have pointed to here. We have considered both classical approaches for tracing continuous solution branches and deflation techniques capable of finding disconnected branches. We have further discussed various phenomena which may be encountered and the relevance of symmetry in the underlying problem.

We have seen that some approaches will typically not scale well to large problems, this will likely include pseudo-arclength continuation due to requirement to solve large bordered systems unless a robust and scalable solver can be devised. Numerical bifurcation algorithms which we believe are possible at large scale are natural, secant and tangent continuation; step size control; time-stepping to find stable solutions; locating bifurcations via tracking eigenvalues of the Jacobian; branch switching; and deflated continuation (including in the presence of simple symmetries).

In the context of NEPTUNE, we have highlighted that many of the existing implementations rely on a (dense) LU-factorisation to solve the linear systems, which is infeasible for large-scale problems arising from PDEs. It is important that the software employed in the project allows the use of custom linear system solvers and, in particular, permits the use of preconditioners that have been tuned for the specific problems of interest. Thus, a core part of the overall development should focus on providing efficient linear algebra and solvers, typically for the Jacobian systems that arise in the chosen numerical continuation method. Subproblem solvers that should be available from the underlying software for NEPTUNE are solving systems involving the Jacobian matrix; Newton's method (or quasi-Newton if desired); computing a small number of eigenvalues of the Jacobian (with real part closest to 0); and finding kernel vectors of the Jacobian and its adjoint.

Another potential issue is that in the presence of discretisation errors, even connected bifurcation diagrams may become disconnected, and so it is important that the continuation software used is paired with a deflation method in order to capture as many of the solution branches as possible. Deflation techniques also have the benefit of simplifying the overall approach, with no necessity to solve augmented systems or rely on potentially difficult branch switching procedures and avoidance of unintentional branch jumping. Nonetheless, in all approaches, having good initial guesses can be a great benefit and may be necessary. In this sense, as much physical knowledge of the underlying problem should be utilised as possible rather than the whole procedure seen as a black-box routine.

References

[1] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. SIAM, 2003.

[2] S. Badia and F. Verdugo. Gridap: An extensible Finite Element toolbox in Julia. *Journal of Open Source Software*, 5(52):2520, 2020.

[3] R. E. Bank and H. D. Mittelmann. Stepsize selection in continuation procedures and damped Newton's method. *Journal of Computational and Applied Mathematics*, 26(1):67–77, 1989.

[4] W.-J. Beyn and V. Thümmler. Phase conditions, symmetries and PDE continuation. *Numerical Continuation Methods for Dynamical Systems: Path following and boundary value problems*, pages 301–330, 2007.

[5] I. Bogle, G. M. Slota, E. G. Boman, K. D. Devine, and S. Rajamanickam. Parallel graph coloring algorithms for distributed GPU environments. *Parallel Computing*, 110:102896, 2022.

[6] N. Boullé, E. G. Charalampidis, P. E. Farrell, and P. G. Kevrekidis. Deflation-based identification of nonlinear excitations of the three-dimensional Gross-Pitaevskii equation. *Physical Review A*, 102 (5):053307, 2020.

[7] N. Boullé, V. Dallas, and P. E. Farrell. Bifurcation analysis of two-dimensional Rayleigh-Bénard convection using deflation. *Physical Review E*, 105(5):055106, 2022.

[8] K. M. Brown and W. B. Gearhart. Deflation techniques for the calculation of further solutions of a nonlinear system. *Numerische Mathematik*, 16:334–342, 1971.

[9] T. F. Chan. Deflation techniques and block-elimination algorithms for solving bordered singular systems. *SIAM Journal on Scientific and Statistical Computing*, 5(1):121–134, 1984.

[10] T. F. Chan and D. C. Resasco. Generalized deflated block-elimination. *SIAM Journal on Numerical Analysis*, 23(5):913–924, 1986.

[11] E. Charalampidis, P. Kevrekidis, and P. E. Farrell. Computing stationary solutions of the two-dimensional Gross–Pitaevskii equation with deflated continuation. *Communications in Nonlinear Science and Numerical Simulation*, 54:482–499, 2018.

[12] E. Charalampidis, N. Boullé, P. E. Farrell, and P. G. Kevrekidis. Bifurcation analysis of stationary solutions of two-dimensional coupled Gross–Pitaevskii equations using deflated continuation. *Communications in Nonlinear Science and Numerical Simulation*, 87:105255, 2020.

[13] H. Dankowicz and F. Shilder. `COCO`, January 2020. URL `https://sourceforge.net/projects/cocotools/`.

[14] C. den Heijer and W. C. Rheinboldt. On steplength algorithms for a class of continuation methods. *SIAM Journal on Numerical Analysis*, 18(5):925–948, 1981.

[15] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer, 2011.

[16] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems*. CRC Press, third edition, 2022.

[17] A. Dhooge, W. Govaerts, Y. A. Kuznetsov, H. G. E. Meijer, and B. Sautois. `MatCont`, 2023. URL `https://sourceforge.net/projects/matcont/`.

[18] H. A. Dijkstra and F. W. Wubs. *Bifurcation Analysis of Fluid Flows*. Cambridge University Press, 2023.

[19] H. A. Dijkstra, F. W. Wubs, A. K. Cliffe, E. Doedel, I. F. Dragomirescu, B. Eckhardt, A. Y. Gelfgat, A. L. Hazel, V. Lucarini, A. G. Salinger, et al. Numerical bifurcation methods and their application to fluid dynamics: analysis beyond simulation. *Communications in Computational Physics*, 15(1): 1–45, 2014.

[20] E. Doedel, A. Champneys, F. Dercole, T. Fairgrieve, Y. Kuznetsov, B. Oldeman, R. Paffenroth, B. Sandstede, X. Wang, and C. Zhang. `AUTO`, 2021. URL `http://cmvl.cs.concordia.ca/auto/`.

[21] P. G. Drazin. *Nonlinear systems*. Cambridge University Press, 1992.

[22] D. B. Emerson, P. E. Farrell, J. H. Adler, S. P. MacLachlan, and T. J. Atherton. Computing equilibrium states of cholesteric liquid crystals in elliptical channels with deflation algorithms. *Liquid Crystals*, 45(3):341–350, 2018.

[23] P. E. Farrell. Lecture notes from a short course on numerical bifurcation analysis at the EMS summer school on mathematical modelling, numerical analysis and scientific computing in Kácov, Czechia, 2023. `https://pefarrell.org/files/kacov2023.pdf`. Accessed: 5th June 2023.

[24] P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393, 2013.

[25] P. E. Farrell, A. Birkisson, and S. W. Funke. Deflation techniques for finding distinct solutions of nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 37(4):A2026–A2045, 2015.

[26] P. E. Farrell, C. H. L. Beentjes, and Á. Birkisson. The computation of disconnected bifurcation diagrams. *arXiv preprint arXiv:1603.00809*, 2016.

[27] P. E. Farrell, M. Croci, and T. M. Surowiec. Deflation for semismooth equations. *Optimization Methods and Software*, 35(6):1248–1271, 2020.

[28] P. E. Farrell, J. Pollard, R. C. Kirby, J. Blechta, M. Croci, N. J. C. Sime, and N. Boullé. `defcon`, 2023. URL `https://bitbucket.org/pefarrell/defcon/src/master/`.

[29] A. H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.

[30] K. Georg. A note on stepsize control for numerical curve following. In B. Curtis Eaves, F. J. Gould, H.-O. Peitgen, and M. J. Todd, editors, *Homotopy Methods and Global Convergence*, pages 145–154. Springer, 1983.

[31] W. Govaerts. Stable solvers and block elimination for bordered systems. *SIAM Journal on Matrix Analysis and Applications*, 12(3):469–483, 1991.

[32] W. Govaerts and J. Pryce. Mixed block elimination for linear systems with wider borders. *IMA Journal of Numerical Analysis*, 13(2):161–180, 1993.

[33] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

[34] L. Guillot, B. Cochelin, and C. Vergez. `ManLab`, January 2019. URL `http://diamanlab.lma.cnrs-mrs.fr/`. v4.1.7.

[35] L. Guillot, B. Cochelin, and C. Vergez. A generic and efficient Taylor series–based continuation method using a quadratic recast of smooth nonlinear systems. *International Journal for Numerical Methods in Engineering*, 119(4):261–280, 2019.

[36] L. Guillot, B. Cochelin, and C. Vergez. A Taylor series-based continuation method for solutions of dynamical systems. *Nonlinear Dynamics*, 98:2827–2845, 2019.

[37] M. Heil and A. Hazel. `oomph`, 2022. URL `https://oomph-lib.github.io/oomph-lib/doc/html/`.

[38] V. Henson et al. Multigrid methods nonlinear problems: an overview. *Computational Imaging*, 5016: 36–48, 2003.

[39] R. B. Hoyle. *Pattern Formation: An Introduction to Methods*. Cambridge University Press, 2006.

[40] H. B. Keller. The bordering algorithm and path following near singular points of higher nullity. *SIAM Journal on Scientific and Statistical Computing*, 4(4):573–582, 1983.

[41] H. B. Keller. Lectures on numerical methods in bifurcation problems. *Lectures on Mathematics and Physics, Tata Institute of Fundamental Research (Bombay), 1987*, 1987.

[42] Y. A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer, fourth edition, 2023.

[43] S. Léger, J. Deteix, and A. Fortin. A Moore–Penrose continuation method based on a Schur complement approach for nonlinear finite element bifurcation problems. *Computers & Structures*, 152: 173–184, 2015.

[44] S. Léger, P. Larocque, and D. LeBlanc. Improved Moore-Penrose continuation algorithm for the computation of problems with critical points. *Computers & Structures*, 281:107009, 2023.

[45] C. C. Margossian. A review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 9(4):e1305, 2019.

[46] V. S. Medvedev. The bifurcation of the "blue sky catastrophe" on two-dimensional manifolds. *Mathematical Notes*, 51(1):76–81, 1992.

[47] Z. Mei. *Numerical Bifurcation Analysis for Reaction-Diffusion Equations*. Springer, 2000.

[48] H. Meijer, F. Dercole, and B. Oldeman. Numerical bifurcation analysis. In R. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 6329–6352. Springer, 2009.

[49] A. Meiners, J. Rademacher, H. Uecker, H. deWitt, T. Dohnal, and D. Wetzel. `pde2path`, July 2023. URL `https://www.staff.uni-oldenburg.de/hannes.uecker/pde2path/`.

[50] S. Mitusch, S. Funke, and J. Dokken. dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake. *Journal of Open Source Software*, 4(38):1292, 2019.

[51] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[52] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker. Globalization techniques for Newton–Krylov methods and applications to the fully coupled solution of the Navier–Stokes equations. *SIAM Review*, 48(4):700–721, 2006.

[53] B. T. Polyak. Newton–Kantorovich method and its global convergence. *Journal of Mathematical Sciences*, 133(4):1513–1523, 2006.

[54] J. Sánchez, M. Net, B. Garcıa-Archilla, and C. Simó. Newton–Krylov continuation of periodic orbits for Navier–Stokes flows. *Journal of Computational Physics*, 201(1):13–33, 2004.

[55] R. Seydel. *Practical Bifurcation and Stability Analysis*. Springer Science & Business Media, 2009.

[56] G. M. Shroff and H. B. Keller. Stabilization of unstable procedures: the recursive projection method. *SIAM Journal on Numerical Analysis*, 30(4):1099–1120, 1993.

[57] J. Swift and P. C. Hohenberg. Hydrodynamic fluctuations at the convective instability. *Physical Review A*, 15(1):319, 1977.

[58] The LOCA Project Team (Sandia National Laboratories). `LOCA`, 2023. URL `https://trilinos.github.io/nox_and_loca.html`.

[59] L. S. Tuckerman. Computational challenges of nonlinear systems. *Emerging Frontiers in Nonlinear Science*, pages 249–277, 2020.

[60] L. S. Tuckerman and D. Barkley. Bifurcation analysis for timesteppers. In E. Doedel and L. Tuckerman, editors, *Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems*, pages 453–466. Springer, 2000.

[61] H. Uecker. *Numerical Continuation and Bifurcation in Nonlinear PDEs*. SIAM, 2021.

[62] H. Uecker. Continuation and bifurcation in nonlinear PDEs–Algorithms, applications, and experiments. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, pages 1–38, 2022.

[63] R. Veltz. `GridBifurcationKit.jl`, 2022. URL `https://github.com/bifurcationkit/GridapBifurcationKit`.

[64] R. Veltz. `BifurcationKit.jl`, June 2023. URL `https://bifurcationkit.github.io/BifurcationKitDocs.jl/stable/`.

[65] H. F. Walker. An adaptation of Krylov subspace methods to path following problems. *SIAM Journal on Scientific Computing*, 21(3):1191–1198, 1999.

[66] I. Waugh, S. Illingworth, and M. Juniper. Matrix-free continuation of limit cycles for bifurcation analysis of large thermoacoustic systems. *Journal of Computational Physics*, 240:225–247, 2013.