Chapter 4

Arclength Continuation

In this chapter, we discuss the arclength continuation algorithms employed in the present work to compute branches of steady-state solutions. Readers already familiar with pseudo-arclength continuation methods may want to skip this chapter and proceed directly to Ch. 5 where the results of the computations are discussed.

This chapter is arranged in the following way: in §4.1, we introduce the notation and several of the basic ideas associated with continuation schemes, as well as giving several references to relevant literature. In §4.2 we discuss the solution of an auxiliary "tangent" system for the generation of initial guesses for the Newton iterations used to solve the augmented arclength continuation system. In §4.3, additional details regarding the pseudo-arclength constraint used in this and other works are given.

Practical considerations of the method, including the use of a scaled form of the arclength constraint, the technique for starting the method, some adaptive arclength stepsize selection techniques, and finally a simple algorithm for solving the sparse, bordered linear systems arising from the arclength continuation method are given in §4.4. Finally, a discussion of the very important

topic of linear stability analysis and of issues related to the numerical solution of the associated generalized eigenvalue problem are given in §4.5.

4.1 Introduction and Notation

Suppose that we have a PDE which depends on a parameter λ. We write this in an abstract notation as

$$G(u, \lambda) = 0 \tag{4.1}$$

where (in the continuous setting) $G : \mathbb{B} \times \mathbb{R} \to \mathbb{B}$ for some Banach space $\mathbb{B}$, or in the finite-dimensional case which we will be considering here, $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, and $u \in \mathbb{R}^n$ can be thought of as e.g. a vector of finite element solution coefficients. Certain smoothness properties of G are assumed to exist. Since u depends on the parameter λ, we may think of a "branch" of solutions $u(\lambda)$ which satisfy Eqn. (4.1).

In many physically meaningful situations, it happens that for a particular value of λ, the Jacobian G_u is singular. This occurs for non-isolated solutions (points where two different solution branches cross) and for turning (also known as limit) points, in which solutions do not exist for particular ranges of the parameter or where the character of the solution changes abruptly with small changes in λ. A popular method for dealing with this situation is the arclength continuation method described by Keller [106]. We have in fact adopted the same notation as Keller in the preceding paragraph. In the arclength continuation method, Eqn. (4.1) is augmented with a constraint

equation as

$$G(u, \lambda) = 0 \tag{4.2}$$

$$N(u, \lambda, s) = 0 \tag{4.3}$$

where s is the arclength parameter, defined as the distance along the solution branch (i.e. s is non-decreasing even if λ is decreasing along the branch). The nonlinear constraint N may depend explicitly on the arclength parameter s, and of course the solution $u = u(s)$ and also the parameter $\lambda = \lambda(s)$ depend implicitly on the arclength as well.

The main difference in the solution algorithm is that now instead of solving (for $k = 0, 1, \ldots$, via the inexact Newton method) the system

$$G_u^k \delta u^k = -G^k \tag{4.4}$$

associated with Eqn. (4.1), we now must consider the *augmented* Jacobian system

$$\begin{bmatrix} G_u & G_\lambda \\ N_u^t & N_\lambda \end{bmatrix}^k \begin{bmatrix} \delta u \\ \delta \lambda \end{bmatrix}^{k+1} = \begin{bmatrix} -G \\ -N \end{bmatrix}^k \tag{4.5}$$

associated with Eqs. (4.2) and (4.3). There are of course many different numerical schemes [25, 51] for solving Eqn. (4.5) efficiently. A large number of practical implementations of the arclength continuation [49, 86] and bifurcation detection [26] methods are to be found in the literature, often in combination with other techniques for determining e.g. the stability of the steady states computed using the method [173, 186].

4.2 Generation of Initial Guesses

An important aspect in these methods (which is sometimes left unmentioned, or not well-explained) is the technique by which initial guesses for the Newton iterations of the system given in Eqn. (4.5) are generated. Clearly, this is an important issue because a bad initial guess may not converge or may converge to the wrong solution (i.e. a solution on a different branch) if the initial guess is not sufficiently "good".

The main idea in generating the initial guess at each step is to introduce the variable $x := [u, \lambda]$, and think of the pair of Eqs. (4.2) and (4.3) as

$$P(x(s), s) := \begin{bmatrix} G(x(s)) \\ N(x(s), s) \end{bmatrix} \tag{4.6}$$

On a solution arc $x(s) = [u(s), \lambda(s)]$ of Eqs. (4.2) and (4.3) we therefore have $P(x(s), s) = 0$. Another way of thinking of this (and the key point in this discussion) is that if we were to "plot" P given above versus s the result would simply be the constant value of zero for each s. For such a constant function, it must also be true that the "derivative" of P with respect to s is the zero vector, i.e.

$$\frac{dP}{ds} = 0 \tag{4.7}$$

By chain-differentiation of P we obtain

$$\begin{aligned} \frac{dP}{ds} &= \begin{bmatrix} \frac{dG}{ds} \\ \frac{dN}{ds} \end{bmatrix} \\ &= \begin{bmatrix} G_u & G_\lambda \\ N_u^t & N_\lambda \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial s} \\ \frac{\partial \lambda}{\partial s} \end{bmatrix} + \begin{bmatrix} 0 \\ N_s \end{bmatrix} \end{aligned} \tag{4.8}$$

Combining Eqn. (4.8) with Eqn. (4.7) one arrives at

$$\begin{bmatrix} G_u & G_\lambda \\ N_u^t & N_\lambda \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial s} \\ \frac{\partial \lambda}{\partial s} \end{bmatrix} = \begin{bmatrix} 0 \\ -N_s \end{bmatrix} \tag{4.9}$$

which is an equation for determining the "tangent" vector $\dot{x}(s) := \partial x/\partial s$ to the solution arc $x(s)$. Eqn. (4.9) conveniently involves the same matrix as the augmented Jacobian system of Eqn. (4.5). Of course, the purpose of computing this tangent vector is that it leads us to a predictor (i.e. initial guess) for Newton's method, given by

$$x^{n+1} = x^n + \Delta s_n \dot{x}^n \tag{4.10}$$

for a fixed arclength step size Δs_n. There are various methods for solving the system associated with Eqn. (4.9), the "best" way depends on the underlying numerical method which is used to solve the original PDE. The goal is for the additional tangent calculation to fit simply and naturally into the framework of the existing code.

Aside: we note that the same concept of a solution arc can also be applied to the simpler case (sometimes called first-order continuation) where an auxiliary arclength equation is not present. In this case, the solution arc is $G(u(\lambda), \lambda) = 0$ for each value of λ on which the arc is defined, and thus

$$\begin{aligned} \frac{dG}{d\lambda} &= G_u \frac{\partial u}{\partial \lambda} + G_\lambda \\ &= 0 \end{aligned} \tag{4.11}$$

This defines the linear system

$$G_u \frac{\partial u}{\partial \lambda} = -G_\lambda \tag{4.12}$$

to be solved for $\frac{\partial u}{\partial \lambda}$. Eqn. (4.12) involves the same Jacobian matrix as for the original problem, but a different right-hand side. The benefit of solving this extra system of equations is an improved initial guess for u at the new value of the control parameter, given by

$$u^{n+1} = u^n + \left(\frac{\partial u}{\partial \lambda}\right)^n \Delta\lambda \tag{4.13}$$

for a given fixed step size $\Delta\lambda$. While it may indeed be possible to improve the convergence of the Newton iterations using this scheme, we should point out that it does not in general allow one to navigate a turning point in the solution path, and in practice this first-order continuation technique has not converged for us in any case where the even simpler "zeroth-order" continuation did not converge as well.

4.3 The Pseudo-Arclength Constraint

We still need to specify more precisely the form of the arclength constraint in Eqn. (4.3) which is to be used. Consider two "nearby" points (u, λ_u) and (v, λ_v) on a solution path P in $\mathbb{R}^{n+1}$, where n is the dimension of the finite element solution vector. This is depicted for $n = 2$ in Fig. 4.1. Note that the same concepts hold in general normed linear spaces, though they are easier to describe geometrically. Now consider the projection of the vector $u - v$ onto $\mathbb{R}^n$ defined by $\Delta u := (u_1 - v_1, \ldots, u_n - v_n, 0)$ and the projection onto $\mathbb{R}$ defined by $\Delta\lambda := (0, \ldots, 0, \lambda_u - \lambda_v)$. Then clearly the vectors Δu and $\Delta\lambda$ are

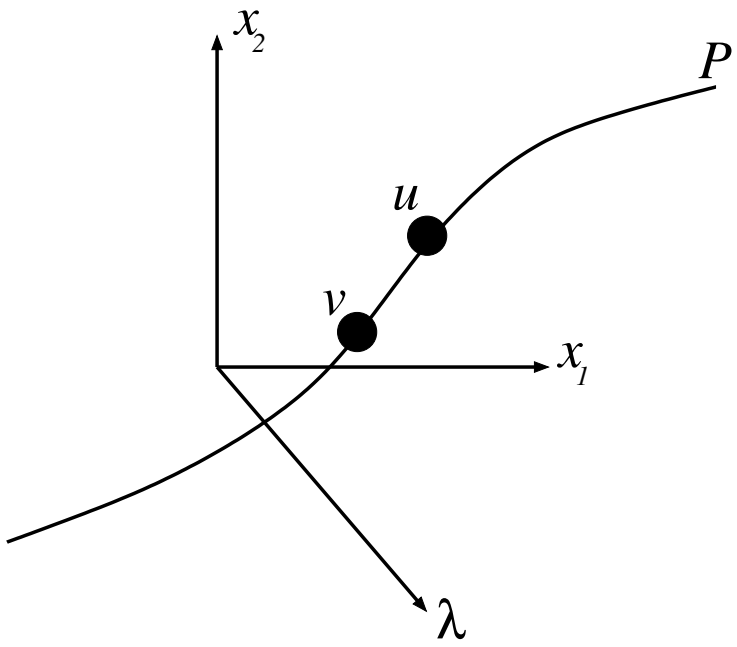

Figure 4.1: Solution path P with representative solutions u and v.

orthogonal and so their respective lengths are related by

$$\|\Delta u\|^2 + \|\Delta\lambda\|^2 = \|\Delta s\|^2 \tag{4.14}$$

in the limit as $u \to v$, for some vector Δs. In the limit as $\Delta s \to 0$, we obtain

$$\left\|\frac{\partial u}{\partial s}\right\|^2 + \left(\frac{\partial \lambda}{\partial s}\right)^2 = 1 \tag{4.15}$$

where the norm $\|\cdot\|$ is the standard Euclidian norm on $\mathbb{R}^{n+1}$. Eqn. (4.15) is in the form of Eqn. (4.3), and can used as the supplemental arclength constraint equation. However, most authors [106] agree that the use of a *nonlinear* constraint such as Eqn. (4.15) is cumbersome in practice, and instead a linearized constraint of some form is preferred. Such a linearized constraint follows from

Eqn. (4.14) by assuming $v = u(s_i)$ for some arbitrary point s_i is known, and setting $\|\Delta s\| = s - s_i$ for arbitrary s. We obtain the (discrete in s) nonlinear arclength constraint

$$N^{nl}(u, \lambda, s) := \|u - u(s_i)\|^2 + \|\lambda - \lambda(s_i)\|^2 - (s - s_i)^2 = 0 \tag{4.16}$$

which is locally accurate as $s \to s_i$. Recall that for the augmented Newton's method described in Eqn. (4.5), we have

$$(N_u^{nl})^t \frac{\partial u}{\partial s} + N_\lambda^{nl} \frac{\partial \lambda}{\partial s} + N_s^{nl} = 0 \tag{4.17}$$

Now we suppose that, instead of just being the equation to satisfy for Newton's method, Eqn. (4.17) with $\frac{\partial u}{\partial s}$ and $\frac{\partial \lambda}{\partial s}$ evaluated at the prior solution $u(s_i)$, *is the arclength constraint itself.* In other words, our linearized arclength constraint is:

$$N(u, \lambda, s) := (N_u^{nl})^t \left.\frac{\partial u}{\partial s}\right|_{s_i} + N_\lambda^{nl} \left.\frac{\partial \lambda}{\partial s}\right|_{s_i} + N_s^{nl} = 0 \tag{4.18}$$

Differentiating N^{nl} with respect to u, λ, and s yields

$$(N_u^{nl})^t \phi = 2\,(u - u(s_i))^t\,\phi \tag{4.19}$$

$$N_\lambda^{nl} = 2(\lambda - \lambda(s_i)) \tag{4.20}$$

$$N_s^{nl} = -2(s - s_i) \tag{4.21}$$

for arbitrary vector ϕ. Hence, our linearized arclength constraint becomes, in its final form,

$$N(u, \lambda, s) := (u - u(s_i))^t \left.\frac{\partial u}{\partial s}\right|_{s_i} + (\lambda - \lambda(s_i)) \left.\frac{\partial \lambda}{\partial s}\right|_{s_i} - (s - s_i) \tag{4.22}$$

4.4 Practical Considerations

In this section, we discuss several of the practical aspects involved with solving the augmented system numerically, including the details of the numerical solution of the augmented (or bordered) system in §4.4.4, and some of the details of starting up the method in §4.4.2.

4.4.1 Scaling

It is not in general feasible to use Eqn. (4.22) (as given) as a pseudo-arclength constraint, due primarily to the vastly different scales the first two terms in Eqn. (4.22) may assume. Therefore, in practice we will use a penalized or regularized version of pseudo-arclength, given by

$$N_\alpha(u, \lambda, s) := \alpha_i^2 \left(u - u(s_i)\right)^t \left.\frac{\partial u}{\partial s}\right|_{s_i} + \left(\lambda - \lambda(s_i)\right) \left.\frac{\partial \lambda}{\partial s}\right|_{s_i} - (s - s_i) \qquad (4.23)$$

where $\alpha_i \in \mathbb{R}$ are scaling parameters chosen in order to balance the relative magnitudes of the first two terms of Eqn. (4.23). To motivate the definition of α, consider the normalized "solution triangle" shown in Fig. 4.2, which has hypotenuse 1 and legs of length $\left|\frac{\partial \lambda}{\partial s}\right|$ and $\left\|\frac{\partial u}{\partial s}\right\|$ (in the continuous setting, or $\left|\frac{\Delta \lambda}{\Delta s}\right|$ and $\left\|\frac{\Delta u}{\Delta s}\right\|$ in the discrete-in-s setting). Clearly, if either of the legs is much longer than the other, then one of the terms on the left-hand side of the full arclength equation (originally Eqn. (4.15), repeated here for convenience)

$$\left\|\frac{\partial u}{\partial s}\right\|^2 + \left(\frac{\partial \lambda}{\partial s}\right)^2 = 1 \qquad (4.24)$$

will be nearly 1, while the other is nearly zero. In such cases, the included angle γ (shown in Fig. 4.2) will tend to either 0 or $\pi/2$, and the benefits of performing

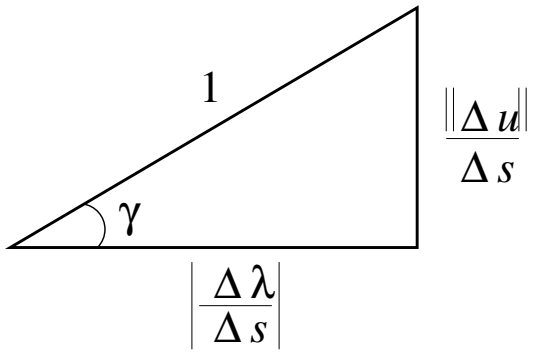

Figure 4.2: Normalized solution triangle with angle γ shown.

arclength continuation will be effectively lost. For arc-step $i = 1, 2, \ldots$, we define the regularization parameter α_i such that

$$\alpha_i \tan \gamma_i = \alpha_0 \tag{4.25}$$

for some constant α_0, whose determination we will discuss in §4.4.2 on initialization. There is apparently nothing unique in this definition of α_i, it is defined similarly to the method implemented in the Library of Continuation Algorithms (LOCA) from Sandia National Labs [172]. Note that the "equivalent" scaled full arclength equation is now

$$\alpha^2 \left\| \frac{\partial u}{\partial s} \right\|^2 + \left(\frac{\partial \lambda}{\partial s} \right)^2 = 1 \tag{4.26}$$

and the included angle may be computed at step i via

$$\tan \gamma_i := \frac{\sqrt{1 - \left(\frac{\partial \lambda}{\partial s}\big|_{s_i} \right)^2}}{\left| \frac{\partial \lambda}{\partial s}\big|_{s_i} \right|} \tag{4.27}$$

4.4.2 Initialization

Initializing the continuation algorithm can be done in a few different ways, we will only discuss one possible way here. The main difficulty is that, when one is using the linearized arclength constraint form given in Eqn. (4.22), or more precisely the suitably-normalized constraint given by Eqn. (4.23), one requires the solution and parameter derivatives with respect to s from the previous step.

At the first arc-step, of course, there is no previous step, and so we assume that two previous solutions: $\{u_0, \lambda_0\}$ and $\{u_1, \lambda_1\}$ have been computed *without* requiring arclength continuation. For example, in some problems $\{u_0, \lambda_0\} = \{0, 0\}$, and we can obtain u_1 with a small increment of the control parameter, making use of e.g. zeroth-order continuation. Then, on the first step, we impose the requirement

$$\left.\frac{\partial \lambda}{\partial s}\right|_{s_0} = \pm \frac{1}{\sqrt{2}} \tag{4.28}$$

where the sign is the same as the sign of $\lambda_1 - \lambda_0$. In other words, we use Eqn. (4.28) to ensure that the initial solution triangle is right-isosceles. Then we seek α_0 such that the scaled arclength constraint of Eqn. (4.26),

$$\alpha_0^2 \left\| \left.\frac{\partial u}{\partial s}\right|_{s_0} \right\|^2 + \left(\left.\frac{\partial \lambda}{\partial s}\right|_{s_0} \right)^2 = 1 \tag{4.29}$$

is satisfied subject to the requirement of Eqn. (4.28). We can factor out $\partial\lambda/\partial s$ via chain differentiation to obtain

$$\left(\left.\frac{\partial \lambda}{\partial s}\right|_{s_0} \right)^2 \left[1 + \alpha_0^2 \left\| \left.\frac{\partial u}{\partial \lambda}\right|_{s_0} \right\|^2 \right] = 1 \tag{4.30}$$

Then, we can solve for α_0 to obtain

$$\alpha_0 = \left\| \left. \frac{\partial u}{\partial \lambda} \right|_{s_0} \right\|^{-1} \tag{4.31}$$

where $\left. \frac{\partial u}{\partial \lambda} \right|_{s_0}$ is approximated from the initial two solutions via finite differences

$$\left. \frac{\partial u}{\partial \lambda} \right|_{s_0} \approx \frac{u_1 - u_0}{\lambda_1 - \lambda_0} \tag{4.32}$$

The initial solution tangent with respect to s is then

$$\left. \frac{\partial u}{\partial s} \right|_{s_0} = \left. \frac{\partial \lambda}{\partial s} \right|_{s_0} \frac{u_1 - u_0}{\lambda_1 - \lambda_0} \tag{4.33}$$

and the arc-distance traveled in the first step is

$$\Delta s = \frac{\lambda_1 - \lambda_0}{\left. \frac{\partial \lambda}{\partial s} \right|_{s_0}} \tag{4.34}$$

Now that we have the initial tangents $\left. \frac{\partial \lambda}{\partial s} \right|_{s_0}$ and $\left. \frac{\partial u}{\partial s} \right|_{s_0}$, we can assemble the Jacobian and residual at $\{u_1, \lambda_1\}$ and solve the tangency system (Eqns. (4.9)) to obtain $\left. \frac{\partial \lambda}{\partial s} \right|_1$ and $\left. \frac{\partial u}{\partial s} \right|_1$. Finally, α_1 can be computed according to the formula given in Eqn. (4.25).

The initial guess (represented by the dashed line in Fig. 4.3) for $\{u_2, \lambda_2\}$ is then obtained using these tangents as

$$\tilde{u}_2 = u_1 + \left. \frac{\partial u}{\partial s} \right|_{s_1} \Delta s \tag{4.35}$$

$$\tilde{\lambda}_2 = \lambda_1 + \left. \frac{\partial \lambda}{\partial s} \right|_{s_1} \Delta s \tag{4.36}$$

The tangents $\left. \frac{\partial u}{\partial s} \right|_{s_1}$ and $\left. \frac{\partial \lambda}{\partial s} \right|_{s_1}$ are then also used in the Newton iterations for obtaining $\{u_2, \lambda_2\}$, as well as updating the tangents at point 2. The continuation algorithm then proceeds as described previously for steps 3,4,...

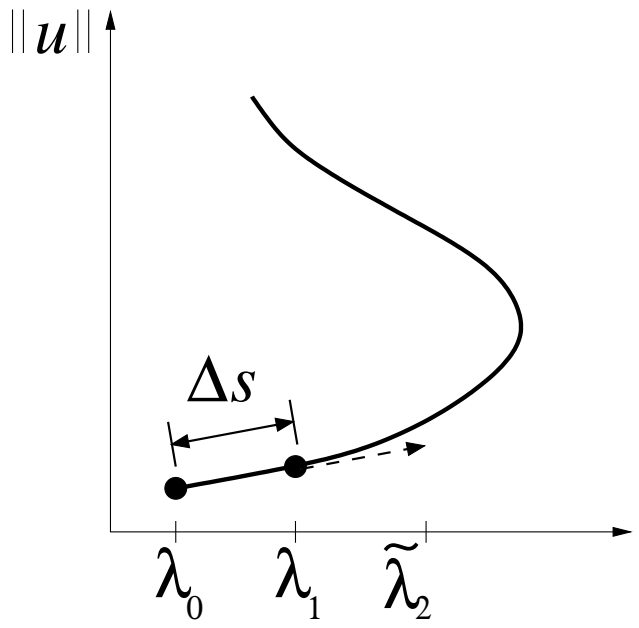

Figure 4.3: Procedure for starting the arclength continuation procedure.

One particularly attractive feature of this form of pseudo-arclength constraint is that enough information is present (if one is willing to save a single additional vector) to compute a higher-order predictor. This predictor, which is based on the second-order explicit Adams-Bashforth formula with variable-size steps is given by

$$\tilde{u}_{n+1} = u_n + \frac{\Delta s_{n+1}}{2}\left[\left(2 + \frac{\Delta s_{n+1}}{\Delta s_n}\right)\left.\frac{\partial u}{\partial s}\right|_n - \left(\frac{\Delta s_{n+1}}{\Delta s_n}\right)\left.\frac{\partial u}{\partial s}\right|_{n-1}\right] \qquad (4.37)$$

We have observed a moderate reduction (on the order of 5-10%) in the number of linear solver iterations required when solving the bordered systems using this predictor as the starting guess, with no additional calculation required and little additional storage overhead.

4.4.3 Adaptive Arclength Stepsize Selection

It is generally unwise to maintain a fixed arclength stepsize throughout the tracing of a solution branch for the following reasons:

1. The initial arclength stepsize may be small if the initial solutions u_0 and u_1 were relatively close together. It rarely makes sense, from an efficiency standpoint, to continue using such a small arclength stepsize in regions where the solution is not changing much with respect to the parameter λ.

2. Unless we can reduce the arclength stepsize, failure of the Newton iterations to converge must signal a failure for the method. This frequently occurs near bifurcations and turning points. The ability to reduce the arclength stepsize and try the Newton iterations again is thus key to the robustness of the branch tracing algorithm.

3. More detail (additional arc-steps) are often desirable near turning and bifurcation points, since the character of the solution typically changes rapidly in these regions, and the transition from one solution regime to another is a topic of special interest when computing the branches.

4. Without the ability to shrink the arclength stepsize, important features of the solution branch (such as e.g. turning points) can actually be "stepped over" and missed entirely. An example in which a particular tracing of a branch misses two turning points is shown in Fig. 4.4.

There are several possible techniques for adaptively choosing the arclength stepsize. One, which is used in the LOCA [172] library, is to compute the parameter

$$\tau_i := \frac{\frac{\partial u}{\partial \lambda}\Big|_{s_i} \cdot \frac{\partial u}{\partial \lambda}\Big|_{s_{i-1}}}{\left\| \frac{\partial u}{\partial \lambda}\Big|_{s_i} \right\| \left\| \frac{\partial u}{\partial \lambda}\Big|_{s_{i-1}} \right\|} \tag{4.38}$$

which is the cosine of the angle between the two most recent solution tangent vectors (with respect to λ). If the two most recent tangent vectors were in essentially the same direction, $\tau \to 1$, while if they were nearly orthogonal (such as near a turning point) $\tau \to 0$. The next arclength stepsize is then given by

$$\Delta s_{i+1} = \tau_i \Delta s_i \tag{4.39}$$

This method does require the storage of an additional vector (the old solution tangent) however this is a common characteristic of most steplength selection methods. In our experience, we found that when the solution vector u has entries varying by several orders of magnitude, then the contributions of the smaller components to the dot product in the numerator of τ can be essentially lost, rendering $\tau \approx 1$ even near turning points.

Another possibility is to simply scale the stepsize by the ratio of the norms of the two most recent solution tangents with respect to λ, i.e.

$$\Delta s_{i+1} = \frac{\left\| \frac{\partial u}{\partial \lambda}\Big|_{s_{i-1}} \right\|}{\left\| \frac{\partial u}{\partial \lambda}\Big|_{s_i} \right\|} \Delta s_i \tag{4.40}$$

This avoids the need to store the entire old tangent vector (we need store only its norm) and seems to work reasonably well at shrinking the stepsize in

practice. Unfortunately, it frequently appears to grow the stepsize more slowly than one might otherwise prefer in regions of slow solution change.

Another commonly-used technique for growing the arclength stepsize is to look at the number of Inexact Newton iterations required to converge the previous step. As in Seydel's excellent book [184], we may assume an "optimal" number of Newton iterations N_{opt} exists based on the other tolerances in a given problem, and then if the most recent continuation step i required N_i inexact Newton iterations, we scale the arclength stepsize according to

$$\Delta s_{i+1} = \frac{N_{\text{opt}}}{N_i} \Delta s_i \tag{4.41}$$

In a variation on this theme, and in acknowledgment of the fact that N_{opt} can be difficult to define in general, the LOCA authors adopt

$$\Delta s_{i+1} = \left[1 + a \left(\frac{N_{\text{max}} - N_i}{N_{\text{max}} - 1} \right)^2 \right] \Delta s_i \tag{4.42}$$

where N_{max} is the maximum-allowable number of Newton iterations, and a is an "aggressiveness" factor determining how quickly to grow the step. In practice, we have found that combining both Eqns. (4.40) and (4.42) with an aggressiveness factor $a \approx 1$ works reasonably well for arclength stepsize selection at negligible extra cost.

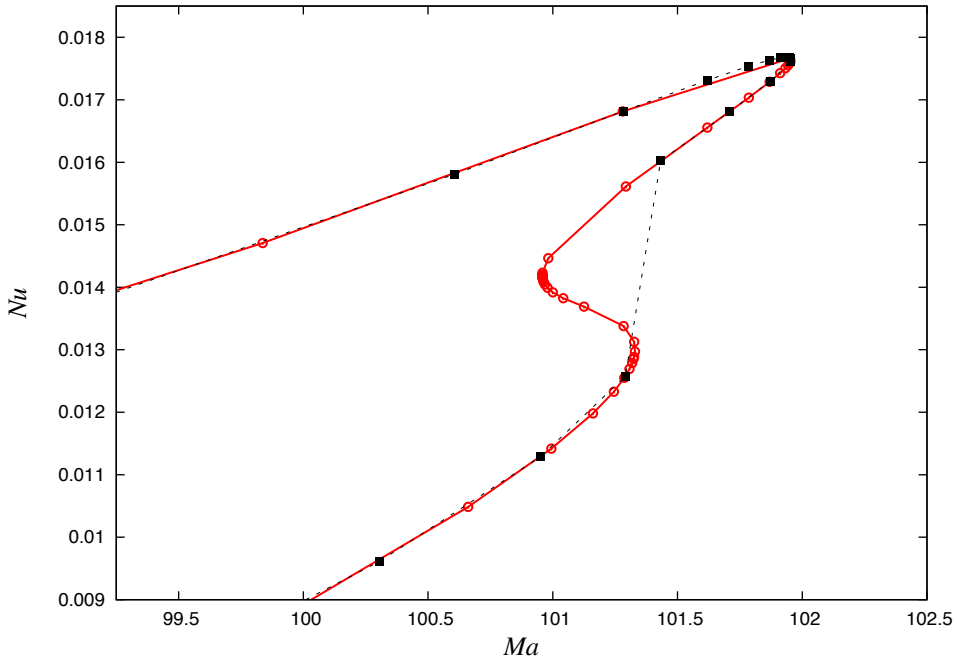

Figure 4.4: Example of two tracings (computed in opposite directions) showing the effect of too large an arclength stepsize. In this case, the continuation scheme was conducted in the Ma variable, first on the red branch (circular markers) going from top to bottom, and a second time with the black dashed-line branch (square markers) which goes from bottom to top. The second branch tracing misses two turning points because it fails to reduce the arclength stepsize quickly enough.

4.4.4 Numerical Solution of Bordered System

By inspection of Eqns. (4.5) and (4.9), it is obvious that one needs to frequently solve linear systems of the form

$$\begin{bmatrix} G_u & G_\lambda \\ N_u^t & N_\lambda \end{bmatrix} \begin{bmatrix} \delta u \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} r \\ \rho \end{bmatrix} \tag{4.43}$$

where the Jacobian matrix is reused, but the right-hand side and unknown vectors are interchangeable depending on whether we are solving the augmented system (Eqn. (4.5)) or the tangency condition (Eqn. (4.9)). While it would technically be possible to solve (4.43) directly or iteratively as a sparse, bordered system, it turns out to be more convenient (in terms of reuse of existing computational tools) to use the following two-solve procedure for each solve of Eqn. (4.43). In the case of Eqn. (4.5), Algorithm 1 must be employed at

Algorithm 1 Two-step solve procedure for systems in the form of Eqn. (4.43).

Solve $G_u y = G_\lambda$ for y (first solve).
Solve $G_u z = r$ for z (second solve).
Set $\delta\lambda = \frac{\rho - N_u^t z}{N_\lambda - N_u^t y}$
Set $\delta u = z - (\delta\lambda) y$

each Newton step, making this scheme roughly twice as expensive as a normal Newton solve, and therefore only attractive to use near turning points or regions of rapid change in the solution relative to the control parameter λ. To see how Algorithm 1 arises, simply write out the first "row" of Eqn. (4.43)

$$G_u \delta u + (\delta\lambda) G_\lambda = r \tag{4.44}$$

(Recall that $\delta\lambda$ is a scalar, G_λ is a vector of the same length as the residual vector G.) Multiplying Eqn. (4.44) by a notional G_u^{-1} (note: we of course do not explicitly form the inverse Jacobian, just compute its action) we obtain

$$\delta u + (\delta\lambda) G_u^{-1} G_\lambda = G_u^{-1} r \tag{4.45}$$

Now, by our definitions of y and z given in Algorithm 1 we have

$$\delta u + (\delta\lambda) y = z \tag{4.46}$$

Or simply,

$$\delta u = z - (\delta\lambda) y \tag{4.47}$$

From the second "row" of Eqn. (4.43) we obtain

$$N_u^t \delta u + (\delta\lambda) N_\lambda = \rho \tag{4.48}$$

Or, plugging in Eqn. (4.47) for δu

$$N_u^t \left(z - (\delta\lambda)\, y\right) + (\delta\lambda) N_\lambda = \rho \tag{4.49}$$

Finally, Eqn. (4.49) can be rearranged to obtain a scalar equation for $\delta\lambda$

$$\delta\lambda = \frac{\rho - N_u^t z}{N_\lambda - N_u^t y} \tag{4.50}$$

and plugging $\delta\lambda$ back into Eqn. (4.47) we obtain the final result for δu.

We observe that this two-step solve procedure never involves the "full" augmented matrix, and still requires linear system solves with the Jacobian G_u which may be singular. Therefore, while the scheme is attractive to implement in an existing numerical code because it is non-intrusive (does not

require changing the existing Jacobian assembly routine) it may not succeed if the solution ends up precisely at a turning point. However, Keller gives a proof [106] of how successive approximation iterations (and hints for Newton iterations as well) will always be able to "jump" over such singular points provided the initial guess lies in a particular domain of attraction around the singular point. In practice, we have never seen the scheme fail to pass a turning point provided it is allowed enough arclength stepsize reductions.

Another important issue is choosing the correct tolerance for the iterative solutions of the $G_u y = G_\lambda$ and $G_u z = -G$ systems at each Newton step. We typically employ the method of Eisenstat and Walker [76] discussed in §2.3 for the solution of the z system, since solving this system is the prototypical application of the inexact Newton method.

Some additional care must be taken when selecting the tolerance for the y system, since its right-hand side $G_\lambda \nrightarrow 0$ as the Newton iterations proceed. Our linear algebra package (PETSc [19]) requires the selection of a "relative" tolerance where relative in this instance means relative to $\|G_\lambda\|$. Therefore, we modify the standard tolerance selection of Eqn. (B-2.27) in a rather straightforward way by taking

$$\eta_k \leftarrow \eta_k \frac{\|G\|}{\|G_\lambda\|} \tag{4.51}$$

where $\|G\|$ is the current nonlinear system residual.

4.4.5 Boundary Conditions

There is also the question of how the boundary conditions, which are posed on the original problem $G(u, \lambda) = 0$ should enter into the augmented system. For example, how should boundary conditions be applied in the tangency system $G_u \frac{\partial u}{\partial \lambda} = -G_\lambda$? It seems logical that any non-homogeneous Dirichlet boundary conditions imposed on the original problem should be imposed as homogeneous Dirichlet conditions on the tangency condition, effectively enforcing that the change in that finite element coefficient with respect to changes in λ is zero.

4.5 Linear Stability Analysis

In this section, we consider the linear stability of the solutions u obtained on solution arcs using the pseudo-arclength continuation methods described previously. The general time-dependent form of Eqn. (4.1) may be written as

$$B\frac{\partial u}{\partial t} = G(u, \lambda) \tag{4.52}$$

where B is a linear operator (an $n \times n$ matrix in the discrete setting) which we will refer to from here on as the "mass matrix." A wide variety of problems can be handled by taking B as the $n \times n$ identity matrix, but for our purposes we will consider a general linear operator. Suppose further that a steady solution u_0 satisfying

$$0 = G(u_0, \lambda) \tag{4.53}$$

is known, and the linear stability of u_0 is of interest. We define

$$\eta := u - u_0 \tag{4.54}$$

as a "small" perturbation away from the known solution u_0. Differentiating η with respect to time, we observe that

$$\begin{aligned} B\dot{\eta} &= B\dot{u} - B\dot{u_0} \\ &= G(u,\lambda) - G(u_0,\lambda) \\ &= G(u,\lambda) \end{aligned} \tag{4.55}$$

Holding λ fixed, we expand about u_0 to obtain

$$\begin{aligned} B\dot{\eta} &= G(u_0+\eta,\lambda) \\ &= G(u_0,\lambda) + G_u(u_0,\lambda)\eta + \mathcal{O}(\|\eta\|^2) \\ &\approx G_u(u_0,\lambda)\eta \end{aligned} \tag{4.56}$$

since the first term on the right-hand side of Eqn. (4.56) vanishes by Eqn. (4.53). We now have a linear evolution problem for η. Inserting the *ansatz*

$$\eta = \exp(\sigma t)x \tag{4.57}$$

for σ a (possibly complex) scalar and $x \in \mathbb{R}^n$ (or $\mathbb{C}^n$) into Eqn. (4.56), and letting $A := G_u(u_0,\lambda)$ stand for the Jacobian matrix evaluated at u_0, one obtains

$$\sigma Bx = Ax \tag{4.58}$$

Eqn. (4.58) is in the standard form of a generalized eigenvalue problem, with eigenvalue σ and eigenvector x.

4.5.1 Numerical Solution of the Generalized Eigenvalue Problem

Direct calculation of the eigenvalues of an $n \times n$ matrix is an expensive $\mathcal{O}(n^3)$ operation, and is not feasible for the three-dimensional coupled heat transfer and fluid-flow systems considered here. Fortunately, linear stability analysis requires only knowledge of the sign of the "rightmost" (on the real axis) eigenvalue. If at least one eigenvalue has positive real part, the solution is linearly unstable, whereas, if all the eigenvalues have negative real part, the solution is said to be linearly stable.

A classical method for computing the largest-in-magnitude eigenvalue of the spectrum is the power iteration. Unfortunately, the power iteration is unsuitable for our purposes because (1) the largest-in-magnitude eigenvalue is not necessarily the right-most eigenvalue, and (2) it is known to have very poor convergence properties when σ_1 and σ_2 (the largest and second-largest in magnitude eigenvalues, respectively) are close together. (The convergence rate is geometric with ratio $|\sigma_1/\sigma_2|$.)

The Arnoldi iteration [11] somewhat overcomes difficulty (2) listed above by computing the eigenvalues of the orthogonal projection of the matrix A onto the Krylov subspace. When A is symmetric, the Arnoldi iteration is equivalent to the Lanczos iteration [171], another method for determining eigenvalues. In this work, we employ the Arnoldi iteration as implemented by the SLEPc [94] eigensolver library, which is a natural extension of the PETSc library, and makes extensive use of its Krylov subspace solvers.

The Arnoldi iteration, like the power iteration, also converges to the largest-in-magnitude eigenvalues of A, and hence does not overcome difficulty (1) listed above. For example, in our particular application, the time-independent constraint (continuity equation) rows of A effectively lead to $-\infty$ eigenvalues, and as such will be the eigenvalues obtained by the power or Arnoldi iterations. To overcome this issue, we typically employ the shift-and-invert spectral transformation (as implemented by the SLEPc library). Beginning with Eqn. (4.58), we choose a shift $s \neq \sigma$ and subtract sBx from both sides to obtain

$$(A - sB)\, x = (\sigma - s)\, Bx \tag{4.59}$$

In general it is possible to use a complex shift s, although in this work we employ only real-valued shifts. Next, multiplying both sides of Eqn. (4.59) by $(\sigma - s)^{-1} (A - sB)^{-1}$ we obtain

$$(\sigma - s)^{-1}\, x = (A - sB)^{-1}\, Bx \tag{4.60}$$

Now, Eqn. (4.60) is a standard eigenvalue problem

$$Tx = \gamma x \tag{4.61}$$

where $T := (A - sB)^{-1} B$ is the transformed matrix and $\gamma := (\sigma - s)^{-1}$ is the transformed eigenvalue. The inverse is of course never computed explicitly; its action is determined with linear system solves using PETSc's iterative methods.

The shift-and-invert transform maps eigenvalues $|\sigma| \gg |s|$ to $\gamma \approx 0$, ensuring that they are no longer the dominant eigenvalues of the spectrum and

that they are not converged by the Arnoldi iteration. This is ideal behavior for e.g. the $-\infty$ eigenvalues in our systems of interest. The Arnoldi iteration can also be made to converge very quickly to a suspected eigenvalue σ by choosing a shift s very close to said eigenvalue. This shift essentially maps σ to the dominant eigenvalue of the spectrum, and can be useful in cases where a positive eigenvalue is perhaps partially (but not to a sufficient tolerance, say) converged in a preceding Arnoldi iteration.

The shift-and-invert spectral transformation just described still suffers from the drawback that the eigenvalues which it converges are not guaranteed to be rightmost eigenvalues, just those closest to the shift s. A different, but related spectral transformation which attempts to map rightmost eigenvalues to the dominant eigenvalues of the spectrum is the Cayley transform [127]. In addition to the shift s, we also choose "anti-shift" $t \neq \sigma$, and beginning again with Eqn. (4.58) we now add tBx to both sides to obtain

$$(A + tB)\, x = (\sigma + t)\, Bx \tag{4.62}$$

Multiplying both sides of Eqn. (4.62) by $(\sigma - s)$ we obtain

$$\begin{aligned} (\sigma - s)\,(A + tB)\, x &= (\sigma - s)\,(\sigma + t)\, Bx \\ &= (\sigma + t)\,(\sigma Bx - sBx) \\ &= (\sigma + t)\,(Ax - sBx) \\ &= (\sigma + t)\,(A - sB)\, x \end{aligned} \tag{4.63}$$

Finally, we multiply both sides of Eqn. (4.63) by $(\sigma - s)^{-1}\,(A - sB)^{-1}$ to ob-

tain

$$(A - sB)^{-1}(A + tB)\,x = \left(\frac{\sigma + t}{\sigma - s}\right)x \tag{4.64}$$

Eqn. (4.64) is thus a standard eigenvalue problem in the same form as Eqn. (4.61), where now $T := (A - sB)^{-1}(A + tB)$ and $\gamma := \left(\frac{\sigma+t}{\sigma-s}\right)$. The Cayley transform has the property that it maps the line $Re\{\sigma\} = \frac{1}{2}(s - t)$ in the σ-plane to the unit circle in the γ-plane [127]. To see this, we can evaluate γ (as defined above) at an arbitrary point $\sigma^* = \frac{1}{2}(s - t) + bi$ ($b \in \mathbb{R}$ arbitrary) on the line $Re\{\sigma\} = \frac{1}{2}(s - t)$ defined previously to obtain

$$\gamma|_{\sigma^*} = \frac{\frac{1}{2}(s + t) + bi}{-\frac{1}{2}(s + t) + bi} \tag{4.65}$$

which, in polar coordinates, becomes

$$\gamma|_{\sigma^*} = \exp(2i\theta) \tag{4.66}$$

with $\theta := \tan^{-1}\left(\frac{2b}{s+t}\right)$, i.e., the unit circle in the γ-plane. Since, for linear stability analysis, we are mostly interested in eigenvalues σ with positive real part, it has been suggested [41, 118] that one should select the shift and anti-shift according to the following guidelines

1. Select $s > 0 \in \mathbb{R}$ such that $s > Re\{\sigma_1\}$ and, if possible, $|s| \approx Im\{\sigma_1\}$, where σ_1 is the rightmost eigenvalue of interest. (Note that this guideline is "implicit," since of course σ_1 is the eigenvalue we are trying to find. However, if an approximation to σ_1 is available, one may use it to select the shift by the preceding guideline.)

2. $t = s$

The second guideline is designed to ensure that the imaginary axis ($Re\{\sigma\} = 0$) is mapped to the unit circle in the γ-plane, and that eigenvalues to the right of the imaginary axis are mapped to dominant eigenvalues, and hence converged by Arnoldi iterations. Conversely, eigenvalues with negative real parts are mapped by this transformation to small magnitudes and hence not converged by the eigensolver.

In practice, the Cayley transform will converge to eigenvalues at $+\infty$ if any exist. This is in contrast to the shift-invert transform which maps eigenvalues at $\pm\infty$ to zero, and (though we are not completely clear why) appears to lead to problems when the penalty formulation is used to enforce Dirichlet boundary conditions. We believe the penalty terms may lead to spurious positive eigenvalues which are $\mathcal{O}$(penalty) in size, and are currently working to understand exactly what is happening.

Finally, even when using the Cayley transform, Lehoucq [118] states that "There is no available theory to verify that the rightmost eigenvalue has been computed." The Cayley transform is therefore no panacea, and the practitioner must rely on additional information which is available about the solution when determining if an eigenvalue with positive real part truly exists.

This additional information includes nearby converged time-accurate solutions (time-dependent solvers should only converge to linearly-stable steady states) as well as trusted eigenvalue information from other nearby steady so-

lutions. It is quite often possible to track the movement of a single negative eigenvalue as it approaches, and finally crosses over, the imaginary axis to become unstable. Finally, features of the solution branch, including turning points and symmetry-breaking bifurcations, generally signal the birth of a positive eigenvalue. For example, we know this is the case near turning points since the Jacobian is singular (has a zero eigenvalue) there, implying a change in sign of at least one eigenvalue near such points.