A parallel adaptive method for pseudo-arclength continuation

D A Aruliah\textsuperscript{1}, L van Veen\textsuperscript{1} and A Dubitski\textsuperscript{2}

\textsuperscript{1} Faculty of Science, University of Ontario Institute of Technology, Oshawa, ON, Canada, L1H 7K4
E-mail: \textsuperscript{1}\{Dhavide.Aruliah,Lennaert.vanVeen\}@uoit.ca, \textsuperscript{2}alex.dubitski@gmail.com

Abstract. Pseudo-arclength continuation is a well-established method for constructing a numerical curve comprising solutions of a system of nonlinear equations. In many complicated high-dimensional systems, the corrector steps within pseudo-arclength continuation are extremely costly to compute; as a result, the step-length of the preceding prediction step must be adapted carefully to avoid prohibitively many failed steps. We describe the essence of a parallel method for adapting the step-length of pseudo-arclength continuation. Our method employs several predictor-corrector sequences with differing step-lengths running concurrently on distinct processors. Our parallel framework permits intermediate results of correction sequences that have not yet converged to seed new predictor-corrector sequences with various step-lengths; the goal is to amortize the cost of corrector steps to make further progress along the underlying numerical curve. Results from numerical experiments suggest a three-fold speed-up is attainable when the continuation curve sought has great topological complexity and the corrector steps require significant processor time.

1. Introduction

Continuation (or homotopy) problems arise naturally in numerous application domains. For instance, in numerical studies of turbulent fluid flow, the Reynolds number Re can be treated as a scalar that parametrizes a family of discretized versions of the Navier-Stokes equations. Certain computations—for instance, computations of time-periodic solutions in fluid flow—are significantly less challenging at low Reynolds numbers than at high Reynolds numbers, where dynamical processes occur on a larger range of spatial scales. Continuation can be used to extend the results obtained at some low Reynolds number into the physically more interesting regime, in other words to construct a homotopy between turbulent flows with very different characteristics. Homotopies are also used to relate flows in different geometries and study their similarities and differences (see [1] and references therein).

In mathematical terms, continuation problems are nonlinear systems of equations where the number of equations is one fewer than the number of variables. That is, given a mapping $F : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$, the equation

$$F(x, \lambda) = 0$$

(1)

with vector $x \in \mathbb{R}^n$ and scalar $\lambda \in \mathbb{R}$ defines a continuation problem. In geometric terms, (1) defines a one-dimensional implicit curve in $\mathbb{R}^{n+1}$ under suitable smoothness and consistency properties of the mapping $F$. 

Published under licence by IOP Publishing Ltd
The essential idea of continuation (sometimes called homotopy) is to follow the underlying curve of solutions. Specifically, we assume that the parameter $\lambda$ lies in some interval $[\lambda_{\text{initial}}, \lambda_{\text{final}}] \subseteq \mathbb{R}$. Often, the problem can be formulated in such a way, that solving (1) for $x$ is easy when $\lambda = \lambda_{\text{initial}}$, and solving (1) for $x$ is challenging when $\lambda = \lambda_{\text{final}}$. Thus, continuation is the process of gradually morphing the solution of a straightforward problem into the solution of a formidable problem in small increments.

Numerical continuation refers to families of numerical algorithms for generating points on a homotopy curve. Natural continuation and pseudo-arclength continuation are examples of predictor-corrector methods [2, 3, 4]. Predictor-corrector algorithms are inherently sequential: one solution point is used to predict the next solution point on the curve. Parallelization can be introduced within the corrector steps, but the predictor steps need to be computed in sequence.

More importantly, predictor-corrector methods rely on adaptive step-size selection to make optimal progress in moving along the curve of solutions. Strategies for adapting step-sizes selection are largely heuristic: when corrector steps fail to converge, we reduce the step-size, we reject the predictor step, and we start again. This turns out to be the principal bottleneck in many continuation problems, as the computation time devoted to nonconvergent sequences of corrector steps is wasted. Thus, identifying strategies that can improve the performance of predictor-corrector schemes by reducing the cost of failed predictor steps is a significant challenge in modern High-Performance Computing.

We describe in the present work a framework for parallelizing predictor-corrector methods to amortize the cost of failed predictor steps. Specifically, we carry out several predictor steps of different step-sizes for pseudo-arclength continuation in parallel on distinct processors. Our algorithm manages the different steps in parallel using a colored rooted tree. The principal context within which this approach is practical is when users have a serial application code that takes a long time to evaluate corrector steps. If the underlying numerical curve is sufficiently complex that predictor steps fail often, our parallel adaptive strategy offers some hope of progress.

2. Background and motivation
Before describing our parallel adaptive algorithm, we briefly review previous numerical continuation algorithms—notably natural parameter and pseudo-arclength continuation—for solving problems of the form (1), where $\lambda \in [\lambda_{\text{initial}}, \lambda_{\text{final}}] \subseteq \mathbb{R}$, $x \in \mathbb{R}^n$, and $F : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$. In doing so, we sometimes write the underlying equation (1) in the form

$$F(z) = 0, \quad \text{where } z = (x, \lambda) \in \mathbb{R}^{n+1}. \quad (2)$$

That is, we treat the concatenation of the $n$-vector $x$ and the scalar $\lambda$ as a single $(n + 1)$-vector $z$ while using the same symbol $F$ to denote the mapping. Also, the appropriately-sized matrices representing Jacobian derivatives are denoted by

$$F_\lambda \equiv \frac{\partial F}{\partial \lambda} \in \mathbb{R}^{n \times 1}, \quad F_x \equiv \frac{\partial F}{\partial x} \in \mathbb{R}^{n \times n}, \quad \text{and} \quad F_z \equiv \frac{\partial F}{\partial z} \in \mathbb{R}^{n \times (n+1)} \quad \text{respectively.} \quad (3)$$

An obvious method for constructing a curve of numerical solutions of (1) is to increment the parameter $\lambda$ gradually from $\lambda_{\text{initial}}$ to $\lambda_{\text{final}}$. That is, given a point $(x, \lambda) \in \mathbb{R}^{n+1}$ known to lie on the curve, a new point on the curve $(\xi, \lambda + h)$ is found

(i) by incrementing $\lambda$ by a small amount $h > 0$; and
(ii) by solving $F(\xi, \lambda + h) = 0$ for the unknown $\xi$.

This approach is referred to as natural parameter continuation (or, more simply, natural continuation, see Algorithm 1) [2, 3, 4]. Fixing the known parameter value $\lambda + h$ in the
underdetermined nonlinear system $F(\xi, \lambda + h) = 0$ yields a closed $n \times n$ system in the $n$ unknown components of $\xi$. Intuitively, if $h$ is sufficiently small, the vector $\xi$ should be easy to obtain using a quadratically-convergent Newton iteration (see, e.g., [5]).

**Algorithm 1:** Natural parameter continuation.

Input: $[\lambda_{\text{initial}}, \lambda_{\text{final}}] \subseteq \mathbb{R}$; vector $x^{(0)} \in \mathbb{R}^n$ where $F(x^{(0)}, \lambda_{\text{initial}}) = 0$; step-size $h > 0$

Output: vector $x^{(k)} \in \mathbb{R}^n$ and scalar $\lambda^{(k)} \geq \lambda_{\text{final}}$ such that $F(x^{(k)}, \lambda^{(k)}) = 0$

1. $k \leftarrow 0$
2. $\lambda^{(0)} \leftarrow \lambda_{\text{initial}}$ // initialization
3. while $\lambda^{(k)} < \lambda_{\text{final}}$ do // loop to generate successive points on curve
4.  
5.  
6. if iteration in line 5 converges to $\xi$ then // accept next point on curve
7.  
8.  
9. else // reject predictor step & repeat
10.  
11.  
12. return $x^{(k)}, \lambda^{(k)}$

Natural continuation is conceptually simple and easy to implement; however, it breaks down when the curve implicitly defined by (1) admits a fold point (i.e., a point $(x, \lambda)$ where the Jacobian matrix $F_x(x, \lambda)$ is singular) somewhere along the curve (see [6] for an alternative characterization of fold points). At a fold point, systematically incrementing $\lambda$ as in Algorithm 1 yields an inconsistent system of nonlinear equations that cannot be solved regardless of how small $h$ is. Fold points do occur in practical continuation problems, so other continuation strategies need to be devised [7, 8].

*Pseudo-arclength continuation* (as outlined in Algorithm 2) is a standard approach to circumvent fold point singularities that capitalizes on the smoothness of the underlying curve [2, 6, 9]. Under the assumption that both $x$ and $\lambda$ are smooth functions of arclength, pseudo-arclength continuation constitutes using the unit direction $T \in \mathbb{R}^{n+1}$ tangent to the curve for prediction. The term “pseudo-arclength” applies because the step-size $h$ is a close approximation of the Euclidean distance (in $\mathbb{R}^{n+1}$) between successive points on the numerical curve (which, in turn, is almost the arclength between the two points as measured along the curve). The tangent direction $T$ can be determined by computing a null vector of the $n \times (n+1)$ Jacobian matrix $F(x, \lambda)$ or by computing finite differences between successive points on the curve. The underdetermined nonlinear system of equations (1) is then closed by requiring that the solution $(x, \lambda)$ sought must lie in the hyperplane orthogonal to the tangent direction $T$ at distance $h$ from the last known point; see line 7 of Algorithm 2.

Natural continuation and pseudo-arclength continuation both fit into a broader framework of *predictor-corrector methods* [2]. Predictor-corrector methods involve three important components:

- a *predictor step* of a prescribed step-size;
- a sequence of iterative *corrector steps*; and
- an adaptive step-size selection strategy.
The most expensive part of the algorithm is computation of the corrector steps (because each corrector step usually involves solving a linear system of equations).

Rejected predictor steps are extremely expensive (because numerous corrector steps are computed before deciding to reject the predictor).

Rejected predictor steps may not be prohibitively expensive for moderate sized systems amenable to dense, direct linear algebra solvers. However, as systems grow in size, the use of direct solvers to compute corrector steps becomes infeasible. For instance, Krylov subspace iterations can be applied within Newton iterations to determine the corrector steps within a continuation algorithm (a process referred to as Newton-Krylov continuation [10, 11]). In such large-scale problems, individual corrector steps can require hours of computation and the penalty incurred

\begin{algorithm}
\textbf{Algorithm 2:} Pseudo-arclength continuation.

\textbf{Input}: [$\lambda_{\text{initial}}, \lambda_{\text{final}}$] $\subseteq \mathbb{R}$; vector $x^{(0)} \in \mathbb{R}^n$ where $F(x^{(0)}, \lambda_{\text{initial}}) = 0$; step-size $h > 0$

\textbf{Output}: vector $x^{(k)} \in \mathbb{R}^n$ and scalar $\lambda^{(k)} \geq \lambda_{\text{final}}$ such that $F(x^{(k)}, \lambda^{(k)}) = 0$

1. $k \leftarrow 0$
2. $\lambda^{(0)} \leftarrow \lambda_{\text{initial}}$ // initialization
3. $z^{(0)} \leftarrow (x^{(0)}, \lambda^{(0)}) \in \mathbb{R}^{n+1}$
4. Determine approximate tangent vector $T^{(0)} \in \mathbb{R}^{n+1}$ at $z^{(0)}$
5. while $\lambda^{(k)} < \lambda_{\text{final}}$ do // loop to generate successive points on curve
6. \hspace{1em} $w \leftarrow z^{(k)} + h T^{(k)}$ // predictor step
7. \hspace{1em} Iteratively solve $(n + 1) \times (n + 1)$ nonlinear system of equations
8. \hspace{2em} $F(\zeta) = 0,$
9. \hspace{2em} $T^{(k)T} (\zeta - z^{(k)}) = h$
10. to obtain $\zeta \in \mathbb{R}^{n+1}$ starting from initial iterate $\zeta^{(0)} = w$ // corrector steps
11. if iteration in line 7 converges to $\zeta$ then // accept next point on curve
12. \hspace{2em} $x^{(k+1)} \leftarrow \zeta_{1:n}$ // extract subvector from $\zeta$
13. \hspace{2em} $\lambda^{(k+1)} \leftarrow \zeta_{n+1}$ // extract last element from $\zeta$
14. \hspace{2em} $z^{(k+1)} \leftarrow (x^{(k+1)}, \lambda^{(k+1)})$
15. \hspace{2em} Determine approximate tangent vector $T^{(k+1)} \in \mathbb{R}^{n+1}$ at $z^{(k+1)}$
16. \hspace{2em} Heuristically adjust the step-size $h$
17. \hspace{2em} $k \leftarrow k + 1$
18. else // reject predictor step $w$ and repeat
19. \hspace{2em} Reduce step-size $h$
20. \hspace{1em} return $z^{(k)} = (x^{(k)}, \lambda^{(k)})$
\end{algorithm}
for failed predictor steps is punitive. Given that using too large a step-size can result in failed predictor steps, an obvious strategy might be to use very small step-sizes. Unfortunately, using too small a step-size impedes progress along the curve from $\lambda_{\text{initial}}$ to $\lambda_{\text{final}}$ (in both natural and pseudo-arclength continuation). Efficient step-size adaptivity involves trading off between these conflicting concerns while minimizing wasted computation.

Thus, we develop a parallel adaptive method for step-size selection within the predictor-corrector framework of pseudo-arclength continuation. The method we present is independent of the strategy used for the corrections steps and is designed for problems in which the correction steps are particularly expensive.

3. Outline of the parallel adaptive algorithm

There are two essential ways to achieve parallelism in the selection of step-sizes. The first is most obvious: to use a sequence of different step-sizes $t_1, t_2, \ldots, t_W$ for some positive scalars $\{t_k\}_{k=1}^W$. That is, given $W$ processors, an initial point $z \in \mathbb{R}^{n+1}$ known to satisfy $F(z) = 0$, and a unit tangent direction $T \in \mathbb{R}^{n+1}$, each processor $\alpha$ computes a predictor step $\zeta^{(\alpha,0)} = z + t_0 h_\alpha T^{(\alpha)}$ and uses that prediction to seed a sequence of corrector steps $\{\zeta^{(\alpha,0)}, \zeta^{(\alpha,1)}, \zeta^{(\alpha,2)}, \ldots\}$. Since the correction steps are independent of each other, no communication is required between processors except in generating the predictions. Each processor $\alpha$ keeps track of its own iteration counter $\nu_\alpha$ and computes its own nonlinear residuals.

A second strategy to achieve parallelism in adapting step-sizes is to use intermediate computations to seed new predictions. That is, suppose processor $\alpha$ starts from the predictor $\zeta^{(\alpha,0)} = z^{(\alpha)} + h_\alpha T^{(\alpha)}$ and proceeds to compute a sequence of correction steps $\{\zeta^{(\alpha,0)}, \zeta^{(\alpha,1)}, \zeta^{(\alpha,2)}, \ldots\}$. Rather than waiting for the corrector iterations to converge, the intermediate iterates can be used to compute a normalized secant direction $\hat{T}^{(\alpha)}$ in the direction of $\zeta^{(\alpha,k)} - z^{(\alpha)}$. Assuming the corrector iterates are sufficiently close to the curve of solutions, the new secant direction $\hat{T}^{(\alpha)}$ approximates a tangent to the curve and can be handed off to another processor $\beta$ to generate a new predictor point $z^{(\beta)} + h_\beta T^{(\beta)}$, where $z^{(\beta)} = \zeta^{(\alpha,k)}$ for some iterate $k$, $h_\beta = t_\ell h_\alpha$ for some $\ell \in \{1, \ldots, W\}$, and $T^{(\beta)} = \hat{T}^{(\alpha)}$.

To manage both these strategies, each processor in use by a node in a rooted tree with the master processor at the root. We assume that the tree has a width $W > 0$ (which corresponds to the scalars $t_1$, $t_2$, $\ldots$, $t_W$ that multiply the step-length $h$) and a depth $D$ (which corresponds to the number of extrapolations to carry out from intermediate calculations. Each processor has certain data associated with it and processors are allocated to the maximum width and depth of the tree.

Specifically, processor $\alpha$ is associated with a base point $z^{(\alpha)} \in \mathbb{R}^{n+1}$, a step-size $h_\alpha$, a unit tangent direction $T^{(\alpha)}$, a current iteration counter $\nu_\alpha$, a current iterate $\zeta^{(\alpha,\nu_\alpha)} \in \mathbb{R}^{n+1}$, a nonlinear residual $r^{(\alpha,\nu_\alpha)} \in \mathbb{R}^n$, and a color $c_\alpha$ where $c_\alpha \in \{\text{GREEN}, \text{YELLOW}, \text{RED}, \text{BLACK}\}$. In the preceding list, the nonlinear residual satisfies

$$ r^{(\alpha,\nu_\alpha)} = F \left( \zeta^{(\nu_\alpha)}; z_\alpha, T_\alpha, h_\alpha \right), \quad \text{where} \quad G(\zeta; z, T, h) := \begin{bmatrix} F(\zeta) \\ T^T(\zeta - z) - h \end{bmatrix}. $$

The color of each node $\alpha$ is assigned using the 2-norm of the residual at the current corrector step, i.e., $\left\| r^{(\alpha,\nu_\alpha)} \right\|_2$. Given a positive tolerance TOL, the node $\alpha$ is assigned a color $c_\alpha$ as follows:

- $c_\alpha = \text{GREEN}$ if the current iterate $\zeta^{(\alpha,\nu_\alpha)}$ is deemed to be sufficiently near to the curve of solutions, i.e., $\left\| r^{(\alpha,\nu_\alpha)} \right\|_2 \leq \text{TOL}$;
- $c_\alpha = \text{YELLOW}$ if the next iterate $\zeta^{(\alpha,\nu_\alpha+1)}$ is expected to be sufficiently near to the curve of solutions, i.e., $\log \left\| r^{(\alpha,\nu_\alpha)} \right\|_2 \leq \frac{1}{2} \log(\text{TOL})$;
• $c_\alpha = \text{BLACK}$ if the iteration sequence is deemed to have diverged; and
• $c_\alpha = \text{RED}$ otherwise.

The specific criterion for detecting divergence of a sequence of corrector iterates depends on another user-specified parameter due to problem-specific challenges in inferring divergence. The criterion for coloring a node yellow is based on the assumption that the corrector step converges quadratically. In case the corrector steps are convergent of a different order—for instance, linearly convergent fixed-point iterations—this criterion can readily be changed. This might also be necessary for strongly nonlinear problems or for systems with a lot of numerical noise; in either of these cases, the quadratic convergence of Newton iteration is realized only very close to the solution. New processes are assigned the color red by default until the nonlinear residual is computed and the color can be reassigned.

With the preceding ideas in mind, it is possible to describe the underlying parallel adaptive algorithm. Algorithm 3 is a high level summary of the essential steps. Notice that the details of tree traversal and the precise heuristic for pruning redundant subtrees are not included in the high level algorithm description; the former is a standard recursive procedure and the latter involves comparing pseudo-arclengths of paths from the root node to the leaves. The precise details will be described in a forthcoming paper to accompany a public release of the software.

**ALGORITHM 3:** Essential parallel adaptive algorithm

**Input:** $[\lambda_{\text{initial}}, \lambda_{\text{final}}] \subseteq \mathbb{R}$; vector $x^{(0)} \in \mathbb{R}^n$ where $F(x^{(0)}, \lambda_{\text{initial}}) = 0$; step-size $h > 0$; tangent direction $T^{(0)} \in \mathbb{R}^{n+1}$ where $\|T^{(0)}\|_2 = 1$; tolerance TOL; depth $D > 0$; width $W > 0$; positive scaling parameters $\{t_1, t_2, \ldots, t_W\}$

**Output:** vector $x^{(k)} \in \mathbb{R}^n$ and scalar $\lambda^{(k)} \geq \lambda_{\text{final}}$ such that $F(x^{(k)}, \lambda^{(k)}) = 0$

1. Seed root node $\alpha$ with data $(z^{(\alpha)}, T^{(\alpha)}, h_\alpha) \leftarrow ((x^{(0)}, \lambda_{\text{initial}}), T^{(0)}, h)$ // initialization

2. repeat // loop to generate successive points on curve

   foreach leaf node $\alpha$ do

   if depth of $\alpha < D$ then

   $\hat{T}^{(\alpha)} \leftarrow \|\zeta^{(\alpha, v_\alpha)} - z^{(\alpha)}\|_2^{-1} (\zeta^{(\alpha, v_\alpha)} - z^{(\alpha)})$ // secant direction

   for $\ell = 1 : W$ do

   if processors are available then

   Assign processor $\beta_\ell$ the data $(z^{(\beta_\ell)}, T^{(\beta_\ell)}, h_{\beta_\ell}, \nu_{\beta_\ell}, \zeta^{(\beta_\ell, v_{\beta_\ell})}, c_{\beta_\ell})$ according to

   $z^{(\beta_\ell)} \leftarrow z^{(\alpha)}, T^{(\beta_\ell)} \leftarrow \hat{T}^{(\alpha)}, h_{\beta_\ell} \leftarrow t_\ell h_\alpha, \nu_{\beta_\ell} \leftarrow 0,$

   $\zeta^{(\beta_\ell, v_{\beta_\ell})} \leftarrow \sum_{k} \zeta^{(\alpha)} + t_\ell h_\alpha \hat{T}^{(\alpha)}, c_{\beta_\ell} \leftarrow \text{RED}$

7. Compute single corrector steps concurrently on all RED and YELLOW nodes of the tree.

8. Traverse nodes of tree updating residuals

9. $r^{(\alpha, v_\alpha)} \leftarrow F(\zeta^{(v_\alpha)}; z_\alpha, T_\alpha, h_\alpha)$ and updating colors $c_\alpha$ accordingly.

10. Prune the tree, deleting BLACK nodes and eliminating redundant subtrees.

11. until tree consists of solitary node $\alpha$ with $\lambda^{(\alpha)} \leq \lambda_{\text{final}}$

12.
4. Test case: a modified Kuramoto-Sivashinsky equation

To illustrate the application of Algorithm 3 in practice, we compute a family of solutions of a modified Kuramoto-Sivashinsky equation. Specifically, we consider the PDE

\begin{align}
    u_t + uu_x + \nu u_{xxxx} - A \sin(u) &= 0, \quad x \in (0, 2\pi), \; t \geq 0, \tag{5a} \\
    u^{(k)}(0, t) &= u^{(k)}(2\pi, t), \quad k \in \{0, 1, 2, 3, 4\}, \tag{5b}
\end{align}

where the solutions \( u = u(x, t) \) sought are expected to be at least 4 times continually differentiable on the periodic one-dimensional domain \((0, 2\pi)\). This equation has a reflection symmetry given by \( S : (u, x) \mapsto (-u, -x) \) and a continuous translational symmetry. We fix \( A = 8 \) and consider \( \nu > 0 \) as homotopy parameter.

For all values of the control parameter \( \nu \), there are trivial solutions at \( u = q\pi \) for \( q \in \mathbb{Z} \). The equilibria with even \( q \) are unstable and those with odd \( q \) are stable for \( \nu < 1/(4A) \). Beyond that point, stable, spatially varying equilibrium solutions arise. From these equilibria, several branches of travelling wave solutions bifurcate. One such branch is shown in Fig. 1. This branch has a number of fold points and the travelling wave changes its shape rapidly along the branch. Therefore, the traditional pseudo-arclength continuation approach will suffer from many failed steps. Moreover, the wave becomes increasingly localized for small values of the control parameter. In order to resolve it correctly, we need to use fine spatial discretization which will result in time-consuming corrector steps.

We compute the travelling waves as \( u(x, t) = w(x - ct) \), which results in the following Boundary Value Problem (BVP):

\[-cw' + w'' + w'' + \nu w''' - A \sin(w) = 0; \quad w(0) = w(2\pi).\]

The linear terms in this equation are efficiently computed in Fourier space, while the nonlinear terms are best computed on a regular, periodic grid. Thus, we approximate solutions as

\[w_j = \sum_{k=0}^{N-1} a_k e^{ikx_j}, \quad x_j = \frac{2\pi j}{N}, \; j = 0, \ldots, N - 1\]

and switch between the representations \( \{w_j\} \) and \( \{a_k\} \) by the discrete Fourier transform, implemented using the FFTW library. The computational cost of each transform is \( O(N \ln N) \). The resolution is set by the parameter \( N \), fixed to \( N = 2048 \) to resolve the solutions shown in Fig. 1 and avoid aliasing issues.

The continuation problem now takes the form \( F(a_0, \ldots, a_{N-1}, c, \nu) = 0 \), where \( F \) and its derivatives are evaluated using the pseudo-spectral method outlined above. Since we have an extra unknown, the wave speed \( c \), we must add an extra equation to ensure uniqueness of the corrector steps. We impose that the Newton update steps be orthogonal to \( w_x \), the generator of the symmetry group of translations. This condition ensures that the successive iterates under corrector steps do not slide along the \( x \)-direction. Because of the second nonlinear term in Eq. 5, the Jacobian matrix \( F_x \) is dense. We employ a direct solver to find the Newton corrector steps, so that the computational complexity of each corrector step is \( O(N^3) \).

The wall-clock time for the computation of the entire curve shown in Fig. 1 using various tree structures and step size distributions is shown in Fig. 2. The maximal speed-up is about a factor of three, obtained with a tree depth and width of three. In that computation, there is a lot of redundancy as many processors will be working on very similar approximate solutions. A speed-up of more than a factor of two, however, can be obtained using as few as three processors with the same step size multiplication factor.
Figure 1. The branch of travelling wave solutions to Eq.5 used for testing the parallel continuation algorithm. Shown is the wave speed, $c$, versus the control parameter, $\nu$. The inlays show snap shots of the solution $u(x,t)$ at four points along the continuation curve. At $c = 0$, the wave bifurcates from an equilibrium, which is symmetric under the reflection symmetry. For $\nu \lesssim 0.01$ the solution becomes strongly localized.

5. Summary
The algorithm described here is designed with nonlinear continuation problems in mind where the relative cost of failed corrector steps is high and the cost of processors is low. The situation we have in mind is where a user might have a serial code and several cores on their desktop workstation with no obvious means of parallelization. The method developed here does not exhibit good scaling as desired in High-Performance Computing applications employing hundreds or thousands of processors. However, given that most machines now have four to eight cores, a user with minimal HPC experience might employ the strategies described here to make some improvements on the serial performance of their codes. A more detailed paper and a public release of the software is currently under development.

References
[1] Kawahara G, Uhlmann M and van Veen L 2012 Ann. Rev. Fluid Mech. 44 203–225
[2] Allgower E L and Georg K 2003 Introduction to Numerical Continuation Methods (Classics in Applied Mathematics vol 45) (Philadelphia: SIAM)
[3] Govaerts W 2000 Numerical Methods for Bifurcations of Dynamic Equilibria (Philadelphia: SIAM)
[4] Kuznetsov Y 1998 Elements of Applied Bifurcation Theory (New York: Springer-Verlag)
[5] Kelley C 2003 Solving Nonlinear Equations with Newton’s Method Fundamentals of Algorithms (Philadelphia: SIAM)
[6] Dickson K, Kelley C, Ipsen I and Kevrekidis I 2007 SIAM J. Numer. Anal. 45 263–276
Figure 2. Wall-clock time for the computation of the continuation curve shown in Fig. 1 using a tree width and depth up to three. The numbers between brackets denote the multipliers for the step size, $\{t_i\}_i=1^W$, and the integer denotes the number of CPUs concurrently processing corrector steps.

[7] Yang Z H and Keller H 1986 SIAM J. Sci. Stat. Comput. 7 351–361
[8] Doedel E J 2007 Numerical Continuation Methods for Dynamical Systems ed Krauskopf B, Osinga H M and Galán-Vioque J (Springer Netherlands)
[9] Keller H 1977 Applications of Bifurcation Theory ed Rabinowitz P (New York: Academic Press) pp 359–384
[10] Sánchez J, Net M, García Archilla B and Simó C 2004 J. Comp. Phys. 201 13–33
[11] Knoll D and Keyes D 2004 J. Comp. Phys. 193 357–397