Towards a full higher order AD-based continuation and bifurcation framework

Isabelle Charpentier, Bruno Cochelin

To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–19

ARTICLE

Towards a full higher-order AD-based continuation and bifurcation framework

I. Charpentier[a]* and Bruno Cochelin[b]

[a]*ICube, UMR 7357, CNRS and University of Strasbourg, France;*[b]*LMA, UPR CNRS 7051, Marseille, France*

(*v1.0 released Month YYYY*)

Some of the theoretical aspects of continuation and bifurcation methods devoted to the solution for nonlinear parametric systems are presented in a higher-order automatic differentiation (HOAD) framework. Besides benefits in terms of generality and ease of use, HOAD is used to assess fold and simple bifurcations points. In particular, the formation of a geometric series in successive Taylor coefficients allows for the implementation of an efficient detection and branch switching method at simple bifurcation points.

Some comparisons with the Auto and MatCont continuation software are proposed. Strengths are then exemplified on a classical case study in structural mechanics.

Keywords: nonlinear parametric problems; continuation method; bifurcation analysis; geometric series; higher order automatic differentiation; Diamant

AMS Subject Classification: 41A58; 65P30; 68N19; 70E60

1. Introduction

Numerical continuation and bifurcation analysis are classical tools for the study of nonlinear parametric equations, see [29] and the references therein for instance. In general terms, these methods allow for the solution of nonlinear algebraic systems of n equations,

$$\mathcal{R}(U) = \mathcal{R}(u, \lambda) = 0, \tag{1}$$

where the unknown $U = (u, \lambda)$ of the nonlinear function $\mathcal{R}$ comprises the state vector u of dimension n and some scalar control parameter λ to be varied. Frequently, this parameter has a physical meaning, representing a load parameter in structural mechanics or a volumetric flow rate in fluid mechanics, for instance. The solutions of the underdetermined problem (1) are one-dimensional continua, referred to as solution branches, that may contain singular points such as folds (limit points), and bifurcation points where two branches intersect. Note that a number of additional bifurcations may arise in the solution of nonlinear parametric ordinary differential equations [13, 17, 28]. Such singular points indicate a qualitative change in the behavior of the system under study. They can be detected and classified from determinant calculations and/or derivative computations, see for instance [20, 21, 29].

*Corresponding author. Email: icharpentier@unistra.fr

During the last thirty years, general-purpose software, free and commercial ones, have been proposed to engineers and scientists to draw bifurcation diagrams without embarking into the heavy task of programming their own continuation algorithm. Most of these continuation software, see MatCont [13] and Auto [17] for instance, follow the first order predictor-corrector principles described in [26, 29]. Besides, under analyticity assumptions, solution branches may be approximated as higher-order truncated Taylor expansions [4, 9, 10] to obtain continuous solutions and representations. Their range of validity is estimated *a posteriori* from the remainder of the series to provide large adaptive stepsize.

Today, Taylor-based nonlinear solvers take full benefit of higher-order automatic differentiation (HOAD), notably in terms of generality, efficiency and automation, see [4] and references therein. Our HOAD continuation framework, namely Diamant, moreover agrees with homotopy for the solution of nonlinear eigenvalue problems and with sensitivity analysis for both continuation and homotopy methods [6]. It has been recently adapted to the determination of singular boundaries [25] of particular piecewise smooth nonlinear systems by using the extended system of equations introduced in [28].

Higher order information can be used to locate singular points [18, 20, 21] and AD can be used to extract it [13, 24]. In particular, simple folds can be detected in a series in straightforward manner. As discussed for higher-order perturbation methods [18], the formal power series arising near to a singularity reveals the analytical structure of the solution. It can be used to "improve the utility of the series" and the convergence of the numerical method. This has been applied to specific quadratic formulations of nonlinear parametric problems near to simple bifurcation points [11]. In this paper, we take advantage of HOAD to rework this proposal in the general framework of (1) in order to implement an efficient detection and branch switching method.

Two tutorial examples from MatCont [27] and Auto [15, 17] are used to illustrate some strengths and limitations of the Matlab implementation of Diamant [7] in comparison with these two classical first order continuation software. Then, the Elastica beam problem is used to exemplify our higher-order continuation framework on a classical partial differential equation (PDE) problem, relevant for numerical bifurcation analysis [11, 14, 19] and structural mechanics.

The layout of the paper is as follows. Fundamentals of the higher-order continuation tool Diamant are briefly recalled in Section 2. The higher-order bifurcation analysis is presented in Section 3, then illustrated by several case studies in Section 4. Section 5 provides a summary.

2. Higher order continuation framework

Numerical continuation is a method devoted to the solution for the system of nonlinear parametric equations (1). The solutions of such an under-determined problem are one-dimensional continua, referred to as solution branches, that may intersect at bifurcation points. The pseudo-arc length continuation algorithm [26] is a classical answer to the calculation of solution branches. The branches $U(a) = (u(a), \lambda(a))$ are locally parameterized as

$$(U(a) - U(0))^* . \left(\frac{\partial U}{\partial a} \right) - a = 0, \tag{2}$$

where * is the transpose operator and $\partial U/\partial a$ is the normalized direction vector.

Under analyticity assumptions, the solution branches may be approximated as a collection of continuous truncated Taylor expansions [4, 10],

$$U(a) = \left(\sum_{k=0}^{K} \frac{a^k}{k!} \frac{\partial^k u}{\partial a^k}(0), \sum_{k=0}^{K} \frac{a^k}{k!} \frac{\partial^k \lambda}{\partial a^k}(0) \right) = \sum_{k=0}^{K} a^k (u_k, \lambda_k) = \sum_{k=0}^{K} a^k U_k, \forall a \in (0, a_m), \quad (3)$$

where K is the truncation order of the series (typically $K = 20$), $U_k = (u_k, \lambda_k)$ are the unknown Taylor coefficients of $U(0)$ for $k = 0, .., K$ and a_m is the range of validity for the series. As in other higher-order solvers (see [4] and the references therein), this series is introduced into (1) to deduce a general iterative sequence of equations satisfied by the Taylor coefficients U_k, that is

$$(\mathcal{R} \circ U)_k = \{\mathcal{R}_1\} U_k + \{(\mathcal{R} \circ U)_k | U_k = 0\} = 0, \quad \text{for } k = 1, .., K. \quad (4)$$

Therein, the Jacobian $\{\mathcal{R}_1\}$ of $\mathcal{R}$ is constructed once. The higher-order term $\{(\mathcal{R} \circ U)_k | U_k = 0\}$ is the Taylor coefficient at order k of $(\mathcal{R} \circ U)$ evaluated with a null value for the unknown U_k. All these derivatives may be computed using HOAD.

The solution for (1)–(2) then relies on the sequence of linear systems

$$\begin{pmatrix} \{\mathcal{R}_1\} \\ U_1^* \end{pmatrix} . U_k = \begin{pmatrix} -\{(\mathcal{R} \circ U)_k | U_k = 0\} \\ \delta_{1k} \end{pmatrix}, \quad \text{for } k = 1, .., K, \quad (5)$$

where δ_{1k} is the Kronecker's delta. The left-hand side matrix of (5) is factorized once. The range of validity a_m is deduced from the remainder of the series as

$$a_m = \left(\frac{\varepsilon}{|| \{(\mathcal{R} \circ U)_k | U_k = 0\} ||} \right)^{1/K}, \quad (6)$$

to provide large adaptive stepsize. The threshold ε is typically set to 10^{-6}. This higher-order continuation method may often be used without Newton corrections.

Listing 1 Nonlinear solver for the computation of one branch of solutions

```
% Given a solution point U0, a tangent U_t, a nonlinear function R
Compute the Jacobian {R_1}
Form the left hand side matrix of (5) and factorize it
Set U_0 = U0; U_1 = U_t
for k=2:K
     Evaluate R(U) at order k using U_k = 0
     Solve (5)
     Set U_k
end
Compute a_m following (6)
% Room for HOAD simple bifurcation detection, see List. 4
```

The iterative sequence (5) is the basis for the general high level nonlinear solver known as Diamant. Its HOAD is similar to the HOAD described in other Taylor-based nonlinear solvers. Thanks to AD generality, this solver is able to deal with any user-defined

analytical nonlinear residual problem satisfying (1). The series information – Taylor coefficients, tangent to the branch and range of validity – are stored into checkpoints [3, 23] managed through the continuation driver described in List. 2. Singular point information discussed at Section 3 – fold and bifurcation loci, tangent to the second branch, series and related range of validity – is accounted in the checkpoint where relevant. This offers the opportunity to start a continuation at a bifurcation point in the direction of the second tangent, see List. 5.

Listing 2 Continuation driver

```
% Given a point U0, a tangent U_t, a nonlinear function R
% Given a number of forward steps (Nbforward)
for i=1:Nbforward
     % Newton–Raphson corrections (NR)
     if (|R(U_0)|>NR_threshold), NR corrections; end
     % Compute branch i
     if (regular point)  % initial point or regular checkpoint
          Compute the series U and a_m using List. 1
          Create a checkpoint
               % starting point U_0, tangent, series U,
               % validity range a_m, end point U(a_m) and tangent
               % bifurcation information from List. 4
          Check for a fold using List. 3
          Store the checkpoint
          U_0 = U(a_m) and U_t = U'(a_m)
     else
          % branching at a bifurcation point
          Compute the series U^b using List 5
          Create a checkpoint
               % starting point U^b(a_m^b) and second tangent
     end
end
```

Previous Diamant versions did not provide a higher-order bifurcation analysis to improve the utility of the series near to a bifurcation point. This significant shortcoming is addressed by reworking the bifurcation strategies presented in [11, 15] to propose a general higher-order AD implementation of bifurcation detection, location and branch switching.

3. Bifurcation analysis

Near to a singular point, a very small theoretical or numerical perturbation of the solution induces a qualitative change in the behavior of the residual function under study. Such a change can be monitored and classified from determinant calculations and/or derivative computations, see for instance [20, 21, 29]. Among the numerous kind of bifurcations, this paper consider fold points (or snap through in structural mechanics) and simple bifurcation points (or buckling) that are of special interest in PDE problems studied in structural engineering.

As an illustration, we consider the tutorial example of MatCont [27],

$$\begin{cases} 0 = -2u_1 + u_2 + \alpha \exp(u_1), \\ 0 = u_1 - 2u_2 + \alpha \exp(u_2). \end{cases} \tag{7}$$

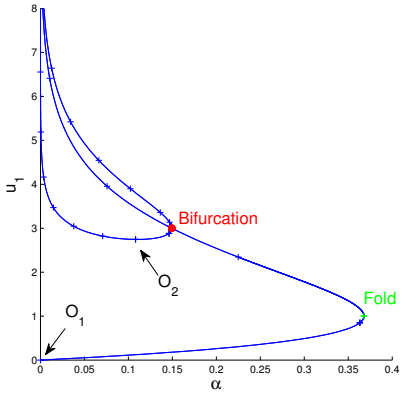

Figure 1. Tutorial bifurcation diagram using Diamanalab.

where the variables u_1 and u_2 denote the components of the unknown state vector u, and α is the parameter to be varied. A trivial solution is $O_1 = (u_1, u_2, \alpha) = (0, 0, 0)$. The branch issued from O_1 satisfies $\alpha \exp(u_1) - u_1 = 0$ and $u_1 = u_2$. It exhibits a simple fold at (1,1/ exp(1)) and a simple bifurcation point at (3,3,3/ exp(3)).
The bifurcation diagram computed using the Matlab implementation of Diamant, namely Diamanlab, is plotted in Fig. 1. The branch issued from O_1 requires the computation of 5 checkpoints only to run from $u_1 = 0$ to $u_1 = 8$ with large continuation steps. The fold $(0.999998, 0.999998, 0.367879)$, indicated with a green +, is identified from the series computed at the second checkpoint ($a_m = 2.1068$ at checkpoint 2). The simple bifurcation point, indicated with a red disc, is located at $(3.000001, 3.000001, 0.149361)$. As expected, the accuracy in the results agrees with the threshold of the method. The branching method (List. 5) allows to compute a series (with a range of validity $a_m = 0.813490$), then the solution point O_2 on the second branch, far from the bifurcation point. The second branch is traveled forward and backward.

Fold and bifurcation detection as well as branch switching methods are described in subsections 3.1, 3.2 and 3.3, respectively.

3.1 Fold detection and location

As recalled in [15], a solution $U^f = U(a^f) = (u(a^f), \lambda(a^f))$ of (1) is a simple fold if $\lambda'(a^f) = \frac{\partial \lambda}{\partial a}(a^f) = 0$, for some $a^f \in (0, a_{max})$. In other words, the tangent to the solution branch at that point is vertical on a projected bifurcation diagram that plots λ along the horizontal axis.
When the branch of solutions is approximated using series (3), the fold may be detected by monitoring the sign of $\lambda'(a)$ between successive checkpoints. This derivative is deduced from the series of λ in a straightforward manner as

$$\lambda'(a) = \sum_{k=1}^{K} k a^{k-1} \lambda_k(a). \tag{8}$$

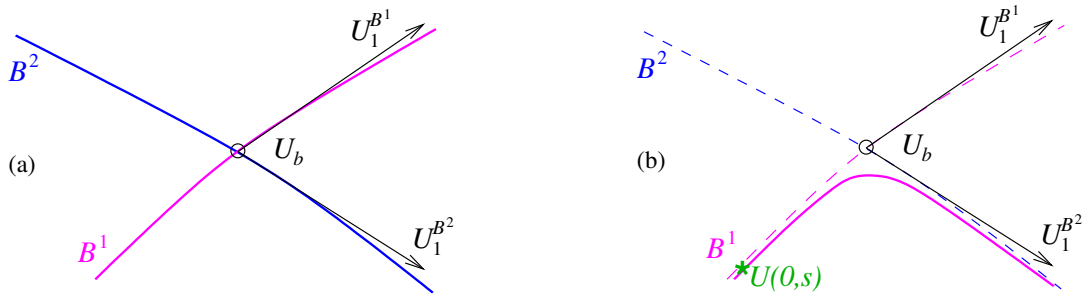

Figure 2. Bifurcation: (a) Perfect case, (b) Numerical case.

Note that the curvature $\lambda''(a^f)$ at the fold can be approximated from the series as

$$\lambda''(a^f) = \sum_{k=2}^{K} k(k-1)(a^f)^{k-2}\lambda_k(a^f), \tag{9}$$

and allows to characterize U^f as a simple quadratic fold when $\lambda''(a^f) \neq 0$.

The fold detection summarized in List. 3 takes place in the continuation driver (List. 2).

Listing 3 Fold detection and location carried out in the continuation driver

```
% Given a checkpoint point U computed using List. 1, that is
%      known series U(a) = ∑_{k=0}^{K} a^k U_k, known range a_m and
%      known sign for λ'(0)
Evaluate U(a_m) = ∑_{k=1}^{K} a_m^k U_k          % Solution point
Evaluate λ'(a_m) = ∑_{k=1}^{K} k a^{k-1} λ_k(a_m)   % Derivative of λ
% Fold detection
if (sign(λ'(0)) ~= sign(λ'(a_m)))
     % Fold location
     Find a^f such that ∑_{k=1}^{K} k(a^f)^{k-1} λ_k(a^f) = 0   %dichotomy
     Evaluate λ''(a_f) = ∑_{k=2}^{K} (k-1)k(a^f)^{k-1} λ_k(a^f)
     Update the checkpoint with fold information
end
```

3.2 Bifurcation detection and location

Bifurcation and branch switching is a more complex task. In the perfect case, Fig. 2.a, the two solution branches B^1 and B^2 intersect at the bifurcation point U^b. Tangents to the branches at U^b are denoted by U_1^{t1} and U_1^{t2}. Even a very small error, inherent to floating-point calculations, may induce the branch switching presented in Fig. 2.b. Bifurcation and stability analysis is thus of prime importance in computational and engineering sciences [1, 15, 26]. Various methods exist to detect and to locate a bifurcation, to pass through it along the traveled solution branch or to switch from one branch to the other.

Unsurprisingly, such a singular behavior may be monitored by means of the derivatives of the nonlinear system, and notably using AD [13, 24]. In this section, we take advantage of the HOAD computed Taylor series to detect a geometric series [11, 18] that indicates a bifurcation, and to implement an efficient branch switching method at simple bifurcation

points. The paper reworks using a HOAD formalism the method discussed in [11] for quadratic formulations $\mathcal{R}(U) = L_0 + L(U) + Q(U,U)$ (comprising a constant vector L_0, a linear operator L and a quadratic operator Q) to extend it to the general nonlinear algebraic system (1).

3.2.1 Brief state of the art

Let U^b be a simple bifurcation point on the solution branches B^1 and B^2, Fig. 2.a. At point U^b, the Jacobian $\{\mathcal{R}_1^b\}$ admits two null right vectors ϕ^1 and ϕ^2 and a left null vector ψ [15] that satisfy

$$\{\mathcal{R}_1^b\}\phi^i = 0, \quad \text{for } i = 1,2, \quad \text{and} \quad \psi^*.\{\mathcal{R}_1^b\} = 0. \tag{10}$$

The null space and its orthogonal complement are denoted by $\mathcal{N}(\{\mathcal{R}_1^b\}) = \text{span}\{\phi^1, \phi^2\}$ and $\mathcal{N}^\perp(\{\mathcal{R}_1^b\}) = \mathbb{R}^n \setminus \mathcal{N}(\{\mathcal{R}_1^b\})$, respectively.

Let $(\alpha, \beta) \in \mathbb{R}^2$ and U_1^b be such that

$$U_1^b = (\alpha\phi^1 + \beta\phi^2) \in \mathcal{N}(\{\mathcal{R}_1^b\}). \tag{11}$$

Differentiating (1) twice at point U^b,

$$\{\mathcal{R}_1^b\}U_2^b + \{\mathcal{R}_2^b\}U_1^bU_1^b = \{\mathcal{R}_1^b\}U_2^b + \{\mathcal{R}_2^b\}(\alpha\phi^1 + \beta\phi^2)(\alpha\phi^1 + \beta\phi^2) = 0, \tag{12}$$

and multiplying the result by ψ^* yield

$$\psi^*.\{R_2^b\}(\alpha^2\phi^1\phi^1 + 2\alpha\beta\phi^1\phi^2 + \beta^2\phi^2\phi^2) = 0, \tag{13}$$

and the so-called Algebraic Bifurcation Equation (ABE)

$$\alpha^2 c_{11} + 2\alpha\beta c_{12} + \beta^2 c_{22} = 0, \quad \text{where} \quad c_{ij} = \psi^*.\{\mathcal{R}_2^b\}\phi^i\phi^j \quad \text{for } i = 1,2. \tag{14}$$

The solutions of the ABE allow for the determination of the two tangents. The practical implementation reported in [15] monitors the determinant of $\{R_1\}$ along the traveled branch to detect a possible bifurcation between two successive continuation points.

As demonstrated in [11], a geometric series emerges between successive Taylor coefficients near to a bifurcation point. This particular event may be used for detection and branch switching. Four stages are necessary. Firstly, the Taylor coefficient sequence is monitored to detect a possible geometric series. Secondly, the geometric series is taken out to restore a clean series with an optimal range of validity along the traveled branch. Then, the bifurcation locus is determined. Thirdly, the tangent to the second branch is computed. Finally, an optimal (large) range of validity can be computed at the bifurcation point.

3.2.2 Evidence for a geometric series

Choosing $\phi^1 = U_1^{t1}$ and $\phi^2 = U_1^{t2}$ in (11) allows to write the first order representation of branches B^1 and B^2 as

$$U(\alpha, \beta) = U^b + \alpha U_1^{t1} + \beta U_1^{t2}. \tag{15}$$

In the perfect case, Fig. 2.a, the ABE simplifies into $\alpha\beta = 0$, and either α or β is null. This agrees with the particular choices done for ϕ^1 and ϕ^2. The approximate solution $U(\alpha, \beta)$ theoretically lies on one or the other branch.

Numerical residual errors, Fig. 2.b, are usually not zero from a computer point of view. The ABE becomes $\alpha\beta = \mu$ where μ is a small real number. Using $\alpha = a - s$ and $\beta = \mu/(a - s)$ in (15) allows to write

$$U(a,s) = \underbrace{U^b - sU_1^{t1} - \left(\frac{\mu}{s}\right)U_1^{t2}}_{U(0,s)} + aU_1^{t1} + \left(\frac{\mu}{s}\right)\left(\frac{a}{a-s}\right)U_1^{t2}, \tag{16}$$

where the shift s represents the distance from $U(0, s)$ to U^b.

Equation (16) provides a first order representation of the perturbed branch B^1 evaluated at a point U_0 near to the simple bifurcation under study. Since μ/s is very small, the contribution of U_1^{t2} is negligible except when a is almost equal to s. In that case, the rational fraction $a/(a - s)$ may be written as the geometric series $\sum_k (a/s)^k$ with common ratio a/s. This happens more easily for a small distance s and/or for a large residual error contribution μ at point U^b. The interested reader is referred to [18] for the analysis of functional singularities through series.

The formation of a geometric series is monitored in the highest four Taylor coefficients of the last computed solution for (5). The candidates as common ratio and scale factor are

$$\sigma = \frac{U_K^* U_K}{U_{K-1}^* U_K}, \tag{17}$$

and the Taylor coefficient U_K, respectively. A geometric series is detected when collinearity (18) and proportionality (19) properties are satisfied

$$\sqrt{\sum_{k=1}^{3}\left(1 - \frac{U_{K-k}^* U_K}{||U_{K-k}||.||U_K||}\right)^2} \leq \varepsilon_{coll} = 10^{-4}, \tag{18}$$

and

$$\sum_{k=1}^{3} \frac{|(U_{K-k} - \sigma U_{K-k+1})^*.U_{K-k}|}{U_{K-k}^*.U_{K-k}} \leq \varepsilon_{prop} = 10^{-3}. \tag{19}$$

In practice, the geometric series is detected along the traveled branch either before or after the bifurcation point U^b, depending on the distance s to U^b and on the norm of the residual.

3.2.3 Cleaning of the current Taylor series and bifurcation location

The identified geometric series is taken off the Taylor coefficients to improve the utility of the series, that is to avoid (i) interactions between the two solution branches, (ii) a possible amplification of computational errors, (iii) small ranges of validity, or even (iv) an undesired branch switching as in Fig. 2.b.

Listing 4 Simple bifurcation detection, location and computation of the tangents.

```
% ... This goes into Listing 1
% Given a series U(a) = sum_{k=0}^{K} a^k U_k and a nonlinear function R
% Detection of a possible geometric series
if (18) && (19)
     'simple bifurcation point'
     ∃ Geometric series: Common ratio (17), scale factor U_K
     % "improve the utility of the series"
     Compute the clean series Û following (20)
     Locate the bifurcation U^b = Û(σ)
     Deduce the tangent U_1^{t1} from Û
     Compute the second tangent U_1^{t2}
     Compute the range of validity along the current branch
%else
%    'regular point'
end
```

The clean series $\widehat{U}(a)$,

$$\widehat{U}(a) = \sum_{k=0}^{K-1} a^k \widehat{U}_k = \sum_{k=0}^{K-1} a^k (U_k - \sigma^{K-k} U_k), \tag{20}$$

provides the perfect branch of solutions in the current tangent direction. Its range of validity is evaluated using (6) at order $K-1$.

By construction, the bifurcation point satisfies $U^b = \widehat{U}(\sigma)$. An approximation of the tangent to the traveled branch at that point is $U_1^{t1} = \dfrac{\partial \widehat{U}}{\partial a}(\sigma)$.

3.2.4 Determination of the second tangent

At the bifurcation point U^b, the Taylor coefficient U_1^{t2} defining the second tangent belongs to the kernel $\mathcal{N}(\{\mathcal{R}_1^b\})$ and satisfies the ABE. Using $U_1^{t2} = \alpha U_1^{t1} + \beta\phi^2$ with ϕ^2 belonging to $\mathcal{N}(\{\mathcal{R}_1^b\})$ allows to compute U_1^{t2} from (13).

3.3 Branch switching

The series at the bifurcation point U^b may be written as

$$U(a) = U^b + aU_1^b + \sum_{k=2}^{K} a^k U_k^b, \tag{21}$$

where U_1^b is one of the two tangents and U_k^b, for $k = 2, .., K$, are unknown Taylor coefficients. This series is introduced into (1). Resulting equations are projected onto $\mathcal{N}(\{\mathcal{R}_1^b\})$,

$$\pi_\psi(\{\mathcal{R}_1^b\}U_k^b + \{\mathcal{R}_k^b | U_k^b = 0\}) = 0, \tag{22}$$

and $\mathcal{N}^{\perp}(\{\mathcal{R}_1^b\})$,

$$\psi^*.(\{\mathcal{R}_{k+1}^b|U_k^b=1,U_{k+1}^b=0\}U_k^b+\{\mathcal{R}_{k+1}^b|U_k^b=U_{k+1}^b=0\})=0, \tag{23}$$

to obtain the iterative sequence of linear systems satisfied by the unknowns U_k^b, $k\geq 2$. The path equation contribution is

$$(U_1^b)^*.U_k^b=0. \tag{24}$$

The solution for (22)–(24) is managed assuming

$$U_k^b=\alpha_k U_1^{t1}+\beta_k U_1^{t2}+\Upsilon_k, \tag{25}$$

where the unknowns α_k and β_k are real parameters and $\Upsilon_k\in\mathcal{N}^{\perp}(\{\mathcal{R}_1^b\})$. Firstly, the linear system for Υ_k is deduced from the projection (22) and the path equation, that is

$$\left(\begin{array}{cc}\{\mathcal{R}_1^b\} & \psi\\ (U_1^{t1})^* & 0\\ (U_1^{t2})^* & 0\end{array}\right).\left(\begin{array}{c}\Upsilon_k\\ \\ \gamma\end{array}\right)=\left(\begin{array}{c}-\{\mathcal{R}_k^b|U_k^b=0\}\\ 0\\ 0\end{array}\right), \tag{26}$$

where the $(n+2)\times(n+2)$ left-hand side matrix is nonsingular and γ is a Lagrange multiplier. Secondly, replacing U_1^b by U_1^{t1} and using (25) into the path equation (24) allow to deduce

$$(U_1^{t1})^*.U_k^b=\alpha_k(U_1^{t1})^*.U_1^{t1}+\beta_k(U_1^{t1})^*.U_1^{t2}=0, \tag{27}$$

that is

$$\alpha_k=-\beta_k\frac{(U_1^{t1})^*.U_1^{t2}}{(U_1^{t1})^*.U_1^{t1}}. \tag{28}$$

Finally, equation (23) allows to determine β_k and the final equation for U_k^b,

$$U_k^b=\beta_k\left(U_1^{t2}-\frac{(U_1^{t1})^*.U_1^{t2}}{(U_1^{t1})^*.U_1^{t1}}\right)+\Upsilon_k. \tag{29}$$

The optimal range of validity for the second series is deduced from the Taylor coefficient U_k^b following (6).

These different stages are implemented as in Listing 5.

4. Numerical results

This section first presents numerical results obtained for two tutorial examples from MatCont and Auto. Numerical experiments are carried out using matcont5p3 [22], AUTO-07P [16], and the Matlab implementation of Diamant called Diamanlab [7]. These three software are usable for research purposes. Simulations are effected on commodity hardware comprising an Intel processor at a maximal frequency of 2.9 GHz.

Listing 5 Series computation at a bifurcation point, the transpose operator is here denoted by ' as in Matlab

```
% Given the bifurcation point U = U^b, the tangents U1t1 and U1t2,
%            and the nonlinear function R
Compute the Jacobian {R_1} and the left null vector ψ
Factorize the left-hand side matrix of (26)
Set U_0 = U(0); U_1 = U1t1;
% Order 2
Set V_2 = U_2 computed from (29) with β_2 = 0 and ϒ_2 = 0
Project V_2 to compute pV_2 = ψ'.{R_3|U_2 = V_2, U_3 = 0}
Solve (26) to get ϒ_2
Set U_2 = ϒ_2 and evaluate W_3 = {R_3|U_3 = 0}
Project W_3 to get pW_3
Compute β_2 = −pW_3/pV_2
Update U_2 using (29)
Update the right-hand side term of (26)
% Higher orders
for k=3:K % in brief
     Solve (26) to get ϒ_k
     Evaluate W_{k+1} = {R_{k+1}|U_{k+1} = 0}  and project it
     Compute β_k
     Update U_k and the right-handside term of (26)
end
Compute a_m following (6)
```

Before proceeding, major practical differences in the implementations should be noted. Among them, Matcont and Auto solve ordinary differential equations while Diamanlab solves algebraic equations. Note that AUTO embeds a collocation method (spatial discretization). MatCont and Diamanlab deal with Matlab and propose interactive graphical interfaces, while Auto deals with Fortran codes and command lines. The use of software interfaces (ergonomy is out of scope here) and executable codes for matcont5p3 and AUTO-07P interfere with time measurements. Consequently, the software are mainly compared in terms of number of continuation steps.

The clamped-clamped Elastica beam [11, 14, 19] then illustrates the interest for higher-order continuation and bifurcation methods for PDE problems in structural mechanics.

4.1 Stationary solutions of Bratu-Gelfand problems

The Bratu-Gelfand problem,

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u + \lambda \exp(u), \quad t \geq 0,\ x \in (0,1), \\ u(0) = u(1) = 0, \end{cases} \tag{30}$$

is a classical case study in continuation, see [15, 29] for instance, and some numerical results have already been reported for the three software packages [6, 17, 27].

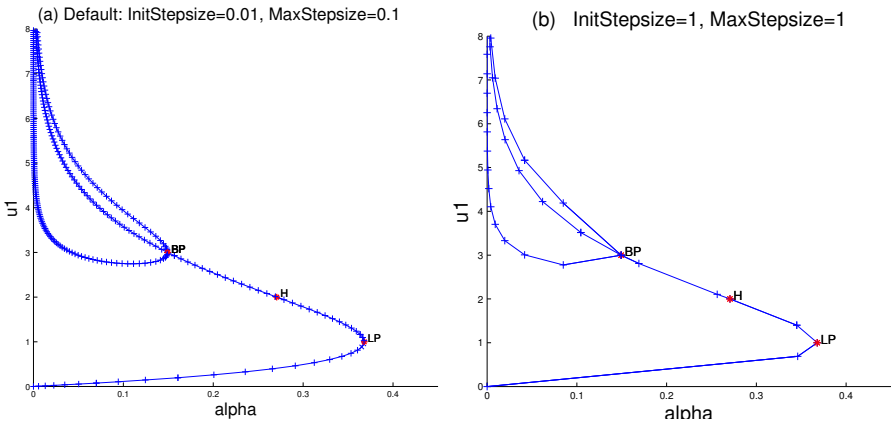

Figure 3. Tutorial bifurcation diagram using MatCont. (a): Default stepsizes. (b): Large Stepsizes.

4.1.1 Two internal point discretization

The tutorial [27] studies equilibrium solutions for a discretization of (30) by means of two internal points. In MatCont, the nonlinear system is implemented as

$$\begin{cases} u_1' = -2u_1 + u_2 + \alpha \exp(u_1), \\ u_2' = u_1 - 2u_2 + \alpha \exp(u_2), \end{cases} \tag{31}$$

by using $\alpha = \lambda/9$ and a symbolic differentiation for the first two derivatives. MatCont saves symbolically computed derivatives in a Matlab function. AD was not introduced in the ODE version of MatCont because it turned out to be slower than symbolic derivatives [12]. Note that the (small) time necessary to precompute the model derivatives when using Matcont is not indicated and is not accounted at simulation time. The equilibrium solutions of (31) satisfy (7) as well.

Default initial (`InitStepsize`) and maximum stepsize (`MaxStepsize`) values are equal to 0.01 and 0.1. In comparison, assuming that the computed a_m has a finite value, there is no limitation on the maximum range of validity for the series in Diamanlab. Solution branches plotted in Fig. 3.a show the fold (LP) and the branch point (BP). As MatCont solves the ODE system (31), it can also find out its Hopf bifurcation (H). The two methods, Figs. 1 and 3.a, are equally accurate at computed points and singular points since they satisfy the same threshold.

For numerical experiments, MatCont is then run using various stepsizes to enable a comparison between the adaptive stepsize methods of MatCont and Diamanlab. Computed solution points are indicated with + in Figs. 3.a and 3.b. Clearly, see Fig. 3.b, using a few larger stepsizes in a discrete method degrades the graphical representation. In comparison, a few series are sufficient to be obtain a very accurate plot, even near to singular points.

The numbers of continuation steps and checkpoints computed with MatCont and Diamant are reported in Tab. 1. Results indicate that, for an equivalent quality in the plots, the HOAD continuation can be carried out using less steps, less Jacobian calculations and less matrix factorizations, than a first order continuation method. This can be a valuable asset when the factorization of the left-hand side of (5) is more expensive than the computation of the higher order right-hand side terms. This is not the case in Matlab where performance and operator overloading (OO) look mutually exclusive. Time measurement is of difficult handling when dealing with interactive tools and so small

Table 1. Number of continuation steps/checkpoints computed with MatCont and Diamant for the two internal point discretization.

	MatCont's	MaxStepsize		Diamant
`InitStepsize`	0.01	1	10	–
`MaxStepsize`	0.1	1	10	–
Branch 1	124	30	10	5
Branch 2	210	52	13	17

Table 2. Number of continuation steps/checkpoints and CPU times necessary to Auto and Diamanlab.

	AUTO-07P			Diamanlab	
	Label	Nb steps	time	Nb checkpoints	time
Fold	2	33		2	
Last point	3	100	0.044s	6	1.49 s

systems. It results that Diamanlab's OO implementation needs 0.37 s to compute and to plot the first checkpoint, what is acceptable in an interactive continuation, where MatCont's needs 0.2 s to compute and plot the first 20 points.

4.1.2 General case

On the one hand, the `exp` demo proposed in AUTO-07P [17] computes the equilibrium solutions of (30) by means of the system of first order ODEs

$$\begin{cases} u' = v, \\ v' = -\lambda \exp(u), \\ u(0) = u(1) = 0, \end{cases} \tag{32}$$

where u and v are functions of the variable x only. These are discretized by means of a collocation method involving 10 mesh intervals and 4 Gauss points per interval.

On the other hand, using Diamanlab, we consider the stationary discrete system comprising N equidistributed points

$$\begin{cases} R_1(u,\lambda) & = (-2u_1 + u_2)/h^2 + \lambda \exp(u_1), \\ R_i(u,\lambda) & = (u_{i-1} - 2u_i + u_{i+1})/h^2 + \lambda \exp(u_i), \quad \text{for } i = 2,..,N-1, \\ R_N(u,\lambda) & = (u_{N-1} - 2u_N)/h^2 + \lambda \exp(u_N), \end{cases} \tag{33}$$

where $h = 1/(N+1)$ and u_i, for $i = 1,..,N$, are the unknowns at discretization points.

The run of the `exp` demo is carried out from the trivial null solution as described in [17]. Within Auto, the adaptive discretization method used 10 mesh intervals and 4 Gauss points per interval. Min and max stepsizes are set to 10^{-3} and 2.10^{-1} respectively. One hundred continuation steps allow to reach the last point, at label 3 in Fig 4(a). The fold at label 2 is reached after 33 continuation steps. Diamanlab is run using a finite difference discretization involving $N = 39$ equidistributed points to work with a similar number of unknowns. The fold is detected in the series of the second checkpoint. Bifurcation diagrams (λ, u_{max}) are plotted in Fig. 4 using Plaut and Matlab, respectively. Although computing six checkpoints is sufficient, see Tab. 2, any Matlab interpreted implementation can compete with a compiled code as far as computer time is considered.

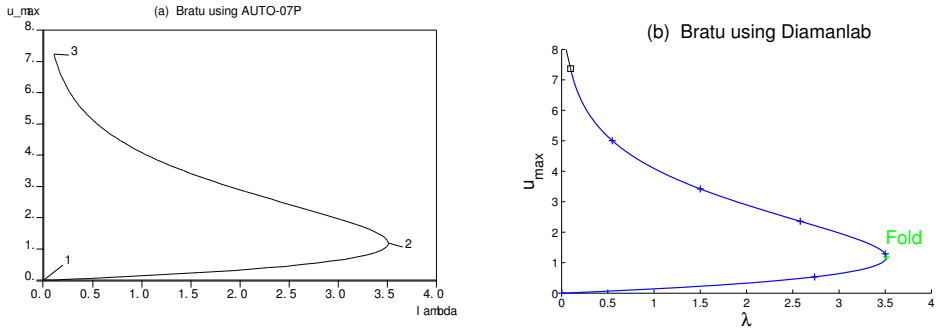

Figure 4. Bifurcation diagram for the equilibriums of (30). (a) AUTO-07P, (b) Diamanlab.

4.2 Stationary solutions of an enzyme model

The two-compartment enzyme model (Kernevez,1980) implemented in the `enz` demo of AUTO-07P,

$$\begin{cases} s_1' = (s_0 - s_1) + (s_2 - s_1) - \dfrac{100 s_1}{1 + s_1 + s_1^2}, \\ s_2' = (s_0 - s_2) + (s_1 - s_2) - \dfrac{100 s_2}{1 + s_2 + s_2^2}, \end{cases} \tag{34}$$

is now chosen as a case study to compare the three software. Therein concentrations s_1 and s_2 are computed with respect to the variation of the concentration s_0. Simulations are carried out using the three software to observe the number the continuation steps necessary to reach the singular points as labeled in Fig. 5(a) by using AUTO-07P. Surprisingly, the `enz` demo also makes use of an *a priori* non-necessary discretization (`NTST`=15 mesh intervals, `NCOL`= 4 Gauss collocation points). The maximum stepsize is set to 0.25 for both AUTO-07P and MatCont simulations.

Bifurcation diagram produced by Matcont and Diamanlab are plotted to observe the Hopf points detected by MatCont, Fig. 5(b) and the checkpoints computed by Diamanlab, Fig. 5(c). In this figure, the second branch is computed the point BP2 for a better rendering ($a_m = 2.81$ between BP2 and O_2). Branching at BP1 yields a series with a smaller range of validity of $a_m = 0.092$ because of neighboring singularities.

Conclusions raised from the first case study still apply, the series computed at the bifurcation point following List. 5 may have a large range of validity. Time measurements reveal that MatCont uses 0.4 s to compute the first 150 steps (just before the first fold), but it requires 1.5 s when it computes and plots the solution simultaneously. In comparison, Diamanlab computes and plots the first 8 checkpoints (outreaching the fold) in 1.57 s. The overhead due to OO is acceptable in an interactive continuation.

4.3 Elastica beam

We now consider the classical clamped-clamped inextensible elastic beam subject to a uniaxial compression [11, 14, 19], the bifurcation diagram of which contains many interconnected solution branches, Fig. 6. This case study is representative of large nonlinear models related to the static behavior of mechanical structures discretized by means of a finite element method.

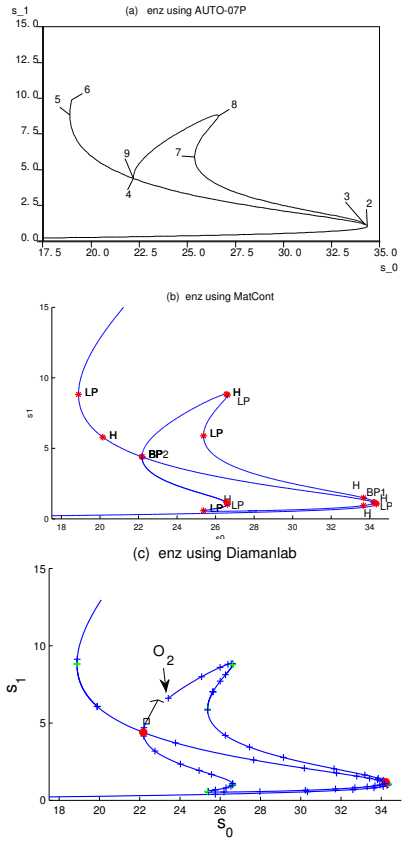

Figure 5. Bifurcation diagram for the `enz` demo. (a) AUTO-07P, (b) MatCont, (c) Diamanlab from BP2.

Table 3. Enzyme model: Number of continuation steps/checkpoints necessary to Auto, MatCont and Diamanlab.

Branch 1	Type	Label	AUTO-07P	matcont5p3	Diamanlab
	LP	2	141	152	8
	H	–	–	155	–
	BP1	3	145	157	10
	BP2	4	214	211	18
	H	–	–	224	–
	LP	5	244	243	19
Branch 2	Type	Label	AUTO-07P	matcont5p3	Diamanlab
	H	–	–	13	–
	LP	7	84	54	12
	LP	8	104	69	17
	H	–	–	74	–
	LP	9	172	–	26
	BP2	–	–	107	26

4.3.1 Equations and discretization

Let $s \in (0,1)$ be the dimensionless curvilinear coordinate. The beam equilibriums satisfy

$$\begin{cases} x'(s) = \cos(\theta(s)), \\ y'(s) = \sin(\theta(s)), \\ \theta'(s) = m(s), \\ m'(s) = -4\pi^2(P\sin(\theta(s)) + T\cos(\theta(s)), \end{cases} \tag{35}$$

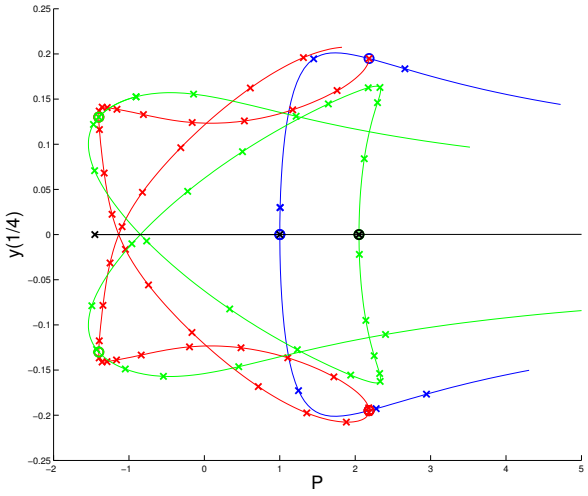

Figure 6. Bifurcation diagram for the clamped-clamped Elastica beam. The first two buckling branches (blue and green curves) are connected to the red closed loop at four bifurcation points.

where $(x(s), y(s))$ is the position of the beam centerline, $\theta(s)$ is the rotation of the beam cross-section and $m(s)$ is the bending momentum. Boundary conditions are $x(0) = y(0) = \theta(0) = 0$ and $m(0) = 2\pi C$ on the one side, and $y(1) = \theta(1) = 0$ on the other side. The compressive load P varies while T and C are the lateral and torque reactions at the boundary. The ODE system is discretized on a regular mesh of N_e elements with a cubic interpolation of the four unknown variables. An orthogonal collocation method at Gauss points is used to build an algebraic system that comprises about $3 \times N_e \times 4$ equations (55 in the present simulation). The truncation order, the threshold (6) and the Newton-Raphson threshold are set to $K = 20$, $\varepsilon = 10^{-6}$ and 2.10^{-5}, respectively.

4.3.2 Simulation results

The simulation is carried out as follows. The initial point is located at (0.030,1.000) on the bifurcation diagram that plots the coordinate $y(1/4)$ with respect to the variations of the load P. The initial tangent points downward. The lower part of the blue solution branch is obtained in 4 continuation steps. Checkpoints are indicated by crosses. The upper part is computed from the initial point by reversing the sign of the tangent at point $(0.030, 1.000)$. Three more steps are computed. This blue curve corresponds to the first buckling branch and presents 3 bifurcation points indicated with circles and located at points (1.000,0.000) and (2.184,$\pm$0.195).

For visualization purposes, Diamanlab has been modified to change the line color to another one when a bifurcation point is selected from the graphical user interface as the beginning for a continuation along the second tangent. The red closed loop is plotted from the bifurcation located on the blue curve at point (2.184,0.195) in about 40 Taylor series computations. Bifurcations appear at (-1.400,$\pm$0.130). One observes that the distance between successive checkpoints varies and becomes smaller near to sharp folds. Bifurcations are detected and located on the fly without the need for small steps.

The green curve corresponds to the second buckling branch. It is plotted from the bifurcation located at (-1.400,0.130) on the red loop. About 30 checkpoints are computed and a new bifurcation is found. Former observations about steps size near to folds and

bifurcations are confirmed. Finally the trivial compression branch is followed and plotted as a black line. The bifurcation diagram can be built from the trivial branch too.

The complex bifurcation diagram is accurately and continuously handled with the proposed HOAD continuation and bifurcation framework. This approach is especially interesting for large structures involving numerous degrees of freedom since the continuation steps between successive checkpoints are large, notably near to the bifurcations. The number of Jacobian calculations and linear system solutions is thus limited.

4.3.3 Discussion

The use of higher order methods (continuation and bifurcation analysis) yields continuous solution branches that can be plotted as they are in a bifurcation diagram. It also reduces the number of checkpoints that has to be computed, including near to the singular points.

The efficiency in terms of computer time of this Matlab prototype of Diamant is obviously penalized by the use of operator overloading. In spite of this, the compromise between interactivity, generality and time remains actually acceptable for prototyping numerical methods and for educational purposes, notably in computational mechanics. In particular, we plane to interface Diamanlab with PDE codes involving plate or shell finite elements through the user-defined `R.m` method that implements the nonlinear function $\mathcal{R}$ of (1).

To that end, time performances can be optimized in several manners. First, sparse Jacobian calculations or, better, Jacobian calculation at assembly time can be implemented for PDE problems. Second, the HOAD computations in (5) and (26) could be improved to avoid re-computations by working on the HOAD library to preserve generality. Finally, HOAD and Diamant can be implemented in another language.

An attractive option will be to consider `Arbogast` [5], the toolbox for HOAD based on Modular C [8], as it provides an interface to differentiate C programs and to implement new differential operators from scratch. As Diamant solvers for continuation (5), bifurcation analysis (26), homotopy [2] and sensitivity analysis [6], are differential operators, they can be good candidates to benefit from `Arbogast` in terms of code and performance optimization.

5. Conclusions

Theoretical aspects of higher-order continuation and bifurcation analysis are presented in a HOAD formalism to emphasize the generality of this framework. The key point is in the series computations that enable for large adaptive continuation steps notably near to bifurcation points, and that limit the number of solutions/checkpoints to be computed while preserving the quality of bifurcation diagram plots. Strengths and current limitations are discussed on classical case studies from Auto and MatCont. Then, the Elastica beam example shows that complex bifurcation diagrams and simple bifurcations are handled in an accurate and robust manner. The proposed method is especially interesting for large structures involving numerous degrees of freedom because the number of Jacobian calculations and linear system solutions get smaller as continuation steps between successive checkpoints are large.

Future work is concerned with the solution of PDE problems through a finite element modeling (plate and shell elements) by using Diamanlab. The implementation in `Arbogast` will be then studied.

References

[1] E. L. Allgower and K. Georg. Numerical continuation methods: an introduction. Springer-Verlag, New-York, 1990.

[2] M. Bilasse, I. Charpentier, E.M. Daya, and Y. Koutsawa. A generic approach for the solution of nonlinear residual equations. Part II: Homotopy and complex nonlinear eigenvalue method. Comput. Method. Appl. M., 198:3999–4004, 2009.

[3] I Charpentier. Checkpointing schemes or adjoint codes: Application to the meteorological model Meso-NH. SIAM J. Sci. Comput., 22:2135–2151, 2001.

[4] I. Charpentier. On higher-order differentiation in nonlinear mechanics. Optim. Method. Softw., 27:221–232, 2012.

[5] I. Charpentier and J. Gustedt. Arbogast, higher-order AD for special functions with Modular C. Optimization methods&Software, 33:963–987, 2018.

[6] I. Charpentier and K. Lampoh. Sensitivity computations in higher order continuation methods. Applied Mathematical Modelling, 40:3365–3380, 2016.

[7] Isabelle Charpentier, Bruno Cochelin, and Komlanvi Lampoh. Diamanlab - An interactive Taylor-based continuation tool in MATLAB. March 2013.

[8] Pierre-Nicolas Clauss and Jens Gustedt. Iterative Computations with Ordered Read-Write Locks. Journal of Parallel and Distributed Computing, 70(5):496–504, 2010.

[9] B. Cochelin. A path-following technique via an asymptotic-numerical method. Comput. Struct., 53:1181 – 1192, 1994.

[10] B. Cochelin, N. Damil, and M. Potier-Ferry. Méthode asymptotique numérique. Hermes Science Publications, Paris, 2007.

[11] B. Cochelin and M. Médale. Power series analysis as a major breakthrough to improve the efficiency of asymptotic numerical method in the vicinity of bifurcations. J. Comput. Phys., 236:594–607, 2013.

[12] Virginie De Witte and Willy Govaerts. Numerical computation of normal form coefficients of ODEs in Matlab. In Wei Feng, Zhaosheng Fent, Maurizio Grasselli, Akif Ibragimov, Xin Lu, Stefan Siegmund, and Jürgen Voigt, editors, Discrete and Continuous Dynamical Systems Supplements, volume 2011, v. I, pages 362–372. American Institute of Mathematical Sciences (AIMS), 2011.

[13] A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, H.G.E. Meijer, and B. Sautois. New features of the software matcont for bifurcation analysis of dynamical systems. Mathematical and Computer Modelling of Dynamical Systems, 14(2):147–175, 2008.

[14] E. Doedel. Lecture notes on numerical analysis of nonlinear equations. In B. Krauskopf, H.M. Osinga, and J. Galan-Vioque, editors, Numerical Continuation Methods for Dynamical Systems, pages 1–49. Springer Netherlands, 2007.

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

[16] E.J. Doedel, B.E. Oldeman, and Collaborators. Auto-07p. `https://sourceforge.net/projects/auto-07p/`, 2015.

[17] Eusebius J. Doedel and Bart E. Oldeman. AUTO-07P: Continuation and Bifurcation Software for Ordinary Differential E February 2012.

[18] M. Van Dyke. Analysis and improvement of perturbation series. Q. J. Mech. Appl. Math., 27:423–450, 1974.

[19] P.E. Farrell, C.H.L. Beentjes, and Ásgeir Birkisson. The computation of disconnected bifurcation diagrams. 2016.

[20] M. Golubitsky and D.G. Schaeffer. Singularities and Groups in Bifurcation Theory - Volume I. Springer, New-York, 1985.

[21] M. Golubitsky, M. Stewart, and D. Schaeffer. Singularities and Groups in Bifurcation Theory - Volume II. Springer, New-York, 1988.

[22] W. Govaerts, Yu.A Kuznetsov, and Collaborators. Matcont: Numerical bifurcation analysis toolbox in matlab. `https://sourceforge.net/projects/matcont/files/matcont/matcont5p3/`, 2013.

[23] Andreas Griewank and Andrea Walther. Algorithm 799: Revolve: An implementation of checkpoint for the reverse or adjoint mode of computational differentiation. ACM Transactions on Mathematical Software, 26(1):19–45, mar 2000. Also appeared as Technical University of Dresden, Technical Report IOKOMO-04-1997.

[24] John Guckenheimer and Brian Meloon. Computing periodic orbits and their bifurcations with automatic differentiation. SIAM Journal on Scientific Computing, 22(3):951–985, 2000.

[25] G. Hentz, I. Charpentier, and P. Renaud. Higher-order continuation for the determination of robot workspace boundaries. C.R. Mecanique, 344:95–101, 2016.
[26] H.B. Keller. Lectures on numerical methods in bifurcation problems. Springer-Verlag, Berlin, 1987.
[27] Yu.A Kuznetsov. Tutorial II: One-parameter bifurcation analysis of equilibria with matcont, 2011.
[28] R. Seydel. Numerical computation of branch points in nonlinear equation. Numer. Math., 33:339–352, 1979.
[29] R. Seydel. Practical Bifurcation and Stability Analysis. Number 5 in Interdisciplinary Applied Mathematics. Springer, 3rd edition, 2009.