Bifurcation Analysis of Bogdanov-Takens Bifurcations in Delay Differential Equations

BOSSCHAERT, Maikel & Kuznetsov, Yu. A. (2024) Bifurcation Analysis of Bogdanov-Takens Bifurcations in Delay Differential Equations. In: SIAM journal on applied dynamical systems, 23 (1) , p. 553 -591.

DOI: 10.1137/22M1527532
Handle: http://hdl.handle.net/1942/42672

Bifurcation analysis of Bogdanov-Takens bifurcations in delay differential equations*

M.M. Bosschaert† Yu.A. Kuznetsov‡

October 7, 2022

Abstract: In this paper, we will perform the parameter-dependent center manifold reduction near the generic and transcritical codimension two Bogdanov–Takens bifurcation in classical delay differential equations (DDEs). Using a generalization of the Lindstedt-Poincaré method to approximate the homoclinic solution allows us to initialize the continuation of the homoclinic bifurcation curves emanating from these points. The normal form transformation is derived in the functional analytic perturbation framework for dual semigroups (sun-star calculus) using a normalization technique based on the Fredholm alternative. The obtained expressions give explicit formulas, which have been implemented in the freely available bifurcation software package `DDE-BifTool`. The effectiveness is demonstrated on various models.

Keywords: generic Bogdanov–Takens bifurcation, transcritical Bogdanov–Takens bifurcation, homoclinic solutions, delay differential equations, sun-star calculus, strongly continuous semi-groups, Center Manifold Theorem, DDE-BifTool

2020 MSC: 37G05, 37G10, 65P30, 34K16, 34K18, 34K19

1 Introduction

The *Bogdanov–Takens bifurcation* caused by the presence of an equilibrium with a double zero eigenvalue is a well-studied singularity in dynamical systems. It implies existence of saddle homoclinic orbits nearby, which is a global phenomenon. In particular, the codimension two Bogdanov–Takens bifurcation in finite-dimensional ordinary differential equations (ODEs) has been studied theoretically and applied in numerous reserach publications. The same is true for the infinite-dimensional dynamical systems generated by delay differential equations (DDEs). In the simplest case, often encountered in applications, such DDEs have the form

$$\dot{x}(t) = f(x(t), x(t-\tau_1), \ldots, x(t-\tau_m), \alpha), \qquad t \geq 0, \tag{1}$$

where $x(t) \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^p$, while $0 < \tau_1 < \tau_2 < \cdots < \tau_m$ are constant delays, and $f : \mathbb{R}^{n\times(m+1)} \times \mathbb{R}^p \to \mathbb{R}^n$ is a smooth mapping. These are known as *discrete* DDEs.

Up to date, the standard available parameter-dependent center manifold theorem for DDEs in [13] assumed that the equilibrium persists for all nearby parameter values. This was a serious limitation, since in generic unfoldings of the codimension two Bogdanov–Takens singularity it is not the case.

*Submitted to the editors DATE.

†Department of Mathematics, Hasselt University, Diepenbeek Campus, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium (maikel.bosschaert@uhasselt.be).

‡Department of Mathematics, Utrecht University, Budapestlaan 6, 3508 TA Utrecht, The Netherlands and Department of Applied Mathematics, University of Twente, Zilverling Building, 7500AE Enschede, The Netherlands (I.A.Kouznetsov@uu.nl).

However, recently, in [2], this obstruction has been removed and the existence of finite-dimensional smooth parameter-dependent local center manifolds has been rigorously established in the functional analytic perturbation framework for dual semigroups (sun-star calculus) developed in [5, 6, 7, 8]. Once the existence of these invariant manifolds is proved, the normalization technique for local bifurcations of ODEs developed in [27] can be lifted rather easily to the infinite dimensional setting of DDEs. The advantages of this normalization technique are that the center manifold reduction and the calculation of the normal form coefficients are performed simultaneously by solving the so-called *homological equation*. This method gives explicit expressions for the coefficients rather than a procedure as developed in [17, 18]. The explicit expressions make them particularly suitable for both symbolic and numerical evaluation.

Indeed, utilizing the normalization method, the authors in [2] obtained asymptotics to initialize the continuation of codimension one bifurcation curves of nonhyperbolic equilibria and cycles emanating from the codimension two *generalized Hopf, fold-Hopf, Hopf-Hopf* and *transcritical-Hopf* bifurcation points in DDEs of the form (1). These asymptotics have been implemented into the fully `GNU Octave` compatible `MATLAB` package `DDE-BifTool` [16, 35].

Another recent developent is the rigorous derivation of higher-order asymptotics for the codimension one homoclinic bifurcation curve emanating from the generic codimension two Bogdanov–Takens bifurcation point in in two-parameter ODEs [3]. Thus, by combining the results of the parameter-dependent center manifolds in DDEs from [2] and the homoclinic asymptotics in ODEs from [3], we are in the position to perform the parameter-dependent center manifold reduction and normalization for the *generic* and *transcritical codimension two Bogdanov–Takens* bifurcations. This will allow us to initialize the continuation of codimension one bifurcation curves of nonhyperbolic equilibria and homoclinic solutions emanating from these codimension two points. Hopefully, our results and their software implementation will make the numerical analysis of Bogdanov–Takens bifurcations in DDEs from applications rather routine.

This paper is organized as follows. We begin in Section 2 with a short summary from [2] on parameter-dependent center manifolds for classical DDEs and we state various results needed for the normalization technique. In Section 3 we describe the general technique that we use to derive the transformation from the orbital normal form on the parameter-dependent center manifold in the infinite-dimensional setting of DDEs. In Section 4 the method is then applied to the generic and transcritical codimension two Bogdanov–Takens bifurcations. We provide explicit transformations necessary for the predictors of codimension one bifurcation curves. We do this in a form suitable for classical DDEs, covering cases that are more general than (1). It is here where we see the true benefit of allowing for orbital normal forms on the center manifold. Indeed, we do not need to derive homoclinic asymptotics for the transcritical codimension two Bogdanov–Takens bifurcation separately. Instead, we only need to derive the center manifold transformation for the transcritical codimension two Bogdanov–Takens bifurcation. Then, using the blow-up transformations (82), we obtain the same perturbed Hamiltonian system (up to order three) as in the generic Bogdanov–Takens bifurcation.

We employ our implementation in `DDE-BifTool` to illustrate the accuracy of the codimension one bifurcation curve predictors through various example models, displaying the generic and transcritical codimension two Bogdanov–Takens bifurcations in Section 5. An in-depth treatment of the examples, including the `MATLAB` and Julia source code to reproduce the obtained results, as well as a more in-depth analysis of the examples, are provided in the Supplement.

2 Parameter-dependent center manifolds for DDEs

Here we summarize those results from [2] on parameter-dependent center manifold for classical DDEs, which are required for the normalization technique in Section 4. For a general introduction on perturbation theory for dual semigroups (also known as sun-star calculus) we refer to [13].

Consider the classical parameter-dependent DDE

$$\dot{x}(t) = F(x_t, \alpha), \qquad t \geq 0, \tag{DDE}$$

where $F : X \times \mathbb{R}^p \to \mathbb{R}^n$ is C^k-smooth for some $k \geq 1$ with $F(0,0) = 0$ and $X := C([-h, 0], \mathbb{R}^n)$. Here for each $t \geq 0$, the *history function* $x_t : [-h, 0] \to \mathbb{R}^n$ defined by

$$x_t(\theta) := x(t + \theta), \qquad \forall\, \theta \in [-h, 0].$$

It is convenient to split the right hand-side into its linear and nonlinear parts and write

$$F(\phi, \alpha) = \langle \zeta, \phi \rangle + D_2F(0,0)\alpha + G(x_t, \alpha). \tag{2}$$

Here $\zeta : [0, h] \to \mathbb{R}^{n\times n}$ is a matrix-valued function of bounded variation, normalized by the requirement that $\zeta(0) = 0$ and is right-continuous on the open interval $(0, h)$, and $G : X \to \mathbb{R}^n$ is a C^k-smooth nonlinear operator with $G(0,0) = G_1(0,0) = G_2(0,0) = 0$. The pairing is defined by

$$\langle \zeta, \phi \rangle := \int_0^h d\zeta(\theta)\phi(-\theta), \tag{3}$$

where the integral is of the Riemann–Stieltjes type.

Let T be the $\mathcal{C}_0$-semigroup on X corresponding to the linearization of (DDE) at $0 \in X$ for the critical parameter value $\alpha = 0$. Suppose that the generator

$$\mathcal{D}(A) = \{\phi \mid \dot{\phi} \in X, \dot{\phi}(0) = \langle \zeta, \phi \rangle\}, \qquad A\phi = \dot{\phi},$$

of T has $1 \leq n_0 < \infty$ purely imaginary eigenvalues with corresponding n_0-dimensional real center eigenspace X_0. Then by [2, Corollary 20] there exists a C^k-smooth map $\mathcal{C} : U \times U_p \to X$ defined in a neighborhood of the origin in $X_0 \times \mathbb{R}^p$ and such that for every sufficiently small $\alpha \in \mathbb{R}^p$ the manifold $\mathcal{W}^c_{\text{loc}}(\alpha) := \mathcal{C}(U, \alpha)$ is locally positively invariant for the semiflow generated by (DDE) at parameter value α.

Since X is not reflexive, i.e. does not isomorphic to its dual space $X^\star$, the adjoint semigroup $T^\star$ is only weak* continuous on $X^\star$ and $A^\star$ generates $T^\star$ only in the weak* sense. The maximal subspace of strong continuity

$$X^\odot := \{x^\star \in X^\star \,:\, t \mapsto T^\star(t)x^\star \text{ is norm-continuous on } \mathbb{R}_+\}$$

is invariant under $T^\star$, and we have the representation

$$X^\odot = \mathbb{R}^{n\star} \times L^1([0, h], \mathbb{R}^{n\star}). \tag{4}$$

The duality pairing between $\phi^\odot = (c, g) \in X^\odot$ and $\phi \in X$ is

$$\langle \phi^\odot, \phi \rangle = c\phi(0) + \int_0^h g(\theta)\, \phi(-\theta)\, d\theta. \tag{5}$$

At this stage, we again have a $\mathcal{C}_0$-semigroup $T^\odot$ with generator $A^\odot$ on a Banach space $X^\odot$ so we can iterate the above construction once more. On the dual space

$$X^{\odot\star} = \mathbb{R}^n \times L^\infty([-h, 0], \mathbb{R}^n),$$

we obtain the adjoint semigroup $T^{\odot\star}$ with the weak* generator

$$\mathcal{D}(A^{\odot\star}) = \{(\alpha, \phi) \in X^{\odot\star} \mid \phi \in \text{Lip}(\alpha)\}, \qquad A^{\odot\star}(\alpha, \phi) = (\langle \zeta, \phi \rangle, \dot{\phi}). \tag{6}$$

The duality pairing between $\phi^{\odot\star} = (a, \psi) \in X^{\odot\star}$ and $\phi^{\odot} = (c, g) \in X^{\odot}$ is

$$\langle \phi^{\odot\star}, \phi^{\odot} \rangle = ca + \int_0^h g(\theta)\, \psi(-\theta)\, d\theta. \tag{7}$$

By restriction to the maximal subspace of strong continuity $X^{\odot\odot} = \overline{\mathcal{D}(A^{\odot\star})}$, we end up with the $\mathcal{C}$-semigroup $T^{\odot\odot}$. Its generator $A^{\odot\odot}$ is the part of $A^{\odot\star}$ in $X^{\odot\odot}$. The canonical injection $j : X \to X^{\odot\star}$ is given by

$$j(\phi) = (\phi(0), \phi)\,, \tag{8}$$

mapping X *onto* $X^{\odot\odot}$. Therefore, X is sun-reflexive with respect to the shift semigroup T.

We are now in the position to state the second part of [2, Corollary 20]. That is, if the history x_t associated with a solution of (DDE) exists on some nondegenerate interval I and $x_t \in \mathcal{W}^c_{\mathrm{loc}}(\alpha)$ for all $t \in I$, then $u : I \to X$ defined by $u(t) \coloneqq x_t$ is differentiable and satisfies

$$j\dot{u}(t) = A^{\odot\star} j u(t) + (D_2 F(0,0)\alpha) r^{\odot\star} + G(u(t), \alpha) r^{\odot\star}, \qquad \forall\, t \in I.$$

Here, for $i = 1, \dots, n$, we denote $r_i^{\odot\star} \coloneqq (e_i, 0)$, where e_i is the ith standard basis vector of $\mathbb{R}^n$ and

$$w r^{\odot\star} \coloneqq \sum_{i=1}^{n} w_i r_i^{\odot\star}, \qquad \forall\, w = (w_1, \dots, w_n) \in \mathbb{R}^n.$$

2.1 Spectral computations for classical DDEs in case of multiple eigenvalues

It is well known that for classical DDEs all spectral information about the generator A is contained in a holomorphic *characteristic matrix function* $\Delta : \mathbb{C} \to \mathbb{C}^{n\times n}$ defined by

$$\Delta(z) \coloneqq zI - \hat{\zeta}(z) \qquad \text{with} \qquad \hat{\zeta}(z) \coloneqq \int_0^h e^{-z\theta}\, d\zeta(\theta), \tag{9}$$

where ζ is the real kernel from (3), see [13, Sections IV.4 and IV.5]. In particular, the eigenvalues of A are the roots of the *characteristic equation*

$$\det \Delta(z) = 0, \tag{10}$$

and the algebraic multiplicity of an eigenvalue equals its order as a root of (10).

For the normalization technique in Section 4, we will need normalized representations for the (generalized) eigenfunctions and adjoint (generalized) eigenfunctions of the generator A and $A^\star$, respectively. In this section, we will consider the more general case where λ is an eigenvalue of algebraic multiplicity $k \in \mathbb{N}$ and geometric multiplicity 1. Although in Section 4 we will only need the special case where $\lambda = 0$ is a double eigenvalue of A, the expressions are useful when considering for example the 1:1 resonant Hopf bifurcation and the triple zero bifurcation.

Proposition 1. *Let λ be an eigenvalue of the generator A with algebraic multiplicity $k \in \mathbb{N}$ and geometric multiplicity one, then there are (generalized) eigenfunctions ϕ_i such that*

$$A\phi_0 = \lambda\phi_0, \qquad A\phi_i = \lambda\phi_i + \phi_{i-1}, \qquad i \in \{1, \dots, k-1\}, \tag{11}$$

and adjoint (generalized) eigenfunctions ψ_i such that

$$A^\star\psi_{k-1} = \lambda\psi_{k-1}, \qquad A^\star\psi_{k-i} = \lambda\psi_{k-i} + \psi_{k-i+1}, \qquad i \in \{2, \dots, k\}. \tag{12}$$

Let the ordered set $(q_0, \dots, q_{k-1})$ of vectors be a Jordan chain for $\Delta(\lambda)$, i.e. $q_0 \neq 0$ and

$$\Delta(z)[q_0 + (z-\lambda)q_1 + \cdots + (z-\lambda)^k q_{k-1}] = \mathcal{O}((z-\lambda)^k).$$

Similarly, let $(p_{k-1},\dots,p_0)$ *be a Jordan chain for* $\Delta^T(\lambda)$*. Then the (generalized)) eigenfunctions and adjoint (generalized) eigenfunctions are given by*

$$\begin{aligned}\phi_i\colon[-h,0]\to\mathbb{R}^n\colon\theta\mapsto e^{\lambda\theta}\sum_{l=0}^{i}q_{i-l}\frac{\theta^l}{l!},\\ \psi_i\colon[0,h]\to\mathbb{R}^n\colon\theta\mapsto p_i+\sum_{l=0}^{k-1-i}p_{i+l}\int_0^\theta\int_\sigma^h e^{\lambda(\sigma-s)}\frac{(\sigma-s)^l}{l!}d\zeta(s)d\sigma,\end{aligned}\tag{13}$$

for $i\in\{0,\dots,k-1\}$*, respectively. Furthermore, the following identities hold*

$$\begin{aligned}\langle\psi_i,\phi_j\rangle&=\langle\psi_{i+1},\phi_{j+1}\rangle, && i,j\in\{0,\dots,k-2\},\\ \langle\psi_{k-1},\phi_{k-1}\rangle&=p_{k-1}\sum_{l=0}^{k-1}\frac{\Delta^{(l+1)}(\lambda)}{(l+1)!}q_{k-1-l},\\ \langle\psi_i,\phi_j\rangle&=0, && i>j,\\ \langle\psi_0,\phi_j\rangle&=\sum_{l=0}^{k-1}\sum_{m=0}^{j}p_l\frac{\Delta^{(l+m+1)}(\lambda)}{(l+m+1)!}q_{j-m}, && j>0,\end{aligned}\tag{14}$$

which can be normalized to satisfy

$$\langle\psi_i,\phi_j\rangle=\delta_{ij}.\tag{15}$$

Proof. The (generalized) eigenspace at an eigenvalue λ of A of algebraic multiplicity k and geometric multiplicity 1 is given by

$$\mathcal{N}((A-\lambda)^k),$$

which leads to the expressions in (11) and similarly for (12). The representations of the (generalized) eigenfunctions and adjoint (generalized) eigenfunctions can be found in Theorem IV.5.5 and IV.5.9 in [13], respectively.

The first identity in (14) follows directly from

$$\langle\psi_{i+1},\phi_{j+1}\rangle=\langle(\lambda-A^\star)\psi_i,\phi_{j+1}\rangle=\langle\psi_i,(\lambda-A)\phi_{j+1}\rangle=\langle\psi_i,\phi_j\rangle,$$

where $i,j\in\{0,\dots,k-2\}$. For the second identity in (14) we notice that

$$\begin{aligned}\langle\psi_{k-1},\phi_{k-1}\rangle&=\int_0^h d\psi_{k-1}(\theta)\phi_{k-1}(-\theta)=p_{k-1}q_0+\int_0^h\psi'_{k-1}(\theta)\phi_{k-1}(-\theta)d\theta\\ &=p_{k-1}q_0+\int_0^h p_{k-i}\int_\theta^h e^{\lambda(\theta-s)}d\zeta(s)e^{-\lambda\theta}\sum_{l=0}^{k-1}q_{i-l}\frac{(-1)^l\theta^l}{l!}d\theta\\ &=p_{k-1}q_0+p_{k-i}\sum_{l=0}^{k-1}\int_0^h\int_\theta^h e^{-\lambda s}\frac{(-1)^l\theta^l}{l!}d\zeta(s)d\theta q_{i-l}\\ &=p_{k-1}q_0+p_{k-i}\sum_{l=0}^{k-1}\int_0^h\int_0^s e^{-\lambda s}\frac{(-1)^l\theta^l}{l!}d\theta d\zeta(s)q_{i-l}\\ &=p_{k-1}q_0+p_{k-i}\sum_{l=0}^{k-1}\int_0^h e^{-\lambda s}\frac{(-1)^l s^{l+1}}{(l+1)!}d\zeta(s)q_{i-l}=p_{k-1}\sum_{l=0}^{k-1}\frac{\Delta^{(l)}(\lambda)}{(l+1)!}q_{i-l},\end{aligned}$$

where we used Fubini's theorem to change the order of integration. The last equality holds since

$$\Delta'(z)=zI+\int_0^h\theta e^{-z\theta}\,d\zeta(\theta)$$

and

$$\Delta^{(n)}(z) = (-1)^{n+1} \int_0^h \theta^n e^{-z\theta}\, d\zeta(\theta), \quad n > 1.$$

Using the first identity in (14) and that for $j > 0$ we have

$$\langle \psi_j, \phi_0 \rangle = \langle (\lambda - A^\star)\psi_{j-1}, \phi_0 \rangle = \langle \psi_{j-1}, (\lambda - A)\phi_0 \rangle = 0,$$

the third identity in (14) follows.

For the last identity in (14), we have that

$$\begin{aligned}
\langle \psi_0, \phi_j \rangle &= \int_0^h d\psi_0(\theta)\phi_j(-\theta) = p_0 q_j + \int_0^h \psi_0'(\theta)\phi_j(-\theta)d\theta \\
&= p_0 q_j + \int_0^h \left(\sum_{l=0}^{k-1} p_l \int_\theta^h e^{\lambda(\theta - s)} \frac{(\theta - s)^l}{l!} d\zeta(s) e^{-\lambda\theta} \sum_{m=0}^{j} q_{j-m}(-1)^m \frac{\theta^m}{m!} \right) d\theta \\
&= p_0 q_j + \sum_{l=0}^{k-1} \sum_{m=0}^{j} p_l \int_0^h \int_\theta^h e^{-s\lambda} \frac{(\theta - s)^l}{l!} d\zeta(s)(-1)^m \frac{\theta^m}{m!} d\theta q_{j-m} \\
&= p_0 q_j + \sum_{l=0}^{k-1} \sum_{m=0}^{j} p_l \int_0^h \int_0^s e^{-s\lambda} \frac{(\theta - s)^l}{l!} (-1)^m \frac{\theta^m}{m!} d\theta d\zeta(s) q_{j-m} \\
&= p_0 q_j + \sum_{l=0}^{k-1} \sum_{m=0}^{j} p_l \int_0^h e^{-s\lambda} \frac{(-1)^{l+m} s^{l+m+1}}{(l+m+1)!} d\zeta(s) q_{j-m} = \sum_{l=0}^{k-1} \sum_{m=0}^{j} p_l \frac{\Delta^{(l+m+1)}(\lambda)}{(l+m+1)!} q_{j-m}.
\end{aligned}$$

Here we again used Fubini's theorem to reverse the order of integration and the beta function of Euler to integrate the term

$$\int_0^s (\theta - s)^l \theta^m d\theta.$$

To prove the normalization condition $\langle \psi_0, \phi_0 \rangle = 1$, we start by showing that $\langle \psi_0, \phi_0 \rangle$ is non-vanishing. Consider the direct sum decomposition

$$X = \mathcal{N}((\lambda - A)^k) \oplus \overline{\mathcal{R}((\lambda - A)^k)} = \mathcal{N}((\lambda - A)^k) \oplus {}^\perp\mathcal{N}((\lambda - A^\star)^k),$$

see Theorem IV.2.5 in [13]. Since $\phi_0 \in \mathcal{N}((\lambda - A)^k)$ and $\mathcal{N}((\lambda - A^\star)^k)$ is spanned by $\{\psi_0, \cdots \psi_k\}$ it follows that $\langle \psi_0, \phi_0 \rangle \neq 0$.

In order to achieve the normalization in (15), we observe that the eigenfunctions ϕ_j ($j \in \{0, \dots k-1\}$), are invariant under the transformations

$$\phi_0 \to \alpha\phi_0, \tag{16}$$

$$\phi_j \to \alpha(\phi_j + \delta_j \phi_0), \quad j \in \{1, \dots, n\}, \tag{17}$$

where $\alpha, \delta_j \in \mathbb{R}$ for $j \in \{0, \dots, k-1\}$ and $\alpha \neq 0$. Using the first identity in (14), we see that it is sufficient to normalize $\langle \psi_0, \phi_0 \rangle$ to 1 and $\langle \psi_0, \phi_j \rangle$ to 0. Thus, using the invariance of the eigenfunctions, we obtain the solutions

$$\alpha = \frac{1}{\langle \psi_0, \phi_0 \rangle}, \quad \delta_j = -\langle \psi_0, \phi_j \rangle, \quad j \in \{1, \dots, n\}.$$

We finish with the remark that the invariance of the eigenfunctions is equivalent to the invariance for the Jordan chains $(q_0, q_1, \dots, q_{k-1}) \to \alpha(q_0, q_1 + \delta_1 q_0, \dots, q_{k-1} + \delta_{k-1} q_0)$. □

Remark 2. The Jordan chains for $(q_0, \dots, q_{k-1})$ and $(p_{k-1}, \dots, p_0)$ in the previous Proposition 1 can be computed as follows. Set for $j \in \mathbb{N}$

$$P_j = P_j(z) = \frac{\Delta^{(j-1)}(z)}{(j-1)!},$$

and define $A_k = A_k(z)$ to be the $(nk) \times (nk)$-matrix

$$A_k = \begin{pmatrix} P_1 & 0 & \cdots & 0 \\ P_1 & P_2 & \cdots & 0 \\ \cdots & & \ddots & \vdots \\ P_k & P_{k-1} & \cdots & P_1, \end{pmatrix} \qquad k \in \mathbb{N}.$$

Then the Jordan chains for $(q_0, \dots, q_{k-1})$ and $(p_{k-1}, \dots, p_0)$ can be constructed via

$$A_k(\lambda) \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{k-1} \end{pmatrix} = 0, \qquad \begin{pmatrix} p_0 & p_1 & \cdots & p_{k-1} \end{pmatrix} A_k(\lambda) = 0,$$

see [13, Chapter IV Exercise 5.11].

2.2 Solvability of linear operator equations with double eigenvalues

When computing the normal form coefficients for the Bogdanov–Takens bifurcation in Section 4, using the normalization technique described in Section 3, we will encounter linear operator equations of the following form

$$\left(\lambda - A^{\odot\star}\right) jv = (w_0, w), \tag{18}$$

where λ is a double eigenvalue, while no other eigenvalues are present on the imaginary axis, $(w_0, w) \in X^{\odot\star}$ is given and $v \in D(A)$ is the unknown. Note that in general $w_0 \neq w(0)$. Although we will in Section 4 only need the special case where $\lambda = 0$, the results below hold for non-zero (complex) eigenvalues as well.

Firstly, since $\lambda = 0$ is an eigenvalue, we need to assure that (18) is solvable. Therefore, let ψ_1, as given in Proposition 1 with $k = 2$ and $\lambda = 0$, be the adjoint eigenfunction of $A^\star$. Then (18) has a solution $(\phi_0, \phi) \in \mathcal{D}(A^{\odot\star})$ if and only if (w_0, w) annihilates the null space $\mathcal{N}(\lambda I - A^\star)$, i.e., if and only if

$$\langle (w_0, w), \psi_1 \rangle = 0. \tag{FSC}$$

A proof can be found in [22, Lemma 3.2]. This condition is often referred to as the *Fredholm solvability condition.*

Proposition 3. *Suppose λ is a double eigenvalue of A and assume that (18) is consistent for a given $(w_0, w) \in X^{\odot\star}$. Let (q_0, q_1) and (p_1, p_0) be the Jordan chains for $\Delta(0)$ and $\Delta^T(0)$, respectively. Let ϕ_0 be an eigenfunction of A as in Proposition 1. Then the solution to (18) is given by*

$$v(\theta) = e^{\lambda\theta}\xi + \int_\theta^0 e^{\lambda(\theta-s)} w(s)\, ds + \gamma\phi_0(\theta), \qquad \theta \in [-h, 0],$$

with

$$\xi = \Delta(\lambda)^{\mathrm{INV}} \left[w_0 + \int_0^h d\zeta(\theta) \int_{-\theta}^0 e^{-\lambda(\theta+s)} w(s)\, ds \right].$$

and γ some constant.

Proof. From formula (6) we see that (18) is nothing more than solving the first order ordinary differential equation

$$\lambda v - \dot{v} = w, \tag{19}$$

which must satisfy

$$\lambda v(0) - \int_0^h d\zeta(\theta) v(-\theta) = w_0. \tag{20}$$

Solving (19) for $v \in \mathcal{D}(A)$ we obtain

$$v(\theta) = e^{\lambda\theta} v(0) + \int_\theta^0 e^{\lambda(\theta-s)} w(s)\, ds \qquad (\theta \in [-h, 0])\,.$$

It then follows from (20) that

$$\Delta(\lambda) v(0) = w_0 + \int_0^h d\zeta(\theta) \int_{-\theta}^0 e^{-\lambda(\theta+s)} w(s)\, ds,$$

where we used the definition of the characteristic matrix in (9). Solving for $v(0)$ yields

$$v(0) = \Delta(\lambda)^{\mathrm{INV}} \left\{ w_0 + \int_0^h d\zeta(\theta) \int_{-\theta}^0 e^{-\lambda(\theta+s)} w(s)\, ds \right\} + \gamma q_0,$$

where γ is some constant. Now define

$$\tilde{v}(\theta) = e^{\lambda\theta}\xi + \int_\theta^0 e^{\lambda(\theta-s)} w(s)\, ds,$$

so that

$$v(\theta) = \tilde{v}(\theta) + \gamma\phi_0(\theta).$$

□

Remark 4. We observe that the expression for $v(0)$ itself involves a bordered *matrix* inverse,

$$\Delta^{\mathrm{INV}}(\lambda) : \mathcal{R}(\Delta(\lambda)) \to \mathbb{C}^n,$$

which assigns the unique solution of the extended linear system

$$\Delta(\lambda) x = y, \qquad q_0 \cdot x = 0,$$

to every $y \in \mathbb{C}^n$ for which the system $\Delta(\lambda)x = y$ is consistent. In practice, $x = \Delta^{\mathrm{INV}}(\lambda) y$ can be obtained by solving the nonsingular bordered *matrix* system

$$\begin{pmatrix} \Delta(\lambda) & p_1^T \\ q_0^T & 0 \end{pmatrix} \begin{pmatrix} x \\ s \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

for the unknown $(x, s) \in \mathbb{C}^{n+1}$ that, by Cramer's rule, necessarily satisfies $s = 0$. The properties of (finite dimensional) bordered linear systems and their role in numerical bifurcation analysis are discussed more extensively in [26] and [20, Chapter 3].

In Section 4, we will encounter solely systems in which $\lambda = 0$ and w is of polynomial type. For this situation, we have the following results.

Corollary 5. *Suppose that in addition to the assumptions in Proposition 3 that $\lambda = 0$ and the right-hand side in (18) is given by*

$$\begin{pmatrix} w_0 \\ w \end{pmatrix} = \kappa r^{\odot\star} - j\left(\theta \mapsto c_0 + c_1\theta + \cdots + c_n\theta^n\right),$$

then

$$v(\theta) = \xi + c_0\theta + \frac{c_1}{2}\theta^2 + \cdots + \frac{c_n}{n+1}\theta^{n+1} + \gamma\phi_0(\theta), \qquad (\theta \in [-h, 0]),$$

with

$$\xi = \Delta^{\mathrm{INV}}(0)\left[\kappa - \Delta'(0)c_0 - \frac{\Delta''(0)}{2}c_1 - \cdots - \frac{\Delta^{(n+1)}(0)}{n+1}c_n\right],$$

is the solution to the system

$$-A^{\odot\star}jv = \begin{pmatrix} w_0 \\ w \end{pmatrix}.$$

Remark 6. In Section 4 we will use the shorthand notation

$$v = B_0^{\mathrm{INV}}(\kappa - w)$$

for the solutions in Corollary 5.

Lemma 7. *Suppose that in addition to the assumptions in Proposition 3 that $\lambda = 0$ and let $w \in X$ be given by*

$$w\colon [-h, 0] \to \mathbb{R}^n\colon \theta \mapsto c_0 + c_1\theta + \cdots + c_n\theta^n.$$

Then the pairing with the adjoint (generalized) eigenfunctions ψ_0 and ψ_1 in Proposition 1 with $k = 0$ and $\lambda = 0$ are given by

$$\begin{aligned}\langle \psi_0, w\rangle = p_0 &\left(\Delta'(0)c_0 + \frac{\Delta''(0)}{2}c_1 + \cdots + \frac{\Delta^{(n+1)}(0)}{n+1}c_n\right)\\ &+ p_1\left(\frac{\Delta''(0)}{2}c_0 + \frac{\Delta'''(0)}{3\times 2}c_1 + \cdots + \frac{\Delta^{(n+2)}(0)}{(n+2)(n+1)}c_n\right)\end{aligned}$$

and

$$\langle \psi_1, w\rangle = p_1\left(\Delta'(0)c_0 + \frac{\Delta''(0)}{2}c_1 + \cdots + \frac{\Delta^{(n+1)}(0)}{n+1}c_n\right),$$

respectively.

3 Parameter-dependent center manifold reduction combined with normalization and time-reparametrization

In this section, we combine the results from [3] and [2]. That is, we lift the normalization technique for local bifurcations of ODEs to the infinite dimensional settings of DDEs [2], while simultaneously incorporating a time-reparametrization to allow for a further simplification of the normal form [3]. Thus, suppose that $0 \in X$ is a stationary state of (DDE) at the critical parameter value $0 \in \mathbb{R}^p$ and assume there are $n_0 \geq 1$ eigenvalues on the imaginary axis, counting algebraic multiplicities. Let P_0 be the corresponding *real* spectral projector on X, so the range X_0 of P_0 is the *real* n_0-dimensional center eigenspace. [2, Corollary 20] applies to give a parameter-dependent local center manifold $\mathcal{W}^c_{\mathrm{loc}}(\alpha)$ for (DDE).

We allow for the introduction of a new parameter β defined in a neighborhood of $0 \in \mathbb{R}^p$ such that $\alpha = K(\beta)$ for some locally defined C^k-diffeomorphism $K : \mathbb{R}^p \to \mathbb{R}^p$ that is to be determined below. If $u : I \to X$ with $u(t) \coloneqq x_t \in \mathcal{W}^c_{\mathrm{loc}}(\alpha)$ is as in [2, Corollary 20], then u is differentiable on I and satisfies

$$j\dot{u}(t) = A^{\odot\star} j u(t) + (D_2F(0,0)K(\beta))r^{\odot\star} + R(u(t), K(\beta)), \qquad \forall\, t \in I, \tag{21}$$

where R encodes the nonlinear part of F as given by $G(u(t), K(\beta))r^{\odot\star}$. Choose a basis Φ of X_0 and let $\mathcal{H} : \mathbb{R}^{n_0} \times \mathbb{R}^p \to X$ be a locally defined C^k-smooth parametrization of $\mathcal{W}^c_{\mathrm{loc}}(\alpha)$ with respect to Φ and in terms of the new parameter β. For every $t \in I$ we define $v(t) \in \mathbb{R}^{n_0}$ as the coordinate vector of $P_0 u(t)$ with respect to Φ. Then $v : I \to \mathbb{R}^{n_0}$ satisfies a parameter-dependent ordinary differential equation of the form

$$\dot{v}(\eta) = \sum_{|\nu|+|\mu|\geq 1} \frac{1}{\nu!\,\mu!} g_{\nu\mu} v^\nu(\eta) \beta^\mu. \tag{22}$$

The multi-indices ν and μ have lengths n_0 and p, respectively. We assume that (22) is known. Since $\mathcal{H}$ parametrizes $\mathcal{W}^c_{\mathrm{loc}}(\alpha)$,

$$u(t(\eta)) = \mathcal{H}(v(\eta), \beta), \qquad t \in I, \tag{23}$$

with both u and v depending on the parameter, although this is left implicit in the notation. Next, let the time t in (21) and the time η in the normal form (22) be related through the parameter-dependent time-rescaling

$$\frac{dt}{d\eta} = \vartheta(v, \beta), \qquad \vartheta \colon \mathbb{R}^{n_c} \times \mathbb{R}^2 \to \mathbb{R}. \tag{24}$$

Substituting the above relation (23) into (21) and taking into account (24) produces the *homological equation*

$$\left(A^{\odot\star} j\mathcal{H}(v,\beta) + (D_2F(0,0)K(\beta))r^{\odot\star} + R(\mathcal{H}(v,\beta), K(\beta))\right)\vartheta(v,\beta) = jD_1\mathcal{H}(v,\beta)\dot{v}, \tag{HOM}$$

with $\dot{v}$ given by the parameter-dependent normal form (22). The unknowns in (HOM) are $\mathcal{H}$, K, ϑ, and the coefficients $g_{\nu\mu}$ from (22). For $r, s \geq 0$ with $r + s \geq 1$ we denote by $D_1^r D_2^s F(0,0) : X^r \times [\mathbb{R}^p]^s \to \mathbb{R}^n$ the mixed Fréchet derivative of order $r+s$, evaluated at $(0,0) \in X \times \mathbb{R}^p$, with the understanding that at most one of the factor spaces X^r or $[\mathbb{R}^p]^s$ is absent if either $r = 0$ or $s = 0$. We expand the nonlinearity R as

$$R(\phi, \alpha) = \sum_{r+s>1} \frac{1}{r!\,s!} D_1^r D_2^s F(0,0)(\phi^{(r)}, \alpha^{(s)}) r^{\odot\star}, \tag{25}$$

where $\phi^{(r)} \coloneqq (\phi, \ldots, \phi) \in X^r$ and $\alpha^{(s)} \coloneqq (\alpha, \ldots, \alpha) \in [\mathbb{R}^p]^s$. The mappings $\mathcal{H}$, K, and θ can be expanded as

$$\begin{aligned} \mathcal{H}(v,\beta) &= \sum_{|\nu|+|\mu|\geq 1} \frac{1}{\nu!\,\mu!} h_{\nu\mu} v^\nu \beta^\mu, \\ K(\beta) &= \sum_{|\mu|\geq 1} \frac{1}{\mu!} K_\mu \beta^\mu, \\ \vartheta(w,\beta) &= \sum_{|\nu|+|\mu|\geq 0} \frac{1}{\nu!\,\mu!} \theta_{\nu\mu} w^\nu \beta^\mu. \end{aligned} \tag{26}$$

Substituting (22), (25), and (26) into (HOM), collecting coefficients of terms $v^\nu \beta^\mu$ from lower to higher order and solving the resulting linear systems, one can solve recursively for the unknown coefficients $g_{\nu\mu}$, $h_{\nu\mu}$, and K_μ by applying the Fredholm alternative and taking bordered inverses, as explained in Section 2.2.

Remark 8. To determine the coefficients $g_{\nu\mu}$, $h_{\nu\mu}$, K_μ and $\theta_{\nu\mu}$ that are needed to include in order to translate asymptotics to solutions in the normal form (22) in such a way that the approximation order to the asymptotics are also obtained in the original system under consideration, is a non-trivial task. Indeed, one needs to understand precisely which coefficients $g_{\nu\mu}$ in (22) affect the obtained asymptotic up to a certain order. Then, using [3, Proposition 1], we can derive which coefficients $h_{\nu\mu}$, K_μ and $\theta_{\nu\mu}$ are sufficient to include into the transformations.

4 Parameter-dependent center manifold reduction near Bogdanov–Takens points

Using the method as outlined in Section 3, we derive here the coefficients needed to translate the homoclinic third-order asymptotics emanating from generic and transcritical codimension two Bogdanov–Takens bifurcations to the parameter-dependent center manifold. Additionally, we derive asymptotics for the codimension one equilibria bifurcations emanating from these Bogdanov–Takens bifurcation points.

Thus, suppose that (DDE) has an equilibrium $x_0 \equiv 0$ at the critical parameter value $\alpha_0 = (0,0) \in \mathbb{R}^2$ with a double (but not semisimple) zero eigenvalue

$$\lambda_{1,2} = 0, \tag{27}$$

which are the only eigenvalues on the imaginary axis. Furthermore, we assume that the critical normal coefficients a and b, to be defined below, are non-zero. We then consider two different cases, depending on whether the equilibrium remains fixed under parameter variation or not. For both cases there are additional transversality conditions which need to be met, guaranteeing the local invertibility of the parameter mapping K, see (31).

4.1 Generic Bogdanov–Takens bifurcation

The C^∞-equivalent normal form on the parameter-dependent center manifold is

$$\begin{cases} \dot{w}_0 = w_1, \\ \dot{w}_1 = \beta_1 + \beta_2 w_1 + a w_0^2 + b w_0 w_1 + w_0^2 w_1 h(w_0, \beta) + w_1^2 Q(w_0, w_1, \beta), \end{cases} \tag{28}$$

where h is C^∞ and Q is N-flat for an a priori given N, see [4]. Here, the dot represents the derivative with respect to the new time η of $w_i(\eta)(i = 0, 1)$. Furthermore, it is shown in [3] that we can assume $h(0,0) = 0$. By [3, Proposition 1], it is both necessary and sufficient in order to translate the third-order homoclinic predictor to the system (DDE) to expand the functions $R\colon X \times \mathbb{R}^2 \to X^{\odot\star}$, $\mathcal{H}\colon \mathbb{R}^2 \times \mathbb{R}^2 \to X$,

$K\colon \mathbb{R}^2 \to \mathbb{R}^2$, and $\vartheta\colon \mathbb{R}^2 \to \mathbb{R}$ defined in Section 3 as follows

$$R(u,\alpha) = \Big(\frac{1}{2}B(u,u) + A_1(u,\alpha) + \frac{1}{2}J_2(\alpha,\alpha) + \frac{1}{6}C(u,u,u) + \frac{1}{2}B_1(u,u,\alpha) + \frac{1}{2}A_2(u,\alpha,\alpha) + \frac{1}{6}J_3(\alpha,\alpha,\alpha) + \mathcal{O}(\|(u\|,\|\alpha)\|^4)\Big) r^{\odot\star}, \tag{29}$$

$$\begin{aligned}\mathcal{H}(w,\beta) = {} & h_{0010}\beta_1 + h_{0001}\beta_2 + \frac{1}{2}h_{2000}w_0^2 + h_{1100}w_0w_1 + \frac{1}{2}h_{0200}w_1^2 \\ & + h_{1010}w_0\beta_1 + h_{1001}w_0\beta_2 + h_{0110}w_1\beta_1 + h_{0101}w_1\beta_2 + \frac{1}{2}h_{0002}\beta_2^2 \\ & + h_{0011}\beta_1\beta_2 + \frac{1}{6}h_{3000}w_0^3 + \frac{1}{2}h_{2100}w_0^2w_1 + h_{1101}w_0w_1\beta_2 + \frac{1}{2}h_{2001}w_0^2\beta_2 \\ & + \frac{1}{6}h_{0003}\beta_2^3 + \frac{1}{2}h_{1002}w_0\beta_2^2 + \frac{1}{2}h_{0102}w_1\beta_2^2 \\ & + \mathcal{O}(|w_1|^3 + |w_0w_1^2| + |\beta_2w_1^2| + |\beta_1|\|w\|^2 + |\beta_1^2|\|w\| + |\beta_1^2| + \|(w,\beta)\|^4),\end{aligned} \tag{30}$$

$$\begin{aligned}K(\beta) = {} & K_{10}\beta_1 + K_{01}\beta_2 + \frac{1}{2}K_{02}\beta_2^2 + K_{11}\beta_1\beta_2 + \frac{1}{6}K_{03}\beta_2^3 \\ & + \mathcal{O}(|\beta_1|^2 + |\beta_1||\beta_2|^2 + |\beta_1|^2|\beta_2| + \|\beta\|^4),\end{aligned} \tag{31}$$

$$\vartheta(w,\beta) = 1 + \vartheta_{1000}w_0 + \vartheta_{0001}\beta_2 + \mathcal{O}\left(|w_1| + |\beta_2| + \|(w,\beta)\|^2\right). \tag{32}$$

Here B, A_1, J_2, C, B_1, A_2, and J_3 are the standard multilinear forms arising from the expansion of $F(u,\alpha)$. For example,

$$B(u,u) = D_1^2F(0,0)(u,u),\ J_2(\alpha,\alpha) = D_2^2F(0,0)(\alpha,\alpha),\ B_1(u,u,\alpha) = D_2^1D_1^2F(0,0)(u,u,\alpha),$$

etc. Explicit formulas for the multilinear forms for the simplest DDE (1) are given in [2, Section 6].

We insert the expansions (29)–(32) into the homological equation (HOM).

4.1.1 (Generalized) eigenfunctions

By collecting the coefficients of the linear terms in w in the homological equation, we obtain precisely the systems defining the (generalized) eigenfunctions. By Proposition 1 with $k = 2$ and $\lambda = 0$ we obtain

$$\begin{aligned}\phi_0 &= \vartheta \mapsto q_0,\\ \phi_1 &= \vartheta \mapsto \vartheta q_0 + q_1,\\ \psi_1 &= \left(p_1, \vartheta \mapsto p_1 \int_\vartheta^h d\zeta(s)\right),\\ \psi_0 &= \left(p_0, \vartheta \mapsto p_0 \int_\vartheta^h d\zeta(s) + p_1 \int_\vartheta^h (\vartheta - s)\, d\zeta(s)\right),\end{aligned}$$

where (q_0, q_1) and (p_1, p_0) are the Jordan chain for $\Delta(0)$ and $\Delta^T(0)$, respectively. Furthermore, we assume the eigenfunctions to be normalized such that

$$\langle \psi_i, \phi_j\rangle = \delta_{ij}, \qquad 0 \le i, j \le 1. \tag{33}$$

Remark 9. Note that the normalization condition above does not uniquely define the eigenfunction. Indeed, the transformation

$$\psi_1 \mapsto \frac{1}{c}\psi_1, \qquad \psi_0 \mapsto \frac{1}{c}\psi_0, \qquad \phi_0 \mapsto c\phi_0, \qquad \phi_1 \mapsto c\phi_1, \tag{34}$$

for some non-zero constant c, leaves (33) invariant. The same holds true for the transformation

$$\psi_0 \mapsto \psi_0 + \tilde{c}\psi_1, \qquad \phi_1 \mapsto \phi_1 - \tilde{c}\phi_0, \tag{35}$$

for some constant $\tilde{c}$. In [28], the condition

$$q_0^T q_0 = 1, \qquad q_1^T q_0 = 0, \tag{36}$$

is imposed to uniquely define the vectors $\{q_0, q_1, p_1, p_0\}$ up to a plus or minus sign. However, since the non-uniqueness in the eigenfunctions does not alter the order of accuracy of the homoclinic predictors, we will not explicitly impose (36).

4.1.2 Critical coefficients

Collecting the w_0^2, $w_0 w_1$ and w_1^2 terms in the homological equation (HOM), lead to the systems

$$-A^{\odot\star} j h_{2000} = B(\phi_0, \phi_0) r^{\odot\star} - 2aj\phi_1, \tag{37}$$
$$-A^{\odot\star} j h_{1100} = B(\phi_0, \phi_1) r^{\odot\star} - j\left(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000}\right), \tag{38}$$
$$-A^{\odot\star} j h_{0200} = B(\phi_1, \phi_1) r^{\odot\star} - 2j h_{1100}. \tag{39}$$

Pairing equations (37) and (38) with the adjoint eigenfunctions ψ_1 and ψ_0 yields critical normal form coefficients

$$a = \frac{1}{2} p_1 B(\phi_0, \phi_0),$$
$$b = p_0 B(\phi_0, \phi_0) + p_1 B(\phi_0, \phi_1).$$

Now that (37) and (38) are solvable, we use Corollary 5 to define the functions

$$\hat{h}_{2000} = B_0^{\mathrm{INV}}\left(B(\phi_0, \phi_0) - 2aj\phi_1\right),$$
$$\hat{h}_{1100} = B_0^{\mathrm{INV}}\left(B(\phi_0, \phi_1) - (b\phi_1 + \hat{h}_{2000})\right).$$

It follows that the general solutions of the systems (37) and (38) are given by

$$h_{2000} = \hat{h}_{2000} + \gamma_1\phi_0,$$
$$h_{1100} = \hat{h}_{1100} + \gamma_1\phi_1 - \vartheta_{1000}\phi_1 + \gamma_2\phi_0.$$

The constant γ_1 is determined by the solvability condition from (39), which gives

$$\gamma_1 = p_0 B(q_0, q_1) - \langle \psi_0, \hat{h}_{2000}\rangle + \frac{1}{2} p_1 B(q_1, q_1) + \vartheta_{1000},$$

where $\langle \psi_0, \hat{h}_{2000}\rangle$ is given through Lemma 7.

To determine the constant γ_2 and the coefficient ϑ_{1000} we consider the w_0^3 and $w_0^2 w_1$ terms in the homological equation (HOM). After some simplification, we obtain the systems

$$-A^{\odot\star} j h_{3000} = \left[3B(h_{2000}, \phi_0) + C(\phi_0, \phi_0, \phi_0)\right] r^{\odot\star} - 6aj\left(h_{1100} - \vartheta_{1000}\phi_1\right), \tag{40}$$
$$-A^{\odot\star} j h_{2100} = \left[2B(h_{1100}, \phi_0) + B(h_{2000}, \phi_1) + C(\phi_0, \phi_0, \phi_1)\right] r^{\odot\star} \tag{41}$$
$$- j\left(2ah_{0200} + 2bh_{1100} + h_{3000} - 2\vartheta_{1000}(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000})\right). \tag{42}$$

The solvability condition of the first equation determines ϑ_{1000} as

$$\vartheta_{1000} = -\frac{1}{12a} p_1 \left\{3B(\hat{h}_{2000}, \phi_0) + C(\phi_0, \phi_0, \phi_0)\right\} + \frac{1}{2}\langle \psi_1, \hat{h}_{1100}\rangle. \tag{43}$$

The solvability condition for the system in (41) yields, after a rather lengthy calculation, that γ_2 is determined by

$$\gamma_2 = \frac{1}{6a}\Big[p_1\left\{2B(\hat{h}_{1100},q_0)+B(\hat{h}_{2000},q_1)+C(q_0,q_0,q_1)\right\}+2ap_0B(q_1,q_1)$$
$$-bp_1B(\phi_1,\phi_1)+p_0\left(3B(\hat{h}_{2000},q_0)+C(q_0,q_0,q_0)\right)+3\gamma_1 b-10a\langle\hat{h}_{1100},\psi_0\rangle\Big].$$

Since the systems in (39)–(41) are now all consistent, we are allowed to take the bordered inverses to obtain

$$\begin{aligned}
h_{0200} &= B_0^{\mathrm{INV}}\left(B(\phi_1,\phi_1)-h_{1100}\right),\\
h_{3000} &= B_0^{\mathrm{INV}}\left(3B(h_{2000},\phi_0)+C(\phi_0,\phi_0,\phi_0)-6a\left(h_{1100}-\vartheta_{1000}\phi_1\right)\right)\\
h_{2100} &= B_0^{\mathrm{INV}}\left(2B(h_{1100},\phi_0)+B(h_{2000},\phi_1)+C(\phi_0,\phi_0,\phi_1)\right.\\
&\quad\left.-\left(2ah_{0200}+2bh_{1100}+h_{3000}-2\vartheta_{1000}(b\phi_1-\vartheta_{1000}\phi_0+h_{2000})\right)\right).
\end{aligned}$$

4.1.3 Parameter-dependent linear coefficients

The coefficients of the linear terms in β give the systems

$$\begin{aligned}
-A^{\odot\star}jh_{0001} &= J_1K_{01},\\
-A^{\odot\star}jh_{0010} &= J_1K_{10}-\phi_1.
\end{aligned}\tag{44}$$

Since p_1 and J_1 are known, we can calculate

$$\nu := (p_1J_1)^T.$$

By the transversality condition, the vector ν is nonzero. It then follows from the Fredholm alternative that

$$\begin{aligned}
K_{01} &= \delta_1\hat{K}_{01},\\
h_{0001} &= \delta_1\left(\hat{h}_{0001}+\gamma_3\phi_0\right),\\
K_{10} &= \hat{K}_{10}+\delta_2K_{01},\\
h_{0010} &= \hat{h}_{0010}+\delta_2h_{0001}+\gamma_4\phi_0,
\end{aligned}$$

where

$$\begin{aligned}
\hat{K}_{10} &= \frac{1}{\|\nu\|^2}\nu,\\
\hat{h}_{0010} &= B_0^{\mathrm{INV}}\left(J_1K_{10}-\phi_1\right)\\
\hat{K}_{01} &= \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\hat{K}_{10},\\
\hat{h}_{0001} &= B_0^{\mathrm{INV}}\left(J_1\hat{K}_{01}\right),
\end{aligned}$$

and $\delta_{1,2}$, $\gamma_{3,4}$ are real constants determined by the solvability condition of the $w\beta$ terms in the homological equation. Collecting the corresponding systems in the homological equation yields

$$-A^{\odot\star}jh_{1001} = \left(B(h_{0001},\phi_0)+A_1(\phi_0,K_{01})\right)r^{\odot\star},\tag{45}$$
$$-A^{\odot\star}jh_{0101} = \left(B(h_{0001},\phi_1)+A_1(\phi_1,K_{01})\right)r^{\odot\star}-j(h_{1001}+\phi_1-\vartheta_{0001}\phi_0),\tag{46}$$
$$-A^{\odot\star}jh_{1010} = \left(B(h_{0010},\phi_0)+A_1(\phi_0,K_{10})\right)r^{\odot\star}-j(h_{1100}-\vartheta_{1000}\phi_1),$$
$$-A^{\odot\star}jh_{0110} = \left(B(h_{0010},\phi_1)+A_1(\phi_1,K_{10})\right)r^{\odot\star}-j(h_{0200}+h_{1010}).$$

The solvability condition for the first two systems yields

$$\gamma_3 = -\frac{p_1\left(B(\hat{h}_{0001},\phi_0)+A_1(\phi_0,\hat{K}_{01})\right)}{2a},$$
$$\delta_1 = \frac{1}{p_1\left(B(\hat{h}_{0001},\phi_1)+A_1(\phi_1,\hat{K}_{01})\right)+p_0\left(B(\hat{h}_{0001},\phi_0)+A_1(\phi_0,\hat{K}_{01})\right)+\gamma_3 b},$$

while the solvability condition for the latter two systems yields

$$\gamma_4 = \frac{\langle\psi_1,h_{1100}\rangle-\vartheta_{1000}-p_1\left(B(\hat{h}_{0010},\phi_0)+A_1(\phi_0,\hat{K}_{10})\right)}{2a},$$
$$\begin{aligned}\delta_2 = &-p_1\left(B(\hat{h}_{0010},\phi_1)+A_1(\phi_1,\hat{K}_{10})\right)-\gamma_4 b+\langle\psi_1,h_{0200}\rangle\\ &-p_0\left(B(\hat{h}_{0010},\phi_0)+A_1(\phi_0,\hat{K}_{10})\right)+\langle\psi_1,h_{1100}\rangle\,.\end{aligned}$$

Note that the denominator in δ_1 is nonzero by the transversality condition.

4.1.4 Coefficients h_{1010} and h_{0110}

Since we do not need to use the non-uniqueness in the systems for the coefficients h_{1010} and h_{0110} to simplify higher-order systems, it is sufficient to let

$$\begin{aligned}h_{1010} &= B_0^{\mathrm{INV}}\left(B(h_{0010},\phi_0)+A_1(\phi_0,K_{10})-(h_{1100}-\vartheta_{1000}\phi_1)\right),\\ h_{0110} &= B_0^{\mathrm{INV}}\left(B(h_{0010},\phi_1)+A_1(\phi_1,K_{10})-(h_{0200}+h_{1010})\right).\end{aligned}$$

4.1.5 Coefficients $(\vartheta_{0001},\gamma_5), h_{1001}, h_{0101}, h_{2001}, h_{1101}$

Define

$$\begin{aligned}\hat{h}_{1001} &= B_0^{\mathrm{INV}}\left(B(h_{0001},\phi_0)+A_1(\phi_0,K_{01})\right),\\ \hat{h}_{0101} &= B_0^{\mathrm{INV}}\left(\left(B(h_{0001},\phi_1)+A_1(\phi_1,K_{01})\right)-\left(\hat{h}_{1001}+\phi_1\right)\right).\end{aligned}$$

Then the general solutions to the systems in (45) and (46) are given by

$$\begin{aligned}h_{1001} &= \hat{h}_{1001}+\gamma_5\phi_0,\\ h_{0101} &= \hat{h}_{0101}+\gamma_5\phi_1-\vartheta_{0001}\phi_1.\end{aligned}$$

In order to determine γ_5 and ϑ_{0001}, we consider the systems corresponding to the $w_0^2\beta_2$ and $w_0w_1\beta_2$ terms in the homological equation. These are given by

$$\begin{aligned}-A^{\odot\star}jh_{2001} = {}& A_1(h_{2000},K_{01})+B(h_{0001},h_{2000})+2B(h_{1001},\phi_0)\\ &+B_1(\phi_0,\phi_0,K_{01})+C(h_{0001},\phi_0,\phi_0)-2aj(h_{0101}-\vartheta_{0001}\phi_1),\\ -A^{\odot\star}jh_{1101} = {}& A_1(h_{1100},K_{01})+B(h_{0001},h_{1100})+B(h_{0101},\phi_0)+B(h_{1001},\phi_1)\\ &+B_1(\phi_0,\phi_1,K_{01})+C(h_{0001},\phi_0,\phi_1)-j\left[bh_{0101}+h_{1100}+h_{2001}\right.\\ &\left.-\vartheta_{1000}(h_{1001}+\phi_1-\vartheta_{0001}\phi_0)-\vartheta_{0001}(h_{2000}+b\phi_1-\vartheta_{1000}\phi_0)\right].\end{aligned}\tag{47}$$

The Fredholm solvability condition leads to the following system to be solved

$$\begin{pmatrix}2a & 4a\\ b & b\end{pmatrix}\begin{pmatrix}\gamma_5\\ \vartheta_{0001}\end{pmatrix}=\begin{pmatrix}\zeta_1\\ \zeta_2\end{pmatrix}.\tag{48}$$

Here the right-hand side is given by

$$\begin{aligned}
\zeta_1 &= 2a\langle\psi_1,\hat{h}_{0101}\rangle - p_1\left[A_1(h_{2000},K_{01}) + B(h_{0001},h_{2000})\right.\\
&\quad \left. +2B(\hat{h}_{1001},\phi_0) + B_1(\phi_0,\phi_0,K_{01}) + C(h_{0001},\phi_0,\phi_0)\right],\\
\zeta_2 &= \langle\psi_1, b\hat{h}_{0101} + h_{1100} - \vartheta_{1000}\hat{h}_{1001}\rangle - \vartheta_{1000}\\
&\quad - p_1\left[A_1(h_{1100},K_{01}) + B(h_{0001},h_{1100}) + B(\hat{h}_{0101},\phi_0)\right.\\
&\quad \left. +B(\hat{h}_{1001},\phi_1) + B_1(\phi_0,\phi_1,K_{01}) + C(h_{0001},\phi_0,\phi_1)\right]\\
&\quad + 2a\langle\psi_0,\hat{h}_{0101}\rangle - p_0\left[A_1(h_{2000},K_{01}) + B(h_{0001},h_{2000})\right.\\
&\quad \left. + 2B(\hat{h}_{1001},\phi_0) + B_1(\phi_0,\phi_0,K_{01}) + C(h_{0001},\phi_0,\phi_0)\right].
\end{aligned} \tag{49}$$

Notice that the matrix in (48) is invertible by the non-degeneracy condition.

Now that the systems in (47) are solvable, we obtain

$$\begin{aligned}
h_{2001} &= B_0^{\mathrm{INV}}\left((A_1(h_{2000},K_{01}) + B(h_{0001},h_{2000}) + 2B(h_{1001},\phi_0)\right.\\
&\quad \left. +\ B_1(\phi_0,\phi_0,K_{01}) + C(h_{0001},\phi_0,\phi_0)) - 2a\left(h_{0101} - \vartheta_{0001}\phi_1\right)\right),\\
h_{1101} &= B_0^{\mathrm{INV}}\left((A_1(h_{1100},K_{01}) + B(h_{0001},h_{1100}) + B(h_{0101},\phi_0) + B(h_{1001},\phi_1)\right.\\
&\quad \left. +B_1(\phi_0,\phi_1,K_{01}) + C(h_{0001},\phi_0,\phi_1)) - [bh_{0101} + h_{1100} + h_{2001}\right.\\
&\quad \left. -\vartheta_{1000}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0) - \vartheta_{0001}(h_{2000} + b\phi_1 - \vartheta_{1000}\phi_0)]\right).
\end{aligned} \tag{50}$$

4.1.6 Coefficients K_{11} and h_{0011}

Collecting the systems corresponding to the $\beta_1\beta_2$ term in the homological equation yields

$$\begin{aligned}
-A^{\odot\star}jh_{0011} &= (J_1K_{11} + A_1(h_{0001},K_{10}) + A_1(h_{0010},K_{01})\\
&\quad +\ B(h_{0001},h_{0010}) + J_2(K_{01},K_{10}))\, r^{\odot\star} - j\left(h_{0101} - \vartheta_{0001}\phi_1\right).
\end{aligned} \tag{51}$$

Using the identity

$$p_1J_1K_{10} = 1$$

from the second system in (44), combined with the solvability condition, yields

$$\begin{aligned}
K_{11} = &\left[\langle\psi_1,\hat{h}_{0101}\rangle + \gamma_5 - 2\vartheta_{0001} - p_1\left(A_1(h_{0001},K_{10}) + A_1(h_{0010},K_{01})\right.\right.\\
&\left.\left. +B(h_{0010},h_{0001}) + J_2(K_{10},K_{01})\right)\right]K_{10}.
\end{aligned}$$

It follows that

$$\begin{aligned}
h_{0011}(\vartheta) &= B_0^{\mathrm{INV}}\left(J_1K_{11} + A_1(h_{0001},K_{10}) + A_1(h_{0010},K_{01})\right.\\
&\quad \left. +B(h_{0001},h_{0010}) + J_2(K_{01},K_{10}) - (h_{0101} - \vartheta_{0001}\phi_1)\right).
\end{aligned} \tag{52}$$

4.1.7 Coefficients $K_{02}, h_{0002}, h_{1002}, h_{0102}$

The systems corresponding to the $\beta_2^2, w_0\beta_2^2$, and $w_1\beta_2^2$, terms in the homological equation yields

$$\begin{aligned}
-A^{\odot\star} j h_{0002} =& \left(J_1 K_{02} + 2A_1(h_{0001}, K_{01}) + B(h_{0001}, h_{0001}) + J_2(K_{01}, K_{01})\right) r^{\odot\star},\\
-A^{\odot\star} j h_{1002} =& \left(2A_1(h_{1001}, K_{01}) + A_1(\phi_0, K_{02}) + A_2(\phi_0, K_{01}, K_{01})\right.\\
&+ B(\phi_0, h_{0002}) + 2B(h_{0001}, h_{1001}) + 2B_1(\phi_0, h_{0001}, K_{01})\\
&\left.+C(\phi_0, h_{0001}, h_{0001})\right) r^{\odot\star},\\
-A^{\odot\star} j h_{0102} =& \left(2A_1(h_{0101}, K_{01}) + A_1(\phi_1, K_{02}) + A_2(\phi_1, K_{01}, K_{01})\right.\\
&+ B(\phi_1, h_{0002}) + 2B(h_{0001}, h_{0101}) + 2B_1(\phi_1, h_{0001}, K_{01})\\
&\left.+C(\phi_1, h_{0001}, h_{0001})\right) r^{\odot\star} - j\left[2h_{0101} + h_{1002}\right.\\
&\left.-2\vartheta_{0001}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\right].
\end{aligned} \tag{53}$$

The first system is solved similarly as (51). In order to make the second and third systems consistent, define

$$\begin{aligned}
\hat{K}_{02} =& -p_1\left[2A_1(h_{0001}, K_{01}) + B(h_{0001}, h_{0001}) + J_2(K_{01}, K_{01})\right] K_{10},\\
\hat{h}_{0002} =& B_0^{\mathrm{INV}}\left(J_1\hat{K}_{02} + 2A_1(h_{0001}, K_{01}) + B(h_{0001}, h_{0001}) + J_2(K_{01}, K_{01})\right).
\end{aligned}$$

Then the general solutions to the first system in (53) can be written as

$$\begin{aligned}
K_{02} =& \hat{K}_{02} + \delta_3 K_{01},\\
h_{0002} =& \hat{h}_{0002} + \delta_3 h_{0001} + \gamma_6\phi_0.
\end{aligned}$$

Substituting these two expressions into the last two systems of (53) and using the solvability condition yields

$$\begin{aligned}
\gamma_6 =& -\frac{1}{2a}p_1\left[2A_1(h_{1001}, K_{01}) + A_1(\phi_0, \hat{K}_{02}) + A_2(\phi_0, K_{01}, K_{01})\right.\\
&+ B(\phi_0, \hat{h}_{0002}) + 2B(h_{0001}, h_{1001}) + 2B_1(\phi_0, h_{0001}, K_{01})\\
&\left.+C(\phi_0, h_{0001}, h_{0001})\right],\\
\delta_3 =& 2\langle\psi_1, \hat{h}_{0101}\rangle + 2\gamma_5 - 2\vartheta_{0001}\langle\psi_1, \hat{h}_{1001}\rangle - 4\vartheta_{0001} - p_1\left[2A_1(h_{0101}, K_{01})\right.\\
&+ A_1(\phi_1, \hat{K}_{02}) + A_2(\phi_1, K_{01}, K_{01}) + B(\phi_1, \hat{h}_{0002})\\
&\left.+2B(h_{0001}, h_{0101}) + 2B_1(\phi_1, h_{0001}, K_{01}) + C(\phi_1, h_{0001}, h_{0001})\right]\\
&- p_0\left[2A_1(h_{1001}, K_{01}) + A_1(\phi_0, \hat{K}_{02}) + A_2(\phi_0, K_{01}, K_{01})\right.\\
&+ B(\phi_0, \hat{h}_{0002}) + 2B(h_{0001}, h_{1001}) + 2B_1(\phi_0, h_{0001}, K_{01})\\
&\left.+C(\phi_0, h_{0001}, h_{0001})\right] - \gamma_6 b,
\end{aligned}$$

where $\langle\psi_1, \hat{h}_{1001}\rangle = p_0\left(B(h_{0001}, \phi_0) + A_1(\phi_0, K_{01})\right)$.

Now that the last two systems in (53) are consistent, we obtain

$$\begin{aligned}
h_{1002} =& B_0^{\mathrm{INV}}\left(2A_1(h_{1001}, K_{01}) + A_1(\phi_0, K_{02}) + A_2(\phi_0, K_{01}, K_{01})\right.\\
&+ B(\phi_0, h_{0002}) + 2B(h_{0001}, h_{1001}) + 2B_1(\phi_0, h_{0001}, K_{01})\\
&\left.+C(\phi_0, h_{0001}, h_{0001})\right),\\
h_{0102} =& B_0^{\mathrm{INV}}\left(\left(2A_1(h_{0101}, K_{01}) + A_1(\phi_1, K_{02}) + A_2(\phi_1, K_{01}, K_{01})\right.\right.\\
&+ B(\phi_1, h_{0002}) + 2B(h_{0001}, h_{0101}) + 2B_1(\phi_1, h_{0001}, K_{01})\\
&\left.+C(\phi_1, h_{0001}, h_{0001})\right) - \left[2h_{0101} + h_{1002}\right.\\
&\left.\left.-2\vartheta_{0001}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\right]\right).
\end{aligned}$$

4.1.8 Coefficients K_{03} and h_{0003}

Collecting the systems corresponding to the β_2^3 term in the homological equation yields

$$\begin{aligned}-A^{\odot\star} j h_{0003} = \; & (J_1 K_{03} + A_1(h_{0001}, K_{02}) + A_1(h_{0002}, K_{01}) + 2(A_1(h_{0001}, K_{02}) \\ & + A_1(h_{0002}, K_{01}) + 3B(h_{0001}, h_{0002}) + 3J_2(K_{01}, K_{02}) \\ & + 3A_2(h_{0001}, K_{01}, K_{01}) + 3B_1(h_{0001}, h_{0001}, K_{01}) \\ & + C(h_{0001}, h_{0001}, h_{0001}) + J_3(K_{01}, K_{01}, K_{01}))\, r^{\odot\star}.\end{aligned}$$

This equation is solved similarly as equation (51). We obtain

$$\begin{aligned}K_{03} = \; & -p_1 \left[A_1(h_{0001}, K_{02}) + A_1(h_{0002}, K_{01}) + 2A_1(h_{0001}, K_{02})\right. \\ & + 2A_1(h_{0002}, K_{01}) + 3B(h_{0001}, h_{0002}) + 3J_2(K_{01}, K_{02}) \\ & + 3A_2(h_{0001}, K_{01}, K_{01}) + 3B_1(h_{0001}, h_{0001}, K_{01}) \\ & \left. + C(h_{0001}, h_{0001}, h_{0001}) + J_3(K_{01}, K_{01}, K_{01})\right] K_{10}, \\ h_{0003} = \; & B_0^{\mathrm{INV}} \left(J_1 K_{03} + A_1(h_{0001}, K_{02}) + A_1(h_{0002}, K_{01}) + 2A_1(h_{0001}, K_{02})\right. \\ & + 2A_1(h_{0002}, K_{01}) + 3B(h_{0001}, h_{0002}) + 3J_2(K_{01}, K_{02}) \\ & + 3A_2(h_{0001}, K_{01}, K_{01}) + 3B_1(h_{0001}, h_{0001}, K_{01}) \\ & \left. + C(h_{0001}, h_{0001}, h_{0001}) + J_3(K_{01}, K_{01}, K_{01})\right).\end{aligned}$$

4.1.9 Homoclinic asymptotics

The third-order homoclinic asymptotics for the homoclinic orbits emanating from (28) have been derived in [3] and are given by

$$\begin{cases} w_0(\eta) = \dfrac{a}{b^2}\tilde{u}\left(\tanh\left(\xi\left(\dfrac{a}{b}\epsilon\eta\right)\right)\right)\epsilon^2, \\ w_1(\eta) = \dfrac{a^2}{b^3}\tilde{v}\left(\tanh\left(\xi\left(\dfrac{a}{b}\epsilon\eta\right)\right)\right)\epsilon^3, \\ \beta_1 = -4\dfrac{a^3}{b^4}\epsilon^4, \\ \beta_2 = \dfrac{a}{b}\epsilon^2\tau, \end{cases} \tag{54}$$

where

$$\begin{aligned}\tau &= \frac{10}{7} + \frac{288}{2401}\epsilon^2 + \mathcal{O}(\epsilon^4), \\ \tilde{u}(\zeta) &= 2 - (1-\zeta^2)\left(6 + \frac{18}{49}\epsilon^2\right) + \mathcal{O}(\epsilon^4), \\ \tilde{v}(\zeta) &= = -\left[-12 + \frac{72}{7}\zeta\epsilon - \left(\frac{90}{49} + \frac{162}{49}\zeta^2\right)\epsilon^2 + \left(\frac{3888}{2401}\zeta - \frac{216}{343}\zeta^3\right)\epsilon^3\right](1-\zeta^2)\zeta + \mathcal{O}(\epsilon^4), \\ \xi(s) &= s - \frac{6}{7}\log(\cosh(s))\epsilon + \left(-\frac{18s}{49} + \frac{45\tanh(s)}{98} + \frac{36}{49}\tanh(s)\log(\cosh(s))\right)\epsilon^2 \\ &\quad + \left(-\frac{117}{343} + \frac{3\,\mathrm{sech}^2(s)}{4802}\left(-504\log^2(\cosh(s)) - 276\cosh(2s)\log(\cosh(s))\right.\right. \\ &\quad \left.\left. + 102\log(\cosh(s)) + 252 s\sinh(2s)\right)\right)\epsilon^3 + \mathcal{O}(\epsilon^4).\end{aligned} \tag{55}$$

To translate the asymptotics to a generic codimension two Bogdanov–Takens point in (DDE), one should substitute the expressions in (54) into (30) and (31). Furthermore, to accurately approximate

the profiles of the homoclinic orbit, one needs to numerically solve the equation

$$t(\eta) - t = 0,$$

for η, where

$$\begin{aligned} t(\eta) = \eta \Bigg(1 + \vartheta_{0001}\frac{a}{b}\epsilon^2\left(\frac{10}{7} + \frac{288}{2401}\epsilon^2 + \mathcal{O}(\epsilon^4)\right)\Bigg) + \vartheta_{1000}\frac{1}{b}\epsilon\left(\xi(\frac{a}{b}\eta\epsilon) - 6\tanh\Big(\xi(\frac{a}{b}\eta\epsilon)\Big) + \right. \\ \left(\frac{18\,\mathrm{sech}^2(\xi(\frac{a}{b}\eta\epsilon))}{7} + \frac{12}{7}\log\Big(\cosh\Big(\xi(\frac{a}{b}\eta\epsilon)\Big)\Big)\right)\epsilon \\ + \frac{9}{49}\left(4\xi(\frac{a}{b}\eta\epsilon) - 9\tanh\Big(\xi(\frac{a}{b}\eta\epsilon)\Big) + 5\tanh\Big(\xi(\frac{a}{b}\eta\epsilon)\Big)\,\mathrm{sech}^2(\xi(\frac{a}{b}\eta\epsilon))\right)\epsilon^2 \\ \left. + \frac{18\left(-21\,\mathrm{sech}^4(\xi(\frac{a}{b}\eta\epsilon)) + 47\,\mathrm{sech}^2(\xi(\frac{a}{b}\eta\epsilon)) + 8\log\big(\cosh\big(\xi(\frac{a}{b}\eta\epsilon)\big)\big)\right)}{2401}\epsilon^3\right) + \mathcal{O}(\epsilon^4). \end{aligned}$$

4.1.10 Codimension-one equilibria bifurcations

To approximate the fold and Hopf curves and their corresponding equilibria in (28), one should substitute the expressions for β and the equilibrium coordinates into the expansions (30) and (31). It follows that the fold curve is approximated by

$$\begin{cases} x = \mathcal{H}\,(0,0,0,0)\,, \\ \alpha = K(0,\epsilon), \end{cases}$$

for $|\epsilon|$ small, and the Hopf curve by

$$\begin{cases} \omega = \epsilon, \\ x = \mathcal{H}\left(-\dfrac{\epsilon^2}{2a}, 0, -\dfrac{\epsilon^4}{4a)}, \dfrac{b\epsilon^2}{2a}\right), \\ \alpha = K\left(-\dfrac{\epsilon^4}{4a}, \dfrac{b\epsilon^2}{2a}\right), \end{cases}$$

for $\epsilon > 0$ small.

4.2 Transcritical Bogdanov–Takens bifurcation

In this section, we consider the situation in which the origin remains an equilibrium under variation of the parameters.

Following [4], it is not difficult to show that the C^∞-equivalent normal form on the parameter-dependent center manifold takes the form

$$\begin{cases} \dot{w}_0 = w_1, \\ \dot{w}_1 = \beta_1 w_0 + \beta_2 w_1 + a w_0^2 + b w_0 w_1 + w_0^2 w_1 h(w_0,\beta) + w_1^2 Q(w_0, w_1, \beta), \end{cases} \tag{56}$$

where h is C^∞ and Q is N-flat for an a priori given N. Here, the dot represents the derivative with respect to the new time η of $w_i(\eta)(i = 0, 1)$.

In Table 1, we have listed the terms in the parameter-dependent orbital normal form (56) affecting the third-order homoclinic asymptotics in (83). Therefore, by [3, Theorem 1], it is both necessary and sufficient in order to translate the third-order homoclinic predictor to the system (DDE) to expand the

order in ϵ	affected terms
ϵ^{-2}	w_0
ϵ^{-1}	w_1
ϵ^{0}	w_0^2, $w_0\beta_1$, $w_0\beta_2$
ϵ^{1}	$w_1\beta_1$, $w_1\beta_2$, w_0w_1
ϵ^{2}	$w_0\beta_1^2$, $w_0\beta_1\beta_2$, $w_0\beta_2^2$, $w_0^2\beta_1$, $w_0^2\beta_2$, w_0^3, w_1^2
ϵ^{3}	$w_1\beta_1^2$, $w_1\beta_1\beta_2$, $w_1\beta_2^2$, $w_0w_1\beta_1$, $w_0w_1\beta_2$, $w_0^2w_1$

Table 1: Terms in the reduced system restricted to the parameter-dependent center manifold of the codimension two Bogdanov–Takens point in (1) that affect the third-order homoclinic predictor.

functions $R\colon X\times\mathbb{R}^2\to X^{\odot\star}$, $\mathcal{H}\colon\mathbb{R}^2\times\mathbb{R}^2\to X$, $K\colon\mathbb{R}^2\to\mathbb{R}^2$, and $\vartheta\colon\mathbb{R}^2\to\mathbb{R}$ defined in Section 3 as follows

$$\begin{aligned} R(u,\alpha) = &\left(\frac{1}{2}B(u,u) + A_1(u,\alpha) + \frac{1}{6}C(u,u,u) + \frac{1}{2}B_1(u,u,\alpha)\right.\\ &\left. + \frac{1}{2}A_2(u,\alpha,\alpha) + \mathcal{O}(\|(u\|,\|\alpha)\|^4)\right) r^{\odot\star}, \end{aligned} \tag{57}$$

$$\begin{aligned} \mathcal{H}(w,\beta) = &\phi_0w_0 + \phi_1w_1 + \frac{1}{2}h_{2000}w_0^2 + h_{1100}w_0w_1 + \frac{1}{2}h_{0200}w_1^2 + h_{1010}w_0\beta_1\\ &+ h_{1001}w_0\beta_2 + h_{0110}w_1\beta_1 + h_{0101}w_1\beta_2 + \frac{1}{2}h_{0102}w_1\beta_2^2\\ &+ h_{0111}w_1\beta_1\beta_2 + \frac{1}{2}h_{0120}w_1\beta_1^2 + \frac{1}{2}h_{1002}w_0\beta_2^2 + h_{1011}w_0\beta_1\beta_2\\ &+ \frac{1}{2}h_{1020}w_0\beta_1^2 + h_{1101}w_0w_1\beta_2 + h_{1110}w_0w_1\beta_1\\ &+ \frac{1}{2}h_{2001}w_0^2\beta_2 + \frac{1}{2}h_{2010}w_0^2\beta_1 + \frac{1}{2}h_{2100}w_0^2w_1 + \frac{1}{6}h_{3000}w_0^3\\ &+ \mathcal{O}(|\beta_2w_1^2|+|\beta_1w_1^2|+|w_1^3|+|w_0w_1^2|) + \mathcal{O}(\|(w,\beta)\|^4), \end{aligned} \tag{58}$$

$$K(\beta) = K_{10}\beta_1 + K_{01}\beta_2 + \frac{1}{2}K_{20}\beta_1^2 + K_{11}\beta_1\beta_2 + \frac{1}{2}K_{02}\beta_2^2 + \mathcal{O}(\|\beta\|^3). \tag{59}$$

$$\vartheta(w,\beta) = 1 + \vartheta_{1000}w_0 + \vartheta_{0010}\beta_1 + \vartheta_{0001}\beta_2 + \mathcal{O}\left(|w_1|+\|(w,\beta)\|^2\right). \tag{60}$$

The coefficients of the time-reparametrization ϑ have been determined such that the linear systems resulting from the homological equations are solvable, see [3, Remark 2.2]. Note that, since the steady-state remains fixed under variations of parameters, we leave out all coefficients in the expansion of $\mathcal{H}$ which solely depend on the parameters.

Also note that the critical coefficients, i.e. coefficients not depending on any parameters, have already been derived in Section 4.1.2. Therefore, we will start directly with the parameter-dependent systems.

4.2.1 Quadratic terms

Collecting the coefficients of the linear and quadratic terms in the homological equation lead to the systems

$$\begin{aligned}
-A^{\odot\star} j h_{1010} &= A_1(\phi_0, K_{10}) - j\phi_1, \\
-A^{\odot\star} j h_{1001} &= A_1(\phi_0, K_{01}), \\
-A^{\odot\star} j h_{0110} &= A_1(\phi_1, K_{10}) - j(h_{1010} - \vartheta_{0010}\phi_0), \\
-A^{\odot\star} j h_{0101} &= A_1(\phi_1, K_{01}) - j(h_{1001} - \vartheta_{0001}\phi_0) - \phi_1.
\end{aligned} \tag{61}$$

The solvability condition implies that

$$\begin{aligned}
0 &= 1 - p_1 A_1(\phi_0, K_{10}), \\
0 &= -p_1 A(\phi_0, K_{01}), \\
0 &= -p_1 A_1(\phi_1, K_{10}) + \langle \psi_1, h_{1010} \rangle, \\
0 &= -p_1 A_1(\phi_1, K_{01}) + \langle \psi_1, h_{1001} \rangle + 1.
\end{aligned} \tag{62}$$

Pairing the first two systems in (61) with ψ_0 yields

$$\begin{aligned}
\langle \psi_1, h_{1010} \rangle &= -p_0 A_1(\phi_0, K_{10}), \\
\langle \psi_1, h_{1001} \rangle &= -p_0 A_1(q_0, K_{01}).
\end{aligned}$$

Using these identities in the last two equations of (62) yields

$$\begin{cases}
0 &= -p_1 A_1(\phi_1, K_{10}) - p_0 A_1(\phi_0, K_{10}), \\
0 &= -p_1 A_1(\phi_1, K_{01}) - p_0 A_1(\phi_0, K_{01}) + 1.
\end{cases} \tag{63}$$

Now, let

$$\begin{aligned}
K_{10} &= \delta_1 e_1 + \delta_2 e_2, \\
K_{01} &= \delta_3 e_1 + \delta_4 e_2,
\end{aligned}$$

where e_1 and e_2 are the standard basis vectors in $\mathbb{R}^2$ and $\delta_i (i = 1 \dots 4)$ are constants to be determined. Substituting the above expressions for K_{10} and K_{01} into equations (63), and the first two equations of (62), we obtain, by using the linearity of the multilinear forms, that

$$\begin{cases}
0 &= \delta_1 \left(p_1 A_1(\phi_1, e_1) + p_0 A_1(\phi_0, e_1) \right) + \delta_2 \left(p_0 A_1(\phi_0, e_2) + p_1 A_1(\phi_1, e_2) \right), \\
1 &= \delta_3 \left(p_1 A_1(\phi_1, e_1) + p_0 A_1(q_0, e_1) \right) + \delta_4 \left(p_0 A_1(\phi_1, e_2) + p_1 A_1(q_0, e_2) \right).
\end{cases} \tag{64}$$

and

$$\begin{cases}
1 &= \delta_1 p_1 A_1(\phi_0, e_1) + \delta_2 p_1 A_1(\phi_0, e_2), \\
0 &= \delta_3 p_1 (A(\phi_0, e_1) + \delta_4 p_1 (A(\phi_0, e_2),
\end{cases} \tag{65}$$

respectively. Thus, the constants $\delta_i (i = 1 \dots 4)$ are obtained by solving the system

$$\begin{pmatrix}
p_1 A_1(\phi_0, e_1) & p_1 A_1(\phi_0, e_2) \\
p_1 A_1(\phi_1, e_1) + p_0 A_1(\phi_0, e_1) & p_0 A_1(\phi_0, e_2) + p_1 A_1(\phi_1, e_2)
\end{pmatrix}$$

$$\begin{pmatrix} K_{10} & K_{01} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Using Corollary 5 we now define the coefficients

$$\hat{h}_{1010} = B_0^{\mathrm{INV}}\left(A_1(\phi_0, K_{10}), \phi_1\right) \tag{66}$$
$$\hat{h}_{1001} = B_0^{\mathrm{INV}}\left(A_1(\phi_0, K_{01}), \right. \tag{67}$$
$$\hat{h}_{0110} = B_0^{\mathrm{INV}}\left(A_1(\phi_1, K_{10}), h_{1010}\right), \tag{68}$$
$$\hat{h}_{0101} = B_0^{\mathrm{INV}}\left(A_1(\phi_1, K_{01}), h_{1001} + \phi_1\right). \tag{69}$$

Then it follows that solutions to (61) are given by

$$\begin{aligned} h_{1010} &= \hat{h}_{1010} + \gamma_3\phi_0, \\ h_{1001} &= \hat{h}_{1001} + \gamma_4\phi_0, \\ h_{0110} &= \hat{h}_{0110} + \gamma_3\phi_1 - \vartheta_{0010}\phi_1, \\ h_{0101} &= \hat{h}_{0101} + \gamma_4\phi_1 - \vartheta_{0001}\phi_1, \end{aligned} \tag{70}$$

where γ_3 and γ_4 are constants to be determined. Note that we still have freedom in the coefficients h_{0110} and h_{0101} by an addition of a multiple of the eigenfunction ϕ_0. However, for our purposes, we will not need to include it here.

4.2.2 Cubic terms

By collecting the cubic terms in the homological equation, we obtain the following systems

$$\begin{aligned}
-A^{\odot\star} j h_{2010} =\ & \left[A_1(h_{2000}, K_{10}) + 2B(h_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10})\right] r^{\odot\star} \\
& - 2j\left(a h_{0110} + h_{1100} - \vartheta_{1000}\phi_1 - a\vartheta_{0010}\phi_1\right), \\
-A^{\odot\star} j h_{1110} =\ & \left[A_1(h_{1100}, K_{10}) + B(h_{0110}, \phi_0) + B(h_{1010}, \phi_1) + B_1(\phi_0, \phi_1, K_{10})\right] r^{\odot\star} \\
& - j\left(b h_{0110} + h_{0200} + h_{2010} - \vartheta_{1000}(h_{1010} - \vartheta_{0010}\phi_0)\right. \\
& \left. -\vartheta_{0010}(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000})\right), \\
-A^{\odot\star} j h_{2001} =\ & \left[A_1(h_{2000}, K_{01}) + 2B(h_{1001}, \phi_0) + B_1(\phi_0, \phi_0, K_{01})\right] r^{\odot\star} \\
& - 2aj(h_{0101} - \vartheta_{0001}\phi_1), \\
-A^{\odot\star} j h_{1101} =\ & \left[A_1(h_{1100}, K_{01}) + B(h_{0101}, \phi_0) + B(h_{1001}, \phi_1) + B_1(\phi_0, \phi_1, K_{01})\right] r^{\odot\star} \\
& - j\left(b h_{0101} + h_{1100} + h_{2001} - \vartheta_{1000}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\right. \\
& \left. -\vartheta_{0001}(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000})\right), \\
-A^{\odot\star} j h_{1002} =\ & \left[2A_1(h_{1001}, K_{01}) + A_1(\phi_0, K_{02}) + A_2(\phi_0, K_{01}, K_{01})\right] r^{\odot\star}, \\
-A^{\odot\star} j h_{0102} =\ & \left[2A_1(h_{0101}, K_{01}) + A_1(\phi_1, K_{02}) + A_2(\phi_1, K_{01}, K_{01})\right] r^{\odot\star} \\
& - j\left(2h_{0101} + h_{1002} - 2\vartheta_{0001}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\right), \\
-A^{\odot\star} j h_{1011} =\ & \left[A_1(h_{1001}, K_{10}) + A_1(h_{1010}, K_{01}) + A_1(\phi_0, K_{11}) + A_2(\phi_0, K_{01}, K_{10})\right] r^{\odot\star} \\
& - j\left(h_{0101} - \vartheta_{0001}\phi_1\right), \\
-A^{\odot\star} j h_{0111} =\ & \left[A_1(h_{0101}, K_{10}) + A_1(h_{0110}, K_{01}) + A_1(\phi_1, K_{11}) + A_2(\phi_1, K_{01}, K_{10})\right] r^{\odot\star}, \\
& - j\left(h_{0110} + h_{1011} - \vartheta_{0010}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0) - \vartheta_{0001}(h_{1010} - \vartheta_{0010}\phi_0)\right), \\
-A^{\odot\star} j h_{1020} =\ & \left[2A_1(h_{1010}, K_{10}) + A_1(\phi_0, K_{20}) + A_2(\phi_0, K_{10}, K_{10})\right] r^{\odot\star} \\
& - 2j\left(h_{0110} - \vartheta_{0010}\phi_1\right), \\
-A^{\odot\star} j h_{0120} =\ & \left[2A_1(h_{0110}, K_{10}) + A_1(\phi_1, K_{20}) + A_2(\phi_1, K_{10}, K_{10})\right] r^{\odot\star} \\
& - j\left(h_{1020} - 2\vartheta_{0010}(h_{1010} - \vartheta_{0010}\phi_0)\right).
\end{aligned} \tag{71}$$

To solve the above systems, we write the second order parameter coefficients in K with respect to a basis in K_{10} and K_{01} as follows

$$\begin{aligned} K_{02} &= \gamma_5 K_{10} + \gamma_6 K_{01}, \\ K_{11} &= \gamma_7 K_{10} + \gamma_8 K_{01}, \\ K_{20} &= \gamma_9 K_{10} + \gamma_{10} K_{01}. \end{aligned} \tag{72}$$

By applying the Fredholm alternative to the systems in (71), we obtain ten equations to be solved for (also the ten) variables ϑ_{0010}, ϑ_{0001}, γ_3, γ_4, γ_5, γ_6, γ_7, γ_8, γ_9, γ_{10}.

Pairing the first two equations in (71) with ψ_1 and using (70) yields

$$\begin{aligned} 0 = {} & p_1 \left[A_1(h_{2000}, K_{10}) + 2B(\hat{h}_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10}) \right] \\ & - 2\langle \psi_1, a\hat{h}_{0110} + h_{1100} \rangle + 2\vartheta_{1000} + 2a\gamma_3 + 4a\vartheta_{0010}, \\ 0 = {} & p_1 \left[A_1(h_{1100}, K_{10}) + B(h_{0110}, \phi_0) + B(h_{1010}, \phi_1) + B_1(\phi_0, \phi_1, K_{10}) \right] \\ & - \langle \psi_1, ah_{0110} + h_{0200} + h_{2010} - \vartheta_{1000}(bh_{1010} - \vartheta_{0010}\phi_0) \\ & - \vartheta_{0010}(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000}) \rangle . \end{aligned} \tag{73}$$

Since h_{2010} is still to be determined we use that

$$\langle \psi_1, h_{2010} \rangle = \left\langle j h_{2010}, A^{\odot} \psi_0 \right\rangle = \left\langle A^{\odot\star} j h_{2010}, \psi_0 \right\rangle ,$$

Furthermore, from (38) we obtain that

$$\langle \psi_1, b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000} \rangle = p_1 B(\phi_0, \phi_1). \tag{74}$$

Thus, the second equation in (73) becomes

$$\begin{aligned} 0 = {} & p_1 \left[A_1(h_{1100}, K_{10}) + B(h_{0110}, \phi_0) + B(h_{1010}, \phi_1) + B_1(\phi_0, \phi_1, K_{10}) \right] \\ & + p_0 \left[A_1(h_{2000}, K_{10}) + 2B(h_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10}) \right] \\ & - 2 \left\langle \psi_0, ah_{0110} + h_{1100} \right\rangle - \left\langle \psi_1, bh_{0110} + h_{0200} - \vartheta_{1000}(h_{1010} - \vartheta_{0010}\phi_0) \right\rangle \\ & + \vartheta_{0010} p_1 B(\phi_0, \phi_1). \end{aligned}$$

Substituting (72) into the above expression, and using the lineariry of the multilinear forms yields

$$\begin{aligned} 0 = {} & p_1 \left[A_1(h_{1100}, K_{10}) + B(\hat{h}_{0110}, \phi_0) + B(\hat{h}_{1010}, \phi_1) + B_1(\phi_0, \phi_1, K_{10}) \right] \\ & + p_0 \left[A_1(h_{2000}, K_{10}) + 2B(\hat{h}_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10}) \right] \\ & - 2\langle \psi_0, a\hat{h}_{0110} + h_{1100} \rangle - \langle \psi_1, b\hat{h}_{0110} + h_{0200} - \vartheta_{1000}\hat{h}_{1010} \rangle \\ & + b\gamma_3 + b\vartheta_{0010}. \end{aligned}$$

It follows that the constants γ_3 and ϑ_{0010} are give by

$$\begin{pmatrix} 2a & 4a \\ b & b \end{pmatrix} \begin{pmatrix} \gamma_3 \\ \vartheta_{0010} \end{pmatrix} = \begin{pmatrix} \zeta_1 \\ \zeta_2 \end{pmatrix}, \tag{75}$$

where the right-hand side is given by

$$\begin{aligned} \zeta_1 = {} & - p_1 \left[A_1(h_{2000}, K_{10}) + 2B(\hat{h}_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10}) \right] \\ & + 2\langle \psi_1, a\hat{h}_{0110} + h_{1100} \rangle - 2\vartheta_{1000}, \\ \zeta_2 = {} & - p_1 \left[A_1(h_{1100}, K_{10}) + B(\hat{h}_{0110}, \phi_0) + B(\hat{h}_{1010}, \phi_1) + B_1(\phi_0, \phi_1, K_{10}) \right] \\ & - p_0 \left[A_1(h_{2000}, K_{10}) + 2B(\hat{h}_{1010}, \phi_0) + B_1(\phi_0, \phi_0, K_{10}) \right] \\ & + 2\langle \psi_0, a\hat{h}_{0110} + h_{1100} \rangle + \langle \psi_1, b\hat{h}_{0110} + h_{0200} - \vartheta_{1000}\hat{h}_{1010} \rangle. \end{aligned} \tag{76}$$

Notice here that the matrix given in the left-hand side in (75) is equivalent to the matrix in (48) when performing the center manifold reduction near the generic Bogdanov–Takens bifurcation.

The third and fourth systems in (71) lead to a similar system. Indeed, by pairing these systems with the adjoint eigenfunction ψ_1 and using (70) leads to the equations

$$\begin{aligned}
0 &= p_1 \left[A_1(h_{2000}, K_{01}) + 2B(\hat{h}_{1001}, \phi_0) + B_1(\phi_0, \phi_0, K_{01})\right] - 2a\langle \psi_1, \hat{h}_{0101}\rangle + 2a\gamma_4 + 4a\vartheta_{0001},\\
0 &= p_1 \left[A_1(h_{1100}, K_{01}) + B(h_{0101}, \phi_0) + B(h_{1001}, \phi_1) + B_1(\phi_0, \phi_1, K_{01})\right]\\
&\quad - \langle \psi_1, bh_{0101} + h_{1100} + h_{2001} - \vartheta_{1000}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\\
&\quad -\vartheta_{0001}(b\phi_1 - \vartheta_{1000}\phi_0 + h_{2000})\rangle\,.
\end{aligned} \tag{77}$$

By using (70) and (74) the second equation becomes

$$\begin{aligned}
0 = p_1 &\left[A_1(h_{1100}, K_{01}) + B(\hat{h}_{0101}, \phi_0) + B(\hat{h}_{1001}, \phi_1) + B_1(\phi_0, \phi_1, K_{01})\right] +\\
p_0 &\left[A_1(h_{2000}, K_{01}) + 2B(\hat{h}_{1001}, \phi_0) + B_1(\phi_0, \phi_0, K_{01})\right] - 2a\langle \psi_0, \hat{h}_{0101}\rangle\\
&- \langle \psi_1, b\hat{h}_{0101} + h_{1100} - \vartheta_{1000}\hat{h}_{1001}\rangle + \vartheta_{1000} + b\gamma_4 + b\vartheta_{0001}.
\end{aligned}$$

It follows that the constants γ_4 and ϑ_{0001} are give by

$$\begin{pmatrix} 2a & 4a \\ b & b \end{pmatrix} \begin{pmatrix} \gamma_4 \\ \vartheta_{0001} \end{pmatrix} = \begin{pmatrix} \zeta_3 \\ \zeta_4 \end{pmatrix}, \tag{78}$$

where the right-hand side is given by

$$\begin{aligned}
\zeta_3 = &- p_1 \left[A_1(h_{2000}, K_{01}) + 2B(\hat{h}_{1001}, \phi_0) + B_1(\phi_0, \phi_0, K_{01})\right] + 2a\langle \psi_1, \hat{h}_{0101}\rangle\\
\zeta_4 = &- p_1 \left[A_1(h_{1100}, K_{01}) + B(\hat{h}_{0101}, \phi_0) + B(\hat{h}_{1001}, \phi_1) + B_1(\phi_0, \phi_1, K_{01})\right] +\\
&- p_0 \left[A_1(h_{2000}, K_{01}) + 2B(\hat{h}_{1001}, \phi_0) + B_1(\phi_0, \phi_0, K_{01})\right] + 2a\langle \psi_0, \hat{h}_{0101}\rangle\\
&+ \langle \psi_1, b\hat{h}_{0101} + h_{1100} - \vartheta_{1000}\hat{h}_{1001}\rangle + \vartheta_{1000}.
\end{aligned} \tag{79}$$

Now that ϑ_{0010}, ϑ_{0001}, γ_3, and γ_4 are known, we can apply the Fredholm alternative to the remaining six systems in (71). This yields equations which can be solved separately, i.e. without a matrix as in (76) and (79). To keep the derivation readable, we divide six equations into three groups. We start by pairing the fifth and sixth equations in (71) with ψ_1. We obtain that

$$\begin{aligned}
\gamma_5 = &- p_1 \left[2A_1(\hat{h}_{1001}, K_{01}) + A_2(\phi_0, K_{01}, K_{01})\right].\\
0 &= p_1 \left[2A_1(h_{0101}, K_{01}) + A_1(\phi_1, K_{02}) + A_2(\phi_1, K_{01}, K_{01})\right]\\
&\quad - \langle \psi_1, 2h_{0101} + h_{1002} - 2\vartheta_{0001}(h_{1001} + \phi_1 - \vartheta_{0001}\phi_0)\rangle\,,\\
&= p_1 \left[2A_1(\hat{h}_{0101}, K_{01}) + A_2(\phi_1, K_{01}, K_{01})\right] + p_0 \left[2A_1(\hat{h}_{1001}, K_{01}) + A_2(\phi_0, K_{01}, K_{01})\right]\\
&\quad - 2\langle \psi_1, \hat{h}_{0101}\rangle + \gamma_6 + 2\vartheta_{0001}
\end{aligned}$$

Here we used (70) and (72), and the (by now) equalities given (62) and (63). It follows that

$$\begin{aligned}
\gamma_6 = &- p_1 \left[2A_1(\hat{h}_{0101}, K_{01}) + A_2(\phi_1, K_{01}, K_{01})\right] - p_0 \left[2A_1(\hat{h}_{1001}, K_{01}) + A_2(\phi_0, K_{01}, K_{01})\right]\\
&+ 2\langle \psi_1, \hat{h}_{0101}\rangle - 2\vartheta_{0001}.
\end{aligned}$$

By applying the Fredholm alternative to the last four systems in (71), we obtain equations which can be solved similarly as the last two equations. We obtain that

$$\begin{aligned}
\gamma_7 = & -p_1\Big[A_1(\hat{h}_{1001},K_{10})+A_1(\hat{h}_{1010},K_{01})+A_2(\phi_0,K_{01},K_{10})\Big]+\langle\psi_1,\hat{h}_{0101}\rangle-2\vartheta_{0001},\\
\gamma_8 = & -p_1\Big[A_1(\hat{h}_{0101},K_{10})+A_1(\hat{h}_{0110},K_{01})+A_2(\phi_1,K_{01},K_{10})\Big]+\langle\psi_1,\hat{h}_{0110}\rangle\\
& -p_0\Big[A_1(\hat{h}_{1001},K_{10})+A_1(\hat{h}_{1010},K_{01})+A_2(\phi_0,K_{01},K_{10})\Big]+\langle\psi_0,\hat{h}_{0101}\rangle-\vartheta_{0010},\\
\gamma_9 = & -p_1\Big[2A_1(\hat{h}_{1010},K_{10})+A_2(\phi_0,K_{10},K_{10})\Big]+2\langle\psi_1,\hat{h}_{0110}\rangle-4\vartheta_{0010},\\
\gamma_{10} = & -p_1\Big[2A_1(\hat{h}_{0110},K_{10})+A_2(\phi_1,K_{10},K_{10})\Big]\\
& -p_0\Big[2A_1(\hat{h}_{1010},K_{10})+A_2(\phi_0,K_{10},K_{10})\Big]+2\langle\psi_0,\hat{h}_{0110}\rangle.
\end{aligned}\tag{80}$$

Now when all systems in (71) are consistent, we use Corollary 5 to obtain the solutions

$$\begin{aligned}
h_{2010} = & B_0^{\mathrm{INV}}\left(A_1(h_{2000},K_{10})+2B(h_{1010},\phi_0)+B_1(\phi_0,\phi_0,K_{10})\right.,\\
& \left.2\left(ah_{0110}+h_{1100}-\vartheta_{1000}\phi_1-a\vartheta_{0010}\phi_1\right)\right),\\
h_{1110} = & B_0^{\mathrm{INV}}\left(A_1(h_{1100},K_{10})+B(h_{0110},\phi_0)+B(h_{1010},\phi_1)+B_1(\phi_0,\phi_1,K_{10})\right.,\\
& bh_{0110}+h_{0200}+h_{2010}-\vartheta_{1000}(h_{1010}-\vartheta_{0010}\phi_0)\\
& \left.-\vartheta_{0010}(b\phi_1-\vartheta_{1000}\phi_0+h_{2000})\right),\\
h_{2001} = & B_0^{\mathrm{INV}}\left(A_1(h_{2000},K_{01})+2B(h_{1001},\phi_0)+B_1(\phi_0,\phi_0,K_{01}),2a(h_{0101}-\vartheta_{0001}\phi_1)\right),\\
h_{1101} = & B_0^{\mathrm{INV}}\left(A_1(h_{1100},K_{01})+B(h_{0101},\phi_0)+B(h_{1001},\phi_1)+B_1(\phi_0,\phi_1,K_{01})\right.,\\
& bh_{0101}+h_{1100}+h_{2001}-\vartheta_{1000}(h_{1001}+\phi_1-\vartheta_{0001}\phi_0)\\
& \left.-\vartheta_{0001}(b\phi_1-\vartheta_{1000}\phi_0+h_{2000})\right),\\
h_{1002} = & B_0^{\mathrm{INV}}\left(2A_1(h_{1001},K_{01})+A_1(\phi_0,K_{02})+A_2(\phi_0,K_{01},K_{01})\right),\\
h_{0102} = & B_0^{\mathrm{INV}}\left(2A_1(h_{0101},K_{01})+A_1(\phi_1,K_{02})+A_2(\phi_1,K_{01},K_{01})\right.,\\
& \left.2h_{0101}+h_{1002}-2\vartheta_{0001}(h_{1001}+\phi_1-\vartheta_{0001}\phi_0)\right),\\
h_{1011} = & B_0^{\mathrm{INV}}\left(A_1(h_{1001},K_{10})+A_1(h_{1010},K_{01})+A_1(\phi_0,K_{11})+A_2(\phi_0,K_{01},K_{10})\right.,\\
& \left.h_{0101}-\vartheta_{0001}\phi_1\right),\\
h_{0111} = & B_0^{\mathrm{INV}}\left(A_1(h_{0101},K_{10})+A_1(h_{0110},K_{01})+A_1(\phi_1,K_{11})+A_2(\phi_1,K_{01},K_{10})\right.,\\
& \left.h_{0110}+h_{1011}-\vartheta_{0010}(h_{1001}+\phi_1-\vartheta_{0001}\phi_0)-\vartheta_{0001}(h_{1010}-\vartheta_{0010}\phi_0)\right),\\
h_{1020} = & B_0^{\mathrm{INV}}\left(2A_1(h_{1010},K_{10})+A_1(\phi_0,K_{20})+A_2(\phi_0,K_{10},K_{10}),2(h_{0110}-\vartheta_{0010}\phi_1)\right),\\
h_{0120} = & B_0^{\mathrm{INV}}\left(2A_1(h_{0110},K_{10})+A_1(\phi_1,K_{20})+A_2(\phi_1,K_{10},K_{10})\right.,\\
& \left.h_{1020}-2\vartheta_{0010}(h_{1010}-\vartheta_{0010}\phi_0)\right).
\end{aligned}\tag{81}$$

4.2.3 Homolinic asymptotics

To approximate the homoclinic solutions emanating from the transcritical Bogdanov–Takens point, we apply the singular rescaling

$$\beta_1=\pm4\frac{a^2}{b^2}\epsilon^2,\quad \beta_2=\frac{a}{b}\left(\tau\pm2\right)\epsilon^2,\quad w_0=\frac{a}{b^2}(u\mp2)\epsilon^2,\quad w_1=\frac{a^2}{b^3}v\epsilon^3,\quad s=\frac{a}{b}\epsilon\eta,\quad (\epsilon\neq0),\tag{82}$$

to (56) with $h(0,0)=0$ to obtain the second-order nonlinear oscillator

$$\ddot{u} = -4 + u^2 + \dot{u}\,(u+\tau)\,\epsilon + \mathcal{O}(\epsilon^4). \tag{83}$$

Note that this is precisely the second-order nonlinear oscillator obtained in [3] when deriving asymptotics for the homoclinic solutions emanating from the generic Bogdanov–Takens bifurcation. Therefore, the third-order homoclinic asymptotics for the homoclinic orbits emanating from (56) are given by

$$\begin{cases} w_0(\eta) = \dfrac{a}{b^2}\left[\tilde{u}\left(\tanh\left(\xi\left(\dfrac{a}{b}\epsilon\eta\right)\right)\right) \mp 2\right]\epsilon^2, \\ w_1(\eta) = \dfrac{a^2}{b^3}\tilde{v}\left(\tanh\left(\xi\left(\dfrac{a}{b}\epsilon\eta\right)\right)\right)\epsilon^3, \\ \beta_1 = \pm 4\dfrac{a^2}{b^2}\epsilon^2, \\ \beta_2 = \dfrac{a}{b}\,(\tau \pm 2)\,\epsilon^2, \end{cases} \tag{84}$$

where $\tau, \tilde{u}, \tilde{v}$, and ξ are again given by (55). Next, we need to obtain $\eta(t)$ to approximate the profiles of the homoclinic solution. For this we integrate (60) to obtain

$$t(\eta) = (1 + \vartheta_{0010}\beta_1 + \vartheta_{0001}\beta_2)\,\eta + \vartheta_{1000}\frac{a}{b^2}\int\left[\tilde{u}\left(\tanh\left(\xi\left(\frac{a}{b}\epsilon\eta\right)\right)\right) \mp 2\right]\epsilon^2\,d\eta.$$

Here, β_1 and β_2 are given by (84) and from [3] we obtain that

$$\begin{aligned} \epsilon\frac{a}{b}\int\tilde{u}\left(\tanh\left(\xi\left(\frac{a}{b}\epsilon\eta\right)\right)\right)\,d\eta = {} & 2\tilde{\xi} - 6\tanh\left(\tilde{\xi}\right) + \left(\frac{18\,\mathrm{sech}^2(\tilde{\xi})}{7} + \frac{12}{7}\log\left(\cosh\left(\tilde{\xi}\right)\right)\right)\epsilon \\ & + \frac{9}{49}\left(4\tilde{\xi} - 9\tanh\left(\tilde{\xi}\right) + 5\tanh\left(\tilde{\xi}\right)\mathrm{sech}^2(\tilde{\xi})\right)\epsilon^2 \\ & + \frac{18\left(-21\,\mathrm{sech}^4(\tilde{\xi}) + 47\,\mathrm{sech}^2(\tilde{\xi}) + 8\log\left(\cosh\left(\tilde{\xi}\right)\right)\right)}{2401}\epsilon^3 \\ & + \mathcal{O}(\epsilon^4), \end{aligned} \tag{85}$$

where $\tilde{\xi} = \xi(\frac{a}{b}\eta\epsilon)$.

It remains to numerically solve the equation

$$t(\eta) - t = 0,$$

for η.

4.2.4 Codimension-one equilibrium bifurcations

To approximate the transcritical and Hopf curves and their corresponding equilibria in (56), one should substitute the expressions for β and the equilibrium coordinates into the expansions (58) and (59). It follows that the transcritical curve is approximated by

$$\begin{cases} x = \mathcal{H}\,(0,0,0,0)\,, \\ \alpha = K(0,\epsilon), \end{cases}$$

for $|\epsilon|$ small. The Hopf curves are approximated by

$$\begin{cases} \omega = \epsilon, \\ x = \mathcal{H}\,(0,0,-\epsilon^2,0)\,, \\ \alpha = K\,(-\epsilon^2,0)\,, \end{cases}$$

and

$$\begin{cases} \omega = \epsilon, \\ x = \mathcal{H}\left(-\frac{\epsilon^2}{a}, 0, \epsilon^2, \frac{b}{a}\epsilon^2\right), \\ \alpha = K\left(\epsilon^2, \frac{b}{a}\epsilon^2\right), \end{cases}$$

for $\epsilon > 0$ small.

5 Examples

In this section, we will demonstrate the correctness of the center manifold reduction and the homoclinic asymptotics from Section 4 on four different models from [15, 19, 24, 25] in which Bogdanov–Takens bifurcation points are present. Using our implementation in `DDE-BifTool`, we will compute the local unfolding of the Bogdanov–Takens singularities and compare the emanating codimension one curves with the derived asymptotics. We will show a clear improvement to the homoclinic solutions by using the third-order instead of the first order asymptotics.

Additionally, following [3], we will also show that the approximation order of the homoclinic asymptotic lifts correctly to the parameter-dependent center manifold, see in particular Figures 3 and 4.

Lastly, we will perform numerical simulations near the bifurcation point under consideration. The reason for this extra step is twofold. Firstly, it will provide additional verification of the given results. Secondly, it seems to be a standard part in papers studying Bogdanov–Takens bifurcation points in DDEs. We will integrate the DDE directly, not the reduced system on the center manifold, which is often the case in literature. The simulation is performed using the Julia package `DifferentialEquations.jl` [30]. However, since in this section only the main results are given, we provide full details (including simulation results) in the Supplement. Furthermore, the source code of the examples has been included into the `DDE-BifTool` software package. Alltogether, this will hopefully provide a good starting point when considering other models.

5.1 Bogdanov–Takens bifurcation in a predator-prey system with double Allee effect

In [25] the following predator-prey model with double Allee effect and delay is considered

$$\begin{cases} \dot{x}(t) = \frac{rx}{x+n_0}\left(1-\frac{1}{K}\right)(x-m_0) - \frac{cxy}{x+\varrho y}, \\ \dot{y}(t) = -dy + \frac{c_1 x(t-\tau)y}{x(t-\tau)+\varrho y(t-\tau)}. \end{cases} \tag{86}$$

Here, the time delay $\tau \geq 0$ is introduced due to the fact that the reproduction of predator after consuming the prey is not instantaneous, but is mediated by some time lag required for gestation. The variables and parameters occurring in (86) have the following meaning:

- $x, y \colon \mathbb{R} \to \mathbb{R}$ denote the prey and predator population densities, respectively.
- r denotes the maximum prey population growth in absence of the Allee effect.
- K is the carrying capacity of the environment.
- m_0 is the Allee threshold.
- n_0 is the auxiliary parameter in order to quantify the strength of the Allee effect.

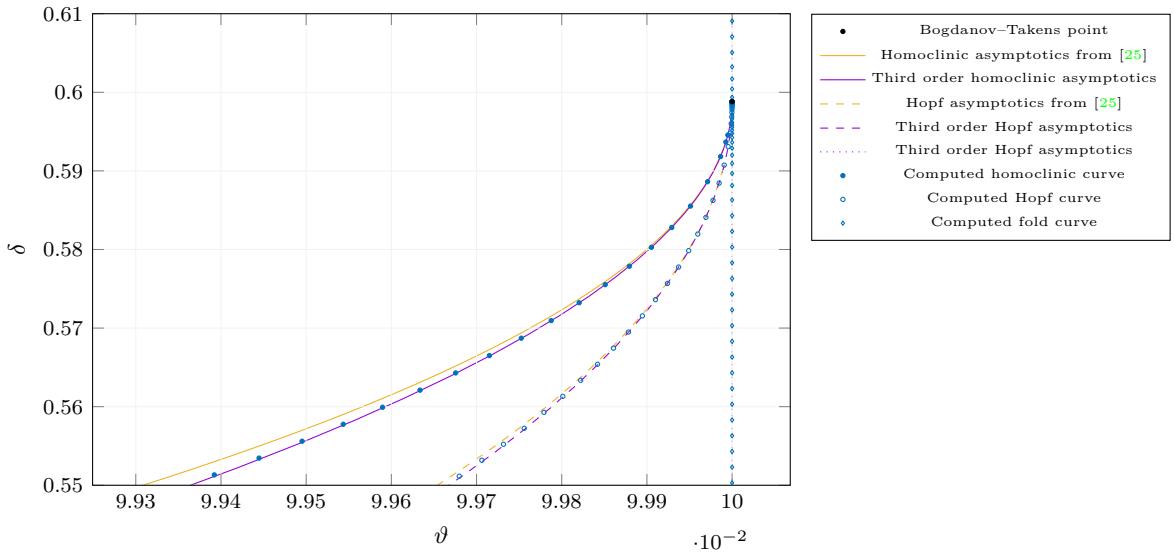

Figure 1: Bifurcation diagram near the derived generic Bogdanov-Takens point in (87) comparing computed codimension one using `DDE-BifTool` with the asymptotics obtained in this paper and in [25].

- c denotes the capturing rate of the predator.
- ϱ is the half capturing saturation constant.
- c_1 is the conversion rate of prey into predators biomass.
- d is the per capita predator mortality rate.

Following [25], let $(x, y, t) = \left(K\bar{x}, \frac{K}{\varrho}\bar{y}, \frac{\bar{t}}{r}\right)$, and immediately dropping the bars again for readability, then (86) becomes

$$\begin{cases} \dot{x}(t) = x\left(\dfrac{(1-x)(x-\gamma)}{x+\vartheta} - \dfrac{\alpha y}{x+y}\right), \\ \dot{y}(t) = \delta y\left(-1 + \dfrac{mx(t-\tau)}{x(t-\tau)+y(t-\tau)}\right), \end{cases} \tag{87}$$

where $\vartheta = \frac{n_0}{K}, \alpha = \frac{c}{r\varrho}, \gamma = \frac{m_0}{K}, \delta = \frac{d}{r}$ and $m = \frac{c_1}{d}$. In [25] the phase-space is (although possible) unnecessarily enlarged to include a parameter. The resulting extended system is used to derive conditions for a Bogdanov–Takens bifurcation to occur and obtain the normal form on the center manifold. In our opinion, one should only use the extended system to prove the existence of a parameter-dependent center manifold, from which it then follows that we can perform the normalization as described in Section 3. Thus, solving (87) for a double equilibrium with respect to (x, y) and ϑ yields

$$\begin{cases} x_0 = \dfrac{m+\alpha-m\alpha+m\gamma}{2m}, \\ y_0 = (m-1)x_0, \\ \vartheta_0 = \dfrac{(\alpha+m(1-\alpha+\gamma))^2 - 4m^2\gamma}{4(m-1)m\alpha}, \end{cases} \tag{88}$$

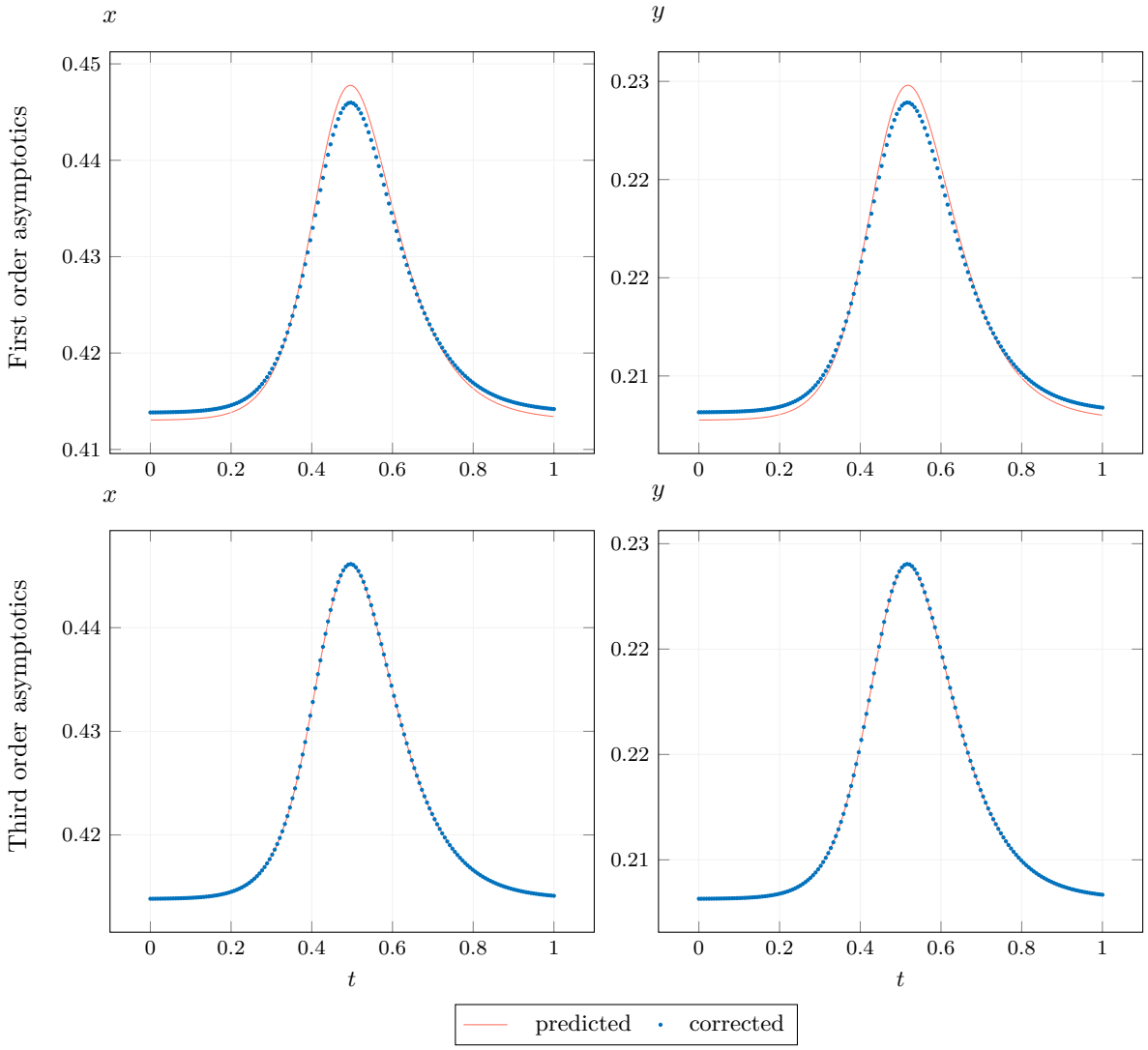

Figure 2: Comparison between the profiles of the first and third-order asymptotics from Section 4.1.9 with the Newton correct solutions near the generic Bogdanov–Takens bifurcation in (87) with the perturbation parameter set to $\epsilon = 0.3$.

for $m \neq 1$. The parameters should of course be chosen such that x and y are non-negative. The characteristic matrix at $(x, y, \vartheta) = (x_0, y_0, \vartheta_0)$ becomes

$$\Delta(z) = \begin{pmatrix} z + \dfrac{1-m}{m^2}\alpha & \dfrac{\alpha}{m^2} \\ -\dfrac{(m-1)^2\delta}{m}e^{-z\tau} & z + \dfrac{(m-1)\delta}{m}e^{-z\tau}, \end{pmatrix}$$

for $m \neq 0$. A simple calculation confirms indeed that $\det \Delta(0) = 0$. Furthermore, $\det \Delta'(0) = \frac{(m-1)(m\delta-\alpha)}{m^2}$ and $\det \Delta''(0) = 2 - \frac{2(m-1)\delta\tau}{m}$. Thus, for $(x, y, \vartheta, \delta) = (x_0, y_0, \vartheta_0, \delta_0)$, with $\delta_0 = \frac{\alpha}{m}$, we have a double zero eigenvalue, provided that $\det \Delta''(0) \neq 0$. To confirm their analytical findings in [25]

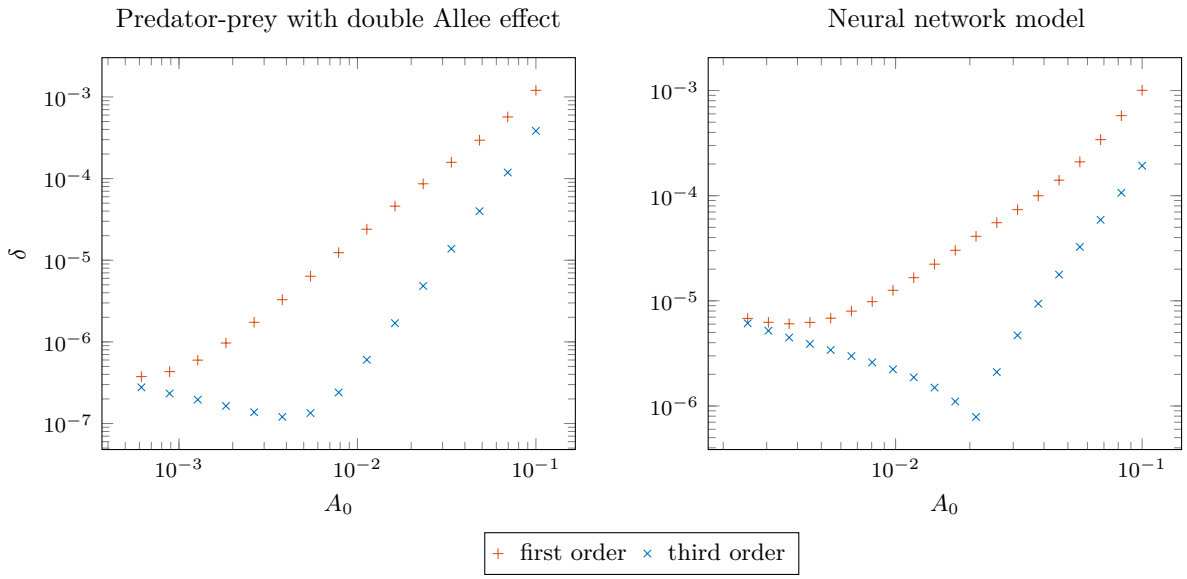

Figure 3: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution to the defining system [32] for the homoclinic orbit and the Newton corrected solution.

numerically, the parameters $\gamma = 0.15, \alpha = 0.9$ and $m = 1.50298303$ are fixed and (ϑ, δ) are taken as unfolding parameters. For these parameters, we indeed have that $\det \Delta''(0) \neq 0$. Calculating the normal form coefficients a and b reveal that,

$$a \approx 0.1479, \qquad b \approx 1.4457.$$

Thus, the codimension two Bogdanov–Takens bifurcation is non-degenerate. Furthermore, since the equilibrium (x_0, y_0) depends on the unfolding parameters, we are in the generic case. Using our implementation of the homoclinic predictor Section 4.1.9 in `DDE-BifTool`, we start the continuation of the homoclinic branch emanating from the Bogdanov–Takens point.

In Figure 1, we compare the parameters of the computed homoclinic branch with our third-order parameter asymptotics and the parameters for the homoclinic branch given in [25]. We observe that although the derivation of the asymptotics for the parameters is non-rigorous, it provides a good predictor. Nonetheless, our third-order asymptotics for the parameter values provides clearly a better approximation to the parameter values of the computed homoclinic branch.

Of course, most work lies in the derivation of the asymptotics for the homoclinic solutions in phase-space itself, which is not available in [25]. In Figure 2 we compare the first and third-order asymptotics for the profiles of the homoclinic solution. Here, we set the perturbation parameter to $\epsilon = 0.3$ such that both approximations converge, and we observe a visual difference.

However, a better way to compare the first and third-order asymptotics for the homoclinic solution numerically is through the convergence plots, similarly as in [3]. In Figure 3 we clearly see an improvement of using the third-order asymptotics over the first order asymptotics. We furthermore observe the standard V-shaped graphs due to round-off error.

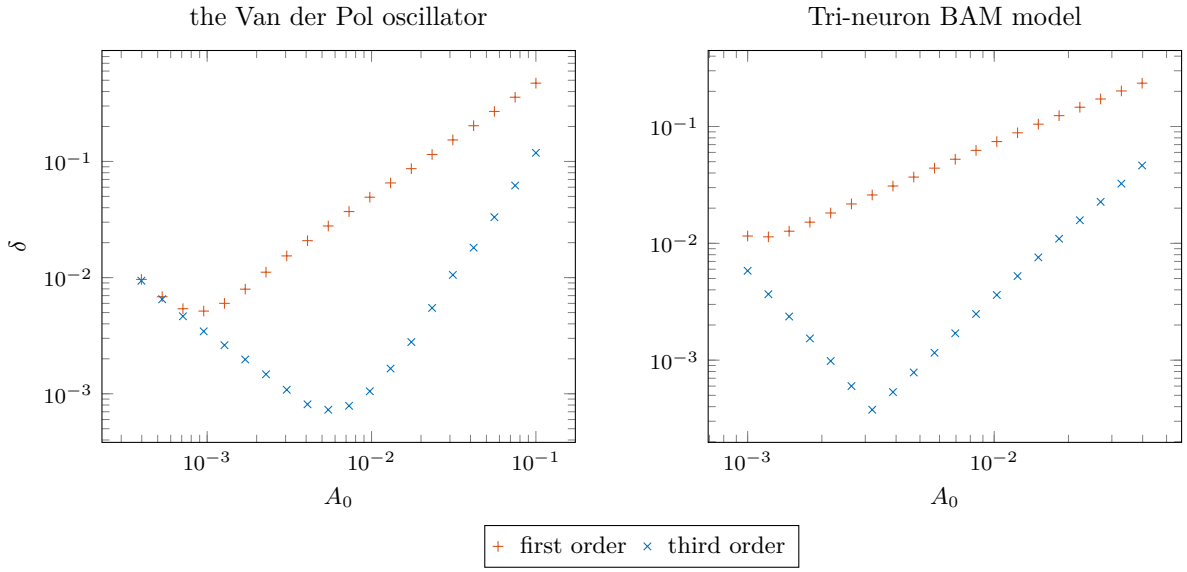

Figure 4: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution to the defining system for the homoclinic orbit and the Newton corrected solution.

5.2 Generic Bogdanov–Takens bifurcation in a neural network model

In this example, we will consider the model

$$\begin{cases} \mu \dot{u}_1(t) = -u_1(t) + q_{11}\alpha(u_1(t-T)) - q_{12}u_2(t-T) + e_1, \\ \mu \dot{u}_2(t) = -u_2(t) + q_{21}\alpha(u_1(t-T)) - q_{22}u_2(t-T) + e_2, \end{cases} \tag{89}$$

which describes the dynamics of a neural network consisting of an excitatory and inhibitory neuron [19]. The variables and parameters occurring in (89) have the following neurophysiological meaning:

- $u_1, u_2 : \mathbb{R} \to \mathbb{R}$ denote the total post-synaptic potentials of the excitatory and inhibitory neurons, respectively.
- $\mu > 0$ is a time constant characterizing the dynamical properties of cell membrane.
- $q_{ik} \geq 0$ represent the strength of the connection line from the kth neuron to the ith neuron.
- $\alpha : \mathbb{R} \to \mathbb{R}$ is the transfer function which describes the activity generation of the excitatory neuron as a function of its total potential u_1. The function α is smooth, increasing and has a unique turning point at $u_1 = \theta$. The transfer function corresponding to the inhibitory neuron is assumed to be the identity.
- $T \geq 0$ is a time delay reflecting synaptic delay, axonal and dendritic propagation time.
- e_1 and e_2 are external stimuli acting on the excitatory and inhibitory neuron, respectively.

Following [19] we consider the equation (89) with

$$\alpha(u_1) = \frac{1}{1+e^{-4u_1}} - \frac{1}{2}, \qquad q_{11} = 2.6, \qquad q_{21} = 1.0, \qquad q_{22} = 0.0,$$
$$\mu = 1.0, \qquad T = 1.0, \qquad e_2 = 0.0,$$

and $Q := q_{12}$, $E := e_1$ as bifurcation parameters. Substituting into (89) yields

$$\begin{cases} \dot{u}_1(t) = -u_1(t) + 2.6\alpha(u_1(t-1)) - Qu_2(t-1) + E, \\ \dot{u}_2(t) = -u_2(t) + \alpha(u_1(t-1)). \end{cases} \tag{90}$$

Notice that for any steady-state we have the symmetry $(u_1, u_2, E) \to (-u_1, -u_2, -E)$. It is easy to explicitly derive that the system has a double eigenvalue zero for

$$\begin{cases} u_1(t) = \dfrac{1}{4}\log\left(\dfrac{8-\sqrt{39}}{5}\right) \approx -0.2617, \\ u_2(t) = -\dfrac{1}{2}\sqrt{\dfrac{3}{13}} \approx -0.2402, \\ Q = \dfrac{13}{10}, \\ E = \dfrac{\sqrt{39} - 10\,\mathrm{atanh}\sqrt{\frac{3}{13}}}{20} \approx 0.0505. \end{cases} \tag{91}$$

The dependence of the equilibria on the parameters (Q, E) yields a generic Bogdanov–Takens bifurcation, see [19]. Notice that the normal form reduction in [19] is incorrect, which leads to the normal form for a transcritical Bogdanov–Takens bifurcation.

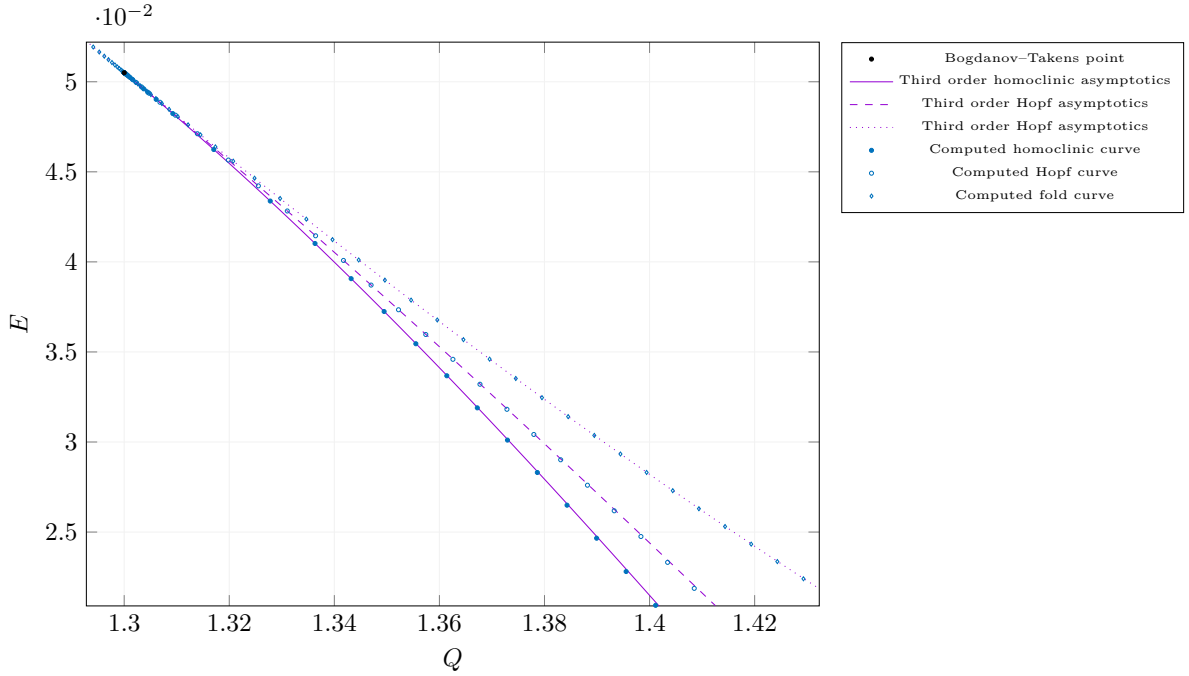

Figure 5: Bifurcation diagram near the derived generic Bogdanov-Takens point in (90) comparing computed codimension one curves using `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.1.9.

In Figure 5 we compared the computed codimension one curves emanating from a generic Bogdanov–Takens point in (90) using our implementation in `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.1.9. We see that the predicted and computed curves are almost

indistinguishable. Furthermore, in Figure 3, we see that using the third-order homoclinic asymptotics over the first order is clearly superior.

5.3 Transcritical Bogdanov–Takens bifurcation in the Van der Pol oscillator with delay feedback

We consider the Van der Pol oscillator with delay feedback [24] given by

$$\ddot{x}(t) + \epsilon(x^2(t) - 1)\dot{x}(t) + x(t) = \epsilon g(x(t - \tau)), \tag{92}$$

where $\epsilon > 0$ is a parameter, $\tau > 0$ is a delay and $g : \mathbb{R} \to \mathbb{R}$ is a smooth function with $g(0) = 0$ and $g'(0) \neq 0$. We rewrite the Van der Pol equation (92) as

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \epsilon g(x_1(t - \tau)) - \epsilon(x_1^2 - 1)x_2 - x_1. \end{cases} \tag{93}$$

Rescaling time with $t \to \dfrac{t}{\tau}$ to normalize the delay yields

$$\begin{cases} \dot{x}_1 = \tau x_2, \\ \dot{x}_2 = \tau \left(\epsilon g(x_1(t - 1)) - \epsilon(x_1^2 - 1)x_2 - x_1 \right). \end{cases} \tag{94}$$

This allows us to treat τ as a bifurcation parameter.

Following [24], we consider (92) with

$$g(x) = \frac{e^x - 1}{c_1 e^x + c_2},$$

where $c_1 = \dfrac{1}{4}$ and $c_2 = \dfrac{1}{2}$. Then the trivial equilibrium undergoes a transcritical Bogdanov–Takens bifurcation at parameter values $(\epsilon, \tau) = (0.75, 0.75)$, see [24] and the supplement.

In Figure 6 we compared the computed codimension one curves emanating from a transcritical Bogdanov–Takens point in (94) using our implementation in `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.2.3. We see that the predicted and computed curves are almost indistinguishable. Furthermore, in Figure 4 we see that using the third-order homoclinic asymptotics over the first order is clearly superior.

5.4 Transcritical Bogdanov–Takens bifurcation in a tri-neuron BAM neural network model

We consider a three-component system of a tri-neuron bidirectional associative memory (BAM) neural network model with multiple delays [15]. The architecture of this BAM model is illustrated in Figure 7.

In this model, there is only one neuron with the activation function f_1 on the I-layer and there are two neurons with respective activation functions f_2 and f_3 on the J-layer. It is assumed that the time delay from the I-layer to the J-layer is τ_1, while the time delay from the J-layer to the I-layer is τ_2. Then the network can be described by the following delay differential equation

$$\begin{cases} \dot{x}_1(t) &= -\mu_1 x_1(t) + c_{21} f_1(x_2(t - \tau_2)) + c_{31} f_1(x_3(t - \tau_2)), \\ \dot{x}_2(t) &= -\mu_2 x_2(t) + c_{12} f_2(x_1(t - \tau_1)), \\ \dot{x}_3(t) &= -\mu_3 x_3(t) + c_{13} f_3(x_1(t - \tau_1)). \end{cases} \tag{95}$$

Here

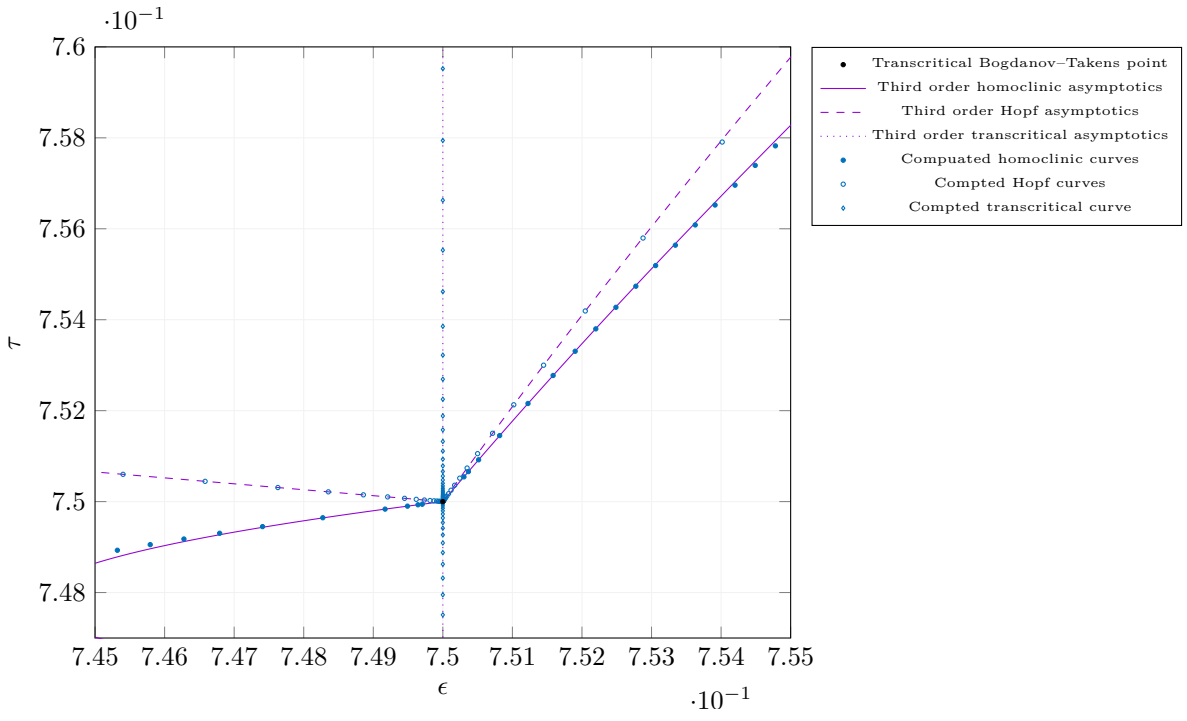

Figure 6: Bifurcation diagram near the derived generic Bogdanov-Takens point in (94) comparing computed codimension one curves using `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.2.3.

- $x_i(t)\,(i=1,2,3)$ denote the state of the neuron at time t;
- $\mu_i (i=1,2,3)$ describe the attenuation rate of internal neurons processing on the I-layer and the J-layer and $\mu_i > 0$;
- the real constants $c_{i1}$and $c_{1i}\,(2,3)$ denote the neurons in two layers: the I-layer and the J-layer.

Letting $u_1(t) = x_1(t-\tau_1), u_2(t) = x_2(t), u_3(t) = x_3(t)$ and $\tau = \tau_1 + \tau_2$, then system (95) is equivalent to the following system

$$\begin{cases} \dot{u}_1(t) &= -\mu_1 u_1(t) + c_{21} f_1(u_2(t-\tau)) + c_{31} f_1(u_3(t-\tau)), \\ \dot{u}_2(t) &= -\mu_2 u_2(t) + c_{12} f_2(u_1(t)), \\ \dot{u}_3(t) &= -\mu_3 u_3(t) + c_{13} f_3(u_1(t)). \end{cases} \tag{96}$$

In [15] conditions for (96) are derived for the origin to have a double zero eigenvalue, while all other eigenvalues have negative real parts. Since the provided expression did not give the correct results, we corrected the given expression from [15]. We postpone the proof of the next two lemma's to the Supplement.

Lemma 10. *Assume that* $f_i(0) = 0\,(i=1,2,3)$, $f_i'(0) \neq 0\,(i=1,2,3)$ *and* $\mu_2 \neq \mu_3$, *then the steady-*

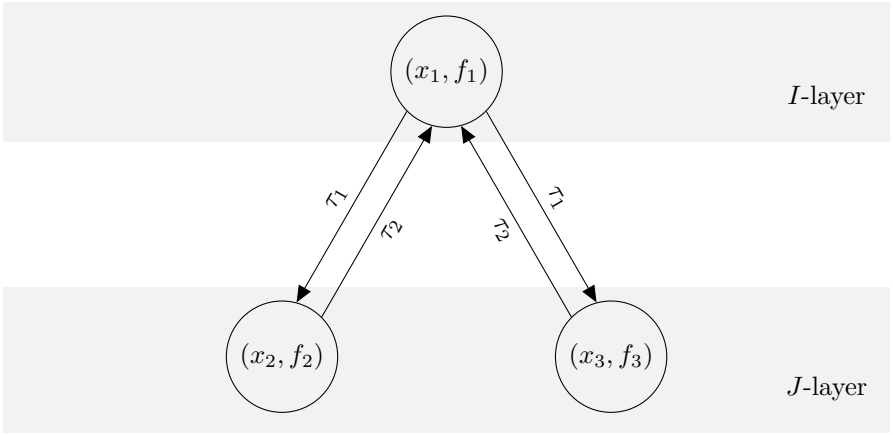

Figure 7: The graph of architecture for model (95).

state $(u_1, u_2, u_3) = (0, 0, 0)$ *has a double zero eigenvalue at*

$$c_{21} = c_{21}^0 = \frac{\mu_2^2\left(\mu_1\left(\mu_3\tau + 1\right) + \mu_3\right)}{c_{12}\left(\mu_2 - \mu_3\right) f_1'(0) f_2'(0)},$$
$$c_{31} = c_{31}^0 = \frac{\mu_3^2\left(\mu_1\left(\mu_2\tau + 1\right) + \mu_2\right)}{c_{13}\left(\mu_3 - \mu_2\right) f_1'(0) f_3'(0)}.$$

Lemma 11. Correction to [15, Lemma 3]. *Let* $(c_{21}, c_{31}) = (c_{21}^0, c_{31}^0)$,

$$\omega_0 = \frac{\sqrt{-\mu_1^2 - \mu_2^2 - \mu_3^2 + \sqrt{\zeta_0}}}{\sqrt{2}} \tag{97}$$

and $0 < \tau < \tau_0$, *where* τ_0 *is the minimum positive solution to the nonlinear equation*

$$\tan(\tau\omega_0) = \frac{b_0\zeta_1 - a_0\zeta_2}{a_0\zeta_1 + b_0\zeta_2}, \tag{98}$$

with

$$\begin{aligned}
a_0 &= -\mu_1\mu_2\mu_3,\\
b_0 &= -\omega_0(\mu_2\mu_3 + \mu_1(\mu_2 + \mu_3 + \mu_2\mu_3\tau)),\\
\zeta_0 &= \mu_1^4 + (\mu_2^2 + \mu_3^2)^2 + 8\mu_1\mu_2\mu_3(\mu_2 + \mu_3 + \mu_2\mu_3\tau) +\\
&\qquad 2\mu_1^2(\mu_3^2 + 4\mu_2\mu_3(1 + \mu_3\tau) + \mu_2^2(1 + 2\mu_3\tau(2 + \mu_3\tau))),\\
\zeta_1 &= \mu_1\mu_2\mu_3 - (\mu_1 + \mu_2 + \mu_3)\omega_0^2,\\
\zeta_2 &= \mu_2\mu_3\omega_0 + \mu_1(\mu_2 + \mu_3)\omega_0 - \omega_0^3.
\end{aligned}$$

Then the center manifold near the Bogdanov–Takens point is locally attractive.

For the numerical verification we consider, as in the simulations in [15, Example 1], the system (96) with the activation functions

$$f_1(x) = \tanh(x) + 0.1x^2, \quad f_2(x) = f_3(x) = \tanh(x),$$

and parameter values

$$\mu_1 = 0.1, \mu_2 = 0.3, \mu_3 = 0.2, c_{12} = c_{13} = 1, \tau = 5.$$

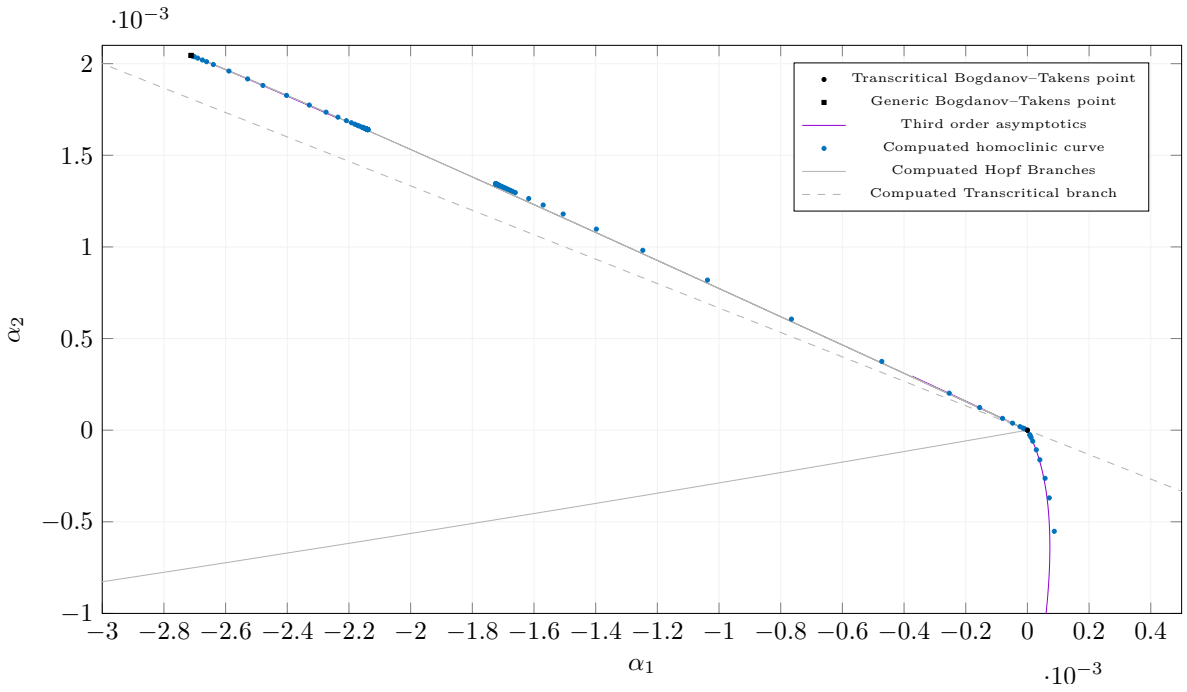

Figure 8: Bifurcation diagram near the transcritical Bogdanov–Takens bifurcation and a generic Bogdanov–Takens in (96) comparing computed codimension one using `DDE-BifTool` with the asymptotics obtained in this paper.

Then, from Lemma 10, we obtain two critical values

$$(c_{21}^0, c_{31}^0) = (0.36, -0.22),$$

at which there is a transcritical Bogdanov–Takens point. Furthermore, since $\tau < \tau_0 \approx 5.4320$ the center manifold is attractive. In fact, we will show in the Supplement that the center manifold is attractive for $0 < \tau < 13.2309348879375$. We write the system (96) as

$$\begin{cases} \dot{u}_1(t) &= -\mu_1 u_1(t) + \left(c_{21}^0 + \alpha_1\right) f_1(u_2(t-\tau)) + \left(c_{31}^0 + \alpha_2\right) f_1(u_3(t-\tau)), \\ \dot{u}_2(t) &= -\mu_2 u_2(t) + c_{12} f_2(u_1(t)), \\ \dot{u}_3(t) &= -\mu_3 u_3(t) + c_{13} f_3(u_1(t)), \end{cases} \tag{99}$$

where (α_1, α_2) are the new parameter values such that at $(\alpha_1, \alpha_2) = (0, 0)$ we have a Bogdanov-Takens bifurcation. The critical normal form coefficients

$$(a, b) \approx (0.0012, -0.0135),$$

indicate stable cycles. In Figure 8, we have plotted the local unfolding of the transcritical Bogdanov-Takens bifurcation using our implementation in DDE-BifTool to start the continuation of the transcritical, Hopf, and homoclinic codimension one bifurcation curves. Furthermore, using the ability to detect codimension two bifurcation points while continuing Hopf points, we discovered another Bogdanov–Takens point. From the transversality conditions, we see the secondary Bogdanov–Takens point is of the generic case. Thus, we can start continuation of the homoclinic orbits emanating from this point as well. We see that the numerically continued homoclinic curves in the upper half plane only exists in a

very small parameter region. Without our predictor, this homoclinic curve would be extremely difficult to locate. Lastly, in Figure 4 the first and third-order homoclinic asymptotics are compared. Here we again see expected improvements of using the third-order homoclinic asymptotics. We refer to the Supplement to see the beautiful homoclinic orbits along the homoclinic curve in the lower half plane, and for an overall more detailed treatment of this example. In fact, we will show that the transcritical Bogdanov–Takens and generic Bogdanov–Takens points are connected, not only by a Hopf curve, but also through a homoclinic curve.

6 Concluding remarks

We have provided explicit formulas needed to initialize the codimension one equilibrium and homoclinic bifurcation curves emanating from the generic and transcritical codimension two Bogdanov–Takens bifurcation points in classical DDEs. Applications to four different models from the literature are given, confirming the correctness of the derivation of the time-reparametrization parameter-dependent center manifold transformation and the codimension one asymptotics.

By extending the normalization technique to include the time-reparametrization we are allowed to use orbital normal forms, instead of only smooth normal forms. One benefit of this approach is the reusability of the codimension one curves emanating from the universal unfolding of the Bogdanov–Takens codimension two bifurcation. Indeed, by a simple transformation (82), we obtain the homoclinic asymptotics for the transcritical Bogdanov–Takens bifurcation.

In this paper, we have restricted to the class of classical DDEs, and for the applications to the class of discrete DDEs. However, the proof in [2] of the existence of a smooth parameter-dependent center manifold is given in the general context of perturbation theory for dual semigroups (sun-star calculus). Therefore, the applicability of this result extends beyond classical DDEs. For example, in [36] and [14] the technique was used to calculate the critical normal form coefficients for Hopf and Hopf-Hopf bifurcations occurring in neural field models with propagation delays. For these models, sun-reflexivity is lost, which is typical for delay equations in abstract spaces or with infinite delay. However, it is often possible to overcome this functional analytic complication, so dual perturbation theory can still be employed successfully [11, 12, 23, 36]. It follows that the derived coefficients in Section 4 are valid in these settings as well.

Similarly, in [34] it is demonstrated that formally the normalization method still works for state-dependent delay differential equations. However, to employ our formulas in this situation, one first needs to implement the continuation of homoclinic orbits for state-dependent DDEs. Of course, the asymptotics are still useful to see where the homoclinic orbit should be located. Furthermore, the asymptotics can be used to numerically approximate the homoclinic solutions by periodic orbits with a large period.

Returning to the setting of classical DDEs, the most obvious next challenge is to derive normal forms for bifurcations of periodic orbits by generalizing [9, 10, 29]. The resulting formulas can then be implemented in `DDE-BifTool` to facilitate numerical bifurcation analysis of periodic orbits in supported types of classical DDEs.

SUPPLEMENTARY MATERIALS FOR:

Bifurcation analysis of Bogdanov-Takens bifurcations in delay differential equations

M.M. Bosschaert* Yu.A. Kuznetsov†

October 7, 2022

In this supplement, we will provide a full walk-through of the examples given in Section 5 with the open source bifurcation software package `DDE-BifTool`[1] [35]. Additionally, the Julia code, used for the numerical simulation, with the package `DifferentialEquations.jl`[2] [30] is shown. This allows other researchers to replicate the findings in the main text fully. The given code can easily be modified to study other DDE models undergoing generic or transcritical Bogdanov-Takens bifurcations. While studying these models, some new results were obtained.

The code in this supplement has also been included into the `DDE-BifTool` package on the sourceforge repository and can be executed without the need to copy and paste. In this supplement, we mainly focus on the initialization and continuation of the various codimension one equilibrium and homoclinic bifurcation curves emanating from the Bogdanov–Takens points and on numerical simulation near the bifurcation points. The online tutorials, as well as the manual and its references, provide a comprehensive overview of `DDE-BifTool`'s capabilities and functionality.

Note that reading this supplement may feel, at times, somewhat repetitive. We see this as a positive sign. Indeed, our predictors need little to none adjustment before starting continuation of the codimension one curves emanating from the generic and transcritical Bogdanov–Takens bifurcation.

To follow the examples below, we recommend using the `DDE-BifTool` package supplied with the article. Also, note that the `MATLAB` code has been tested on `MATLAB` 2020b and `GNU Octave` 6.4.0 using `GNU Octave` symbolic package version 2.9.0. Different results may occur on other versions of `MATLAB` or `GNU Octave`.

S1 Generic Bogdanov–Takens bifurcation in a predator-prey system with double Allee effect

In [25] the following predator-prey model with double Allee effect and delay is considered

$$\begin{cases} \dot{x}(t) = \dfrac{rx}{x+n_0}\left(1-\dfrac{1}{K}\right)(x-m_0) - \dfrac{cxy}{x+\varrho y}, \\ \dot{y}(t) = -dy + \dfrac{c_1 x(t-\tau)y}{x(t-\tau)+\varrho y(t-\tau)}. \end{cases} \tag{S1}$$

*Department of Mathematics, Hasselt University, Diepenbeek Campus, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium (maikel.bosschaert@uhasselt.be).

†Department of Mathematics, Utrecht University, Budapestlaan 6, 3508 TA Utrecht, The Netherlands and Department of Applied Mathematics, University of Twente, Zilverling Building, 7500AE Enschede, The Netherlands (I.A.Kouznetsov@uu.nl).

[1] http://ddebiftool.sourceforge.net/

[2] https://github.com/SciML/DifferentialEquations.jl

Here, the time delay $\tau \geq 0$ is introduced due to the fact that the reproduction of predator after consuming the prey is not instantaneous, but is mediated by some time lag required for gestation. The variables and parameters occurring in (S1) have the following meaning:

- $x, y \colon \mathbb{R} \to \mathbb{R}$ denote the prey and predator population densities, respectively.
- r denotes the maximum prey population growth in absence of the Allee effect.
- K is the carrying capacity of the environment.
- m_0 is the Allee threshold.
- n_0 is the auxiliary parameter in order to quantify the strength of the Allee effect.
- c denotes the capturing rate of the predator.
- ϱ is the half-capturing saturation constant.
- c_1 is the conversion rate of prey into predators biomass.
- d is the per capita predator mortality rate.

Following [25] let $(x, y, t) = \left(K\bar{x}, \frac{K}{\varrho}\bar{y}, \frac{\bar{t}}{r}\right)$, and immediately dropping the bars again for readability, then (S1) becomes

$$\begin{cases} \dot{x}(t) = x\left(\dfrac{(1-x)(x-\gamma)}{x+\vartheta} - \dfrac{\alpha y}{x+y}\right), \\ \dot{y}(t) = \delta y\left(-1 + \dfrac{mx(t-\tau)}{x(t-\tau)+y(t-\tau)}\right), \end{cases} \tag{S2}$$

where $\vartheta = \frac{n_0}{K}, \alpha = \frac{c}{r\varrho}, \gamma = \frac{m_0}{K}, \delta = \frac{d}{r}$ and $m = \frac{c_1}{d}$.

Thus, $(x, y, \vartheta, \delta) = (x_0, y_0, \vartheta_0, \delta_0)$, with $\delta_0 = \frac{\alpha}{m}$, we have a double zero eigenvalue, provided that $\det \Delta''(0) \neq 0$. To confirm their analytical findings in [25] numerically, the parameters $\gamma = 0.15, \alpha = 0.9$ and $m = 1.50298303$ are fixed and (ϑ, δ) are taken as unfolding parameters. For these parameters, we indeed have that $\det \Delta''(0) \neq 0$.

Remark 12. The `MATLAB` files for this demonstration can be found in the directory `demos/tutorial/VII/neural_network_model` relative to the main directory of the `DDE-BifTool` package.

S1.1 Generate system files

Before we start to analyze the system with `DDE-BifTool`, we first create a *system file*. This file contains the definition of the system (S2), the standard derivatives needed for calculation of the eigenvalues and eigenvectors, the continuation of bifurcation points, cycles, and also the multilinear forms, see [2, Section 6], used for the calculation of the coefficients of the critical and parameter-dependent normal forms and center manifold transformation. Alternatively, one can only supply the system itself, see Listing S2. Then finite difference is used to approximate the derivatives. However, this is less efficient and less accurate, and therefore not recommended. A separate script `gen_sym_predator_prey.m` is used to create a system file. The most important parts of this script are listed and discussed below.

```
18 %% Add paths and load sym package if GNU Octave is used
19 clear
20 ddebiftoolpath='../../../../';
21 addpath(strcat(ddebiftoolpath,'ddebiftool'),...
22     strcat(ddebiftoolpath,'ddebiftool_extra_symbolic'));
```

```
if dde_isoctave()
    pkg load symbolic
end
%% Create parameter names as strings and define fixed parameters
% The demo has the parameters |theta|, |delta| and |tau|
parnames={'theta','delta','tau'};
%% Create symbols for parameters, states and delays states
% The array |par| is the array of symbols in the same order as parnames.
% Due to the following two lines we may, for example, use either theta or
% par(1) to refer to the delay.
syms(parnames{:});       % create symbols for theta, delta and tua
par=cell2sym(parnames);  % now theta is par(1) etc
%% Define system using symbolic algebra
syms x xt y yt % create symbols for u1(t) u1(t-tau), u2(t), u2(t-tau)
m = 1.502983803;
alpha = 0.9;
gamma = 0.15;
dx_dt = x*((1-x)*(x-gamma)/(x+theta) - alpha*y/(x+y))
dy_dt = delta*y*(-1 + m*xt/(xt+yt))
model = [dx_dt;dy_dt];
vars = [x,xt;y,yt];
modelname = 'predator_prey';
%% Differentiate and generate code, exporting it to sym_FHN_mf (multi-linear forms)
dde_sym2funcs(model, vars, par, 'filename',strcat('sym_',modelname,'_mf'), ...
    'directional_derivative',false);
%% Differentiate and generate code, exporting it to sym_FHN (directional
  ↪  derivatives)
dde_sym2funcs(model, vars, par, 'filename',strcat('sym_',modelname), ...
    'directional_derivative',true);
```

The variable `ddebiftoolpath` is directed to the `DDE-BifTool` main folder, which should have been extracted somewhere on the computer. Here, a path relative to the current working directory is used. Note that although we only use the parameters (θ, δ) as unfolding parameters, in the current version of `DDE-BifTool`, we also need to include the delay(s) in the list of parameters. After running the script, the function `dde_sym2funcs` creates two system files `sym_predator_prey_mf.m` and `sym_predator_prey.m`. The first file `sym_predator_prey_mf.m` implements the higher order derivatives as multilinear forms, as explained in [2, Section 6], and therefore will be the only file used. The second file `sym_predator_prey.m` uses directional derivatives to implement the higher order derivatives. The directional derivatives approach *formally* allows the use of state-dependent delays, see [34]. Although both approaches yield (up to rounding errors) identical normal form coefficients and center manifold transformations, multilinear forms are more efficient to compute.

S1.2 Loading the `DDE-BifTool` package

Now that a system file is created, we continue with `DDE-BifTool` to analyze (S2) numerically. The code in the following sections highlight the important parts of the file `predator_prey.m`. The package `DDE-BifTool` consists of a set of `MATLAB` routines. Thus, in order to start using `DDE-BifTool`, we only need to add `DDE-BifTool` (sub)directories to the search path. There are four subdirectories added to the search path:

ddebiftool Containing the core files of `DDE-BifTool`.

```
%% Clean workspace and add DDE-BifTool scripts to the MATLAB search path
clear;      % clear variables
close all;        % close figures
ddebiftoolpath='../../../../';
addpath(strcat(ddebiftoolpath,'ddebiftool'),...
        strcat(ddebiftoolpath,'ddebiftool_extra_psol'),...
        strcat(ddebiftoolpath,'ddebiftool_extra_nmfm'),...
        strcat(ddebiftoolpath,'ddebiftool_utilities'),...
        '../');
```

Listing S1: Code to add `DDE-BifTool` scripts to the search path.

ddebiftool_extra_psol An extension for enabling continuation of periodic orbit bifurcations for delay-differential equations with constant or state-dependent delay.

ddebiftool_extra_nmfm An extension for normal form computation.

ddebiftool_utilities Containing various utilities.

S1.3 Set parameter names

The following code allows us to use `ind.theta` instead of remembering the index of the parameter β in the parameter array, and similarly for the other parameters.

```
%% Set parameter names
parnames={'theta','delta','tau'};
cind=[parnames;num2cell(1:length(parnames))];
ind=struct(cind{:});
```

In this way, fewer mistakes are likely to be made, and the code is easier to read.

S1.4 Initialization

Next, we set up the `funcs` structure, containing information about where the system and its derivatives are stored, a function pointing to which parameters are delays, and various other settings.

```
%% Set the funcs structure
% We load the precalculated multilinear forms. These have been generated
% with the file gen_sym_predator_prey.m.
funcs=set_symfuncs(@sym_predator_prey_mf,'sys_tau',@()ind.tau);
```

Alternatively, when no system files have been generated, one could initialize the system (S2) as in Listing S2.

Inspecting the output of the `funcs` handle gives.

```
>> funcs
```

```
%% Define funcs structure without symbolic derivatives
m = 1.502983803;
alpha = 0.9;
gamma = 0.15;
dx_dt = @(x,y,theta) x.*((1-x).*(x-gamma)./(x+theta) ...
                     - alpha.*y./(x+y))
dy_dt = @(y,xt,yt,delta) delta.*y.*(-1 + m.*xt./(xt+yt))
predator_prey_sys = @(xx,par)  ...
   [dx_dt(xx(1,1,:),xx(2,1,:),par(1,ind.theta,:)); ...
    dy_dt(xx(2,1,:),xx(1,2,:),xx(2,2,:),par(1,ind.delta,:))];
% % Set funcs structure
funcs = set_funcs('sys_rhs', predator_prey_sys, ...
    'sys_tau', @() ind.tau,...
    'x_vectorized', true, 'p_vectorized', true);
```

Listing S2: Code to define the system without a system file.

```
funcs =

  struct with fields:

                sys_rhs: @(x,p)dde_wrap_rhs(x,p,funcs.sys_rhs,funcs.x_vectorized,
                           funcs.p_vectorized)
              sys_ntau: @()0
               sys_tau: @()ind.tau
              sys_cond: @dde_dummy_cond
              sys_deri: @(x,p,nx,np,v)dde_gen_deriv(funcs.sys_dirderi,x,p,nx,np,v,1)
              sys_dtau: []
            sys_mfderi: {}
           sys_dirderi: {@(x,p,dx,dp)dde_dirderiv(derivbase,x,p,dx,dp,
                           order-ldirderi, 'nf',size(x,1),'hjac',
                           funcs.hjac(order))  [function_handle]}
           sys_dirdtau: []
          x_vectorized: 1
          p_vectorized: 1
                  hjac: @(ord)eps^(1/(2+ord))
    sys_cond_reference: 0
            lhs_matrix: @(sz)lhs_matrix(sz,funcs.lhs_matrix)
                tp_del: 0
     sys_deri_provided: 0
  sys_dirderi_provided: 0
```

The output shows that no derivative file is supplied. In this case, the derivatives are calculated using finite-difference approximations with the function `dde_dirderiv`. Again, we do not recommend using the latter approach. However, it can be useful for debugging purposes.

S1.5 Set parameter range

Since we are only interested here in the local unfolding, we restrict the allowed parameter range for the unfolding parameters. In practice, one may have physical restrictions which must be satisfied. Additionally, we also limit the maximum allowed step size during continuation. By doing so, we obtain more refined data to compare against our predictors.

```
%% Set bifurcation parameter range and step size bounds
brpars={'max_bound', [ind.delta 0.7],...
        'min_bound', [ind.delta 0.4],...
        'max_step', [ind.theta 0.002; ind.delta 0.002]};
```

S1.6 Stability and coefficients of the generic Bogdanov–Takens point

We manually construct a steady-state at the Bogdanov–Takens point derived in Section 5.1, see also [25].

```
%% Define constants
m = 1.502983803; alpha = 0.9; gamma = 0.15;

%% Define analytically derived Bogdanov-Takens point
% construct steady-state point
x0 = (m*gamma-m*alpha+m+alpha)/(2*m);
stst=dde_stst_create('x',[x0;  (m-1)*x0]);
stst.parameter(ind.theta)  = (m^2*gamma^2 - 2*m*(m*alpha + m - alpha)*gamma +
  ↪  (m*alpha - m - alpha)^2)/(4*m*alpha*(m-1));
stst.parameter(ind.delta) = alpha/m;
stst.parameter(ind.tau)   = 0.01;
% Calculate stability
method=df_mthod(funcs,'stst');
stst.stability=p_stabil(funcs,stst,method.stability);
stst.stability.l1
```

Inspecting the `stst.stability` structure yields

```
>> stst.stability.l1

ans =

   1.0e-06 *

   0.197590320692559
  -0.197590756777676

>>
```

The eigenvalues confirm that the point under consideration is indeed (an approximation to) a Bogdanov–Takens point. Next, we convert the steady-state point to a Bogdanov–Takens point and calculate the normal form coefficients with the function `nmfm_bt_orbital`, which implements the

coefficients derived in Section 4.1. For this, we need to set the argument `free_pars` to the unfolding parameter (θ, δ). These coefficients will be used to start the continuation of the codimension one branches emanating from the Bogdanov–Takens point.

```
%% Calculate normal form coefficients
free_pars=[ind.theta, ind.delta];
bt = p_tobt(funcs,stst);
bt = nmfm_bt_orbital(funcs, bt,'free_pars', free_pars);
```

The coefficients for the normal form, the time-reparametrization, and the center manifold transformation coefficients are stored in the `bt.nmfm` structure. Additionally, also the approximation to the parameters and center manifold transformation is given, which is used for the predictors.

```
>> bt.nmfm

ans =

  struct with fields:

          a: -0.145177185481861
          b: -1.446000370122628
  theta1000: -0.200700969579673
  theta0001: 5.128950206684711
        K10: [2x1 double]
        K01: [2x1 double]
        K02: [2x1 double]
        K11: [2x1 double]
        K03: [2x1 double]
       phi0: [1x1 struct]
       phi1: [1x1 struct]
      h0010: [1x1 struct]
      h0001: [1x1 struct]
      h2000: [1x1 struct]
      h1100: [1x1 struct]
      h0200: [1x1 struct]
      h1010: [1x1 struct]
      h1001: [1x1 struct]
      h0110: [1x1 struct]
      h0101: [1x1 struct]
      h0002: [1x1 struct]
      h0011: [1x1 struct]
      h3000: [1x1 struct]
      h2100: [1x1 struct]
      h1101: [1x1 struct]
      h2001: [1x1 struct]
      h0003: [1x1 struct]
      h1002: [1x1 struct]
      h0102: [1x1 struct]
          K: @(beta1,beta2)K10*beta1+K01*beta2+1/2*K02*beta2.^2
```

```
            +K11*beta1.*beta2+1/6*K03*beta2^3
        H: [function_handle]
```

Since the sign of ab is positive, we expect to find unstable periodic orbits nearby the Bogdanov–Takens point.

S1.7 Comparing profiles of computed and predicted homoclinic orbits

To test the homoclinic asymptotics from Section 4.1.9 we compare the first and third order asymptotics to the Newton corrected solution. For this, we use the function `C1branch_from_C2point`. This function returns a branch, which by default returns two initial corrected approximations in order to start continuation of the codimension one curve under consideration. By setting the argument `'predictor'` to `true` the approximations are left uncorrected. To make the comparison visually clear, we set the perturbation parameter $\epsilon = 0.3$ (`step = 0.3` in the code below). The code below produces Figure S1. The difference between the two approximations is clearly noticeable. While the first order asymptotics is close to the Newton corrected solution, the third order asymptotic is indistinguishable at this scale from the Newton corrected solution.

```
%% Plot comparing profiles computed and predicted homoclinic orbits
figure(1); clf; hold on;
cm=colormap('lines');
tiledlayout(2,2)
ylabels = {'$x$','$y$'};
step = 0.3;
for order=[1,3]

    [~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
        'codim2','BT','codim1','hcli','step',step,'predictor',true);

    [~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
        'codim2','BT','codim1','hcli','step',step);
    for j=1:2
        ax = nexttile; hold on; title(sprintf('order %d',order))

    plot(ax,hcli_br_approx.point(1).mesh,hcli_br_approx.point(1).profile(j,:),'*')
        plot(ax,hcli_br_correc.point(1).mesh,hcli_br_correc.point(1).profile(j,:))
        xlabel('$t$','Interpreter','LaTeX');
        ylabel(string(ylabels(j)),'Interpreter','LaTeX')
    end
end
```

S1.8 Continuation of the emanating codimension one curves

To continue the three codimension one curves emanating from the generic Bogdanov–Takens point, we can simply use the function `C1branch_from_C2point`, as shown in the code below. To monitor the continuation process, the argument `plot` must be set to `1`. The most important setting is the perturbation parameter (or multiple), `step` in the code below. If left out, default step sizes are defined. However, depending on the problem, no convergence may then be obtained.

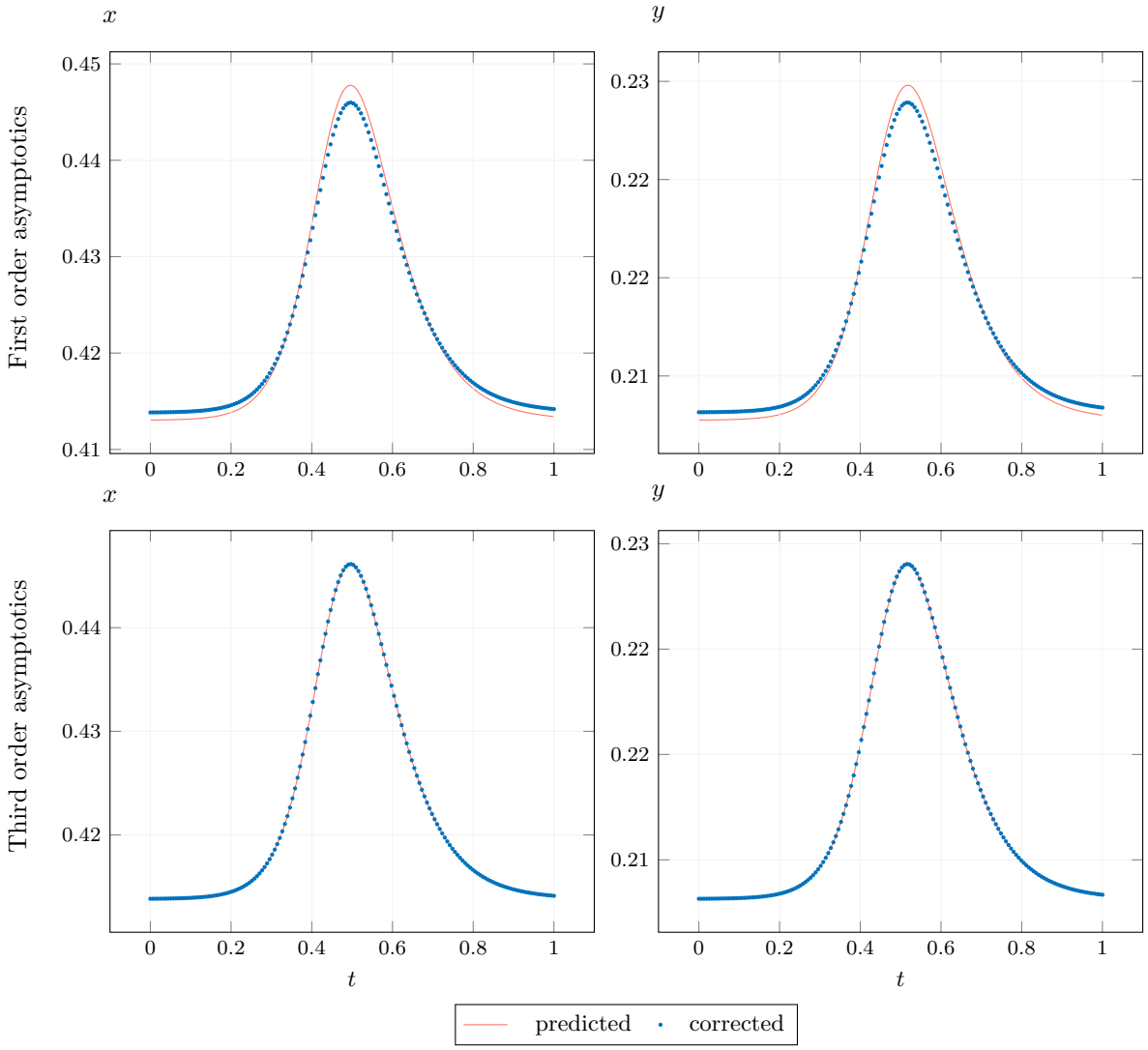

Figure S1: Comparison between the first and third-order asymptotics from Section 4.1.9 near the generic Bogdanov–Takens bifurcation in (S2) with the perturbation parameter set to $\epsilon = 0.3$.

```
%% Continue homoclinic solutions emanating from the BT point
[~,hcli_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli',brpars{:},'step',[0.1;0.11],'plot',0);
assert(all(suc(:)>0))
nop=300; hcli_br=br_contn(funcs,hcli_br,nop);assert(suc>0)

%% Continue Hopf curve
[~,hbr,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-05,'plot',0);
assert(all(suc(:)>0))
nop=300; [hbr,suc]=br_contn(funcs,hbr,nop);assert(suc>0)
```

```
%% Continue fold curve emanating from Bogdanov-Takens point
[~,fold_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold',brpars{:},'step',1e-03,'plot',0);
assert(all(suc(:)>0))
nop=300; [fold_br,suc]=br_contn(funcs,fold_br(1),nop);assert(suc>0)
fold_br = br_rvers(fold_br);
nop=300; [fold_br,suc]=br_contn(funcs,fold_br(1),nop);assert(suc>0)
```

S1.9 Predictors of the codimension one curves emanating from the Bogdanov–Takens point

Before we provide the bifurcation diagram in the next section, we first obtain the predictors for the codimension one curves. For this, we again use the function `C1branch_from_C2point`. We set the argument `predictor` to `1` and provide a range of perturbation parameters.

```
%% Predictors for homoclinic, Hopf, and fold curve
step = linspace(1e-4,0.6,50);
[~,hcli_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,'debug',0,...
    'codim2','BT','codim1','hcli','step',step,'predictor',true);
step = linspace(0.0001,0.1,50);
[~,hbr_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf','step',step,'predictor',true);
step = linspace(-0.05,0.05,40);
[~,fold_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold','step',step,'predictor',true);
hcli_br_pred_pm  = [getpars(hcli_br_pred, ind.theta), getpars(hcli_br_pred,
 ↪  ind.delta)];
hbr_pm_pred = [getpars(hbr_pred,ind.theta); getpars(hbr_pred,ind.delta)];
fold_br_pm_pred = [getpars(fold_br_pred,ind.theta);
 ↪  getpars(fold_br_pred,ind.delta)];

%% Predictors from [Jiao2021]
mu1 = -linspace(1e-5,0.003,250);
mu2 = -1.386173725.*mu1 - 0.9901240893*sqrt(mu1.*(mu1 - 3.652165730));
hbr_pm_pred_2021 = bt.parameter(free_pars)' + [mu1; mu2];
mu2 = -1.386173725.*mu1 - 1.386173725*sqrt(mu1.*(mu1 - 3.652165730));
hcli_pm_pred_2021 = bt.parameter(free_pars)' + [mu1; mu2];
```

In the last part of the code above, we added the asymptotics obtained from [25].

S1.10 Bifurcation diagram

The code below produces a similar figure as Figure S2 in `MATLAB`.

```
%% Plot the bifurcation curves with predictors
figure(3);clf;hold on
title(['Codimension 1 curves emanating from the generic', ...
    'Bogdanov-Takens point with predictors']);
```

```
plot(hcli_br_pm(1,:),hcli_br_pm(2,:),'.','Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br_pred_pm(:,1),hcli_br_pred_pm(:,2),'Color', cm(7,:), ...
    'DisplayName','Third order homoclinic asymptotics')
plot(hcli_pm_pred_2021(1,:), hcli_pm_pred_2021(2,:), 'Color',cm(3,:), ...
    'DisplayName','Asymptotics from Jiao2021');
plot(hbr_pm(1,:),hbr_pm(2,:),'o','Color',cm(1,:), ...
    'DisplayName','Hopf branch')
plot(hbr_pm_pred(1,:),hbr_pm_pred(2,:),'--','Color',cm(7,:), ...
    'DisplayName','Hopf asymptotics')
plot(hbr_pm_pred_2021(1,:),hbr_pm_pred_2021(2,:),'--','Color',cm(3,:), ...
    'DisplayName','Hopf asymptotics from Jiao2021')
plot(fold_br_pm(1,:),fold_br_pm(2,:),'d','Color',cm(1,:), ...
    'DisplayName','Fold branch')
plot(fold_br_pm_pred(1,:),fold_br_pm_pred(2,:),':','Color',cm(7,:), ...
    'DisplayName','Fold asymptotics')
plot(bt.parameter(ind.theta),bt.parameter(ind.delta),'.k', 'MarkerSize', 18, ...
        'DisplayName','generic Bogdanov-Takens point');
legend('Location','NorthWest')
xlabel('$\theta$','Interpreter','LaTeX');
ylabel('$\delta$','Interpreter','LaTeX');
axis([0.0978    0.1002    0.5086    0.6042])
```

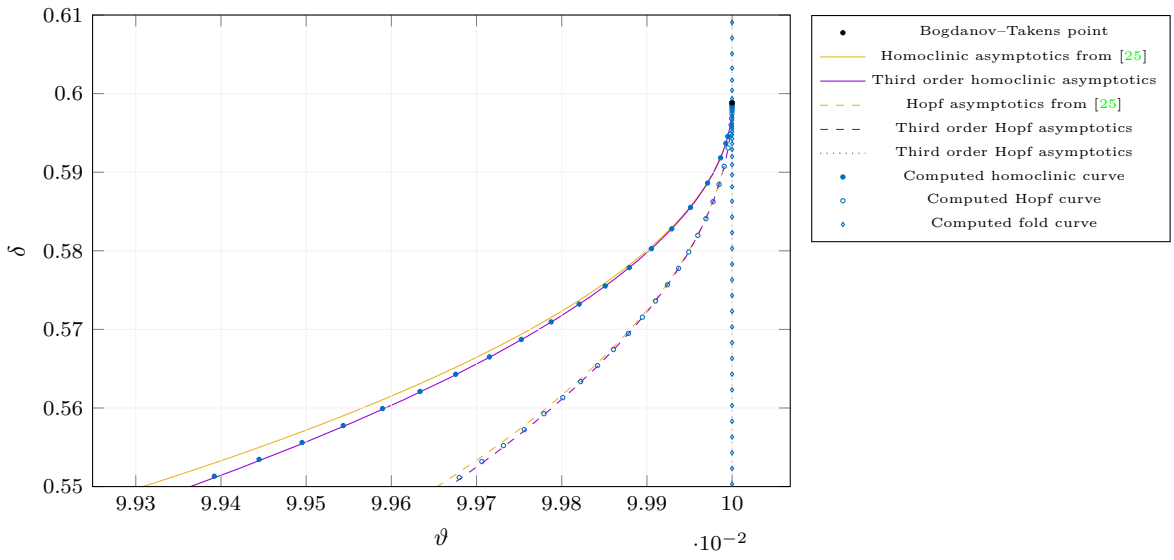

Figure S2: Bifurcation diagram near the analytically derived generic Bogdanov-Takens point in (S2) comparing computed codimension one curves using `DDE-BifTool` with the asymptotics obtained in this paper and in [25].

S1.11 Compare homoclinic solutions in phase-space

To get an impression of the third-order homoclinic asymptotics in phase-space, we compare the corrected and uncorrected homoclinic solutions with the perturbation parameter ranging from 0.1 to 0.3.

The code below results in Figure S3. We see that the corrected and predicted homoclinic orbits are nearly identical.

```
%% Compare homoclinic solutions in phase-space
step = linspace(0.1,0.3,10);
[~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'predictor',true,'debug',0);
[~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'debug',0);
figure(4); clf; hold on
title('Compare homoclinic orbits in phase-space')
for i=1:length(hcli_br_approx.point)
    profile = hcli_br_approx.point(i).profile;
    plot(profile(1,:),profile(2,:), 'Color', cm(2,:))
    profile_correc = hcli_br_correc.point(i).profile;
    plot(profile_correc(1,:),profile_correc(2,:), '.', 'Color', cm(1,:))
end
legend({'Predicted homoclinic order', 'Corrected homoclinic orbir'})
xlabel('$x$','Interpreter','LaTeX');
ylabel('$y$','Interpreter','LaTeX');
```

The MATLAB console shows the following output.

```
hcli from BT: branch 1 of  1 correction of point 1, success=1
hcli from BT: branch 1 of  1 correction of point 2, success=1
hcli from BT: branch 1 of  1 correction of point 3, success=1
hcli from BT: branch 1 of  1 correction of point 4, success=1
hcli from BT: branch 1 of  1 correction of point 5, success=1
hcli from BT: branch 1 of  1 correction of point 6, success=1
hcli from BT: branch 1 of  1 correction of point 7, success=1
hcli from BT: branch 1 of  1 correction of point 8, success=1
hcli from BT: branch 1 of  1 correction of point 9, success=1
hcli from BT: branch 1 of  1 correction of point 10, success=1
```

That is, all predictions in this range are successfully corrected.

S1.12 Convergence plot

In Figure S1 we compared the profiles of the first and third-order homoclinic asymptotics. Although the improvement of the third-order asymptotics is clearly visible, a better way to numerically compare the different orders is by creating a log-log convergence plot. Since we create convergence plots for all examples treated in this supplement, we created the function `convergence_plot`.

```
function relativeErrors = convergence_plot(funcs, bt, orders, amplitudes, varargin)

    default = {'dir',0,'free_pars',[],'orders',3,'TTolerance',1e-5,'ntst',40, ...
               'generic_unfolding',1,'debug',false};
    [options,pass_on] = dde_set_options(default,varargin,'pass_on');

```

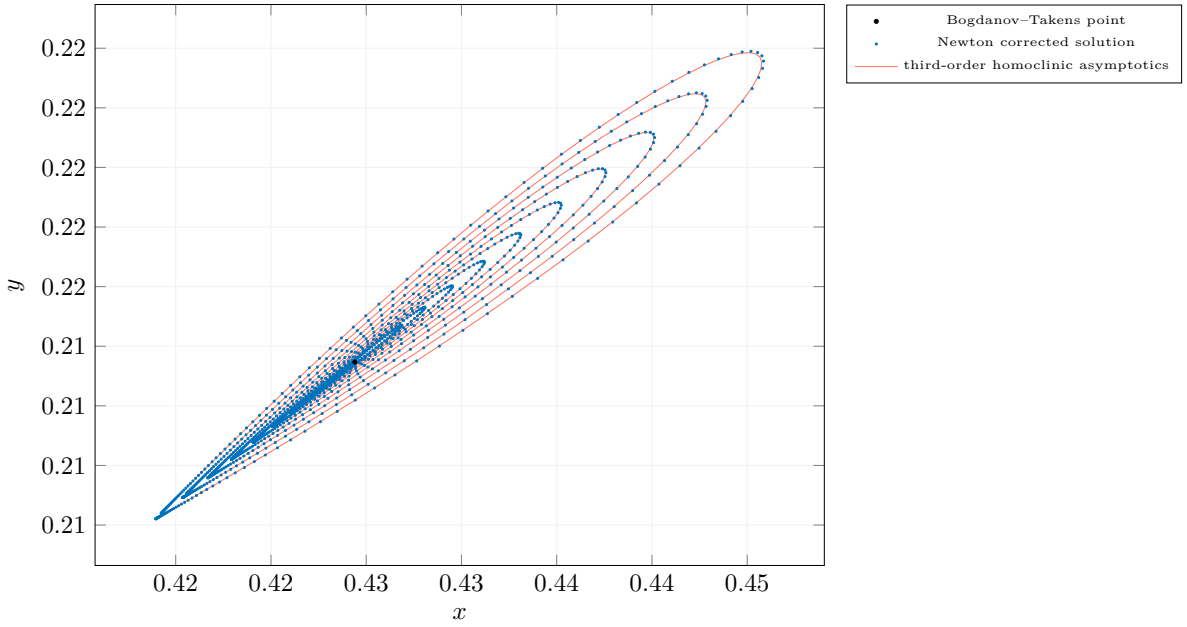

Figure S3: Plot comparing the third-order homoclinic asymptotics from Section 4.1.9 near the generic Bogdanov–Takens bifurcation in (S2) with the Newton correct homoclinic solutions in (x, y) phase-space.

```
    mhcli=df_mthod(funcs,'hcli');
    relativeErrors = {};
    for i=orders
        relativeErrors{i} = zeros(size(amplitudes));
        options.order = i;
        for j=1:length(amplitudes)
            if options.generic_unfolding
                options.amplitude = amplitudes(j);
            else
                options.amplitude = amplitudes(j)*[1,1];
                options.TTolerance = options.TTolerance.*[1,1];
            end
            hcli = init_BT_Hom_orbital(funcs,bt,options);
            if length(hcli) == 1
                hcli_pred = hcli;
            else
                hcli_pred = hcli(2);
            end
            hcli_pred = hcli_pred{1};
            if options.dir > 0
                [hcli_corrected,s]=p_correc(funcs,hcli_pred, ...
                    options.free_pars(options.dir),[],mhcli.point);
            else
                secant=p_tangent(funcs,mhcli.point,hcli_pred,options.free_pars);
                [hcli_corrected,s]=p_correc(funcs,hcli_pred, ...
```

```
                    options.free_pars,secant,mhcli.point);
            end
            if s
                numerator = norm(hcli_corrected.profile-hcli_pred.profile,Inf);
                denominator = norm(hcli_corrected.profile,Inf);
                relativeErrors{i}(j) = numerator/denominator;
            end
        end
    end

end
```

Listing S3: Auxiliary function for creating convergence plots.

Using this function, the code below yields Figure S4.

```
%% Convergence plot
amplitudes = logspace(-4, -1, 20);
orders = [1:3];
relativeerrors = convergence_plot(funcs, bt, orders, amplitudes, 'free_pars', ...
    free_pars, 'orders',orders,'ttolerance',1e-6,'ntst',82,'debug',false,'dir',1);
figure(5); clf; hold on
title('Convergence plot comparing first and third order homoclinic asymptotics')
plot(log10(amplitudes), log10(relativeerrors{1}(:)), '*-')
plot(log10(amplitudes), log10(relativeerrors{3}(:)), '*-')
legend({'first order', 'thrid order'})
xlabel('amplitude')
ylabel('relative error')
```

S1.13 Simulation with `DifferentialEquations.jl`

We finish this demonstration by simulating the dynamics near the generic Bogdanov–Takens point. In Figure S5 we created the full local unfolding of the singularity. Note that, compared with [25], the simulation is done with the original delay differential equations (S2) and not with the ordinary differential equations of the reduced system on the center manifold. We are able to do this since the parameter-dependent center manifold is locally attractive. In order to integrate the system in the reverse direction, i.e., to obtain the orbits in the stable manifold of the equilibria, we multiplied the right-hand side of the system in (S2) by -1. Note that, in general, this will not provide an accurate approximation at all. However, since the delay is relatively small, this approximation is accurate enough for our application. Also, note that the Bogdanov–Takens point still exists for the approximate system. Nonetheless, even without the backward solutions, the bifurcation diagram shows that the numerical analysis obtained in `DDE-BifTool` is correct.

Since the code for creating the local unfolding diagram is rather long, we show the code for reproducing the plot simulating the system near the homoclinic orbit. The code for creating the bifurcation diagram in Figure S5 can be found in the GitHub repository.

S1.13.1 Loading necessary Julia packages

We start by loading the necessary packages. These are

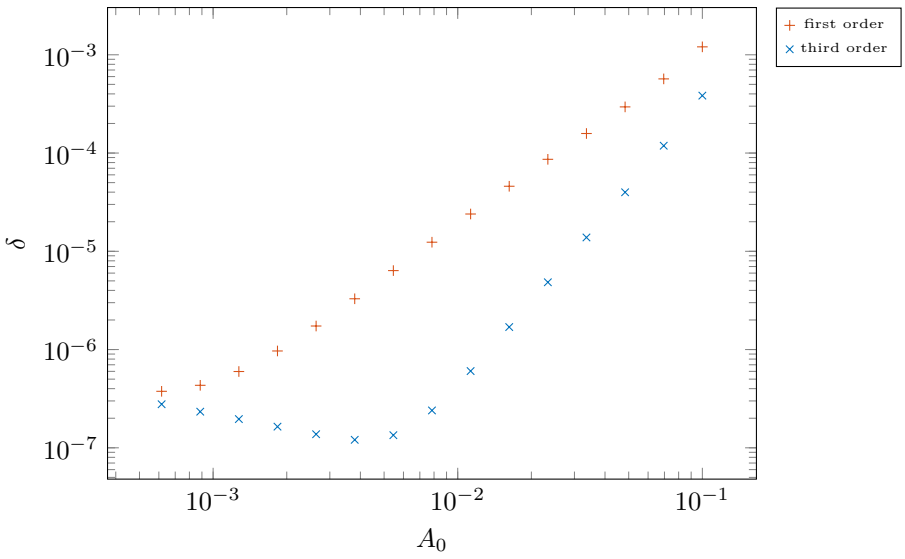

Figure S4: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution `hcli_pred` to the defining system for the homoclinic orbit and the Newton corrected solution `hcli_corrected`.

- `DifferentialEquations.jl` A suite for numerically solving differential equations written in Julia.
- `GLMakie.jl` For high-level plotting on the GPU.
- `NonlinearEigenproblems.jl` A nonlinear eigenvalue problem determine a scalar λ and a vector v such that $M(\lambda)v = 0$. In our case, the matrix $M(\lambda)$ will be the characteristic matrix.
- `DDEBifTool.jl` We created this very minimalistic package to have some functionality for normal form calculations of DDEs in Julia. Here we use it to calculate the derivatives of the system necessary for `NonlinearEigenproblems.jl`.
- `DelimitedFiles.jl` Reading and writing of CSV files.
- `PGFPlotsX.jl` A Julia package for creating publication quality figures using the LaTeX library PGFPlots as the backend.

```
using DifferentialEquations
using GLMakie
using DDEBifTool
using NonlinearEigenproblems
using DelimitedFiles
using PGFPlotsX
using DataFrames
using Colors
```

S1.13.2 Define system

Next we define the system to be integrated, a system to approximate the reverse flow, and also an allocating version used for stability calculations.

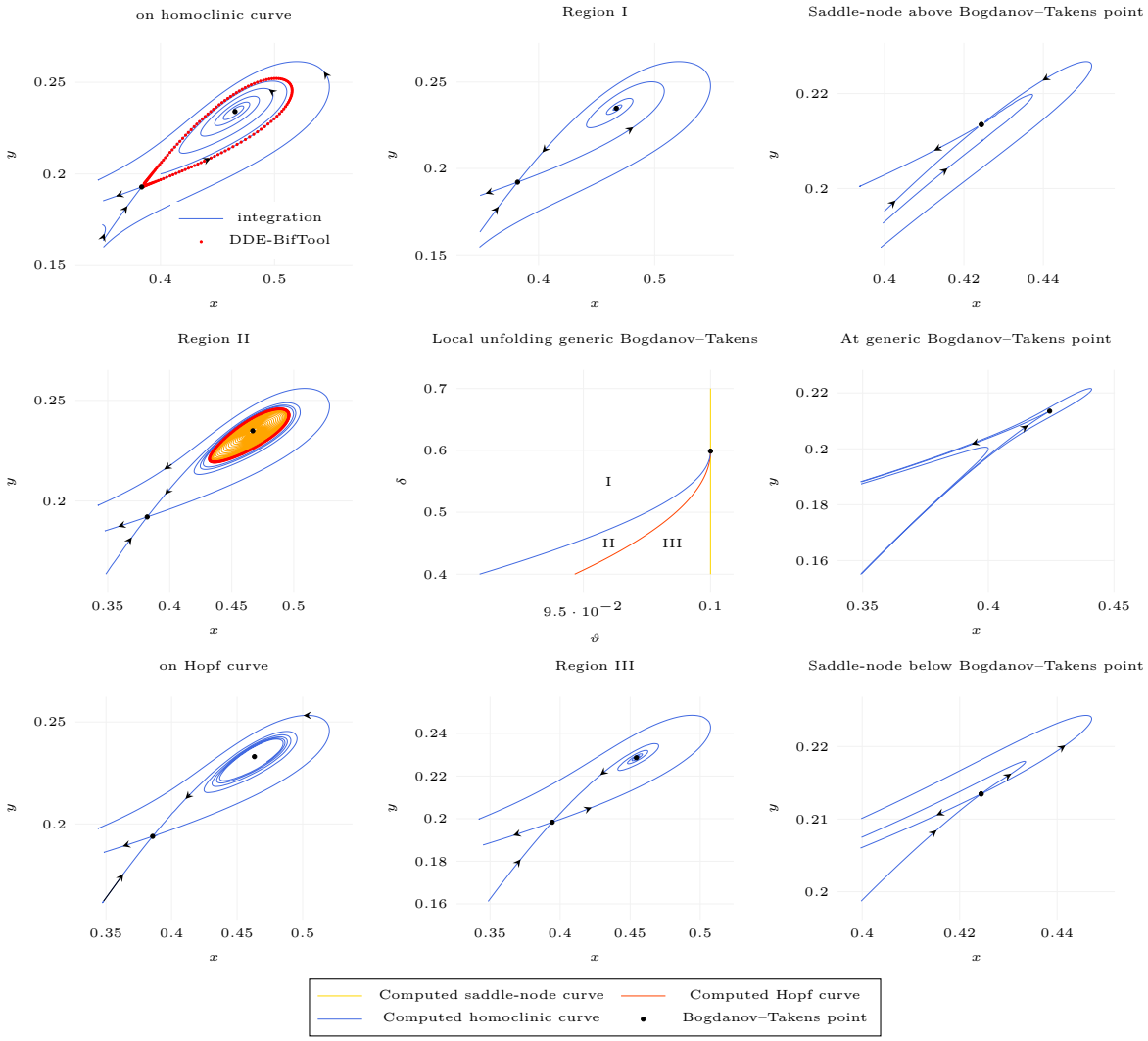

Figure S5: Bifurcation diagram near the derived generic Bogdanov-Takens point in (S3). In the center, we plotted the computed codimension one curves emanating from the Bogdanov–Takens point using `DDE-BifTool` with the third-order homoclinic asymptotics obtained in Section 4.1.9. The simulations surrounding the center plot have been performed in Julia.

```
# define constants
const m = 1.502983803
const α = 0.9
const γ = 0.15
const τ = 0.01

# define model
function predator_prey!(du,u,h,p,t)
    θ, δ = p
    x,y = u
```

```
    xt, yt = h(p, t-τ)
    du[1] = x*((1-x)*(x-γ)/(x+θ) - α*y/(x+y))
    du[2] = δ*y*(-1 + m*xt/(xt+yt))
end

# approximate the reversed flow by multiplying
# the right-hand side by -1
function predator_prey_rev!(du,u,h,p,t)
    predator_prey!(du,u,h,p,t)
    du .= -du
end

# allocating version (used for calculating stability)
function predator_prey(xx, pars)
    du = similar(xx[:,1])
    h(p,t) = xx[:,2]
    predator_prey!(du,xx[:,1],h,p,0)
    du
end
```

S1.13.3 Functions for plotting arrows

We define a function to show in which direction the orbits flow, which is useful when plotting in phase-space. We also define a function to show the direction of the leading eigenvectors of the characteristic matrix.

```
# function to show the direction of the orbits
function draw_arrow_on_solution(ax,sol, pointnumber, direction; arrowsize=30)
    x = [sol[1,pointnumber]]
    y = [sol[2,pointnumber]]
    u = [sol[1,direction(pointnumber,1)]-sol[1,pointnumber]]
    v = [sol[2,direction(pointnumber,1)]-sol[2,pointnumber]]
    arrows!(ax,x,y,u,v,arrowsize=arrowsize)
    Dict(:x=>x[1],:y=>y[1],:u=>u[1],:v=>v[1])
end

# function for plotting arrows into the eigen direction of the saddle-node
function arrow(ax, point, V, δ, direction)
    if direction > 0
        x, y, u, v = [point; -δ*V]
    else
        endpoint = point + δ*V
        lines!(ax,hcat([endpoint, point]...),color=:black)
        x, y, u, v = [endpoint; -0.1δ*V]
    end
    arrows!(ax,[x],[y],[u],[v],arrowsize=30)

    # return dictionary for PGFPlotsX
    Dict(:x=>x,:y=>y,:u=>u,:v=>v)
end
```

```
64
65 function drawEigenDirections(ax, point, Vstable, Vunstable;
66         distance=1e-03)
67     arrow(ax, point.coords, Vunstable,  distance,  1)
68     arrow(ax, point.coords, Vunstable, -distance,  1)
69     arrow(ax, point.coords, Vstable,    distance, -1)
70     arrow(ax, point.coords, Vstable,   -distance, -1)
71 end
```

Listing S4: Functions for plotting arrows.

S1.13.4 Define parameters, equilibria

We define parameters located on the continued homoclinic branch with `DDE-BifTool`. Then define the non-trivial equilibria points in (S2), which can be derived analytically.

```
73 ## Simulation at homoclinic curve
74 θ₀, δ₀ = [0.094448552842823,   0.447783343351055]
75 p = [θ₀; δ₀]
76
77 # define equilibria analytically
78 D(θ) = (α + m - α*m + γ*m)^2 - 4*m*(γ*m + α*(-1 + m)*θ)
79 x₀ = θ -> [(α + m - α*m + γ*m - √D(θ))/(2*m),
80            (α + m - α*m + γ*m + √D(θ))/(2*m)]
```

S1.13.5 Plot equilibria and homoclinic orbit

By plotting the homoclinic orbit obtained with `DDE-BifTool`, we can compare with the numerical simulations.

```
82 ## simlation from saddle-node
83 point = (coords = [x₀(θ₀)[1]; (m-1)*x₀(θ₀)[1]], pars = [θ₀, δ₀])
84
85 # plot equilibria
86 scatter(Point2f.(x₀(θ₀), (m-1)*x₀(θ₀)),color=:black)
87
88 # plot homoclinic orbit from DDE-BifTool
89 csvdir = "/home/maikel/Documents/MyPapers/BTPaper/data/doubleAlleeEffect/"
90 homoclinic_orbit = readdlm(csvdir * "homoclinicorbit.csv")
91 scatter!(homoclinic_orbit, color=:orangered, label="Solution from DDE-BifTool")
```

S1.13.6 Eigenvectors

Next, we calculate and plot the leading eigenvectors of the characteristic matrix at the saddle-node bifurcation point.

```
93 # calculate stability
94 τs = [0.0 τ]
```

```
M = coefficient_matrices(predator_prey, point, τs)
dep=DEP([M[j](point.coords,point.pars) for j in 1:length(τs)],vec(τs))
λ,V=iar_chebyshev(dep,maxit=100,neigs=2)
V = real(V)
indx_unstable = findall(λ -> real(λ) > 0, λ)
indx_stable = findall(λ -> real(λ) < 0, λ)
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])

# draw eigenvectors
drawEigenDirections(current_axis(), point, Vstable, Vunstable; distance=1e-03)
```

S1.13.7 Define callback

Since we are only interested in the flow near the equilibria points, we create a discrete callback to ensure the orbits do not become too large.

```
# create callback
condition(u,t,integrator) = u[1] < 0.35
affect!(integrator) = terminate!(integrator)
cb = DiscreteCallback(condition,affect!)
```

S1.13.8 Integrate the system

Now we define the problem to be integrated and choose the algorithm to be used. The first of the five numerical simulations below starts near the unstable eigendirection. We rotated the eigenvector slightly to follow the homoclinic orbit very close. The rotation value of α_0 was actually obtained by using the bisection method. However, we did not include this code here.

```
# define DDEProblem and choose integration algorithm
prob(h,p,tspan,direction) =
    DDEProblem(direction > 0 ? predator_prey! : predator_prey_rev!,
                h(p,0), h, tspan,p; constant_lags=[τ], callback=cb)
alg = MethodOfSteps(Tsit5())

# orbit startin in the unstable eigendirection crossing
# all points of the homoclinic orbit obtained in DDE-BifTool
ϵ = sign(Vunstable[1])*1e-03
R(α) = [cos(α) -sin(α);sin(α) cos(α)]
α₀ = 0.00032200554789652436
h(p,t) = point.coords + ϵ*R(α₀)*Vunstable
tspan =  (0.0, 310.0);
sol1 = solve(prob(h,p,tspan,1),alg,reltol=1e-12,abstol=1e-12)
lines!(sol1,color=:royalblue, label="Solution from integration")

# orbit converging to inner equilibria
h(p,t) = [0.4; 0.2]
sol2 = solve(prob(h,p,tspan,1),alg,reltol=1e-12,abstol=1e-12)
lines!(sol2,color=:royalblue1)
```

```
132
133 # orbit starting in the leading stable eigendirection
134 # and solved with the approximated system
135 ϵ = sign(Vstable[1])*1e-04
136 h(p,t) = point.coords - ϵ*Vstable
137 sol3 = solve(prob(h,p,tspan,-1),alg,reltol=1e-12,abstol=1e-12)
138 lines!(sol3,color=:royalblue, linestyle=:dash)
139
140 # the next two orbits show the flow around the stable eigendirection
141 h(p,t) = [0.35; 0.16]
142 sol4 = solve(prob(h,p,tspan,1),alg,reltol=1e-12,abstol=1e-12)
143 lines!(sol4,color=:royalblue)
144
145 h(p,t) = [0.35; 0.166]
146 sol5 = solve(prob(h,p,tspan,1),alg,reltol=1e-12,abstol=1e-12)
147 lines!(sol5,color=:royalblue)
```

S1.13.9 Finish the plot

Lastly, we add arrows to the obtained solutions using the function `draw_arrow_on_solution` defined above. Also, we add the legend and re-plot the equilibria, so that they appear on top.

```
149 # add arrows on solution
150 draw_arrow_on_solution(current_axis(), sol1, 120, +)
151 draw_arrow_on_solution(current_axis(), sol1, 600, +)
152 draw_arrow_on_solution(current_axis(), sol2, 370, +)
153 draw_arrow_on_solution(current_axis(), sol3, 150, -)
154 draw_arrow_on_solution(current_axis(), sol4, 150, +)
155 draw_arrow_on_solution(current_axis(), sol5, 24, +)
156
157 # add legend
158 axislegend(position = :lt)
159
160 # redraw equilbria
161 scatter!(Point2f.(x0(θ0), (m-1)*x0(θ0)),color=:black)
```

We should now obtain an interactive figure similar to the left figure in Figure S6.

S2 Generic Bogdanov–Takens bifurcation in a neural network model

In this example, we will consider the model

$$\begin{cases} \mu \dot{u}_1(t) = -u_1(t) + q_{11}\alpha(u_1(t\text{-}T)) - q_{12}u_2(t\text{-}T) + e_1, \\ \mu \dot{u}_2(t) = -u_2(t) + q_{21}\alpha(u_1(t\text{-}T)) - q_{22}u_2(t\text{-}T) + e_2, \end{cases} \tag{S3}$$

which describes the dynamics of a neural network consisting of excitatory and inhibitory neurons [19]. The variables and parameters occurring in (S3) have the following neurophysiological meaning:

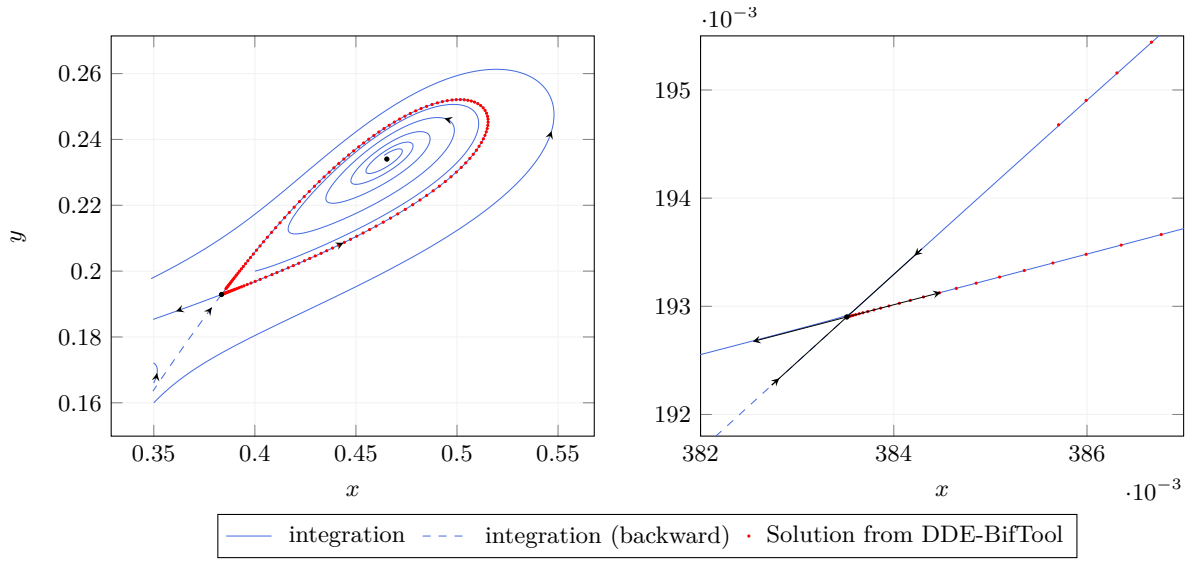

Figure S6: Integration of the delayed predator-prey model (S2) at parameter values $(\theta, \delta) = (0.094448552842823, 0.447783343351055)$ obtained from continuation of the homoclinic curve emanating from the Bogdanov–Takens point using `DDE-BifTool`. In the plot to the right a close-up near the saddle is given. Additionally, the leading eigenvectors of the characteristic matrix are shown.

- $u_1, u_2 : \mathbb{R} \to \mathbb{R}$ denote the total post-synaptic potential of the excitatory and inhibitory neurons, respectively.
- $\mu > 0$ is a time constant characterizing the dynamical properties of the cell membrane.
- $q_{ik} \geq 0$ represent the strength of the connection line from the kth neuron to the ith neuron.
- $\alpha : \mathbb{R} \to \mathbb{R}$ is the transfer function which describes the activity generation of the excitatory neuron as a function of its total potential u_1. The function α is smooth, increasing and has an unique turning point at $u_1 = \theta$. The transfer function corresponding to the inhibitory neuron is assumed to be the identity.
- $T \geq 0$ is a time delay reflecting synaptic delay, axonal and dendritic propagation time.
- e_1 and e_2 are external stimuli acting on the excitatory and inhibitory neuron, respectively.

Following [19], we consider equation (S3) with

$$\alpha(u_1) = \frac{1}{1+e^{-4u_1}} - \frac{1}{2}, \qquad q_{11} = 2.6, \qquad q_{21} = 1.0, \qquad q_{22} = 0.0,$$
$$\mu = 1.0, \qquad T = 1.0, \qquad e_2 = 0.0,$$

and $Q := q_{12}$, $E := e_1$ as bifurcation parameters. Substituting into (S3) yields

$$\begin{cases} \dot{u}_1(t) = -u_1(t) + 2.6\alpha(u_1(t-1)) - Qu_2(t-1) + E, \\ \dot{u}_2(t) = -u_2(t) + \alpha(u_1(t-1)). \end{cases} \tag{S4}$$

Notice that for any steady-state we have the symmetry

$$(u_1, u_2, E) \to (-u_1, -u_2, -E). \tag{S5}$$

It is easy to explicitly derive that the system has a double eigenvalue zero for

$$\begin{cases} u_1(t) = \dfrac{1}{4}\log\left(\dfrac{8-\sqrt{39}}{5}\right) \approx -0.2617, \\ u_2(t) = -\dfrac{1}{2}\sqrt{\dfrac{3}{13}} \approx -0.2402, \\ Q = \dfrac{13}{10}, \\ E = \dfrac{\sqrt{39} - 10\,\mathrm{atanh}\sqrt{\frac{3}{13}}}{20} \approx 0.0505. \end{cases} \tag{S6}$$

Remark 13. The `MATLAB` files for this demonstration can be found in the directory `demos/tutorial/VII/neural_network_model` relative to the main directory of the `DDE-BifTool` package. Here, we omit the code to generate the system file. We assume that the system file `sym_neural_network_mf.m` has been generated with the script `sym_neural_network.m`. Also, we assume that the `DDE-BifTool` package has been loaded as in Listing S1. The code in Sections S2.1 to S2.11 highlights the important parts of the file `neural_network_model.m`.

S2.1 Set parameter names and funcs structure

As in the previous example, we set the parameter names and define the `funcs` structure.

```
%% Set parameter names
% This could also be loaded from mat file |'FHN_parnames.mat'|.
parnames={'Q','E','tau'};
cind=[parnames;num2cell(1:length(parnames))];
ind=struct(cind{:});

%% Set the funcs structure
% We load the precalculated multilinear forms. These have been generated
% with the file gen_sym_neural_network.m.
funcs=set_symfuncs(@sym_neural_network_mf,'sys_tau',@()ind.tau);
% Alternatively, uncomment the line below to use directional derivaties.
% funcs=set_symfuncs(@sym_FHN,'sys_tau',@()ind.tau);
```

S2.2 Set parameter range

Since we are only interested here in the local unfolding, we restrict the allowed parameter range for the unfolding parameters. In practice, one may have physical restrictions which must be satisfied. Additionally, we also limit the maximum allowed step size during continuation. By doing so, we obtain more refined data to compare against our predictors.

```
%% Set bifurcation parameter range and step size bounds
brpars={'max_bound', [ind.Q 1.7],...
        'min_bound', [ind.Q 1.2; ind.E 0],...
        'max_step', [ind.Q 0.005; ind.E 0.005]};
```

S2.3 Stability and coefficients of the generic Bogdanov–Takens point

We manually construct a steady-state at the generic Bogdanov–Takens point.

```
%% Define analytically derived Bogdanov-Takens point
% construct steady-state point
Q0 = 13/10; E0 = (sqrt(39) - 10*atanh(sqrt(3/13)))/20; tau0 = 1;
stst=dde_stst_create('x',[1/4*log((8-sqrt(39))/5); -1/2*sqrt(3/13)]);
stst.parameter(ind.Q)  = Q0;
stst.parameter(ind.E) = E0;
stst.parameter(ind.tau)   = tau0;
% Calculate stability
method=df_mthod(funcs,'stst');
stst.stability=p_stabil(funcs,stst,method.stability);
stst.stability.l1
```

The MATLAB console shows the following output.

```
ans =

   1.0e-07 *

   0.548156278544666
  -0.548156314155979
```

The eigenvalues confirm that the point under consideration is indeed (an approximation to) a Bogdanov–Takens point. Furthermore, the remaining eigenvalues have negative real parts. Next, we calculate the normal form coefficients, the time-reparametrization, and the transformation to the center manifold with the function `nmfm_bt_orbital`, which implements the coefficients as derived in Section 4.1. For this, we need to set the argument `free_pars` to the unfolding parameter (Q, E). These coefficients will be used to start the continuation of the codimension one branches emanating from the Bogdanov–Takens point.

```
%% Calculate coefficients
free_pars=[ind.Q, ind.E];
bt = p_tobt(funcs,stst);
bt = nmfm_bt_orbital(funcs, bt,'free_pars', free_pars);
bt.nmfm
```

The MATLAB console shows the following output.

```
ans =

  struct with fields:

          a: -0.190382124055415
          b: -0.951910620277072
  theta1000: -0.546026508575597
```

```
theta0001: 1.473611111111115
      K10: [2x1 double]
      K01: [2x1 double]
      K02: [2x1 double]
      K11: [2x1 double]
      K03: [2x1 double]
     phi0: [1x1 struct]
     phi1: [1x1 struct]
    h0010: [1x1 struct]
    h0001: [1x1 struct]
    h2000: [1x1 struct]
    h1100: [1x1 struct]
    h0200: [1x1 struct]
    h1010: [1x1 struct]
    h1001: [1x1 struct]
    h0110: [1x1 struct]
    h0101: [1x1 struct]
    h0002: [1x1 struct]
    h0011: [1x1 struct]
    h3000: [1x1 struct]
    h2100: [1x1 struct]
    h1101: [1x1 struct]
    h2001: [1x1 struct]
    h0003: [1x1 struct]
    h1002: [1x1 struct]
    h0102: [1x1 struct]
        K: @(beta1,beta2)K10*beta1+K01*beta2+1/2*K02*beta2.^2
             +K11*beta1.*beta2+1/6*K03*beta2^3
        H: [function_handle]
```

Since the sign of ab is positive, we expect to find unstable periodic orbits nearby the Bogdanov–Takens point.

S2.4 Comparing profiles of computed and predicted homoclinic orbits

To test the homoclinic asymptotics from Section 4.1.9 we compare the first and third order asymptotics to the Newton corrected solution. For this, we use the function `C1branch_from_C2point`. This function returns a branch, which by default returns two initial corrected approximations in order to start continuation of the codimension one curve under consideration. By setting the argument `'predictor'` to `true` the approximations are left uncorrected. To make the comparison visually clear, we set the perturbation parameter $\epsilon = 0.25$ (`step = 0.25` in the code below). The code below produces Figure S7. The difference between the two approximations is clearly noticeable. While the first order asymptotics is close to the Newton corrected solution, the third order asymptotics is indistinguishable at this scale from the Newton corrected solution.

```
%% Plot comparing profiles computed and predicted homoclinic orbits
figure(1); clf; hold on;
cm=colormap('lines');
tiledlayout(2,2)
ylabels = {'$x$','$y$'};
```

```
step = 0.2;
for order=[1,3]

  [~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
      'codim2','BT','codim1','hcli','step',step,'predictor',true,'debug',0);

  [~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
      'codim2','BT','codim1','hcli','step',step,'debug',0);
    for j=1:2
        ax = nexttile; hold on; title(sprintf('order %d',order))

  plot(ax,hcli_br_approx.point(1).mesh,hcli_br_approx.point(1).profile(j,:),'*')
        plot(ax,hcli_br_correc.point(1).mesh,hcli_br_correc.point(1).profile(j,:))
        xlabel('$t$','Interpreter','LaTeX');
        ylabel(string(ylabels(j)),'Interpreter','LaTeX')
    end
end
```

S2.5 Continuation of the codimension one curves emanating

To continue the three codimension one curves emanating from the generic Bogdanov–Takens point, we can simply use the function `C1branch_from_C2point`, as shown in the code below. To monitor the continuation process, the argument `plot` must be set to `1`. The most important setting is the perturbation parameter (or multiple), `step` in the code below. If left out, default step sizes are defined. However, depending on the problem, no convergence may then be obtained.

```
%% Continue homoclinic solutions emanating from the BT point
[~,hcli_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli',brpars{:},'plot',0,'step',0.1);
assert(all(suc(:)>0))
nop=300; [hcli_br,suc]=br_contn(funcs,hcli_br,nop);assert(suc>0)

%% Continue Hopf curve
[~,hbr,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-05,'plot',0);
assert(all(suc(:)>0))
nop=300; [hbr,suc]=br_contn(funcs,hbr,nop);assert(suc>0)

%% Continue fold curve emanating from Bogdanov-Takens point
[~,fold_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold',brpars{:},'step',1e-03,'plot',0);
assert(all(suc(:)>0))
nop=300; [fold_br,suc]=br_contn(funcs,fold_br(1),nop);assert(suc>0)
fold_br = br_rvers(fold_br);
nop=300; [fold_br,suc]=br_contn(funcs,fold_br(1),nop);assert(suc>0)
```

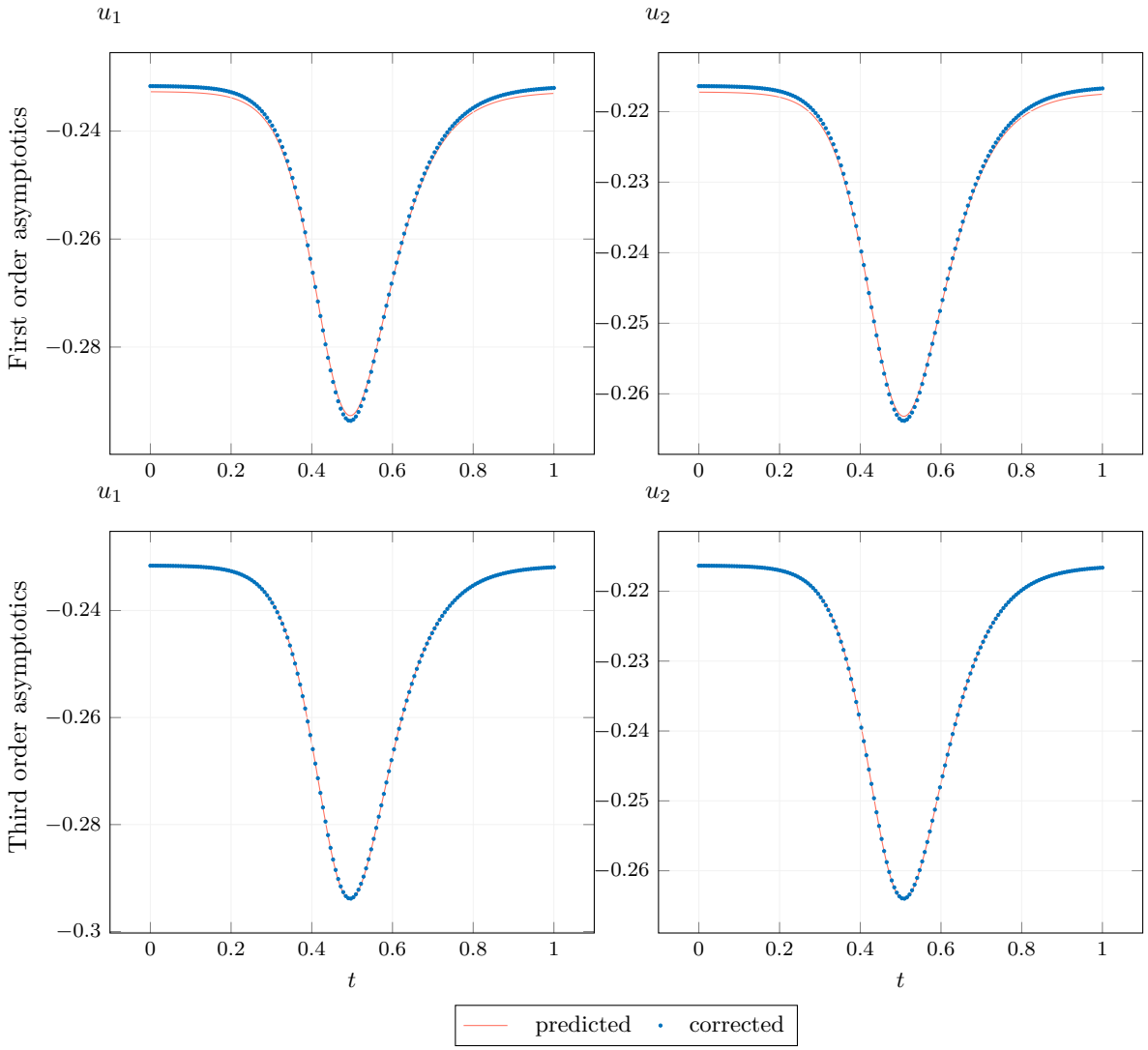

Figure S7: Comparison between the first and third-order asymptotics from Section 4.1.9 near the generic Bogdanov–Takens bifurcation in (S3) with the perturbation parameter set to $\epsilon = 0.25$.

S2.6 Predictors of the codimension one curves emanating from the Bogdanov–Takens point

Before we provide the bifurcation diagram in the next section, we first obtain the predictors for the codimension one curves. For this, we again use the function `C1branch_from_C2point`. We set the argument `predictor` to `1` and provide a range of perturbation parameters.

```
%% Predictors for homoclinic, Hopf, and transcritical curves
step = linspace(1e-4,0.6,50);
[~,hcli_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,'debug',0,...
    'codim2','BT','codim1','hcli','step',step,'predictor',true);
step = linspace(0.0001,0.2,50);
```

```
[~,hbr_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf','step',step,'predictor',true);
step = linspace(-0.1,0.3,40);
[~,fold_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold','step',step,'predictor',true);
hcli_br_pred_pm  = [getpars(hcli_br_pred, ind.Q), getpars(hcli_br_pred, ind.E)];
hbr_pm_pred = [getpars(hbr_pred,ind.Q); getpars(hbr_pred,ind.E)];
fold_br_pm_pred = [getpars(fold_br_pred,ind.Q); getpars(fold_br_pred,ind.E)];
```

In the last part of the code above, we added the asymptotics obtained from [25].

S2.7 Bifurcation diagram

The code below produces (a figure similar to) Figure S8.

```
%% Plot the bifurcation curves with predictors
figure(3);clf;hold on
title(['Codimension 1 curves emanating from the generic ', ...
       'Bogdanov-Takens point with predictors']);
plot(hcli_br_pm(1,:),hcli_br_pm(2,:),'.','Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br_pred_pm(:,1),hcli_br_pred_pm(:,2),'Color', cm(7,:), ...
    'DisplayName','Third order homoclinic asymptotics')
plot(hbr_pm(1,:),hbr_pm(2,:),'o','Color',cm(1,:), ...
    'DisplayName','Hopf branch')
plot(hbr_pm_pred(1,:),hbr_pm_pred(2,:),'--','Color',cm(7,:), ...
    'DisplayName','Hopf asymptotics')
plot(fold_br_pm(1,:),fold_br_pm(2,:),'d','Color',cm(1,:), ...
    'DisplayName','Fold branch')
plot(fold_br_pm_pred(1,:),fold_br_pm_pred(2,:),':','Color',cm(7,:), ...
    'DisplayName','Fold asymptotics')
plot(bt.parameter(ind.Q),bt.parameter(ind.E),'.k', 'MarkerSize', 18, ...
        'DisplayName','generic Bogdanov-Takens point');
xlabel('$Q$','Interpreter','LaTeX')
ylabel('$E$','Interpreter','LaTeX')
axis([1.2928    1.4323    0.0209    0.0522])
legend
```

S2.8 Compare homoclinic solutions in phase-space

To obtain an impression of the third-order homoclinic asymptotics in phase-space, we compare the corrected and uncorrected homoclinic solutions with the perturbation parameter ranging from 0.1 to 0.3. The code below results in Figure S9. We see that the corrected and predicted homoclinic orbits are nearly identical.

```
%% Compare predicted and corrected orbits in phase-space
step = linspace(0.1,0.3,10);
[~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
```

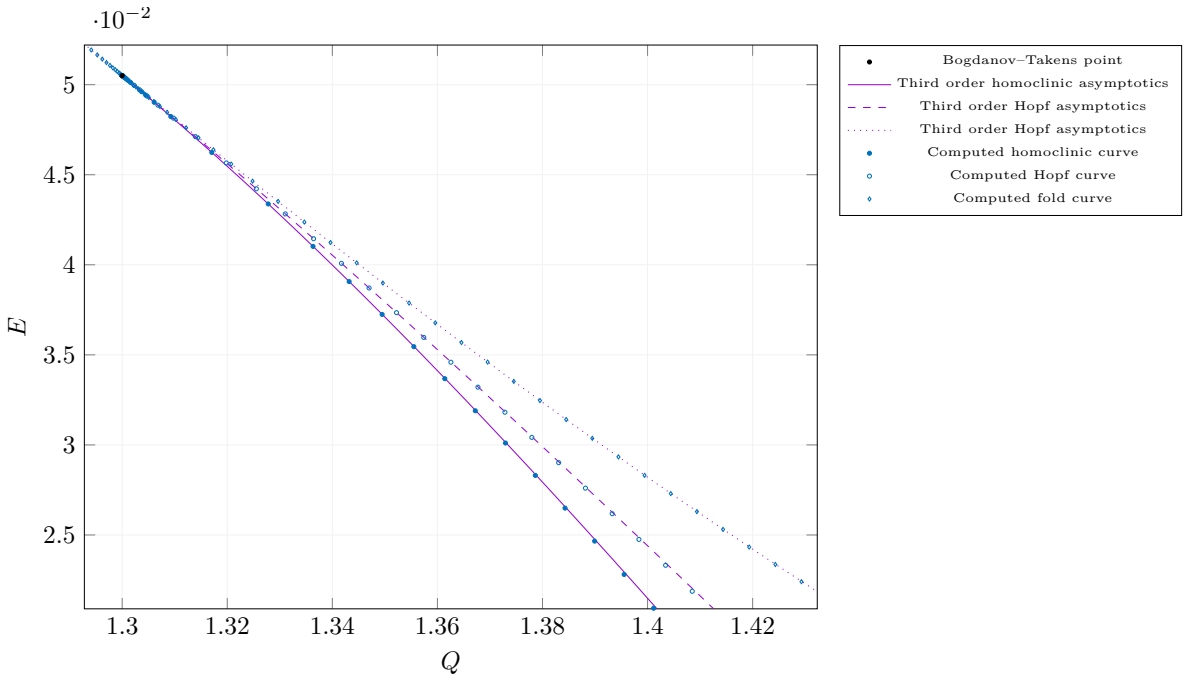

Figure S8: Bifurcation diagram near the derived generic Bogdanov-Takens point in (S3) comparing computed codimension one curves using `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.1.9.

```
    'codim2','BT','codim1','hcli','step',step,'predictor',true,'debug',0);
[~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,'order',order,...
    'codim2','BT','codim1','hcli','step',step,'debug',0);
figure(4); clf; hold on
title('Continued homoclinic orbits in phase-space')
for i=1:length(hcli_br_approx.point)
    profile = hcli_br_approx.point(i).profile;
    plot(profile(1,:),profile(2,:), 'Color', cm(2,:))
    profile = hcli_br_correc.point(i).profile;
    plot(profile(1,:),profile(2,:), '.', 'Color', cm(1,:))
end
legend({'Predicted homoclinic order', 'Corrected homoclinic orbir'})
xlabel('$u_1$','Interpreter','LaTeX');
ylabel('$u_2$','Interpreter','LaTeX');
```

S2.9 Convergence plot

Using the function from Section S1.12, we create a log-log convergence plot comparing the convergence order of the first and third order homoclinic asymptotics from Section 4.1.9. The code below yields Figure S10.

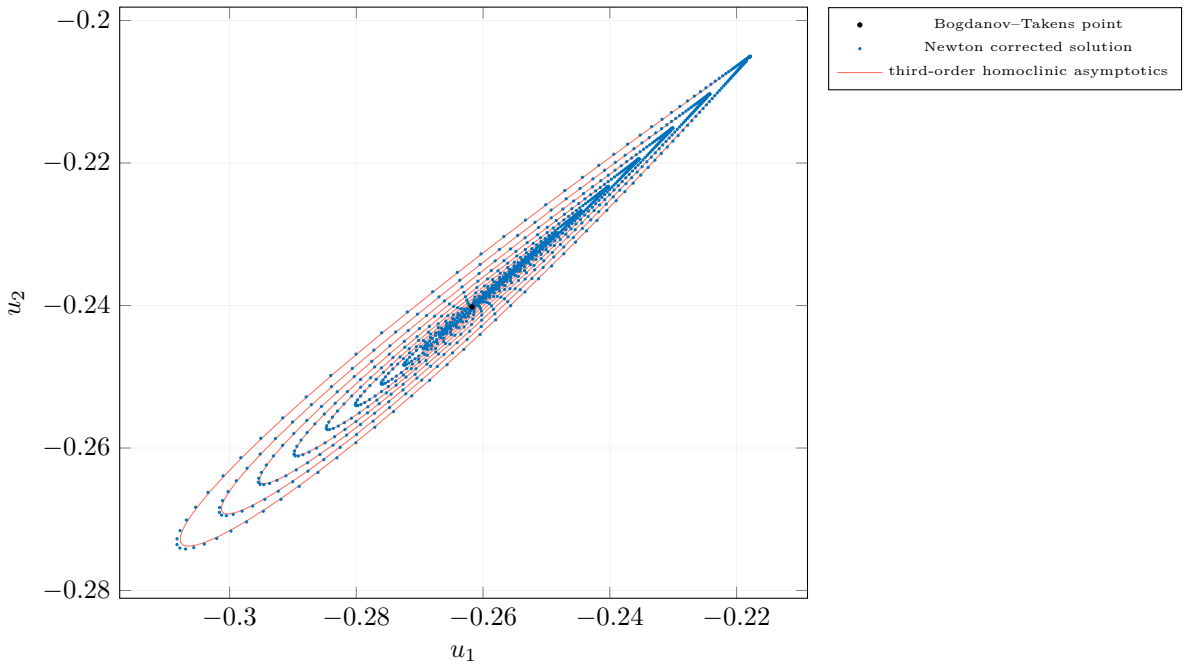

Figure S9: Plot comparing the third-order homoclinic asymptotics from Section 4.1.9 near the generic Bogdanov–Takens bifurcation in (S3) with the Newton correct homoclinic solutions in (u_1, u_2) phase-space.

```
%% Convergence plot
amplitudes = logspace(-2.6, -1, 20);
orders = [1:3];
relativeerrors = convergence_plot(funcs, bt, orders, amplitudes, 'free_pars', ...
    free_pars, 'orders',orders,'TTolerance',1e-4,'ntst',82,'debug',false,'dir',1);
figure(5); clf; hold on
title('Convergence plot comparing first and third order homoclinic asymptotics')
plot(log10(amplitudes), log10(relativeerrors{1}(:)), '*-')
plot(log10(amplitudes), log10(relativeerrors{3}(:)), '*-')
legend({'first order', 'thrid order'})
xlabel('amplitude')
ylabel('relative error')
```

S2.10 Continuation of the codimension one curves emanating from the second Bogdanov–Takens point

For completeness, we also continue the codimension one curves emanating from the second Bogdanov–Takens point, which exists due to the symmetry (S5). Of course, we could just use the symmetry instead of computing the curves numerically. However, we use it as an additional verification of our derived asymptotics. The code below defines the second Bogdanov–Takens point, calculates the stability, and continues the Hopf and homoclinic bifurcation curves.

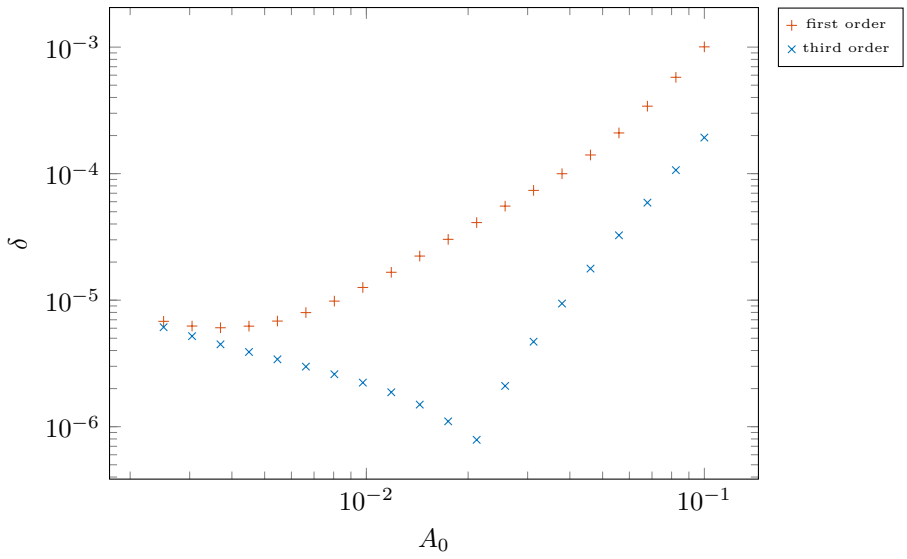

Figure S10: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution `hcli_pred` to the defining system for the homoclinic orbit and the Newton corrected solution `hcli_corrected`.

```
%% Define second analytically derived Bogdanov-Takens point
% construct steady-state point
stst=dde_stst_create('x',-[1/4*log((8-sqrt(39))/5); -1/2*sqrt(3/13)]);
stst.parameter(ind.Q)  = Q0;
stst.parameter(ind.E) = -E0;
stst.parameter(ind.tau) = tau0;
% Calculate stability
method=df_mthod(funcs,'stst');
stst.stability=p_stabil(funcs,stst,method.stability);
stst.stability.l1

%% Calculate coefficients
free_pars=[ind.Q, ind.E];
bt2 = p_tobt(funcs,stst);
bt2 = nmfm_bt_orbital(funcs, bt2,'free_pars', free_pars);
bt2.nmfm

%% Continue homoclinic solutions emanating from the BT point
[~,hcli_br2,suc]=C1branch_from_C2point(funcs,bt2,free_pars,...
    'codim2','BT','codim1','hcli',brpars{:},'plot',0,'step',0.1);
assert(all(suc(:)>0))
nop=300; hcli_br2=br_contn(funcs,hcli_br2,nop);assert(suc>0)

%% Continue Hopf curve
[~,hbr2,suc]=C1branch_from_C2point(funcs,bt2,free_pars,...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-05,'plot',0);
assert(all(suc(:)>0))
```

```
nop=300; [hbr2,suc]=br_contn(funcs,hbr2,nop);assert(suc>0)
```

S2.11 Bifurcation diagram with two Bogdanov–Takens points

Now that we continued the Hopf and homoclinic bifurcation curves emanating from the second Bogdanov–Takens point, we can reconstruct the bifurcation diagram given in [19, Figure 7]. The code below results into a similar figure as Figure S11 in MATLAB.

```
%% Plot the bifurcation curves
figure(6);clf;hold on
title(['Codimension 1 curves emanating from the ', ...
            'transcritical Bogdanov-Takens point']);
getpars=@(br,ind) arrayfun(@(p)p.parameter(ind),br.point);
hcli_br_pm = [getpars(hcli_br,ind.Q), getpars(hcli_br,ind.E)]';
hcli_br2_pm = [getpars(hcli_br2,ind.Q), getpars(hcli_br2,ind.E)]';
hbr_pm = [getpars(hbr,ind.Q); getpars(hbr,ind.E)];
hbr2_pm = [getpars(hbr2,ind.Q); getpars(hbr2,ind.E)];
fold_br_pm = [getpars(fold_br,ind.Q); getpars(fold_br,ind.E)];
plot(hcli_br_pm(1,:),hcli_br_pm(2,:),'Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br2_pm(1,:),hcli_br2_pm(2,:),'Color',cm(1,:), ...
    'HandleVisibility','Off')
plot(hbr_pm(1,:),hbr_pm(2,:),'Color',cm(2,:), ...
    'DisplayName','Hopf branches')
plot(hbr2_pm(1,:),hbr2_pm(2,:),'Color',cm(2,:), ...
    'HandleVisibility','Off')
plot(fold_br_pm(1,:),fold_br_pm(2,:),'Color',cm(3,:), ...
    'DisplayName','Transcritical branch')
plot(bt.parameter(ind.Q),bt.parameter(ind.E),'.k', ...
        'MarkerSize', 18, ...
        'DisplayName','generic Bogdanov-Takens point');
plot(bt2.parameter(ind.Q),bt2.parameter(ind.E),'.k', ...
        'MarkerSize', 18, 'HandleVisibility', 'Off');
xlabel('$Q$','Interpreter','LaTeX')
ylabel('$E$','Interpreter','LaTeX')
legend
```

S2.12 Simulation with `DifferentialEquations.jl`

Here we will perform two simulations. The first simulation will be at the double homoclinic orbits, which will confirm the continuation of both homoclinic orbits and is also ascetically pleasing, see Figure S12. The second simulation will be in the region where there should be unstable periodic orbits, see Figure S13.

S2.12.1 Loading necessary Julia packages

We start by loading the necessary packages.

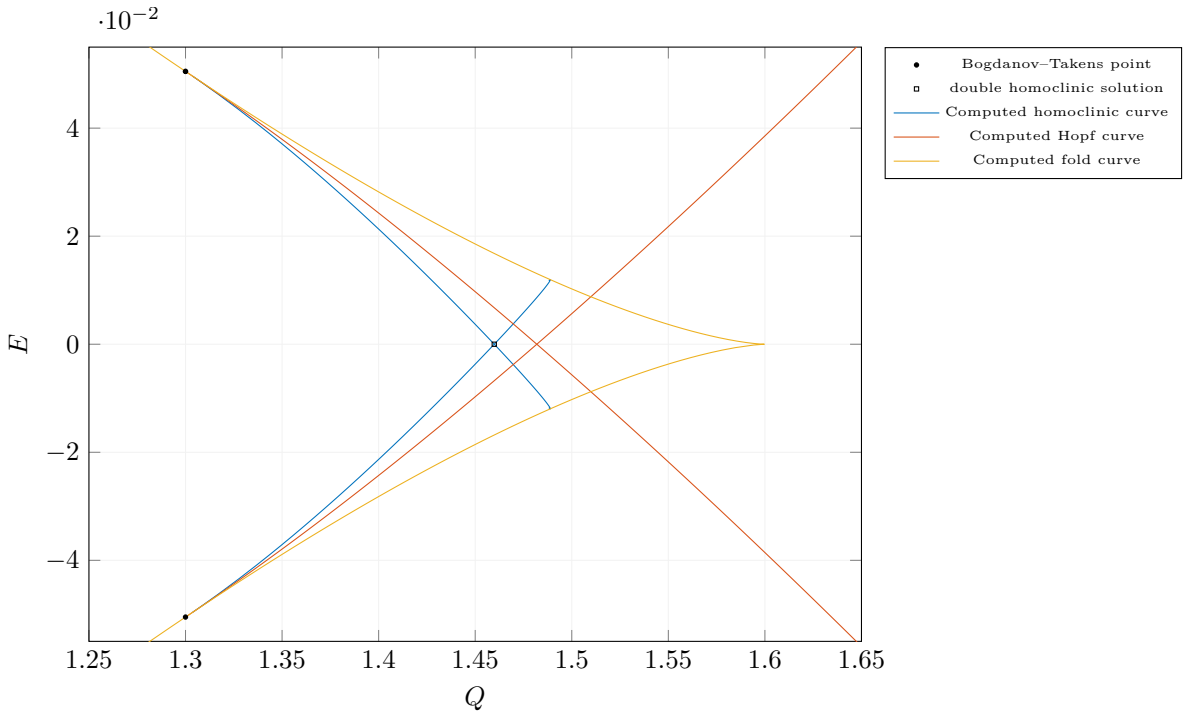

Figure S11: Reconstruction of the bifurcation diagram given in [19, Figure 7]. We could have used the symmetry (S5) instead of computing the additional curves numerically. However, it provides us an additional verification of our derived asymptotics.

```
using DifferentialEquations
using GLMakie
using DelimitedFiles
using IntervalArithmetic, IntervalRootFinding
using DDEBifTool
using NonlinearEigenproblems
```

Listing S5: Loading Julia packages for simulation in (S3).

In the previous demonstration we were able to derive the equilibria analytically. Here we will solve for the equilibria numerically with the packages `IntervalArithmetic.jl` [33] and `IntervalRootFinding.jl` [1].

S2.12.2 Define system

We define the system to be integrated, a system to approximate the reverse flow, and also an allocating version used for stability calculations.

```
# define constants
const τ = 1.0
```

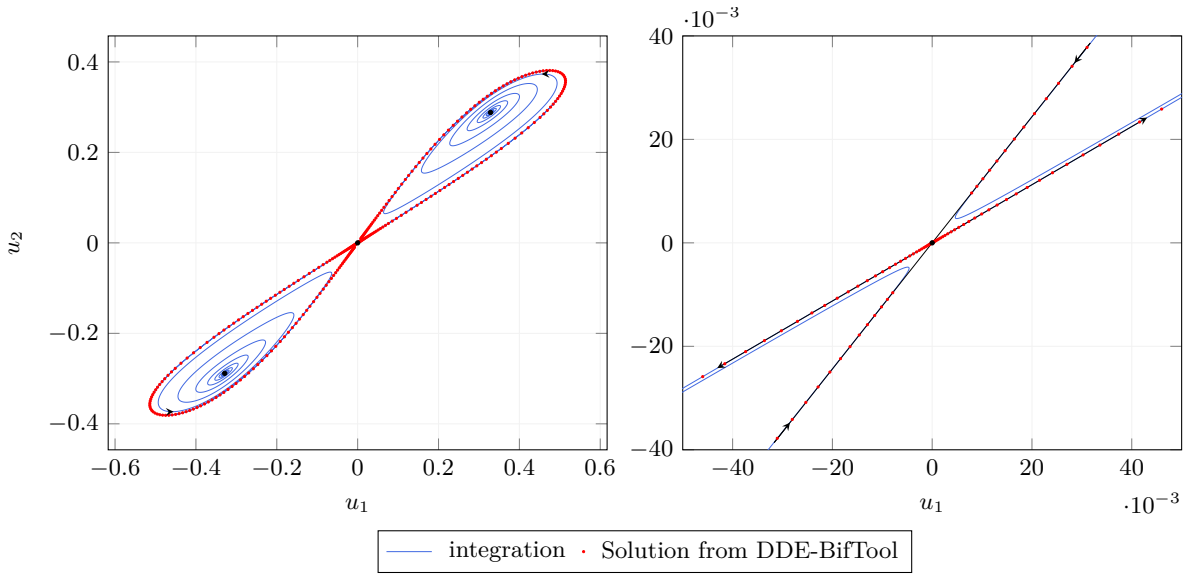

Figure S12: Comparing the computed double homoclinic orbit in (S3) with `DDE-BifTool` with the solutions obtained from numerical simulation with Julia. In the right plot is a close-up near the equilibrium at the origin. Also, the leading stable and unstable eigenvectors of the characteristic matrix are plotted. We see the numerical integrated solution intersects all the red points from the solution from `DDE-BifTool`.

```
10
11 # define model
12 @inline α(u₁) = 1.0/(1.0+exp(-4.0*u₁)) - 0.5
13 function neural_network_model!(du,u,h,p,t)
14     Q, E = p
15     u₁,u₂ = u
16     u₁t, u₂t = h(p, t-τ)
17     du[1] = -u₁ + 2.6*α(u₁t) - Q*u₂t + E;
18     du[2] = -u₂ + α(u₁t);
19 end
20
21 # allocating version (used for calculating stability)
22 function neural_network_model(u, p)
23     du = similar(u[:,1])
24     h(p,t) = u[:,2]
25     neural_network_model!(du,u[:,1],h,p,0)
26     du
27 end
```

S2.12.3 Functions for plotting arrows

We define a function to show in which direction the orbits flow, which is useful when plotting in phase-space. We also define functions to show the direction of the leading eigenvectors of the characteristic matrix. The code is shown in Listing S4.

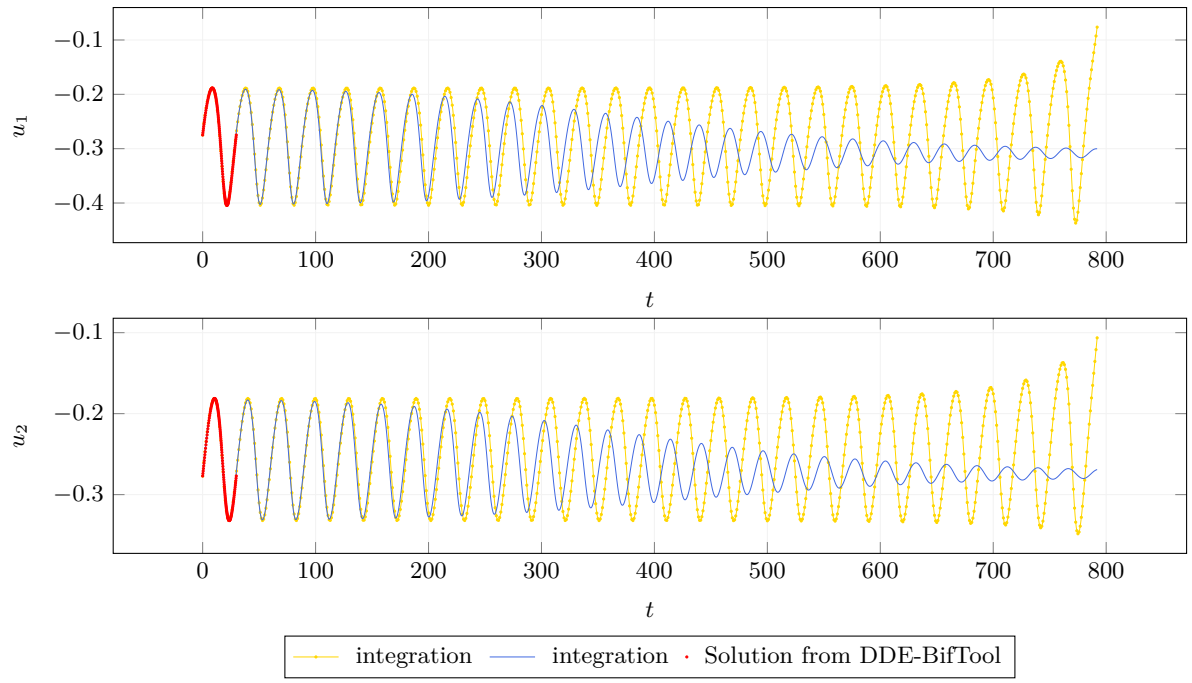

Figure S13: Comparing a computed periodic orbit in (S3) with `DDE-BifTool` at $(Q, E) = (1.476442865781454, 0.0)$ with the solution obtained from numerical simulation with Julia near the periodic orbit. The yellow dotted line has the constant history function $(u_1, u_2) = (-0.274863341578762, -0.27715979849863204)$ slightly below the periodic orbit. The blue line has the constant history function $(u_1, u_2) = (-0.274863341578762, -0.276969798498632)$, a point on the periodic orbit (red dots) located with `DDE-BifTool`.

S2.12.4 Create figure with several axes

We create a figure containing multiple axis in which we will plot the bifurcation diagram and the homoclinic and periodic orbits.

```
fig = Figure()
ax1 = Axis(fig[1:4,1], xlabel=L"u_1", ylabel=L"u_2",
           title="Double Homoclinic orbit")
ax2 = Axis(fig[1:4,2], xlabel=L"Q",   ylabel=L"E",
           title="Bifucation diagram")
ax3 = Axis(fig[1,3], xlabel=L"t", ylabel = L"u_1",
           title="Near periodic orbit")
ax4 = Axis(fig[2,3], xlabel=L"t", ylabel = L"u_2")
ax5 = Axis(fig[3,3], xlabel=L"t", ylabel=L"u_1",
           title="Near periodic orbit (zoom in)")
ax6 = Axis(fig[4,3], xlabel=L"t", ylabel = L"u_2")
```

S2.12.5 Plot bifurcation diagram in the middle

Loading the continued bifurcation curves obtained with `DDE-BifTool` and plot these in the middle axis. This should give a similar bifurcation diagram as in Figure S11.

```
csvdir = "../data/neuralNetworkModel/"
fold_br  = readdlm(csvdir * "foldbifurcationcurve.csv")
hopf_br  = readdlm(csvdir * "hopfbifurcationcurve.csv")
hom_br   = readdlm(csvdir * "homoclinicbifurcationcurve.csv")
hopf_br2 = readdlm(csvdir * "hopfbifurcationcurve2.csv")
hom_br2  = readdlm(csvdir * "homoclinicbifurcationcurve2.csv")
lines!(ax2, fold_br,  label = "computed fold curve", color=:gold)
lines!(ax2, hopf_br,  label = "computed Hopf curve", color=:orangered)
lines!(ax2, hom_br,   label = "computed homoclinic curve", color=:royalblue)
lines!(ax2, hopf_br2, color=:orangered)
lines!(ax2, hom_br2,  color=:royalblue)

# define Bogdanov-Takens point
u₁ = 1/4*log((8-sqrt(39))/5)
u₂ = -1/2*sqrt(3/13)
Q₀ = 13/10
E₀ = (√39 - 10*atanh(√(3/13)))/20;
bt = (coords = [u₁, u₂], pars = [Q₀, E₀])
scatter!(ax2, Point2f(Q₀, E₀), label = "generic Bogdanov-Takens point", color=:black)
scatter!(ax2, Point2f(Q₀, -E₀), color=:black)

# add double homoclinic bifurcation point
scatter!(ax2, Point2f(1.459868437376255,0), label ="Double homoclinic orbit",
↪  color=:blue)

# add region labels
text!(ax2, "II",  position = (1.4609, 3.304e-03), align = (:center,:center),
↪  textsize = 20)
axislegend(ax2, position = :ct)
```

S2.12.6 Simulation at the double homoclinic orbit

In the code below, we integrate at parameter values $(Q, E) = (1.459868437376222, 0)$, i.e., where we located the double homoclinic orbit. We start by locating the three equilibria points. Note that by the symmetry, we actually only need to solve for one of them. We calculate the stability of the equilibria and filter out the leading stable and unstable eigenvectors from the characteristic matrix of the saddle-node point. The rest of the code should be pretty straight forward, since it is very similar as in the simulation in the previous demonstration. After running this code, we should obtain a similar plot as in the left plot of Figure S12.

```
Q₀, E₀ = 1.459868437376222,0
p = [Q₀; E₀]

# derive equilbria
y(u₁) = -u₁ + (2.6 - Q₀)*α(u₁) + E₀
rts = roots(y,interval(-10,10), Newton, 1e-16)
u₁ = sort(mid.(interval.(rts)))
u₂ = α.(u₁)

```

```
# calculate stability
τs = [0.0 τ]
point = [(coords = [u₁[i]; u₂[i]], pars = p) for i ∈ 1:length(u₁)]
M = [coefficient_matrices(neural_network_model, point[i], τs) for i ∈ 1:length(u₁)]
dep = [DEP([M[i][j](point[i].coords,point[i].pars) for j in 1:length(τs)],vec(τs))
   ↪  for i ∈ 1:length(u₁)]
λ = iar_chebyshev.(dep,maxit=100,neigs=2)
saddle_indx = findall(λ -> abs(λ[1]) < 1e-14, imag([λ[i][1] for i ∈ 1:length(u₁)]))
saddle = point[saddle_indx][]
V = real.(λ[saddle_indx][][2])
indx_unstable = findall(λ -> real(λ) > 0, λ[saddle_indx][][1])
indx_stable = findall(λ -> real(λ) < 0, λ[saddle_indx][][1])
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])

# plot equilibria
scatter!(ax1,Point2f.([point[i].coords for i ∈ 1:length(u₁)]),color=:black)

# plot homoclinic orbit from DDE-BifTool
homoclinic_orbit = readdlm(csvdir * "doudble_homoclinicorbit.csv")
scatter!(ax1, homoclinic_orbit, color=:orangered)

# create callback
condition(u,t,integrator) = !(abs(u[1]) < 0.55)
affect!(integrator) = terminate!(integrator)
cb = DiscreteCallback(condition,affect!)

# define DDEProblem and choose integration algorithm
prob(h,p,tspan) = DDEProblem(neural_network_model!, h(p,0),h,tspan,p;
                             constant_lags=[τ], callback=cb)
alg = MethodOfSteps(Tsit5())

# integration from unstable direction
ϵ = sign(Vunstable[1])*1e-04
R(α) = [cos(α) -sin(α);sin(α) cos(α)]
α₀ = 0.006
h(p,t) = saddle.coords + ϵ*R(α₀)*Vunstable
tspan =  (0.0, 350.0);
hom_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax1,hom_sol1,color=:royalblue1)

h(p,t) = saddle.coords - ϵ*R(α₀)*Vunstable
hom_sol2 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax1,hom_sol2,color=:royalblue1)

# add arrows
draw_arrow_on_solution(ax1, hom_sol1, 700, +)
draw_arrow_on_solution(ax1, hom_sol2, 700, +)

# redraw equilbria
scatter!(ax1,Point2f.([point[i].coords for i ∈ 1:length(u₁)]),color=:black)
```

```
# draw leading eigenvectors from the characteristic matrix
drawEigenDirections(ax1, point[2], Vstable, Vunstable; distance=5e-02)
```

S2.12.7 Simulation near an unstable periodic orbit

To show by integration the existence of an unstable periodic orbit, we first located a periodic orbit in `DDE-BifTool`. This can be done by continuing a branch of periodic orbits emanating from a point on the continued Hopf curve. Then we load the profiles of the periodic orbits into Julia and start integration near the periodic orbits. After running the code below, we should obtain a similar plot as in Figure S13.

```
Q₀, E₀ = [1.476442865781454,   0.0]
scatter!(ax2, Point2f(Q₀,E₀),color=:black)
p = [Q₀; E₀]

# plot periodic orbit computed with DDE-BifTool
periodic_orbit = readdlm(csvdir * "periodicorbit_20.csv")
scatter!(ax3, periodic_orbit[:,1:2], color=:orangered)
scatter!(ax4, periodic_orbit[:,[1,3]], color=:orangered)

# plot again for close up
scatter!(ax5, periodic_orbit[:,1:2], color=:orangered)
scatter!(ax6, periodic_orbit[:,[1,3]], color=:orangered)

# create callback
condition(u,t,integrator) = !(abs(u[1]) < 0.7)
affect!(integrator) = terminate!(integrator)
cb = DiscreteCallback(condition,affect!)

# integration from first point on the periodic solution
h(p,t) = periodic_orbit[1,2:3]
h(0,0)
tspan =  (0.0, 792.0);
periodic_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-08,abstol=1e-08)
lines!(ax3,periodic_sol1.t,periodic_sol1[1,:])
lines!(ax4,periodic_sol1.t,periodic_sol1[2,:])
lines!(ax5,periodic_sol1.t,periodic_sol1[1,:])
lines!(ax6,periodic_sol1.t,periodic_sol1[2,:])
xlims!(ax5, [0, 29.789440842376618]) # as vector
ylims!(ax5, [-0.42, -0.15]) # as vector
xlims!(ax6, [0, 29.789440842376618]) # as vector
ylims!(ax6, [-0.42, -0.15]) # as vector

# integration from a little below first point on the first periodic solution
h(p,t) = periodic_orbit[1,2:3] - [0; 1.90e-04]
tspan =  (0.0, 792.0);
periodic_sol2 =
  ↪  solve(prob(h,p,tspan),MethodOfSteps(Tsit5()),reltol=1e-08,abstol=1e-08)
lines!(ax3,periodic_sol2.t,periodic_sol2[1,:])
```

```
lines!(ax4,periodic_sol2.t,periodic_sol2[2,:])
lines!(ax5,periodic_sol2.t,periodic_sol2[1,:])
lines!(ax6,periodic_sol2.t,periodic_sol2[2,:])
```

S3 Transcritical Bogdanov–Takens bifurcation in the Van der Pol oscillator with delay feedback

We consider the Van der Pol oscillator with delay feedback [24] given by

$$\ddot{x}(t) + \epsilon(x^2(t) - 1)\dot{x}(t) + x(t) = \epsilon g(x(t - \tau)) \tag{S7}$$

where $\epsilon > 0$ is a parameter, $\tau > 0$ is a delay and $g : \mathbb{R} \to \mathbb{R}$ is a smooth function with $g(0) = 0$ and $g'(0) \neq 0$. We rewrite the Van der Pol equation (S7) as

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \epsilon g(x_1(t - \tau)) - \epsilon(x_1^2 - 1)x_2 - x_1. \end{cases} \tag{S8}$$

Rescaling time with $t \to \dfrac{t}{\tau}$ to normalize the delay yields

$$\begin{cases} \dot{x}_1 = \tau x_2, \\ \dot{x}_2 = \tau \left(\epsilon g(x_1(t - 1)) - \epsilon(x_1^2 - 1)x_2 - x_1 \right). \end{cases} \tag{S9}$$

This allows to treat τ as a bifurcation parameter.

Following [24], we consider (S7) with

$$g(x) = \frac{e^x - 1}{c_1 e^x + c_2},$$

where $c_1 = \dfrac{1}{4}$ and $c_2 = \dfrac{1}{2}$. Then the trivial equilibrium undergoes a transcritical Bogdanov–Takens bifurcation at parameter values $(\epsilon, \tau) = (0.75, 0.75)$, [24] and the supplement.

Remark 14. The `MATLAB` files for this demonstration can be found in the directory `demos/tutorial/VII/vdpo_bt_transcritical` relative to the main directory of the `DDE-BifTool` package. Here, we omit the code to generate a system file. The system file `sym_vdpo_mf.m` has been generated with the script `sym_vdpo_mf.m`. Also, we assume that the `DDE-BifTool` package has been loaded as in Listing S1. The code in Sections S3.1 to S3.9 highlights the important parts of the file `vanderPolOscillator.m`.

S3.1 Set parameter names and funcs structure

As in the previous example, we set the parameter names and define the `funcs` structure.

```
%% Set parameter names
parnames={'epsilon','tau','vartau'};
cind=[parnames;num2cell(1:length(parnames))];
ind=struct(cind{:});

%% Set the funcs structure
funcs=set_symfuncs(@sym_vdpo_mf,'sys_tau',@()ind.vartau);
```

S3.2 Set parameter range

Since we are only interested here in the local unfolding, we restrict the allowed parameter range for the unfolding parameters. In practice, one may have physical restrictions which must be satisfied. Additionally, we also limit the maximum allowed step size during continuation. By doing so, we obtain more refined data to compare against our predictors.

```
%% Set bifurcatiun parameter range and step size bounds
brpars={'min_bound', [ind.epsilon 0.7122; ind.tau 0.7154],...
        'max_bound', [ind.epsilon 0.7998; ind.tau 0.7905],...
        'max_step',  [ind.epsilon 0.0001; ind.tau 0.0001]};
```

S3.3 Stability and coefficients of the transcritical Bogdanov–Takens point

We manually construct a steady-state at the transcritical Bogdanov–Takens point and calculate its stability.

```
%% Define analytically derived Bogdanov-Takens point
% manually construct steady-state point
epsilon = 0.75; tau = 0.75;
stst=dde_stst_create('x',[0;0]);
stst.parameter(ind.epsilon)  = epsilon;
stst.parameter(ind.tau) = tau;
stst.parameter(ind.vartau) = 1;

%% Compute stability
method=df_mthod(funcs,'stst');
stst.stability=p_stabil(funcs,stst,method.stability);
stst.stability.l1
```

The `MATLAB` console shows the following output.

```
ans =

   1.0e-07 *

       0.6223
      -0.6223
```

The eigenvalues confirm that the point under consideration is indeed (an approximation to) a Bogdanov–Takens point. Furthermore, the remaining eigenvalues have negative real parts. Next, we calculate the normal form coefficients, the time-reparametrization, and the transformation to the center manifold with the function `nmfm_bt_orbital`, which implements the coefficients as derived in Section 4.2. For this, we need to set the argument `free_pars` to the unfolding parameter (Q, E). These coefficients will be used to start the continuation of the codimension one branches emanating from the Bogdanov–Takens point. Also, since we are in the transcritical case, we set the argument `generic_unfolding` to `false`.

```
%% Calculate parameter-dependent normal form coefficients
free_pars=[ind.epsilon, ind.tau];
bt = p_tobt(funcs,stst);
bt = nmfm_bt_orbital(funcs, bt,'free_pars', free_pars, 'generic_unfolding', false);
```

The MATLAB console shows the following output.

```
ans =

  struct with fields:

          a: 0.1304
          b: -0.2949
  theta1000: 0.0780
  theta0010: -21.1293
  theta0001: -0.3811
       phi0: [1x1 struct]
       phi1: [1x1 struct]
      h2000: [1x1 struct]
      h1100: [1x1 struct]
      h0200: [1x1 struct]
      h3000: [1x1 struct]
      h2100: [1x1 struct]
        K10: [2x1 double]
        K01: [2x1 double]
        K02: [2x1 double]
        K11: [2x1 double]
        K20: [2x1 double]
      h1010: [1x1 struct]
      h1001: [1x1 struct]
      h0110: [1x1 struct]
      h0101: [1x1 struct]
      h2010: [1x1 struct]
      h1110: [1x1 struct]
      h2001: [1x1 struct]
      h1101: [1x1 struct]
      h1002: [1x1 struct]
      h0102: [1x1 struct]
      h1020: [1x1 struct]
      h0120: [1x1 struct]
      h1011: [1x1 struct]
      h0111: [1x1 struct]
          K: @(beta1,beta2)K10*beta1+K01*beta2+1/2*K20*beta1^2
                 +K11*beta1*beta2+1/2*K02*beta2^2
          H: [function_handle]
```

Since the sign of ab is negative, we expect to find stable periodic orbits nearby the Bogdanov–Takens point.

S3.4 Comparing profiles of computed and predicted homoclinic orbits

To test the homoclinic asymptotics from Section 4.1.9 we compare the first and third order asymptotics to the Newton corrected solution. For this, we use the function `C1branch_from_C2point`. This function returns a branch, which by default returns two initial corrected approximations in order to start continuation of the codimension one curve under consideration. By setting the argument `'predictor'` to `true` the approximations are left uncorrected. To make the comparison visually clear, we set the perturbation parameter $\epsilon = 0.1$ (`step = 0.1` in the code below). The code below produces Figure S14. The difference between the two approximations is clearly noticeable. While the first order asymptotics is close to the Newton corrected solution, the third order asymptotics is indistinguishable at this scale from the Newton corrected solution.

```
%% Plot comparing profiles computed and predicted homoclinic orbits
figure(1); clf; hold on;
cm=colormap('lines');
tiledlayout(2,4)
ylabels = {'$x$','$y$'};
step = 0.1;
for order=[1,3]
    [~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
        'debug',0,'codim2','BT','codim1','hcli','step',step,'plot',0, ...
        'generic_unfolding',false,'predictor',true,'order',order);
    [~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
        'order',order,'codim2','BT','codim1','hcli','step',step, ...
        'plot',0,'generic_unfolding',false,'debug',0);
    for i=1:2
        for j=1:2
            ax = nexttile; hold on; title(sprintf('order %d',order))
            plot(ax,hcli_br_approx(i).point(1).mesh, ...
                hcli_br_approx(i).point(1).profile(j,:),'*')
            plot(ax,hcli_br_correc(i).point(1).mesh, ...
                hcli_br_correc(i).point(1).profile(j,:))
            xlabel('$t$','Interpreter','LaTeX');
            ylabel(string(ylabels(j)),'Interpreter','LaTeX')
        end
    end
end
```

S3.5 Continuation of the codimension one curves emanating

To continue the three codimension one curves emanating from the generic Bogdanov–Takens point, we can simply use the function `C1branch_from_C2point`, as shown in the code below. To monitor the continuation process, the argument `plot` must be set to `1`. The most important setting is the perturbation parameter (or multiple), `step` in the code below. If left out, default step sizes are defined. However, depending on the problem, no convergence may then be obtained.

```
%% Continue homoclinic curve emanating from Bogdanov-Takens point
[~,hcli_br,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
    'codim2','BT','codim1','hcli',brpars{:},'step',0.02, ...
```

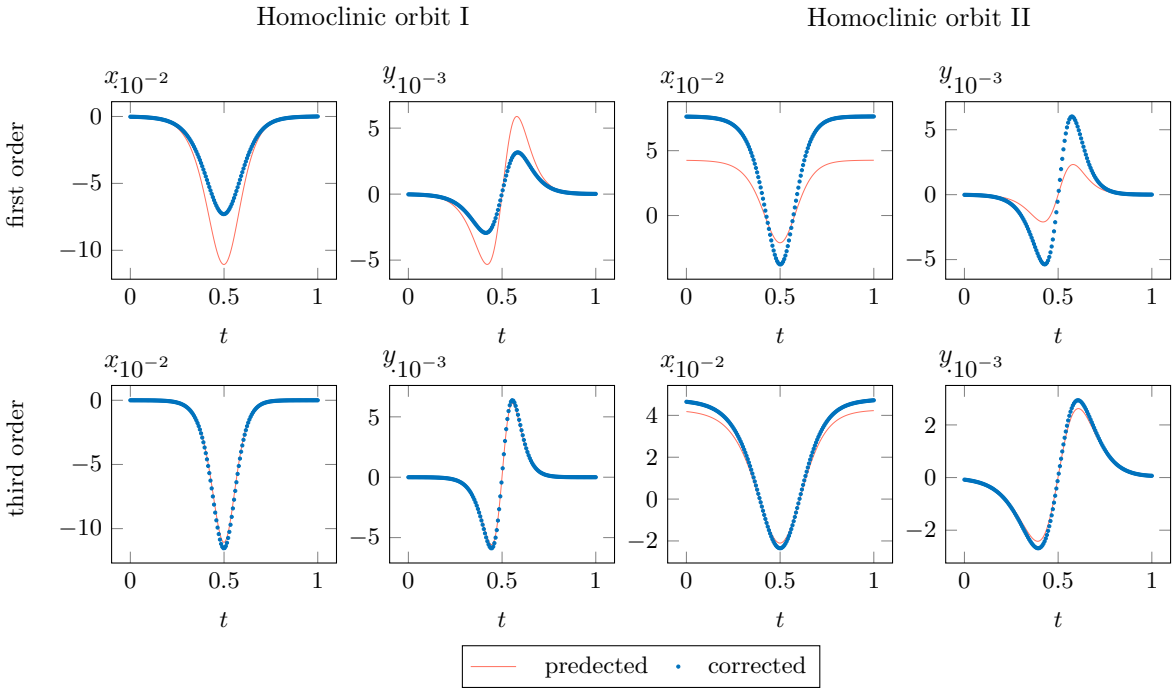

Figure S14: Comparison between the first and third-order homoclinic asymptotics from Section 4.2.3 near the transcritical Bogdanov–Takens bifurcation in (S9) with the perturbation parameter set to $\epsilon = 0.1$.

```
    'plot',0,'generic_unfolding',false);
assert(all(suc(:)>0))
nop=300;[hcli_br(1),suc]=br_contn(funcs,hcli_br(1),nop);assert(suc>0)
nop=300;[hcli_br(2),suc]=br_contn(funcs,hcli_br(2),nop);assert(suc>0)

%% Continue Hopf curve emanating from Bogdanov-Takens point
[~,hbr,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-05, ...
    'plot',0,'generic_unfolding',false);
assert(all(suc(:)>0))
nop=300; [hbr(1),suc]=br_contn(funcs,hbr(1),nop); assert(suc>0)
nop=300; [hbr(2),suc]=br_contn(funcs,hbr(2),nop); assert(suc>0)
[hbr(2),suc]=br_contn(funcs,hbr(2),nop); assert(suc>0)

%% Continue transcritical curve emanating from Bogdanov-Takens point
[~,tc_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,'dir',2, ...
    'codim2','BT','codim1','fold',brpars{:},'step',-1e-4, ...
    'plot',0,'generic_unfolding',false);
assert(all(suc(:)>0))
nop=300; [tc_br,suc]=br_contn(funcs,tc_br(1),nop);assert(suc>0)
tc_br = br_rvers(tc_br);
nop=300; [tc_br,suc]=br_contn(funcs,tc_br(1),nop);assert(suc>0)
```

S3.6 Predictors of the codimension one curves emanating from the Bogdanov–Takens point

Before we provide the bifurcation diagram in the next section, we first obtain the predictors for the codimension one curves. For this, we again use the function `C1branch_from_C2point`. We set the argument `predictor` to `1` and provide a range of perturbation parameters.

```
%% Predictors for homoclinic, Hopf, and transcritical curves
step = [linspace(1e-4,0.1,20);linspace(1e-4,0.1,20)];
[~,hcli_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,'debug',0, ...
    'codim2','BT','codim1','hcli','step',step,'generic_unfolding',false, ...
    'predictor',true);
step = [linspace(0.0001,0.1,20);linspace(0.0001,0.1,20)];
[~,hbr_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf','step',step,'generic_unfolding',false, ...
    'predictor',true);
step = linspace(-0.04,0.04,40);
[~,tc_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold','step',step,'generic_unfolding',false, ...
    'predictor',true);
hcli_br1_pm_pred  = [getpars(hcli_br_pred(1), ind.epsilon), ...
    getpars(hcli_br_pred(1), ind.tau)];
hcli_br2_pm_pred  = [getpars(hcli_br_pred(2), ind.epsilon), ...
    getpars(hcli_br_pred(2), ind.tau)];
hbr1_pm_pred = [getpars(hbr_pred(1),ind.epsilon); getpars(hbr_pred(1),ind.tau)];
hbr2_pm_pred = [getpars(hbr_pred(2),ind.epsilon); getpars(hbr_pred(2),ind.tau)];
tc_br_pm_pred = [getpars(tc_br_pred,ind.epsilon); getpars(tc_br_pred,ind.tau)];
```

S3.7 Bifurcation diagram

The code below produces (a figure similar to) Figure S2.

```
%% Plot the bifurcation curves with predictors
figure(3);clf;hold on
title(['Codimension 1 curves emanating from the transcritical', ...
    'Bogdanov-Takens point with predictors']);
plot(hcli_br1_pm(1,:),hcli_br1_pm(2,:),'.','Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br2_pm(1,:),hcli_br2_pm(2,:),'.','Color',cm(1,:), ...
    'HandleVisibility','off')
plot(hbr1_pm(1,:),hbr1_pm(2,:),'o','Color',cm(2,:),'DisplayName','Hopf branches')
plot(hbr2_pm(1,:),hbr2_pm(2,:),'o','Color',cm(2,:),'HandleVisibility','off')
plot(tc_br_pm(1,:),tc_br_pm(2,:),'d','Color',cm(3,:), ...
    'DisplayName','Transcritical branch')
plot(hcli_br1_pm_pred(:,1),hcli_br1_pm_pred(:,2),'Color', cm(7,:), ...
    'DisplayName','Third order homoclinic asymptotics')
plot(hcli_br2_pm_pred(:,1),hcli_br2_pm_pred(:,2),'Color', cm(7,:), ...
    'HandleVisibility','off')
plot(hbr1_pm_pred(1,:),hbr1_pm_pred(2,:),'--','Color',cm(7,:), ...
```

```
    'DisplayName','Hopf asymptotics')
plot(hbr2_pm_pred(1,:),hbr2_pm_pred(2,:),'--','Color',cm(7,:), ...
    'HandleVisibility','off')
plot(tc_br_pm_pred(1,:),tc_br_pm_pred(2,:),':','Color',cm(7,:), ...
    'DisplayName','Transcritical asymptotics')
plot(bt.parameter(ind.epsilon),bt.parameter(ind.tau),'.k', 'MarkerSize', 18, ...
    'DisplayName','generic Bogdanov-Takens point');
legend
xlabel('$\epsilon$','Interpreter','LaTeX');
ylabel('$\tau$','Interpreter','LaTeX');
```

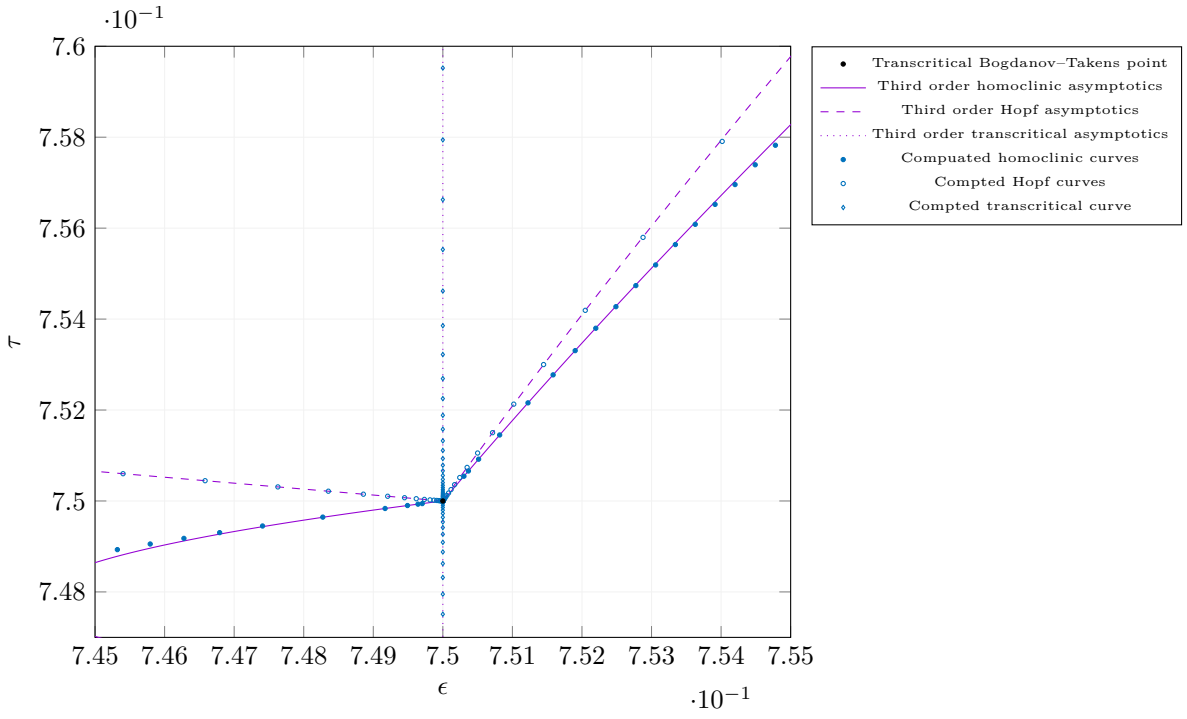

Figure S15: Bifurcation diagram near the derived transcritical Bogdanov-Takens point in (S9) comparing computed codimension one curves using `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.2.3.

S3.8 Compare homoclinic solutions in phase-space

To obtain an impression of the third-order homoclinic asymptotics in phase-space, we compare the corrected and uncorrected homoclinic solutions with the perturbation parameter ranging from 0.01 to 0.03. The code below produces (a figure similar to) Figure S16. We see that the corrected and predicted homoclinic orbits are nearly identical.

```
%% Compare homoclinic solutions in phase-space
step = linspace(0.01,0.03,10);
```

```
[~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'predictor',true,'debug',0, ...
    'generic_unfolding', false);
[~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'debug',0, ...
    'generic_unfolding', false);
figure(5); clf; hold on
title('Compare homoclinic orbits in phase-space')
for i=1:length(hcli_br_approx(1).point)
    for j=1:2
        point = hcli_br_approx(j).point(i);
        profile = point.profile;
        plot3(point.parameter(ind.epsilon)*ones(size(profile(1,:))), ...
            profile(1,:),profile(2,:), 'Color',cm(2,:))
        point_correc = hcli_br_correc(j).point(i);
        profile_correc = point.profile;
        plot3(point_correc.parameter(ind.epsilon)*ones(size(profile(1,:))), ...
            profile_correc(1,:),profile_correc(2,:), '.', 'Color',cm(1,:))
    end
end
xlabel('$\epsilon$','Interpreter','LaTeX')
ylabel('$x_1$','Interpreter','LaTeX')
zlabel('$x_2$','Interpreter','LaTeX')
grid on
view(-46,7)
legend({'Predicted homoclinic order', 'Corrected homoclinic orbir'})
```

S3.9 Convergence plot

Using the function from Section S1.12, we create a log-log convergence plot comparing the convergence order of the first and third order homoclinic asymptotics from Section 4.2.3. The code below yields Figure S17.

```
%% Convergence plot
amplitudes = logspace(-3.4, -1, 20);
orders = [1:3];
relativeErrors = convergence_plot(funcs, bt, orders, amplitudes, 'free_pars',
↪   free_pars,...
    'orders',orders,'TTolerance',9e-6,'generic_unfolding',0,'debug',false);
figure(6); clf; hold on
title('Convergence plot comparing first and third order homoclinic asymptotics')
plot(log10(amplitudes), log10(relativeErrors{1}(:)), '*-')
plot(log10(amplitudes), log10(relativeErrors{3}(:)), '*-')
legend({'first order', 'thrid order'})
xlabel('amplitude')
ylabel('relative error')
```

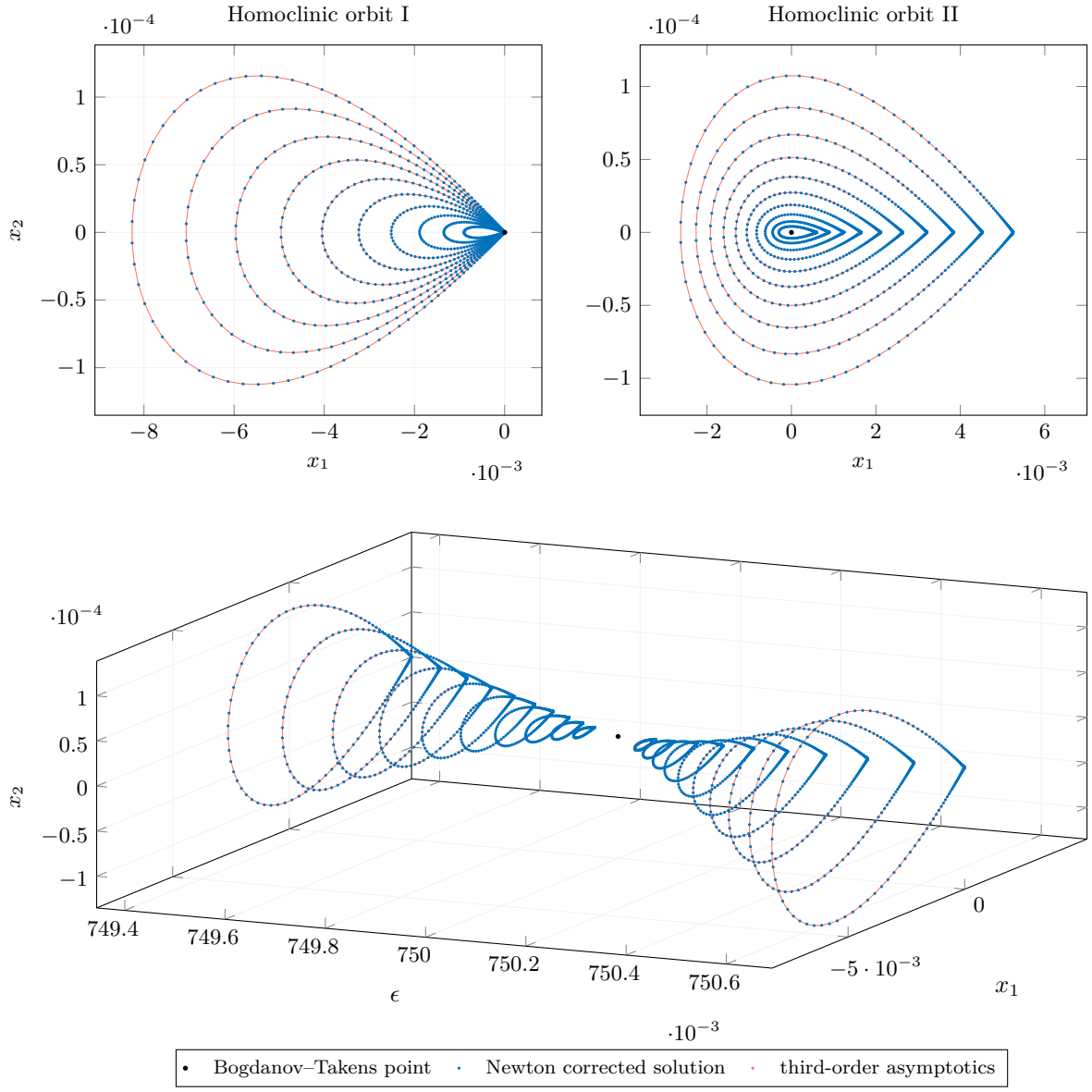

Figure S16: Plot comparing the third-order homoclinic asymptotics from Section 4.2.3 near the transcritical Bogdanov–Taken in (S9) with the Newton correct homoclinic solutions phase-space with the perturbation parameter ϵ ranging from 0.01 to 0.03.

S3.10 Simulation with `DifferentialEquations.jl`

Here we will perform four simulations. The first two simulations will be at two homoclinic orbits located on the two homoclinic curves emanating from the transcritical Bogdanov–Takens point continued with `DDE-BifTool`, see Figure S18. The second two simulations will be in the regions where there should be stable periodic orbits, see Figure S19.

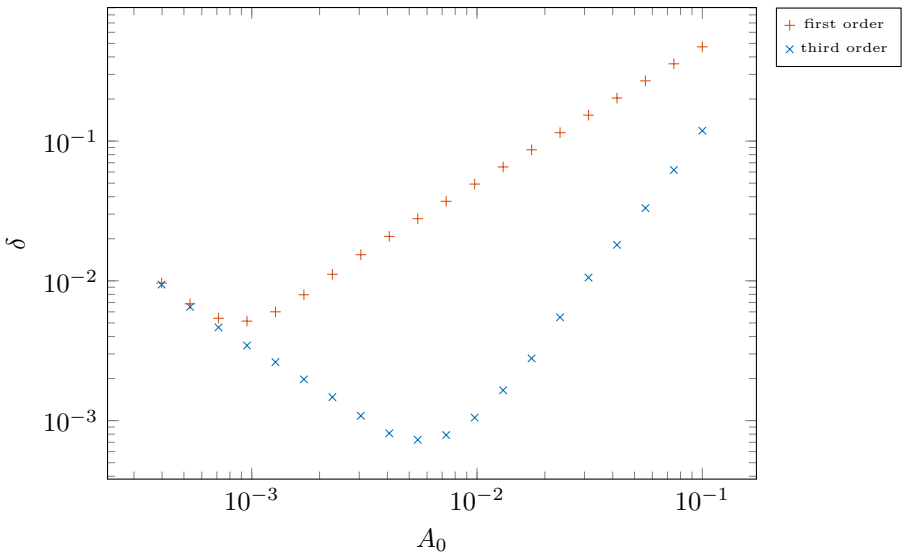

Figure S17: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution `hcli_pred` to the defining system for the homoclinic orbit and the Newton corrected solution `hcli_corrected`.

S3.10.1 Loading necessary Julia packages

Since we do not have analytical expressions for the equilibria, we load the same Julia packages as in the previous demonstration, see Listing S5.

S3.10.2 Define system

We define the system to be integrated and also an allocating version used for stability calculations.

```
# define constants
const τ = 1.0
const c₁ = 0.25
const c₂ = 0.5
const T = 1.0
@inline g(x) = (exp(x) - 1)/(c₁*exp(x) + c₂);

# define model
function vanderPolOscillator!(dx,x,h,p,t)
    ϵ, τ = p
    x₁,x₂ = x
    x₁t, x₂t = h(p, t-T)
    dx[1] = τ*x₂
    dx[2] = τ*(ϵ*g(x₁t) - ϵ*(x₁^2 - 1)*x₂ - x₁)
end

# allocating version (used for calculating stability)
function vanderPolOscillator(x, p)
    dx = similar(x[:,1])
```

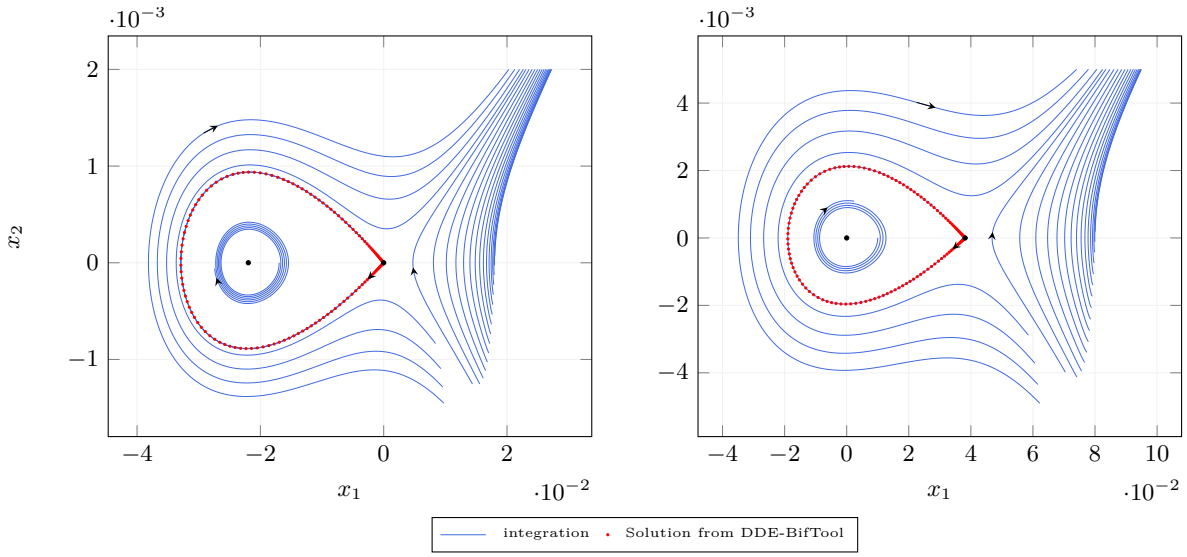

Figure S18: Comparing the computed homoclinic orbits in (S9) with `DDE-BifTool` with the solutions obtained from numerical simulation with Julia. We see the numerical integrated solution going through all the red points from the solution from `DDE-BifTool`.

```
    h(p,t) = x[:,2]
    vanderPolOscillator!(dx,x[:,1],h,p,0)
    dx
end
```

S3.10.3 Functions for plotting arrows

We define a function to show in which direction the orbits flow, which is useful when plotting in phase-space. We also define functions to show the direction of the leading eigenvectors of the characteristic matrix. The code is shown in Listing S4.

S3.10.4 Function for creating streamlines plot

To obtain an impression of the flow near transcritical Bogdanov–Takens point, we create a streamlines function. This is particularly useful for seeing the flow around the stable manifold of the saddle-note.

```
# function for creating streamlines plot
function streamlines(Tend, abscissa, ordinate, p, alg)
    tspan =  (0.0, Tend);
    sols = []
    for x ∈ abscissa
        for y ∈ ordinate
            h(p,t) = [x;y]
            sol =  solve(prob(h,p,tspan),alg,reltol=1e-06,abstol=1e-06)
            push!(sols,sol)
        end
    end
```

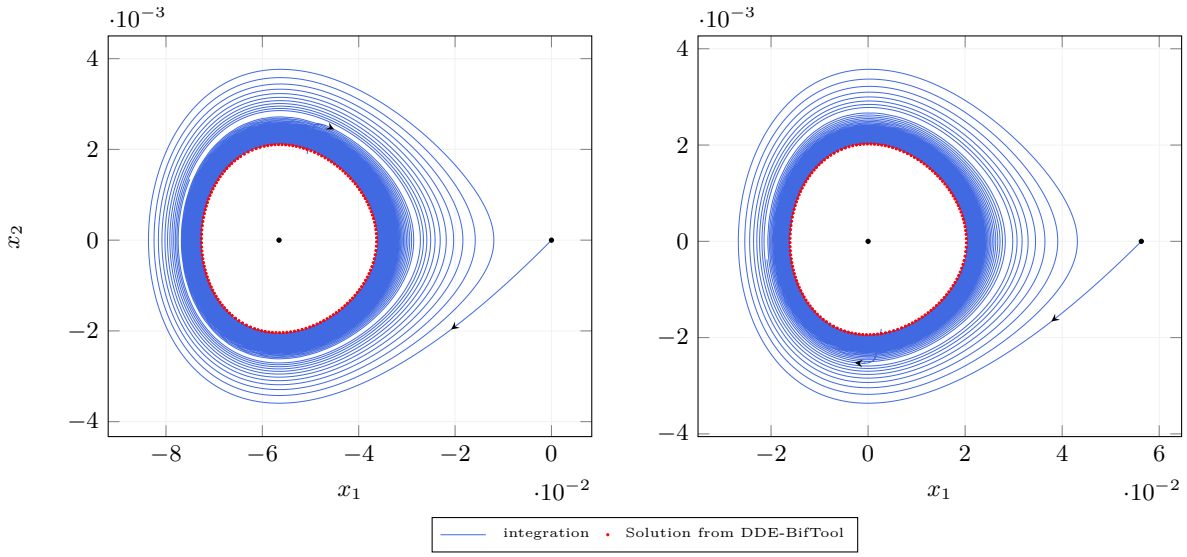

Figure S19: Comparing the computed periodic orbits in (S9) with `DDE-BifTool` with the solutions obtained from numerical simulation with Julia. We see the numerical integrated solution going through all the red points from the solution from `DDE-BifTool`.

```
    sols
end
```

S3.10.5 Create figure with several axes

We create a figure containing multiple axis in which we will plot the two homoclinic and two periodic orbits.

```
# create figure
fig = Figure()
ax1 = Axis(fig[1,1], xlabel=L"x_1", ylabel=L"x_2", title="Homoclinic orbit I")
ax2 = Axis(fig[1,2], xlabel=L"x_1", ylabel=L"x_2", title="Homoclinic orbit II")
ax3 = Axis(fig[2,1], xlabel=L"x_1", ylabel=L"x_2", title="Periodic orbit I")
ax4 = Axis(fig[2,2], xlabel=L"x_1", ylabel=L"x_2", title="Periodic orbit II")
```

S3.10.6 Define parameters, equilibria

We define parameters located on the continued homoclinic branch with `DDE-BifTool`. Then calculate the equilibria points in (S9) near the transcritical Bogdanov–Takens point.

```
## Simulation homoclinic orbit I
ϵ₀, τ₀  =  0.752774810893411,0.754736729675371
p = [ϵ₀; τ₀]

# derive equilbria numerically
y(ϵ, x₁) = ϵ*g(x₁) - x₁
```

```
rts = roots(x₁ -> y(ϵ₀,x₁),interval(-1,1), Newton, 1e-16)
x₁ = sort(mid.(interval.(rts)))
```

S3.10.7 Plot equilibria and homoclinic orbit

By plotting the homoclinic orbit obtained with `DDE-BifTool` located at parameter values

$$(\epsilon_0, \tau_0) = (0.752774810893411, 0.754736729675371),$$

we can compare with the numerical simulations.

```
# plot equilibria
scatter!(ax1,Point2f.([point[i].coords for i ∈ 1:length(x₁)]),color=:black)

# plot homoclinic orbit from DDE-BifTool
csvdir = "/home/maikel/Documents/MyPapers/BTPaper/data/vanderPolOscillator/"
homoclinicorbitI = readdlm(csvdir * "homoclinicorbitI.csv")
scatter!(ax1, homoclinicorbitI, color=:orangered)
```

S3.10.8 Leading eigenvectors

Next, we calculate and plot the leading eigenvectors of the characteristic matrix at the saddle-node bifurcation point.

```
# calculate stability
τs = [0.0 T]
point = [(coords = [x₁[i]; 0], pars = [ϵ₀, τ₀]) for i ∈ 1:length(x₁)]
M = [coefficient_matrices(vanderPolOscillator, point[i], τs)
     for i ∈ 1:length(x₁)]
dep = [DEP([M[i][j](point[i].coords,point[i].pars)
            for j in 1:length(τs)],vec(τs)) for i ∈ 1:length(x₁)]
λ = iar_chebyshev.(dep,maxit=100,neigs=2)
saddle_indx = findall(λ -> abs(λ[1]) < 1e-14, imag([λ[i][1] for i ∈ 1:length(x₁)]))
saddle = point[saddle_indx][]
V = real.(λ[saddle_indx][][2])
indx_unstable = findall(λ -> real(λ) > 0, λ[saddle_indx][][1])
indx_stable = findall(λ -> real(λ) < 0, λ[saddle_indx][][1])
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])
```

S3.10.9 Define callback

Since we are only interested in the flow near the equilibria points, we create a discrete callback to ensure the orbits do not become too large.

```
# create continous callback
condition(u,t,integrator) = u[2] - 0.002
affect!(integrator) = terminate!(integrator)
cb = ContinuousCallback(condition,affect!)
```

S3.10.10 Integrate the system at homoclinic orbits I

Now we define the problem to be integrated and choose the algorithm to be used. Then we integrate the system for a range of initial history functions using the function `streamlines`. Next, we integrate the system near the inner equilibrium, i.e., the equilibrium inside the homoclinic orbit. This equilibria should be an unstable spiral. By using the unstable eigenvector of the characteristic matrix, we obtain a solution going through the homoclinic solution obtained with `DDE-BifTool`.

```
# define DDEProblem and choose integration algorithm
prob(h,p,tspan) = DDEProblem(vanderPolOscillator!, h(p,0),h,tspan,p;
                             constant_lags=[T], callback=cb)
alg = MethodOfSteps(Tsit5())

# create stream lines plot
solsI = streamlines(1000.0, 0.018, range(-0.0015,0.0,20), p, alg);
for sol ∈ solsI
    Tend = sol.t[end]
    lines!(ax1,sol(range(Tend-95*Tend/100,Tend,100)),color=:royalblue1)
    # lines!(ax1,sol,color=:royalblue1)
end

# integrate near inner equilbria
h(p,t) = point[1].coords + [0.005; 0]
tspan =  (0.0, 650.0);
homI_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-06,abstol=1e-09)
lines!(ax1,homI_sol1,color=:royalblue1)

# integrate from the unstable eigenvector
ϵ = sign(Vunstable[1])*1e-04
R(α) = [cos(α) -sin(α);sin(α) cos(α)]
α₀ = -0.030
h(p,t) = saddle.coords - ϵ*R(α₀)*Vunstable
homI_sol2 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax1,homI_sol2,color=:royalblue1)
```

S3.10.11 Add arrows on solutions

Lastly, we add arrows to the obtained solutions using the function `draw_arrow_on_solution` defined above.

```
# add arrows
draw_arrow_on_solution(ax1, homI_sol1, 540, +)
draw_arrow_on_solution(ax1, homI_sol2,  70, +)
draw_arrow_on_solution(ax1,  solsI[1],  50, +)
draw_arrow_on_solution(ax1,  solsI[5],  30, +)
```

We should now obtain an interactive figure similar to the left figure in Figure S18.

S3.10.12 Simulation near stable periodic orbit I

The code for numerical simulation near the second homoclinic orbit, see the right plot in Figure S18, is almost identical to the code above for the first homoclinic orbit and is therefore not included here.

To show by integration the existence of an stable periodic orbit, we first located a periodic orbit in `DDE-BifTool`. This can be done by continuing a branch of periodic orbits emanating from a point on the continued Hopf curves. Then we load the profiles of the periodic orbits into Julia. We perform two simulations. For the first simulation, we integrate with a constant history function equal to a point inside the periodic orbit. The second starts from the unstable eigenvector of the characteristic matrix calculated above.

```
## Simulation periodic orbit I
ϵ₀, τ₀  = 0.757267474481081, 0.762991619845455
p = [ϵ₀; τ₀]

# derive equilbria
rts = roots(x₁ -> y(ϵ₀,x₁),interval(-1,1), Newton, 1e-16)
x₁ = sort(mid.(interval.(rts)))

# calculate stability
τs = [0.0 T]
point = [(coords = [x₁[i]; 0], pars = [ϵ₀, τ₀]) for i ∈ 1:length(x₁)]
M = [coefficient_matrices(vanderPolOscillator, point[i], τs) for i ∈ 1:length(x₁)]
dep = [DEP([M[i][j](point[i].coords,point[i].pars) for j in 1:length(τs)],vec(τs))
  ↪  for i ∈ 1:length(x₁)]
λ = iar_chebyshev.(dep,maxit=100,neigs=2)
saddle_indx = findall(λ -> abs(λ[1]) < 1e-14, imag([λ[i][1] for i ∈ 1:length(x₁)]))
saddle = point[saddle_indx][]
V = real.(λ[saddle_indx][][2])
indx_unstable = findall(λ -> real(λ) > 0, λ[saddle_indx][][1])
indx_stable = findall(λ -> real(λ) < 0, λ[saddle_indx][][1])
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])

# plot equilibria
scatter!(ax3,Point2f.([point[i].coords for i ∈ 1:length(x₁)]),color=:black)

# plot homoclinic orbit from DDE-BifTool
periodicorbitI = readdlm(csvdir * "periodicorbitI.csv")
scatter!(ax3, periodicorbitI[:,[2,3]], color=:orangered)

# create continous callback
condition(u,t,integrator) = true
affect!(integrator) = terminate!(integrator)
cb = ContinuousCallback(condition,affect!)

# integrate from inside periodic orbit I
h(p,t) = periodicorbitI[1,[2,3]] - [0.0; 0.0001]
tspan =  (0.0, 15050.0);
periodicI_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-06,abstol=1e-06)
lines!(ax3,periodicI_sol1(range(0.0,tspan[2],60000)),color=:royalblue1)
```

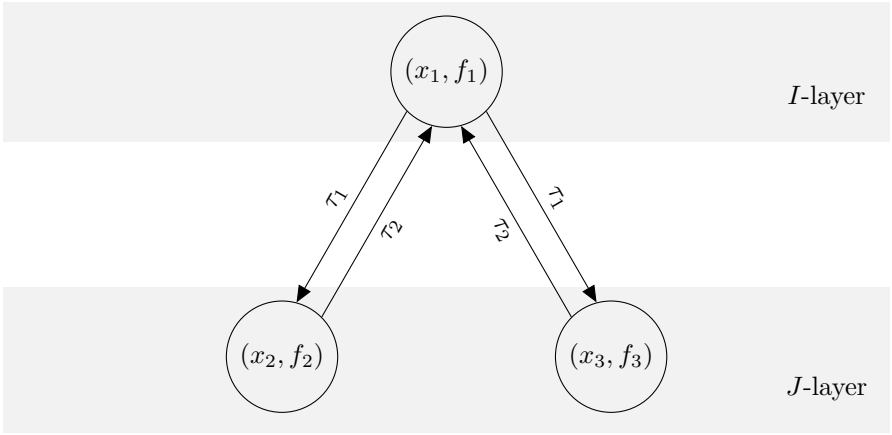

Figure S20: The graph of architecture for model (S10)

```
269
270 # integrate from unstable eigendirection of the saddle-node
271 tspan =  (0.0, 1000.0);
272 ϵ = sign(Vunstable[1])*1e-04
273 h(p,t) = saddle.coords - ϵ*Vunstable
274 periodicI_sol2 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
275 lines!(ax3,periodicI_sol2,color=:royalblue1)
276
277 # add arrows
278 draw_arrow_on_solution(ax3, periodicI_sol1, 10, +)
279 draw_arrow_on_solution(ax3, periodicI_sol2, 100, +)
```

After running the above code, we should obtain a similar plot as in Figure S19.

Remark 15. By the intersection of the orbits in the first simulation, we see that, although the system on the center manifold is equivalent to an ODE, the system we integrate is still a DDE.

S4 Transcritical Bogdanov–Takens bifurcation in a tri-neuron BAM neural network model

We consider a three-component system of a tri-neuron bidirectional associative memory (BAM) neural network model with multiple delays [15]. The architecture of this BAM model is illustrated in Figure S20.

In this model, there is only one neuron with the activation function f_1 on the I-layer and there are two neurons with respective activation functions f_2 and f_3 on the J-layer. We assume that the time delay from the I-layer to the J-layer is τ_1, while the time delay from the J-layer to the I-layer is τ_2. Then the network can be described by the following delay differential equation:

$$\begin{cases} \dot{x}_1(t) = -\mu_1 x_1(t) + c_{21} f_1(x_2(t-\tau_2)) + c_{31} f_1(x_3(t-\tau_2)), \\ \dot{x}_2(t) = -\mu_2 x_2(t) + c_{12} f_2(x_1(t-\tau_1)), \\ \dot{x}_3(t) = -\mu_3 x_3(t) + c_{13} f_3(x_1(t-\tau_1)), \end{cases} \tag{S10}$$

where:

- $x_i(t)\,(i=1,2,3)$ denote the state of the neuron at time t;
- $\mu_i(i=1,2,3)$ describe the attenuation rate of internal neurons processing on the I-layer and the J-layer and $\mu_i>0$;
- the real constants $c_{i1}$and $c_{1i}\,(2,3)$ denote the neurons in two layers: the I-layer and the J-layer.

Letting $u_1(t)=x_1(t-\tau_1), u_2(t)=x_2(t), u_3(t)=x_3(t)$ and $\tau=\tau_1+\tau_2$, then system (S10) is equivalent to the following system:

$$\begin{cases}\dot{u}_1(t)=-\mu_1u_1(t)+c_{21}f_1(u_2(t-\tau))+c_{31}f_1(u_3(t-\tau)),\\ \dot{u}_2(t)=-\mu_2u_2(t)+c_{12}f_2(u_1(t)),\\ \dot{u}_3(t)=-\mu_3u_3(t)+c_{13}f_3(u_1(t)).\end{cases}\tag{S11}$$

Lemma 16. *Assume that $f_i(0)=0\,(i=1,2,3)$, $f_i'(0)\neq 0\,(i=1,2,3)$ and $\mu_2\neq\mu_3$, then the steady-state $(u_1,u_2,u_3)=(0,0,0)$ has a double zero eigenvalue at*

$$c_{21}=c_{21}^0=\frac{\mu_2^2\left(\mu_1\left(\mu_3\tau+1\right)+\mu_3\right)}{c_{12}\left(\mu_2-\mu_3\right)f_1'(0)f_2'(0)},$$
$$c_{31}=c_{31}^0=\frac{\mu_3^2\left(\mu_1\left(\mu_2\tau+1\right)+\mu_2\right)}{c_{13}\left(\mu_3-\mu_2\right)f_1'(0)f_3'(0)}.$$

Proof. The characteristic matrix of (S11) is given by

$$\Delta(\lambda)=\begin{pmatrix}\lambda+\mu_1 & -e^{-\lambda\tau}c_{21}f_1'(0) & -e^{-\lambda\tau}c_{31}f_1'(0)\\ -c_{12}f_2'(0) & \lambda+\mu_2 & 0\\ -c_{13}f_3'(0) & 0 & \lambda+\mu_3\end{pmatrix}.$$

Thus, the characteristic equation becomes

$$\begin{aligned}\det\Delta(\lambda)=\lambda^3+\left(\mu_1+\mu_2+\mu_3\right)\lambda^2+\big(&-c_{12}c_{21}f_1'(0)f_2'(0)e^{-\lambda\tau}\\ &-c_{13}c_{31}f_1'(0)f_3'(0)e^{-\lambda\tau}+\mu_1\mu_2+\mu_3\mu_2+\mu_1\mu_3\big)\lambda\\ &+\mu_1\mu_2\mu_3-e^{-\lambda\tau}\left(c_{12}c_{21}\mu_3f_2'(0)+c_{13}c_{31}\mu_2f_3'(0)\right)f_1'(0)=0.\end{aligned}\tag{S12}$$

Clearly, $\lambda=0$ is a root if and only if

$$\mu_1\mu_2\mu_3=\left(c_{12}c_{21}\mu_3f_2'(0)+c_{13}c_{31}\mu_2f_3'(0)\right)f_1'(0).$$

Taking the derivative of (S12) with respect to λ gives

$$\begin{aligned}\frac{d}{d\lambda}\det\Delta(\lambda)=3\lambda^2+2\left(\mu_1+\mu_2+\mu_3\right)\lambda+\big(&-c_{12}c_{21}f_1'(0)f_2'(0)e^{-\lambda\tau}\\ &-c_{13}c_{31}f_1'(0)f_3'(0)e^{-\lambda\tau}+\mu_1\mu_2+\mu_3\mu_2+\mu_1\mu_3\big)\\ &+\tau\left(c_{12}c_{21}f_2'(0)e^{-\lambda\tau}+c_{13}c_{31}f_3'(0)e^{-\lambda\tau}\right)f_1'(0)\lambda\\ &+\tau e^{-\lambda\tau}\left(c_{12}c_{21}\mu_3f_2'(0)+c_{13}c_{31}\mu_2f_3'(0)\right)f_1'(0)=0.\end{aligned}\tag{S13}$$

Therefore, we have

$$\det\Delta'(0)=\big(-c_{12}c_{21}f_1'(0)f_2'(0)-c_{13}c_{31}f_1'(0)f_3'(0)+\mu_1\mu_2+\mu_3\mu_2+\mu_1\mu_3\big)=0.$$

For any $\tau>0$, it is easy to see that $\det\Delta(\lambda)=\det\Delta'(\lambda)=0$, if and only if the following conditions are satisfied

$$\begin{cases}\left((1-\tau\mu_3)c_{12}c_{21}f_2'(0)+(1-\tau\mu_2)c_{13}c_{31}f_3'(0)\right)f_1'(0)=\mu_1\mu_2+\mu_3\mu_2+\mu_1\mu_3,\\ \\ \left(c_{12}c_{21}\mu_3f_2'(0)+c_{13}c_{31}\mu_2f_3'(0)\right)f_1'(0)=\mu_1\mu_2\mu_3.\end{cases}\tag{S14}$$

By solving (S14) for (c_{21}, c_{31}) we get $(c_{21}, c_{31}) = (c_{21}^0, c_{31}^0)$. Taking the derivative of (S13) yields

$$\begin{aligned}\frac{d^2}{d\lambda^2}\det\Delta(\lambda) = 6\lambda + 2\left(\mu_1+\mu_2+\mu_3\right) + \tau f_1'(0)(c_{12}c_{21}f_2'(0)e^{-\lambda\tau} + c_{13}c_{31}f_3'(0)e^{-\lambda\tau}) \\ + \tau\left(c_{12}c_{21}f_2'(0)e^{-\lambda\tau} + c_{13}c_{31}f_3'(0)e^{-\lambda\tau}\right) f_1'(0) \\ - \tau^2\left(c_{12}c_{21}f_2'(0)e^{-\lambda\tau} + c_{13}c_{31}f_3'(0)e^{-\lambda\tau}\right) f_1'(0)\lambda \\ - \tau^2 e^{-\lambda\tau}\left(c_{12}c_{21}\mu_3 f_2'(0) + c_{13}c_{31}\mu_2 f_3'(0)\right) f_1'(0) = 0. \end{aligned} \tag{S15}$$

Then we can obtain

$$\begin{aligned}&\frac{d^2}{d\lambda^2}\det\Delta(0)|_{(c_{21},c_{31})=(c_{21}^0,c_{31}^0)} \\ &= 2\left(\mu_1+\mu_2+\mu_3\right) + 2\tau f_1'(0)(c_{12}c_{21}^0 f_2'(0) + c_{13}c_{31}^0 f_3'(0)) \\ &\quad - \tau^2 f_1'(0)\left(c_{12}c_{21}^0\mu_3 f_2'(0) + c_{13}c_{31}^0\mu_2 f_3'(0)\right) \\ &= 2\left(\mu_1+\mu_2+\mu_3\right) + \tau\left(\frac{\mu_2^2\left(\mu_1\left(\mu_3\tau+1\right)+\mu_3\right)}{\left(\mu_2-\mu_3\right)} + \frac{\mu_3^2\left(\mu_1\left(\mu_2\tau+1\right)+\mu_2\right)}{\left(\mu_3-\mu_2\right)}\right) \\ &\quad - \tau^2\left(\frac{\mu_2^2\left(\mu_1\left(\mu_3\tau+1\right)+\mu_3\right)}{\left(\mu_2-\mu_3\right)}\mu_3 + \frac{\mu_3^2\left(\mu_1\left(\mu_2\tau+1\right)+\mu_2\right)}{\left(\mu_3-\mu_2\right)}\mu_2\right) \\ &= 2\left(\mu_1+\mu_2+\mu_3\right) + 2\tau\left(\mu_1\mu_2+\mu_1\mu_3+\mu_2\mu_3\right) + \tau^2\mu_1\mu_2\mu_3. \end{aligned}$$

Since $\tau > 0$ and $\mu_i > 0 (i = 1, 2, 3)$ the second derivative of the characteristic equations at $(\lambda, c_{21}, c_{31}) = (0, c_{21}^0, c_{31}^0)$ does not vanish, and we obtain a double zero eigenvalue. □

Lemma 17. Correction to [15, Lemma 3] *Let* $(c_{21}, c_{31}) = (c_{21}^0, c_{31}^0)$,

$$\omega_0 = \frac{\sqrt{-\mu_1^2-\mu_2^2-\mu_3^2+\sqrt{\zeta_0}}}{\sqrt{2}} \tag{S16}$$

and $0 < \tau < \tau_0$, *where* τ_0 *is the minimum positive solution to the nonlinear equation*

$$\tan(\tau\omega_0) = \frac{b_0\zeta_1 - a_0\zeta_2}{a_0\zeta_1 + b_0\zeta_2}, \tag{S17}$$

with

$$\begin{aligned} a_0 &= -\mu_1\mu_2\mu_3, \\ b_0 &= -\omega_0(\mu_2\mu_3 + \mu_1(\mu_2+\mu_3+\mu_2\mu_3\tau)), \\ \zeta_0 &= \mu_1^4 + (\mu_2^2+\mu_3^2)^2 + 8\mu_1\mu_2\mu_3(\mu_2+\mu_3+\mu_2\mu_3\tau) + \\ &\qquad 2\mu_1^2(\mu_3^2 + 4\mu_2\mu_3(1+\mu_3\tau) + \mu_2^2(1+2\mu_3\tau(2+\mu_3\tau))), \\ \zeta_1 &= \mu_1\mu_2\mu_3 - (\mu_1+\mu_2+\mu_3)\omega_0^2, \\ \zeta_2 &= \mu_2\mu_3\omega_0 + \mu_1(\mu_2+\mu_3)\omega_0 - \omega_0^3. \end{aligned}$$

Then all roots of the characteristic equation (S12), except the double zero roots, have negative real parts.

Proof. Consider that there are eigenvalues $\lambda \neq 0$ on the imaginary axis for (c_{21}^0, c_{31}^0). Substituting $\lambda = i\omega, (\omega > 0)$ and (c_{21}^0, c_{31}^0) into (S12), and rearranging terms according to its real and imaginary part yields

$$-\begin{pmatrix}\zeta_1\\ \zeta_2\end{pmatrix} = \begin{pmatrix}a_0 & b_0\\ b_0 & -a_0\end{pmatrix}\begin{pmatrix}\cos\tau\omega\\ \sin\tau\omega\end{pmatrix}. \tag{S18}$$

By squaring and adding the above equations, it follows that

$$\zeta_1^2 + \zeta_2^2 = a_0^2 + b_0^2. \tag{S19}$$

Solving the equation for positive $\omega > 0$ yields (S16).

Next, from (S18), we obtain (S17). First, notice that $\tau \mapsto \omega_0\tau$ is a strictly increasing function since it is the product of two strictly increasing positive functions. It follows that $\tau \mapsto \tan\omega_0\tau$ is periodic in τ with range $\mathbb{R}$. The next observation is that the denominator in the right-hand side of (S17) has a unique positive root $\tau = \tau_1$, at which that numerator does not vanish. In fact, it can be checked that the numerator does never vanish. Lastly, taking the limit of the right-hand side of (S17) of τ to infinity is 0, i.e.

$$\lim_{\tau\to\infty} \frac{-b_0\zeta_1 + a_0\zeta_2}{a_0\zeta_1 + b_0\zeta_2} = 0.$$

By the continuity of the right-hand side, it follows that (S17) has countable many solutions in τ. Let τ_0 be the minimum positive solution. Since for $\tau = 0$ all solutions to the characteristic equation, except of the double zero eigenvalue at the origin, have negative real parts. We conclude by [31, Corollary 2.3] that all eigenvalues, except for the double zero eigenvalues, are located in the left half plane for $0 < \tau < \tau_0$. □

Remark 18. Note that, although $(\omega, \tau) = (\omega_0, \tau_0)$, solves (S17) and (S19), this not necessary means it solves the original equations (S18). Thus, the center manifold may still be stable for values $\tau > \tau_0$. We will demonstrate this in the example below.

For the numerical verification we consider, as in the simulations in [15, Example 1], the system (S11) with the activation functions

$$f_1(x) = \tanh(x) + 0.1x^2, \quad f_2(x) = f_3(x) = \tanh(x), \tag{S20}$$

and parameter values

$$\mu_1 = 0.1, \mu_2 = 0.3, \mu_3 = 0.2, c_{12} = c_{13} = 1, \tau = 5. \tag{S21}$$

Then, from Lemma 16, we obtain two critical values

$$(c_{21}^0, c_{31}^0) = (0.36, -0.22),$$

at which there is a transcritical Bogdanov–Takens point. Furthermore, since $\tau < \tau_0 \approx 5.4320$ the center manifold is locally attractive. In fact, we will show below that the center manifold is locally attractive for $0 < \tau < 13.2309348879375$.

Remark 19. The `MATLAB` files for this demonstration can be found in the directory `demos/tutorial/VII/BAM_neural_network_model` relative to the main directory of the package `DDE-BifTool`. Here, we omit the code to generate a system file. The system file `sym_BAMnn_mf.m` has been generated with the script `sym_BAMnn.m`. Also, we assume that the `DDE-BifTool` package has been loaded as in Listing S1. The code in Sections S4.1 to S4.17 highlights the important parts of the file `BAMnn.m`.

S4.1 Set parameter names and funcs structure

As in the previous example, we set the parameter names and define the `funcs` structure.

```
27 %% Set parameter names
28 parnames={'alpha1','alpha2','tau'};
29 cind=[parnames;num2cell(1:length(parnames))];
30 ind=struct(cind{:});
```

```

%% Set the funcs structure
funcs=set_symfuncs(@sym_BAMnn_mf,'sys_tau',@()ind.tau);
```

S4.2 Set parameter range

Since we are only interested here in the local unfolding, we restrict the allowed parameter range for the unfolding parameters. In practice, one may have physical restrictions which must be satisfied. Additionally, we also limit the maximum allowed step size during continuation. By doing so, we obtain more refined data to compare against our predictors.

```
%% Set bifurcatiun parameter range and step size bounds
brpars={'max_bound', [ind.alpha1 0.5],...
        'min_bound', [ind.alpha1 -0.35],...
        'max_step',  [ind.alpha1 0.01; ind.alpha2 0.01]};
```

S4.3 Stability and coefficients of the transcritical Bogdanov–Takens point

We manually construct a steady-state at the transcritical Bogdanov–Takens point and calculate its stability.

```
%% Define analytically derived Bogdanov-Takens point
% construct steady-state point
stst=dde_stst_create('x',zeros(3,1));
stst.parameter(ind.alpha1) = 0;
stst.parameter(ind.alpha2) = 0;
stst.parameter(ind.tau) = 5;
% Calculate stability
method=df_mthod(funcs,'stst');
stst.stability=p_stabil(funcs,stst,method.stability);
stst.stability.l1
```

The MATLAB console shows the following output.

```
ans =

  -0.0000 + 0.0000i
  -0.0000 - 0.0000i
  -0.2246 + 0.6600i
  -0.2246 - 0.6600i
  -0.6371 + 1.8063i
  -0.6371 - 1.8063i
  -0.8483 + 3.0681i
  -0.8483 - 3.0681i
  -0.9849 + 4.3336i
  -0.9849 - 4.3336i
```

The eigenvalues confirm that the point under consideration is indeed (an approximation to) a Bogdanov–Takens point. Furthermore, the remaining eigenvalues have negative real parts. Next, we calculate the normal form coefficients, the time-reparametrization, and the transformation to the center manifold with the function `nmfm_bt_orbital`, which implements the coefficients as derived in Section 4.2. For this, we need to set the argument `free_pars` to the unfolding parameter (α_1, α_2). These coefficients will be used to start the continuation of the codimension one branches emanating from the Bogdanov–Takens point. Also, since we are in the transcritical case, we set the argument `generic_unfolding` to `false`.

```
%% Calculate parameter-dependent normal form coefficients
free_pars=[ind.alpha1, ind.alpha2];
bt = p_tobt(funcs,stst);
bt = nmfm_bt_orbital(funcs, bt,'free_pars', free_pars, 'generic_unfolding', false);
bt.nmfm
```

The `MATLAB` console shows the following output.

```
ans =

  struct with fields:

          a: 0.0012
          b: -0.0135
  theta1000: -2.5813
  theta0010: 3.1322e+03
  theta0001: -190.0753
       phi0: [1x1 struct]
       phi1: [1x1 struct]
      h2000: [1x1 struct]
      h1100: [1x1 struct]
      h0200: [1x1 struct]
      h3000: [1x1 struct]
      h2100: [1x1 struct]
        K10: [2x1 double]
        K01: [2x1 double]
        K02: [2x1 double]
        K11: [2x1 double]
        K20: [2x1 double]
      h1010: [1x1 struct]
      h1001: [1x1 struct]
      h0110: [1x1 struct]
      h0101: [1x1 struct]
      h2010: [1x1 struct]
      h1110: [1x1 struct]
      h2001: [1x1 struct]
      h1101: [1x1 struct]
      h1002: [1x1 struct]
      h0102: [1x1 struct]
      h1020: [1x1 struct]
      h0120: [1x1 struct]
```

```
    h1011: [1x1 struct]
    h0111: [1x1 struct]
        K: @(beta1,beta2)K10*beta1+K01*beta2+1/2*K20*beta1^2
                +K11*beta1*beta2+1/2*K02*beta2^2
        H: [function_handle]
```

Since the sign of ab is negative, we expect to find stable periodic orbits nearby the Bogdanov–Takens point.

S4.4 Comparing profiles of computed and predicted homoclinic orbits

To test the homoclinic asymptotics from Section 4.1.9 we compare the first and third order asymptotics to the Newton corrected solution. For this, we use the function `C1branch_from_C2point`. This function returns a branch, which by default returns two initial corrected approximations in order to start continuation of the codimension one curve under consideration. By setting the argument `'predictor'` to `true` the approximations are left uncorrected. To make the comparison visually clear, we set the perturbation parameter $\epsilon = 0.02$ (`step = 0.02` in the code below). The code below produces Figures S21 and S22. The difference between the two approximations is clearly noticeable. While the first order asymptotics is close to the Newton corrected solution, the third order asymptotics is indistinguishable at this scale from the Newton corrected solution.

```
%% Plot comparing profiles computed and predicted homoclinic orbits
figure(1); clf; hold on;
cm=colormap('lines');
tiledlayout(2,6)
ylabels = {'u1','u2','u3'};
step = 0.02;
for order=[1,3]
    [~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
        'debug',0,'codim2','BT','codim1','hcli','step',step,'plot',0, ...
        'generic_unfolding',false,'predictor',true,'order',order);
    [~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars, ...
        'order',order,'codim2','BT','codim1','hcli','step',step, ...
        'plot',0,'generic_unfolding',false,'debug',0);
    for i=1:2
        for j=1:3
            ax = nexttile; hold on; title(sprintf('order %d',order))
            plot(ax,hcli_br_approx(i).point(1).mesh, ...
                hcli_br_approx(i).point(1).profile(j,:),'*')
            plot(ax,hcli_br_correc(i).point(1).mesh, ...
                hcli_br_correc(i).point(1).profile(j,:))
            xlabel('$t$','Interpreter','LaTeX');
            ylabel(string(ylabels(j)),'Interpreter','LaTeX')
        end
    end
end
```

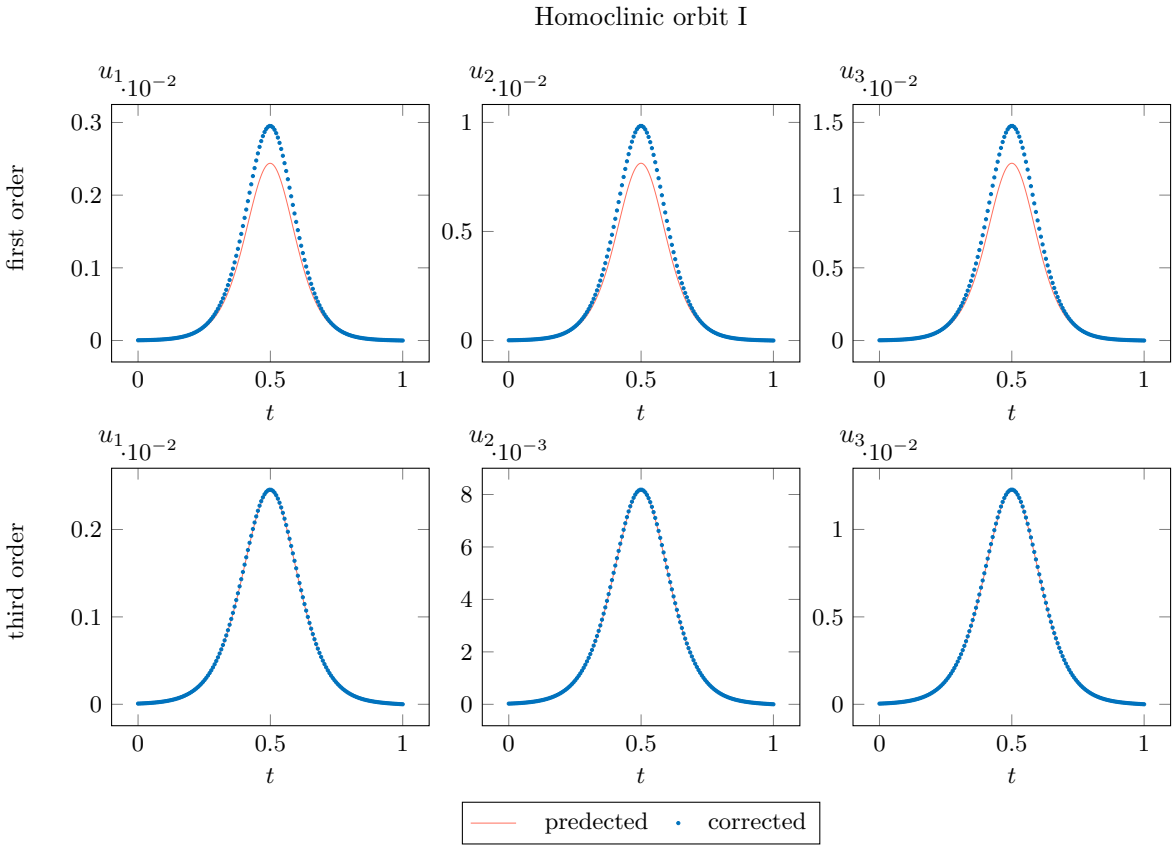

Figure S21: Comparison between the first and third-order homoclinic asymptotics from Section 4.2.3 near the transcritical Bogdanov–Takens bifurcation in (S10) with the perturbation parameter set to $\epsilon = 0.02$.

S4.5 Continuation of the codimension one curves emanating

To continue the three codimension one curves emanating from the generic Bogdanov–Takens point, we can simply use the function `C1branch_from_C2point`, as shown in the code below. To monitor the continuation process, the argument `plot` must be set to `1`. The most important setting is the perturbation parameter (or multiple), `step` in the code below. If left out, default step sizes are defined. However, depending on the problem, no convergence may then be obtained.

```
%% Continue homoclinic curves emanating from Bogdanov-Takens point
[~,hcli_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli',brpars{:},'step',[0.003;0.012], ...
    'plot',0,'generic_unfolding',false); assert(all(suc(:)>0))
hcli_br(1)=br_contn(funcs,hcli_br(1),80);
hcli_br(2)=br_contn(funcs,hcli_br(2),80);

%% Continue Hopf curves emanating from Bogdanov-Takens point
[~,hbr,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-04, ...
    'plot',0,'generic_unfolding',false); assert(all(suc(:)>0))
```

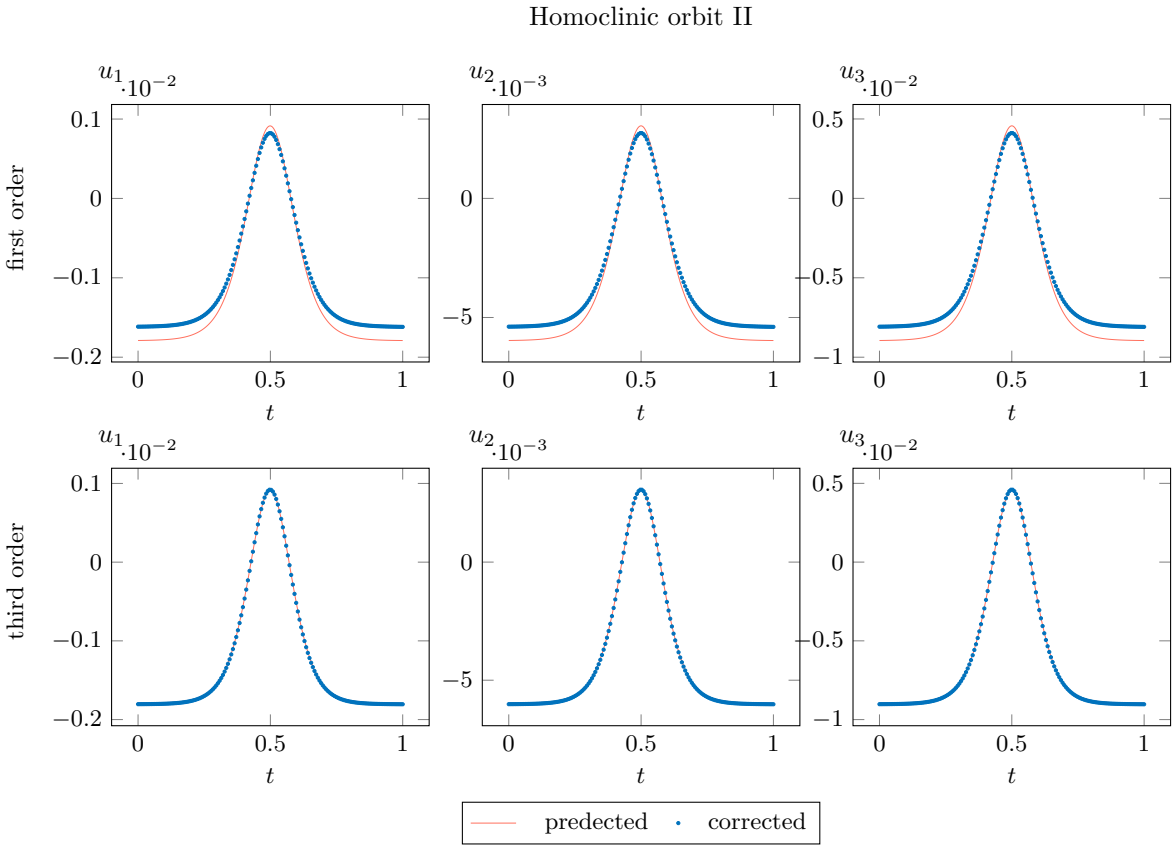

Figure S22: Comparison between the first and third-order homoclinic asymptotics from Section 4.2.3 near the transcritical Bogdanov–Takens bifurcation in (S10) with the perturbation parameter set to $\epsilon = 0.02$.

```
nop=300; [hbr(1),suc]=br_contn(funcs,hbr(1),nop);assert(suc>0)
nop=80; [hbr(2),suc]=br_contn(funcs,hbr(2),nop);assert(suc>0)

%% Continue transcritical curve emanating from Bogdanov-Takens point
[~,tc_br,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold',brpars{:},'step',1e-03, ...
    'plot',0,'generic_unfolding',false); assert(all(suc(:)>0))
nop=50; [tc_br,suc]=br_contn(funcs,tc_br(1),nop);assert(suc>0)
tc_br = br_rvers(tc_br);
nop=50; [tc_br,suc]=br_contn(funcs,tc_br(1),nop);assert(suc>0)
```

S4.6 Predictors of the codimension one curves emanating from the Bogdanov–Takens point

Before we provide the bifurcation diagram in the next section, we first obtain the predictors for the codimension one curves. For this, we again use the function `C1branch_from_C2point`. We set the argument `predictor` to `1` and provide a range of perturbation parameters.

```
%% Predictors for homoclinic, Hopf, and transcritical curves
step = [linspace(1e-4,0.02,20);linspace(1e-4,0.07,20)]
[~,hcli_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,'debug',0,...
    'codim2','BT','codim1','hcli','step',step, ...
    'generic_unfolding',false,'predictor',true);
step = [linspace(0.0001,0.014,20);linspace(0.0001,0.006,20)];
[~,hbr_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf','step',step, ...
    'generic_unfolding',false,'predictor',true);
step = linspace(-0.001,0.001,40);
[~,tc_br_pred,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','fold','step',step, ...
    'generic_unfolding',false,'predictor',true);
hcli_br1_pm_pred  = [getpars(hcli_br_pred(1), ind.alpha1), ...
    getpars(hcli_br_pred(1), ind.alpha2)];
hcli_br2_pm_pred  = [getpars(hcli_br_pred(2), ind.alpha1), ...
    getpars(hcli_br_pred(2), ind.alpha2)];
hbr1_pm_pred = [getpars(hbr_pred(1),ind.alpha1); getpars(hbr_pred(1),ind.alpha2)];
hbr2_pm_pred = [getpars(hbr_pred(2),ind.alpha1); getpars(hbr_pred(2),ind.alpha2)];
tc_br_pm_pred = [getpars(tc_br_pred,ind.alpha1); getpars(tc_br_pred,ind.alpha2)];
```

S4.7 Bifurcation diagram

The code below produces (a figure similar to) Figure S23.

```
%% Plot the bifurcation curves with predictors
figure(3);clf;hold on
title(['Codimension 1 curves emanating from the transcritical ', ...
       'Bogdanov-Takens point with predictors']);
plot(hcli_br1_pm(1,:),hcli_br1_pm(2,:),'.','Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br2_pm(1,:),hcli_br2_pm(2,:),'.','Color',cm(1,:), ...
    'HandleVisibility','off')
plot(hbr1_pm(1,:),hbr1_pm(2,:),'o','Color',cm(1,:),'DisplayName','Hopf branches')
plot(hbr2_pm(1,:),hbr2_pm(2,:),'o','Color',cm(1,:),'HandleVisibility','off')
plot(tc_br_pm(1,:),tc_br_pm(2,:),'d','Color',cm(1,:), ...
    'DisplayName','Transcritical branch')
plot(hcli_br1_pm_pred(:,1),hcli_br1_pm_pred(:,2),'Color', cm(7,:), ...
    'DisplayName','Third order homoclinic asymptotics')
plot(hcli_br2_pm_pred(:,1),hcli_br2_pm_pred(:,2),'Color', cm(7,:), ...
    'HandleVisibility','off')
plot(hbr1_pm_pred(1,:),hbr1_pm_pred(2,:),'--','Color',cm(7,:), ...
    'DisplayName','Hopf asymptotics')
plot(hbr2_pm_pred(1,:),hbr2_pm_pred(2,:),'--','Color',cm(7,:), ...
    'HandleVisibility','off')
plot(tc_br_pm_pred(1,:),tc_br_pm_pred(2,:),':','Color',cm(7,:), ...
    'DisplayName','Transcritical asymptotics')
plot(bt.parameter(ind.alpha1),bt.parameter(ind.alpha2),'.k', ...
    'MarkerSize', 18, 'DisplayName','generic Bogdanov-Takens point');
```

```
legend
xlabel('$\alpha_1$','Interpreter','LaTeX');
ylabel('$\alpha_2$','Interpreter','LaTeX');
axis(1.0e-03*[-0.3696    0.0714   -0.2838    0.2934])
```

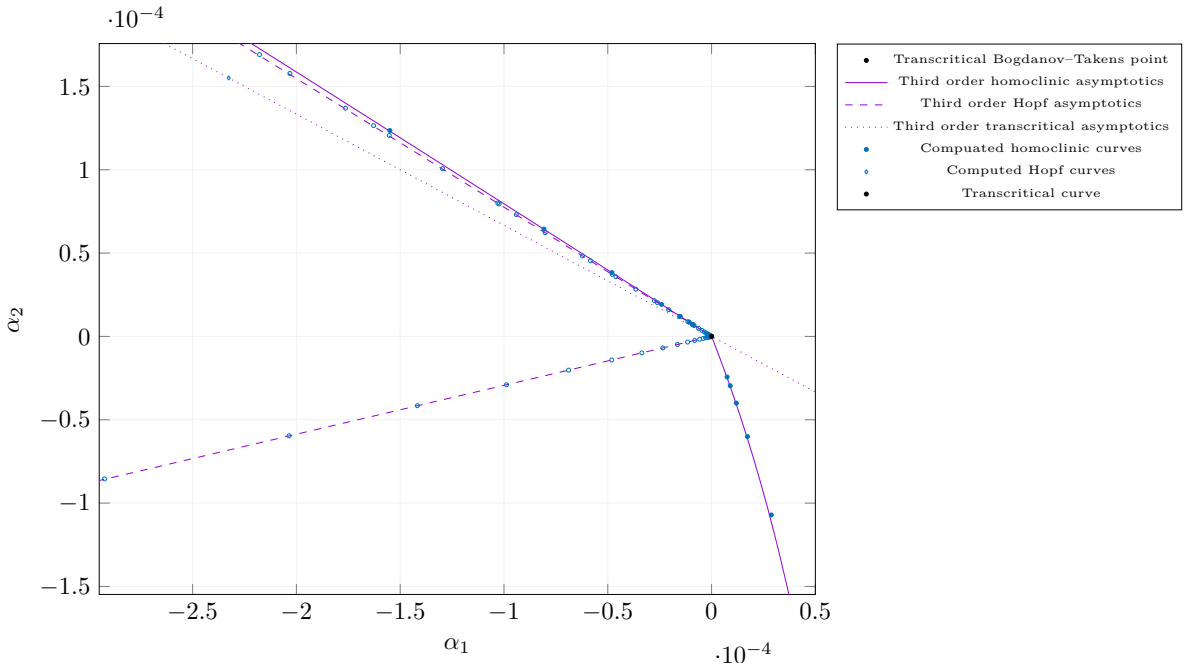

Figure S23: Bifurcation diagram near the analytically derived transcritical Bogdanov–Takens point in (S10) comparing the computed codimension one curves emanating form the Bogdanov–Takens point using `DDE-BifTool` with the third-order homoclinic parameter asymptotics obtained in Section 4.2.3.

S4.8 Detect bifurcations on the second Hopf branch

We can use the `DDE-BifTool` function `LocateSpecialPoints` to locate bifurcation points on the second Hopf branch.

```
%% Detect bifurcations on Hopf branch II
[hbr2_wbifs,hopftests,bifindx,~]=LocateSpecialPoints(funcs,hbr(2));
```

Inspecting the `MATLAB` output gives.

```
HopfCodimension2: calculate stability if not yet present
HopfCodimension2: calculate L1 coefficients
HopfCodimension2: (provisional) 2 gen. Hopf 2 Takens-Bogdanov  detected.
br_insert: detected 1 of 4: genh. Normalform:
    L2: 5.7933e+03
    L1: 1.0106e-09
```

```
br_insert: detected 2 of 4: BT. Normalform:
    a2: -5.8499e-04
    b2: -0.0031

br_insert: detected 3 of 4: genh. Normalform:
    L2: -5.7933e+03
    L1: -2.1214e-09

br_insert: detected 4 of 4: BT. Normalform:
    a2: -0.0012
    b2: 0.0135
```

Thus, there are two Bogdanov–Takens points detected. By inspection of the normal form coefficients a and b (`a2` and `b2` in the output above) with the coefficients of the transcritical Bogdanov–Takens point, we see that one is (very likely) already known. Indeed, while continuing the second Hopf curve from the transcritical Bogdanov–Takens point, we encounter another Bogdanov–Takens point, at which the Hopf curve turns around, and continues in the reverse direction, back to the transcritical Bogdanov–Takens point. Similarly, we also deduce that there is only one generalized Hopf point detected on the Hopf curve. Inspecting the parameters indeed confirm our claim.

Since the newly detected Bogdanov–Takens point is on the Hopf curve on which the equilibrium changes position under variation of the parameters, we extract the Bogdanov–Takens point and computed the coefficients of the generic case.

```
179 bt2 = hbr2_wbifs.point(bifindx(2));
180 bt2 = p_tobt(funcs,bt2);
181 bt2 = nmfm_bt_orbital(funcs, bt2,'free_pars',free_pars);
```

S4.9 Plot Bogdanov-Takens test function along the Hopf curve

Before we continue the homoclinic branch emanating from the generic Bogdanov–Takens point, we first plot the test function for the Bogdanov–Takens point, i.e. we plot the imaginary part for the critical eigenvalues along the second Hopf curve. The code below produces (a figure similar to) Figure S24.

```
185 %% Plot Bogdanov-Takens test function along Hopf curve
186 figure(4);clf;hold on
187 title('Bogdanov--Takens testfunction along Hopf bifurcation curve')
188 plot3(getpars(hbr2_wbifs,ind.alpha1),getpars(hbr2_wbifs,ind.alpha2),hopftests.bt)
189 plot3(bt.parameter(ind.alpha1), bt.parameter(ind.alpha2), ...
190     hopftests.bt(bifindx(4)), '*k')
191 plot3(bt2.parameter(ind.alpha1), bt2.parameter(ind.alpha2), ...
192     hopftests.bt(bifindx(2)), '*k')
193 xlabel('$\alpha_1$','Interpreter','LaTeX');
194 ylabel('$\alpha_2$','Interpreter','LaTeX');
195 zlabel('\omega')
196 grid on
197 view(-47, 29)
```

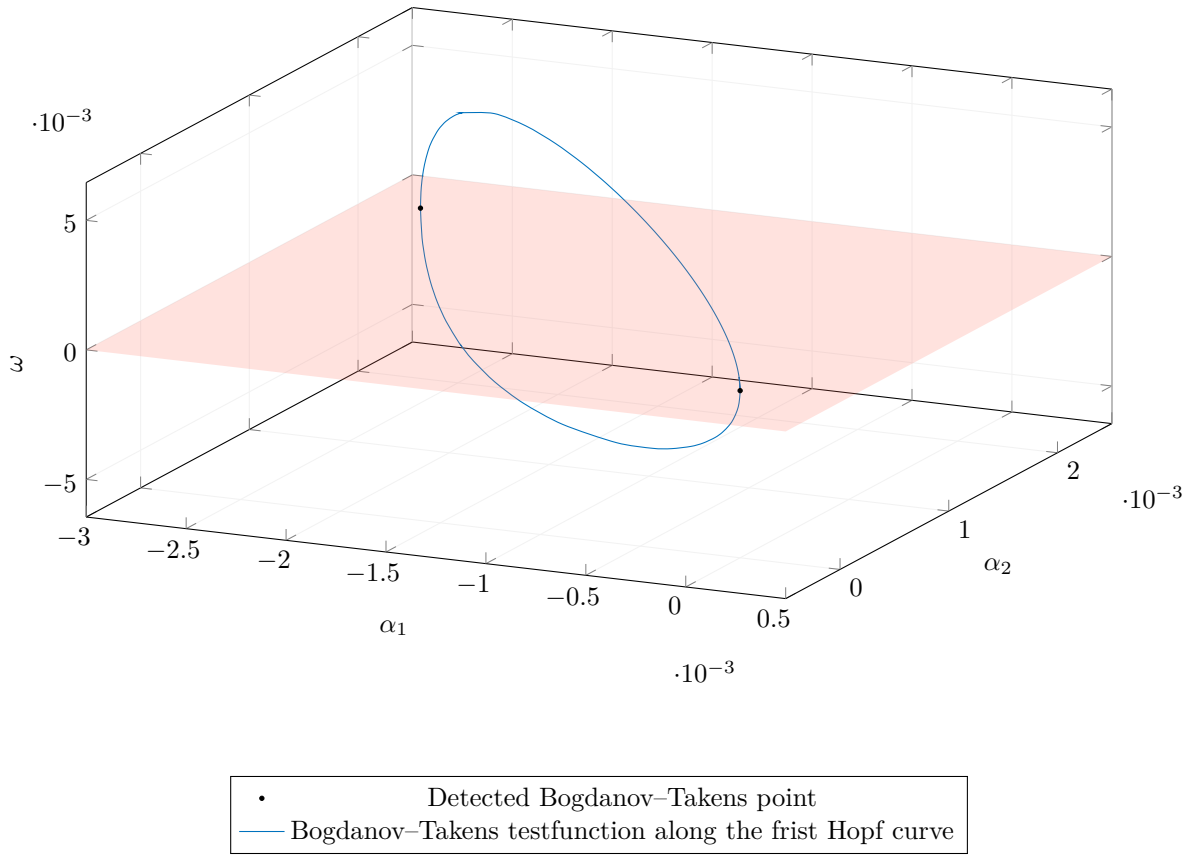

Figure S24: Plot of the Bogdanov–Takens test function along the second Hopf curve. We see that surface $\omega = 0$ is intersected transversally two times while continuing the Hopf curve.

S4.10 Continue third homoclinic curve

The code below continuous the homoclinic solution emanating from the generic Bogdanov–Takens detected on the second Hopf branch. The last line extracts the parameters used for plotting.

```
%% Continue homoclinic solutions emanating from the detected BT point
[~,hcli_br(3),suc]=C1branch_from_C2point(funcs,bt2,free_pars,...
    'codim2','BT','codim1','hcli',brpars{:},'step',0.002,'plot',0);
assert(all(suc(:)>0))
hcli_br(3) = br_contn(funcs,hcli_br(3),300);
hcli_br3_pm = [getpars(hcli_br(3),ind.alpha1), getpars(hcli_br(3),ind.alpha2)]';

```

S4.11 Bifurcation plot

We are now in the position to recreate the bifurcation plot as shown in the main text. There, we left out the predictors for the codimension one equilibria bifurcation curves and changed the color of the computed codimension one equilibria curves to gray. In this way, focus will be on the homoclinic

curves. The code below produces (a figure similar to) Figure S25.

```
%% Plot the bifurcation curves with predictors
figure(5);clf;hold on
title(['Codimension 1 curves emanating from the generic ', ...
       'and transcritical Bogdanov-Takens point with predictors']);
plot(hcli_br1_pm(1,:),hcli_br1_pm(2,:),'.','Color',cm(1,:), ...
    'DisplayName','Homoclinic branches', 'MarkerSize', 20)
plot(hcli_br2_pm(1,:),hcli_br2_pm(2,:),'.','Color',cm(1,:), ...
    'HandleVisibility','off', 'MarkerSize', 20)
plot(hcli_br3_pm(1,:),hcli_br3_pm(2,:),'.','Color',cm(1,:), ...
    'HandleVisibility','off', 'MarkerSize', 20)
plot(hbr1_pm(1,:),hbr1_pm(2,:),'Color','#ccc', ...
    'DisplayName','Hopf branches')
plot(hbr2_pm(1,:),hbr2_pm(2,:),'Color','#ccc','HandleVisibility','off')
plot(tc_br_pm(1,:),tc_br_pm(2,:),'--','Color','#ccc', ...
    'DisplayName','Transcritical branch')
plot(hcli_br1_pm_pred(:,1),hcli_br1_pm_pred(:,2),'Color', cm(7,:), ...
    'DisplayName','Third order homoclinic asymptotics')
plot(hcli_br2_pm_pred(:,1),hcli_br2_pm_pred(:,2),'Color', cm(7,:), ...
    'HandleVisibility','off')
plot(hcli_br3_pm_pred(:,1),hcli_br3_pm_pred(:,2),'Color', cm(7,:), ...
    'HandleVisibility','off')
plot(bt.parameter(ind.alpha1),bt.parameter(ind.alpha2),'ok', ...
    'DisplayName','Transcritical Bogdanov-Takens point', ...
    'MarkerFaceColor',[0 0 0], 'MarkerSize', 09);
plot(bt2.parameter(ind.alpha1),bt2.parameter(ind.alpha2),'sk', ...
    'DisplayName','generic Bogdanov-Takens point','MarkerFaceColor', ...
    [0 0 0], 'MarkerSize', 10);
legend
xlabel('$\alpha_1$','Interpreter','LaTeX');
ylabel('$\alpha_2$','Interpreter','LaTeX');
axis([-0.0030, 0.0005, -0.0010, 0.0021])
```

S4.12 Large view bifurcation plot without predictors

Although the first and third homoclinic bifurcation curves exist only in a very small parameter region, the second is continued in a relatively large parameter range. The code below produces (a figure similar to) Figure S26.

```
%% Plot larger bifurcation bifurcation diagram without predictors
figure(6);clf;hold on
tiledlayout(1,2)
hold on;
title(['Codimension 1 curves emanating from the generic ', ...
       'and transcritical Bogdanov-Takens point with predictors']);
plot(hcli_br1_pm(1,:),hcli_br1_pm(2,:),'Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br2_pm(1,:),hcli_br2_pm(2,:),'Color',cm(1,:),'HandleVisibility','off')
```

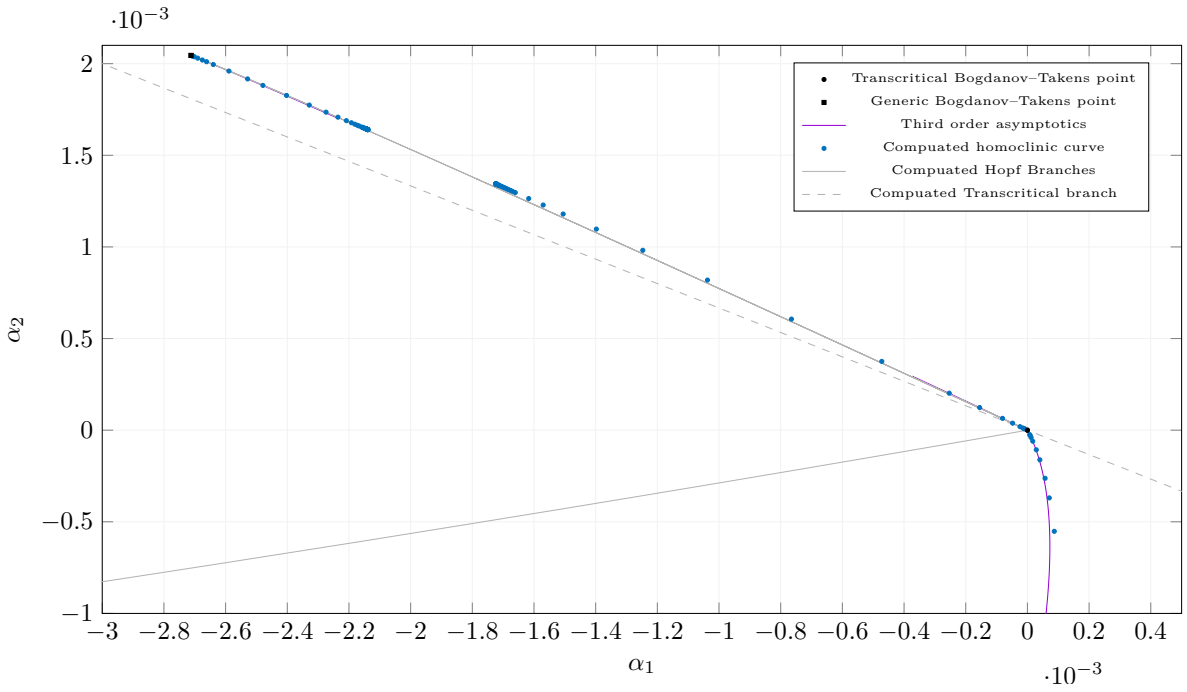

Figure S25: Bifurcation diagram near the transcritical and generic Bogdanov–Takens bifurcation points in (S11) comparing computed codimension one curves using `DDE-BifTool` with the asymptotics obtained in the main text.

```
plot(hcli_br3_pm(1,:),hcli_br3_pm(2,:),'Color',cm(1,:),'HandleVisibility','off')
plot(hbr1_pm(1,:),hbr1_pm(2,:),'Color',cm(2,:),'DisplayName','Hopf branches')
plot(hbr2_pm(1,:),hbr2_pm(2,:),'Color',cm(2,:),'HandleVisibility','off')
plot(tc_br_pm(1,:),tc_br_pm(2,:),'Color',cm(3,:), ...
    'DisplayName','Transcritical branch')
plot(bt.parameter(ind.alpha1),bt.parameter(ind.alpha2),'ok', ...
    'DisplayName','Transcritical Bogdanov-Takens point', ...
    'MarkerFaceColor',[0 0 0], 'MarkerSize', 09);
plot(bt2.parameter(ind.alpha1),bt2.parameter(ind.alpha2),'sk', ...
    'DisplayName','generic Bogdanov-Takens point', ...
    'MarkerFaceColor',[0 0 0], 'MarkerSize', 10);
legend
xlabel('$\alpha_1$','Interpreter','LaTeX');
ylabel('$\alpha_2$','Interpreter','LaTeX');
```

S4.13 Homoclinic orbits in phase-spase

To obtain an expression of the continued homoclinic orbits, we plot the solutions on the various homoclinic branches in phase-space. The code below produces (a figure similar to) Figure S27. In the top left plot, the homoclinic orbits connected to the origin are shown. However, we see that this does not provide much insight. One way to visualize homoclinic solutions is by inspection the profiles of the solutions. This will be done in the next section. Another way to reveal the homoclinic solutions

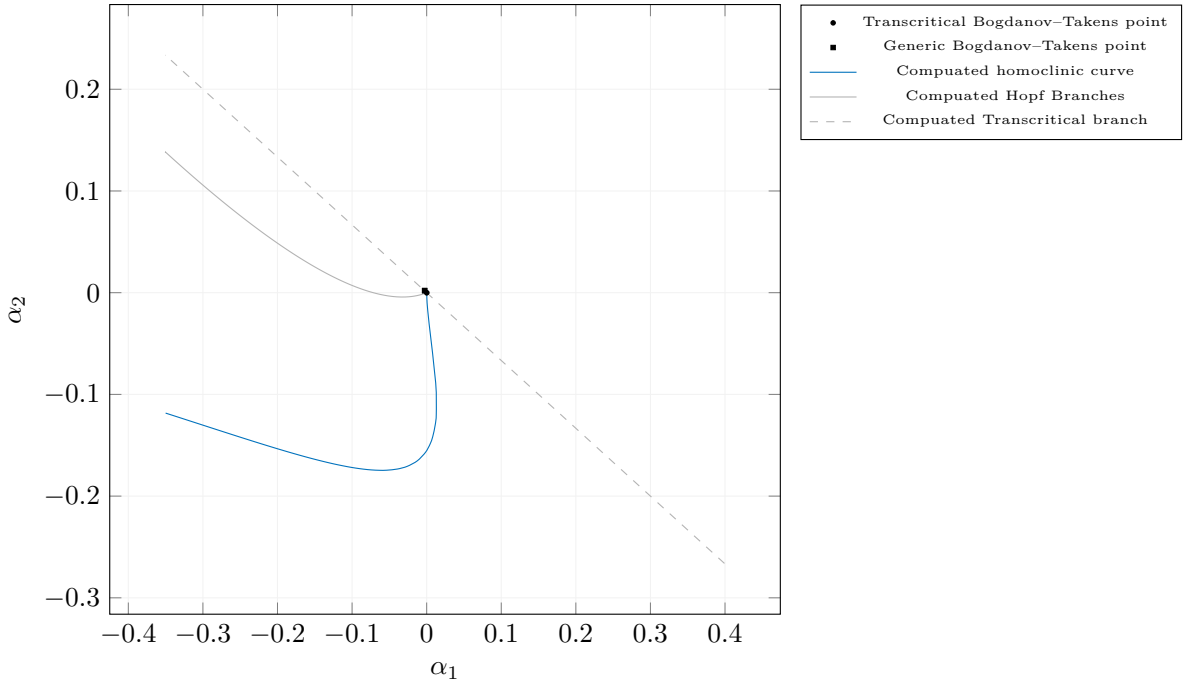

Figure S26: Bifurcation diagram near the transcritical and a generic Bogdanov–Takens bifurcation points in (S11) showing the fully continued homoclinic branch connected to the non-trivial equilibrium emanating from the transcritical Bogdanov–Takens point.

connected to the origin is by rotating the coordinates (u_1, u_2). The result is seen in the top right plot in Figure S27. In the bottom left plot the solutions on the second homoclinic branch emanating from the transcritical Bogdanov–Takens point are shown. Lastly, we plotted the rotated homoclinic orbits emanating from the generic Bogdanov–Takens point in the bottom right plot of Figure S28.

```
%% Plot continued homoclinic orbits I
figure(7); clf
title('Continued homoclinic orbits in phase-space');
tiledlayout(1,5)
nexttile; hold on;
for i=1:length(hcli_br(1).point)
    profile = hcli_br(1).point(i).profile;
    plot3(profile(1,:),profile(2,:),profile(3,:), 'Color',cm(1,:))
end
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')

%% Plot continued homoclinic orbits I (rotated)
nexttile; hold on;
```

```
R = @(alpha) [cos(alpha) -sin(alpha); sin(alpha) cos(alpha)]
for i=1:length(hcli_br(1).point)
    profile = hcli_br(1).point(i).profile;
    profileRotated = R(0.29)*profile([1,2],:)
    plot3(profileRotated(1,:),profileRotated(2,:),profile(3,:), 'Color',cm(1,:))
end
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')

%% Plot continued homoclinic orbits II
nexttile; hold on;
for i=1:length(hcli_br(2).point)
    profile = hcli_br(2).point(i).profile;
    plot3(profile(1,:),profile(2,:),profile(3,:), 'Color',cm(1,:))
end
profile = hcli_br(2).point(end).profile;
plot3(profile(1,:),profile(2,:),profile(3,:), 'Color',cm(2,:))
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')

%% Plot continued homoclinic orbits III
nexttile; hold on;
for i=1:length(hcli_br(3).point)
    profile = hcli_br(3).point(i).profile;
    plot3(profile(1,:),profile(2,:),profile(3,:), 'Color',cm(1,:))
end
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')

%% Plot continued homoclinic orbits III (rotated)
nexttile; hold on;
for i=1:length(hcli_br(3).point)
    profile = hcli_br(3).point(i).profile;
    profileRotated = R(0.29)*profile([1,2],:)
    plot3(profileRotated(1,:),profileRotated(2,:),profile(3,:), 'Color',cm(1,:))
end
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')
```

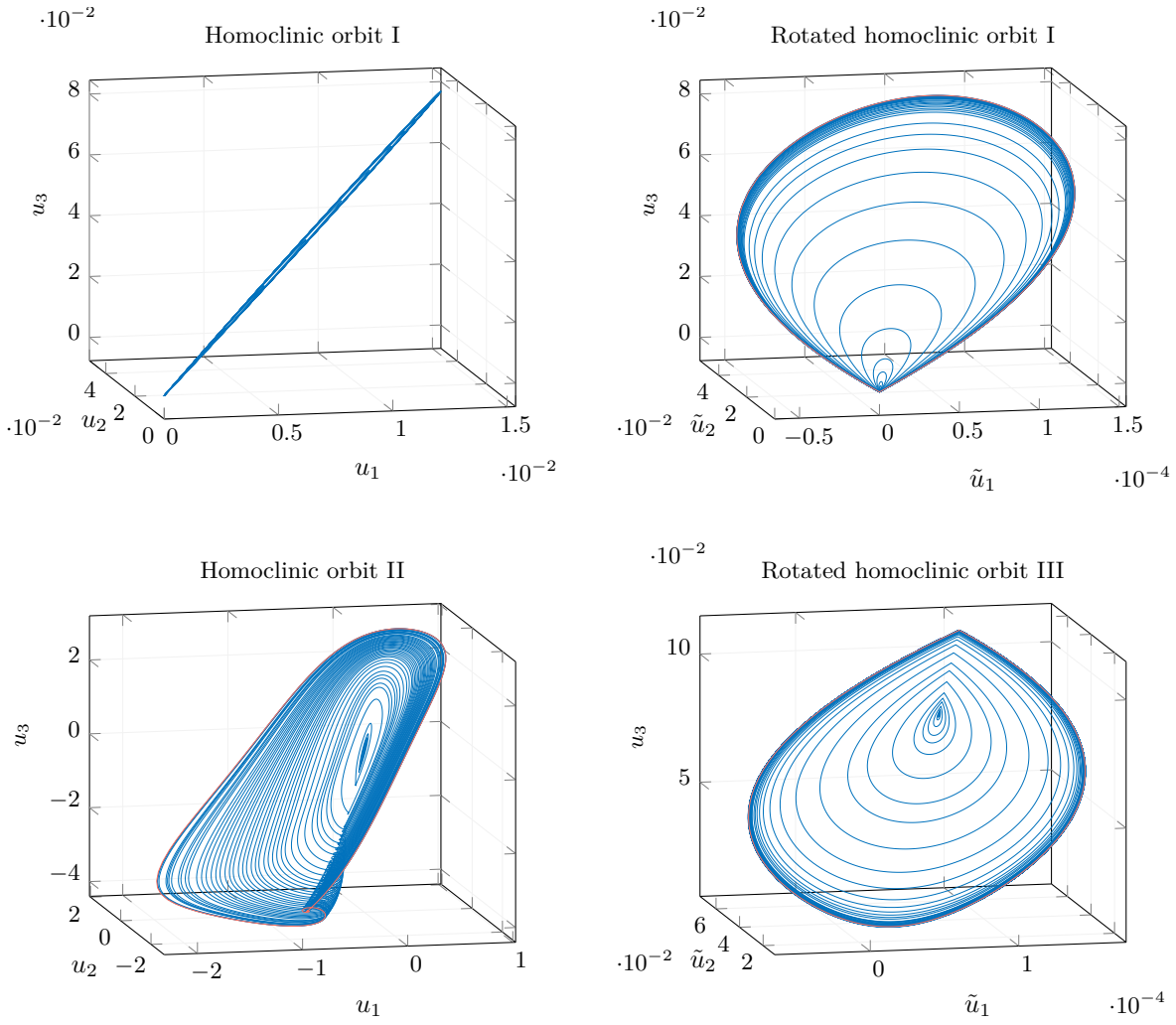

Figure S27: Plots of homoclinic solutions emanating from the generic and transcritical Bogdanov–Takens bifurcations in (S11). In the top left plot the homoclinic solutions emanating from the transcritical Bogdanov–Takens point with fixed saddle points are shown. In the top right plot, we rotated the (u_1, u_2) coordinates in order to make these homoclinic solutions visible. The bottom left plot shows the second branch of homoclinic solutions emanating from the transcritical Bogdanov–Takens point. Lastly, in the bottom right plot, the homoclinic solutions emanating from the generic Bogdanov–Takens point are shown. Here also the (u_1, u_2) coordinates are rotated.

S4.14 Compare homoclinic solutions in phase-space

Here, we compare the corrected and uncorrected profiles of the homoclinic solutions on the two homoclinic curves emanating from the transcritical Bogdanov–Takens point with the perturbation parameter ranging from 0.003 to 0.009. The code below produces (a figure similar to) Figure S28. We see that the corrected and predicted homoclinic orbits are nearly identical.

```
%% Compare homoclinic solutions in phase-space
step = linspace(0.003,0.009,10);
```

```
[~,hcli_br_approx,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'predictor',true,'debug',0, ...
    'generic_unfolding', false);
[~,hcli_br_correc,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hcli','step',step,'debug',0, ...
    'generic_unfolding', false);
figure(9); clf; hold on
title('Compare homoclinic orbits in phase-space')
for i=1:length(hcli_br_approx(1).point)
    for j=1:2
        point = hcli_br_approx(j).point(i);
        profile = point.profile;
        mesh = point.mesh;
        plot3(point.parameter(ind.alpha2)*ones(size(profile(1,:))), ...
            mesh,profile(2,:), 'Color',cm(2,:))
        point_correc = hcli_br_correc(j).point(i);
        profile_correc = point.profile;
        mesh_correc = point_correc.mesh;
        plot3(point_correc.parameter(ind.alpha2)*ones(size(profile(1,:))), ...
            mesh,profile_correc(2,:), '.', 'Color',cm(1,:))
    end
end
xlabel('$\epsilon$','Interpreter','LaTeX')
ylabel('$\tilde t$','Interpreter','LaTeX')
zlabel('$u_1$','Interpreter','LaTeX')
grid on
view(-46,7)
legend({'Predicted homoclinic order', 'Corrected homoclinic orbir'})
```

S4.15 Continue periodic solutions from Hopf branch to homoclinic branch

We continue a branch of periodic solutions emanating from point number 29 on the first Hopf branch emanating from the transcritical Bogdanov–Takens point. The periodic solution converges to a homoclinic orbit located on the second homoclinic branch emanating from transcritical Bogdanov–Takens point. We will use point number 28 on the periodic solution branch to compare against the simulation in Julia in Section S4.18. The code below produces (a figure similar to) Figure S29.

```
%% Continue periodic orbit from Hopf point on the first Hopf branch
psol_br1=SetupPsol(funcs,hbr(1),29,'contpar',ind.alpha1, ...
    'degree',3,'intervals',50,brpars{1:4},'max_step',[0,inf],'plot',0,'radius',1e-03);
psol_br1=br_contn(funcs,psol_br1,40);
psol_br1_pm = [getpars(psol_br1,ind.alpha1); getpars(psol_br1,ind.alpha2)]';
```

```
%% Plot periodic branch
% we see that the perioidic orbit converges to a homoclinic orbit
figure(10); clf; hold on;
title('Continued periodic orbits in phase-space');
```

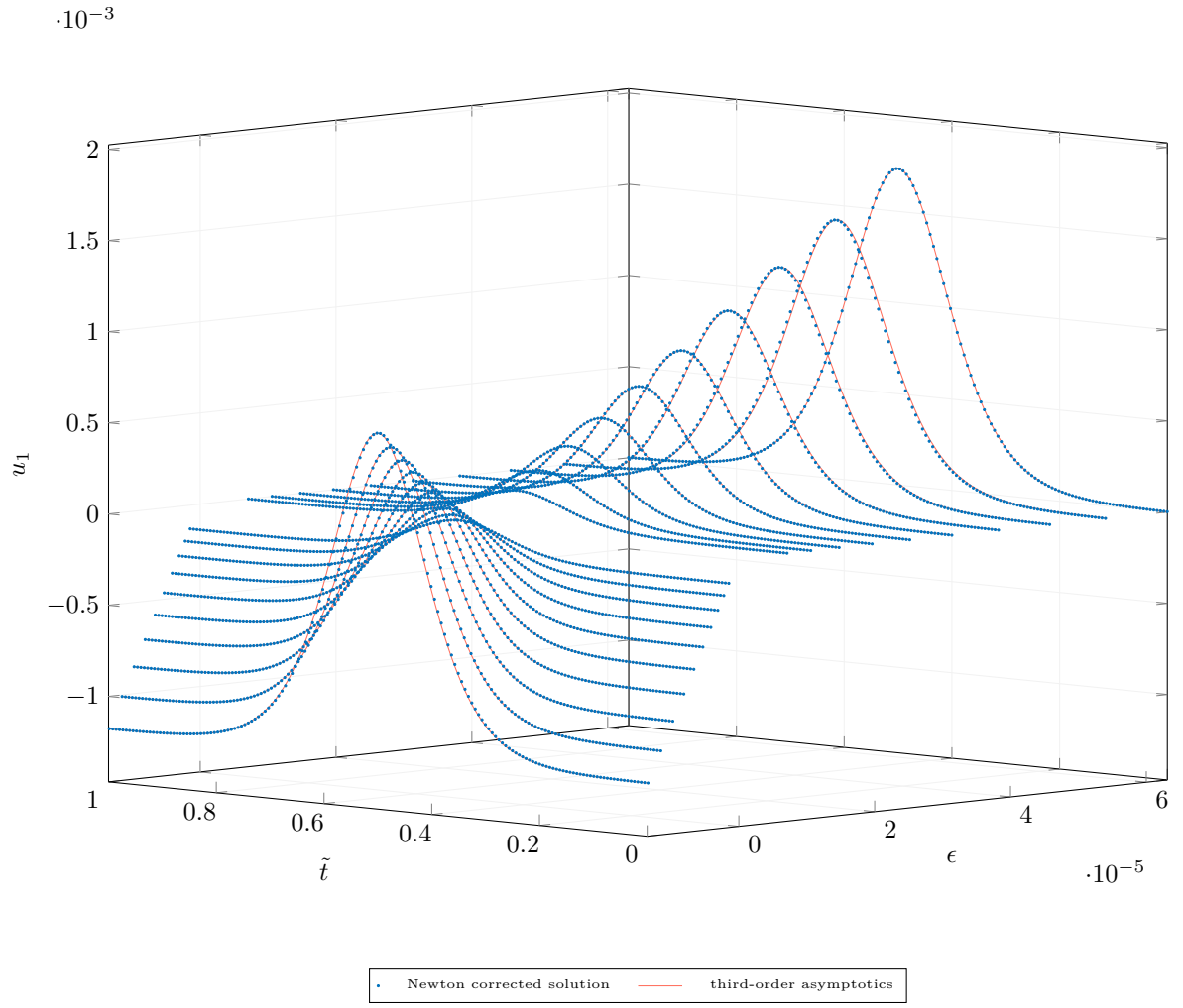

Figure S28: Plot comparing the third-order homoclinic asymptotics with the Newton correct homoclinic solutions in $(\epsilon, \tilde{t}, u_1)$ phase-space. Here $\tilde{t}$ is the time t rescaled to the interval $[0, 1]$.

```
for i=1:length(psol_br1.point)
    profile = psol_br1.point(i).profile;
    profileRotated = R(0.29)*profile([1,2],:);
    plot3(profileRotated(1,:),profileRotated(2,:),profile(3,:), 'Color',cm(1,:))
end
profile = psol_br1.point(end).profile;
profileRotated = R(0.29)*profile([1,2],:);
plot3(profileRotated(1,:),profileRotated(2,:),profile(3,:),'.', 'Color',cm(2,:))
grid on
view(-12,12)
xlabel('$u_1$','Interpreter','LaTeX')
ylabel('$u_2$','Interpreter','LaTeX')
zlabel('$u_3$','Interpreter','LaTeX')
```

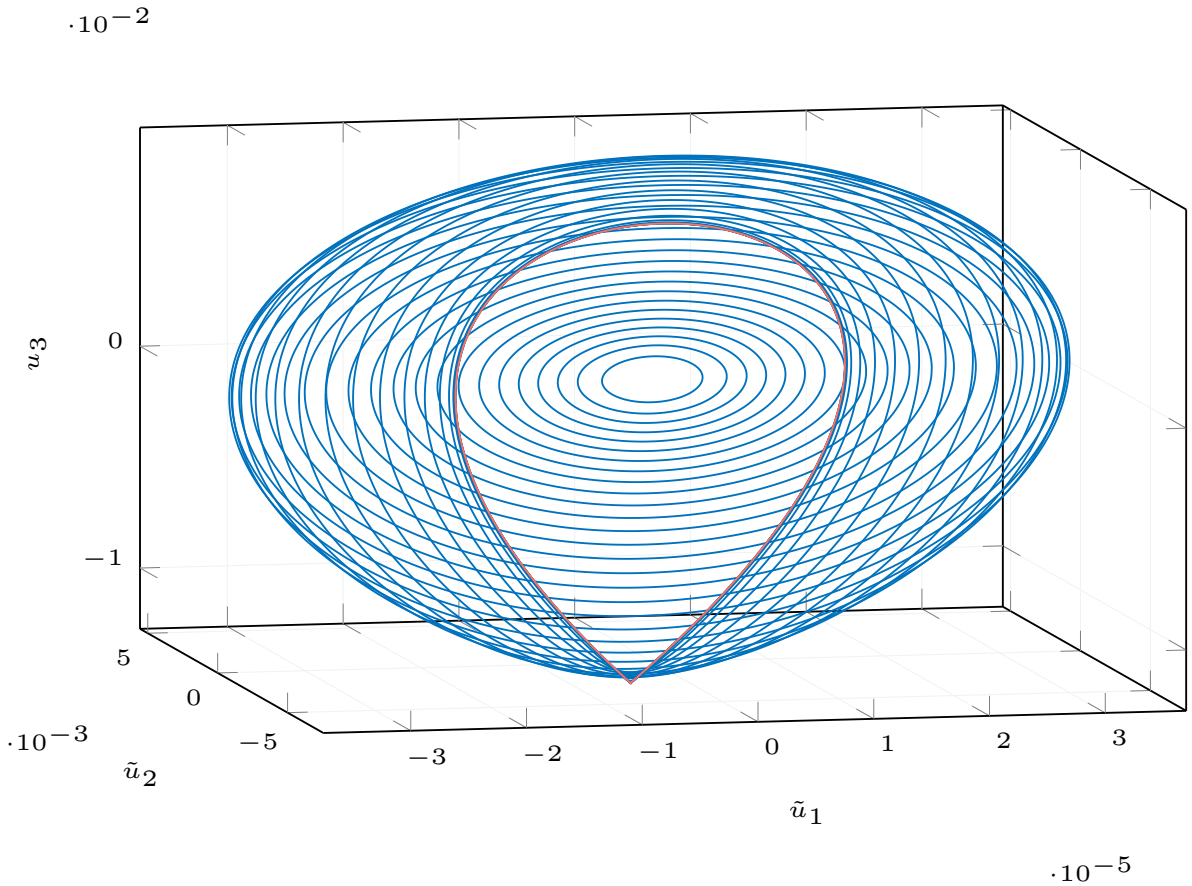

Figure S29: Branch of periodic solutions emanating from the last point on the first Hopf branch emanating from the transcritical Bogdanov–Takens point in (S11). The last point on the periodic solution branch is colored red, at which the periodic orbits have converged to a homoclinic orbit.

S4.16 Homoclinic branch connecting the two Bogdanov–Takens points

Here we will show that the transcritical and generic Bogdanov–Takens points are connected, not only by a Hopf curve, but also through a homoclinic curve. For this, we first continue the second Hopf branch emanating from the transcritical Bogdanov–Takens point again, but with a smaller step size.

```
%% Set bifurcatiun parameter range and step size bounds
brpars={'max_bound', [ind.alpha1 0.5],...
        'min_bound', [ind.alpha1 -0.35],...
        'max_step',  [ind.alpha1 1e-5; ind.alpha2 1e-5]};

%% Continue Hopf curve emanating from Bogdanov-Takens point again
% but with smaller stepsize
[~,hbr_small_stepsize,suc]=C1branch_from_C2point(funcs,bt,free_pars,...
    'codim2','BT','codim1','hopf',brpars{:},'step',1e-04, ...
    'plot',0,'generic_unfolding',false); assert(all(suc(:)>0))
nop=285; [hbr_small_stepsize(2),suc]=br_contn(funcs,hbr_small_stepsize(2),nop);
```

```
assert(suc>0)
hbr_small_stepsize_pm = [getpars(hbr_small_stepsize(2),ind.alpha1);
    getpars(hbr_small_stepsize(2),ind.alpha2)];
```

Next, we continue from (almost) each point on the new Hopf branch the emerging periodic solutions in the parameter α_2.

```
%% Continue periodic orbits from every Hopf point on the second Hopf branch
% (takes a few minutes)
start_ind = 19;
end_ind = length(hbr_small_stepsize_pm)-5;
for i=start_ind:end_ind
    psol_br(i-start_ind+1)=SetupPsol(funcs,hbr_small_stepsize(2),i,'contpar',...
        ind.alpha2, ...
        'degree',3,'intervals',50,brpars{1:4},'max_step',[0,inf],'plot',0, ...
        'radius', 0.001);
    psol_br(i-start_ind+1)=br_contn(funcs,psol_br(i-start_ind+1),50);
end
```

By plotting the last point on each of the periodic solutions branches in $(\alpha_1, \tilde{u}_1, \tilde{u}_2)$-space, together with the homoclinic solutions on the first and third homoclinic branches continued above, it is indeed clear that the two homoclinic branches are connected through a single homoclinic curve, see Figure S30. We also see how the transition is made from the homoclinic orbits with a fixed saddle point at the origin to the homoclinic orbits with a moving saddle. Clearly, `DDE-BifTool` has difficulties continuing the homoclinic orbits near the global homoclinic bifurcation point.

```
%% Plot homoclinic solutions in (\alpha_2,\tilde u_1, \tilde u_2) space
figure(12); clf; hold on
for i=1:end_ind-start_ind+1
    profile = psol_br(i).point(end).profile;
    profileRotated = R(0.29)*profile([1,2],:);
    plot3(psol_br(i).point(end).parameter(ind.alpha1)*ones(size(profile(1,:))), ...
        profileRotated(1,:),profileRotated(2,:),'Color',cm(2,:))
end
for i=1:length(hcli_br(1).point)
    profile = hcli_br(1).point(i).profile;
    profileRotated = R(0.29)*profile([1,2],:);
    plot3(hcli_br(1).point(i).parameter(ind.alpha1)*ones(size(profile(1,:))), ...
        profileRotated(1,:),profileRotated(2,:), 'Color',cm(1,:))
end
for i=1:length(hcli_br(3).point)
    profile = hcli_br(3).point(i).profile;
    profileRotated = R(0.29)*profile([1,2],:);
    plot3(hcli_br(3).point(i).parameter(ind.alpha1)*ones(size(profile(1,:))), ...
        profileRotated(1,:),profileRotated(2,:), 'Color',cm(1,:))
end
grid on
view(-12,12)
xlabel('$\alpha_1$','Interpreter','LaTeX')
```

```
ylabel('$\tilde u_1$','Interpreter','LaTeX')
zlabel('$\tilde u_2$','Interpreter','LaTeX')
plot3(bt.parameter(ind.alpha1),bt.x(1),bt.x(2),'.k', ...
    'MarkerSize', 18, 'DisplayName','transcritical Bogdanov-Takens point');
bt2rotated = R(0.29)*bt2.x([1,2])
plot3(bt2.parameter(ind.alpha1),bt2rotated(1),bt2rotated(2),'.k', ...
    'MarkerSize', 18, 'DisplayName','Bogdanov-Takens point');
```

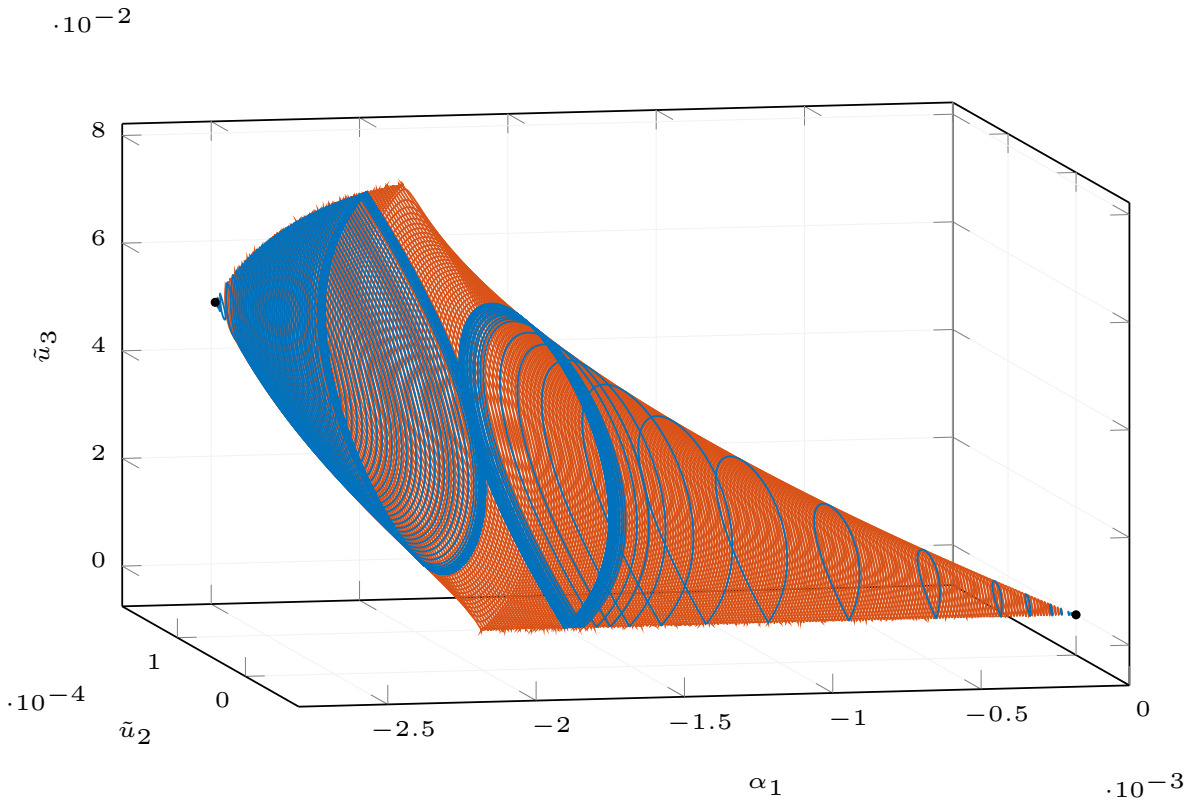

Figure S30: Branch of homoclinic solutions (orange) connecting the transcritical Bogdanov–Takens point (the right black dot) with the generic Bogdanov–Takens point (the left black dot) in (S11). The blue homoclinic curve emerging from the transcritical and generic Bogdanov–Takens points are the previously continued homoclinic branches.

To obtain an impression of the parameter curves connecting the transcritical and generic Bogdanov–Takens points, we rotated the curves through an angle of $\theta_2 = 0.646045233034992$. By doing so, the two Bogdanov–Takens points are aligned on the abscissa. In Figure S31, we plotted the first and third homoclinic branch, emanating from the transcritical Bogdanov–Takens and generic Bogdanov–Takens points respectively, the Hopf curve connecting the two Bogdanov–Takens points, and the newly obtained homoclinic bifurcation curve.

```
%% Plot rotated bifurcation homoclinic bifurcation curves
figure(13);clf;hold on
```

```
title(['Codimension 1 curves emanating from the generic ', ...
       'and transcritical Bogdanov-Takens']);
theta2 = abs(atan(bt2.parameter(2)/bt2.parameter(1)))
hcli_br1_rotated_pm = R(theta2)*hcli_br1_pm;
hcli_br3_rotated_pm = R(theta2)*hcli_br3_pm;
hcli_bifurcation_br_rotated_pm = R(theta2)*hcli_bifurcation_br_pm';
hbr2_rotated_pm = R(theta2)*hbr2_pm(:,1:45);
bt_rotated_pm = R(theta2)*bt.parameter([ind.alpha1, ind.alpha2])';
bt2_rotated_pm = R(theta2)*bt2.parameter([ind.alpha1, ind.alpha2])';
plot(hcli_br1_rotated_pm(1,:),hcli_br1_rotated_pm(2,:),'Color',cm(1,:), ...
    'DisplayName','Homoclinic branches')
plot(hcli_br3_rotated_pm(1,:),hcli_br3_rotated_pm(2,:),'Color',cm(1,:), ...
    'HandleVisibility','off')
plot(hcli_bifurcation_br_rotated_pm(1,:),hcli_bifurcation_br_rotated_pm(2,:), ...
    ':','Color',cm(4,:), ...
    'DisplayName','Homoclinic bifurcation branch')
plot(hbr2_rotated_pm(1,:),hbr2_rotated_pm(2,:),'Color',cm(2,:), ...
    'DisplayName','Hopf branch')
plot(bt_rotated_pm(1),bt_rotated_pm(2),'ok', ...
    'DisplayName','Transcritical Bogdanov-Takens point', ...
    'MarkerFaceColor',[0 0 0], 'MarkerSize', 09);
plot(bt2_rotated_pm(1),bt2_rotated_pm(2),'dk', ...
    'DisplayName','generic Bogdanov-Takens point', ...
    'MarkerFaceColor',[0 0 0], 'MarkerSize', 10);
legend
xlabel('$\tilde \alpha_1$','Interpreter','LaTeX');
ylabel('$\tilde \alpha_2$','Interpreter','LaTeX');
```

S4.17 Convergence plot

Using the function from Section S1.12, we create a log-log convergence plot comparing the convergence order of the first and third order homoclinic asymptotics from Section 4.2.3. The code below yields Figure S32.

```
%% Convergence plot
amplitudes = logspace(-3, -1.4, 20);
orders = [1:3];
relativeErrors = convergence_plot(funcs, bt, orders, amplitudes, 'free_pars',
↪  free_pars,...

↪  'orders',orders,'TTolerance',1e-5,'ntst',82,'generic_unfolding',0,'debug',false);
figure(14); clf; hold on
title('Convergence plot comparing first and third order homoclinic asymptotics')
plot(log10(amplitudes), log10(relativeErrors{1}(:)), '*-')
plot(log10(amplitudes), log10(relativeErrors{3}(:)), '*-')
legend({'first order', 'thrid order'})
xlabel('amplitude')
ylabel('relative error')
```

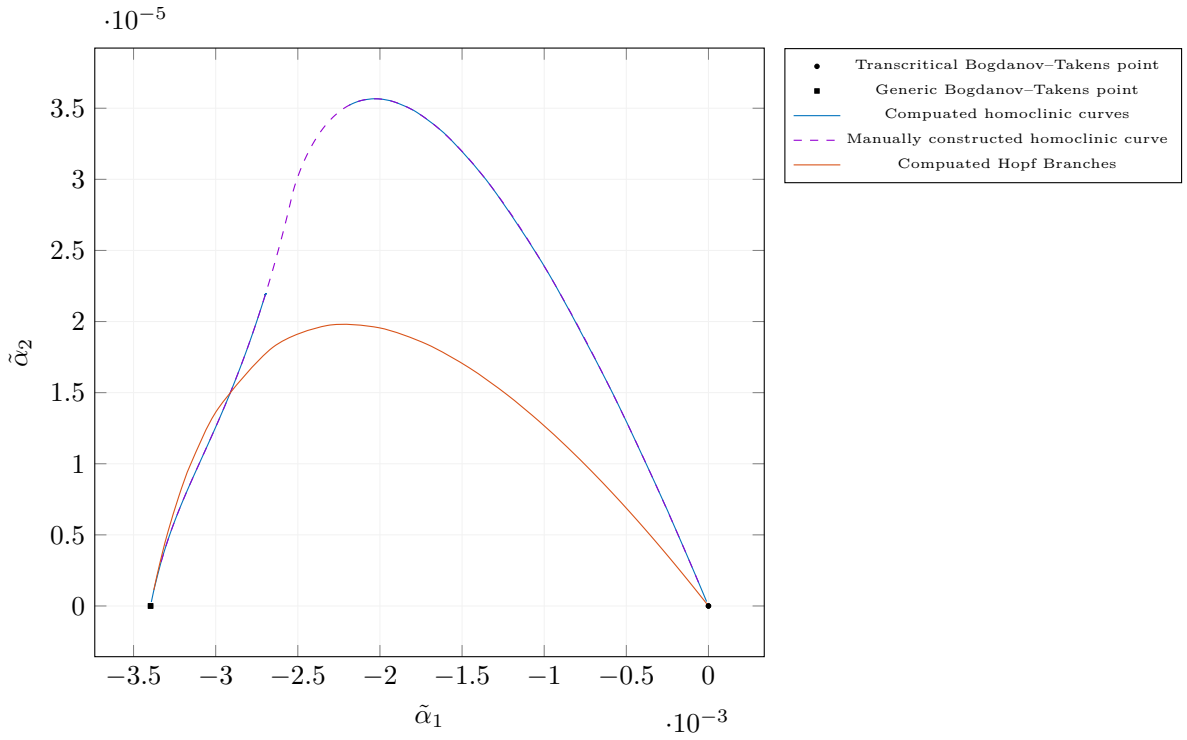

Figure S31: Bifurcation diagram near the transcritical and generic Bogdanov–Takens bifurcation points in (S11) showing the fully continued homoclinic branch connected to the non-trivial equilibrium emanating from the transcritical Bogdanov–Takens point.

S4.18 Simulation with `DifferentialEquations.jl`

Here we will perform four simulations. The first two simulations will be at two homoclinic orbits located on the two homoclinic curves emanating from the transcritical Bogdanov–Takens point continued with `DDE-BifTool`, see Figure S33. The second two simulations will be in the regions where there should be stable periodic orbits, see Figure S34.

S4.18.1 Loading necessary Julia packages

We start by loading the necessary Julia packages. Compared with the previous demonstrations, we also need to load the Julia package `Symbolics.jl` [21] to differentiate the activation functions f_1, f_2, f_3.

```
using DifferentialEquations
using GLMakie
using DelimitedFiles
using DDEBifTool
using NonlinearEigenproblems
using Colors
using IntervalArithmetic, IntervalRootFinding
using LinearAlgebra
using Symbolics
```

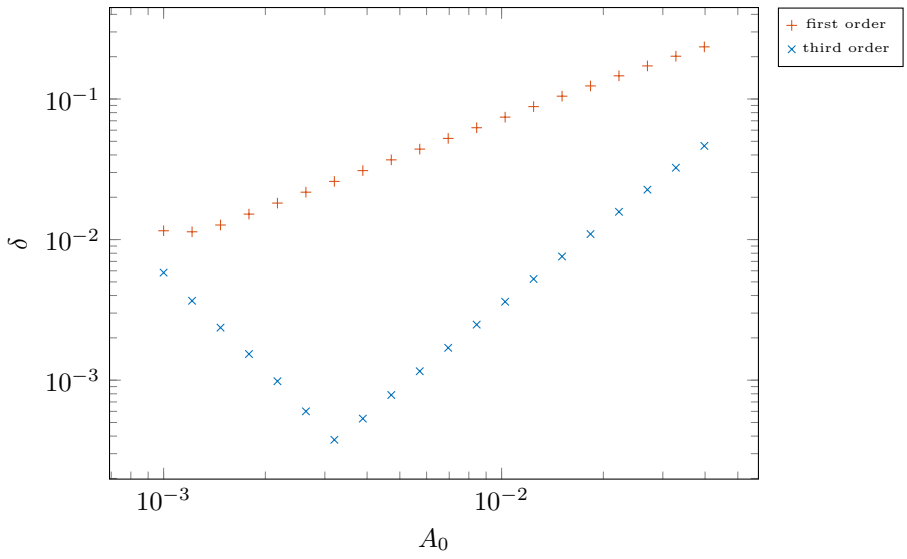

Figure S32: On the abscissa is the approximation to the amplitude A_0 and on the ordinate the relative error δ between the constructed solution `hcli_pred` to the defining system for the homoclinic orbit and the Newton corrected solution `hcli_corrected`.

S4.18.2 Define system

Next we define the system to be integrated, a system to approximate the reverse flow, and also an allocating version used for stability calculations. Note that we need the Julia `Symbolics.jl` to differentiate the activation functions.

```
# define constants
const τ = 5.0
const μ1 = 0.1
const μ2 = 0.3
const μ3 = 0.2
const c12 = 1.0
const c13 = 1.0

# define functions used in the model
@inline f1(x) = tanh(x) + 0.1*x^2
@inline f2(x) = tanh(x)
f3 = f2

# define derivatives of functions used in the model
@variables x
df1 = eval(build_function(Symbolics.derivative(f1(x), x),x))
df2 = eval(build_function(Symbolics.derivative(f2(x), x),x))
df3 = df2

const c21^0 = μ2^2*(μ1*(μ3*τ + 1.0) + μ3)/(c12*(μ2 - μ3)*df1(0)*df2(0))
const c31^0 = μ3^2*(μ1*(μ2*τ + 1.0) + μ2)/(c13*(μ3 - μ2)*df1(0)*df3(0))

```

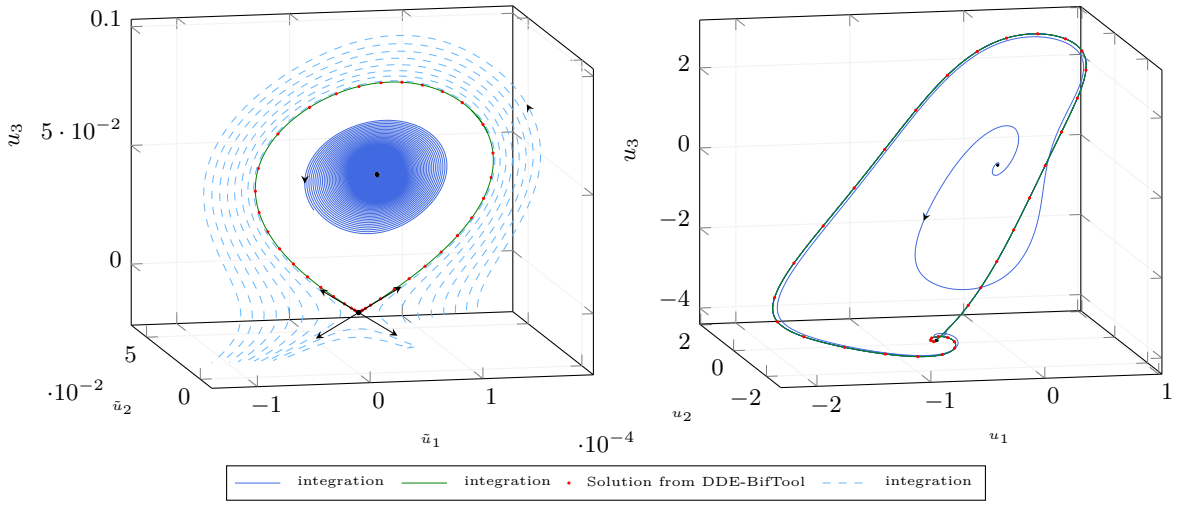

Figure S33: Comparing the computed homoclinic orbits in (S11) with `DDE-BifTool` with the solutions obtained from numerical simulation with Julia. We see the numerical integrated solution going through all the red points from the solution from `DDE-BifTool`.

```
# define model
function triNeuralBAMNetworkModel!(du,u,h,p,t)
    α₁, α₂ = p
    u₁, u₂, u₃ = u
    u₁t, u₂t, u₃t = h(p, t-τ)
    du[1] = -μ₁*u₁ + (c₂₁⁰ + α₁)*f₁(u₂t) + (c₃₁⁰ + α₂)*f₁(u₃t);
    du[2] = -μ₂*u₂ + c₁₂*f₂(u₁);
    du[3] = -μ₃*u₃ + c₁₃*f₃(u₁);
end

# allocating version (used for calculating stability)
function triNeuralBAMNetworkModel(u, p)
    du = similar(u[:,1])
    h(p,t) = u[:,2]
    triNeuralBAMNetworkModel!(du,u[:,1],h,p,0)
    du
end

# version used for calculating stability at the end of the script
function triNeuralBAMNetworkModel(u, p, τ)
    α₁, α₂ = p
    u₁, u₂, u₃ = u[:,1]
    u₁t, u₂t, u₃t = u[:,2]
    c₂₁⁰ = μ₂^2*(μ₁*(μ₃*τ + 1.0) + μ₃)/(c₁₂*(μ₂ - μ₃)*df₁(0)*df₂(0))
    c₃₁⁰ = μ₃^2*(μ₁*(μ₂*τ + 1.0) + μ₂)/(c₁₃*(μ₃ - μ₂)*df₁(0)*df₃(0))
    du = similar(u[:,1])
    du[1] = -μ₁*u₁ + (c₂₁⁰ + α₁)*f₁(u₂t) + (c₃₁⁰ + α₂)*f₁(u₃t);
    du[2] = -μ₂*u₂ + c₁₂*f₂(u₁);
    du[3] = -μ₃*u₃ + c₁₃*f₃(u₁);
```

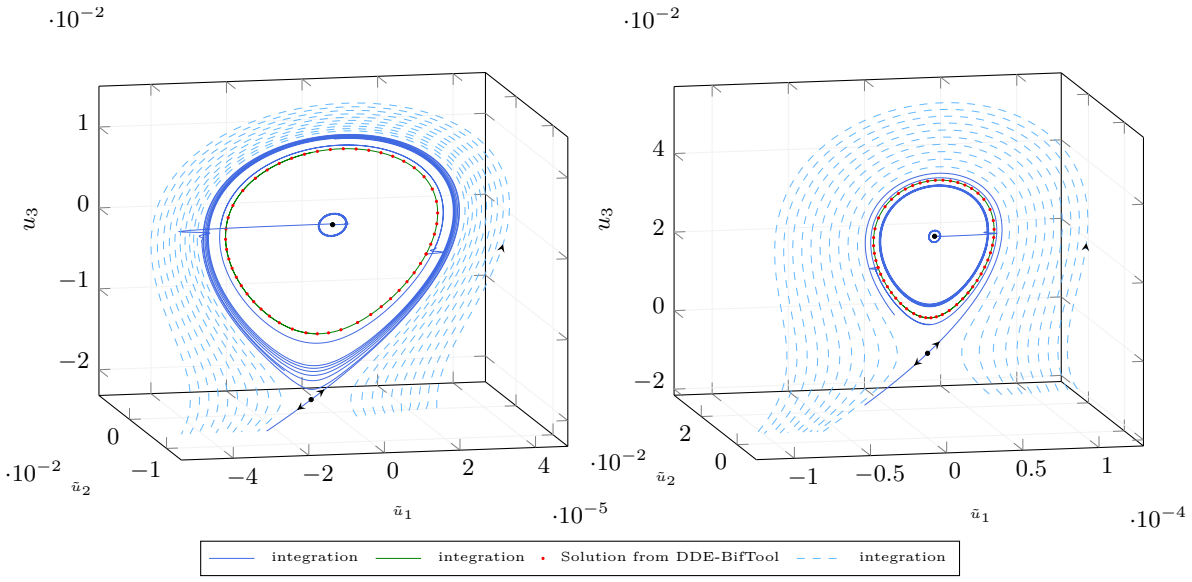

Figure S34: Comparing the computed periodic orbits in (S11) with `DDE-BifTool` with the solutions obtained from numerical simulation with Julia. We see the numerical integrated solution going through all the red points from the solution from `DDE-BifTool`.

```
    du
end
```

S4.18.3 Function for creating streamlines plot

To obtain an impression of the flow near transcritical Bogdanov–Takens point, we create a streamlines function. This is particularly useful for seeing the flow around the stable manifold of the saddle-note.

```
# stream lines
function stream_lines(Tend, abscissa, ordinate, z, p, alg)
    tspan =  (0.0, Tend);
    sols = []
    for x ∈ abscissa
        for y ∈ ordinate
            h(p,t) = [x;y;z]
            sol =  solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
            push!(sols,sol)
        end
    end
    sols
end
```

S4.18.4 Create figure with several axes

We create a figure containing multiple axis in which we will plot the homoclinic, periodic orbits, and the left and right-hand sides of (S17).

```
# create figure
fig = Figure()
set_theme!(theme_light())
labels = (xlabel=L"\tilde{u}_1", ylabel=L"\tilde{u}_2", zlabel=L"u_2")
ax1 = Axis3(fig[1,1], title="Homoclinic orbit I";  labels...)
ax2 = Axis3(fig[1,2], title="Homoclinic orbit II"; labels...)
ax3 = Axis3(fig[2,1], title="Periodic orbit I";    labels...)
ax4 = Axis3(fig[2,2], title="Periodic orbit II";   labels...)
ax5 = GLMakie.Axis(fig[3,1], title="Stablitiy determining functions",
  ↪  xlabel=L"\tau")
ax6 = GLMakie.Axis(fig[3,2], title="Eigenvalues",
                   xlabel=L"\Re(\lambda)", ylabel=L"\Im(\lambda)")
```

S4.18.5 Define parameters, equilibria

We define parameters located on the continued homoclinic branch with `DDE-BifTool`. Then calculate the equilibria points in (S10) near the transcritical Bogdanov–Takens point.

```
## Simulation homoclinic orbit I
α₁, α₂ =  -0.001724521613831, 0.001344362436730
p = [α₁; α₂]

# derive equilbria
rts = roots( u₁ -> -μ₁*u₁ + (c₂₁⁰ + α₁)*f₁(c₁₂*f₂(u₁)/μ₂) +
            (c₃₁⁰ + α₂)*f₁(c₁₃*f₃(u₁)/μ₃), interval(-0.01,0.01))
u₁ = sort(mid.(interval.(rts)))
```

S4.18.6 Plot equilibria and homoclinic orbit

By plotting the homoclinic orbit obtained with `DDE-BifTool` located at parameter values

$$(\alpha_1, \alpha_2) = (-0.001724521613831, 0.001344362436730),$$

we can compare with the numerical simulations. As in the analysis with `DDE-BifTool`, we rotate the coordinates of the solutions to visualize the solutions in phase-space.

```
# rotation matrix
R(α) = [cos(α) -sin(α) 0; sin(α) cos(α) 0; 0 0 1]

# plot equilibria
scatter!(ax1,Point3f.([R(0.29)*p.coords for p ∈ point]), color=:black)

# plot homoclinic orbit from DDE-BifTool
csvdir = "/home/maikel/Documents/MyPapers/BTPaper"*
         "/data/triNeuronBAMNeuralNetworkModel/"
homoclinicorbitI = readdlm(csvdir * "homoclinicorbitI_45.csv")
scatter!(ax1, R(0.29)*homoclinicorbitI[1:3:end,:]', color=:red)
```

S4.18.7 Leading eigenvectors

Next, we calculate and plot the leading eigenvectors of the characteristic matrix at the saddle-node bifurcation point.

```
# calculate stability
τs = [0.0 τ]
point = [(coords = [u₁, c₁₂*f₂(u₁)/μ₂, c₁₃*f₃(u₁)/μ₃], pars = p) for u₁ ∈ u₁]
M = [coefficient_matrices(triNeuralBAMNetworkModel, point[i], τs)
     for i ∈ 1:length(u₁)]
dep = [DEP([M[i][j](point[i].coords,point[i].pars) for j in 1:length(τs)],
           vec(τs)) for i ∈ 1:length(u₁)]
λ = iar_chebyshev.(dep,maxit=100,neigs=2)
saddle_indx = findall(λ -> abs(λ[1]) < 1e-14,
                      imag([λ[i][1] for i ∈ 1:length(u₁)]))
saddle = point[saddle_indx][]
V = real.(λ[saddle_indx][][2])
indx_unstable = findall(λ -> real(λ) > 0, λ[saddle_indx][][1])
indx_stable = findall(λ -> real(λ) < 0, λ[saddle_indx][][1])
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])

# draw leading eigenvectors of the characteristic matrix at the saddle-node
psHomIEigenvectors = repeat([Point3f(saddle.coords)],4)
nsHomIEigenvectors = [op.(0,1e-02Vec3f(R(0.29)*V)) for op in [+,-]
                      for V in [[Vunstable]; [Vstable]]]
arrows!(ax1, psHomIEigenvectors, nsHomIEigenvectors, linewidth = 0.000003,
    arrowsize = Vec3f(1.0e-5,1.0e-3,1.0e-3),
)
```

S4.18.8 Define callback

Since we are only interested in the flow near the equilibria points, we create a continuous callback to ensure the orbits do not become too large.

```
# create continous callback
condition(u,t,integrator) = u[3] + 0.015
affect!(integrator) = terminate!(integrator)
cb = ContinuousCallback(condition,affect!)
```

S4.18.9 Integrate the system at homoclinic orbits I

Now we define the problem to be integrated and choose the algorithm to be used. Then we integrate the system for a range of initial history functions using the function `streamlines`. Next, we integrate the system near the inner equilibrium, i.e., the equilibrium inside the homoclinic orbit. This equilibria should be an unstable spiral. We show the first and last part of the obtained solutions. We see that the last part of the integrated solutions completely overlap the homoclinic solution obtained with `DDE-BifTool`.

```
# define DDEProblem and choose integration algorithm
prob(h,p,tspan) = DDEProblem(triNeuralBAMNetworkModel!, h(p,0),h,tspan,p;
                             constant_lags=[τ], callback=cb)
alg = MethodOfSteps(Tsit5())

# integrate over a range of history functions
homSolsI = stream_lines(3000.0, range(-0.00205,-0.0016,10),
                        -0.010183777508431096, -0.014999999999999973, p, alg);
for sol ∈ homSolsI
    Tend = sol.t[end]
    lines!(ax1,R(0.29)*sol(range(Tend-94*Tend/100,Tend,100)),
           color=:steelblue1, linestyle=:dash)
end

# integrate near inner equilbria
h(p,t) = [u₁[2], c₁₂*f₂(u₁[2])/μ₂, c₁₃*f₃(u₁[2])/μ₃] + [0.0; 0.0; 0.00001]
tspan =  (0.0, 200000.0);
homI_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-06,abstol=1e-09)

# plot first part of the orbit, showing the solution spirals out
lines!(ax1,R(0.29)*homI_sol1(range(0,100000,10000)), color=:royalblue1)

# plot last part to show that it overlaps the homoclinic solution from DDE-BifTool
lines!(ax1,R(0.29)*homI_sol1(range(197400,200000,1000)),color=:green)
```

S4.18.10 Add arrows on solutions

Lastly, we add arrows to the obtained solutions and redraw the equilibria.

```
# add direction arrows
pointnumber = [180, 99900]
psHomI = Point3f.([R(0.29)*homSolsI[10][pointnumber[1]],
                   R(0.29)*homI_sol1(pointnumber[2])])
nsHomI = Vec3f.([R(0.29)*homSolsI[10][pointnumber[1]+1],
                 R(0.29)*homI_sol1(pointnumber[2]+1)] - psHomI)
arrows!(ax1, psHomI,nsHomI, linewidth = 0.000003,
        arrowsize = Vec3f(1.0e-5,1.0e-3,1.0e-3), lengthscale = 1e-02)

# redraw equilibria
scatter!(ax1,Point3f.([R(0.29)*[u₁, c₁₂*f₂(u₁)/μ₂, c₁₃*f₃(u₁)/μ₃] for u₁ ∈ u₁]),
         color=:black)
```

We should now obtain an interactive figure similar to the left figure in Figure S33.

S4.18.11 Simulation near stable periodic orbit I

The code for numerical simulation near the second homoclinic orbit, see the right plot in Figure S33, is not included here.

To show by integration the existence of a stable periodic orbit, we first located a periodic orbit in DDE-BifTool. This can be done by continuing a branch of periodic orbits emanating from a point on the continued Hopf curves. Then we load the profiles of the periodic orbits into Julia. We perform two simulations. For the first we integrate with a constant history function equal to a point inside the periodic orbit. The second simulation starts from the unstable eigenvector of the characteristic matrix calculated above.

```
## Simulation periodic orbit I
α₁, α₂ = -4.211633510571706e-05, -8.548339944470298e-05
p = [α₁; α₂]

# derive equilbria
rts = roots( u₁ -> -μ₁*u₁ + (c₂₁⁰ + α₁)*f₁(c₁₂*f₂(u₁)/μ₂) +
             (c₃₁⁰ + α₂)*f₁(c₁₃*f₃(u₁)/μ₃), interval(-0.01,0.0))
u₁ = sort(mid.(interval.(rts)))

# calculate stability
τs = [0.0 τ]
point = [(coords = [u₁, c₁₂*f₂(u₁)/μ₂, c₁₃*f₃(u₁)/μ₃], pars = p) for u₁ ∈ u₁]
M = [coefficient_matrices(triNeuralBAMNetworkModel, point[i], τs)
     for i ∈ 1:length(u₁)]
dep = [DEP([M[i][j](point[i].coords,point[i].pars)
            for j in 1:length(τs)],vec(τs)) for i ∈ 1:length(u₁)]
λ = iar_chebyshev.(dep,maxit=100,neigs=2)
λ[1][1]
λ[2][1]
saddle_indx = findall(λ -> abs(λ[1]) < 1e-14,
                      imag([λ[i][1] for i ∈ 1:length(u₁)]))
saddle = point[saddle_indx][]
V = real.(λ[saddle_indx][][2])
indx_unstable = findall(λ -> real(λ) > 0, λ[saddle_indx][][1])
indx_stable = findall(λ -> real(λ) < 0, λ[saddle_indx][][1])
Vunstable = vec(V[:,indx_unstable])
Vstable = vec(V[:,indx_stable])

# plot equilibria
scatter!(ax3,Point3f.([R(0.29)*point[i].coords for i ∈ 1:length(u₁)]),color=:black)

# plot homoclinic orbit from DDE-BifTool
psolI = readdlm(csvdir * "psolI_28.csv")
scatter!(ax3, R(0.29)*psolI[1:3:end,:]',color=:orangered)

# create continous callback
condition(u,t,integrator) = u[3] + 0.02
affect!(integrator) = terminate!(integrator)
cb = ContinuousCallback(condition,affect!)

# integrate from periodic orbit I
h(p,t) = psolI[1,:]
tspan =  (0.0, 800000.0);
```

```
periodicI_sol1 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax3,R(0.29)*periodicI_sol1(range(0.0,2000,1000)),color=:royalblue1)
lines!(ax3,R(0.29)*periodicI_sol1(range(tspan[2]-2000,tspan[2],1000)),color=:green)

# integrate from near inner equilbrium
h(p,t) =[0.0; 0.0; -0.00001]
tspan =  (0.0, 400000.0);
periodicI_sol2 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax3,R(0.29)*periodicI_sol2(range(0.0,10000,10000)),color=:royalblue1)

# integrate from near inner equilbrium
h(p,t) =[0.0; 0.0; 0.0001]
tspan =  (0.0, 300000.0);
periodicI_sol3 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax3,R(0.29)*periodicI_sol3(range(0.0,10000,10000)),color=:royalblue1)

# integrate from unstable eigendirection
ϵ = sign(Vunstable[1])*1e-04
h(p,t) = saddle.coords - ϵ*Vunstable
periodicI_sol4 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax3,R(0.29)*periodicI_sol4,color=:royalblue1)

# integrate from unstable eigendirection
tspan =  (0.0, 40000.0);
ϵ = sign(Vunstable[1])*1e-04
h(p,t) = saddle.coords + ϵ*Vunstable
periodicI_sol5 = solve(prob(h,p,tspan),alg,reltol=1e-09,abstol=1e-09)
lines!(ax3,R(0.29)*periodicI_sol5(range(0.0,2000,10000)),color=:royalblue1)

# integrate over a range of history functions
psolSolsI = stream_lines(30000.0, range(-0.00373,-0.0036,10),
                      -0.013391304482590667,
                      -0.019999999999999997 , p, alg);
for sol ∈ psolSolsI
    Tend = sol.t[end]
    lines!(ax3,R(0.29)*sol(range(Tend-98*Tend/100,Tend,100)),color=:steelblue1,
           linestyle=:dash)
end

# add direction arrows
pointnumber = [100, 100, 120]
psPeriodicI = Point3f.([R(0.29)*periodicI_sol4[pointnumber[1]],
                        R(0.29)*periodicI_sol5[pointnumber[2]],
                        R(0.29)*psolSolsI[10][pointnumber[3]]])
nsPeriodicI = Vec3f.([R(0.29)*periodicI_sol4[pointnumber[1]+1],
                      R(0.29)*periodicI_sol5[pointnumber[2]+1],
                      R(0.29)*psolSolsI[10][pointnumber[3]+1]] - psPeriodicI)
arrows!(ax3, psPeriodicI, nsPeriodicI, linewidth = 0.0,
        arrowsize = 0.5*Vec3f(5.0e-6,5.0e-4,5.0e-4))
```

After running the above code, we should obtain a similar plot as in Figure S34.

S4.18.12 Stability of the center manifold

To confirm Lemma 17 numerically, we consider again (S20) and (S21). The code below plots the left and right-hand sides of (S17). We see in Figure S35 that there are two points of intersection.

```
## Stability of the center manifold
ω₀(τ) =  0.7071067811865476*sqrt(-0.14 + sqrt(0.0484 + 0.00528*τ +
                                                0.00014400000000000006*τ^2))
a₀ = -μ₁*μ₂*μ₃
b₀(τ) =  -ω₀(τ)*(μ₂*μ₃ + μ₁*(μ₂ + μ₃ + μ₂*μ₃*τ))
ζ₀(τ) = μ₁^4 + (μ₂^2 + μ₃^2)^2 + 8*μ₁*μ₂*μ₃*(μ₂ + μ₃ + μ₂*μ₃*τ) +
           2*μ₁^2*(μ₃^2 + 4*μ₂*μ₃*(1+μ₃*τ) + μ₂^2*(1 + 2*μ₃*τ*(2 + μ₃*τ)))
ζ₁(τ) = μ₁*μ₂*μ₃ - (μ₁ + μ₂ + μ₃)*ω₀(τ)^2
ζ₂(τ) = μ₂*μ₃*ω₀(τ) + μ₁*(μ₂ + μ₃)*ω₀(τ) - ω₀(τ)^3

τs = range(0.0,20.0,1000)
lhs(τ) = tan(τ*ω₀(τ))
rhs(τ) =  (b₀(τ)*ζ₁(τ) - a₀*ζ₂(τ))/(a₀*ζ₁(τ) + b₀(τ)*ζ₂(τ))
empty!(ax5)
scatter!(ax5, τs, lhs.(τs) ,markersize=2, label=L"\tan(\tau \omega_0(\tau))"),
scatter!(ax5, τs, rhs.(τs),markersize=2,
         label=L"\frac{b_0\zeta_1 - a_0\zeta_2}{a_0\zeta_1 + b_0\zeta_2}")
ylims!(ax5,-40,40)
axislegend(ax5, position = :lt)
```

Using the function `roots` from the package `IntervalRootFinding.jl` we search for points of intersection.

```
# locate points of intersection
rts = roots( τ -> lhs(τ) - rhs(τ), interval(0.0,20.0))
rts_unique = rts[isunique.(rts)]
τ₀ = sort(mid.(interval.(rts_unique)))
```

In the Julia output we obtain

```
7-element Vector{Root{Interval{Float64}}}:
 Root([13.2309, 13.231], :unique)
 Root([19.2374, 19.2375], :unique)
 Root([19.7337, 19.7338], :unknown)
 Root([6.00301, 6.00302], :unique)
 Root([8.33333, 8.33334], :unknown)
 Root([13.5353, 13.5354], :unknown)
 Root([5.75402, 5.75403], :unknown)
 Root([8.33333, 8.33334], :unknown)
```

Note that, since there are multiple discontinuities, there are many unknown solutions returned. We extract the unique solutions from the list of solutions and test if these provide solutions to the characteristic equation.

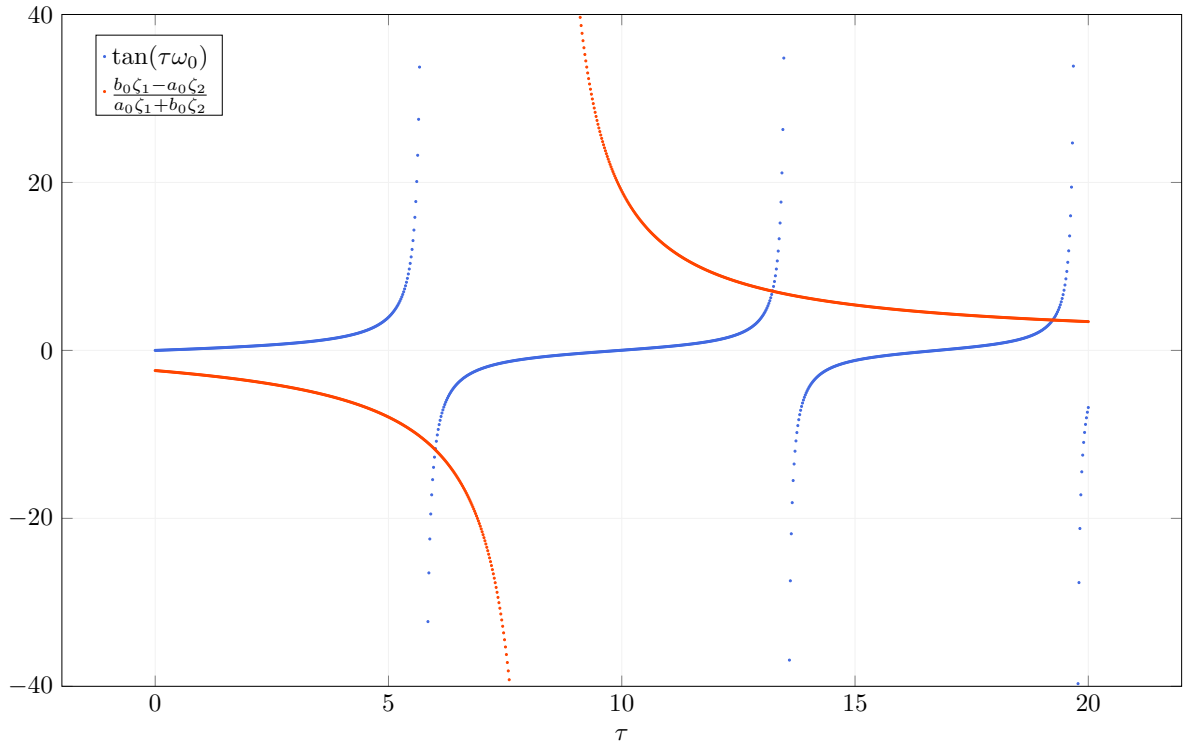

Figure S35: Plot of the left and right-hand sides of (S17). Points of intersection are candidates for the center manifold to lose its stability.

```
# characteristic equation
bt = (coords = zeros(3), pars = zeros(2))
Δ(λ,τ) = -λ*I + M[1](bt.coords,bt.pars) + M[2](bt.coords,bt.pars)*exp(-τ*λ)
for τ ∈ τ0
    M = coefficient_matrices((u,p) -> triNeuralBAMNetworkModel(u, p, τ),
                             bt, zeros(2))
    if ≈(det(Δ(ω0(τ)im,τ)), 0.0 + 0.0im, atol=1e-12)
        println("The centermanifold is attractive for 0 < τ < $τ.")
        τ0 = τ
        break
    end
end
```

In the Julia output we obtain

```
The centermanifold is locally attractive for 0 < τ < 13.230934887939895.
```

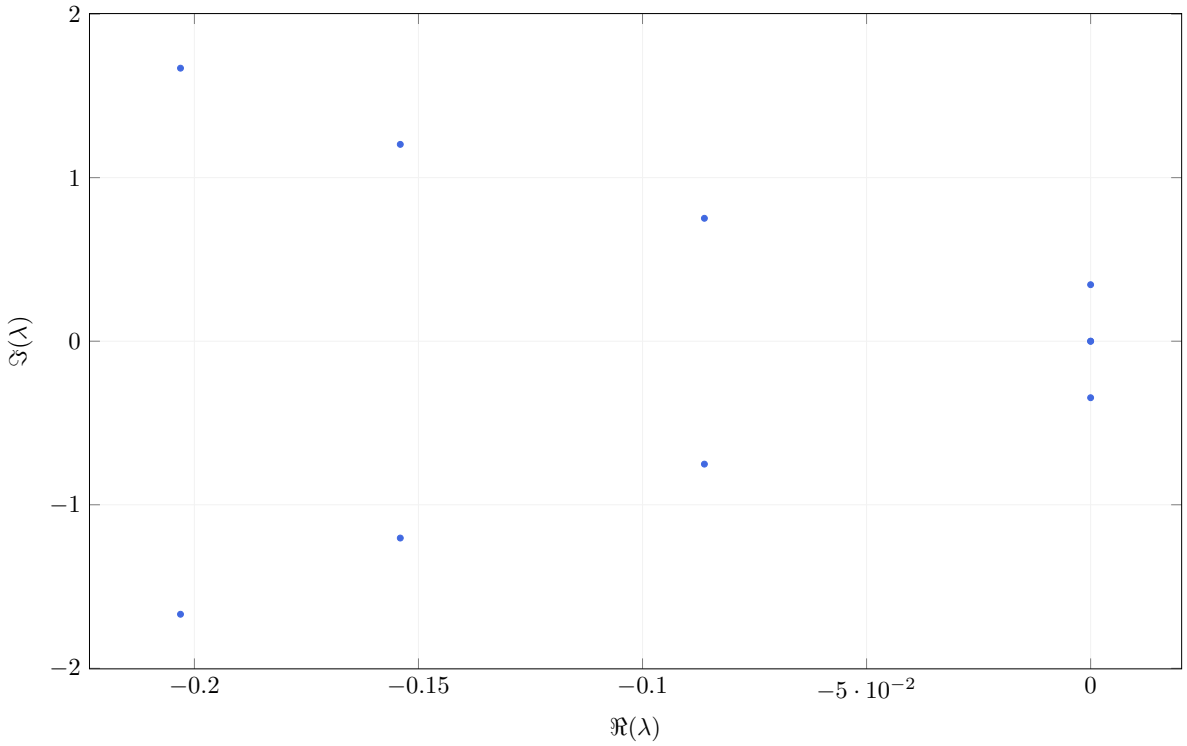

Figure S36: Plot of the leading eigenvalues at the analytically derived transcritical Bogdanov–Takens point with $\tau = 13.230934887939895$ and $\omega = \omega(\tau)$ from (S19). At this point there are four eigenvalues on the imaginary axis. For $\tau > 13.230934887939895$ the center manifold is locally unstable.

S4.18.13 Calculate and plot stability at Bogdanov–Takens point

We finish this demonstration by confirming the stability of the transcritical Bogdanov–Takens point at $\tau = 13.230934887939895$ obtained in the previous section. In Figure S36 we have plotted the eigenvalues. We indeed see that at $\tau = 13.230934887939895$ the center manifold loses stability. Lastly, we also verified in the code below that the eigenvalues with positive imaginary part on the imaginary axis is approximately equal to the expression for ω obtained in (S19).

```
# calculate and plot leading eiganvalues at Bogdanov--Takens point with τ=τ₀
τs = [0.0, τ₀]
M = coefficient_matrices((u,p) -> triNeuralBAMNetworkModel(u, p, τ₀), bt, τs)
dep = DEP([M[j](bt.coords,bt.pars) for j in 1:length(τs)],τs)
λ = iar_chebyshev(dep,maxit=200,neigs=10,σ=-.1)
λ[1]
hopf_indx = findall(λ -> abs(real(λ)) < 1e-10 && imag(λ) > 0.1, λ[1])
imag(λ[1][hopf_indx])[1] ≈ ω₀(τ₀)
scatter!(ax6, real(λ[1]), imag(λ[1]))
```

References for main text and supplement

[1] L. Benet and D. P. Sanders, *IntervalRootFinding.jl*. https://github.com/JuliaIntervals/IntervalRootFinding.jl, 2021.

[2] M. M. Bosschaert, S. G. Janssens, and Yu. A. Kuznetsov, *Switching to nonhyperbolic cycles from codimension two bifurcations of equilibria of delay differential equations*, SIAM J. Appl. Dyn. Syst., 19 (2020), pp. 252–303, https://doi.org/10.1137/19M1243993.

[3] M. M. Bosschaert and Yu. A. Kuznetsov, *Interplay between normal forms and center manifold reduction for homoclinic predictors near bogdanov-takens bifurcation*, 2021, https://arxiv.org/abs/arXiv:2109.12570.

[4] H. W. Broer, F. Dumortier, S. J. van Strien, and F. Takens, *Structures in dynamics*, vol. 2 of Studies in Mathematical Physics, North-Holland Publishing Co., Amsterdam, 1991. Finite-dimensional deterministic studies.

[5] P. Clément, O. Diekmann, M. Gyllenberg, H. J. A. M. Heijmans, and H. R. Thieme, *Perturbation theory for dual semigroups. I. the sun-reflexive case*, Math. Ann., 277 (1987), pp. 709–725, https://doi.org/10.1007/BF01457866.

[6] P. Clément, O. Diekmann, M. Gyllenberg, H. J. A. M. Heijmans, and H. R. Thieme, *Perturbation theory for dual semigroups. II. time-dependent perturbations in the sun-reflexive case*, Proc. Roy. Soc. Edinburgh Sect. A, 109 (1988), pp. 145–172, https://doi.org/10.1017/S0308210500026731.

[7] P. Clément, O. Diekmann, M. Gyllenberg, H. J. A. M. Heijmans, and H. R. Thieme, *Perturbation theory for dual semigroups. III. nonlinear Lipschitz continuous perturbations in the sun-reflexive case*, in Volterra integrodifferential equations in Banach spaces and applications (Trento, 1987), vol. 190 of Pitman Res. Notes Math. Ser., Longman Sci. Tech., Harlow, 1989, pp. 67–89.

[8] P. Clément, O. Diekmann, M. Gyllenberg, H. J. A. M. Heijmans, and H. R. Thieme, *Perturbation theory for dual semigroups. IV. the intertwining formula and the canonical pairing*, in Semigroup Theory and Applications (Trieste, 1987), vol. 116 of Lecture Notes in Pure and Appl. Math., Dekker, New York, 1989, pp. 95–116.

[9] V. De Witte, F. Della Rossa, W. Govaerts, and Yu. A. Kuznetsov, *Numerical periodic normalization for codim 2 bifurcations of limit cycles: computational formulas, numerical implementation, and examples*, SIAM J. Appl. Dyn. Syst., 12 (2013), pp. 722–788, https://doi.org/10.1137/120874904.

[10] V. De Witte, W. Govaerts, Yu. A. Kuznetsov, and H. G. E. Meijer, *Analysis of bifurcations of limit cycles with Lyapunov exponents and numerical normal forms*, Phys. D, 269 (2014), pp. 126–141, https://doi.org/10.1016/j.physd.2013.12.002.

[11] O. Diekmann and M. Gyllenberg, *Abstract delay equations inspired by population dynamics*, in Functional Analysis and Evolution Equations, Birkhäuser, 2008, pp. 187–200, https://doi.org/10.1007/978-3-7643-7794-6_12.

[12] O. Diekmann and M. Gyllenberg, *Equations with infinite delay: blending the abstract and the concrete*, J. Differential Equations, 252 (2012), pp. 819–851, https://doi.org/10.1016/j.jde.2011.09.038.

[13] O. Diekmann, S. A. van Gils, S. M. Verduyn Lunel, and H.-O. Walther, *Delay Equations: Functional-, Complex-, and Nonlinear Analysis*, Applied Mathematical Sciences, Springer, 1995, https://doi.org/10.1007/978-1-4612-4206-2.

[14] K. Dijkstra, S. A. van Gils, S. G. Janssens, Yu. A. Kuznetsov, and S. Visser, *Pitchfork-Hopf bifurcations in 1D neural field models with transmission delays*, Phys. D, 297 (2015), pp. 88–101, https://doi.org/10.1016/j.physd.2015.01.004.

[15] T. Dong and X. Liao, *Bogdanov-Takens bifurcation in a tri-neuron BAM neural network model with multiple delays*, Nonlinear Dynam., 71 (2013), pp. 583–595, https://doi.org/10.1007/s11071-012-0683-9.

[16] K. Engelborghs, T. Luzyanina, and D. Roose, *Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL*, ACM Trans. Math. Software, 28 (2002), pp. 1–21, https://doi.org/10.1145/513001.513002.

[17] T. Faria and L. T. Magalhães, *Normal forms for retarded functional-differential equations and applications to Bogdanov-Takens singularity*, J. Differential Equations, 122 (1995), pp. 201–224, https://doi.org/10.1006/jdeq.1995.1145.

[18] T. Faria and L. T. Magalhães, *Normal forms for retarded functional differential equations with parameters and applications to Hopf bifurcation*, Journal of differential equations, 122 (1995), pp. 181–200, https://doi.org/10.1006/jdeq.1995.1144.

[19] F. Giannakopoulos and A. Zapp, *Bifurcations in a planar system of differential delay equations modeling neural activity*, Physica D: Nonlinear Phenomena, 159 (2001), pp. 215–232.

[20] W. J. F. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000, https://doi.org/10.1137/1.9780898719543.

[21] S. Gowda, Y. Ma, A. Cheli, M. Gwóźdź, V. B. Shah, A. Edelman, and C. Rackauckas, *High-performance symbolic-numerics via multiple dispatch*, ACM Commun. Comput. Algebra, 55 (2022), p. 92–96, https://doi.org/10.1145/3511528.3511535, https://doi.org/10.1145/3511528.3511535.

[22] S. G. Janssens, *On a Normalization Technique for Codimension Two Bifurcations of Equilibria of Delay Differential Equations*, master's thesis, Utrecht University, The Netherlands, 2010, https://dspace.library.uu.nl/handle/1874/312252. Corrections and updates are available via http://sebastiaanjanssens.nl/pdf/normalization.pdf.

[23] S. G. Janssens, *A class of abstract delay differential equations in the light of suns and stars*, Jan. 2019, https://arxiv.org/abs/1901.11526.

[24] W. Jiang and Y. Yuan, *Bogdanov–Takens singularity in van der Pol's oscillator with delayed feedback*, Physica D: Nonlinear Phenomena, 227 (2007), pp. 149–161.

[25] J. Jiao and C. Chen, *Bogdanov–Takens bifurcation analysis of a delayed predator-prey system with double allee effect*, Nonlinear Dynamics, 104 (2021), pp. 1697–1707, https://doi.org/10.1007/s11071-021-06338-x, https://doi.org/10.1007/s11071-021-06338-x.

[26] H. B. Keller, *Lectures on Numerical Methods in Bifurcation Problems*, vol. 79 of Tata Institute of Fundamental Research Lectures on Mathematics and Physics, Published for the Tata Institute of Fundamental Research, Bombay; by Springer-Verlag, Berlin, 1987. With notes by A. K. Nandakumaran and Mythily Ramaswamy.

[27] Yu. A. Kuznetsov, *Numerical normalization techniques for all codim 2 bifurcations of equilibria in ODEs*, SIAM Journal on Numerical Analysis, 36 (1999), pp. 1104–1124, https://doi.org/10.1137/S0036142998335005.

[28] Yu. A. Kuznetsov, *Practical computation of normal forms on center manifolds at degenerate Bogdanov–Takens bifurcations*, International Journal of Bifurcation and Chaos, 15 (2005), pp. 3535–3546.

[29] Yu. A. Kuznetsov, W. Govaerts, E. J. Doedel, and A. Dhooge, *Numerical periodic normalization for codim 1 bifurcations of limit cycles*, SIAM J. Numer. Anal., 43 (2005), pp. 1407–1435, https://doi.org/10.1137/040611306.

[30] C. Rackauckas and Q. Nie, *Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia*, Journal of Open Research Software, 5 (2017).

[31] S. Ruan, *On the zeros of a third degree exponential polynomial with applications to a delayed model for the control of testosterone secretion*, Mathematical Medicine and Biology, 18 (2001), pp. 41–52, https://doi.org/10.1093/imammb/18.1.41, https://doi.org/10.1093%2Fimammb%2F18.1.41.

[32] G. Samaey, K. Engelborghs, and D. Roose, *Numerical computation of connecting orbits in delay differential equations*, Numer. Algorithms, 30 (2002), pp. 335–352, https://doi.org/10.1023/A:1020102317544, https://doi-org.proxy.library.uu.nl/10.1023/A:1020102317544.

[33] D. P. Sanders, L. Benet, Lucaferranti, Krish Agarwal, B. Richard, J. Grawitter, Eeshan Gupta, M. Forets, M. F. Herbst, Yashrajgupta, E. Hanson, Braam Van Dyk, C. Rackauckas, Rushabh Vasani, S. Micluța-Câmpeanu, Sheehan Olver, T. Koolen, C. Wormell, Favio André Vázquez, G. Dalle, J. Sarnoff, J. TagBot, K. O'Bryant, K. Carlsson, M. Piibeleht, M. Giordano, , Reno, R. Deits, T. Holy, and Venkateshprasad, *Juliaintervals/intervalarithmetic.jl: v0.20.6*, 2022, https://doi.org/10.5281/ZENODO.6569509, https://zenodo.org/record/6569509.

[34] J. Sieber, *Local bifurcations in differential equations with state-dependent delay*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 27 (2017), p. 114326, https://doi.org/10.1063/1.5011747, https://arxiv.org/abs/1705.07550.

[35] J. Sieber, K. Engelborghs, T. Luzyanina, G. Samaey, and D. Roose, *DDE-BIFTOOL Manual - Bifurcation analysis of delay differential equations*, June 2014, https://arxiv.org/abs/1406.7144.

[36] S. A. van Gils, S. G. Janssens, Yu. A. Kuznetsov, and S. Visser, *On local bifurcations in neural field models with transmission delays*, Journal of Mathematical Biology, 66 (2013), pp. 837–887, https://doi.org/10.1007/s00285-012-0598-6, https://arxiv.org/abs/1209.2849.