NUMERICAL CONTINUATION OF BIFURCATIONS OF LIMIT CYCLES IN MATLAB

WILLY GOVAERTS*, YURI A. KUZNETSOV†, AND A. DHOOGE‡

Abstract. CL_MATCONT and MATCONT are MATLAB continuation packages for the interactive numerical study of a range of parameterized nonlinear dynamical systems, in particular ODEs. MATCONT is an interactive graphical package and CL_MATCONT is a commandline version. Both packages allow to compute curves of equilibria, limit points, Hopf points, limit cycles, flip, fold and torus bifurcation points of limit cycles. We discuss computational details of the continuation of limit cycles and flip, fold and torus bifurcations of limit cycles in MATCONT and CL_MATCONT using orthogonal collocation. Instead of the more commonly used fully extended systems we use minimally extended systems. We further describe the use of the MATLAB sparse matrix routines and the initialization and adaptation of the bordering vectors that are essential in minimally extended system. Finally, we compare the use of the minimally and the fully extended systems in the MATLAB environment.

Key words. Limit Cycle,Fold, Flip, Torus, Matlab

AMS subject classifications. 65P30

1. Introduction.

Numerical continuation is a well-understood subject, see e.g. [1], [8],[9], [4], [18], [21], [23]. The idea is as follows. Consider a smooth function $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$. We want to compute a solution curve of the equation

$$F(x) = 0. \tag{1.1}$$

Numerical continuation is a technique to compute a sequence of points which approximate the desired solution branch.

In particular, we consider a dynamical system of the form

$$\frac{dx}{dt} = f(x, \alpha) \tag{1.2}$$

with $x \in \mathbb{R}^n$, $f(x,\alpha) \in \mathbb{R}^n$, and α a vector of parameters where equilibria, limit points, limit cycles etcetera can be computed.

The existing software packages such as AUTO [10], CONTENT [22] require the user to rewrite his/her models in a specific form and use special internal formats to store results; this complicates the export of the results, graphical representation etcetera.

The aim of MATCONT and CL_MATCONT is to provide a continuation toolbox which is compatible with the standard MATLAB ODE representation of differential equations. General descriptions of CL_MATCONT and MATCONT are in [7] and [6] respectively. The current version of the package is freely available for download at:
`http://allserv.rug.ac.be/~ajdhooge`
CL_MATCONT requires MATLAB 5.3 whereas MATCONT requires MATLAB 6.*. In the present paper we concentrate on the implementation in CL_MATCONT and MATCONT of the continuation of the flip (Period Doubling, PD), fold (Limit Point of Cycles,

*Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9,B-9000 Gent, Belgium (`Willy.Govaerts@UGent.be`).

†Mathematisch Instituut, Universiteit Utrecht, Budapestlaan 6, 3854 CD Utrecht, The Netherlands (`Yu.A.Kuznetsov@math.uu.nl`).

‡Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9,B-9000 Gent, Belgium (`Annick.Dhooge@UGent.be`).

LPC) and torus (Neimark - Sacker, NS) bifurcations of limit cycles, using minimally extended systems and orthogonal collocation to discretize appearing boundary-value problems (BVPs). The only existing software to perform these continuations is AUTO [10] which uses fully extended systems and their orthogonal collocation. Several algorithms for the continuation of limit cycles and fold bifurcations of limit cycles are studied in [19]. They are also based on MATLAB and minimally extended systems but otherwise very different from ours since they use a combination of multiple shooting and automatic differentiation, using coarse meshes and Taylor series expansions. Accuracy and robustness are the main object of [19] and these are compared to the AUTO results; since no code is available we cannot compare them directly to our results.

The theory of the bordering method for the numerical continuation of bifurcations of limit cycles was developed in [12] and is summarized in a slightly generalized form in §2. We note that the convergence properties of the discretized solutions of the PD, LPC and NS equations can be proved by the method of reformulation of boundary value problems as described in [3] and applied in [9] to include the period of the orbit, a phase condition and parameters in the boundary value problem formulation. In particular, the convergence of the systems for PD, LPC and NS is of order h^m where h is the maximum length of the mesh intervals and m is the number of collocation points in each mesh interval. Also, there is superconvergence (of order h^{2m}) in the endpoints of the mesh intervals and for the scalar equations added to the LC equations.

In this paper we deal with the numerical and computational aspects of the implementation in MATCONT and CL_MATCONT. Note that this makes MATCONT the first fully interactive software that supports the continuation of the limit cycle bifurcations using orthogonal collocation. In §3 we recall the basic methods for the continuation of limit cycles. In §4, 5, 6 we discuss the continuation of the PD, LPC and NS bifurcations, respectively. §7 discusses further details on the initialization and adaptation of the continuation which are largely common to the three cases. In §8 we consider the computation of multipliers along the three curve types. In §9 we provide an example with the LP and NS continuation in MATCONT. In §10 we make a comparison between the use of the minimally and fully extended systems in the case of PD cycles.

2. Mathematical Background on Limit Cycles and Bifurcations of Limit Cycles.

In this section we summarize the main results of [12]; in the flip and torus cases we actually present an easy generalization.

2.1. Limit Cycles.

A *cycle* is a closed orbit corresponding to a periodic solution of (1.2) with period T, i.e. $x(0) = x(T)$. By definition, in a neighbourhood of a *limit cycle* there are no other cycles.

Since T is not known in advance, it is customary to use an equivalent system defined on the fixed interval $[0, 1]$ by rescaling time [10], [22]:

$$\left\{ \begin{array}{l} \dfrac{dx}{dt} - Tf(x,\alpha) = 0 \\ x(0) = x(1) \end{array} \right. \tag{2.1}$$

A phase shifted function $\phi(t) = x(t+s)$ is also a solution of (2.1) for any value of s. To obtain a unique solution an extra constraint is needed. The following integral constraint is often used [10],[22]

$$\int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt = 0 \tag{2.2}$$

where $\dot{x}_{old}(t)$ is the tangent vector of a previously calculated limit cycle and is therefore known, $\langle x, v\rangle$ is just a different notation for $x^T v$. This condition tries to select the solution which has the smallest phase difference with respect to the previous solution x_{old}.

Thus, the complete BVP defining a limit cycle consists of (2.1) and (2.2).

2.2. Flip Bifurcation of Limit Cycles. A flip bifurcation of limit cycles (Period Doubling, PD) generically indicates a period doubling of the periodic solution, i.e., there are nearby periodic solutions of approximately double period. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2) where G is the flip test function specified below. The complete BVP defining a PD point using the minimal extended system is

$$\left\{\begin{array}{ll} \frac{dx}{dt} - Tf(x,\alpha) & = 0 \\ x(0) - x(1) & = 0 \\ \int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt & = 0 \\ G[x,T,\alpha] & = 0 \end{array}\right. \tag{2.3}$$

where G is defined by requiring

$$N_1 \begin{pmatrix} v \\ G \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.4}$$

Here v is a function and G is a scalar and

$$N_1 = \begin{bmatrix} D - Tf_x(x(\cdot),\alpha) & w_{01} \\ \delta_0 + \delta_1 & w_{02} \\ Int_{v_{01}} & 0 \end{bmatrix} \tag{2.5}$$

where the bordering functions v_{01}, w_{01}, vector w_{02} are chosen so that N_1 is nonsingular [12]; δ_0, δ_1 are the Dirac operators defined by $\delta_i(f) = f(i)$ for $f \in \mathcal{C}^1([0,1], \mathbb{R}^n)$. $Int_{v_{01}}$ is the operator defined by $Int_{v_{01}} f = \int_0^1 v_{01}^T(t) f(t) dt$ for $f \in \mathcal{C}^1([0,1], \mathbb{R}^n)$. We note that in [12] the entry corresponding to w_{02} is zero; the generalization to (2.5) is easy.

2.3. Fold Bifurcation of Limit Cycles. A fold bifurcation of limit cycles (Limit Point of Cycles, LPC) generically corresponds to a turning point of a curve of limit cycles. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2) where G is the fold test function specified below. The complete BVP defining a LPC point using the minimally extended system is

$$\left\{\begin{array}{ll} \frac{dx}{dt} - Tf(x,\alpha) & = 0 \\ x(0) - x(1) & = 0 \\ \int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt & = 0 \\ G[x,T,\alpha] & = 0 \end{array}\right. \tag{2.6}$$

where G is defined by requiring

$$N_2 \begin{pmatrix} v \\ S \\ G \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.7}$$

Here v is a function, S and G are scalars and

$$N_2 = \left[\begin{array}{ccc} D - Tf_x(x(\cdot),\alpha) & -f(x(\cdot),\alpha) & w_{01} \\ \delta_1 - \delta_0 & 0 & w_{02} \\ Int_{f(x(\cdot),\alpha)} & 0 & w_{03} \\ Int_{v_{01}} & v_{02} & 0 \end{array} \right] \tag{2.8}$$

where the bordering functions v_{01}, w_{01}, vector w_{02} and scalars v_{02} and w_{03} are chosen so that N_2 is nonsingular [12].

2.4. Torus Bifurcation of Limit Cycles. A torus bifurcation of limit cycles (Neimark-Sacker, NS) generically corresponds to a bifurcation to an invariant torus, on which the flow contains periodic or quasi-periodic motions. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2) where G is the torus test function which has four components. The complete BVP defining a NS point using the minimal extended system is

$$\left\{ \begin{array}{ll} \frac{dx}{dt} - Tf(x,\alpha) & = 0 \\ x(0) - x(1) & = 0 \\ \int_0^1 \langle x(t), \dot{x}_{old}(t) \rangle dt & = 0 \\ G[x,T,\alpha] & = 0 \end{array} \right. \tag{2.9}$$

where

$$G = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix}$$

is defined by requiring

$$N_3 \begin{pmatrix} v^1 & v^2 \\ G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.10}$$

Here v^1 and v^2 are functions and G_{11}, G_{12}, G_{21} and G_{22} are scalars and

$$N_3 = \left[\begin{array}{ccc} D - Tf_x(x(\cdot),\alpha) & w_{11} & w_{12} \\ \delta_0 - 2\kappa\delta_1 + \delta_2 & w_{21} & w_{22} \\ Int_{v_{01}} & 0 & 0 \\ Int_{v_{02}} & 0 & 0 \end{array} \right] \tag{2.11}$$

where the bordering functions $v_{01}, v_{02}, w_{11}, w_{12}$, vectors $w_{21}$and w_{22} are chosen so that N_3 is nonsingular. We note that in [12] the entries corresponding to w_{21} and w_{22} are zero; the generalization to (2.11) is easy.

3. Numerical Continuation of Limit Cycles. For the numerical continuation of a limit cycle with respect to a parameter we need to discretize the system consisting of (2.1) and (2.2); to use a Newton-like method the Jacobian matrix of the discretized system is also needed. This is a sparse matrix and we exploit the sparsity by using the MATLAB routines for sparse matrices. The same applies to the Jacobians of the equations that define bifurcations of limit cycles in §4,5,6, which have the similar sparsity patterns.

3.1. Discrete Representation of the Solution Function.. The method used to discretize the BVP is the same as in COLSYS[2], AUTO[10] and CONTENT[22] and is called the *orthogonal collocation* [5]. First the interval [0 1] is subdivided in N smaller *test intervals*:

$$0 = \tau_0 < \tau_1 < \cdots < \tau_N = 1.$$

On each of these intervals the solution $x(\tau)$ is approximated by an order m vector valued polynomial $x^{(i)}(\tau)$. This is done by defining $m+1$ equidistant *mesh points* on each test interval

$$\tau_{i,j} = \tau_i + \frac{j}{m}(\tau_{i+1} - \tau_i) \ (i = 0, 1, \ldots, N-1)(j = 0, 1, \ldots, m)$$

and defining the polynomials $x^{(i)}(\tau)$ as

$$x^{(i)}(\tau) = \sum_{j=0}^{m} x^{i,j} l_{i,j}(\tau).$$

Here $x^{i,j}$ is the approximation of $x(\tau)$ at $\tau = \tau_{i,j}$ (we note that $x^{i,m} = x^{i+1,0}$ for $i = 0, 1, \ldots, N-1$) and the $l_{i,j}(\tau)$'s are the Lagrange basis polynomials

$$l_{i,j}(\tau) = \prod_{k=0, k\neq j}^{m} \frac{\tau - \tau_{i,k}}{\tau_{i,j} - \tau_{i,k}}.$$

On each test interval $[\tau_i, \tau_{i+1}]$ we require that the polynomials $x^{(i)}(\tau)$ satisfy the BVP exactly in m collocation points $\zeta_{i,j}$ $(j = 1, \ldots, m)$ (which are different from the mesh points $\tau_{i,j}$). It can be proved that the best choice for the collocation points are the Gauss points [5]. These are the roots of the Legendre polynomial relative to the interval $[\tau_i, \tau_{i+1}]$. With this choice of collocation points the error in the approximation is extremely small

$$||x(\tau_{i,j}) - x^{i,j}|| = \mathcal{O}(h^m)$$

where $h = max\{|t_i| : i = 1, 2, \ldots, N\}$, $t_i = \tau_i - \tau_{i-1}$ and for the mesh points τ_i it is even better

$$||x(\tau_i) - x^{i,0}|| = \mathcal{O}(h^{2m}).$$

3.2. Numerical evaluation of integrals. In (2.6) and several other places we need to compute integrals over $[0, 1]$ using the discretization discussed in §3.1. For $N = 3$ test intervals and $m = 2$ collocation points the following data are associated with the discretized interval [0 1]

τ_0		τ_1		τ_2		τ_3
○	○	●	○	●	○	○
$\tau_{0,0}$	$\tau_{0,1}$	$\tau_{0,2}$		$\tau_{2,0}$	$\tau_{2,1}$	$\tau_{2,2}$
		$\tau_{1,0}$	$\tau_{1,1}$	$\tau_{1,2}$		$\tau_{3,0}$
$t_1 w_1$	$t_1 w_2$	$t_1 w_3 + t_2 w_1$	$t_2 w_2$	$t_2 w_3 + t_3 w_1$	$t_3 w_2$	$t_3 w_3$
$\sigma_{0,0}$	$\sigma_{0,1}$	$\sigma_{1,0}$	$\sigma_{1,1}$	$\sigma_{2,0}$	$\sigma_{2,1}$	$\sigma_{3,0}$

The total number of mesh points (*tps*) is $N \times m + 1$, the total number of variables (*ncoords*) is *tps* $\times \dim(x)$. Each mesh point $\tau_{i,j}$ in a test interval $[\tau_i, \tau_{i+1}]$ has a particular weight w_{j+1}, the Gauss-Lagrange quadrature coefficient. Some mesh points (the black bullets) belong to two test intervals. We set $t_i = \tau_i - \tau_{i-1}$ $(i = 1, \ldots, N)$. The integration weight $\sigma_{i,j}$ of $\tau_{i,j}$ is given by $w_{j+1}t_{i+1}$ for $0 \leq i \leq N-1$ and $0 < j < m$. For $i = 0, \ldots, N-2$ the integration weight of $\tau_{i,m} = \tau_{i+1,0}$ is given by $\sigma_{i,m} = w_{m+1}t_{i+1} + w_1 t_{i+2}$ and the integration weights of τ_0 and τ_N are given by $w_1 t_1$ and $w_{m+1}t_N$, respectively. The integral $\int_0^1 f(t)dt$ is, therefore, approximated by $\sum_{i=0}^{N-1}\sum_{j=0}^{m-1} f(\tau_{i,j})\sigma_{i,j} + f(1)\sigma_{N,0}$.

3.3. Discretization of the BVP. Using the discretization described in §3.1 we obtain the discretized BVP

$$\left\{\begin{array}{rl} \left(\sum_{j=0}^{m} x^{i,j} l'_{i,j}(\zeta_{i,k})\right) - Tf(\sum_{j=0}^{m} x^{i,j} l_{i,j}(\zeta_{i,k}), \alpha) & = 0 \\ x^{0,0} - x^{N-1,m} & = 0 \\ \sum_{i=0}^{N-1}\sum_{j=0}^{m-1} \sigma_{i,j}\langle x^{i,j}, \dot{x}_{old}^{i,j}\rangle + \sigma_{N,0}\langle x^{N,0}, \dot{x}_{old}^{N,0}\rangle & = 0 \end{array}\right.$$

The first equation in fact consists of Nm equations, one for each combination of $i = 0, 1, 2, ..., N-1$ and $k = 1, 2, ..., m$.

3.4. The Jacobian of the Discretized Equations. The Jacobian of the discretized system is sparse. In the Newton iterations during the continuation process a system consisting of this Jacobian matrix and an extra row (the tangent vector) is solved. For $N = 3$ test intervals, $m = 2$ collocation points and $\dim(x) = 2$ this matrix has the following sparsity structure (•'s are a priori non-zero's).

$$\left(\begin{array}{cccccccccccccccc} \multicolumn{2}{c}{x^{0,0}} & \multicolumn{2}{c}{x^{0,1}} & \multicolumn{2}{c}{x^{1,0}} & \multicolumn{2}{c}{x^{1,1}} & \multicolumn{2}{c}{x^{2,0}} & \multicolumn{2}{c}{x^{2,1}} & \multicolumn{2}{c}{x^{3,0}} & T & \alpha \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & & & & & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & & & & & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & & & & & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & & & & & \bullet & \bullet \\ & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & \bullet & \bullet \\ & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & \bullet & \bullet \\ & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & \bullet & \bullet \\ & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & & & & \bullet & \bullet \\ & & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & & & & & & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & & & & & & & & & & & \bullet & \bullet & & \\ \bullet & \bullet & & & & & & & & & & & \bullet & \bullet & & \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array}\right) \tag{3.1}$$

The columns of (3.1) label the unknowns of the discretized problem. The first $\dim(x)$ rows correspond to the first collocation point etc. In (2.1) and (2.2) there are 3 unknowns: x, the period T and a parameter α. So the part of the Jacobian corresponding with the first equation of (2.1) has the following form:

$$[D - Tf_x(x, \alpha) \qquad - f(x, \alpha) \qquad - Tf_\alpha(x, \alpha)].$$

In (3.1) $D - Tf_x(x, \alpha)$ corresponds to $N = 3$ blocks with dimension 4×6 $(= (\dim(x) * m) \times (\dim(x) \times (m+1)))$. The part in (3.1) that defines the boundary conditions for limit cycles has the form:

$$[I_{\dim(x)} \quad 0_{\dim(x)\times(Nm-1)\dim(x)} \quad -I_{\dim(x)} \quad 0_{\dim(x)}].$$

These are in (3.1) the $\dim(x) = 2$ rows following the 4×6 blocks. These rows contain two nonzero parts corresponding with $x^{0,0}$ and $x^{N,0}$($\pm 2\times 2$ identity matrix). The last but one row in (3.1) is the derivative of the discretization of (2.2). The last row is added by the continuation code.

4. Continuation of PD Cycles..

4.1. Discretization of the PD Equations.. The last equation in (2.3) expresses the condition that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) \\ \delta_0 + \delta_1 \end{bmatrix} \tag{4.1}$$

that appears in (2.5) is rank deficient [12]. In the implementation in MATCONT and CL_MATCONT we replace this by the condition that the discretized operator (matrix) of (4.1) has rank defect 1. To this end we solve

$$N_1^d \begin{pmatrix} v_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.2}$$

where

$$N_1^d = \begin{bmatrix} [\ D \ - \ Tf_x(x(\cdot),\alpha)\ ]_d & & & w_d^1 \\ I_{\dim(x)} & 0_{\dim(x)\times(Nm-1)\dim(x)} & I_{\dim(x)} & w_d^2 \\ & v_d^1 & & 0 \end{bmatrix} \tag{4.3}$$

and the bordering vectors v_d^1, w_d^1 and w_d^2 are chosen so that (4.3) has full rank. Here and elsewhere the subscript d denotes discretization using orthogonal collocation. The structure is similar to that of (3.1); however, the two last rows and colums of (3.1) are replaced by the single last row and column of (4.3).

4.2. The Jacobian of the Discretized PD Equations. To continue the discretized equations of (2.3) the Jacobian matrix of the system is needed which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

Taking derivatives of (4.2) with respect to z (being a component of $x^{i,j}, T$ or α) we obtain

$$N_1^d \begin{pmatrix} v_{dz} \\ G_{dz} \end{pmatrix} + \begin{bmatrix} ([-Tf_x(x(\cdot),\alpha)]_d)_z & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} v_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Or, equivalently,

$$N_1^d \begin{pmatrix} v_{dz} \\ G_{dz} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot),\alpha]_{dz} v_d \\ 0 \\ 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$(w_1^*, w_2^*, w_3^*) N_1^d = (0, 0, 1) \tag{4.4}$$

where w_1 is a $\dim(x)*N*m$ vector, w_2 a $\dim(x)$ vector and w_3 is a scalar. Combining (4.2) and (4.4) we find

$$G_{dz} = -w_1^*([Tf_x(x(\cdot),\alpha]_{dz}v_d. \tag{4.5}$$

So in each iteration step we solve two systems with the structure of (4.3) or its transpose.

5. Continuation of LPC Cycles..

5.1. Discretization of the LPC Equations.. The last equation in (2.6) expresses that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) & -f(x(\cdot),\alpha) \\ \delta_1 - \delta_0 & 0 \\ Int_{f(x(\cdot),\alpha)} & 0 \end{bmatrix} \tag{5.1}$$

that appears in (2.8) is rank deficient. In the numerical implementation in MATCONT and CL_MATCONT we replace this by the condition that the discretized operator of (5.1) is rank deficient. We solve

$$N_2^d \begin{pmatrix} v_d \\ S_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{5.2}$$

where

$$N_2^d = \begin{bmatrix} [\ D \ - \ Tf_x(x(\cdot),\alpha)\ ]_d & [-f(x(\cdot),\alpha)]_d & w_d^1 \\ I_{\dim(x)}\ 0_{n\times(Nm-1)\dim(x)}\ -I_{\dim(x)} & 0 & w_d^2 \\ Int_{f(x(\cdot),\alpha)_d} & 0 & w_d^3 \\ v_d^1 & v_d^2 & 0 \end{bmatrix} \tag{5.3}$$

where the bordering vectors v_d^1, w_d^1 and w_d^2 and scalars v_d^2 and w_d^3 are chosen so that N_2^d is nonsingular. The structure is similar to that of (3.1); however the two last rows and colums have a different meaning. The last but one row corresponds with $Int_{[f(x(\cdot),\alpha)]_d}$ and the last but one column corresponds with $[-f(x(\cdot),\alpha)]_d$.

5.2. The Jacobian of the Discretized LPC Equations. To continue the discretized equations of (2.6) the Jacobian matrix of the system is needed which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

The derivative with respect to z (being a component of $x^{i,j}, T$ or α) is

$$N_2^d \begin{pmatrix} v_{dz} \\ S_{dz} \\ G_{dz} \end{pmatrix} + \begin{bmatrix} ([-Tf_x(x(\cdot),\alpha)]_d)_z & ([-f(x(\cdot),\alpha)]_d)_z & 0 \\ 0 & 0 & 0 \\ (Int_{[f(x(\cdot),\alpha)]_d})_z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} v_d \\ S_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Simplifying gives

$$N_2^d \begin{pmatrix} v_{dz} \\ S_{dz} \\ G_{dz} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot),\alpha]_{dz}v_d + [f(x(\cdot),\alpha)]_{dz}S_d \\ 0 \\ -\mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}}v_d \\ 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$(w_1^*, w_2^*, w_3, w_4)N_2^d = (0,0,0,1) \tag{5.4}$$

where w_1 is a $\dim(x) * N * m$ vector, w_2 a $\dim(x)$ vector and w_3 and w_4 are scalars. Combining (5.2) and (5.4) we find

$$G_{dz} = w_1^*([Tf_x(x(\cdot),\alpha]_{dz} v_d + [f(x(\cdot),\alpha)]_{dz} S_d) - w_3 \mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d. \tag{5.5}$$

So in each iteration step we solve two systems with the structure of (3.1) or its transpose.

5.3. Details of the computation of G_{dz}. We restrict to the computation of the contribution of the term $-w_3 \mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d$ in (5.5), the computation of the other terms being similar. We first introduce the vector

$$\Sigma = (\sigma_{0,0}, \dots, \sigma_{0,0}, \sigma_{0,1}, \dots, \sigma_{0,1}, \dots, \sigma_{1,0}, \dots, \sigma_{1,0}, \dots, \sigma_{N,0}, \dots, \sigma_{N,0})^T$$

where each weight $\sigma_{i,j}$ is repeated $\dim(x)$ times.

Let $(\Sigma. * v_d)$ be the element-by-element product of the vectors Σ and v_d. Then

$$\mathrm{Int}_{[f(x(\cdot),\alpha)]_d} v_d = (\Sigma. * v_d)^T F((x^{i,j}),\alpha)$$

where $F((x^{i,j}),\alpha)$ is the column vector consisting of the $f(x^{i,j},\alpha)$ for $i = 0,\dots,N-1,\ j = 0,\dots,m-1$ and $f(x^{N,0},\alpha)$. So

$$\mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d = (\Sigma. * v_d)^T F((x^{i,j}),\alpha)_z.$$

Since T does not appear in this expression, there is no contribution to G_{dT}. The contribution to $G_{dx^{i,j}}$ is

$$-w_3(\Sigma. * v_d)^T_{(i+mj)\dim(x)+1,\dots,(i+mj+1)\dim(x)} f_x(x^{i,j},\alpha);$$

The contribution to $G_{d\alpha}$ is

$$-w_3(\Sigma. * v_d)^T_{(i+mj)\dim(x)+1,\dots,(i+mj+1)\dim(x)} f_\alpha(x^{i,j},\alpha).$$

6. Continuation of NS Cycles..

6.1. Discretization of the NS Equations.. The last equation in (2.9) expresses that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) \\ \delta_0 - 2\kappa\delta_1 + \delta_2 \end{bmatrix} \tag{6.1}$$

that appears in (2.11) has rank defect 2. In the numerical implementation in MATCONT and CL_MATCONT we replace this by the requirement that the discretized equation of (6.1) has rank defect 2. We solve

$$N_3^d \begin{pmatrix} v_d^1 & v_d^2 \\ G_d^{11} & G_d^{12} \\ G_d^{21} & G_d^{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{6.2}$$

where

$$N_3^d = \begin{bmatrix} [D - Tf_x(x(\cdot), \alpha)]_d & w_d^{11} & w_d^{12} \\ (\delta_0 - 2\kappa\delta_1 + \delta_2)_d & w_d^{21} & w_d^{22} \\ v_d^{01} & 0 & 0 \\ v_d^{02} & 0 & 0 \end{bmatrix} \tag{6.3}$$

and

$$[(\delta_0 - 2\kappa\delta_1 + \delta_2)_d] =$$

$$= [I_{\dim(x)} \;\; 0_{\dim(x)\times(Nm-1)\dim(x)} \;\; -2\kappa I_{\dim(x)} \;\; 0_{\dim(x)\times(Nm-1)\dim(x)} \;\; I_{\dim(x)}]$$

and the bordering vectors $v_d^{01}, v_d^{02}, w_d^{11}, w_d^{12}, w_d^{21}$ and w_d^{22} are chosen so that (6.3) has full rank. Here the subscript d denotes discretization using orthogonal collocation over the interval [0 2]. For $N = 2$ test intervals and $m = 2$ collocation points the following data are associated with the discretized interval [0 2]:

0				1				2
τ_0		τ_1		τ_2		τ_3		τ_4
○	○	●	○	●	○	●	○	○
$\tau_{0,0}$	$\tau_{0,1}$	$\tau_{0,2}$		$\tau_{2,0}$	$\tau_{2,1}$	$\tau_{2,2}$		$\tau_{4,0}$
		$\tau_{1,0}$	$\tau_{1,1}$	$\tau_{1,2}$		$\tau_{3,0}$	$\tau_{3,1}$	$\tau_{3,2}$
t_1w_1	t_1w_2	$t_1w_3 + t_2w_1$	t_2w_2	$t_2w_3 + t_1w_1$	t_1w_2	$t_1w_3 + t_2w_1$	t_2w_2	t_2w_3
$\sigma_{0,0}$	$\sigma_{0,1}$	$\sigma_{1,0}$	$\sigma_{1,1}$	$\sigma_{2,0}$	$\sigma_{2,1}$	$\sigma_{3,0}$	$\sigma_{3,1}$	$\sigma_{4,0}$

The total number of mesh points (*tps*) is now $2N \times m + 1$, the total number of points (*ncoords*) is *tps* $\times \dim(x)$.

The structure is quite similar to that of (3.1); the first part((2dim$(x)Nm \times$ *ncoords*) is now over [0 2] in (6.3) , so it is duplicated (the dimension is approximately doubled). The two last rows and colums are also different. They correspond with the borders.

6.2. The Jacobian of the Discretized NS Equations. To continue the discretized equations of (2.9) the Jacobian matrix of the system is needed which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

The derivative with respect to z (being a component of $x^{i,j}, T$ or α) is

$$N_3^d \begin{pmatrix} v_{dz}^1 & v_{dz}^2 \\ G_{dz}^{11} & G_{dz}^{12} \\ G_{dz}^{21} & G_{dz}^{22} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot), \alpha]_{dz} v_{dz}^1 & [Tf_x(x(\cdot), \alpha]_{dz} v_{dz}^2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$\begin{pmatrix} w_1^{1*} & w_1^{2*} & G_{dz}^{11} & G_{dz}^{12} \\ w_2^{1*} & w_2^{2*} & G_{dz}^{21} & G_{dz}^{22} \end{pmatrix} N_3^d = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.4}$$

where w_1^{1*}, w_2^{1*} are $\dim(x) * 2N * m$ vectors and w_1^{2*}, w_2^{2*} are $\dim(x)$ vectors. Combining (6.2) and (6.4) we find

$$G_{dz}^{ij} = w_i^{1*}([Tf_x(x(\cdot), \alpha]_d)_z v_d^j. \tag{6.5}$$

So in each iteration step we solve two systems with the almost doubled structure of (3.1) or its transpose.

One also needs the derivatives with respect to κ; for this we find

$$M_d^1 \begin{pmatrix} v^1_{d\kappa} & v^2_{d\kappa} \\ G^{11}_{d\kappa} & G^{12}_{d\kappa} \\ G^{21}_{d\kappa} & G^{22}_{d\kappa} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 2v_d^1(1) & 2v_d^2(1) \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

So for κ we find

$$G^{ij}_{d\kappa} = w_i^{2*} v_d^j(1). \tag{6.6}$$

where $i, j \in \{1, 2\}$.

7. Initialization and adaptation of the borders.

The bordering vectors in (4.3), (5.3), (6.3) must be such that the matrices $N^d_{1,2,3}$ are nonsingular. We actually choose them in such a way that the corresponding matrices $N^d_{1,2,3}$ are as well conditioned as possible. This involves an initialization of the borders when the continuation is started and a subsequent adaptation during the continuation.

7.1. The flip and fold cases. During the initialization the borders must chosen so that the extensions N_1^d of

$$O_1 = \begin{bmatrix} \multicolumn{3}{c}{[\quad D \quad - \quad Tf_x(x(\cdot),\alpha) \quad]_d} \\ I_{\dim(x)} & 0_{\dim(x)\times(Nm-1)\dim(x)} & I_{\dim(x)} \end{bmatrix}$$

in the flip case and N_2^d of

$$O_2 = \begin{bmatrix} \multicolumn{3}{c}{[\ D \quad - \quad Tf_x(x(\cdot),\alpha)\ ]_d} & [-f(x(\cdot),\alpha)]_d \\ I_{\dim(x)} & 0_{\dim(x)\times(Nm-1)\dim(x)} & -I_{\dim(x)} & 0 \\ \multicolumn{3}{c}{Int_{f(x(\cdot),\alpha)_d}} & 0 \end{bmatrix}$$

in the fold case have full rank. We first perform a QR orthogonal-triangular decomposition with column pivoting.

The MATLAB command $[Q, R, E] = QR(full(O_{1,2}))$ produces a permutation matrix E, an upper triangular matrix R of the same dimension as $O_{1,2}$ and a unitary matrix Q so that $O_{1,2} * E = Q * R$. The column pivoting guarantees that the QR pivoting is rank revealing and in particular that $abs(diag(R))$ is decreasing, cf. [16], §5.4. Since $O_{1,2}$ has rank defect 1, the last element on the diagonal of R should be zero (up to approximation). The borders v_d^1 in (4.3) in the flip case and v_d^1 in (5.3) in the fold case are chosen as the right null vector p of respectively O_1 and O_2. It is well known that this makes the bordered matrices nonsingular. This is sometimes called Keller's Lemma, cf. [20]. A detailed study of the rank of bordered matrices is given in [18], Proposition 3.2.2. From $O_{1,2}p = 0$ follows that $RE^T p = 0$. Setting the bottom right element of R to zero, we obtain

$$\left(\begin{array}{cccccc|c} * & * & * & \dots & * & & * \\ 0 & * & * & \dots & * & & * \\ 0 & 0 & * & \dots & * & & * \\ & & & \dots & & & \dots \\ 0 & 0 & 0 & \dots & * & & * \\ \hline 0 & 0 & 0 & \dots & 0 & & 0 \end{array}\right) E^T p = \begin{pmatrix} \\ \\ 0 \\ \\ \hline 0 \end{pmatrix}. \tag{7.1}$$

This defines E^Tp up to a scalar multiple. Since its last entry must be nonzero we represent E^Tp as $E^Tp = \left(\frac{p_1}{1}\right)$. Therefore (7.1) has the form

$$\left(\begin{array}{c|c} R_1 & b \\ \hline 0 & 0 \end{array}\right)\left(\begin{array}{c} p_1 \\ \hline 1 \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \end{array}\right)$$

with R_1 an nonsingular upper triangular matrix. This is equivalent to

$$R_1 p_1 = -b.$$

So the right border is the normalization of $E * [(R_1 \backslash -b); 1]$ (in MATLAB notation). The computation of the left null vector q which corresponds to w_d^1 and w_d^2 in (4.3) in the flip case and to w_d^1, w_d^2 and w_d^3 in (5.3) in the fold case, is easier. In fact q is the last column q_L of Q because $q_L^T * O_{1,2} = q_L^T QRE^T = (0\ldots 0, 1)RE^T = 0E^T = 0$.

It is an attractive idea to use the sparsity of $O_{1,2}$ to do a Q-less QR decomposition, $R = QR(O_{1,2})$ as provided in MATLAB, which returns only R. This is enough to find the right singular vector of $O_{1,2}$. The left null vector would then be computed as the normalized right null vector of the transposed $O_{1,2}$. However, we found that this method is not robust and that a QR factorization with pivoting is needed. We tested this by comparing a sparse Q-less QR decomposition, a full QR decompositon without column pivoting and a full QR decomposition with column pivoting for O_2^T, for the system

$$\left\{\begin{array}{lcl} \dot{u}_1 & = & -u_1 + p_1(1-u_1)e^{u_3} \\ \dot{u}_2 & = & -u_2 + p_1(1-u_1-p_5u_2)e^{u_3} \\ \dot{u}_3 & = & -u_3 - p_3u_3 + p_1p_4(1-u_1+p_2 * p_5 * u_2)e^{u_3} \end{array}\right.$$

where u_1, u_2, u_3 are state variables and p_1, p_2, p_3, p_4, p_5 are parameters. This is the classical model of the $A \to B \to C$ chemical reaction in a stirred tank [11]; we recall that $1 - u_1$ and u_2 denote the concentrations of A and B respectively and u_3 is the temperature. The parameters $p_1, \ldots, p_5$ have a physical meaning; e.g. p_1 is the Damkohler number and p_3 is the heat transfer coefficient. There is a Hopf point for $p_1 = 0.19547, p_2 = 1, p_3 = 1.5, p_4 = 8, p_5 = 0.04$ and $u_1 = 0.57456, u_2 = 0.54511, u_3 = 1.9328$. Starting an LC continuation with p_1 as a free parameter, we find an LPC point at $p_1 = 0.19545$. Starting an LPC continuation from this LPC point with p_1 and p_3 both free parameters for $N = 30$ and $m = 4$, we obtain O_2 as a matrix of dimension 364×364. The matrix

$$R_s = \begin{pmatrix} 0 & 0.000101297163 \\ 0 & 0 \end{pmatrix}$$

is the bottom right 4×4 block of the Q-less decomposition of O_2^T. The matrix

$$R_f = \begin{pmatrix} -0.000000000001 & 0.000088855185 \\ 0 & 0.000048640223 \end{pmatrix}$$

similarly corresponds to the QR decomposition without pivoting of O_2^T as a full matrix and

$$R_{cp} = \begin{pmatrix} 0.009688518843 & 0.000000271536 \\ 0 & 0.000000423924 \end{pmatrix}$$

corresponds to the QR decomposition with column pivoting of O_2^T as a full matrix. Remarkably, there is a clear difference between the bottom right 2×2 blocks of R_s and R_f. The presence of a zero as the first entry on the diagonal of R_s leads to a division by zero in the computation of the left null vector q where q is the result of the analogue of $E * [(R_1 \backslash -b); 1]$. We further note that the QR decomposition as a full matrix without column pivoting is not satisfactory because the third element on the diagonal of R_f is the smallest one and so it does not define the rank. With column pivoting the fourth element on the diagonal of R_{cp} is the smallest and the diagonal elements indeed define the rank.

During the computation of a curve of limit cycle bifurcations the borders v_d^1 in (4.3) in the flip case and v_d^1 and v_d^2 in (5.3) in the fold case can be adapted by replacing them respectively by the normalized v_d in (4.2) and by the normalized v_d and S in (5.2). The borders w_d^1, w_d^2 in (4.3) and w_d^1, w_d^2 and w_d^3 in (5.3) are adapted by solving the transposed equations and replacing them respectively by the normalized w_1^*, w_2^* in (4.4) and by w_1^*, w_2^* and w_3 in (5.4). Only one border is adapted at the same time. An adaptation is performed after each number of k computed continuation points where k is an user-chosen number; $k = 0$ means no adaptation at all.

7.2. The torus case. To initialize the borders in the case of a torus bifurcation, the borders are chosen so that the extension

$$O_3 = \begin{bmatrix} & [\quad D \quad - \quad Tf_x(x(\cdot),\alpha) \quad]_d & \\ I_{\dim(x)} \; 0_{\dim(x)\times(Nm-1)\dim(x)} & -2\kappa I_{\dim(x)} \; 0_{\dim(x)\times(Nm-1)\dim(x)} & I_{\dim(x)} \end{bmatrix}$$

of N_3^d has full rank. We implement this by using a QR orthogonal-triangular decomposition with column pivoting, $[Q, R, E] = QR(full(O_3))$. Here R is an upper triangular matrix whose bottom right 2×2 block consists of zeros up to approximation. Setting this block to zero, we obtain

$$\left(\begin{array}{ccccc|cc} * & * & * & \dots & * & * & * \\ 0 & * & * & \dots & * & * & * \\ 0 & 0 & * & \dots & * & * & * \\ & & & \dots & & \dots & * \\ 0 & 0 & 0 & \dots & * & * & * \\ \hline 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{array}\right) E^T p = \left(\begin{array}{cc} 0 & 0 \\ \hline 0 & 0 \end{array}\right)$$

if p is a two-column matrix whose columns span the right null space of O_3. By imposing some structure on $E^T p$ we get

$$\left(\begin{array}{c|cc} R_1 & b_1 & b_2 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c|c} p_1 & p_2 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right) = \left(\begin{array}{cc} 0 & 0 \\ \hline 0 & 0 \end{array}\right).$$

or

$$R_1(p_1 \; p_2) = -(b_1 \; b_2)$$

where R_1 is a nonsingular square upper triangular matrix. So the two right bordering vectors in (6.3) are initially chosen as the normalization and orthogonalization of

$E * [(R_1\backslash[-b_1,-b_2]); eye(2)]$ where $eye(2)$ is the 2-by-2 identity matrix. The left null matrix consists of the two last columns of Q. Indeed, if we denote this part of Q as Q_L then

$$Q_L^T * O_3 = Q_L^T QRE^T = \begin{pmatrix} 0\ldots 0,1,0 \\ 0\ldots 0,0,1 \end{pmatrix} RE^T = 0E^T = 0.$$

From (2.9) and (2.10) we get four equations $G_{ij} = 0$ $((i,j) \in \{1,2\})$ in the case of a torus bifurcation. The continuation algorithm needs only two of them. To select those two we start with the full QR decomposition, $[Q_1, R_1] = QR((jac\, jacT\, jacp\, 0)')$ where $(jac\ jacT\ jacp)$ is the Jacobi matrix of the limit cycle equations with respect to the state variables, period and parameters, respectively. Extending the equality $(jac\ jacT\ jacp\ 0)Q_1 = R_1^T$ by some simple computations we obtain a decomposition

$$\begin{pmatrix} jac & jacT & jacp & 0 \\ G_{11x} & G_{11T} & G_{11p} & G_{11\kappa} \\ G_{12x} & G_{12T} & G_{12p} & G_{12\kappa} \\ G_{21x} & G_{21T} & G_{21p} & G_{21\kappa} \\ G_{22x} & G_{22T} & G_{22p} & G_{22\kappa} \end{pmatrix} Q_1 = \left(\begin{array}{cccc|ccc} * & 0 & \ldots & 0 & 0 & 0 & 0 \\ * & * & \ldots & 0 & 0 & 0 & 0 \\ & & \ldots & & & \ldots & \\ * & * & \ldots & * & 0 & 0 & 0 \\ \hline ** & * & \ldots & * & & J'_{res} & \end{array}\right)$$

(7.2)

where J_{res} is a 3×4 matrix with rank 2. We now want to choose two among the last four rows of the right-hand-side of (7.2) to make the right-hand-side as wel conditioned as possible. We perform a QR decomposition with pivoting $[Q_2, R_2, E_2] = QR(full(J_{res}))$. So

$$\begin{aligned} J_{res}E_2 &= (q_1\ \ q_2\ \ q_3) \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & 0 & 0 \end{pmatrix}, \\ &= (\ q_1\ \ q_2\ ) \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \end{pmatrix} \end{aligned}$$

from which it follows that the first two columns of $J_{res}E_2$ are linearly independent. This also means that the columns of J_{res} we need to use (equivalently, which G_{ij} we need to choose), are those where the first or second column of E_2 contains an entry equal to 1.

The borders v_d^{01} and v_d^{02} in (6.3) are adapted by replacing them respectively by the normalized and orthogonalized v_d^1 and v_d^2 in (6.2). The borders w_d^{11}, w_d^{22} and w_d^{12}, w_d^{22} in (6.3) are adapted by solving the transposed equations and replacing them respectively by the normalized and orthogonalized w_1^{1*}, w_1^{2*} and w_2^{1*}, w_2^{2*} in (6.4). The new indices are computed in the same way as in the initialization. The general strategy for adaptation during the computation of a curve of NS cycles is the same as in the fold and flip cases.

8. Multipliers.

Multipliers are (optionally) computed in MATCONT as in AUTO[10] and CONTENT[22] by making a special decomposition(condensation of parameters) in (3.1). A periodic solution always has a multiplier equal to 1. At a fold, the multiplier 1 has algebraic multiplicity 2 and geometric multiplicity 1, at a flip there is a simple multiplier equal to -1 and at a torus bifurcation there is a simple conjugate pair of complex eigenvalues with modulus 1. This can be used to check the correctness of the continuation. We note that the computation of multipliers is expensive because the sparse matrix is not handled with the sparse MATLAB routines.

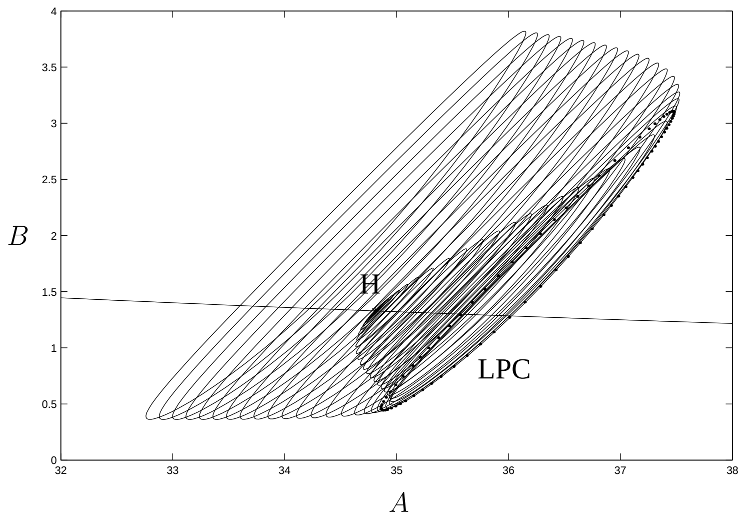


Fig. 9.1. *LPC in the Steinmetz – Larter model.*

9. Example : Fold and torus bifurcations in a chemical model. The following model of the peroxidase - oxidase reaction was studied by Steinmetz and Larter [24] where A, B, X, Y are state variables and k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_{-7} are parameters:

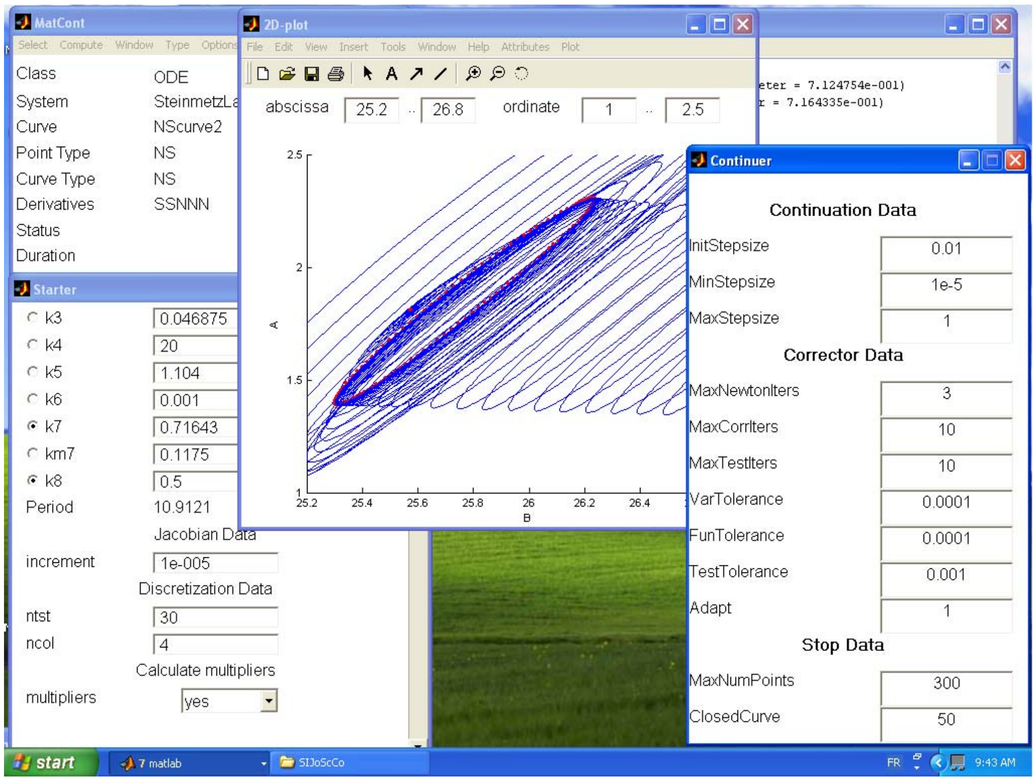
$$(9.1)\qquad \left\{ \begin{array}{rcl} \dot{A} & = & -k_1ABX - k_3ABY + k_7 - k_{-7}A, \\ \dot{B} & = & -k_1ABX - k_3ABY + k_8, \\ \dot{X} & = & k_1ABX - 2k_2X^2 + 2k_3ABY - k_4X + k_6, \\ \dot{Y} & = & -k_3ABY + 2k_2X^2 - k_5Y. \end{array} \right.$$

In [24] Hopf bifurcations, torus bifurcations and onset of chaos are studied. It has been shown recently that (9.1) also possesses generalized Hopf points and associated limit points of cycles (see [17]).

The following values correspond to an unstable equilibrium in (9.1):

Variable	*Value*	*Parameter*	*Value*
A	31.78997	k_1	0.1631021
B	1.45468	k_2	1250
X	0.01524586	k_3	0.046875
Y	0.1776113	k_4	20
		k_5	1.104
		k_6	0.001
		k_7	4.235322
		k_8	0.5
		k_{-7}	0.1175

First, we continue this equilibrium with increasing k_7 while keeping all other parameters fixed. The computed equilibrium branch is a nearly-horizontal curve in **Fig. 9.1**. At $k_7 = 4.59004\ldots$ a subcritical Hopf bifurcation is located (point H). Indeed, there are two complex eigenvalues of the equilibrium with Re $\lambda_{1,2} \approx 0$ at this parameter, while the first Lyapunov coefficient l_1 is positive. Thus, there should exist an unstable limit cycle bifurcating from this equilibrium. Selecting the found



```
eter = 7.124754e-001)
r = 7.164335e-001)
```

Class	ODE
System	SteinmetzLa
Curve	NScurve2
Point Type	NS
Curve Type	NS
Derivatives	SSNNN
Status	
Duration	

abscissa 25.2 .. 26.8 ordinate 1 .. 2.5

k3	0.046875
k4	20
k5	1.104
k6	0.001
k7	0.71643
km7	0.1175
k8	0.5
Period	10.9121
Jacobian Data	
increment	1e-005
Discretization Data	
ntst	30
ncol	4
Calculate multipliers	
multipliers	yes

Continuation Data	
InitStepsize	0.01
MinStepsize	1e-5
MaxStepsize	1
Corrector Data	
MaxNewtonIters	3
MaxCorrIters	10
MaxTestIters	10
VarTolerance	0.0001
FunTolerance	0.0001
TestTolerance	0.001
Adapt	1
Stop Data	
MaxNumPoints	300
ClosedCurve	50

FIG. 9.2. *NS continuation in the Steinmetz - Larter model.*

Hopf point as the new initial point, we can continue in MATCONT a branch of the bifurcating limit cycles and obtain the rest of **Fig. 9.1**. There is a *limit point of cycles* (LPC) at $k_7 = 4.74839\ldots$, where two cycles collide an disappear. The critical cycle has a double multiplier $\mu = 1$ (counting the trivial one). The continuation algorithm automatically follows the second (stable) cycle branch after the LPC point.

Computing the original equilibrium curve in the opposite direction, we get another Hopf point at $k_7 = 0.712475\ldots$, where the first Lyapunov coefficient is negative. This means that a stable limit cycle bifurcates from the equilibrium when it looses stability. Starting from this Hopf point, we obtain the family of stable limit cycles bifurcating from it. At $k_7 = 0.716434\ldots$ a message indicates that a *torus* (NS) bifurcation occurs. Indeed, there are two complex multipliers with (approximately) $|\mu| = 1$. After the bifurcation point the limit cycle becomes unstable (with two multipliers satisfying $|\mu| > 1$) but regains stability at $k_7 = 0.818566\ldots$ through another NS bifurcation.

The found Hopf, LPC and NS points can be used as starting points for the 2-parameter continuation of the corresponding codim 1 bifurcations, using k_7 and k_8 as two control parameters. **Fig. 9.2** gives a graphical impression of a typical MATCONT session. This figure shows the MATCONT main window, the Starter and Continuer windows for the NS continuation and also graphical output in a 2D - plot window; in the latter A is plotted versus B. For more information on the MATCONT windows we refer to the MATCONT website.

The computed with MATCONT bifurcation curves are presented in **Fig. 9.3**, where a (partial) bifurcation diagram in the (k_7, k_8)-plane is shown together with the

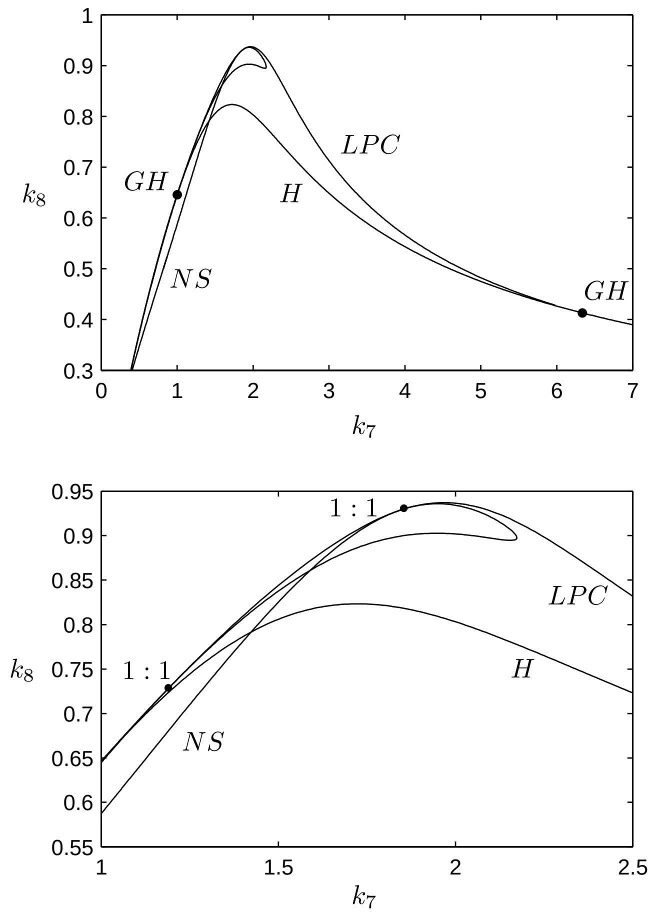


Fig. 9.3. *Bifurcation curves in Steinmetz - Larter model: H - Hopf bifurcation with two generalized Hopf points (labeld by GH); LPC - limit point of cycles; NS - Neimark-Sacker (torus) bifurcation curve with two* 1 : 1 *strong resonances.*

enlargement of an interesting region. There are two *generalized Hopf points* in the Hopf bifurcation curve (labeld by GH), where the associated first Lyapunov coefficient l_1 vanishes [17]. As theory predicts (see, for example, [21]), from each GH point emanates a LPC-curve. Actually, there is just one LPC curve connecting the GH-points; while approaching these points, the critical nonhyperbolic cycle shrinks to the equilibrium point. The shape of the NS curve is more complicated: There are two more codim 2 points on it, where a triple multiplier $\mu = 1$ is present (counting the trivial one). These are 1:1 *strong resonance* points [21], where the LPC and NS curves meet tangentially. Between the 1:1 points, the NS curve is a *neutral saddle cycle* curve. Near such codim 2 points complicated homoclinic structures exist.

It should be noted that the implemented in MATCONT algorithms for the LPC and NS continuation are robust enough to pass through the 1:1 resonance points and to closely approach the GH points (within the 10^{-3}-range in the parameters).

10. Fully versus minimally extended system. The methods described in this paper to continue bifurcation points of limit cycles are different from the traditional ones. Classical software like AUTO uses fully extended systems where additional vector functions are added to the system for the limit cycle to form a big system of approximately double size in the fold and flip cases and triple size in the torus case. In the PD case this leads to

$$\begin{cases} \dfrac{dx}{dt} - Tf(x,\alpha) & = 0 \\ x(0) - x(1) & = 0 \\ \int_0^1 \langle x(\cdot), \dot{x}_{old}(t)\rangle dt & = 0 \\ \dot{v}(t) - Tf_x(x,\alpha)v(t) & = 0 \\ v(0) + v(1) & = 0 \\ \int_0^1 \langle v_{old}(t), v(t)\rangle dt - 1 & = 0 \end{cases}$$

where x, v, T, α are the unknowns and v is a rescaled version of the variable with the same name in (2.4).

This system is easier to code since there is no need to find and adapt the bordering vectors. However, it has a big disadvantage: it has approximately twice the size of the minimally extended system. On the other hand, for the minimally extended system the additional system (2.4) has to be solved. But solving 2 systems of size $n \times n$ takes in general less time then solving one system of size $2n \times 2n$ (in the case of full matrices about a quarter). This suggests that using the big system will be slower than using the minimally extended system.

We did some comparisons in the case of the PD bifurcations in a feedback control system, described in [14], [15] and further used in [21] (Example 5.4, p. 178):

$$\begin{cases} \dot{x} & = & y \\ \dot{y} & = & z \\ \dot{z} & = & -\alpha z - \beta y - x + x^2 \end{cases}$$

Due to the special structure of this system, a good approximation to the PD curve can be found by the harmonic balance method, cf. [25], [26].

We compute numerically a branch of PD cycles as described in the CL_MATCONT manual and [7]. To avoid the overhead of the GUI all computations were done in CL_MATCONT. Both the minimally and the fully extended systems were implemented and tested for several numbers of test and collocation points. In each case we compute 300 continuation points with the same continuation parameters.

From **Table 10.1** it appears that the minimally extended system is faster; moreover it computes a longer stretch of the PD branch. We illustrate this in **Fig. 10** in the case (e) ($N = 30, m = 4$) but it holds in all cases.

This comparison is crude because the continuation variables are in different spaces and therefore thresholds and stepsizes have a different meaning in the two cases. To describe the more refined comparisons we note that in MATCONT and CL_MATCONT the convergence of Newton iterations in the continuation of a branch of solutions to (1.1) is declared if two conditions are satisfied, namely

$$\|F(x)\| \leq F_t, \|\delta x\| \leq V_t,$$

where δx is the Newton correction and F_t (function tolerance) and V_t (variable tolerance) are user - chosen threshold values. Also, the code uses an integer parameter

	N	m	minimally extended system	fully extended system
(a)	10	4	183,2 s	198,3 s
(b)	10	5	230,2 s	251,7 s
(c)	20	4	360,7 s	409,5 s
(d)	20	5	458,0 s	534,1 s
(e)	30	4	540,6 s	642,0 s
(f)	30	5	683,6 s	857,8 s
(g)	50	4	885,6 s	1184,8 s
(h)	50	5	1150,6 s	1616,5 s

Table 10.1
Time comparison between two methods for 300 *PD cycles.*

N_a which indicates that the settings of the continuation are adapted after each N_a computed points. Adaptation is a curve - type dependent process. In the continuation of limit cycles it always includes the adaptation of the mesh. In the minimally extended systems it also includes updating the borders as described in §7. Setting $N_a = 0$ means no adaptation at all. This cannot be recommended but it can be studied usefully as the limit case for large N_a.

In the other experiments we choose situations where both methods compute the same stretch of curve with the same number of points but with F_t, V_t adapted to the dimension of the space of continuation variables. In all cases we have $N = 50, m = 5$, $\dim(x) = 3$. The object function of the minimally extended system (F in (1.1)) has dimension $Nm\dim(x) + \dim(x) + 2 = 755$. The dimension of the space of independent variables (x in (1.1)) is therefore 756. For the fully extended system these dimensions are, respectively, $2Nm\dim(x) + 2\dim(x) + 2 = 1508$ and 1509. In the case of the minimally extended system we choose $F_t = V_t = 10^{-4}$. If we ignore scaling problems then corresponding bounds for the fully extended systems are $10^{-4}\sqrt{\frac{1207}{605}} = 1.4133e-4$ and $10^{-4}\sqrt{\frac{1508}{606}} = 1.4128e-4$, respectively.

Both algorithms were forced to compute 150 points in the given stretch by varying the maximal stepsizes. With $N_a = 0$ the minimally extended system took 561.0 seconds, the fully extended system 617.2 seconds. With $N_a = 1$ the minimally extended system took 586.3 seconds, the fully extended system took 687.4 seconds.

In a similar experiment with only 25 computed points and $N_a = 0$ the minimally and fully extended systems took 108.56 and 111.70 seconds, respectively. With $N_a = 1$ the time spans were 111.39 and 118.43 seconds, respectively.

Since scaling is a difficult issue, one might wonder if setting F_t, V_t in the above way really corresponds to the same accuracy in the computed solutions. We therefore did an additional test in the last mentioned experiment (25 points, $N_a = 1$) by computing the norms of the (x, T, α) parts in the last Newton correction in each continuation point. For the minimally extended system the average value of these norms was $5.086e-8$, for the fully extended system it was $1.111e-6$. This clearly indicates that the solutions obtained by the minimally extended system are an order of magnitude more precise. Also, it shows that the computations are actually much more accurate than the modest bounds for F_t, V_t suggest.

To obtain more detailed information we used the MATLAB Profiler. The lines where the most time was spent are given in **Table 10.2** and **Table 10.3**, respectively.

The most obvious conclusion is that in both cases the `newtcorr` procedure (the process of Newton iterates from a predicted to a declared continuation point) con-

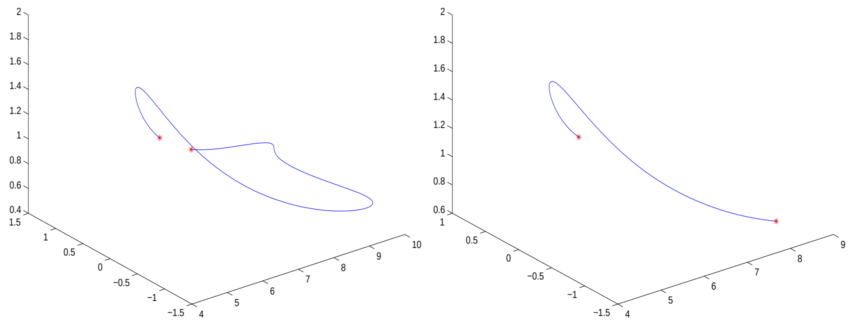


Fig. 10.1. *Period doubling curves computed in* **Table 10.1(e)**

Filename	Calls	Total Time
cont	1	111.390 s
perioddoubling	177	107.093 s
newtcorr	25	105.654 s
contjac	50	84.091 s

Table 10.2
Profile report of PD curve continuation with minimally extended system.

sumes nearly all the time spent in the continuation process; therefore a separate compilation of this algorithm seems highly desirable.

The second observation is that in `newtcorr` most of the time is spent in `contjac`, i.e. in the evaluation of the Jacobian matrix of the system, rather than in the solvers. In **Table 10.4** and **Table 10.5** we provide more detailed information. An application of the sparse solver in the first system takes only 0.0487 seconds; in the second system it takes 0.1381 seconds, a clear advantage for the minimally extended system. The total time spent during solves is 5.9790 s versus 10.3550 s; we note that the 5.9790 s include not only the times in the 4th and 5th rows of Table 10.4 but also a part of the 3rd row since the evaluation of the object function of the minimally extended system involves solving the linear system (2.4). These more detailed data were also provided by the MATLAB Profiler.

For the evaluation of the Jacobians the times are 1.6824 and 1.8014 seconds respectively, a small advantage for the minimally extended system.

The superior performance of the minimally extended system is even more apparent if a higher precision is required. In another experiment we put $F_t = V_t = 10^{-6}$ for the minimally extended system and correspondingly $F_t = 10^{-6}\sqrt{\frac{1207}{605}} = 1.4133e-6$, $V_t = 10^{-6}\sqrt{\frac{1508}{606}} = 1.4128e-6$ for the fully extended system. We computed 25 points with $N_a = 1$; the minimally extended system needed 117.57 seconds, the fully extended system 176.21 seconds.

Our conclusion is that the minimally extended system outperforms the fully extended system. However, the gain is modest because most of the computing time is not spent in the solves (these are performed by the compiled built-in solvers of MATLAB) but in constructing the Jacobian matrices. We can therefore expect that future developments (compiling parts of the code or better handling of for-loops in MATLAB) will make the advantages of the minimally extended system even greater.

Filename	Calls	Total Time
cont	1	118.430 s
perioddoubling2	178	107.243 s
newtcorr	25	110.169 s
contjac	50	90.071 s

Table 10.3
Profile report of PD curve continuation with fully extended system.

Code	Calls	Total Time
A = contjac(x)	25	42.081 s
A=contjac(x)	25	42.040 s
Q =[feval(cds.curve,x);. . .	49	17.756 s
D=b \[Q R']	49	2.484 s
D=([A;v'] \R'	25	1.123 s
All other lines		0.170 s
Totals		105.654 s

Table 10.4
Detailed profile report of newtcorr in Table 10.2.

Code	Calls	Total Time
A = contjac(x)	25	45.064 s
A=contjac(x)	25	45.007 s
Q =[feval(cds.curve,x);. . .	50	9.443 s
D=b \[Q R']	50	7.141 s
D=([A;v'] \R'	25	3.214 s
All other lines		0.300 s
Totals		110.169 s

Table 10.5
Detailed profile report of newtcorr in Table 10.3.

REFERENCES

[1] E. L. Allgower, K. Georg, *Numerical Continuation Methods: An introduction*, Springer-Verlag (1990).

[2] U. Ascher, J. Christiansen and R. D. Russell, *Collocation software for boundary value ODE's*, ACM TOMS 7(2) (1981), pp. 209–222.

[3] U. Ascher and R. D. Russell, *Reformulation of boundary value problems into standard form*, SIAM Review 23(2) (1981), pp. 238–254.

[4] W. J. Beyn, A. Champneys, E. Doedel, W. Govaerts, Yu. A. Kuznetsov, B. Sandstede, *Numerical continuation and computation of normal forms.* In: B. Fiedler, G. Iooss, and N. Kopell (eds.) "Handbook of Dynamical Systems : Vol 2", Elsevier (2002), pp. 149–219.

[5] C. De Boor and B. Swartz, *Collocation at Gaussian points*, SIAM Journal on Numerical Analysis 10(1973), pp. 582–606.

[6] A. Dhooge, W. Govaerts, Yu. A. Kuznetsov ,matcont*: A* matlab *package for numerical bifurcation analysis of ODEs*, ACM TOMS 29(2) (2003), pp. 141–164.

[7] A. Dhooge, W. Govaerts, Yu. A. Kuznetsov , W. Mestrom and A. M. Riet,cl_matcont*: A continuation toolbox in* matlab, Proceedings of the 2003 ACM symposium on applied computing, Melbourne, Florida (2003), pp. 161–166.

[8] E. J. Doedel, H. B. Keller and J. P. Kernevez, *Numerical analysis and control of bifurcation problems I, Bifurcation in finite dimensions*, Int. J. Bifurcation and Chaos, 1 (1991), pp. 493 - 520.

[9] E. J. Doedel, H. B. Keller and J. P. Kernevez, *Numerical analysis and control of bifurcation problems II, Bifurcation in infinite dimensions*, Int. J. Bifurcation and Chaos, 1 (1991), pp. 745 - 772.

[10] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Yu. A. Kuznetsov, B. Sandstede, X. J. Wang, auto97-auto2000 *: Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont)*, User's Guide, Concordia University, Montreal, Canada (1997-2000). (`http://indy.cs.concordia.ca`).

[11] E. J. Doedel and R. F. Heinemann, *Numerical computation of periodic solution branches and oscillatory dynamics of the stirred tank reactor with $A \to B \to C$ reactions*, Chemical Engineering Science, Vol. 38(9)(1983), pp. 1493–1499.

[12] E. J. Doedel, W. Govaerts, Yu. A. Kuznetsov, *Computation of periodic solution bifurcations in ODEs using bordered systems*, SIAM J. Numer. Analysis 41 (2003), pp. 401–435.

[13] E. Freire, A. Rodriguez-Luis, E. Gamero and E. Ponce, *A case study for homoclinic chaos in an autonomous electronic circuit: A trip from Takens-Bogdanov to Hopf- Shilnikov*, Physica D 62 (1993), pp. 230–253.

[14] R. Genesio and A. Tesi, *Harmonic balance methods for the analysis of chaotic dynamics in nonlinear systems.* Automatica 28 (1992), pp. 531–548.

[15] R. Genesio, A. Tesi and F. Villoresi, *Models of complex dynamics in nonlinear systems.* Systems Control Lett. 25 (1995), pp. 185–192.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins, Baltimore, 3rd ed. (1996).

[17] W. Govaerts, Yu. A. Kuznetsov and B. Sijnave, *Numerical methods for the generalized Hopf bifurcation.* SIAM J. Numer Anal. 38 (2000), pp. 329–346.

[18] W. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, SIAM, Philadelphia (2000).

[19] J. Guckenheimer and B. Meloon, *Computing periodic orbits and their bifurcations with automatic differentiation*, SIAM J. Sci. Comput. 22 (2000), pp. 951–985.

[20] H. B. Keller, *Numerical Methods in Bifurcation Problems*, Springer Verlag, Berlin, New York, 1987.

[21] Yu. A. Kuznetsov,*Elements of Applied Bifurcation Theory*, 2nd edition, Springer-Verlag, New York (1998).

[22] Yu. A. Kuznetsov, V. V Levitin,content: Integrated environment for analysis of dynamical systems. CWI, Amsterdam (1997): `ftp://ftp.cwi.nl/pub/CONTENT`

[23] D. Roose D. et al.,*Aspects of continuation software, in : Continuation and Bifurcations: Numerical Techniques and Applications*, (eds. Roose, D., De Dier, B., Spence, A.), NATO ASI series, Series C, Vol. 313, Kluwer (1990), pp. 261–268.

[24] C. Steinmetz and R. Larter, *The quasiperiodic route to chaos in a model of the peroxidase - oxidase reaction*, J. Chem. Phys. 74 (1991), pp. 1388–1396.

[25] A. Tesi, E. H. Abed, R. Genesio and H. O Wang, *Harmonic balance analysis of period - doubling bifurcations with implications for control of nonlinear dynamics*, Automatica 32(9) (1996), pp. 1255–1271.

[26] G. Torrini, R. Genesio and A. Tesi, *On the computation of characteristic multipliers for predicting limit cycle bifurcations*, Chaos, Solitions and Fractals 9 (1/2) (1998), pp. 121–133.