Universität Potsdam

Wolfgang Jansen

CANDYS/QA : algorithms, programs, and user‘s manual

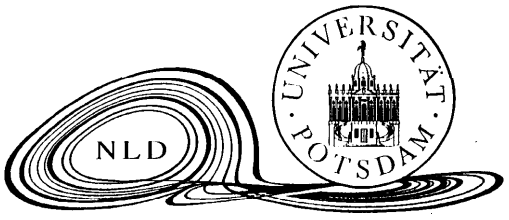
NLD Preprints ; 27

December 1995 Preprint NLD-027

Interdisziplinäres Zentrum für Nichtlineare Dynamik

CANDYS/QA
Algorithms, Programs, and User's Manual

W. Jansen



Universität Potsdam

NLD Preprints[1]**:**

1. F. Feudel and N. Seehafer: On the Bifurcation Phenomena in Truncations of the 2D Navier-Stokes Equations, March 1994
2. A.S. Pikovsky and U. Feudel: Characterizing Strange Nonchaotic Attractors, April 1994
3. J. Kurths, A. Pikovsky and C. Scheffczyk: Roughening Interfaces in Deterministic Dynamics, July 1994
4. A. Witt, J. Kurths, F. Krause and K. Fischer: On the Validity of a Model for the Reversals of the Earth's Magnetic Field, August 1994
5. J. Kurths, A. Voß, A. Witt, P. Saparin, H.J. Kleiner and N. Wessel: Quantitative Analysis of Heart Rate Variability, August 1994
6. Helmholtz – Kolloquium, Institut für Mathematik, 8. September 1994, Vortragsauszüge, September 1994
7. C.-V. Meister: Stochastic Forces on Electrons in the Solar Flare Plasma, November 1994
8. U. Schwarz, T. Förster, A. Jacob, A. Jannasch, D. Kappel, J. Kosche, A. Schütze und S. Wricke: Einführung in die Nichtlineare Dynamik. Skriptum, March 1994
9. F. Feudel, N. Seehafer and O. Schmidtmann: Bifurcation Phenomena of the Magnetofluid Equations, January 1995
10. C.-V. Meister: Stochastic Forces in Space Plasmas with Ion-Acoustic and Lower-Hybrid-Drift Turbulence, January 1995
11. V.A. Liperovsky, C.-V. Meister, K. Schlegel and C. Haldoupis: Currents and Turbulence in and near Mid-Latitude Sporadic E-Layers Caused by Strong Acoustic Impulses, February 1995
12. M. Rosenblum and J. Kurths: A Model of Neural Control of Heart Rate, March 1995
13. K.-U. Thiessenhusen, L.W. Esposito, J. Kurths and F. Spahn: Detection of Hidden Resonances in Saturn's B-Ring, March 1995
14. F. Spahn, J.-M. Hertzsch and N.V. Brilliantov: The Role of Particle Collisions for the Dynamics in Planetary Rings, March 1995
15. A.S. Pikovsky, M.A. Zaks, U. Feudel and J. Kurths: Singular Continuous Spectra in Dissipative Dynamics, March 1995
16. A. Neiman, U. Feudel and J. Kurths: The Cumulant Approach for Investigating the Noise Influence on Mode-Locking Bifurcations, March 1995
17. O. Schmidtmann: Modelling of the Interaction of Lower and Higher Modes in Two-Dimensional MHD-Equations, March 1995
18. F. Feudel, N. Seehafer and O. Schmidtmann: Fluid Helicity and Dynamo Bifurcations, April 1995
19. P. Morin: Visualization in the Geosciences. Course Notes, July 1995
20. V.A. Liperovsky and C.-V. Meister: Sporadic E-Layers as Current Generator. Two-Dimensional Model, August 1995
21. V.A. Liperovsky, C.-V. Meister, K.V. Popov, S.A. Senchenkov, M.A. Oleynik and E.V. Liperovskaya: Quasi-Three Dimensional Model of Current Generation in the Ionosphere Caused by Neutral Wind Action on E_s-Clouds, August 1995

[1]Preprints can be ordered from Udo Schwarz : Zentrum für Nichtlineare Dynamik, Universität Potsdam, Am Neuen Palais, Postfach 601553, D-14415 Potsdam, Phone: (49-331)-977-1658, Fax: (49-331)-977-1142, E-Mail: USchwarz@agnld.Uni-Potsdam.de, WWW-Page: http://www.agnld.uni-potsdam.de/. Manuscripts can be also sent to him.

22. V. Dicken and P. Maaß: Wavelet-Galerkin-methods for ill-posed problems, August 1995
23. F. Feudel and N. Seehafer: Bifurcations and Pattern Formation in a 2D Navier-Stokes Fluid, August 1995
24. I.I. Blekhman, P.S. Landa and M.G. Rosenblum: Synchronization and Chaotization in Interacting Dynamical Systems, August 1995
25. N. Seehafer: Nature of the α Effect in Magnetohydrodynamics, October 1995
26. C.-V. Meister and I. Kubyshkin: Recalculation of the Diffusion Tensor for Plasmas with Ion-Acoustic Turbulence, November 1995
27. W. Jansen: CANDYS/QA: Algorithms, Programs, and User's Manual, December 1995
28. H. Voss, J. Kurths and U. Schwarz: Reconstruction of Grand Minima of Solar Activity from Δ^{14}C Data — Linear and Nonlinear Signal Analysis, January 1996
29. R. Braun and F. Feudel: Supertransient Chaos in the Two-Dimensional Complex Ginzburg-Landau Equation, March 1996
30. P. Maaß and A. Rieder: Wavelet-Accelerated Tikhonov-Phillips Regularization with Applications, June 1996
31. F. Feudel, N. Seehafer, B. Galanti and S. Rüdiger: Symmetry Breaking Bifurcations for the Magnetohydrodynamic Equations with Helical Forcing, June 1996
32. N. Seehafer, E. Zienicke and F. Feudel: Absence of Magnetohydrodynamic Activity in the Voltage-Driven Sheet Pinch, June 1996
33. W. Jansen: A Note on the Determination of the Type of Communication Areas, July 1996
34. U. Feudel: Komplexes Verhalten in multistabilen, schwach dissipativen Systemen, September 1996
35. O. Schmidtmann, F. Feudel and N. Seehafer: Nonlinear Galerkin Methods for the 3D Magnetohydrodynamic Equations, January 1997
36. A. Witt, A. Neiman, and J. Kurths: Characterizing the Dynamics of Stochastic Bistable Systems by Measures of Complexity, January 1997
37. R. Braun, F. Feudel, and N. Seehafer: Bifurcations and Chaos in an Array of Forced Vortices, March 1997
38. Ch. Böckmann, J. Biele, R. Neuber, and J. Niebsch: Retrieval of Multimodal Aerosol Size Distribution by Inversion of Multiwavelength Data, April 1997
39. S. Scheel and N. Seehafer: Bifurcation to Oscillations in Three-Dimensional Rayleigh-Bénard Convection, August 1997

Preface

The mathematical models used in natural and human sciences to describe dynamical processes often possess, despite their sometimes simply looking form, very complicate patterns of behavior. This fascinating property of dynamical systems, also called self-organization, is due to the nonlinear interactions between the system's components. The transition from one pattern to another appears suddenly if some parameter that describes the environment or some property of the system is changing. Very small changes in the vicinity of such a parameter value may cause large qualitative changes in the behavior of the system, a phenomenon which is called bifurcation. This is a problem of high relevance in many systems which are studied in physics, chemistry, biology, or ecology.

The investigation of nonlinear effects has been rapidly developed over the last two decades. One of the main aspects in the study of such phenomena is the computation of stationary states and periodic solutions, their stability with respect to small perturbations, and their dependence on some characteristic model parameters. It is important to note that, in general, nonlinear dynamical systems cannot be solved analytically. As a first powerful tool, simulations have been used to study numerically the behavior of nonlinear dynamical systems. In many applications, however, it is not sufficient to look at a single trajectory, and enormous computer time is wasted to gain a general idea of the global behavior of the system using simulation only. Moreover, this computational effort is multiplied because of the many parameter values for which the system has to be studied to get an insight into the long term limit behavior. Additionally, unstable solutions cannot be obtained with this approach.

Therefore, a lot of numerical nonsimulative methods have been developed by several authors to study bifurcation phenomena occuring in these model systems [8, 27, 29, 33, 44, 34]. The purpose of this manual is to describe the interactive software system CANDYS/QA: its facilities including theoretical and numerical background, and how to manage it. CANDYS/QA is an abbreviation of **C**omputer **A**nalysis of **N**onlinear **DY**namical **S**ystems/**Q**ualitative **A**nalysis. CANDYS/QA was originally written in the programming language *MODULA-2* and it had been implemented on the *VAX* under *VMS*. Later, using the *MODULA-2* to *C* translator EXTACY©[2] also a version in *ANSI-C* was available for workstations under UNIX. Now, this version has been revisited and reprogrammed in *C++*. The last version is described in this manual.

The heart of this report is Part II, the user's manual of CANDYS/QA. The preceding Part I gives the theoretical and algorithmic background. It can be skipped in a first reading by users who are familiar with the bifurcation theory and its algorithms, only

[2] EXTACY© is a software product of *Real Time Associates Ltd.*

the chapter about graphs should be studied. Part I is a revisited version of [11] including new developments since that publication.

Contents

Part I

Algorithms

In this part the algorithms are described that are implemented in CANDYS/QA and that are specific for the qualitative analysis of dynamical systems. It is hoped that these descriptions enforce the understanding of the capabilities of the program, its functioning, but also its failures. This part includes the necessary theoretical background from bifurcation theory.

Chapter 1

Theoretical Background

1.1 Model classes

A dynamical system is, in a strong mathematical sense, a map $\boldsymbol{f} : M \times G \to M$ where M is a nonempty set and G is a semi-group with a neutral element "0" such that

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}, 0) &= x \\ \boldsymbol{f}(\boldsymbol{x}, t+s) &= \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}, s), t) \end{aligned}$$

for all $\boldsymbol{x} \in M$ and all $s, t \in G$. The argument $t \in G$ of F is called the time variable of the dynamical system, while the argument $\boldsymbol{x} \in M$ is its space variable. In practice, the semi-group G is nothing else then one of the following sets:

1. the set of real numbers $\mathbb{R}$,
2. the set of nonnegative real numbers $\mathbb{R}_+$,
3. the set of integer numbers $\mathbb{Z}$,
4. the set of natural numbers (including 0) $\mathbb{N}$.

The first two classes are called time-continuous systems, the latter two time-discrete systems. The systems of classes 1. and 3. are invertible (the semi-group is really a group), while the systems of classes 2. and 4., in general, are not. Moreover, also the set M is often restricted to be the Euclidean space $\mathbb{R}^n$ or a Banach space B, and the map F has to satisfy specific continuity and differentiability conditions. Throughout this report we apply the following convention:

- vectors in $\mathbb{R}^n$ (n may be equal to 1) are denoted by boldface letters, e.g. $\boldsymbol{x}$;
- vector components are indicated by subscripts, e.g. x_i;
- an index for the whole vector is written as a superscript, e.g. $\boldsymbol{x}^S$.

Within that general context, CANDYS/QA treats three families of dynamical systems:

1. autonomous systems with continuous time

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}) \tag{1.1}$$

2. periodically forced systems with a period T

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p}) \tag{1.2}$$

   where $\boldsymbol{f}(t+T, \boldsymbol{x}, \boldsymbol{p}) = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p})$ holds;

3. autonomous systems with discrete time steps $k \cdot \Delta t$

$$\boldsymbol{x}' \equiv \boldsymbol{x}((k+1) \cdot \Delta t) = \boldsymbol{f}(\boldsymbol{x}(k \cdot \Delta t), \boldsymbol{p}) \tag{1.3}$$

   (after a transformation of the time variable t the time step Δt is assumed to be equal to unity).

In these formulae, $\boldsymbol{x} = (x_1, \ldots, x_n)$ denotes the $\mathbb{R}^n$-vector of the state variables, $\boldsymbol{p} = (p_1, \ldots, p_k) \in \mathbb{R}^k$ the parameter vector of the system, and $\dot{\boldsymbol{x}}$ the derivative $\frac{d\boldsymbol{x}}{dt}$. Clearly, systems of class 1 are invertible time-continuous while those of class 3 are, in general, non-invertible time-discrete systems. The periodically forced systems of class 2 are invertible time-discrete systems on the additive group of the numbers $\{0, \pm T, \pm 2T, \ldots\}$: the differential equation should be understood as an algorithmic rule to compute the map F.

We suppose throughout the report that the model satisfies the following two conditions.

- The functions f_i are continuously differentiable with respect to all variables (including the parameters), i.e. that the Jacobians

$$\boldsymbol{J} = \left(\frac{\partial f_i}{\partial x_j}\right) \qquad \text{and} \qquad \boldsymbol{J}_p = \left(\frac{\partial f_i}{\partial p_j}\right) \tag{1.4}$$

  exist. Moreover, for treating bifurcation analysis we assume that the functions are at least three or even four times continuously differentiable.

- The model does not possess symmetries or first integrals such as Hamiltonian systems do. If it did then its bifurcations had a higher "co-dimension" than parameters are variable (c.f. Sec. 1.4 for the definitions). The algorithms of CANDYS/QA do not work in this case, one has to use specifically developed algorithms (see [14] and [16] for an overview).

1.2 Qualitative properties as properties of invariant sets

After defining the term "dynamical system" in Sec. 1.1, it is time to define the term "qualitative property". We will consider system properties which do not depend on the investigator's point of view, especially, which do not depend on the coordinate system. This means, we look for properties which are invariant with respect to coordinate transformations.

The state space M of the dynamical system can be transformed one-to-one in different ways. One can demand that the transformations possess additional properties, e.g. to be continuous or differentiable. All transformations with specific properties constitute a group if also the inverse transformation and the concatenation of transformations have these properties. Prominent examples are the group of all continuous transformations as well as the group of differentiable transformations such that

$$\left|\det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}\right)\right| > \varepsilon > 0. \tag{1.5}$$

We say, a property of the dynamical system is a qualitative property if it is maintained under all transformations of a group. Thus, the term "qualitative" characterizes a property that does not depend on arbitrary model features as the chosen coordinate system. Whether a property is a qualitative one depends on the group of transformations in question. As an example, the eigenvalues λ_i of the matrix $\boldsymbol{J}$ computed at a steady-state $\boldsymbol{x}^*$ are invariant with respect to the differentiable transformations but they are not invariant with respect to arbitrary continuous transformations. In this report we will always suppose that the transformation group defining the qualitative properties is the group of continuous transformations. But the group of differentiable transformation is an important subgroup, and when discussing normal forms, firstly, we apply a transformation out of this subgroup and, secondly, a continuous transformation.

Another basic idea of qualitative analysis is the investigation of time-invariant sets $\mathcal{S}$ in the state space $\mathbb{R}^n$ which are mapped onto themselves via evolution equations (1.1)–(1.3) for all times and which are non-wandering in the following sense: to each point $\boldsymbol{x} \in \mathcal{S}$ and each neighborhood $U(\boldsymbol{x})$ exists a $T > 0$ such that $U(\boldsymbol{x}) \cap U(\boldsymbol{x}(t)) \neq \emptyset$ for all $t > T$, i.e. some kind of recurrence holds on non-wandering sets. These invariant sets remain invariant sets if the space of the state variables is transformed. Therefore, their existence and count are typical qualitative features of the system.

An important point is the question of stability of these invariant sets with respect to small perturbations of the state variables. An invariant set $\mathcal{S}$ is said to be stable if any solution $\boldsymbol{x}(t)$ of equations (1.1)–(1.3) starting nearby $\mathcal{S}$ remains close to $\mathcal{S}$ for all time. If, in addition, a neighborhood U can be chosen such that any solution $\boldsymbol{x}(t)$ starting in U approaches $\mathcal{S}$ as $t \to \infty$ then $\mathcal{S}$ is said to be asymptotically stable or attractive. On the other hand, if there is at least one trajectory leaving any neighborhood of $\mathcal{S}$ then the invariant set is unstable. The Jacobian matrix $\boldsymbol{J}$ and the fundamental matrix $\boldsymbol{H}(\boldsymbol{x}, t) = \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{x}(0)}$ play the essential role for the evaluation of the stability. The second matrix, $\boldsymbol{H}$, is defined by the matrix differential equation

$$\frac{d}{dt}\boldsymbol{H} = \boldsymbol{J}(\boldsymbol{x}(t))\,\boldsymbol{H} \qquad \text{with} \qquad \boldsymbol{H}(\boldsymbol{x}, 0) = \boldsymbol{I} \tag{1.6}$$

or the matrix difference equation

$$\boldsymbol{H}' = \boldsymbol{J}(\boldsymbol{x}(t))\,\boldsymbol{H} \qquad \text{with} \qquad \boldsymbol{H}(\boldsymbol{x}, 0) = \boldsymbol{I} \tag{1.7}$$

computed along the solutions of Eqs. 1.1, (1.2), or (1.3), respectively. Below we will need the derivation $\boldsymbol{H}_p(\boldsymbol{x}, t)$ of $\boldsymbol{x}(t)$ with respect to parameters. It is given by

$$\frac{d}{dt}\boldsymbol{H}_p = \boldsymbol{J}(\boldsymbol{x}(t))\boldsymbol{H}_p + \boldsymbol{J}_p(\boldsymbol{x}(t)) \qquad \text{with} \qquad \boldsymbol{H}_p(\boldsymbol{x}, 0) = \boldsymbol{0} \tag{1.8}$$

or

$$\boldsymbol{H}_p' = \boldsymbol{J}(\boldsymbol{x}(t))\boldsymbol{H}_p + \boldsymbol{J}_p(\boldsymbol{x}(t)) \qquad \text{with} \qquad \boldsymbol{H}_p(\boldsymbol{x},0) = \boldsymbol{0}. \tag{1.9}$$

Several kinds of invariant sets are known for nonlinear dynamical systems, e.g. steady-states, periodic solutions, invariant tori, and strange attractors. According to their importance in practice and numerical tractability only two of them are taken into account in the software system CANDYS/QA.

Steady-states (also called stationary points, rest points, or fixed points) of an autonomous system are invariant sets that consist of one point only. The number of steady-states is a qualitative property of the system. From the definition of a steady-state the equations of its computation immediately follows:

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{p}) = \boldsymbol{0}\,. \tag{1.10}$$

A steady-state is attractive if all eigenvalues of the corresponding Jacobian matrix have negative real parts, and it is unstable if at least eigenvalue has positive real part.

Cycles (also called periodic solutions, periodic motions) of any system family:

$$\boldsymbol{x}(T) = \boldsymbol{x}(0) \qquad \text{for some } T > 0.$$

This invariant set consists of all points in the state space covered by the solution $\boldsymbol{x}(t)$ in the time interval $[t, t+T]$.

In the case of time discrete systems, where $\boldsymbol{x}(T)$ corresponds to the k-th successor of $\boldsymbol{x}(0)$ (i.e. $T = k$) one point of this set can be obtained as the solution of the nonlinear equations system

$$\boldsymbol{f}^k(\boldsymbol{x},\boldsymbol{p}) - \boldsymbol{x} = \boldsymbol{0} \tag{1.11}$$

($\boldsymbol{f}^k$ denotes the k-th iterate of $\boldsymbol{f}$).

In the case of periodically forced systems cycles with the period T of the driving force (or multiples of it) are determined. The system of differential equations (1.2) has to be solved with the boundary condition $\boldsymbol{x}(0) = \boldsymbol{x}(kT)$. The nonlinear equations system

$$\boldsymbol{x}(kT) - \boldsymbol{x}(0) = \boldsymbol{0} \tag{1.12}$$

results. Herein $\boldsymbol{x}(kT)$ is the solution of Eq. (1.2) starting at $\boldsymbol{x}(0)$.

The most complicated case is the computation of cycles of the system family (1.1) because the unknown period has to be determined, too. Using Poincaré's first return map the search for cycles is transformed into the search for fixed points of a discrete map. This case will be discussed in more detail in Sec. 2.1.

The solution of the Eqs. (1.6), (1.7) (integrated from $t = 0$ to $t = T$ or iterated k-times) yields the matrix whose eigenvalues determine the stability of the cycle. A periodic motion is attractive if all eigenvalues lie within the unit circle and it is unstable if at least one eigenvalue lies outside.

At this place one important remark has to be made. Although the qualitative properties abstract from a concrete coordinate system all numerical computations are done in the framework of a given coordinate system. Equations (1.10)–(1.11) express this fact. This way, a couple of quantitative information is gained, and the qualitative properties have to be extracted from them.

1.3 Local qualitative properties

Besides the invariance of a property with respect to transformations of the whole state space $\mathbb{R}^n$, often only the local invariance is of interest. This means the following: given an invariant set $\mathcal{S}$, a property of $\mathcal{S}$ is called locally qualitative if a neighborhood U of $\mathcal{S}$ exists such that for all admissible transformations φ of U the set $\varphi(\mathcal{S})$ has the same property within $\varphi(U)$. For instance, suppose $\mathcal{S}$ is an attractive stable steady-state $\boldsymbol{x}^*$ and U is a neighborhood of $\boldsymbol{x}^*$ such that $\boldsymbol{x}(t) \to \boldsymbol{x}^*$ if $\boldsymbol{x} \in U$. Then for all continuous transformations of U holds $\boldsymbol{y}(t) \equiv \varphi(\boldsymbol{x}(t)) \to \varphi(\boldsymbol{x}^*)$ if $\boldsymbol{y} \in \varphi(U)$, i.e. the attractive stability of $\boldsymbol{x}^*$ is a locally qualitative property of that point. As in the global case, the local qualitativeness of a property depends on the group of admissible transformations.

The numerical methods implemented in CANDYS/QA always compute a single invariant set and its stability. This means, that only local qualitative properties of the dynamical systems can be determined.

The restriction to local qualitative properties has the advantage that, theoretically, transformations to formally simple systems are known. In the remainder of this section, the simplified but, nevertheless, equivalent systems shall be presented. First, we consider steady-states. If n_-, n_0, n_+ eigenvalues of $\boldsymbol{J}$ have negative, zero, or positive real part, respectively, then for almost all systems a differentiable transformation φ exists to the system

$$\left.\begin{array}{rcll} \dot{\boldsymbol{w}} & = & \tilde{\boldsymbol{f}}(\boldsymbol{w}) & , \quad \boldsymbol{w} \in \mathcal{C} \\ \dot{\boldsymbol{y}} & = & \boldsymbol{J}_- \boldsymbol{y} & , \quad \boldsymbol{y} \in \mathbb{R}^{n_-} \\ \dot{\boldsymbol{z}} & = & \boldsymbol{J}_+ \boldsymbol{z} & , \quad \boldsymbol{z} \in \mathbb{R}^{n_+} \end{array}\right\} \tag{1.13}$$

Here, $\boldsymbol{J}_-$ and $\boldsymbol{J}_+$ are constant quadratic matrices possessing the eigenvalues of $\boldsymbol{J}$ with negative or positive real part, respectively. The set $\mathcal{C}$ is an n_0-dimensional manifold in $\varphi(U)$ with $\varphi(\boldsymbol{x}^*) \in \mathcal{C}$ and tangential to the eigenspace of the zero eigenvalues, the so-called center manifold (c.f.[3, 19, 4]). The subsystems in $\mathbb{R}^{n_-}$, $\mathbb{R}^{n_0}$, $\mathbb{R}^{n_+}$ can be simplified such that $\boldsymbol{J}_-$ and $\boldsymbol{J}_+$ have real Jordan normal form, and such that many coefficients of the Taylor expansion of $\tilde{\boldsymbol{f}}(\boldsymbol{w})$ vanish (so-called normal form of $\tilde{\boldsymbol{f}}(\boldsymbol{w})$). Moreover, if also continuous transformations are admissible then a transformation to (1.13) is always possible with $\boldsymbol{J}_- = -\boldsymbol{I}$, $\boldsymbol{J}_+ = \boldsymbol{I}$.

Although the transformation to normal form is only theoretically known it has the following meaning.

- The numbers n_-, n_0, and n_+ are local qualitative properties. If $n_0 = 0$ then no others exist, if $n_0 > 0$ then the qualitative properties of $\tilde{\boldsymbol{f}}(\boldsymbol{w})$ are also those of $f(\boldsymbol{x})$.

- The interesting features of the dynamics are concentrated in the low-dimensional center manifold, and for a deeper insight into the dynamics one has to study only the function $\tilde{\boldsymbol{f}}(\boldsymbol{w})$.
- The numerical procedures that determine steady-states with eigenvalues on the imaginary axis are constructed such that they implicitly also determine the transformation to the normal form.

In the case of a cycle a similar theorem holds. One has to replace $\boldsymbol{J}$ by $\boldsymbol{H}$ as well as negative, zero, and positive real parts by modulus smaller, equal, and larger than 1, respectively, to determine the dimensions n_-, n_0, and n_+.

$$\left.\begin{array}{rcll} \dot{\boldsymbol{w}} & = & \tilde{\boldsymbol{f}}(\boldsymbol{w}) & , \quad \boldsymbol{w} \in \mathcal{C} \\ \dot{\boldsymbol{y}} & = & \boldsymbol{H}_- \boldsymbol{y} & , \quad \boldsymbol{y} \in \mathbb{R}^{n-} \\ \dot{\boldsymbol{z}} & = & \boldsymbol{H}_+ \boldsymbol{z} & , \quad \boldsymbol{z} \in \mathbb{R}^{n+} \end{array}\right\} \tag{1.14}$$

If continuous transformations are admissible then $\boldsymbol{H}_-$ and $\boldsymbol{H}_+$ can be chosen as diagonal matrices with diagonal elements $\pm\frac{1}{2}$ and ± 2, respectively, where at most one minus sign occurs in each of both matrices. After that transformation, the $\boldsymbol{y}$ and $\boldsymbol{z}$ parts have their simplest form. This means, there are $4n$ classes of qualitatively different cycles if $n_0 = 0$.

1.4 Parameter dependency – bifurcations

1.4.1 Overview

In this section we suppose that the dynamical system depends on several parameters (as mentioned in Sec. 1.1) and that for one parameter vector $\boldsymbol{p}^0$ an invariant set $\boldsymbol{x}^0$ (i.e. a steady-state or a point of a cycle) is known together with its Jacobian $\boldsymbol{J}$ or $\boldsymbol{H}$. The question is, how do the invariant set and its local qualitative properties change when the parameter vector is varied.

The answer is simple if no eigenvalue of $\boldsymbol{J}$ or $\boldsymbol{H}$ belongs to the imaginary axis or to the unit circle, respectively. In this case the Implicit function theorem states the following.

1. There is a neighborhood $U_x \times U_p$ of $(\boldsymbol{x}^0, \boldsymbol{p}^0)$ in $\mathbb{R}^n \times \mathbb{R}^k$ where for each $\boldsymbol{p} \in U_p$ exactly one steady-state or cycle point $\boldsymbol{x}(\boldsymbol{p})$ exists. $\boldsymbol{x}(\boldsymbol{p})$ depends differentiably on $\boldsymbol{p}$.
2. In this neighborhood the Jacobians $\boldsymbol{J}(\boldsymbol{x}(\boldsymbol{p}))$ or $\boldsymbol{H}(\boldsymbol{x}(\boldsymbol{p}))$ and their eigenvalues depend continuously on $\boldsymbol{p}$, i.e. if U_p is chosen sufficiently small then no eigenvalues belong to the imaginary axis or to the unit circle, respectively.

This means, in the neighborhood $U_x \times U_p$ all systems have the same local qualitative properties (namely the numbers n_- and n_+).

We call a point $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ a local bifurcation point if it is a steady-state or a cycle point such that in any parameter neighborhood systems exist with different local qualitative

properties. The observation above says, that at a local bifurcation point at least one eigenvalue of $\boldsymbol{J}$ or $\boldsymbol{H}$ must belong to the imaginary axis or to the unit circle.

The qualitative properties of the bifurcation point are those of the reduced system $\tilde{\boldsymbol{f}}(\boldsymbol{w})$ on the center manifold (including the dimensions n_-, n_0, n_+). As we will see, these properties are described by specific nonlinear equations and some inequalities (so-called regularity conditions). If there are s such equations then one says that the bifurcation point has co-dimension s. If $k > s$ and if both, the equations and the inequalities, are satisfied at $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ then there is a $(k-s)$-dimensional manifold in parameter space consisting of equivalent bifurcation points, and any parameter neighborhood of $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ contains points (steady-states or cycle points) with eigenvalues off the imaginary axis or the unit circle. Moreover, there is a neighborhood of that point such that the only bifurcation points of co-dimension $\geq s$ are those in the submanifold mentioned. But this neighborhood contains also bifurcations of co-dimension $< s$, namely those that are described by a subset of the equations characterizing the co-dimension s bifurcation.

The definition of a bifurcation point has an important consequence. We consider any continuous one-to-one transformation of the state-parameter space $\mathbb{R}^n \times \mathbb{R}^k$ (or a transformation of a neighborhood U of $(\boldsymbol{x}^*, \boldsymbol{p}^*)$) such that $\tilde{\boldsymbol{p}} \equiv \varphi(\boldsymbol{p})$ does not depend on $\boldsymbol{x}$. [2] The neighborhood $\tilde{U} = \varphi(U)$ contains $\varphi(\boldsymbol{x}^*, \boldsymbol{p}^*)$ and $\varphi(\boldsymbol{x}, \boldsymbol{p})$, i.e. points with different qualitative properties, provided $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ and $(\boldsymbol{x}, \boldsymbol{p}) \in U$ have different qualitative properties. This means, that the property of $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ to be a bifurcation point is invariant under those transformations. Thus, it is meaningful to look for a simple form of the dynamical system. First of all, Eq. (1.13) can be generalized to systems with variable parameters (c.f. [3]):

$$\begin{aligned} \dot{\boldsymbol{w}} &= \tilde{\boldsymbol{f}}(\boldsymbol{w}, \boldsymbol{p}) \quad , \quad \boldsymbol{w} \in \mathbb{R}^{n_0} \times \mathbb{R}^k \\ \dot{\boldsymbol{y}} &= \boldsymbol{J}_- \boldsymbol{y} \quad , \quad \boldsymbol{y} \in \mathbb{R}^{n-} \\ \dot{\boldsymbol{z}} &= \boldsymbol{J}_+ \boldsymbol{z} \quad , \quad \boldsymbol{z} \in \mathbb{R}^{n+} \end{aligned} \tag{1.15}$$

where $\tilde{\boldsymbol{f}}(\boldsymbol{w}, \boldsymbol{p})$ extends the function $\tilde{\boldsymbol{f}}(\boldsymbol{w})$ in Eq. (1.13). Next, one can try to find a simple form of $\tilde{\boldsymbol{f}}(\boldsymbol{w}, \boldsymbol{p})$, especially one looks for a function that depends effectively only on s parameters in the case of a co-dimension s bifurcation. The resulting function $\tilde{\boldsymbol{f}}(\boldsymbol{w}, \boldsymbol{p})$ is then called a versal unfolding.

An analogous generalization of Eq. (1.14) is valid in the case of cycles. But often versal unfoldings exist only in the following sense: the map is approximated by differential equations such that the cycles become steady-states. The qualitative properties of the latter can be interpreted as qualitative properties of the cycle but many details of the original system are lost.

The hierarchy of bifurcation points of different co-dimension suggests a scheme of the numerical bifurcation analysis. One starts at a parameter vector $\boldsymbol{p}^0$ and determines a steady-state or cycle, we can assume that it is not a bifurcation point. Then the parameter $\boldsymbol{p}$ is changed by varying only one component, i.e. the effective parameter space is 1-dimensional. When one eigenvalue (or a pair of conjugate complex eigenvalues) crosses the imaginary axis or the unit circle then a bifurcation point is found, we can assume that it is of co-dimension 1. Here, one starts the variation of two components of

[2] If $\tilde{\boldsymbol{p}}$ depended on $\boldsymbol{x}$ then $\tilde{\boldsymbol{p}}$ would also depend on the time in contradiction to the parameter character of $\tilde{\boldsymbol{p}}$.

$\boldsymbol{p}$ and computes the curve of these bifurcation points. We proceed inductively: when s components of the parameter vector are varied then curves of bifurcation points of co-dimension $s-1$ are computed and bifurcations of co-dimension s can be determined (if any).

1.4.2 Co-dimension-1 bifurcations

In this subsection we explain the co-dimension-1 bifurcations treated by CANDYS/QA. Figure 1.1 shows a synopsis of them. For each bifurcation the following will be listed:

1. the versal unfolding;
2. phenomena accompanied with it, e.g. curves branching off;
3. the equality conditions on the n-dimensional system, but without mentioning the conditions on higher order derivatives.

The general properties will be explained as properties of the versal unfoldings (if such an unfolding is known). Especially the phrase that "points are stable" has to be understood as "stable in the center manifold", but they are actually unstable if $n_+ > 0$.

In the case that a bifurcation is relevant for cycles only, we list conditions on the eigenvalues of $\boldsymbol{H}$, otherwise those of $\boldsymbol{J}$ for the steady-states. If a bifurcation is relevant for cycles as well then the conditions are easily translated. In the second case we give the versal unfolding for steady-states only.

Turning points (return points, saddle-node-bifurcations)

A turning point is a bifurcation of steady-states and cycles. It has a 1-dimensional center manifold, the versal unfolding is:

$$\dot{x} = x^2 + p. \tag{1.16}$$

The bifurcation point is $(x, p) = (0, 0)$. This means, that at the turning point the curve is locally a parabola with the axis parallel to the parameter axis. The curve changes its direction at the bifurcation point: this explains the term "turning point" or "return point". Moreover, since the single eigenvalue crosses 0, the points on one branch of the parabola are stable while the points on the other branch are unstable. If the system is in reality n-dimensional with the dimensions $n_- = n-1$, $n_+ = 0$ then points at one branch form "nodes" and the others form "saddles", what is reflected by the often used term "saddle-node-bifurcation".

An n-dimensional system can be transformed locally about $(\boldsymbol{x}^*, p^*)$ to Eq. (1.16) if $\boldsymbol{J}$ has no other zero-eigenvalue and if $\frac{\partial \lambda}{\partial p} \neq 0$. The last condition is equivalent to $\operatorname{rank}(\boldsymbol{J}, \boldsymbol{J}_p) = n$.

Hopf bifurcations

The Hopf bifurcation is unique to the steady-states of autonomous differential equations. At a Hopf bifurcation two complex conjugate eigenvalues of $\boldsymbol{J}$ cross the imaginary axis. The center manifold is 2-dimensional and the versal unfolding in a complex state variable z is:

$$\dot{z} = z(i + p \pm z\bar{z}) \tag{1.17}$$

(c.f. [3, 19, 20]). This type of bifurcation is named after E. Hopf who described it first in [22] (an English translation can be found in [31]).

The bifurcation point is $(z, p) = (0, 0)$. At this point two different curves meet: a curve of steady-states which change the stability at the bifurcation, and a curve of cycles whose radii vanish at the bifurcation. The cycles form a paraboloid in the z-p space and the steady-states constitute its axis. The sign of $z\bar{z}$ determines the stability of the cycles: "+" means that the cycles are unstable and exist for $p < 0$ (so-called subcritical bifurcation), otherwise the cycles are stable and exist for $p > 0$ (so-called supercritical bifurcation).

An n-dimensional system can be transformed locally about $(\boldsymbol{x}^*, p^*)$ to Eq. (1.17) if $\boldsymbol{J}$ has no other eigenvalue on the imaginary axis and if $\frac{\partial \Re\lambda}{\partial p} \neq 0$.

Torus bifurcations

This type of bifurcation is the analogue to the Hopf bifurcation in case of cycles of maps. It is characterized by the fact that two conjugate complex eigenvalues cross the unit circle. Analogously to the Hopf bifurcation, the cycle changes stability at the bifurcation point and an invariant set (topologically a circle in $\mathbb{R}^2$) is born where also subcritical and supercritical versions exist.

In the case of cycles of autonomous differential equations this circle is a set in the Poincaré surface $\mathcal{S}$ (c.f. Sec. 2.1) and it is overlaid by the trajectories crossing $\mathcal{S}$. This results in an invariant torus and explains the name of the bifurcation. Sometimes, this bifurcation is called Hopf bifurcation, too.

The character of the motion on this invariant circle depends strongly on the critical eigenvalues $e^{\pi i\alpha}$: if $\alpha = \frac{r}{s}$ is rational (a resonance situation) then there are periodic points on that circle (s saddles and s nodes if $s > 4$), otherwise the circle is rotated without periodic points.

Period doublings (flip bifurcations)

This type is unique to cycles. It is characterized by the outstanding case that one real eigenvalue of $\boldsymbol{H}$ crosses the unit circle at -1. The center manifold is 1-dimensional, and the versal unfolding is:

$$x' = (p - 1)x \pm x^3 \tag{1.18}$$

The cycle changes its stability at the bifurcation point and a cycle of doubled period length is born. The points of these cycles form a parabola where the cycles with simple

length constitute the axis. The points of the cycles of doubled length visit the two branches of the parabola alternately. Analogously to the Hopf bifurcation, subcritical and supercritical period doublings are possible.

An n-dimensional system can be transformed locally about $(\boldsymbol{x}^*, p^*)$ to Eq. (1.18) if $\boldsymbol{H}$ has no other eigenvalue of modulus 1 and if $\frac{\partial|\lambda|}{\partial p} \neq 0$.

Simple bifurcations

A simple bifurcation occurs when two curves of steady-states or cycles intersect. This type of bifurcation is exceptional for two reasons:

- it is actually a co-dimension-2 bifurcation that often occurs in 1-parameter problems;
- it is not invariant with respect to transformations of the parameter space, instead, one parameter has to be distinguished as the leading one.

In 1-parameter space two (not versal) unfoldings exist on a 1-dimensional center manifold:

$$\dot{x} = xp \pm x^3 \tag{1.19}$$

the so-called pitchfork bifurcation (the "$+$" sign corresponds to a subcritical, the "$-$" sign to a supercritical bifurcation),

$$\dot{x} = xp + x^2 \tag{1.20}$$

the so-called transcritical bifurcation. As in the case of a turning point, at the bifurcation one real eigenvalue crosses the imaginary axis. But now $\mathrm{rank}(\boldsymbol{J}) = \mathrm{rank}(\boldsymbol{J}, \boldsymbol{J}_p) = n - 1$ holds.

One steady-state is $x(p) = 0$. When p passes 0 the eigenvalue changes its sign, i.e. the stability changes. In the transcritical case the other steady-state is $x(p) = -p$ and it changes also the stability such that it is unstable if the first is stable. In the pitchfork case the second curve is a parabola symmetrical to the parameter axis. Both branches of parabola are stable if the coexisting steady-state is unstable (and vice versa).

1.4.3 Co-dimension-2 bifurcations

In this subsection we consider the co-dimension-2 bifurcations treated by CANDYS/QA. We follow the lines of the previous subsection.

Cusp points

The cusp is a degenerate turning point where $\boldsymbol{J}$ has the simple eigenvalue 0 but the transformation to the system Eq. (1.16) does not work since an inequality condition on

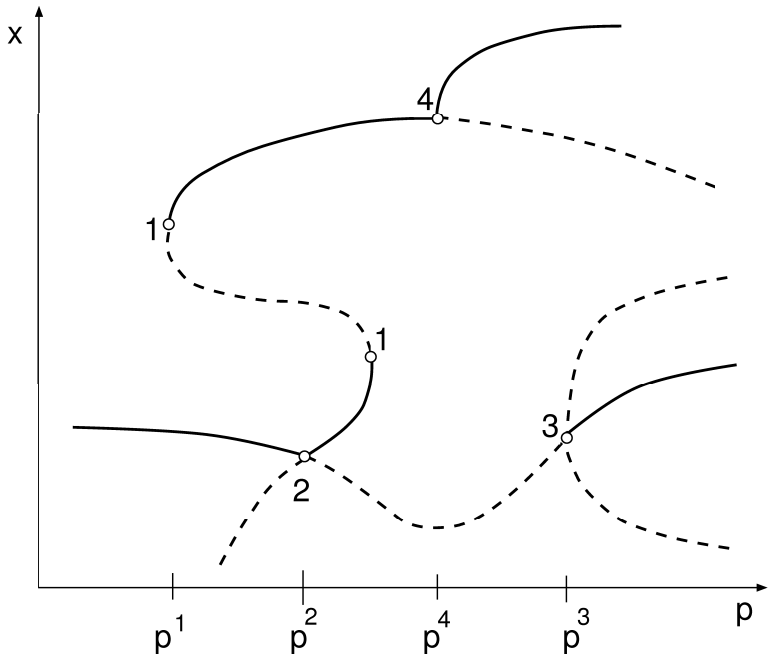


Figure 1.1: Scheme of bifurcation situations:

1 : turning point of steady-states or cycles;
2 : transcritical bifurcation point of steady-states or cycles;
3 : subcritical pitchfork bifurcation point of steady-states or cycles;
4 : supercritical Hopf bifurcation (steady states), period doubling, or torus bifurcation (cycles).

Solid lines: stable invariant sets; dashed lines: unstable invariant sets.

higher order derivatives is not satisfied, here it becomes an equality. Thus, this equality gives an additional equations condition (c.f. Sec. 2.4.2 for the formula). Provided that certain higher order derivatives do not vanish the versal unfolding on the center manifold is

$$\dot{x} = p_1 + p_2 x \pm x^3 \tag{1.21}$$

Two branches of turning point curves meet at $(x, p_1, p_2) = (0, 0, 0)$ such that their projections on the p_1-p_2-parameter plane have the same tangent and both curves approach the bifurcation point from the same direction forming a "cusp". It should be mentioned that the curve is smooth in the state-parameter space.

Degenerate period doublings

The transformation to system (1.18) does not work if certain higher order derivatives vanish (c.f. Sec. 2.4.2), thus, the period doubling is called degenerate.

At such a point on a period doublings curve the following happens: a subcritical period doubling becomes supercritical (or vice versa), and a curve of turning points of

cycles of the doubled period starts here. In the p_1-p_2-plane the last curve is tangent to the period doubling curve.

Takens-Bogdanov points

The Takens-Bogdanov point is characterized by a double zero-eigenvalue such that the matrix $\boldsymbol{J}$ possesses a Jordan block $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$. It was studied first by Bogdanov [5, 6] and Takens [47]. One possible versal unfolding on the 2-dimensional center manifold is the following:

$$\begin{aligned} \dot{x}_1 &= x_2 + p_2 x_1 + x_1^2 \\ \dot{x}_2 &= p_1 \pm x_1^2 \end{aligned} \tag{1.22}$$

The bifurcation point is $(x_1, x_2, p_1, p_2) = (0, 0, 0, 0)$. The straight line $(0, 0, 0, p_2)$ is a curve of turning points while for $p_1 \geq 0$ the parabola $(x_1^H, x_2^H, p_1, -2\sqrt{p_1})$ defines a curve of Hopf bifurcations (with certain x_1^H and x_2^H) that is tangent to the line of turning points in the p_1-p_2-plane. The other branch of the parabola defines a curve of points with real eigenvalues $\pm\sqrt[4]{p_1}$ (i.e. an imaginary Hopf bifurcation in the sense of Sec. 2.4.2).

Two eigenvalues –1

This bifurcation occurs if $\boldsymbol{H}$ possesses a Jordan block $\begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}$. A curve of period doublings crosses a curve of torus bifurcations (where on one side of the period doubling curve the torus bifurcation curve consists of imaginary torus bifurcations in the sense of Sec. 2.4.2). Depending on the sign of higher order derivatives a further curve starts here: a curve of torus bifurcations of the cycles of doubled period.

Constant rotation

The torus bifurcations with the critical eigenvalue $\lambda = e^{\pi i \alpha}$ with $\alpha \in (0, 1)$ rational and fixed possess different qualitative properties for different α (so-called Arnold tongues occur, c.f. [3]). For a given α the condition that $\boldsymbol{H}$ possesses an eigenvalue $re^{\pi i \omega}$ with $r = 1$ and $\alpha = \omega$ defines a co-dimension-2 bifurcation.

Combined eigenvalues

Two different co-dimension-1 bifurcations form a co-dimension-2 bifurcation if the characterizing conditions of both are satisfied at the same time. In the case of steady-states a pair of imaginary eigenvalues can be combined with the eigenvalue 0 or with another pair of imaginary eigenvalues. In the case of cycles combinations of the eigenvalues $+1$, -1, or complex ones on the unit circle are possible. At the co-dimension-2 bifurcation point the curves of the corresponding co-dimension-1 bifurcations intersect (or are tangent in the p_1-p_2-plane). Further curves can start here. For instance, if

two different imaginary pairs of eigenvalues are combined then there start two curves of cycles with different torus bifurcations.

1.4.4 Other special points

The program CANDYS/QA is devoted to the numerical analysis of qualitative properties of a dynamical system. But a small modification of the algorithms allows to compute also some quantitative features in combination with qualitative ones: steady-states, cycles, or bifurcation points where at the same time a user-specified function $Q(\boldsymbol{x}, \boldsymbol{p})$ assumes a pre-described fixed value or assumes an extremum. Such a point is treated like a bifurcation point, e.g. it can be the object of path-following. Since proper bifurcation points and the points described here are not distinguished in many contexts of this report we call them commonly special points.

The function Q is evaluated at the steady-state or at one point of the cycle. The averaging along the cycle has not yet been implemented.

The computation of extrema concerns the extrema of $Q(\boldsymbol{x}, \boldsymbol{p})$ along a curve of steady-states etc., i.e. extrema with equality constraints. This means, that actually the Lagrange function

$$\Phi(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{v}) = Q(\boldsymbol{x}, \boldsymbol{p}) + \boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) \tag{1.23}$$

is minimized or maximized where $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p})$ is the equation for the steady-state etc., and $\boldsymbol{v}$ is the vector of Lagrange multipliers.

Chapter 2

Numerical Procedures

2.1 The Poincaré map

2.1.1 Construction of the map

Let us assume that an autonomous nonlinear system possesses a cycle as one trajectory. The computation of such cycles is based on Poincaré's first return map. For explanation of this map we suppose the existence of a smooth $(n-1)$-dimensional submanifold $\mathcal{S}$ in $\mathbb{R}^n$ defined by the equation

$$g(\boldsymbol{x}, \boldsymbol{p}) = 0$$

($g : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ a smooth function). $\mathcal{S}$ has to be chosen in such a way that the trajectories are transverse to it (c.f. [19]). This means, that at the intersection point $\boldsymbol{x}^*$ of the periodic trajectory with $\mathcal{S}$

$$\left(\left. \frac{\partial g}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}^*} \right)^T \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{p}) \neq 0 \tag{2.1}$$

holds.

Any nearby noncyclic trajectory crosses the submanifold $\mathcal{S}$ in two different points $\boldsymbol{x}^S$ (start), and $\boldsymbol{x}^E$ (end) which are close to the intersection point $\boldsymbol{x}^*$ of the cycle (c.f. Fig. 2.1). The intersection points define a map $P : \mathcal{S} \to \mathcal{S} : \boldsymbol{x}^S \mapsto \boldsymbol{x}^E$ (at least in a neighborhood of $\boldsymbol{x}^*$), called the Poincaré map, which in turn is a discrete dynamical system on $\mathcal{S}$. Hence, a cycle corresponds to a fixed point of that map: $\boldsymbol{x}^E = \boldsymbol{x}^S$. Let T be the time difference between the two points $\boldsymbol{x}^S$ and $\boldsymbol{x}^E$. If we denote in the following $\boldsymbol{x}^S$ by $\boldsymbol{x} = \boldsymbol{x}(0)$ and $\boldsymbol{x}^E = \boldsymbol{x}(T)$ then the cycle is characterized by $\boldsymbol{x}(T) - \boldsymbol{x} = \boldsymbol{0}$. But all points of the cycle fulfill that equation. One single point of the cycle is determined by the additional condition that it belongs to the submanifold $\mathcal{S}$; and we obtain the following equations system

$$\begin{aligned} \boldsymbol{x}(T) - \boldsymbol{x} &= \boldsymbol{0} \\ g(\boldsymbol{x}, \boldsymbol{p}) &= 0. \end{aligned} \tag{2.2}$$

Equation (2.2) constitutes a nonlinear system of $n+1$ equations for the $n+1$ unknowns $\boldsymbol{x}$ and T: the solution is a point of the cycle on $\mathcal{S}$ and its period length. This nonlinear equations system which describes the Poincaré map can be solved by Newton's

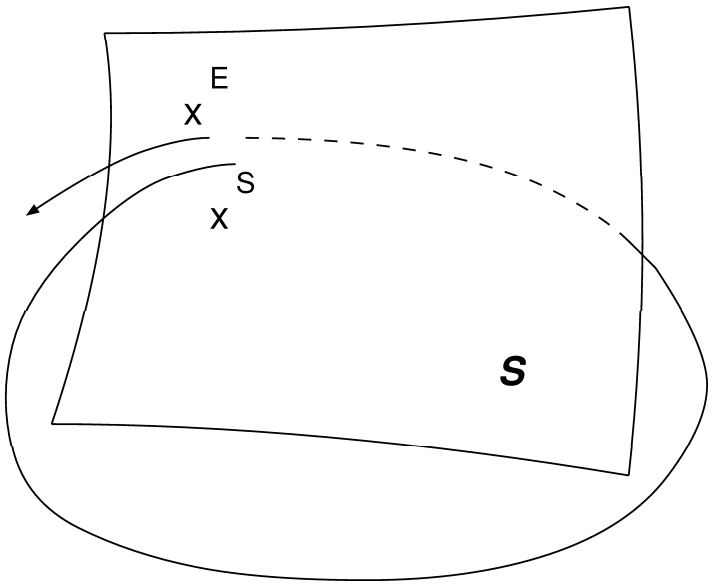


Figure 2.1: The Poincaré map.

method. Equation (2.2) possesses at least two (in general, an even number of) solutions since the cycle crosses the manifold once in "forward" and once in "backward" direction. But under the transversality condition (2.1) the solutions are isolated. The resulting Jacobian is

$$\boldsymbol{J}_P = \begin{pmatrix} \boldsymbol{H} - \boldsymbol{I} & \boldsymbol{f} \\ g_{\boldsymbol{x}} & 0 \end{pmatrix} \tag{2.3}$$

where $\boldsymbol{H}$ is defined by Eq. (1.6), $g_{\boldsymbol{x}}$ is the row vector $\frac{\partial g}{\partial \boldsymbol{x}}$, and where the vectors and matrices are computed at the point $(\boldsymbol{x}, T)$.

We show the regularity of $\boldsymbol{J}_P$ by calculating its eigenvalues and eigenvectors using those of $\boldsymbol{H}$. Suppose that $\boldsymbol{H}$ possesses at the solution vector $(\boldsymbol{x}^*, T^*)$ the eigenvalues $\lambda_1, \ldots, \lambda_n$ with the right eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ and the left eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$.[2] It is well known that one eigenvalue (say λ_n) is equal to 1 and its right eigenvector is $\boldsymbol{u}_n = \boldsymbol{f}(\boldsymbol{x}^*)$. For $i < n$

$$(\boldsymbol{v}_i, 0) \begin{pmatrix} \boldsymbol{H} - \boldsymbol{I} & \boldsymbol{f} \\ g_{\boldsymbol{x}} & 0 \end{pmatrix} = ((\lambda_i - 1)\boldsymbol{v}_i, 0)$$

holds, i.e. $\boldsymbol{J}_P$ possesses the eigenvalues $\lambda_i - 1$ for $i = 1, \ldots, n-1$ (recall $\boldsymbol{v}_i^T \boldsymbol{u}_n = 0$ for the different eigenvalues λ_i and λ_n). For $i = n$ we use the ansatz $\begin{pmatrix} \boldsymbol{f} \\ \alpha \end{pmatrix}$ with a complex unknown α as right eigenvector and μ as eigenvalue:

$$\mu \begin{pmatrix} \boldsymbol{f} \\ \alpha \end{pmatrix} = \begin{pmatrix} \boldsymbol{H} - \boldsymbol{I} & \boldsymbol{f} \\ g_{\boldsymbol{x}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{f} \\ \alpha \end{pmatrix} = \begin{pmatrix} (\lambda_n - 1)\boldsymbol{f} + \alpha \boldsymbol{f} \\ g_{\boldsymbol{x}} \boldsymbol{f} \end{pmatrix} = \begin{pmatrix} \alpha \boldsymbol{f} \\ g_{\boldsymbol{x}} \boldsymbol{f} \end{pmatrix}.$$

[2] For sake of simplicity we assume here simple eigenvalues.

This yields $\mu = \alpha$ and $\mu\alpha = g_{\boldsymbol{x}}\boldsymbol{f}$, i.e.

$$\alpha = \pm\sqrt{g_{\boldsymbol{x}}\boldsymbol{f}} \tag{2.4}$$

are the two outstanding eigenvalues of $\boldsymbol{J}_P$ replacing $\lambda_n = 1$ of $\boldsymbol{H}$. The similar ansatz $\begin{pmatrix} \boldsymbol{u}_k + \beta_k \boldsymbol{f} \\ \alpha_k \end{pmatrix}$ for the k-th eigenvector ($k < n$) yields once more the eigenvalue $\mu_k - 1$ as well as $\alpha_k = \left((\mu_k - 1)^2 - g_{\boldsymbol{x}}\boldsymbol{f}\right)^{-1} g_{\boldsymbol{x}}\boldsymbol{u}_k$ and $\beta_k = (\mu_k - 1)\alpha_k$. The expression in parentheses vanishes iff one of the new eigenvalues is equal to $\mu_k - 1$, in this case the only eigenvector is $\begin{pmatrix} \boldsymbol{f} \\ \alpha \end{pmatrix}$.

It should be emphasis ed that the eigenvalue $\lambda_n = 1$ of $\boldsymbol{H}$ or the eigenvalues $\mu = \pm\alpha$ of $\boldsymbol{J}_P$ are insignificant for the determination of the stability of the cycle. Only the other $n - 1$ eigenvalues are significant since the dynamics is actually a map in $(n - 1)$-dimensional space.

Here, a remark is necessary why stationary points are not misinterpreted as cycles. For all T, the stationary points are solutions of the first equation of the system (2.2) and if the submanifold $\mathcal{S}$ is suitably chosen also of the second one. But in that case $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{0}$ holds and the Jacobian $\boldsymbol{J}_P$ is singular, thus the problem is now not well posed and no stationary point can be the result of Newton's method, i.e. only proper cycles will be computed.

CANDYS/QA provides three options for the function $g(\boldsymbol{x}, \boldsymbol{p})$ from which the user can select one:

1. $\mathcal{S}$ is a fixed plane orthogonal to the k-th axis supported by some chosen point $\boldsymbol{a}$, i.e.

   $$g(\boldsymbol{x}, \boldsymbol{p}) = (\boldsymbol{x} - \boldsymbol{a})_k. \tag{2.5}$$

   This option is the easiest to deal with.

2. $\mathcal{S}$ is the surface of all extrema of the trajectories with respect to one component x_k, i.e. the points characterized by $\dot{x}_k = f_k(\boldsymbol{x}, \boldsymbol{p}) = 0$:

   $$g(\boldsymbol{x}, \boldsymbol{p}) = f_k(\boldsymbol{x}, \boldsymbol{p}). \tag{2.6}$$

   Since $f_k(\boldsymbol{x}, \boldsymbol{p})$ may actually depend on the parameter $\boldsymbol{p}$ the submanifold $\mathcal{S}$ may vary with $\boldsymbol{p}$. For many applications this option is the most interesting one since the extrema of some components are often of practical importance.

3. $\mathcal{S}$ is the plane orthogonal to the trajectory supported by one trajectory point, i.e. with the current trajectory tangent $\boldsymbol{t} = \frac{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{p})}{\|\boldsymbol{f}\|}$ we obtain

   $$g(\boldsymbol{x}, \boldsymbol{p}) = \boldsymbol{t}^T(\boldsymbol{x} - \boldsymbol{x}^0) \tag{2.7}$$

   where $\boldsymbol{x}^0$ is a previously computed point.

The method described above differs from known algorithms to compute periodic solutions of Eq. (1.1). One suitable approach is the transformation of the problem into a

two point boundary value problem in the interval $[0, 1]$ (c.f. [21, 8, 43]) or $[0, 2\pi]$ (c.f. [38]). Standard algorithms like shooting methods [21] or collocation techniques [8] are used to find the solution. The period length is computed as one of the unknowns of the equations system replacing one of the state variables that is fixed at a suitable value (it constitutes the periodic boundary values). Another ansatz is given by Curry [7] and Khibnik [26] who defined explicitly a Poincaré plane by fixing one component or by adding a linear condition. This leads to the reduction of Eq. (1.1) to an $(n-1)$-dimensional algebraic system which describes the Poincaré map.

The disadvantage of the approaches mentioned last is that they have to compute exactly the Poincaré map in each Newton step. This is avoided by our method, since in the course of Newton's method the points $\boldsymbol{x}$ need not belong to the submanifold $\mathcal{S}$: the point $\boldsymbol{x}$ meets $\mathcal{S}$ just by convergence. Consequently, although our procedure uses an $(n+1)$-dimensional equations system it seems to be more efficient than the approaches suggested by Curry and Khibnik. (This situation might be compared with the method of Lagrange multipliers for searching an extremum of a function of n variables with one equality constraint. Instead of solving the equality constraint and so reducing the optimization problem by one dimension the problem is enlarged by one variable, the Lagrange multiplier, yielding an $(n+1)$-dimensional problem of a more homogeneous form.) Moreover, Eq. (2.2) allows the use of Newton's method, path-following algorithms etc. without a special adaptation.

2.1.2 Artificial bifurcations

Let us have a look at the singularities of Eq. (2.2). Equation (2.2) can cause specific numerical failures that come in the disguise of a bifurcation. $\boldsymbol{J}_P$ is regular iff all $\lambda_i \neq 1$ $(i < n)$ and α is nonzero. Then $\boldsymbol{J}_P$ is also regular for all $(\boldsymbol{x}, T)$ close to the cycle $(\boldsymbol{x}^*, T^*)$ and Newton's method is well posed.

Much more is to be said about the case $\alpha = 0$. Although this is forbidden by the transversality condition (2.1), it may occur during a path-following procedure. The following examples show that it is a quite natural situation, but it is an artifact from the point of view of the qualitative system behavior. It results from a not suitable choice of the submanifold $\mathcal{S}$. Now let us discuss the case $\alpha = 0$. It holds

$$\begin{pmatrix} \boldsymbol{H} - \boldsymbol{I} & \boldsymbol{f} \\ g_{\boldsymbol{x}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix},$$

i.e. $\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix}$ is mapped onto the eigenvector $\begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}$ implying that $\boldsymbol{J}_P$ possesses a Jordan block of two rows corresponding to the eigenvalue 0, and the enlarged Jacobian matrix $(\boldsymbol{J}_P, \boldsymbol{J}_p)$ is of rank n or $n+1$. (To see this look at the enlarged Jordan block that is of the form $\begin{pmatrix} 0 & 1 & a \\ 0 & 0 & b \end{pmatrix}$, its rank is 1 or 2 depending on b). Rank $n+1$ means that we have the same constellation of the eigenvalues as in the case of a turning point of the Eq. (2.2) (c.f. Sec. 2.4.1 for definition of turning points and simple bifurcations). However, this "turning point" is an artifact. From the geometrical point of view, two intersections of the trajectory with the submanifold $\mathcal{S}$ meet and annihilate here. That means the path before the "turning point" describes one crossing point and afterward the other one, the same parameter interval is visited twice, and both paths are redundant to each

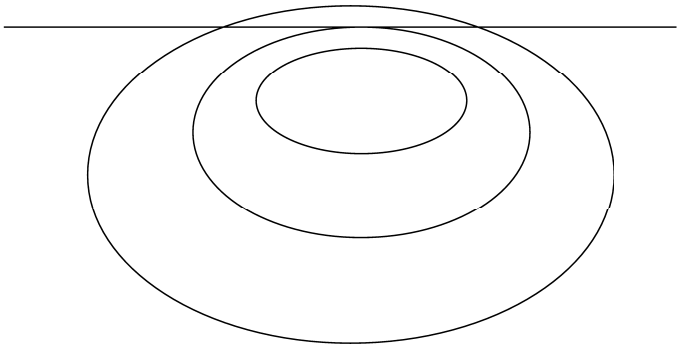


Figure 2.2: An artificial turning point

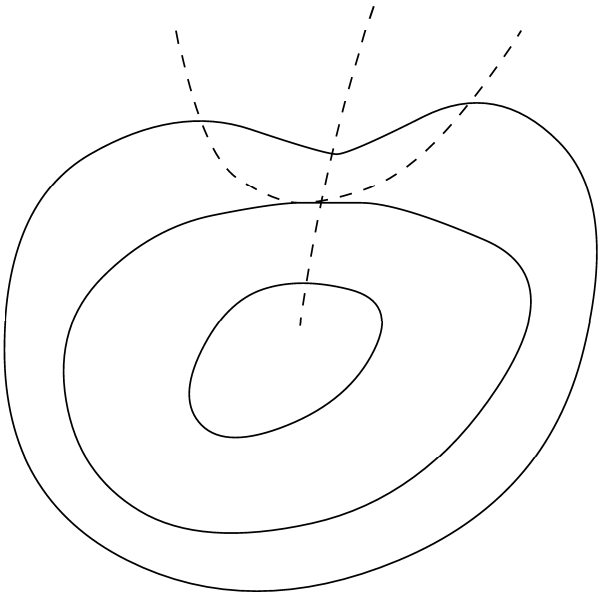


Figure 2.3: An artificial pitchfork bifurcation

other. This may be demonstrated by the following example. Let $\mathcal{S}$ be a fixed plane with the normal vector $g_{\boldsymbol{x}} = \boldsymbol{a}$. Assume that the cycle becomes smaller and smaller with varying $\boldsymbol{p}$, and after a certain $\boldsymbol{p}^*$ the cycle does not longer cross the plane. The solution $(\boldsymbol{x}, \boldsymbol{p})$ changes from the left to the right point of the same cycle (c.f. Fig. 2.2).

On the other hand, rank n corresponds to a simple bifurcation, e.g. a pitchfork, of the path. But Fig. 2.3 shows again that this "bifurcation" is an artifact where one crossing point is split into three ones. Here, we assume that $\mathcal{S}$ is the surface of the extrema of the trajectories with respect to the component x_k , and we assume that with variation of $\boldsymbol{p}$ the cycle is indented from above. At the parameter value $\boldsymbol{p}^*$ the maximum becomes one minimum and two maxima.

It should be emphasis ed that such artificial bifurcation situations cannot occur if the submanifold $\mathcal{S}$ is the plane orthogonal to the trajectory (option 3) since by Eqs. (2.4) and (2.5) $\alpha = 0$ never holds.

Artifacts may be avoided by several means. In CANDYS/QA we monitor the value of α during path following, and if it is shrinking towards zero then the user is interac-

tively asked to define the Poincaré map anew. Generally, this results in a jump or an angle of the solution path.

2.2 Getting a first point

2.2.1 Overview

The numerical investigation of a nonlinear dynamical system starts with the computation of one point of a chosen invariant set for a fixed set of parameters, shortly called the first point. Steady-states of autonomous systems as well as cycles of periodically forced systems and time discrete systems result from the solution of the nonlinear algebraic systems Eqs. (1.10) – (1.12). Using the method of Poincaré's first return map explained above, the search for periodic solutions in autonomous systems can be transformed to such an algebraic problem as well: Eq. (2.2). For this reason, the calculation of fixed points and cycles are based upon the same numerical algorithms. The corresponding equation systems for steady-states and cycles may be unified formally to a single one that has to be solved for $\boldsymbol{y}$ at a fixed parameter vector $\boldsymbol{p}$ (which is dropped in the following formulae for clarity):

$$\boldsymbol{F}(\boldsymbol{y}) = \boldsymbol{0} \quad , \quad \boldsymbol{y} \in \mathbb{R}^m. \tag{2.8}$$

In general, $\boldsymbol{y}$ is identical to $\boldsymbol{x}$, only in the case of Eq. (2.2) $\boldsymbol{y}$ contains besides $\boldsymbol{x}$ the period length T as one additional component. The function $\boldsymbol{F} : \mathbb{R}^m \to \mathbb{R}^m$ is one of the functions in Eqs. (1.10) – (1.12), or (2.2).

Standard numerical tools solving nonlinear algebraic systems need a good initial guess to converge. But in practice we are often faced with the problem, that we do not know so much about a possible location of the invariant sets sought for. Only sometimes the system possesses a trivial solution which can be used as a first point. For this reason CANDYS/QA offers an extensive support for getting a first point.

As a result of one or a combination of all these procedures a certain invariant set (steady-state or cycle) is obtained for a fixed set of model parameters. Additionally, the corresponding Jacobian $\boldsymbol{J}$ or $\boldsymbol{H}$, respectively, and its eigenvalues determining the stability of the chosen invariant set are calculated.

2.2.2 Newton's method

The most accurate algorithm is a damped Newton's method (more precisely: a chord method where after several steps the Jacobian matrix is newly calculated); for simplicity, we refer to this method as Newton's method in the sequel.

When using Newton's method to solve Eq. (2.8) we need a Jacobian $\boldsymbol{J}_N$. For the different nonlinear equations systems Eqs. (1.10) – (1.12), (2.2) different Jacobians have to be supplied:

1. In the case of steady-states Eq. (1.10) it is the ordinary Jacobian $\boldsymbol{J}$ defined by Eq. (1.4).

$$\boldsymbol{J}_N = \boldsymbol{J} \tag{2.9}$$

2. Time discrete systems Eq. (1.11) and periodically forced systems Eq. (1.12) have a Jacobian $\boldsymbol{J}_N$ consisting of the difference of the fundamental matrix $\boldsymbol{H}$ and the

unit matrix $\boldsymbol{I}$

$$\boldsymbol{J}_N = \boldsymbol{H} - \boldsymbol{I} \tag{2.10}$$

where the matrix $\boldsymbol{H}$ is obtained by Eq. (1.7) or Eq. (1.6).

3. The Jacobian $\boldsymbol{J}_N$ of autonomous systems is given by Eq. (2.3):

$$\boldsymbol{J}_N = \begin{pmatrix} \boldsymbol{H} - \boldsymbol{I} & \boldsymbol{f} \\ g_{\boldsymbol{x}} & 0 \end{pmatrix} \tag{2.11}$$

where $g_{\boldsymbol{x}}$ is the row vector $\frac{\partial g}{\partial \boldsymbol{x}}$.

For convergence of Newton's method a suitable initial guess is necessary which may be obtained by application of the other methods.

2.2.3 Monte-Carlo search and evolutionary search

Besides Newton's method as the standard solver for nonlinear equations a Monte-Carlo search in a certain region of the state space is offered. This region may be determined by the user as a parallelepiped around a chosen initial point as its center.

The evolutionary search is a derivative of the Monte-Carlo method towards modeling the Darwinian evolution [37, 40]. At each step, s points $\boldsymbol{y}^1$, ..., $\boldsymbol{y}^s \in \mathbb{R}^m$ exist (the parent generation), from them r further points $\boldsymbol{y}^{s+1}$, ..., $\boldsymbol{y}^{s+r}$ (the child generation) are created randomly with respect to probability distributions which are centered at the points of the parent generation. Thus, the child generation represents a "mutation" of the parent generation. Among all points the best s points (with respect to a certain objective functional) are chosen for "selection" which in turn constitute the parent generation of the next step.

In CANDYS/QA we use the simplest version where $s = r = 1$ and where the selection is made according to the objective functional $\|\boldsymbol{F}(\boldsymbol{y})\| \to \min$, thus the parent $\boldsymbol{y}$ tends to a solution of $\boldsymbol{F}(\boldsymbol{y}) = 0$. If during 5 consecutive steps at least one improvement is achieved then the size of the parallelepiped will be reduced, otherwise it will be enlarged.

Both methods possess the advantage that they act globally: they find a solution within a given (possibly large) rectangle. On the other hand, they converge (compared with Newton's method) very slowly in the vicinity of a solution. This means, that both methods should be used for the approximative localization of a solution only. Then this approximation should be improved by other methods.

2.2.4 Homotopy method

The homotopy approach starts with a simple equations system $\tilde{\boldsymbol{F}}(\boldsymbol{y})$ that is especially chosen to have an obvious solution $\tilde{\boldsymbol{F}}(\boldsymbol{y}^0) = \boldsymbol{0}$. Then we introduce a new nonlinear equations system $\boldsymbol{h}(\boldsymbol{y}, \tau)$ with an additional parameter τ which on one hand yields the simple system $\tilde{\boldsymbol{F}}(\boldsymbol{y})$ at $\tau = 0$, while on the other hand it yields the original system $\boldsymbol{F}(\boldsymbol{y})$ at $\tau = 1$. If we follow the path $\boldsymbol{y}(\tau)$ starting from $\boldsymbol{y}^0$ at $\tau = 0$, then it reaches at

$\tau = 1$ a solution of the original system. Several possibilities exist to define $\boldsymbol{h}(\boldsymbol{y}, \tau)$ with the properties mentioned above. The standard embedding with a Newton homotopy is implemented where $\tilde{\boldsymbol{F}}(\boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{y}^0$:

$$\boldsymbol{h}(\boldsymbol{y}, \tau) = (1 - \tau) \cdot (\boldsymbol{y} - \boldsymbol{y}^0) + \tau \cdot \boldsymbol{F}(\boldsymbol{y}).$$

(For an introduction c.f. [2, 13].)

The homotopy method may fail if the starting point $\boldsymbol{y}^0$ is chosen inappropriately: the path $\boldsymbol{y}(\tau)$ may not reach the value $\tau = 1$ but may tend to infinite values of $\boldsymbol{y}$ while τ varies within the interval $(0, 1)$.

2.2.5 Simulation

The method mostly used to find a suitable initial guess for Newton's method is of course the simulation of the trajectories. Contrarily to all other procedures which are independent of the stability of the invariant sets, simulation leads only to the approximation of stable invariant sets.

2.3 Parameter dependent solution curves

The location and the stability of a steady-state or a cycle vary in dependence on characteristic model parameters. For this reason the calculation of the chosen invariant set is only the first step in the software system CANDYS/QA. The next step is the study of the invariant set found previously if a single model parameter p_1 is varied in some interval. An analogue situation holds if a special point has been computed when parameters $p_1, \ldots, p_k$ ($k > 0$) are variable and if the dependence of this special point on an additionally varied parameter p_{k+1} is of interest. Thus, we consider Eq. (2.8) where the variables and equations have a more general meaning: $F_i(\boldsymbol{y})$ ($i = 1, \ldots, m$) denote all functions and y_j ($j = 1, \ldots, m$) denote all variables that constitute a specific nonlinear equations system.

According to the Implicit function theorem the solution of the m-dimensional nonlinear equations system (2.8) in dependence on the parameter p_{k+1} describes a curve $\mathcal{B}$ in $\mathbb{R}^{m+1}$. For convenience, we define $z_i = y_i \quad \text{for} \quad i = 1, \ldots, m, \quad \text{and} \quad z_{m+1} = p_{k+1}$. The system (2.8) reads now

$$\boldsymbol{F}(\boldsymbol{z}) = \boldsymbol{0} \tag{2.12}$$

and defines implicitly the curve $\mathcal{B}$. It is calculated step by step solving Eq. (2.12) by means of special computational methods, called continuation or path-following techniques (for an introduction to continuation c.f. [2, 45]).

In principle, two different concepts are known to compute the continuation of the curve $\mathcal{B}$ under the supposition that one or several points of the curve are already known. One way is the use of a simplicial algorithm where the state space is divided into simplices and where a chain of simplices is computed which contains the curve (c.f. [13, 15, 2]). The other concept is that of a predictor-corrector procedure [25, 1,

13, 45], which consists of two stages. The first one, the predictor, provides a suitable estimate for the next point lying near the curve, whereas the corrector calculates the exact point on $\mathcal{B}$.

In CANDYS/QA the predictor is implemented on two levels. We assume that the curve $\mathcal{B}$ is parameterized by the arc length s. Then the first level applies simply a linear step in the direction of the tangent unit vector $(\frac{d\boldsymbol{z}}{ds})$:

$$\boldsymbol{z}^{pred} = \boldsymbol{z}^0 + \sigma \left.\frac{d\boldsymbol{z}}{ds}\right|_{\boldsymbol{z}^0} \tag{2.13}$$

where $\boldsymbol{z}^0$ is the point computed last and σ denotes the demanded step size [1]. The disadvantage of the linear approximation is that the distance between $\mathcal{B}$ and the predictor increases too fast. Therefore, higher order predictors have been proposed [15, 41]. We use as second level a higher order correction of the linear predictor. A cubic approximation is obtained by the extrapolation of a Hermite interpolation polynomial which contains the last two calculated points and which is tangent to $\mathcal{B}$ in these points (a similar approach was used in [35]). Let $\boldsymbol{z}^{(-1)}$ be the point on the curve before $\boldsymbol{z}^0$ and define $\sigma_0 = \left\|\boldsymbol{z}^0 - \boldsymbol{z}^{(-1)}\right\|$, $\alpha = \frac{\sigma}{\sigma_0}$, $\beta = \alpha + 1$, $\gamma = (2\beta + 1)\alpha$, and $\delta = \alpha\beta$. Then

$$\boldsymbol{z}^{cubic} = \boldsymbol{z}^0 + \sigma \left(\beta \left.\frac{d\boldsymbol{z}}{ds}\right|_{\boldsymbol{z}^0} - \frac{\gamma}{\sigma_0}(\boldsymbol{z}^0 - \boldsymbol{z}^{-1}) + \delta \left(\left.\frac{d\boldsymbol{z}}{ds}\right|_{\boldsymbol{z}^0} + \left.\frac{d\boldsymbol{z}}{ds}\right|_{\boldsymbol{z}^{-1}} \right) \right) \tag{2.14}$$

is the cubic predictor. It uses only information already known, i.e. there is no need for further function calls.

The higher order predictors cause difficulties if the curvature of the curve is strong, since the predictor begins to "oscillate" around the curve $\mathcal{B}$. Thus, the higher order correction is not applied if the proposed predictor point lies beyond the point of inflection of the cubic polynomial. This way, the "oscillations" of the predictor due to the cubic approximation are avoided.

The corrector step is implemented in the usual way, i.e. a plane is defined which is supported at the predictor point and which is orthogonal to the curve, i.e. the plane defined by $\boldsymbol{F}(\boldsymbol{z}) - \boldsymbol{F}(\boldsymbol{z}^{pred}) = \boldsymbol{0}$ [1, 23]. Now, the nonlinear equations system is solved by Newton's method under the condition that the solution has to belong to this plane:

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{z}) &= \boldsymbol{0} \\ \left(\left.\tfrac{d\boldsymbol{z}}{ds}\right|_{\boldsymbol{z}^{pred}}\right)^T \cdot (\boldsymbol{z} - \boldsymbol{z}^{pred}) &= 0. \end{aligned} \tag{2.15}$$

The required Jacobian $\boldsymbol{J}_C$ for the corrector is the original Jacobian $\boldsymbol{J}_N$ defined by Eqs. (2.9) – (2.11) according to the system family under investigation and enlarged by one row and one column:

$$\boldsymbol{J}_C = \begin{pmatrix} \boldsymbol{J}_N & \boldsymbol{J}_p \\ \frac{d\boldsymbol{y}}{ds} & \frac{dp_k}{ds} \end{pmatrix} \tag{2.16}$$

where $\boldsymbol{J}_p$ denotes $\frac{\partial \boldsymbol{F}}{\partial p_k}$.

The step size control is based on the error of the predictor step. If the error is larger than a certain upper threshold then the prediction step is repeated with decreasing step size until the threshold is beaten or the step size becomes too small. On the other

hand, if the error is smaller than a lower threshold then the predictor step is repeated once with the doubled step size. In each case, the last step size is inherited to the next step.

For each point on the curve $\mathcal{B}$ the Jacobian $\boldsymbol{J}_N$ contains the matrix $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}$ as a submatrix. The eigenvalues of the latter can be evaluated and enable the monitoring of the stability of the steady-states or cycles along the solution branch.

2.4 Special points

2.4.1 General scheme

In Sec. 1.2 we defined that a special point is a point in the state-parameter space where locally different qualitative behavior occurs or that is characterized by a quantitative feature. This means:

1. the point belongs to an invariant set;
2. the point has extra properties distinguishing it from other points.

For the numerical treatment this means that a special point is a solution of an equations system that

1. includes the equations for the invariant set;
2. (a) has additional equations mimicking the equality conditions of the special point,
   (b) fulfills certain inequalities ensuring that the equations system is well posed.

A co-dimension-k bifurcation occurs, generically, in k-parameter space implying that according to 2(a) k equations have to be established additionally. In practice, the extra properties are often described by more equations what implies that as many auxiliary variables are introduced (the number of auxiliary variables can be of the order of the dimension of the state space or even a multiple thereof). An equations system for a special point is called minimally extended if it contains exactly k additional equations.

From the numerical point of view, the equations system should be minimally extended to save memory size. On the other hand, the additional equations should be algorithmically simple to reduce computing time. In CANDYS/QA we made a compromise: if auxiliary variables are used at all then their number depends on the complexity of the special point in question but not on the dimension of the state space. For example, the equations system for a turning point is minimally extended while that for a Hopf bifurcation contains one auxiliary variable.

The scheme of the equations systems to be solved is as follows:

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{p}) &= \boldsymbol{0} \\ -\varphi(\boldsymbol{y}, \boldsymbol{p}, \boldsymbol{c}) &= \boldsymbol{0} \end{aligned} \tag{2.17}$$

where $\boldsymbol{F} : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}^m$ describes the invariant set, $\varphi : \mathbb{R}^m \times \mathbb{R}^k \times \mathbb{R}^q \to \mathbb{R}^{k+q}$ are the additional equations, and $\boldsymbol{c} \in \mathbb{R}^q$ is the vector of the auxiliary variables (i.e. $q = 0$ in case of a minimally extended equations system).

Some or all, say l, components of the function φ are determined as part of the solution of the linear equations system

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{c}) & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \varphi \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix}. \tag{2.18}$$

Herein, $\boldsymbol{R}, \boldsymbol{L} \in \mathbb{R}^{m \times l}$ are certain fixed vectors or matrices with l columns, and $\boldsymbol{M} \in \mathbb{R}^{m \times m}$ is a matrix depending on the derivatives $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}$ such that $\boldsymbol{M}$ is singular at the special point. It is easy to see that the solution of Eq. (2.18) yields $\varphi = \boldsymbol{0}$ iff $\boldsymbol{M}$ is singular, in this case $\boldsymbol{u}$ is an eigenvector to the zero-eigenvalue.

Equation (2.17) is solved by Newton's method, i.e. we need the derivatives of φ. Suppose for simplicity $l = 1$, let $(\boldsymbol{v}^T, \varphi)$ denote the solution of the adjoint linear equations system

$$(\boldsymbol{v}^T, \varphi) \begin{pmatrix} \boldsymbol{M} & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} = (\boldsymbol{0}^T, 1)$$

and let α denote any of the components of $\boldsymbol{y}$, $\boldsymbol{p}$, or $\boldsymbol{c}$. Then the well known relation $\frac{\partial}{\partial \alpha} \boldsymbol{A}^{-1} = -\boldsymbol{A}^{-1} \left(\frac{\partial}{\partial \alpha} \boldsymbol{A} \right) \boldsymbol{A}^{-1}$ for any regular matrix $\boldsymbol{A}$ implies in the case $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{M} & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix}$:

$$\begin{aligned} \frac{\partial}{\partial \alpha}(-\varphi) &= -(\boldsymbol{0}^T, 1) \frac{\partial}{\partial \alpha} \begin{pmatrix} \boldsymbol{u} \\ \varphi \end{pmatrix} \\ &= -(\boldsymbol{0}^T, 1) \frac{\partial}{\partial \alpha} \left[\begin{pmatrix} \boldsymbol{M} & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} \right] \\ &= (\boldsymbol{0}^T, 1) \boldsymbol{A}^{-1} \frac{\partial}{\partial \alpha} \begin{pmatrix} \boldsymbol{M} & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \boldsymbol{A}^{-1} \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} \\ &= (\boldsymbol{v}^T, \varphi) \begin{pmatrix} \frac{\partial}{\partial \alpha} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \varphi \end{pmatrix} \\ &= \boldsymbol{v}^T \frac{\partial}{\partial \alpha} \boldsymbol{M} \boldsymbol{u} \end{aligned}$$

This means that

1. Eq. (2.18) has to be solved yielding both φ for the left hand side of Eq. (2.17) and $\boldsymbol{u}$;
2. the adjoint linear equations system has to solved yielding $\boldsymbol{v}$ (this is done utilizing the same LU-decomposition);
3. $\boldsymbol{v}^T \frac{\partial}{\partial \alpha} \boldsymbol{M} \boldsymbol{u}$ has to be calculated for all components of $\boldsymbol{y}$, $\boldsymbol{p}$ $\boldsymbol{c}$.

One problem is left open: how to determine appropriate vectors or matrices $\boldsymbol{R}$ and $\boldsymbol{L}$? First of all, the choice of $\boldsymbol{R}$ and $\boldsymbol{L}$ has to ensure that $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix}$ is nonsingular at the

special point, especially

$$\operatorname{rank}(\boldsymbol{M}, \boldsymbol{R}) = \operatorname{rank}(\boldsymbol{M}^T, \boldsymbol{L}) = m \tag{2.19}$$

must hold. On the other hand, if $\operatorname{rank} \boldsymbol{M} = m - l$ at the special point then any matrices $\boldsymbol{R}, \boldsymbol{L} \in \mathbb{R}^{m \times l}$ that satisfy condition (2.19) yield a nonsingular extended matrix. Almost all matrices satisfy the condition (2.19), especially one could set L and R equal to the right and left eigenvectors to the zero-eigenvalue of $\boldsymbol{M}$, respectively. Since these vectors are not known before the special point is determined we use an approximation: $\boldsymbol{M}$ is LU-decomposed at the starting point $(\boldsymbol{x}, \boldsymbol{p})$, if the decomposition is possible then the eigenvector $\boldsymbol{u}$ to the smallest eigenvalue is computed by standard methods. Otherwise, the nontrivial solutions of the right and left homogeneous linear equations systems are used.

Many authors use a scheme of explicitly extended equations systems, e.g.

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{p}) &= \boldsymbol{0} \\ \boldsymbol{M}(\boldsymbol{y}, \boldsymbol{p}, \boldsymbol{c})\boldsymbol{u} &= \boldsymbol{0} \\ \boldsymbol{u}^T\boldsymbol{u} - 1 &= 0 \end{aligned} \tag{2.20}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ belongs to the unknowns [3] (c.f. [32, 42, 18, 39]). This way, the solution of the linear equations system (2.18) is avoided. But in the course of Newton's method the Jacobian of Eq. (2.20) has to be decomposed what needs about $C(2m)^3$ operations. In our approach in each Newton step one decomposition of the matrix in Eq. (2.18) and one decomposition of the Jacobian of Eq. (2.17) is performed what together needs only about $2Cm^3$ operations (with the same constant C).

2.4.2 Extended equations systems

In the following, we will describe the extended equations systems implemented in CANDYS/QA. It should become clear that the equality conditions are satisfied while nothing is said about the regularity conditions. Similarly to the previous sections and chapters, we denote by $\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{p})$ the right hand side of the equations of a general invariant set but by $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p})$ the right hand side of the dynamics. In the first case, m denotes the dimension of $\boldsymbol{y}$. Similarly, $\boldsymbol{J}_N$ and $\boldsymbol{J}_{Np}$ denote the Jacobians of one of the equations Eqs. (1.10) – (1.12), (2.2) with respect to $\boldsymbol{y}$ or $\boldsymbol{p}$, respectively, while $\boldsymbol{J}$ and $\boldsymbol{J}_p$ denote the Jacobians defined in Eq. (1.4)

Turning points

The essential condition is that $\boldsymbol{J}_N$ has rank $m - 1$. Thus, we establish the linear equations system (2.18) in the form

$$\begin{pmatrix} \boldsymbol{J}_N & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \varphi \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix}. \tag{2.21}$$

Since there are no other equality conditions and no auxiliary variables, a minimally extended equations system results.

[3] In the case of a complicated bifurcation the vector $\boldsymbol{u}$ of auxiliary variables can be much larger.

Because of $\mathrm{rank}(\boldsymbol{J}_N, \boldsymbol{J}_{Np}) = m$ one could set $\boldsymbol{R} = \boldsymbol{J}_{Np}$ (CANDYS/QA does not). With this setting the method is essentially the same as those described in [36] or [18].

Simple bifurcations

The extended equations system has to cover two problems:

1. it is a co-dimension-2 bifurcation that occurs in 1-parameter space;
2. the equality conditions are $\mathrm{rank}\,\boldsymbol{J}_N = \mathrm{rank}(\boldsymbol{J}_N, \boldsymbol{J}_{Np}) = m - 1$.

The first problem is solved by introducing an artificial parameter α, the so-called imperfection parameter, and a fixed vector $\boldsymbol{z}$ such that $\mathrm{rank}(\boldsymbol{J}_N, \boldsymbol{J}_{Np}, \boldsymbol{z}) = m$ and setting $\tilde{\boldsymbol{F}}(\boldsymbol{y}, \boldsymbol{p}, \alpha) = \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{p}) + \alpha \boldsymbol{z}$. Thus, there are $m + 2$ variables totally.

We can split the second problem. $\mathrm{rank}(\boldsymbol{J}_N) = m - 1$ means that one component of φ, say φ_1, is to be computed as in the case of the turning point. Next, $\mathrm{rank}(\boldsymbol{J}_N, \boldsymbol{J}_{Np}) = m - 1$ means that $\boldsymbol{J}_{Np}$ depends linearly on the columns of $\boldsymbol{J}_N$: $\boldsymbol{J}_{Np} = \boldsymbol{J}_N \boldsymbol{u}_2$, i.e. it is a solution of

$$\begin{pmatrix} \boldsymbol{J}_N & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_2 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_{Np} \\ 0 \end{pmatrix} \tag{2.22}$$

such that $\varphi_2 = 0$. This means, $\varphi_2(\boldsymbol{y}, \boldsymbol{p}) = 0$ gives the second additional equation. Since the equations system has been extended by only 2 equations, a minimally extended equations system results. It should be emphasized that Eq .(2.22) is solved utilizing the same LU-decomposition as for Eq. (2.21)

The right hand side of Eq. (2.22) depends on the derivative of $\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{p})$ to a specific parameter $\boldsymbol{p}$. This reflects the fact that a simple bifurcation is not invariant with respect to transformations in parameter space.

Period doublings

The equality condition demands that $\boldsymbol{H}$ has the eigenvalue -1. i.e. $\boldsymbol{J}_N + 2\boldsymbol{I} = \boldsymbol{H} + \boldsymbol{I}$ has the eigenvalue 0. Thus, we can simply set $\boldsymbol{M} = \boldsymbol{J}_N + 2\boldsymbol{I}$, and a minimally extended equations system results.

Hopf bifurcations

At a Hopf bifurcation point with the eigenvalues $\pm i\omega$ the matrix $\boldsymbol{M}(\boldsymbol{x}, \boldsymbol{p}, c) = \boldsymbol{J}^2 + c\boldsymbol{I}$ (where $c = \omega^2$) has the double eigenvalue 0. If $\boldsymbol{w} = \boldsymbol{u} + i\boldsymbol{v}$ is an eigenvector of $\boldsymbol{J}$ to $i\omega$ then $\boldsymbol{u}$ and $\boldsymbol{J}\boldsymbol{u}$ span the eigenspace of $\boldsymbol{M}$ to the zero-eigenvalue. Thus, we pose the linear equations in the form

$$\begin{pmatrix} \boldsymbol{J}^2 + c\boldsymbol{I} & \boldsymbol{R} & \boldsymbol{J}\boldsymbol{R} \\ \boldsymbol{L}^T & 0 & 0 \\ \boldsymbol{L}^T\boldsymbol{J} & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ 0 \\ 1 \end{pmatrix}. \tag{2.23}$$

Other equations are not needed. But since the variable c is introduced and φ has two components, this does not lead to a minimally extended equations system. (In [49] a minimally extended equations system is proposed but there Eq. (2.23) has to be replaced by a nonlinear equations system.)

Suppose that a solution with $\varphi_1 = \varphi_2 = 0$ has been found. Then $\boldsymbol{z} = i\sqrt{c}\boldsymbol{u} + \boldsymbol{J}\boldsymbol{u}$ is an eigenvector to the eigenvalue $i\sqrt{c}$ since $\boldsymbol{J}^2\boldsymbol{u} = -c\boldsymbol{u}$ and

$$\boldsymbol{J}\boldsymbol{z} = i\sqrt{c}\boldsymbol{J}\boldsymbol{u} - c\boldsymbol{u} = i\sqrt{c}(\boldsymbol{J}\boldsymbol{u} + i\sqrt{c}\boldsymbol{u}) = i\sqrt{c}\boldsymbol{z}.$$

In the case $c > 0$ the imaginary parts of the critical eigenvalues of $\boldsymbol{J}$ are given by $\omega = \pm\sqrt{c}$ while $\boldsymbol{J}\boldsymbol{u}$ and $\pm i\omega\boldsymbol{u}$ are the real and imaginary parts, respectively, of the eigenvectors.

If c is negative the solution is sometimes called an imaginary Hopf bifurcation since now $\omega = \pm\sqrt{c}$ is imaginary (c.f. [49]). In the case of path-following a Hopf bifurcation (i.e. $c > 0$) the sign of c can change ($c = 0$ corresponds to a Takens-Bogdanov point) and the path continues as a curve of imaginary Hopf bifurcations. Although the latter does not have a meaning for the qualitative analysis the path is followed since the possibility exists that the sign of c changes once more yielding another branch of Hopf bifurcations that might be over-seen otherwise.

Torus bifurcations

The equality condition that a pair of conjugate complex eigenvalues of $\boldsymbol{H}$ belong to the unit circle is transformed to the condition that a related matrix has eigenvalues on the imaginary axis. The transformation

$$w = \frac{z}{z+2}$$

of the complex plane maps the circle $|z+1| = 1$ to the imaginary axis where the interior of this circle is mapped to the left half plane (recall that the matrix $\boldsymbol{J}_N$ used in Newton's method has the critical eigenvalues on this circle). This means, $\boldsymbol{J}_N(\boldsymbol{J}_N + 2\boldsymbol{I})^{-1}$ has eigenvalues on the imaginary axis and Eq. (2.23) could be applied to this matrix. This form, however, is not yet appropriate since a new inverse matrix occurs. This can be overcome as follows:

$$\begin{aligned}
\boldsymbol{0} &= [(\boldsymbol{J}_N + 2\boldsymbol{I})^{-2}\boldsymbol{J}_N^2 + \omega^2\boldsymbol{I}]\boldsymbol{u} \\
&= (\boldsymbol{J}_N + 2\boldsymbol{I})^{-2}[(1+\omega^2)\boldsymbol{J}_N^2 + 4\omega^2(\boldsymbol{J}_N + \boldsymbol{I})]\boldsymbol{u} \\
&= (1+\omega^2)(\boldsymbol{J}_N + 2\boldsymbol{I})^{-2}[\boldsymbol{J}_N^2 + 2c(\boldsymbol{J}_N + \boldsymbol{I})]\boldsymbol{u}
\end{aligned}$$

with $c = \frac{2\omega^2}{1+\omega^2}$; c is the new unknown replacing ω. Since the factors in front of the brackets cannot vanish, the equation reduces to

$$[\boldsymbol{J}_N^2 + 2c(\boldsymbol{J}_N + \boldsymbol{I})]\boldsymbol{u} = \boldsymbol{0}.$$

and Eq. (2.17) becomes

$$\begin{pmatrix} \boldsymbol{J}_N^2 + 2c(\boldsymbol{J}_N + \boldsymbol{I}) & \boldsymbol{R} & \boldsymbol{J}_N\boldsymbol{R} \\ \boldsymbol{L}^T & 0 & 0 \\ \boldsymbol{L}^T\boldsymbol{J}_N & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ 0 \\ 1 \end{pmatrix}. \tag{2.24}$$

If $c \in (-2, 0)$ then a torus bifurcation is found with the critical eigenvalues $e^{\pm i\omega}$, $\omega = \arccos(1 + c)$. If $c < -2$ or $c > 0$ then the point is an imaginary torus bifurcation. The path-following of imaginary torus bifurcation is done for the same reasons as it is done for imaginary Hopf bifurcations.

Takens-Bogdanov points and generalizations

A Takens-Bogdanov point is a co-dimension-2 bifurcation, i.e. we need two additional equations. Since $\operatorname{rank} \boldsymbol{J}_N = m - 1$, the first component, φ_1, is given by Eq. (2.21). The next equality condition is the existence of a further zero eigenvalue but no further eigenvector. Instead a vector $\boldsymbol{u}_2$ exists such that $\boldsymbol{M}\boldsymbol{u}_2 = \boldsymbol{u}_1$ where $\boldsymbol{u}_1$ is the eigenvector, i.e. where $\boldsymbol{u}_1$ belongs to the solution of Eq. (2.21). Hence, the second additional equation is $\varphi_2 = 0$ with φ_2 given by

$$\begin{pmatrix} \boldsymbol{J}_N & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_2 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_1 \\ \varphi_1 \end{pmatrix}.$$

The idea can easily be generalized: if 0 is to be an eigenvalue of algebraic multiplicity s and geometrical multiplicity 1 then we solve

$$\begin{pmatrix} \boldsymbol{J}_N & \boldsymbol{R} \\ \boldsymbol{L}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_{r+1} \\ \varphi_{r+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_r \\ \varphi_r \end{pmatrix} \tag{2.25}$$

for $r = 1, \ldots, s-1$, and $\begin{pmatrix} \boldsymbol{u}_1 \\ \varphi_1 \end{pmatrix}$ given by Eq. (2.21) as usually. This equations system is essentially the same as the one proposed in [17].

Another generalization concerns the value of the critical eigenvalue. If in Eq. (2.25) the matrix $\boldsymbol{M} = \boldsymbol{J}_N$ is replaced by the matrix corresponding to a period doubling, a Hopf bifurcation, or a torus bifurcation then a point is computed with eigenvalue -1, $\pm i\omega$, or $e^{\pm i\omega}$, respectively, that has algebraic multiplicity s but geometrical multiplicity 1. In the case of turning points and period doublings minimally extended equations systems result for co-dimension-s bifurcations, while in the case of Hopf and torus bifurcations the extended equations systems are not minimally extended and concern co-dimension-$(2s-1)$ problems ($2s$ additional equations but one auxiliary variable, c).

Cusp points and generalizations

The cusp point is a turning point where also a higher derivative of $\boldsymbol{F}$ vanishes. Thus, one additional equation is $\varphi_1(\boldsymbol{y}, \boldsymbol{p}) = 0$ where φ_1 is given by Eq. (2.21). Next, if the dynamical system is transformed to normal form then on the 1-dimensional center manifold holds: $\frac{\partial^2 f}{\partial x^2} = 0$. The transformation to the center manifold is given (to the order $O(||\boldsymbol{y} - \boldsymbol{y}^*||)$) by the projection in direction of the eigenvector $\boldsymbol{u}$ onto the plane orthogonal to the adjoint eigenvector $\boldsymbol{v}$. In the original coordinate system the condition becomes:

$$\varphi_2(\boldsymbol{y}, \boldsymbol{p}) \equiv \boldsymbol{v}^T \left(\frac{\partial \boldsymbol{J}_N}{\partial \boldsymbol{y}} \boldsymbol{u} \right) \boldsymbol{u} = 0. \tag{2.26}$$

The idea can be generalized to degeneration of other bifurcations: to period doublings, Hopf bifurcations, torus bifurcations, or even the cusp bifurcations. But now the degeneracy condition includes the third derivative of $f(\boldsymbol{x})$, and the projection on the center manifold depends also on the third derivative. This implies that in original coordinates the additional equation is much more complicated, e.g.

$$\varphi_2 \equiv 3\boldsymbol{v}^T \left(\frac{\partial \boldsymbol{J}_N}{\partial \boldsymbol{y}}\boldsymbol{u}\right)(\boldsymbol{J}_N)^{-1}\left(\frac{\partial \boldsymbol{J}_N}{\partial \boldsymbol{y}}\boldsymbol{u}\right)\boldsymbol{u} - \boldsymbol{v}^T\left(\frac{\partial^2 \boldsymbol{J}_N}{\partial \boldsymbol{y}^2}\boldsymbol{u}\boldsymbol{u}\right)\boldsymbol{u} = 0 \tag{2.27}$$

in case of the period doubling. Essentially the same expression yields the equation for a degenerate cusp point: one has simply to replace $(\boldsymbol{J}_N)^{-1}$ by the inverse of the matrix of Eq. (2.21). Because of their complexity, equations systems for degenerate Hopf and torus bifurcations have not yet been implemented.

In the case of a turning point (or a cusp point), the expression in Eq. (2.26) (or in the adapted version of Eq. (2.27)) can be very easily obtained: it is the scalar product of a row of the Jacobian computed during Newton's method with the vector $\boldsymbol{u}$. On the other hand, in the case of a period doubling only $\left(\frac{\partial \boldsymbol{J}_N}{\partial \boldsymbol{y}}\boldsymbol{u}\right)$ is known while $(\boldsymbol{J}_N)^{-1}$ and $\frac{\partial^2 \boldsymbol{J}_N}{\partial \boldsymbol{y}^2}\boldsymbol{u}\boldsymbol{u}$ have to be computed additionally.

In all cases, the value of φ_2 can be understood as the degree of non-degeneracy of the special point: $\varphi_2 = 0$ means a degenerate special point and the larger $|\varphi_2|$ is the more non-degenerate is the point. Moreover, in the case of a period doubling the sign of φ_2 determines the stability of the branching off solutions of doubled period: these solutions are stable iff $(\boldsymbol{v}'^T\boldsymbol{u})^{-1}\varphi_2 > 0$. [4]

Constant rotation

The equations system is simply Eq. (2.17) with the function φ given by Eq. (2.24) where c is no longer a variable, instead its fixed value is used. This way, a minimally extended equations system results.

Other special points

The additional equation for constant quantities is simply given by

$$\varphi \equiv Q(\boldsymbol{x}, \boldsymbol{p}) - Q_0 = 0$$

(similarly for constant return time).

In the case of an extreme quantity the method of Lagrange multipliers, i.e. the zero condition for derivatives of $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) \equiv Q(\boldsymbol{x}, \boldsymbol{p}) + \boldsymbol{v}'^T\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p})$, yields the equality condition

$$(\boldsymbol{v}^T, 1)\begin{pmatrix} \boldsymbol{J} & \boldsymbol{J}_p \\ \frac{\partial Q}{\partial \boldsymbol{x}} & \frac{\partial Q}{\partial \boldsymbol{p}} \end{pmatrix} = (\boldsymbol{0}^T, 0)$$

i.e. the matrix is singular. Thus, we set $\boldsymbol{M}$ to this matrix in Eq. (2.17).

[4] the factor $(\boldsymbol{v}^T\boldsymbol{u})^{-1}$ ensures that the sign of that expression does not depend on the arbitrarily chosen auxiliary vectors $\boldsymbol{R}$ and $\boldsymbol{L}$.

Combinations of special points

Special points can be combined, e.g. to a point that is simultaneously a turning point and a Hopf bifurcation. The corresponding extended equations system is constructed simply as the union of the extended equations systems of the special points to be combined.

2.4.3 Branching-off curves

At special points often two or more curves of invariant sets (or of special points of lower co-dimension) meet. These can be invariant sets of the same kinds as in the case of a simple bifurcation, or of different kinds as in the case of a Hopf bifurcation. If the special point is obtained in the course of path-following then one curve is already known. The task is then to compute also the other ones, the so-called branching off curves. One point of them is, of course, the special point, thus, the tangent of the branching off curves has to determined. Then the predictor-corrector method can be applied.

The tangent depends often on the eigenvector(s) $\boldsymbol{u}$ (and $\boldsymbol{u}_2, \dots$) of the matrix $\boldsymbol{M}$ (c.f. Sec. 2.4.2 for its definition) and, sometimes, on higher derivatives of $\boldsymbol{F}(\boldsymbol{y})$. Both entities are known as a side-effect of the computation of the special point by Newton's method. It has to be remarked that the current version of CANDYS/QA does not yet determine all tangents at the co-dimension-2 bifurcations described in Sec. 1.4.3. [5]

In the following, we list the formulae for the tangents as they are implemented in CANDYS/QA. In some formulae we split the tangent vector $\boldsymbol{t}$ into several subvectors, e.g. $\begin{pmatrix} \boldsymbol{t}_x \\ \boldsymbol{t}_p \end{pmatrix}$ is its splitting into the parts corresponding to state variables and parameters.

Simple bifurcations

The tangents of both curves belong to the plane spanned by the vectors $\begin{pmatrix} \boldsymbol{u}_1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} \boldsymbol{u}_2 \\ -1 \end{pmatrix}$:

$$\boldsymbol{t}^{1,2} = \eta_{1,2} \begin{pmatrix} \boldsymbol{u}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{u}_2 \\ -1 \end{pmatrix},$$

$\eta_{1,2}$ are solutions of the quadratic equation

$$\alpha\eta^2 + 2\beta\eta + \gamma = 0$$

[5] The most important examples of missing tangents are those to the branching off torus bifurcations at a point where a Hopf bifurcation is combined with a turning point or another Hopf bifurcation.

with

$$\begin{aligned}\alpha &= \boldsymbol{v}^T \frac{\partial^2 F}{\partial(\boldsymbol{y},p)^2}\begin{pmatrix}\boldsymbol{u}_1\\0\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_1\\0\end{pmatrix}\\ \beta &= \boldsymbol{v}^T \frac{\partial^2 F}{\partial(\boldsymbol{y},p)^2}\begin{pmatrix}\boldsymbol{u}_1\\0\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_2\\-1\end{pmatrix}\\ \gamma &= \boldsymbol{v}^T \frac{\partial^2 F}{\partial(\boldsymbol{y},p)^2}\begin{pmatrix}\boldsymbol{u}_2\\-1\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_2\\-1\end{pmatrix}\end{aligned}$$

Period doublings

The tangent of the curve of the doubled period is simply $\begin{pmatrix}\boldsymbol{t}_y\\\boldsymbol{t}_p\end{pmatrix} = \begin{pmatrix}\boldsymbol{u}\\0\end{pmatrix}$. The tangent of the simple period is obtained by the standard formula

$$\begin{pmatrix}\boldsymbol{J}_N & Jv_{Np}\end{pmatrix}\begin{pmatrix}\boldsymbol{t}_y\\\boldsymbol{t}_p\end{pmatrix} = 0.$$

Hopf bifurcations

The tangent of the curve of the cycles is $\begin{pmatrix}\boldsymbol{t}_x\\\boldsymbol{t}_T\\\boldsymbol{t}_p\end{pmatrix} = \begin{pmatrix}\tilde{\boldsymbol{u}}\\0\\0\end{pmatrix}$ where $\tilde{\boldsymbol{u}} = \eta\boldsymbol{u} + \theta\boldsymbol{J}_N\boldsymbol{u}$. The coefficients η and θ are determined such a way that the tangent $\boldsymbol{t}_x$ belongs to the tangent space of the Poincaré surface $g(\boldsymbol{x},\boldsymbol{p}) = 0$: $\eta g_{\boldsymbol{x}}\boldsymbol{u} + \theta g_{\boldsymbol{x}}\boldsymbol{J}\boldsymbol{u} = 0$.

The tangent of the steady-state curve is obtained by the standard formula

$$\begin{pmatrix}\boldsymbol{J} & \boldsymbol{J}_p\end{pmatrix}\begin{pmatrix}\boldsymbol{t}_x\\\boldsymbol{t}_p\end{pmatrix} = 0.$$

Combination of special points

If s different special points are combined then there are s curves corresponding to all possible subcombinations of $s-1$ special points. The LU-decomposition of the matrix

$$\begin{pmatrix}\frac{\partial F}{\partial \boldsymbol{y}} & \frac{\partial F}{\partial \boldsymbol{p}} & \frac{\partial F}{\partial \boldsymbol{c}}\\ \frac{\partial \varphi_1}{\partial \boldsymbol{y}} & \frac{\partial \varphi_1}{\partial \boldsymbol{p}} & \frac{\partial \varphi_1}{\partial \boldsymbol{c}}\\ \cdots & \cdots & \cdots\\ \frac{\partial \varphi_s}{\partial \boldsymbol{y}} & \frac{\partial \varphi_s}{\partial \boldsymbol{p}} & \frac{\partial \varphi_s}{\partial \boldsymbol{c}}\end{pmatrix}$$

is known as a result of Newton's method for the s-fold combination. Thus, the equations systems

$$\begin{pmatrix}\frac{\partial F}{\partial \boldsymbol{y}} & \frac{\partial F}{\partial \boldsymbol{p}} & \frac{\partial F}{\partial \boldsymbol{c}}\\ \frac{\partial \varphi_1}{\partial \boldsymbol{y}} & \frac{\partial \varphi_1}{\partial \boldsymbol{p}} & \frac{\partial \varphi_1}{\partial \boldsymbol{c}}\\ \cdots & \cdots & \cdots\\ \frac{\partial \varphi_s}{\partial \boldsymbol{y}} & \frac{\partial \varphi_s}{\partial \boldsymbol{p}} & \frac{\partial \varphi_s}{\partial \boldsymbol{c}}\end{pmatrix}\begin{pmatrix}\boldsymbol{t}_y\\\boldsymbol{t}_p\\\boldsymbol{t}_c\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\1\\\boldsymbol{0}\end{pmatrix} \tag{2.28}$$

yield the tangents. The 1 in the right hand side occurs in the line of the derivatives of the function φ_r to be excluded from the full combination. In the case that a Hopf or torus bifurcation is to be excluded (i.e. φ_r consists of two functions), the right hand side of Eq. (2.28) is a bit more complicated ensuring that the component of the tangent into the direction of the corresponding auxiliary variable becomes zero.

Takens-Bogdanov points and generalizations

At a generalized Takens-Bogdanov point where the zero eigenvalue has algebraic multiplicity s the tangents of the following curves are determined:

- turning points with the zero-eigenvalue of multiplicity $s-1$;
- Hopf bifurcations of multiplicity $\lfloor \frac{s}{2} \rfloor$, they are combined with a turning point if s is odd.

The formulae for these tangents are very similar to Eq. (2.28) with the exception that the tangent of the curve of Hopf bifurcations has to be extended by a component for the auxiliary variable.

2.4.4 Detection of special points

All bifurcation points are characterized by a special configuration of the eigenvalues of $\boldsymbol{J}_N$. Thus, the easiest way to find out sudden changes in the qualitative behavior is the monitoring of the eigenvalues. But this is not unique in some cases and, therefore, it is necessary to define criteria allowing an exact distinction between all special points considered.

Additionally, if the special point is accompanied with the singularity of the Jacobian matrix $\boldsymbol{J}_C$ then this, firstly, may lead to a failure of Newton's method during path-following near the special point, or secondly, may cause the computed path to leave the current solution branch and to switch to a branching off one without changing the constellation of the eigenvalues. (In spite of Keller's theorem 4.4 [25] this situation may occur because the assumption of this theorem cannot always be guaranteed in practical problems.) To avoid these difficulties, it is necessary to detect such points already before reaching them.

All checks yield inaccurate results if the step size on the curve is too large, such that more than one real eigenvalue or one pair of complex eigenvalues crosses the imaginary axis or the unit circle.

Turning points and simple bifurcations

Both, turning and simple bifurcation points are characterized by the fact that one real eigenvalue passes 0. Therefore, a careful distinction is necessary and the check criteria should be explained together. As a test function the change of the Poincaré index I is

used. Here I is defined as the product of the signs of all real parts of the eigenvalues of $\boldsymbol{J}_N$ [13, 19]:

$$I = \prod_i \operatorname{sign}(\Re\lambda_i).$$

To distinguish turning and bifurcation points in a unique manner several additional checks are necessary. Turning points are always connected with a change of the direction of the parameter variation. This means that the derivative of the parameter with respect to the arc length changes its sign if a turning point is passed:

$$\left.\frac{\partial p_1}{\partial s}\right|_{\text{before TP}} \cdot \left.\frac{\partial p_1}{\partial s}\right|_{\text{after TP}} < 0.$$

In the case of a pitchfork bifurcation a change in the direction of the parameter is possible (namely, if one moves along the outer branches) but then this is not accompanied with a change of I.

Additionally, in the case of a turning point the enlarged Jacobian $\boldsymbol{J}_C$ is not singular in contrast to the case of a simple bifurcation point. This feature can be used for a distinction between these two cases, too.

The checks mentioned above suppose that the special point has already been passed during the path-following. To check the approach of a simple bifurcation point (i.e. of a singularity of the Jacobian $\boldsymbol{J}_C$) before passing it, not the zero of the determinant is considered but the logarithm of the derivative of the determinant with respect to the arc length s. Approaching the simple bifurcation point this derivative tends to infinity and some threshold is used to indicate this divergence. This check criterion has the advantage, that it is independent of the scale of the variables. If s_{bif} denotes the length of the arc between the current point and the simple bifurcation point then

$$\frac{d}{ds}\log|\det \boldsymbol{J}_C| = -\frac{1}{s_{bif}} + O(1) \tag{2.29}$$

holds, the proof of this equation will appear elsewhere. Additionally, it can be used to estimate a step size to get a good initial guess for the computation of the special point itself. If the step size on the solution branch is large as in the case of a small curvature of the branch, then the criterion does not help, since the divergence of the logarithm of the derivative of the determinant may be not sufficiently large to reach some threshold. In that case the check is focussed on the change of the index I accompanied with no change in the direction of the parameter variation.

Table 2.1 summaries the check criteria for turning and simple bifurcation points and only the combination of all ensures a unique distinction. Note, that the first three criteria give results if the special point has been already passed, but the last one indicates the approach to a simple bifurcation point.

Hopf bifurcations

During the path-following there are two possibilities to approach a Hopf bifurcation point: Following a branch of steady-states the Hopf bifurcation is indicated by the fact

	change of index I	change of direction of parameter λ_k	derivative of $\log\|\det\|$ >threshold
turning point	yes	yes	no
simple bifurcation	yes	no	no
simple bifurcation	no	yes	no
simple bifurcation	no	no	yes

Table 2.1: Check criteria for turning and simple bifurcation points.

that two complex eigenvalues of the matrix $\boldsymbol{J}_N$ cross the imaginary axis, and this can be used as the unique check criterion.

Following a branch of cycles, the periodic motion vanishes at the Hopf bifurcation point and two branches of steady-states (a stable and an unstable one) arise. To check the approach to a Hopf bifurcation point we monitor the derivative of the logarithm of the norm of $\boldsymbol{f}$ with respect to the arc length s. Analogously to Eq. (2.29) the estimate

$$\frac{d}{ds}\log\|\boldsymbol{f}(\boldsymbol{x},\lambda_k)\| = -\frac{1}{s_{Hopf}} + O(1) \tag{2.30}$$

holds where s_{Hopf} is the arc length between the current and the Hopf bifurcation point. This way we detect that a point of zero of $\boldsymbol{f}(\boldsymbol{x})$, i.e. a stationary solution, is approached.

Torus bifurcations

Two complex eigenvalues of $\boldsymbol{H}$ pass the unit circle (i.e. two eigenvalues of $\boldsymbol{J}_N$ pass the shifted unit circle $|z+1|$), which can be used as a suitable check criterion [24].

Since CANDYS/QA does not compute tori, the counterpart, i.e. approaching the torus bifurcation point from a curve of tori, is out of concern.

Period doublings

The check is based on the fact, that one real eigenvalue of the fundamental matrix $\boldsymbol{H}$ passes the unit circle at -1. Introducing an index I_{per} by

$$I_{per} = \prod_i \operatorname{sign}(\Re\lambda_i + 1)$$

then this index I_{per} changes its sign at a period doubling point.

Following the branch of the doubled period length the period doubling point appears as a pitchfork bifurcation of this branch. In that case the period doubling is detected and computed as a simple bifurcation point, and then the computed point is checked for being a cycle point of half period. Moreover, two branches of this pitchfork have to be identified as branches of the simple period.

Takens-Bogdanov points and generalizations

Takens-Bogdanov points can be detected by monitoring of the eigenvalues of $\boldsymbol{J}_N$. We have to distinguish, whether the Takens-Bogdanov point is approached from a curve of turning points or from a curve of Hopf bifurcations. In the first case one eigenvalue is always equal 1 and is excluded from the monitoring. Thus, a Takens-Bogdanov point is found if one of the remaining real eigenvalues passes through 0. In the second case one could monitor the eigenvalues on the imaginary axis. But this prevents the detection of a Takens-Bogdanov point on a curve of imaginary Hopf bifurcations (here, two real eigenvalues would have to be monitored). Thus, we use another criterion: the value of the auxiliary variable c. Its crossing through 0 signals that a Takens-Bogdanov point has been passed.

It should be clear, that the same ideas are applicable in the case of the generalizations of the Takens-Bogdanov points mentioned in Sec. 1.4.3.

Cusp points and generalizations

A cusp points is detected by monitoring the value of

$$\boldsymbol{v}^T \frac{\partial^2 F}{\partial \boldsymbol{y}^2} \boldsymbol{u}\boldsymbol{u}$$

i.e. the value to be zero at the cusp point (c.f. Eq. (2.26)). Similarly, a degenerate period doubling is found by monitoring the expression in Eq. (2.27).

Extrema and fixed values of quantities

The quantities themselves are monitored. An interesting fixed value is detected when the quantity passes it, while an extremum is detected when for three consecutive curve points the quantity values form a down-up or up-down pattern.

Chapter 3

Graph Representation of the Solutions

3.1 k-parameter graphs

In this chapter we study bifurcation diagrams from another point of view. Given one chosen varying parameter, say p_1, and the values for the fixed parameters, then the set of all steady-states and/or cycles of the system constitutes an undirected, not necessarily connected graph: the special points represent the nodes and the intermediate solutions belong to the edges. The steady-states and/or cycles calculated by CANDYS/QA constitute a subgraph. We call it a 1-parameter graph. Besides special points, a further type of nodes appears: the temporary endpoints where the computation was started or stopped. The temporary endpoints are characterized by the property that they are the nodes in the graph which belong to only one edge. Since the graph structure abstracts from concrete values of calculated points, it is a scheme that assembles exactly the qualitative properties of the dynamical system in the sense of Chap. 1.2. Figure 3.1 shows an adapted version of Fig. 1.1 with already computed and not yet computed invariant sets and the corresponding 1-parameter graph.

In the case that several parameters $p_1, \ldots, p_k$ are varied, the branches of the special points constitute an analogue graph but now cusps etc. are the nodes. A part of this graph may be computed, and CANDYS/QA will store it in memory quite analogously. This graph will be referred to as k-parameter graph. For completeness, CANDYS/QA creates also 0-parameter graphs: the graph consisting only of the first computed point is an example.

Although the graph structure is an abstract image of the qualitative properties of the system, we label its nodes and edges by calculated values to use them for initializing later computations. In particular, CANDYS/QA stores these graphs in memory with the following information:

Nodes: The values of $\boldsymbol{x}$ and varied parameters $p_1, \ldots, p_k$; the kind of invariant set (i.e. steady-state or cycle), in the case of a cycle also its period or return time T; the kind of the point (i.e. type of special point or temporary endpoint); and the stability (neglecting the eigenvalues that pass the imaginary axis or the unit

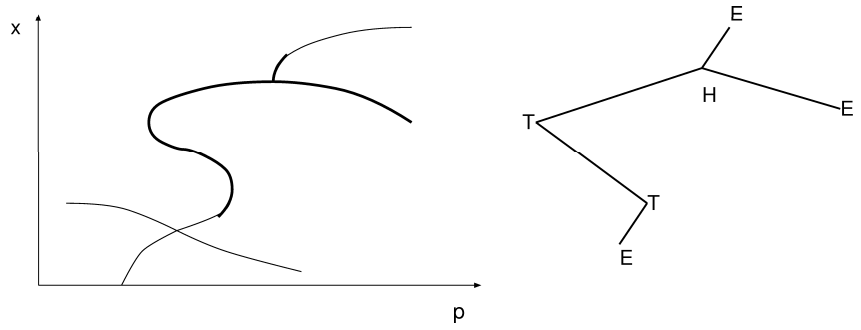


Figure 3.1: 1-parameter graph:

Left: thick lines: calculated invariant sets, thin and thick lines: all invariant sets;
Right: corresponding graph with E: temporary endpoints, T: turning points, H: Hopf-bifurcations.

circle).

Edges: The kind of invariant set, in case of cycles the period, the kind of the points on the arc, and the stability.

CANDYS/QA manages graphs by the following basic operations.

1. Creation of an empty graph. This operation occurs when the first point has been computed (creation of the 0-parameter graph) and when a new parameter is varied (i.e. creation of a $(k+1)$-parameter graph on the basis of a k-parameter graph). See also Sec. 3.3.
2. Creation of a new component of the graph consisting of exactly one node and no edge. This operation occurs, in general, together with the creation of a graph. See also Sec. 3.3.
3. Insertion of a node into one edge, i.e. the edge is split into two edges. This operation occurs when a special point has been computed: the special point is inserted into the current edge (i.e. the edge containing the current temporary end point).
4. Appending a new edge with a new node to an existing node. This operation occurs when the branches of a special point have been computed: as many new edges are appended at the special node, and the new nodes are the temporary endpoints on the branches.
5. Annihilation of two end nodes (i.e. nodes that belong to one edge only). This operation occurs when during the computations the current point meets a previously computed point of the same arc (i.e. in the case that a part of the same arc has already been obtained starting from the other end). In this case both endpoints are annihilated and their edges are united to a single one.

The application of the operations 1, 3, and 4 only yields a tree. Some trees are obtained if also operation 2 is used. Only the operation 5 can result in a loop within the graph,

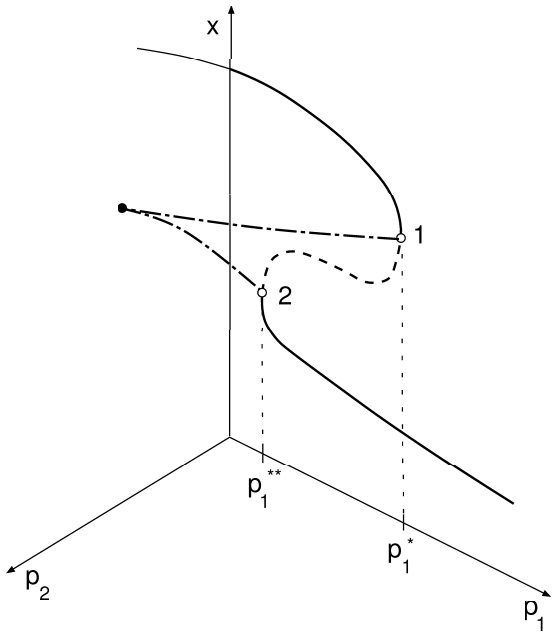


Figure 3.2: Bifurcation diagram with two varied parameters.
The turning point curve crosses the plane $p_2 = p_2^0$ repeatedly thereby inserting point 2 in the 1-parameter graph.
Solid curves: stable points, dashed curves: unstable points, dash-dotted curves: turning points, circles: turning points, filled dot: cusp point.

namely in the case that the two annihilated nodes belong to the same component. In the case that the nodes belong to two different components then these components are joint (without creating a loop).

3.2 The metagraph

The different k-parameter graphs are not independent of each other. Imagine the following situation (c.f. Fig. 3.2). During the path-following with one variable parameter p_1 and fixed parameter $p_2 = p_2^0$ a turning point is found at p_1^*. This point can be used for path-following of turning points in the p_1-p_2-parameter space. Then the new 2-parameter graph is in some sense related to the 1-parameter graph: an arc of the 2-parameter graph contains the special point, i.e. a node, of the 1-parameter graph. It can happen that the curve in p_1-p_2-parameter space turns and crosses once more the value p_2^0 when $p_1 = p_1^{**}$. Then this point is also a turning point in the 1-parameter graph. If this turning point did not yet occur then it should be inserted into the graph as a new node in a new component (since the node has no connections so far). The relations between different graphs will be studied in more detail in Sec. 3.3.

CANDYS/QA maintains the relations between several graphs in form of a directed metagraph: the nodes G are the common k-parameter graphs, and an edge from node G_1 to node G_2 means that G_1 is k-parameter graph, G_2 is a $(k+1)$-parameter graph, in G_2 are the same parameters varied (plus an additional one) as in G_1, and the values of fixed parameters of G_2 are the same as in G_1 (with the exception of the additionally varied parameter). In this case, we say that the $(k+1)$-parameter graph originates from the k-parameter graph.

The nodes and edges of the metagraph are labeled with the following information:

Nodes: The indices of the varied parameters and the values of the fixed parameters.

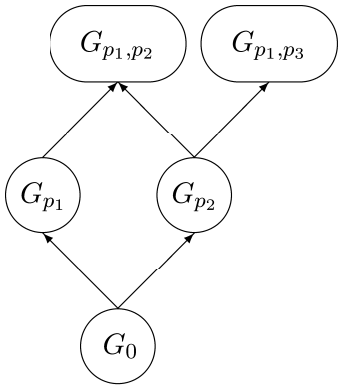


Figure 3.3: A metagraph of bifurcation graphs. The graph G_0 at bottom level contains the first point (with all parameters fixed) only. Then path-following starting at this point was performed with respect to the parameters p_1 and p_2, respectively, yielding graphs G_1 and G_2. Then special points in the graph G_1 are path-followed by variation of the parameter p_2. The graph $G_{1,2}$ is created with links to graph G_1 and to graph G_2. Finally, the bifurcation graph with respect to p_1 has been analysed by path-following with respect to the parameter p_3.

Edges: The index of the additional varied parameter in G_2 and the fixed value of this parameter in G_1,

Figure 3.3 shows an example of a metagraph.

The graphs and the metagraph are used in the following manner. If the path-following has stopped for some reason then the user has the possibility to traverse the graphs jumping from node to node and jumping from graph to graph to look for an interesting point for further investigation. The graph and metagraph structures are the appropriate book-keeping of the results obtained. Hence, it offers an easy switching from branch to branch to complete the bifurcation diagrams. Among all known software packages for the numerical treatment of qualitative properties only CANDYS/QA generates a graph structure corresponding to the paths already calculated.

3.3 Graph interactions

The contents of a k-parameter graph and that of a $(k+1)$-parameter graph originating from the former one are related to each other in the following manner: each special point in the former graph corresponds to a point on a curve in the latter graph. The program ensures that this theoretically given relation holds always in practice. This means, that during the path-following in one graph in specific situations also a few points or curves in another graph are computed. Moreover, these relations will be established when a new graph is created. How this will be done is the topic of this section.

In particular, we distinguish the following actions on the metagraph.

1. We consider the creation of a $(k+1)$-parameter graph: the parameters $p_1, \ldots, p_{k+1}$ are varied and the remaining parameters $p_{k+2}, \ldots$ are held fixed. The program traverses all k-parameter graphs which possess the same fixed parameters at the same values (at least one such graph exists). The new graph is linked to these graphs within the metagraph. Moreover, for each special point in these graphs a short curve (i.e. a component consisting of two nodes and one edge) is inserted in the new graph.

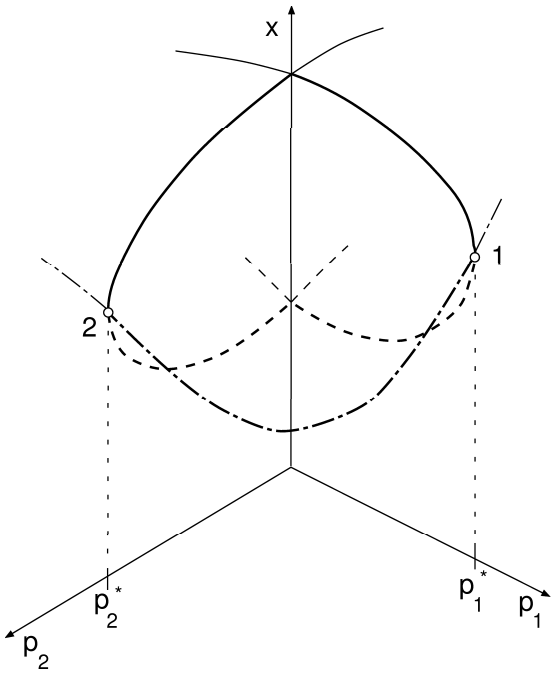


Figure 3.4: Bifurcation diagram with two varied parameters.
A turning point curve crosses the plane $p_1 = p_1^0$ thereby inserting point 2 in the corresponding 1-parameter graph if it exists.
Solid curves: stable points, dashed curves: unstable points, dash-dotted curves: turning points, circles: turning points, filled dot: cusp point.

2. Special points are treated similarly when they occur during the path-following in a k-parameter graph. In all $(k+1)$-parameter graphs originating from the actual graph a short curve of special points is inserted.

3. Things are much more complicated when path-following is done in a $(k+1)$-parameter graph. We suppose once more that the parameters $p_1, \ldots, p_{k+1}$ are varied and the remaining parameters $p_{k+2}, \ldots$ are held fixed. Each k-parameter graph from which our $(k+1)$-parameter graph originates has the same variable parameters with the exception of one, say p_1. The fixed value p_1^* of the exceptional parameter is known to our graph via the label of the link (c.f. Sec. 3.2). After each step of the path-following it is checked whether the curve has passed this value. If so then a special point at value p_1^* is computed and is inserted into the k-parameter graph. Moreover, short arcs of all branching-off curves are determined and appended. Now, the new special point has to be treated in the same manner as described above in case (2): in all $(k+1)$-parameter graph (with the exception of our graph where the curve already exists) short curves are inserted.

   The procedure works in the case $k = 0$ as well: one simply has to replace the phrase "special point" by "steady-state" or "cycle". This means, that the 0-parameter graph is completed by additional "first" points.

The method of maintaining the relations between graphs has still one gap. We will demonstrate it by examining the example of Fig. 3.3. Suppose that the basic graph has been established at the parameter values $p_1 = p_1^0$ and $p_2 = p_2^0$ (for simplicity, $p_1^0 = 0$, $p_2^0 = 0$). Furthermore, suppose that the graph G_1 (with $p_2 = p_2^0$ fixed, i.e. the right plane) has a special point, say a turning point, at $p_1 = p_1^*$ and, analogously, that graph G_2 has a turning point at $p_2 = p_2^*$. Finally, suppose that in graph $G_{1,2}$ both points are connected by a curve of turning points. Several orderings for the creation of the graphs are possible. We consider the scenario that graph G_1 is created first, then

graph $G_{1,2}$, and graph G_2 last. The graph $G_{1,2}$ is created on the basis of the turning point in graph G_1, and at the creation time a short curve of turning points through the point $p_1 = p_1^*$, $p_2 = p_2^0$ is inserted (action 1). Now, it is possible to continue this curve passing the value $p_2 = p_2^0$. If graph G_2 already existed then a turning point would be inserted (action 3). But this graph does not yet exist and nothing happens. If the graph G_2 is created sometimes later and its turning point is computed then a short curve is inserted in graph $G_{1,2}$ at the place of the curve obtained before (action 2). The continuation of this curve would lead back to the point $p_1 = p_1^*$, $p_2 = p_2^0$ and to the replication of that special point in graph G_1. The procedure could be continued forth and back to infinity.

CANDYS/QA avoids the gap the following way. When a $(k+1)$-parameter graph is created by making the parameter p_{k+1} variable then to all existing graphs (with the same values of the fixed parameters $p_{k+2}, \ldots$) a new graph is created (if it does not yet exist) that has the additionally varied parameter p_{k+1}. Thereby all the operations described above become active if necessary to maintain the relations between the graphs. In the example this means, that the creation of graph $G_{1,2}$ stimulates the creation of graph G_2. Thus, the relation between the graphs p_2 and p_1, p_2 are always maintained. This is demonstrated in Fig. 3.5.

(a) The first point S1 is computed when no parameter is varied.

(b) Parameter p_1 is varied: graph G_1 is created and a short curve through S1 is inserted.

(c) Path-following in graph G_1 yields a special point, say the turning point T1, and then the point S2 when the curve crosses p_1^0 the second time.

(d) Turning point T1 becomes object of path-following with respect to parameters p_1 and p_2: graph $G_{1,2}$ is created and a short curve of turning points is inserted. Now, the addional activity mentioned takes place. Together with $G_{1,2}$ also the graph G_2 is created where parameter p_2 is varied and short curves through S1 and S2 are inserted.

(e) Path-following of the turning point curve crosses the plane $p_1 = p_1^0$: truning point T2 and its two branches are inserted in graph G_2.

(f) Graph G_2 is completed: the short curves are joint together. Turning point T2 is not computed repeatedly since the path-following starting at the short curves through S1 or S2 stops before T2 is reached.

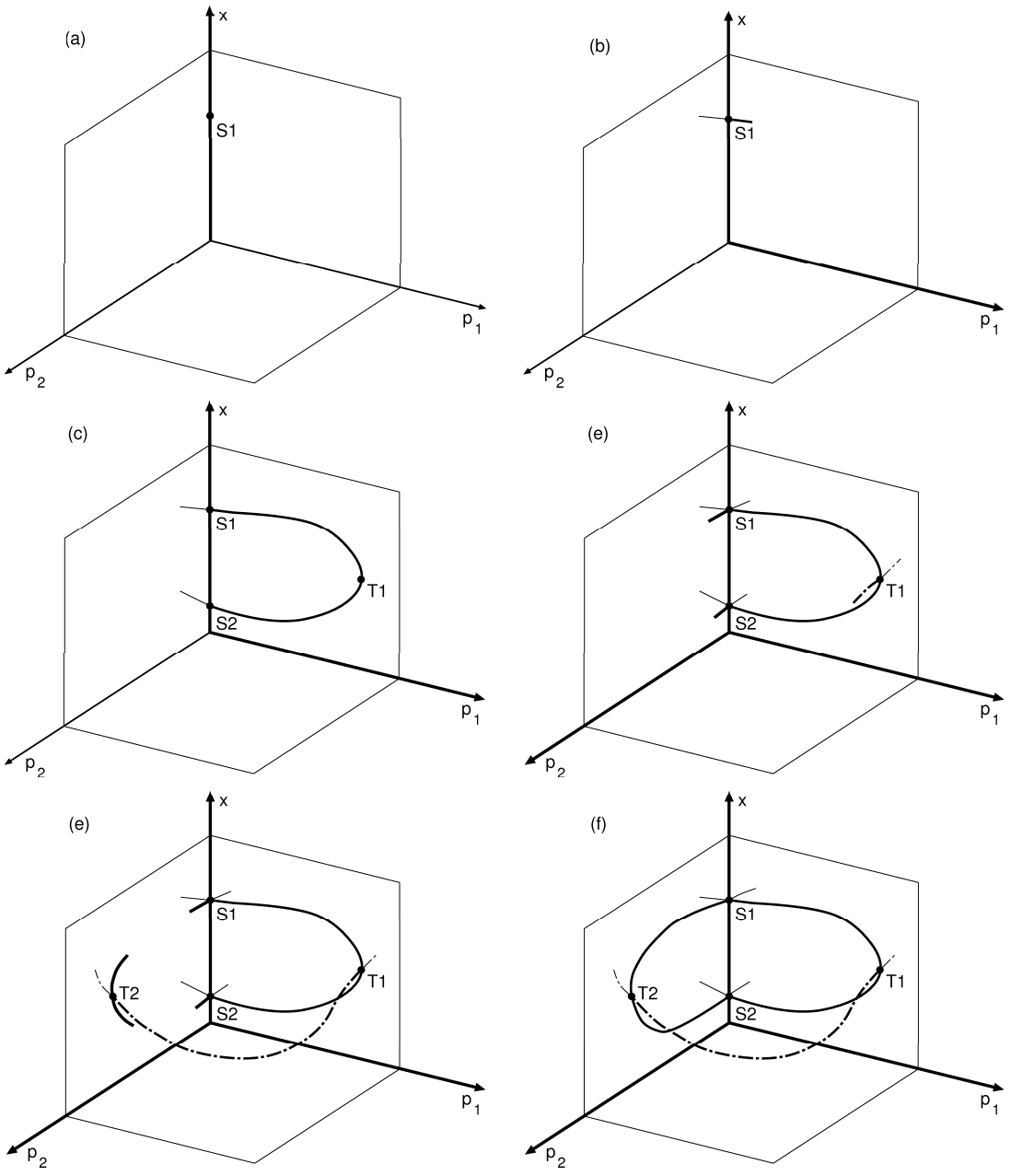


Figure 3.5: Relations between bifurcation graphs; for details see text.

Chapter 4

Applications and Example

The software system CANDYS/QA has been widely tested with many different model systems known from the theory of nonlinear dynamical systems (e.g. the Lorenz system, the Duffing oscillator, the logistic map, the Feigenbaum map, and many others) to assure the reliability of all algorithms. For testing all the algorithms which calculate special points, specific models have been constructed.

Instead of presenting some test examples which reflect special difficulties of the numerical treatment of nonlinear systems (e.g. homoclinic orbits, extremely stiff problems), we would like to demonstrate the facilities of CANDYS/QA by an example out of the variety of cooperations with other scientists. The program system has been applied in many different disciplines of natural sciences. In most cases the dimension of the model system is greater than 3 where the analytical computation of invariant sets and their bifurcation phenomena is, in general, impossible. As examples we would like to mention the study of the qualitative behavior of

- a modified Oregonator model of the Belousov-Zhabotinski reaction (Dim=3, [28]);
- an ecological model for the interaction of nutrients and micro-organisms in a river (Dim=7, Eidner et al. [10], [9]);
- a dynamo model for solar activity (Dim=7, [12]);
- a model for the dynamics of the photon-exciton interaction in a semiconductor-optical device (Dim=5, [48]).

Since the last mentioned model shows a lot of different bifurcation phenomena, we have chosen this model to illustrate the results which can be obtained by CANDYS/QA. The model consists of the following 5 autonomous differential equations:

$$\begin{aligned}
\dot{x}_{1r} &= -x_{1r} - \delta_1 x_{1i} + (g_{1i} + g_{2i}x_3)x_{2r} + (g_{1r} + g_{2r}x_3)x_{2i} + g_3(x_{1r}^2 + x_{1i}^2)x_{1i} \\
\dot{x}_{1i} &= -x_{1i} + \delta_1 x_{1r} - (g_{1r} + g_{2r}x_3)x_{2r} + (g_{1i} + g_{2i}x_3)x_{2i} - g_3(x_{1r}^2 + x_{1i}^2)x_{1r} \\
\dot{x}_{2r} &= -(g_{1i} + g_{2i}x_3)x_{1r} + (g_{1r} + g_{2r}x_3)x_{1i} - \sigma_2 x_{2r} - \delta_3 x_{2i} + d \\
\dot{x}_{2i} &= -(g_{1r} + g_{2r}x_3)x_{1r} - (g_{1i} + g_{2i}x_3)x_{1i} + \delta_3 x_{2r} - \sigma_2 x_{2i} \\
\dot{x}_3 &= -2\sigma_3 2x_3 + g_4(x_{1r}^2 + x_{1i}^2)
\end{aligned} \tag{4.1}$$

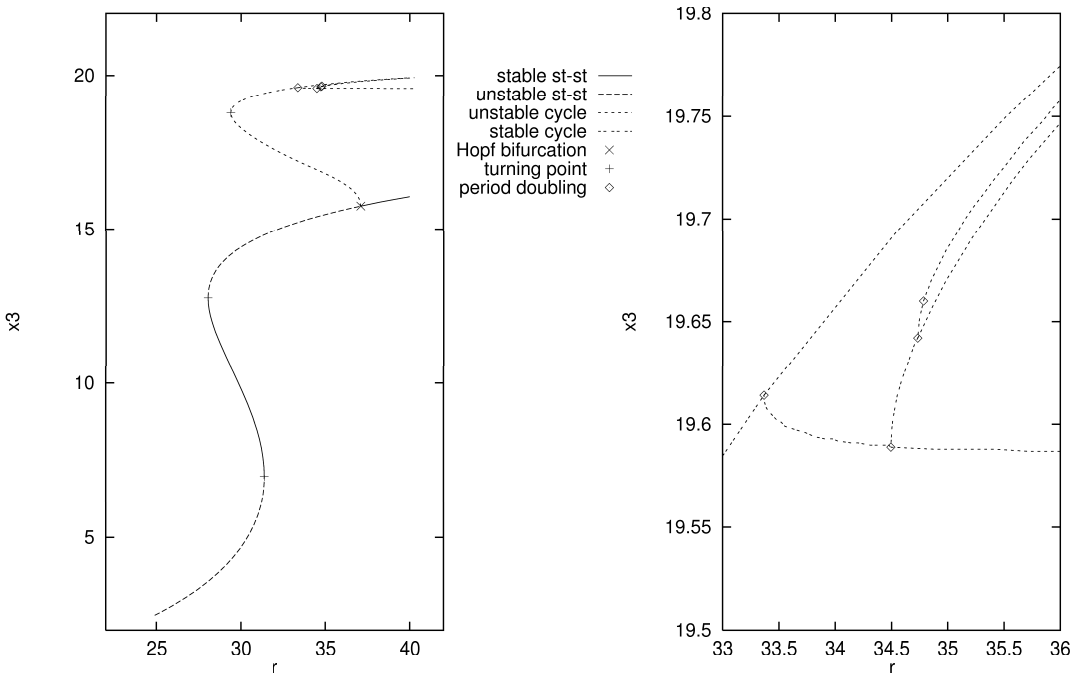


Figure 4.1: Bifurcation diagram of the system Eq. (4.1)
left: steady state with two turning points and a subcritical Hopf bifurcation as well as a turning point and the first period doubling of a cycle

right: period doubling cascade of the cycle, for clarity, only one branch is shown for each cycle

where
$\delta_1 = 0.0,\ g_{1r} = 0.5,\ g_{1i} = 40.0,\ g_{2r} = 0.01,\ g_{2i} = -2.0,$
$g_3 = 5.0,\ g_4 = 5.0,\ \sigma_2 = 10.0,\ \delta_3 = 0.0,\ \sigma_3 = 1.0\,.$
For the physical details c.f. [48]. From the physical point of view, the crucial bifurcation parameter is d which acts as a constant driving force of the system. Figure 4.1 shows a part of the bifurcation diagram in dependence on this parameter. From Eq. (4.1) it is obvious that in the case $d = 0.0$ the trivial solution $x_0 = \cdots = x_4 = 0.0$ is a stable stationary point of the system. This point should be used as an initial point for our study. Continuing this steady-state in dependence on d it loses its stability at $d^1 = 31.39$ which corresponds to a turning point. The unstable steady-state becomes again stable at a further turning point at $d^2 = 28.04$. At $d^3 = 37.10$ a Hopf bifurcation occurs. The steady-state loses its stability and a periodic solution arises. But this cycle is not stable, it is born through a subcritical Hopf bifurcation. The path-following of this unstable cycle with respect to d leads to a turning point at $d^4 = 29.41$ where the stability of the cycle changes. With further increase of d a period doubling cascade is obtained which ends up in a chaotic behavior of the system. All occurring special points are given in Table 4.1.

To demonstrate the computation of curves of bifurcation points in the two-parameter space we continued the turning points, the Hopf bifurcation, and the first period doubling with respect to σ_2. Figure 4.2 shows the results in the d-σ_2-space, the bifurcation points are listed in Table 4.2. The bifurcation diagram in Fig. 4.1 corresponds to a horizontal straight line in the parameter space at $\sigma_2 = 10$. The path-following of the

Type	Invariant set	d
Turning point	steady-state	31.388010
Turning point	steady-state	28.038182
Hopf bifurcation	steady-state	37.104448
Turning point	cycle	29.406514
Period doubling $T \to 2T$	cycle	33.367806
Period doubling $2T \to 4T$	cycle	34.495087
Period doubling $4T \to 8T$	cycle	34.732739
Period doubling $8T \to 16T$	cycle	34.784376

Table 4.1: Special points of the system Eq. (4.1) for one varied parameter, d.

Type	Invariant set	σ_2	d
Takens-Bogdanov point	steady-state	-6.0976511	22.457493
Takens-Bogdanov point	steady-state	-0.0003434	1.3990241
Takens-Bogdanov point	steady-state	-0.10405281	6.0487320
Cusp point	steady-state	18.740908	32.345606
Cusp point	steady-state	-22.108210	30.653020
Cusp point	steady-state	-0.0278109	2.0691930
Turning + Hopf bif.	steady-state	-1.7663408	30.726784
Turning + Hopf bif.	steady-state	-2.3593780	30.704463
Turning + Hopf bif.	steady-state	4.1475056	20.877494
Turning + imag. Hop bif.	steady-state	-3.0808319	17.528581
Hopf bif. + Hopf bif.	steady-state	-4.0826534	24.588115
Takens-Bogdanov point	cycle	-0.00539035	1.0290856
Cusp point	cycle	-3.0006375	3.4734214
Turning + period doubling	cycle	0.39834749	2.4759549
Turning + period doubling	cycle	-0.47640402	0.26807277
Turning + torus bif.	cycle	-2.8980994	3.5785710
Turning + torus bif.	cycle	-2.5045339	10.063238
Turning + torus bif.	cycle	-2.5570243	4.9371809

Table 4.2: Special points of the system Eq. (4.1) for two varied parameters, σ_2 and d.

curves started at this straight line. Then the curves undergo many bifurcations, new curves are born, and some curves meet forming a loop in the bifurcation graph.

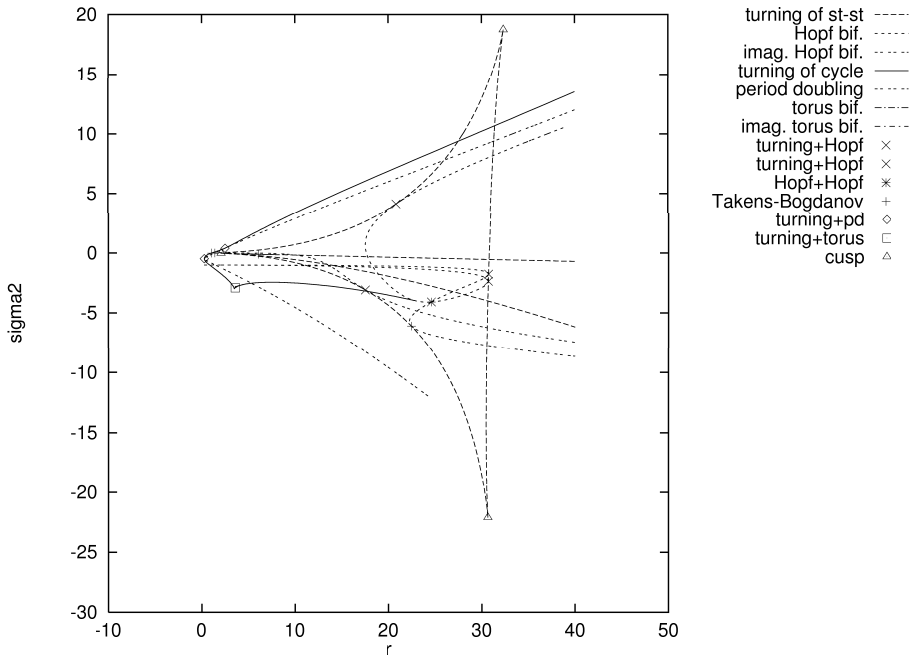


Figure 4.2: Curves of bifurcation points of the system Eq. (4.1).

Part II

Users' Manual

This part describes how to use the program system CANDYS/QA. This includes:

- program installation
- preparation of the right hand sides etc. of the model
- user input
- program output
- use of graphics systems

There is also a detailed example that demonstrates input and output.

For a first reading, focus on the Secs. 7 and 8 where the preparation of the model and the program interactions, respectively, are described.

Chapter 5

About the Program

5.1 License and warranty

Use and copying of this software and the preparation of derivative works based on this software are permitted, so long as the following conditions are met:

- The copyright notice at the beginning of each file and this entire notice are included intact and prominently carried on all copies and supporting documentation.
- No fees or compensation are charged for use, copies, or access to this software. You may charge a nominal distribution fee for the physical act of transferring a copy, but you may not charge for the program itself.
- If you modify this software, you must make the modified file(s) to carry prominent notices (a Change Log) describing the changes, its author, and the date of the changes.
- The source code, except header files and those explicitly granting permission, may not be used in commercial software without consent of the author.
- Any publication of scientific results obtained by this software must cite it appropriately.

This software is made available as is, and is distributed without warranty of any kind, either expressed or implied. The author does not accept responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all. No warranty is made about the software or its performance. In no event will the author or his institutions be liable to you for damages, including lost profits, lost many, or other special, incidental or consequential damages arising out of or in connection with the use or inability to use (including but not limited to loss of data or losses sustained by third parties or a failure of the program to operate as documented) the program, even if you have been advised of the possibility of such damages, or for any claim by any other party, whether in an action of contract, negligence, or other tortuous action.

The author's current address is

Wolfgang Jansen
Universität Potsdam
Fachbereich Physik
Am Neuen Palais, Haus 19 (Physik)
D-14415 Potsdam

e-mail: wjansen@agnld.uni-potsdam.de
internet: http://www.agnld.uni-potsdam.de/ wolfgang/wolfgang.html

5.2 Features of implementation

CANDYS/QA is based on the following features of implementation:

- The program is restricted to the facilities mentioned in Part I. In particular, there are no procedures for computing Lyapunov exponents and fractal dimensions which are characteristics of chaotic systems. The restriction to the three system families described in Sec. 1.1. has been inspired by the applications of CANDYS/QA. But due to the strict modularity of the program it is not difficult to include other system families (only the procedures generating the appropriate algebraic equations have to be modified).

- The only programming activity of the user is the preparation of a module describing the dynamics of the system. To overcome the programming efforts caused by the restrictive format of this module, it is possible to obtain it by MATHEMATICA© in a much more user friendly way.

- The number of the model's state variables and parameters is not bounded.

- The numerical procedures involving matrices (e.g. solving linear equation systems or computing eigen values) treat dense matrices only. The reason is that those matrices occur naturally during the computation of cycles of differential equations, i.e. they cannot be avoided.

- In numerical procedures real numbers are always coded as `double` while integer numbers (e.g. loop counters) are coded as `int` with the exception of the use of `long` local to one single module. The graphical interface uses `float` for the representation of real numbers.

- The program is fully interactive in its use, i.e. the user is prompted for all controlling information such as the choice of a solution branch for continuation or the numerical control parameters (e.g. integration accuracy), where always a default value is offered.

- The program produces graphical output if certain technical prerequisites (hardware and software) are given. First, there is a graphics module coming with CANDYS/QA that works if JAVA is available (c.f. Sec. 11.2.1). Second, other graphics systems can be used (c.f. Secs. 11.2.2 and 11.3.1).

- The program system is accompanied with a version number composed of four entries:

  1. a general version number;
  2. the version of the algorithms implemented;
  3. the version of the structure of the restart files;
  4. the version of the structure of the graphics files.

  The two last entries (together with the first one) serve as correctness checks of files produced by CANDYS/QA when they are read by CANDYS/QA itself or by a post-processor. These files cannot be read if they have been produced by incompatible versions. The current version is 4.4.6.4.

5.3 Version changes

Version 4.3.6.4

- All known bugs have been fixed. [2] In particular, the program implements now its own module for string handling. This was necessary because the string handling modules coming with different compilers tend to become more and more different.
- If during path-following a special point has been detected but it or any of its branches could not be computed for some reasons then the previous version issued one or more warnings and continued the path-following. In version 4.3.6.4 additionally to the warnings an error is issued and the path-following stops.
- CANDYS/QA provides now its own online graphics module on the bases of java. The implementation in JAVA has been chosen because of the strong portability (particularly, the graphics) of that programming language and because JAVA is nowadays widely available.

Version 4.4.6.4

- All known bugs since version 4.3.4.6 have been fixed.
- The integration of stiff ODE has been improved.
- The select option n (where n is an integer) had two meanings: (c.f. Sec. 8.4) making a node actual by its number n as well as switching to the scenario level if n=`0`. This has been changed as follows. Select option `0` switches to the scenario level while select option `"` (quotation mark) makes a node actual. In the latter case the program prompts for a node number.
- The input of `EpsPredic` (c.f. Sec. 8.2.4) has been discarded. Now, that control variable is computed internally. This has the side-effect that the program often makes larger steps during path-following. To avoid too large steps the control variable `MaxCurveStep` should be set to smaller values.

[2] I thank all users of CANDYS/QA who sent a bug report.

- Optionally, CANDYS/QA can utilize the BLAS subroutines and the LAPACK [3] numerical library. Using these subroutines may drastically accelerate the computations, in particular in case of high dimensional models. This feature becomes available if LAPACK and the BLAS have been installed on your platform and if you modify the `makefile` accordingly.
- Starting with this release, the libraries will be installed as shared objects (`*.so`). This causes that the executable files will be much smaller than before. Moreover, when future releases become available only the shared objects have to be reinstalled, the executable programs are then up-to-date automatically.

The format of restart and graphics files have not been changed. This means, investigation of a model started with version 4.2.6.4 can be continued without any problems.

[3] LAPACK is distributed by *netlib*, see http://www.netlib.org/lapack/

5.4 Installation

This section describes the installation of CANDYS/QA for computer platforms running under the operation system UNIX or one of its descentants.

5.4.1 Hardware and software needs

Device capacity :

Source code	0.7 MBytes
Manual	2.8 MBytes
Shared objects	0.6 – 1.7 MBytes
Executable program	15 – 40 kBytes

The last numbers depend strongly on the computer and compiler.

Main memory : 4 – 12 M-Bytes of resident memory size for the running program (but this strongly depends on the computer and compiler, the kind of online graphics, and on the amount of the dynamically allocated memory).

Operations system : UNIX or any of its descendants is preferable, under other operating systems CANDYS/QA runs with a few restrictions.

Compiler : Probably any *C++* compiler (e.g., *g++* version *2.7.2*, the *cxx* version *6.0* of *DEC OSF*, and the *CC* version *7.2.1* of *SGI IRIX*.)

C **prerequisites** : The header files `ctype.h errno.h, float.h, limits.h, math.h, stdio.h, string.h, time.h`, and `unistd.h` from the standard *C* package (usually located in directory `/usr/include`) and possibly `mdefs.h` from MATHEMATICA©.
The libraries `libc.a` (or `libc.so`) and `libm.a` (or `libm.so`), usually located in directory `/usr/lib`.

JAVA **prerequisites** (optional): Version *1.1.6* of JAVA (probably any *1.1.** version will work); under *Linux* version *1.2* is needed.

5.4.2 Installation of shared objects and the post-processor

The installation described refers to UNIX or any of its descendants.

1. Create a directory where the sources of CANDYS/QA should reside. Switch to that directory.
2. Extract all files from the *tar* or *zip* file(s). Several subdirectories are created with the following contents:

   `bin` : empty, will contain the graphical post-processor

   `doc` : this manual `Candys.ps`

   `exa` : the example file `Lorenz.cxx`

   `lib` : `og.zip` and the files `*.lmf` to be used by several graphics modules, will also contain the shared objects

`math` : the files `Candys.m` and `correctmath` for the use of MATHEMATICA©

`spec` : subdirectories containing the source code files `*.cxx` special for several computers and/or compilers; currently only the subdirectory `sgi` for *Silicon Graphics* computers exists

`src` : the source code files `*.h` and `*.cxx` and the `makefile`

3. Adapt file `src/makefile` to your needs.

   (a) Choose compiler as well as compiler and linker options (below the comment "`#--- Choice of compiler etc. ---`").

   (b) Choose mathematical libraries (below the comment "`#--- Choice of math libraries ---`"). If BLAS and LAPACK subroutines are available then you should uncomment the line "`#CXXFALGS= $(CXXFLAGS) -DBLAS}`" and/or the line "`#CXXFALGS= $(CXXFLAGS) -DLAPACK}`". This ensures that the programs invoke the corresponding subroutines instead of CANDYS/QA's own functions. Further, uncomment the line "`#MATHLIB = ...`" that corresponds to your platform (or define a new one). Possibly, you must include a `-L` option.

4. If appropriate, copy the files from the corresponding subdirectory of `spec` to directory `src` (refer to file `README` there).

5. Create the shared objects and the post-processor by entering the command
   `makecandys`
   This creates besides the shared objects `lib/lib*.so` the post-processing program `bin/g2gnu`. The shared objects are configered such a way that they include

   - the JAVA based online graphics if the shell command `java` is available (and if the environment variable *JAVA0* has not been defined)
   - else the PGPLOT based online graphics if the environment variable *PGPLOT_DIR* has been defined
   - otherwise no online graphics.

The shared objects and the post-processor have to be re-installed in the following situations.

- When a new release of CANDYS/QA becomes available.
- When the CANDYS/QA main directory or (if the OG graphics is included) when the JAVA main directory have been moved to another location. In that case remove the shared objects `lib/lib*.so` and the post-processor `bin/g2gnu`, then enter command
  `makecandys`
  Re-installation is necessary since the locations of shared objects are "hard wired" in depending shared objects and the post-processor.

5.4.3 Installation of an executable program

1. Switch to the directory where your model and the result files should reside.
2. Prepare the source code of your model and place it into that directory. Henceforth, the name *Model* denotes the file name of your module without the extension `.cxx` of this file.
3. Possibly, prepare your own module *Handle*`.cxx` and compile it. Then define the environment variables (multiple strings must be enclosed in quotation marks)
   `setenv HANDLE` *Handle* : name of the module (without extension)
   `setenv HDLLIB` *lib_opts* : linker options for additional libraries;
4. Enter command:
   `$CANDYDS_DIR/makecandys` *Model*
   where `$CANDYDS_DIR` denotes the CANDYS/QA main directory. The result is the executable program *Model* in your working directory.
5. Copy the appropriate `*.lmf` files from
   `$CANDYS_DIR/lib/*.lmf`
   to your working directory.

An executable program has to re-installed if the CANDYS/QA main directory has been moved to another location since the locations of shared objects are "hard wired" in the program.

5.4.4 Running the executable program

1. Switch to the directory where your *Model* program resides.
2. Enter the command
   Model [options] [parameters]
   The *options* control the use of graphics modules. The *parameters* are forwarded to the constructor of your *Model* class, they do not influence CANDYS/QA directly.

Chapter 6

The Course of a Qualitative Analysis

6.1 Scenarios

The study of a given model by CANDYS/QA is performed within the shell of so-called scenarios. In a scenario all results are summarized that have been obtained from a certain starting point. Each scenario gets a number (from `01` to `99`) as well as a *ScenarioName* that is composed of

1. the *ModelName* (as provided by the user in the constructor `Model::Model(int argc,char **argv)` for the model, c.f. Sec. 7.2),
2. the underscore character,
3. the scenario number.

All files produced by CANDYS/QA contain this name as part of the file name, thus they can easily be distinguished.

On the other hand, scenarios provide also the frame to continue the work of former computations on the basis of restart-files. Each time when the user wishes, a restart-file is written that saves the complete memory resident information. More precisely, the following is performed. CANDYS/QA writes output to several files corresponding to the irregular scenario number `00`. When a specific program state is reached the user is asked whether the results (since the last saving) shall be saved. If the answer is "yes" then the restart file is written (also few other files) and (most) other files are appended to the corresponding ones with the current scenario number. But if it is "no" then the files with scenario number `00` are deleted and the old restart file is read in (i.e. the previous state is restored, the latest results are lost). On user request, a restart-file is written also at program termination what opens for the possibility to recover the program state in a later computer session.

The rejection of the results obtained last and restoring of the previous ones can be necessary if in the case of an error (e.g. non-convergence of a numerical procedure) the computations are to be repeated (e.g. with other numerical control parameters).

Moreover, the intermediate files with scenario number `00` will not be deleted as long as no new computation is done, i.e. these files can be examined to analyze the error.

Note

The use of the files with a special scenario number, `00`, has the following consequence: only one version of CANDYS/QA can run the at same time in a given working directory. If the program should be run several times (e.g. in several windows) then each process has to be started from another working directory. This way, each process writes to other files.

6.2 Loops of work

The work of CANDYS/QA can be seen as several loops which are controlled interactively by the user.

- Initialization.
  A first steady-state or cycle (henceforth summarized as points) for a fixed parameter vector is computed. It is the first element of the set of known points.
- First loop.
  Starting at an already known point this point is continued by varying one chosen parameter p_1 in some interval. Special points possibly existing on the curve (i.e. turning points, bifurcation points etc., c.f. Sec. 2.4.1) are detected and calculated together with a short portion (a stump) of the curves branching off. The endpoints of the current solution branch as well as those of the intermediately calculated stumps are being included into the set of known points, while the special points themselves are being included into the set of known special points. This way, a graph (i.e. the 1-parameter graph of Sec. 3.1) is constructed whose nodes are the known points, and whose edges represent the connecting solution branches.
- Further loops.
  One can also start at a known special point, thus computing a solution branch of special points with respect to a further parameter p_2. Here, also special points (e.g. cusp points) can be detected and computed. This way, another graph, namely a 2-parameter graph (more generally: a $(k+1)$-parameter graph if the special point belongs to a k-parameter graph) in the sense of Sec. 3.1 is created which is linked to the graph containing the special point. (We say, the new graph originates from this graph);

Besides this, there is also another view onto the mechanism of CANDYS/QA. Since there is the possibility to stop at an arbitrary point and to start from the very beginning, and also to switch freely between the loops the natural view onto the control structure of the program is as follows. There are five levels of program and user activities.

- The scenario level plays the role of a supervisor. This level is entered at the start of the program, here the other levels are activated depending on possibly already

computed curves (more precisely, depending on the existence of graphs), and the program can be terminated.

- The methods level serves for computing a first point by a appropriately selected method (this is the initialization mentioned above).
- If at least on graph exists then one node can be selected in the node-selection level for continuing the bifurcation graphs.
- The several loops mentioned above are summarized as the path-following level, since they are not so different in view of the methods used and of the user's interaction. This level supposes the existence of at least one (possibly rudimentary) graph of computed nodes (c.f. Sec. 3.1).
- The graph-manipulating level creates a new graph and relates it to the existing ones.

6.3 Treatment of constant and extreme quantities

At most one quantity can be maximized or minimized in a scenario. Thus, scenarios differ not only in their starting point but also in that quantity which has to be specified when a new scenario is established. The quantity may be a state variable or one of the functions provided by the procedure `QuantRhs` (c.f. 7.2).

Much more has to be said about quantities with constant value. As opposed to special points and extreme values which are uniquely determined by the system or a quantity function, the constant quantities are arbitrary values. In the following it will be described how the program has to be controlled such that a quantity gets a desired value.

Each path-following can be restricted interactively to admissible intervals for all varied parameters and all monitored quantities. Thus, one sets the interval limits of the quantities to interesting values. When the path-following stops there the user can clone the temporary endpoint. The copy gets all information of its template with one exception: it is flagged to be a special point, a point of constant quantity. What quantity is held constant is defined interactively, its fixed value is simply set to the current value.

The new point is inserted into the graph as a node on the edge leading to the node corresponding to the temporary end point. When the curve at the temporary end point is continued then the new point belongs also geometrically to the arc from the previous point to the new position of the end point.

The following quantities can be held fixed:

- the state variables,
- the values of the functions provided by the procedure `QuantRhs`,
- in the case of cycles of autonomous systems the period length of a cycle,
- in the case of torus bifurcations the angle of the critical eigenvalue (c.f. Sec. 1.4.3).

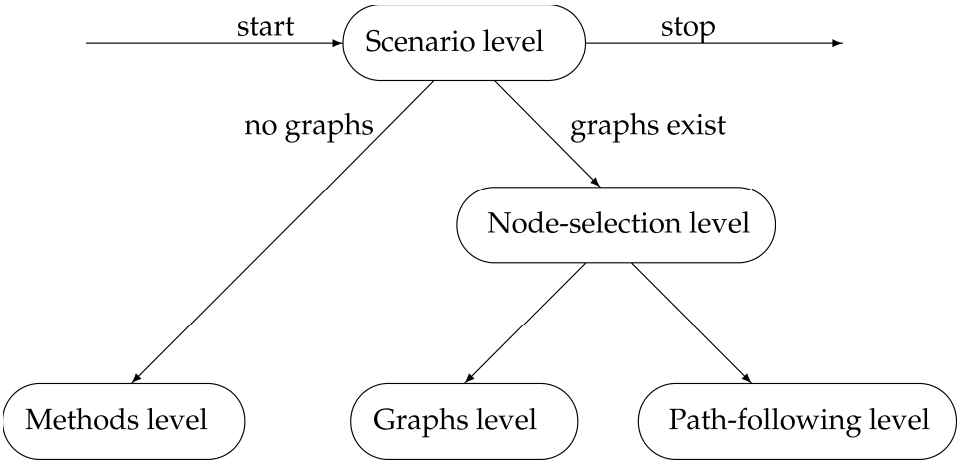


Figure 6.1: The activity levels of CANDYS/QA.

6.4 The scenario and node-selection levels

The scenario level is entered at the program start and each time when the methods or node-selection levels are left. In the latter case the user is asked whether or not the results obtained (since the previous visit of the scenario level) are to be saved. If so, then an updated restart-file is written. Otherwise, the previously saved situation will be restored from the existing restart file.

After saving of the new results or restoring of the old results there are three options for continuation:

1. continuation of the current scenario,
2. starting a new scenario,
3. exiting CANDYS/QA.

In case (2) as well as after program start, the user is asked for a scenario number.

Depending on the success of the computations (if any) within the following may happen:

1. no point has been computed so far (e.g. after start of a new scenario): the program switches to the methods level;
2. at least one graph exists: the program switches to the node-selection level.

The node-selection level can be seen as a special part of the scenario level. It allows the user to traverse all existing graphs looking for an interesting node (i.e. an interesting special point or a temporary end point). Depending on the choice (c.f. Sec. 8.4) the program

1. switches to the path-following level for continuing a curve of points;

2. switches to the graph-manipulating level for creating a new graph;
3. goes back to the scenario level.

6.5 The methods level

The computation of a first point is a complicated task. Therefore, CANDYS/QA provides five numerical methods to compute one (c.f. Sec. 2.4.4) as well as the possibility to try several methods consecutively, e.g. to improve the result of one method by another one. The work in the methods level is implemented as a loop consisting of the following steps.

1. Input of the following entities:
   (a) in the case of autonomous systems the choice of the kind of the invariant set (i.e. between a steady-state and a cycle), in the case of the computation of a cycle also the Poincaré surface has to be chosen,
   (b) the fixed parameter values,
   (c) initial values of the state variables,
   (d) in the case of the computation of a cycle the choice of its length (the user is supported in her decision in the following way: the program simulates some iterates from the initial point and outputs the iterated points as well as the error of the equations system for cycle computation).
2. Choice among one of the algorithms:
   (a) Newton's method;
   (b) homotopy method;
   (c) Monte-Carlo search;
   (d) evolutionary search;
   (e) simulation.
3. The program activates the chosen algorithm and prints its result.
4. The user is asked for choosing among the following options:
   (a) if the computation was successful: whether to use this point as first point of the path-following, if so then the methods level is left;
   (b) input of new values, continuation at (1);
   (c) try another method with the same initial point, continuation at step (2);
   (d) try another method with the last result as new initial point, continuation at step (2);
   (e) exit the methods level without success.

Note

If one selects option (4a) then graph 1 is created consisting of one node only: the result of the method. This node receives the number 1.

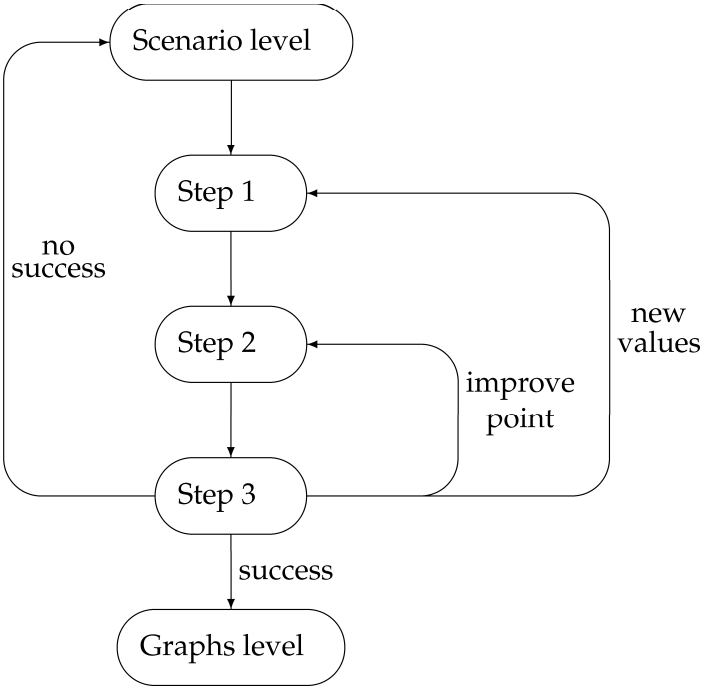


Figure 6.2: The methods level

6.6 The graph-manipulating level

The graph-manipulating level offers two features: creating a new graph and appending new nodes to an existing graph.

1. A new $(k+1)$-parameter graph is created on the basis of the special points in a k-parameter graph, or on the basis of the first point in the 0-parameter graph. To this end, the user has to choose a parameter for variation and a distance. Then the program computes two points to the left and right (with respect to the chosen parameter) of each special point (or of the first point) at the chosen distance. If the computation is successful for both points then a graph is created consisting of all computed nodes and the connecting edges, and the level is left. Otherwise, the level is left without success.

2. At any temporary endpoint of a $(k+1)$-parameter graph one of the varied parameters can be replaced by one fixed parameter, e.g. the varied parameter p_1 becomes fixed at the current value and p_2 is varied instead. In this situation two new graphs are created: one k-parameter graph where p_1 is fixed (if the graph does not already exist) and another $(k+1)$-parameter graph with p_2 as the additionally variable parameter. The new k-parameter graph is filled with the chosen temporary endpoint as first point (if $k = 0$) or a special point (if $k > 0$), in this case also stumps of the branching off solutions are appended. On the basis of this k-parameter graph the new $(k+1)$-parameter graph is filled in the sense of the previous item.

3. At any temporary end point a point with constant quantity can be added. To this end, a temporary endpoint has to be selected (c.f. Sec. 6.3).

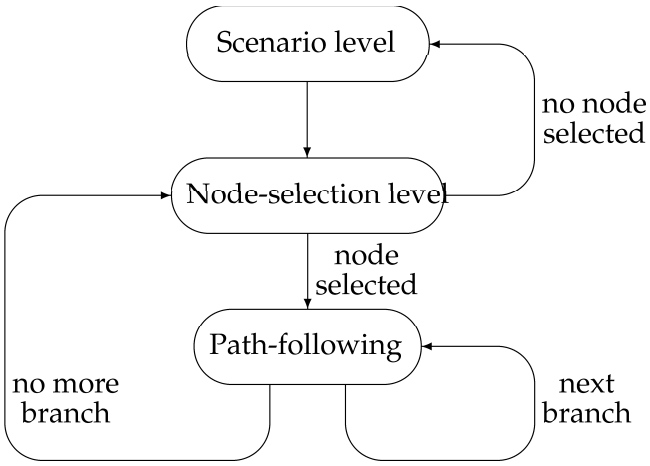


Figure 6.3: The path-following level

If $k > 1$ then in the case of the first to items also all other (not yet existing) graphs are created whose set of variable parameters is a subset of the corresponding set of the $(k+1)$-parameter graph.

All graphs and nodes are numbered consecutively. The graph number is used to construct the extension of the graphics and connection files (c.f. Sec. 9.2) whereas the node number is used for referencing a specific node during the dialogues.

6.7 The path-following level

The path-following is performed by one of the two concepts described next.

1. Continuation of a single branch only: the path-following will be started at a selected temporary endpoint. When an aborting criterion is fulfilled (see below) then the program switches back to the node-selection level.

2. Automatic continuation of all branches: the path-following is performed starting from each temporary endpoint until an aborting criterion is fulfilled. If during the work new branches, i.e. also new temporary endpoints, are created then also these branches are included into the process of path-following. This means that the full bifurcation graph (within the limits posed by aborting criteria) is built up automatically.

After each predictor-corrector step CANDYS/QA checks for special points between the current point and the previous one, also for several special points ahead the current point (c.f. Sec. 2.4.4). If a special point has been detected then the following actions take place.

1. The special point is computed as well as the corresponding stumps of branching-off curves (if any).

2. The special point is included into the graph lying on the current edge, and the endpoints on the new branches are appended to the graph as temporary endpoints linked to the special point by new edges.

3. If the special points belongs to a k-parameter graph and there are already $(k+1)$-parameter graphs originating from it then in each of them a new component is added consisting of one edge and two nodes corresponding to a stump of the curve in these graphs (c.f. Sec. 3.3).

If the special point lies between the current and the previous point then the path-following continues, otherwise it stops.

The program checks after each predictor-corrector step not only for special points. There are some more checks.

- Check whether a parameter limit is reached or whether an error has occurred. Then the path-following of the current branch stops.
- Check for another temporary endpoint lying between the current and the previous point. If such a point is found then this means that the same arc has already been computed from the other parameter direction. The path-following of the current branch stops after having joint the formerly computed and the current arcs. The two endpoints are deleted from the graph structure and the path-following stops.
- Check whether during the path-following in a $(k+1)$-parameter graph the curve crosses a parameter value that is fixed in one of the k-parameter graphs from which the current graph originates. In this case, into the k-parameter graph a new special point is inserted (or a new first point in the case of $k = 0$), also stumps of the branching off curves are computed and appended. Then the path-following in the $(k+1)$-parameter graph continues.
- Check for artifacts of cycles of autonomous systems (c.f. Sec. 2.1.2). If such a situation is detected then the user is asked for another Poincaré surface. The program continues by a new initialization of the path-following corresponding to the new Poincaré surface.

Note

The detection of already computed points on the current arc may fail because of too restrictive tolerance demands. This implies that the current arc is continued and the subsequent results, particularly the special points and branching off stumps computed later, are obtained once again. In an extreme situation, this can lead to an infinite graph. Therefore, care should be taken if the feature of automatic path-following is applied.

Chapter 7

The *Model* Module

7.1 General remarks

The user has to prepare a program module *model*`.cxx` describing her model. In the following it will be described how that can be accomplished. In Sec. 7.2 all its procedures are explained, and Sec. A contains a prototype of the module. Section 7.3 presents how MATHEMATICA© can be used to prepare the module. Finally, the software package contains the module `Lorenz.cxx` that can be used as a prototype.

In this chapter the following convention will be used: The parts which the user has to insert are written in **bold**, given parts in `teletype`, and comments in *italic* letters.

7.2 Model description

7.2.1 Includes and global variables

The program module has to contain the `#include` statements

```
#include "Model.h"
#include "LinAlgebra.h"
#include "Errors.h"
#include "math.h"
```

Further `#include` statements may be necessary.

7.2.2 Initialization part

The header file `Model.h` defines the class `Model`. The program module has to contain a constructor
`Model::Model(int argc,char **argv)`
and the destructor for that class. The program allocates only one variable of this class, i.e. the constructor is called exactly once, it has to initialize the class members. The

parameters `argc, argv` of the constructor are the same as in the call of `main`, i.e. the command line split into strings. These can be used to control the initialization from the command line. [2]

In this subsection we describe the constructor
`Model::Model(int argc,char **argv)`
thereby explaining all class members.

The system class of the model is defined as follows:

`ModelClass =` **autonomous** *autonomous systems of differential equations, or*
`ModelClass =` **periodical** *periodically forced systems of differential equations, or*
`ModelClass =` **difference** *autonomous systems of difference equations*

`Dim`: *number of state variables which have to be indexed from* `0` *to* `Dim-1`

`NbOfParams`: *number of variable parameters which have to be indexed from* `0` *to* `NbOfParams-1` *and named by* `P`

`NbOfQuantities`: *number of quantities which have to be indexed from* `0` *to* `NbOfQuantities-1`

`NbOfOutputs`: *number of state variables which will be printed as result output, see below for details*

`Oindex`: *indices of state variables to be printed, the indices range from* `0` *to* `NbOfOutputs-1`, *their values must belong to the range from* `0` *to* `Dim-1`

`ModelName`: *a model name, it is used as basis of file names*

`TName,XName,PName,QName`: *name or name fields of the model time, states, parameters, and quantities, respectively (i.e. strings of alpha-numerical characters or underscores starting with a letter)*

`TScale,XScale,PScale`: *scales for these variables (the names and scales are used only in input/output operations, it holds:* $\texttt{X}[i] = input/scale$ *and output* $= \texttt{X}[i] * scale$, *similarly for* `T` *and* $\texttt{P}[i]$*), see below for details*

`X0`: *initial values of the state variables*

`P`: *initial and current values of the parameters*

`hasJacobi`: **1** *or* **0** , *Boolean variable specifying whether procedures computing the Jacobian matrices have been implemented*

`stiff`: **1** *or* **0** , *in the case of a system of differential equations a Boolean variable specifying whether the equations are stiff*

Some additional explanations seem to be necessary.

- The command line parameters can be meaningful, among others, in the following situation. Imagine that the model is a system of ordinary autonomous differential equations which approximate a partial differential equation by discretization of the space variable(s). The number of model variables, `Dim`, depends on

[2] But most of the assignments in the constructor are overwritten when a restart file is read in, exception: the names of variables.

the discretization step-size, and the right hand side contains loops over the discretization steps. Thus, different discretizations can be obtained simply by variation of `Dim`. Providing `Dim` as command line parameter makes this very flexible.

- If the dimension of the system is large then the output of results would become very long. But often many of the variables are not of interest (e.g. in the case mentioned in the previous item). Thus, the number of variables to be printed can be restricted by `NbOfOutputs`, and `Oindex` lists the indices of the corresponding variables. The restriction of `NbOfOutputs` has several consequences. First, only the values of the variables listed in `Oindex` can be set interactively, and only they can be presented graphically. Second, also the number of eigenvalues printed is restricted to `NbOfOutputs` (since otherwise the output would become still long).

- The scales `TScale` etc. can be used for two purposes. First, the numerical procedures can yield reasonable results only if all variables are in the same order of magnitude. If, e.g., one variable is typically a thousand times larger than another one then it is wise to re-scale one or both in the model. The scales `TScale` etc. can be used to ensure that the original, unscaled values are printed. Second, the scales can be used to transform the results from the physical coordinate system (metrical coordinates, say) of the model to another one (inches, say).

- The items `XScale`, `PScale`, `X0`, and `P` are variables of the predefined class `Vector`, `Oindex` is a variable of the class `IndexVector` while `XName`, `PName`, and `QName` are variables of the class `StringVector`. Among others, these classes define the following functions (where `v` represents a variable of any of these classes):

  1. `v.zero(n);`
     allocates a vector of `n` elements of the corresponding type (i.e. `double`, `int` or `String`, respectively) and sets the elements to `0.0`, `0`, or to the empty String, respectively;
  2. `v[i] = x;`
     assigns `x` to the `i`-th element of `v`, as usually, `i` ranges from `0` to `n-1`.

  The function `v.zero(n)` should be applied to the vectors `X0` etc. to give them the correct length and zero initial values. After zeroing, the assignment should be used to give the vector elements new values if appropriate. The prototype of the `Model` module in Sec. A demonstrates how these functions can be applied.

7.2.3 Right hand side and Jacobian

A procedure for the computation of the right hand sides of the differential or difference equations has to be provided.

`void Rhs(double Time, const Vector& X, Vector& F)`

The following procedures are optional, i.e. they may be empty:

`void Jacobi(double Time, const Vector& X, Matrix& J)` : *procedure computing the Jacobian matrix* `J`*, i.e. the derivative* `J(i,j)` *of the right-hand-sides* `F[i]` *with respect to the state variables* `X[j]`;

`void JacobiParam(double Time, const Vector& X, int Pidx, Vector& JP)` *: procedure computing the Jacobian matrix* `JP`*, i.e. the derivative of the right-hand-sides with respect to the parameter* `P[Pidx];`

If `Jacobi` or `JacobiParam` are empty procedures then the global variable `hasJacobi` has to be set to **0** within the module initialization. Please, observe that the matrix entries have to be specified by parentheses instead of brackets: `J(i,j)`.

No procedure has a calling variable denoting the model parameters: their values are always given in the global variables `P`. These and all other global variables must not be changed in the other procedures of the module. Also, except for `F`, `J`, `JP`, `Q`, `QX`, and `QP` in `Rhs` etc., the parameters of the procedures must not be changed.

In the case of periodically forced systems, one crucial point is how to describe the periodicity. In CANDYS/QA it is assumed that the parameter `P[0]` is the angular frequency, i.e. that the right hand side contains expressions of the form `cos(P[0]*Time)`. The reason for doing so is that the derivative of the right hand side of the equations system $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ with respect to the frequency parameter has a form deviating from the derivative with respect to other parameters. The program system computes this special derivative exactly for the parameter `P[0]`. In this system class, `Time` in `Rhs` etc. is the current value of the independent variable (the time variable) of the system; in the case of the other system classes this parameter is included only for compatibility of the procedure call and has always the value 0.0 .

It may happen that `Rhs`, `Jacobi`, or `JacobiParam` cannot work correctly (e.g., several `X[i]` must be positive but are actually negative). In such a case these procedures should signal an error by specifying

```
ReturnCode = ModelError;
```

(`ReturnCode` is a global variable.)

7.2.4 Quantities

The following procedures compute the state dependent quantities and their derivatives.

`void QuantRhs(const Vector& X,int Qidx,double *Q)` *: procedure computing the state depending quantity* `Q[Qidx]` *that can be monitored during the path-following;*

`void QuantJacobi(const Vector& X, int Qidx, Vector& QX)` *: procedure computing the derivatives of the quantity* `Q[Qidx]` *with respect to* `X;`

`void QuantJacobiParam(const Vector& X, int Qidx, int Pidx, double *QP)` *: procedure computing the derivative of the quantity* `Q[Qidx]` *with respect to* `P[Pidx];`

These procedures should be empty if `NbOfQuantities` equals 0. Moreover, `QuantJacobi` or `QuantJacobiParam` can be empty if `hasJacobi` equals 0.

7.2.5 Correction phase

In the case that the specifications in the initialization part are contradicting or missing then during the startup phase they are in some sense corrected.

The strings `ModelName` **etc.** : Leading white space is deleted, then the string is cut at the first non-alphanumerical character or is completely emptied if the first character is not

a letter. The underscore character "_" is treated like alphanumerical one. If an empty string results then it is replaced by the following string:

- `ModelName="CANDYS";`
- `TName="T";;`
- `PName="P`*i*`";`
- `XName="X`*i*`";`
- `QName="Q`*i*`";`

where *i* is the index of the corresponding variable.

The number `NbOfOutputs` : If this number is less than 1 or greater than `Dim` then it is replaced by `Dim`.

The indices `Oindex` : If one of these indices is not in the range from 0 to `Dim-1`, or if it occurs multiply then it is replaced by first unused index.

The scales `TScale` **etc.** : If a scale equals 0.0 than it is replaced by 1.0.

7.3 Use of MATHEMATICA©

It is possible to provide the `Model` module by MATHEMATICA©. [3] To this end, the file `Candys.m` is needed.

The following activities have to be performed:

1. If not yet defined define the environment variable *CANDYS_DIR* to denote the CANDYS/QA main directory.
2. Start MATHEMATICA© by the command `math`. MATHEMATICA© prompts with `In[`*count*`]` for inputs.
3. Read the file `Candys.m` by the command
   `<< $CANDYS_DIR/math/Candys.m` where `$CANDYS_DIR` is the envirenment variable is defiend in Sec. 5.4.2.
4. Enter the command
   `Candys[`**sys,funcs,params,class,quant,outp,stiff**`]`
   where the parameters have the following meaning:

   sys name of the model (String);

   funcs right hand sides of the system (List of Lists), each function is a List of
   (a) variable name (Symbol),
   (b) right hand side (Expression),
   (c) possibly the initial value of the variable (Real, default: 0.0);

   params parameters variables (List of Lists), each parameter is a List of
   (a) parameter name (Symbol),
   (b) possibly the initial value of the variable (Real, default: 0.0);
   (c) possibly the switch whether the parameter should be variable (String, default="yes")

   class system class, two variants are possible:
   (a) system class (String, only the first character is checked)
   `autonomous` (default), `periodical`, or `difference`,

[3] MATHEMATICA© is a software product of *Wolfram Research Inc.*

(b) List of at most 4 elements
   i. the string of variant (4a),
   ii. the **time** variable (Symbol, default: `Time`) ,
   iii. the **frequency** parameter (Symbol, default: `Omega`),
   iv. possibly the initial value of the **frequency** variable (Real, default: 1.0);

quant right hand sides of the quantities (List of Lists), each quantity is a List of
   (a) quantity name (Symbol),
   (b) right hand side (Expression);

outp names of variables to be printed (List of Symbols, default: all variables);

stiff stiffness of ODE (String, only the first character is checked) `stiff`, or `nonstiff` (default)

5. Stop MATHEMATICA© by the command `Quit`.

Only the parameters **sys**, **funcs**, and **params** are mandatory, for all others exist default values. In case of the parameter **class** the following is recommended:

- system class `difference` does never use **time** or **frequency**, i.e. variant (4a) is sufficient;
- system class `autonomous` uses **time** only as the name of the return time of cycles, i.e. for almost all models variant (4a) is sufficient;
- in models of system class `periodical`, **time** and **frequency** are the corresponding variables of the right hand side (and, therefore, are needed), the **frequency** parameter must not be an element of **params**.

The item (c) of **params** needs an explanation. In general, all parameters of a model can be varied, i.e. the entry is "yes". But sometimes it is convenient to give a constant a name in the formulas (e.g. `g` for the earth's gravity constant, 9.81). This is supported by declaring those constants as non-variable parameters, i.e. setting this entry to "no".

The call of the program `Candys` results in the following:

- The procedures `Rhs` and `QuantRhs` corresponding to **funcs** and **quant**, respectively, are generated.
- The Jacobians of the functions **funcs** with respect to the variables **vars** and the parameters **params** are computed. If quantities have been defined then also their Jacobians are computed. This means, that the procedures `Jacobi` etc. are generated.
- The integers `Dim` etc. are determined by counting the number of variables **vars** etc.
- The variables **vars** and parameters **params** (including the frequency parameter) are replaced by `X[0], X[1], ...` , and `P[0], P[1], ...` , respectively.
- The names `XName` etc. are set to the names of variables in **vars** etc., and `ModelName` to **sys**.
- All scales are set to 1.0, and all not explicitly defined initial values are set to 0.0 (with the exception that the frequency parameter, if any, is set to 1.0).
- The file **sys**`.cxx` is written containing all needed entities. Moreover, the file **sys**`.m` is written containing the model description in MATHEMATICA© style as well as **sys**`.tex` containing the right hand sides and the Jacobian matrices in LaTeX format.

We will give a simple example: establishing the `Model` module for the well known Lorenz system:

$$\begin{aligned} \dot{x} &= \sigma * (y - x) \\ \dot{y} &= r * x - x * z - y \\ \dot{z} &= x * y - b * z \end{aligned}$$

Thus, we start MATHEMATICA© by the command `math` . It prompts for the first input, we answer as described above:

`In[1]` << **Candys.m**

Next we prepare two lists, one for the description of right hand sides and one for the parameters:

`In[2]` **fun=**{{**x,sigma*(y-x),12.**}, {**y,r*x-x*z-y,20.**}, {**z,x*y-b*z,24.**}}

`In[3]` **par=**{{**r,28.**}, {**sigma,10.**}, {**b,8./3.**}}

This means, e.g., that the time derivative of variable x is given by the expression $sigma*(y-x)$, and the default initial value of x is 12. The preparation of the lists **fun** and **par** is not mandatory but makes the following action clearer. Now, we can produce the `Model` module:

`In[4]` **Candys[”Lorenz”,fun,par,”auto”]**

The parameters **”Lorenz”** and **”auto”** define the `ModelName` and the `ModelClass`, respectively. After some time (depending on the complexity of the right hand sides, i.e. in our example after a short time) MATHEMATICA© prompts anew and we stop:

`In[5]` **Quit**

7.4 User defined point handling

As will be explained in Chap. 9, CANDYS/QA shows results in a fixed style onto the screen and writes them onto a file. Sometimes this may be insufficient. One may wish, e.g., that a graphical presentation of the cycles of autonomous systems is to be produced. Such presentation can be performed by the procedures of the user defined module *Handle*.

The module has to include at least the following header files:

```
#include "Handle.h"
#include "States.h"
```

The header file provides the class `Handle` with the constructor
`Handle::Handle(int argc,char **argv).`
It can be used, e.g., creating a graphics window during the startup phase. [4] During the run of CANDYS/QA exactly one variable of the class `Handle` will be allocated, i.e. the constructor is called exactly once. On the other hand, the destructor
`Handle:: Handle()`
can be used to close the graphics etc.

Additionally, there are the termination and initialization procedures

```
void PostHandle(int save)
void PreHandle()
```

[4] But if you did integrate graphics via the module *graphics* (c.f. Sec. 11) then you can be sure that the graphics has already been initialized.

which are called when the node-selection level is entered (i.e. when a path-following is terminated) or left (i.e. when a new one starts), respectively. The parameter `save` signals that even the scenario-level has been entered and the scenario is to be saved.

The essential procedure is

```
void PointHandling(const FullState& St, const BifGraph& BG)
```

that is called after each point computed on a curve during the path-following. This includes special points and excludes the points on branching off solutions. The class `FullState` includes the following members:

`InvariantSetTypes InvariantSet` *the type of the point,* **SteadyState** *or* **Cycle**

`int Periods` *the number of periods in case of a cycle, 0 in the case of a steady-state*

`Section Poincare` *the type of Poincaré section in case of cycles of an autonomous system*

`ScatteredVector Values` *the values of state variables and the varied parameters*

`int Stable` *equals 1 if the point is stable*

`Vector EVreal` *real parts of nontrivial eigenvalues*

`Vector EVimag` *imaginary parts of nontrivial eigenvalues*

`Vector exEVreal` *real parts of trivial eigenvalues*

`Vector exEVimag` *imaginary parts of trivial eigenvalues*

The class `ScatteredVector` contains the members

`Vector x` *the values of state variables*

`Vector p` *the varied parameters*

`Vector q` *the values of monitored quantities*

`double r` *the return time in case of cycles of an autonomous system*

The class `Vector` is a more secure version of `double*` and provides, among others, the following functions:

`v[i]` *gives the* i^{th} *element of vector* `v`

`v.elems()` *gives the length of vector* `v`

Finally, the class `Section` defines the Poincaré map by the members

`SectionModes Mode` *one of the values* **OrthogonalToTrajectory**, **OrthogonalToAxis**, **ExtremumOfComponent**

`int Component` *the Poincaré plane is defined with respect this state variable*

`double Value` *in case of* **OrthogonalToAxis** *the fixed value of this variable*

The class `BifGraph` describes entities that are the same for all points of a bifurcation graph. It includes the following members:

`IndexSet Indices` *the indices of parameters that are variable in this bifurcation graph*

`Vector Params` *the values of the parameters at the creation time of the bifurcation graph, especially, the values of the fixed parameters*

The class `IndexSet` is a version of `Vector` for `int` elements.

Chapter 8

Input Description

8.1 Overview

The user is led through the program by dialogue. The inputs concern the following topics:

- selection of a scenario (c.f. Sec. 8.2.1);
- input of start values for a first point (c.f. Sec. 8.2.2);
- choice which numerical method should be used for computation of the first point, c.f. Sec. 8.2.2;
- choice of a varying parameter and the corresponding interval limits, c.f. Sec. 8.2.4;
- determination of numerical control parameters (e.g. the maximum number of steps and the aborting tolerance in approximation methods), c.f. Secs. 8.3.1 through 8.3.5;
- selection of a node in the bifurcation graph(s) for continuation of the path-following, c.f. Sec. 8.4.

For all questions of the dialogue a default answer is provided (often the value entered to the same question in a previous loop of the dialogue) that is used if simply the `RETURN` key is pressed. The questions possess the following form (also the brackets and braces are printed), where `Prompt` is a short question or the name of a variable and `default` is the currently valid default value for this input:

- Yes/No decision:
  `Prompt {default}` *The answer has to start with* **Y**, **y**, **N**, *or* **n**
- Integer number:
  `Prompt [lower limit,...,upper limit] {default}`
  If `lower limit` *or* `upper limit` *are bounded by computer precision only then they are replaced by* `<-` *or* `->`*, respectively.*

- Real number:

  `Prompt [lower limit,...,upper limit] {default}`

  If `lower limit` *or* `upper limit` *are bounded by computer precision only then they are replaced by* <- *or* ->*, respectively.*

- Choice among several possibilities:

  `Prompt`

  `key_letter_1 --> meaning`

  `key_letter_2 --> meaning`

  `... : ...`

  `...:...`

  `{default}`

  The answer has to be one of the key_letters, the input is case insensitive

- Name from a list of names (state variables, parameters, or quantities):

  `Prompt [name1 name2 ... ] {default}`

  The answer has to be one of the names, the input is case sensitive

- Index from a list of indices (especially scenario numbers):

  `Prompt [index1 index2 ... ] {default}`

Notes

If the admissible range of values determines the answer uniquely then the program uses this value without dialogue.

Each answer is checked for correctness, and the question will be repeated until a correct answer has been entered. The correct answers and the corresponding prompt are recorded to the file *ScenarioName*`.inp`. Chapter 10 contains an example of such a file.

Each answer consists of one string: leading white space is ignored and the string ends at `EOL` or the first white space following nonwhite characters. All characters following the string to the `EOL` are skipped.

In the following subsections the user's interactions are described which are mandatory or optional at the several working levels. The following convention is used:

- `teletype letters`: computer output, may be restricted to a key word;
- **bold letters**: user input;
- *italic letters*: comments.

If an absolute default value is given for a numerical control parameter then this is the value valid at program start without a restart-file, otherwise the default equals the value entered last. Default values depending on other entities are always newly computed.

8.2 Navigating through the program

8.2.1 Program start and the scenario level

The program is started by entering the command

CANDYS

where the name *CANDYS* is, in general, the name of the system as chosen at compile time (c.f. Sec. 5.4). After start, the program enters the scenario level. Here, one has to choose whether a new scenario should be created or an existing one ought to be continued.

`option =` **N** *newstart of a scenario;*

`option =` **R** *restart of a existing scenario;*

`option =` **X** *exit from* CANDYS/QA.

Then the number of a scenario has to be entered. In the case of a new scenario, the user is asked by a sequence of questions to enter the name(s) of the state dependent quantities that should be monitored during this scenario:

`Enter name of quantity to be extreme:` *the incorrect name* **0** *stops the sequence of questions*

Default: **0**

In the case where the scenario level is entered from node selection, the user is asked whether to save or restore results:

`saving option =` **S** *save new results;*

`saving option =` **R** *restore previously saved results.*

Then the user is asked for how to continue:

`option =` **C** *continue this scenario;*

`option =` **N** *newstart of a scenario;*

`option =` **X** *exit from* CANDYS/QA.

8.2.2 The methods level - computation of a first point

When deciding to start a new scenario, CANDYS/QA switches into the methods level. In the case of autonomous systems the initialization needs to know whether steady-states or cycles should be computed. Hence, the user can interactively choose between the following possibilities:

`option =` **S** *steady-states;*

`option =` **C** *cycles.*

Then the values of the fixed parameters as well as initial values of the state variables are asked for. Their defaults are given

1. after program start without a restart-file by the values specified in
   `Model::Model(int argc,char **argv);`

2. after program start with a restart-file by the values in this file;
3. after change of scenario by the values shown last during the process of node selection (c.f. Sec. 8.4), this way, one can start a new scenario with "good" initial values;
4. by the values entered before (if any).

In the case of autonomous systems, the period length T is an invariant, too, that has to be determined, while in the case of periodically forced systems cycles with a multiple of the forcing period are computed. Fixed points of discrete systems are considered as cycles of the length 1. For supporting the user in defining a suitable period length, up to N periods are iterated and displayed from which the number

`PeriodLength:` *1 to N*

Default: the period matching best the starting point

has to be chosen. The number N is asked by before:

`Enter maximum number of test cycles`

Default: **1**

Then, 5 methods are offered to solve the corresponding nonlinear equations system (2.8) where the parameter vector $\boldsymbol{p}$ is fixed. [2] One can choose one of the following methods.

`method =` **N** *solving the equations system by a damped* NEWTON*'s method where a suitable initial guess is needed (which may be found, e.g., by the other methods); this is the default method, c.f. Sec. 8.3.1.*

`method =` **M** *Monte-Carlo search in a chosen parallelepiped with the initial point* $\boldsymbol{x}^0$ *as its midpoint, c.f. Sec. 8.3.2.*

`method =` **E** *evolutionary search, i.e. random search in a sliding parallelepiped with the midpoint at the initial point* $\boldsymbol{x}^0$ *at the first step and then at the best point found, c.f. Sec. 8.3.2.*

`method =` **H** *performing a simple homotopy method form the start point* $\boldsymbol{x}^0$*, c.f. Sec. 2.2.4. User interactions are those of the path-following (without parameter limits), c.f. Sec. 8.2.4.*

`method =` **S** *simulation of a trajectory starting at* $\boldsymbol{x}^0$*, c.f. Sec. 8.3.3.*

When an approximation has been found by an algorithm the quality of the result is estimated by checking of $\|\boldsymbol{F}(\boldsymbol{x})\|$ against the value `EpsF`. Then the user can select one of the following options to proceed (c.f. step 7 of Sec. 6.5).

`option =` **P** *start path-following at this point*

`option =` **N** *enter new initial values*

`option =` **R** *reset initial values to the previously entered ones*

`option =` **I** *improve last approximated point*

`option =` **X** *exit to the scenario level*

[2] From Sec. 2.2 we recall the following. The function $\boldsymbol{F} : \boldsymbol{R}^m \to \boldsymbol{R}^m$ is derived from the right hand sides of the system dynamics. It holds $m = n + 1$ and $\boldsymbol{y} = (\boldsymbol{x}, T)$ in the case of the computation of cycles of autonomous systems, otherwise $m = n$ and $\boldsymbol{y} = \boldsymbol{x}$.

8.2.3 The graph-manipulating level

The dialogue contains the following questions.

`ParamName:` *The name of a system parameter that has to be varied*

Default: **Name of the first parameter not yet varied**

`Distance:` *The length of the arc from the selected point to the one to be computed*

Default: **0.1**

Then the control parameters of the path-following level (without the limits) are asked.

8.2.4 The path-following level

The solutions of the m-dimensional nonlinear equations system (2.8) depending on the parameter p describe a curve $\boldsymbol{C}$ in $\boldsymbol{R}^{m+1}$ which is called a solution branch. [3] The predictor-corrector algorithm described in Sec. 2.3 is applied.

The user is asked for the limits of the interval(s) within which the variable parameter(s) can vary. Then one can enter some numerical control parameters. Furthermore, in the case that all branches are to be continued, an upper bound for the length of cycles has to be chosen (this way, one can avoid to build up an infinite graph if there is a period doubling cascade).

The varied parameters and the monitored quantities can be limited to some intervals (at least one varied parameter has to be limited). The names of parameters or quantities to be limited are asked in sequence of questions.

`Enter variable to be limited` *Entering* **0** *stops the sequence.*

Default: **Name of** 1^{st} **variable not yet asked**

The following entities can be object of limitation:

- the state variables,
- the values of the functions provided by the procedure `QuantRhs`,
- in the case of cycles of autonomous systems the period length,
- in the case of torus bifurcations the angle of the critical eigenvalue (c.f. Sec. 1.4.3), this case is selected by the specific name `1rot`. [4]

For each variable parameter the corresponding limits are asked:

`Lower limit:` *Lower interval limit*

Default: **current value-10.0*(last step-size)**

`Upper limit:` *Upper interval limit*

[3] In the case of computation of special points, this equations system contains further equations and unknowns, i.e. the dimension m is larger than n or $n+1$.

[4] The name `1rot` is generated automatically. By its construction, it is different from all possible user defined names.

Default: **current value+10.0*(last step-size)**

Here, the **current value** is given by the point that was selected for path-following, analogously the **last step-size**. The difference (`Upper limit - Lower limit`) of the last variable parameter determines the value of **interval length** that is used in the following.

After entering the control parameters of Newton's method (c.f. Sec. 8.3.1) the numerical control parameters of the path-following are asked:

`MaxCurveStep`: *Maximum step-size on the solution branch. If the step-size control yields a larger value than* `MaxCurveStep` *then* `MaxCurveStep` *is used instead.*

Default: **0.1*interval length**

`MinCurveStep`: *Minimum step-size on the solution branch. If the step-size control yields a smaller value than* `MinCurveStep` *then the path-following is aborted.*

Default: **1.0E-5*interval length**

`MaxArcLength`: *Maximum length of the solution branch. If the solution branch becomes longer than* `MaxArcLength` *then the path-following is aborted.*

Default: **100.0*interval length**

Note

The chosen accuracies have a side-effect: the geometrical mean of `EpsF` and an internally computed value (called `EpsPredic`) is the tolerance for the detection of already computed points (c.f. Sec. 6.7).

8.3 Numerical control parameters

8.3.1 Newton's method

Numerical control parameters for Newton's method:

`EpsF`: *Accuracy of Newton's method.*

Default: **1.0E-6**

`MaxIterations`: *Maximum number of steps of Newton's method*

Default: **20**

8.3.2 Monte-Carlo method and evolutionary search

Numerical control parameters for both methods:

`SizeOfParallelepiped`: *Size of the (initial) parallelepiped for Monte-Carlo search and evolutionary strategy. If* $\boldsymbol{x}^0$ *is the initial midpoint then the parallelepiped is bounded by* $\left[x[i]^0 - size[i], x[i]^0 + size[i]\right]$ *for* $i = 0, \ldots, m-1$.

Default: **1.0**

`MaxSearchSteps`: *Maximum number of steps during Monte-Carlo search or evolutionary search.*

Default: **50**

`EpsF`: *Aborting accuracy.*

Default: **1.0E-6**

8.3.3 Simulation

It is possible to simulate the dynamical system at constant time intervals for output (autonomous and periodically forced systems) or to iterate discrete systems. In the case of searching cycles of autonomous systems it is also possible to print the intersections of the trajectory with the Poincaré surface to observe the approximation of a fixed point of the Poincaré map. Furthermore, it is possible to simulate autonomous and periodically forced systems in reversed time. The simulation can be continued (in the same time direction) when it exceeded the chosen time limit. On the other hand, it can be interrupted by entering *CTRL/C*. The simulation results are recorded onto special simulation files. After a simulation of cycles, the question for the cycle length is repeated.

Numerical control parameters for the simulation:

`SimulTime`: *Maximum simulation time (autonomous and periodically forced systems only), may be negative.*

Default: **0.0**

`SimulDeltaT`: *Time interval between two outputs in the case of forward or backward simulation (system classes* `autonomous` *and* `periodical` *only).*

Default: **0.1**

`SimulPeriods`: *Maximum number of periods of the cycles to be simulated.*

Default: **1**

`StepsInPeriod`: *In the case of periodically forced systems the number of outputs within each period length.*

Default: **1**

`EpsF`: *Accuracy for accepting the last simulated point.*

Default: **1.0E-6**

8.3.4 The Poincaré map

Cycles of autonomous systems are computed by use of a Poincaré map where the problem is transformed to an $(n+1)$-dimensional algebraic equations system that allows the calculation of the period length T in addition to a point of the cycle (c.f. Sec. 2.1). CANDYS/QA provides two options for the function $g(\boldsymbol{x}, \boldsymbol{p})$. The following possibilities have been provided:

`SectionMode =` **T** *Plane orthogonal to the trajectory supported by the actual point. (Attention: the Poincaré plane is newly defined at each new point). This option is available only within the methods level and is the default here.*

`SectionMode =` **A** *Plane orthogonal to one coordinate axis supported by a certain point.*

`SectionMode =` **E** *The surface is defined as the set of all extremal points of the trajectories with respect to one component: $f[i](\boldsymbol{x}, \boldsymbol{p}) = 0$. (Attention: the actual point does not necessarily belong to the surface, it is shifted to the surface by this choice. The surface changes as the parameters vary.)*

`Component:` *Name of the component of the state vector $\boldsymbol{x}$ for which the chosen mode becomes effective).*

Default: **Name of the 0-th component**

`Fixed value:` *In the case of* `SectionMode` **A** *when computing the first point the fixed value of the component $x[i]$.*

Default: **Current value of** $x[i]$

The `SectionMode` of the Poincaré surface has to be determined in the following situations:

- computing of a first cycle;
- during the path following if the chosen `SectionMode` is no longer suitable, i.e. in the case of an artifact (c.f. Sec. 2.1.2).

The following items are meaningful only when choosing an initial guess for the period length (i.e. at the methods level), hence, they are asked for only in this case.

`StepSize:` *Length of the integration steps for testing whether the Poincaré surface has been crossed by the trajectory.*

Default: **0.01**

`MaxPeriod:` *Maximum integration length between two consecutive* POINCARÉ *maps.*

Default: **100.0**

8.3.5 Integration procedures

If the system is described by differential equations then at some places the numerical control parameters for the integration procedures can be changed:

`EpsInteg:` *The desired integration accuracy. This is the distance between the approximations of the orders 5 and 6 (or 3 and 4 in the case of stiff differential equations). If this distance is larger than* `EpsInteg` *then the integration step is repeated with half step-size, if the distance is smaller than 0.1**`EpsInteg` *then the step-size is doubled.*

Default: **1.0E-8**

`MinStepSize:` *Minimum integration step-size. If the step-size becomes smaller than* `MinStepSize` *then there are two possibilities: aborting the procedure, or continuing with the minimum step-size (default: continuation with minimum step-size).*

Default: **1.E-7**

8.4 Choice of a node

The selection of a node within the scenario level differs from the other dialogues. Interchangingly, the program prompts for a node (the format of the prompt is described below) and the user answers in a coded form whether the node should be selected, or what else to do. This way, all nodes in all graphs can be traversed.

The prompt takes on the following format consisting of three lines:

- 1^{st} line:
  1. node number,
  2. period length (if system class is `periodical` or `difference`) or return time (if system class is `autonomous` and if the point is a steady-state and not simultaneously a Hopf bifurcation then `0.0`),
  3. in the case of system class `autonomous` the letter "S" or "C" specifying the point as a steady-state or a cycle, respectively,
  4. the point type in the format described in Sec. 9.2;
- 2^{nd} line:
  1. the values of the varied parameters,
  2. a vertical bar "|",
  3. the values of the first state values (as many as can fit onto the line).
- 3^{rd} line: The actual possible answers, see below.

An example of the prompt is the following:

```
7:      0.65250 S   Hopf    stable
    24.742       10.031 |      7.9568       7.9568       23.742
[ 0!*]  { }
```

This means: the node `7` is a steady-state `S`, namely a Hopf bifurcation where the nearby cycles have return time `0.65250`, and the point is a `stable` temporary endpoint during the variation of the 2nd parameter. Moreover, there are two variable parameters with the values `24.742` and `10.031`, respectively, and the state variables have the values `7.9568 7.9568 23.742`. Finally, the characters enclosed within brackets list the possible answers. Particularly, `0` means that any node can be addressed by its index.

In the case of a special point with a multiple eigenvalue on the imaginary axis or on the unit circle the point type is described as sequence of single characteristics, e.g.
`Turn Turn stable`
means that the node actually shown is an endpoint and possesses two eigenvalues 0, in fact it is a Takens-Bogdanov point.

In general, only a single character has to be entered. The following characters possess a meaning during the selection process: [5]

0 *Switch to the scenario level without selecting a node.*

! *The current node is a temporary endpoint: select this node to continue the path-following.*

***** *Continue the path-following automatically for all temporary endpoints.*

? *Start the path-following with respect to a further parameter, i.e. create a new graph. If the current node is a temporary endpoint then two graphs are created (c.f. item (2) of Sec. 6.6), else one graph is created (c.f. item (1) of Sec. 6.6).*

@ *The current node is a temporary end point: select this node as a template of a new special node with a constant quantity (c.f. 6.3).*

(space) switch to the next node in the graph, this is the default.

" *Independently of the current node and graph, make a node actual by its number.*

+ *Make the point under the mouse pointer actual (c.f. Sec. 11.2).*

^ *Switch to the traversing of a graph upward in the meta-graph (c.f. Sec. 3).*

> *Switch to the traversing of the next graph with the same variable parameters.*

_ *Switch to the traversing of a graph downward in the meta-graph.*

& *If online graphics is available: Change the coordinates (c.f. Sec. 11.2).*

In the case of several choices the dialogue is continued concerning the following questions.

? What variable parameter has to be held fixed (only in the case of item (2) of Sec. 6.6) and what parameter has to be varied additionally. The irregular name **0** is a possible answer and prevents the creation of new graphs if "?" was entered accidentally. Furthermore, in the case of item (2) the program asks whether the selection process shall be continued within the new graph (default is "No").

@ What quantity has to be held fixed (at the current value). The possible names are the same which can be selected for limitation (c.f. Sec. 8.2.4). The irregular name **0** is a possible answer and prevents the creation of a new node if "@" was entered accidentally.

_ What variable parameter has to be held fixed in the target graph.

^ What variable parameter has to be variable in the target graph.

" The number of the node to become actual.

& What variables are to be shown in the graphics.

[5] The somewhat cryptic input codes have been introduced mainly for one reason. It should be possible to traverse the current graph or to jump from graph to graph by a simple key stroke. This becomes most important if the bifurcation graph is presented graphically and if the selection process is perfomed in that graphics: graphics systems often support only a very simple input. Thus, a single key stroke seems to be the common minimum denominator across the different systems.

The selection process is a sequence of any length, it is stopped by entering one of the choices "!", "*", "?", or "0". The intermediate choices are not written to the file *ScenarioName*`.inp`. Onto this file the following inputs are written:

1. the number of the node selected, or 0 if no node is selected;
2. if a node has been selected: the choice (i.e. "!" etc.);
3. if a node has been selected as basis for the creation of a new graph: the name of parameter to be held fixed (if any) and the name of parameter to be varied additionally.

The information in file *ScenarioName*`.inp` are sufficient to reproduce the selection.

8.5 Program interrupt

During the run, the computations (as simulation or path following) can be interrupted by entering *CTRL/C*. Pressing *CTRL/C* once, the program stops at the next computed point and switches to the scenario level (in the case of path-following) or to the methods level (in the case of simulation). This means, that the program reaches the same state as in the case of reaching the requested path-following or simulation limit; the interrupt is not considered as an error. The interrupt may be used to change the control parameter of numerical algorithms (e.g. the step-size) for further continuation or to switch to another branch. Pressing *CTRL/C* twice consecutively, the time consuming algorithms (Newton's method, integration) are interrupted. This kind of interrupt is treated as a numerical error.

Note

The interrupt as described above works only under UNIX (or any of its descentants).

Chapter 9

Output Description

9.1 Overview

All results obtained by CANDYS/QA are shown on the terminal and are written onto an output file that is the direct copy of the screen output and that may be used for printing of the results. But also other information is shown on screen, and some other files are written. The terminal output includes the following topics.

- The points computed during path-following (c.f. Sec. 9.2).
- Header information: the start of a new path-following, the creation of a new graph, and the switching between graphs are indicated by a header like

  ```
  ******************************
     Tracing the selected curve
  ******************************
  ```

  accompanied by the graph number and information about the varied and fixed parameters (c.f. Sec. 9.2).
- Simulation results: they are not written to the output file but to special simulation files
- Messages, e.g. a special point has been found or non-convergence of a method (c.f. Sec. 9.3).

During the run the program writes several files, e.g. as already mentioned a copy of the terminal output. Several files are newly written in case of a scenario save operation, others existing for scenario `00` are appended to the one existing for the current scenario. The file name is composed of

1. the *ScenarioName* (c.f. Sec. 6.4),
2. the period character,
3. a specific extension.

ext	meaning	appended	*ASCII*
`out`	result file, it is ready for printing	yes	yes
`crt`	excerpt of the result file containing the special points only	yes	yes
`inp`	file of the inputs entered	yes	yes
`rst`	restart file	no	no
`g00`	graphics file containing the extremum values occurring in the graphs	no	no
g*xx*	graphics file containing the results in the *xx*-th graph	yes	no
`c00`	file containing a list of the connections of the graphs within the meta-graph	no	yes
c*xx*	file containing a list of the connections of nodes and edges in graph *xx*	no	yes
`s00`	file containing the extremum values occurring in the simulations	yes	yes
s*xx*	file containing the results of the *xx*-th simulation	no	yes
`sim`	verbal description of all simulation files	yes	yes

Table 9.1: The files created by CANDYS/QA

All files are listed in table 9.1.

The restart file and the graphics files are devoted to be read in by CANDYS/QA and related post-processing programs only. Thus, it was chosen to write them in binary format since binary files are smaller and much faster to write or read than *ASCII*-files. Unfortunately, this has one drawback: the files cannot, in general, be ported from one computer platform to another.

Note

The restart file contains all information needed for the continuation of the computations, it does not contain the information contained in the files characterized by "appended" in Table 9.1. If some of these files have been destroyed or deleted then they cannot be repaired or reproduced on the basis of the restart file. Instead, if the file has been destroyed then it is prolongated as if were correct, and if the file has been deleted then simply a new one is created containing the last results only.

9.2 Result presentation

Each time when computation is started (for calculation a first point or for path-following) or when the actual graph is temporarily changed the number and the parameters of the actual graph are printed. The parameters have the following format:

```
r=***  sigma=10  b=2.66667
```

The asterisks `***` mean that the corresponding parameter is variable while the others are held fixed at the given values.

Note

This form of parameter representation will be used also in other contexts.

All computed points (the first point, the points on a curve, and the special points) are shown on screen with the following constituents (some may be omitted if not relevant):

1. the node number
2. values of the varied parameter (if any), they occur in the ordering of increasing indices as specified in the module `Model`;
3. in the case of cycles their length or return time;
4. the state variables, they occur in the ordering given by the index vector `Oindex` of the module `Model`;
5. the values of the monitored quantities (if any);
6. the eigenvalues of the corresponding Jacobian;
7. in the case of a special point its degeneracy in the sense of Subsection 2.4.2 (allowing to determine whether a period doubling is subcritical or supercritical);
8. in the case of a special point its type.

The eigenvalues are ordered decreasingly with respect to their real part (steady-states) or modulus (cycles). Moreover, the eigenvalues are separated into two groups: one group contains all eigenvalues with predetermined value (e.g., the eigenvalue +1 in the case of cycles, or the eigenvalues lying on the imaginary axis in the case of a Hopf bifurcation); the other group contains the remaining eigenvalues. The eigenvalues of the first group are emphasized as trivial, the others as nontrivial ones.

An example of a result output containing all items is given below.

```
Node no. 9
Parameters          33.367967
Return time =      0.31985405
State              -1.8881585        0.59802032         3.2527558
                   0.18273098         19.613853
Quantity            3.9227707        0.69613853        0.77229358
Nontrivial eigen values :
Modulus            0.29846322       0.039410856       0.039410856
Degrees             180.00000         3.2240881        -3.2240881
Trivial eigen values :
Modulus            0.99999059         1.0000890
Degrees             0.0000000         180.00000
Degeneracy         -76.454751
PerDoubl
```

Note

Special points and temporary endpoints are printed in a specific format. Suppose that the point under consideration is a special point in a k-parameter graph. Then corresponding to the k conditions on the eigenvalues and/or degeneracies a list of k entries is printed on one line. For instance,
`Turning Turning`
denotes a special point that has two zero eigenvalues of the matrix $\boldsymbol{J}_N$, i.e. the point actually is a Takens-Bogdanov point. The temporary endpoints in a $(k+1)$-parameter graph are printed in the same format with an additional entry denoting its stability. Curves are characterized in the same fashion since they consist, theoretically, of temporary endpoints only.

9.3 Messages

In several situations CANDYS/QA issues a message. There are three kinds of messages:

- informations: the program has performed something successfully, e.g. the end of the path-following interval is reached or a special point has been found;
- warnings: a computation was not successful but the program can work further, e.g. a detected special point could not be computed;
- errors: a computation cannot be completed, e.g. due to a too small step-size, and the program has to change its level.

Each message sent is accompanied with a "beep". The format of the messages is adopted from *DEC/VMS* messages:
`%CANDYS-`*T-SHORT long*
where

- `%CANDYS` is a fixed portion indicating that the message is sent by CANDYS/QA;
- *T* is one of the letters `I`, `W` and `E` meaning an information, a warning, or an error, respectively;
- *SHORT* is an identifier of the internal detection mechanism of up to 9 uppercase letters;
- *long* is a verbal description of the message in lowercase letters.

In the following all *SHORT* portions are listed in alphabetical order, comments are written in *italic letters*.

`ARCLIMIT` *maximum arc length of the solution branch exceeded*

`CONTROLC` *interrupt by CTRL/C*

`EVMISSING` *not all eigenvalues could be computed*

`FORMERROR` *wrong graphics format (c.f. Sec. 11)*

`GRADLIMIT` *internal error of Newton's method*

`LIMREACH` *limit of parameter interval reached*

`LOOPLIMIT` *maximum number of steps exceeded in Newton's method*

`MATSING` *matrix singular*

`MODELERR` *error in the right hand side or Jacobians, to be provided in* `Rhs` *etc.*

`NEGPERIOD` *the period of the cycle approximated last is not positive*

`NOBRANCH` *not all branching off stumps could be computed*

`NOCONV` *the first point is not found*

`NOCYCLE` *no Poincaré map found*

`NOFILE` *the output file cannot be opened*

`NULLCYCLE` *instead of a cycle a steady-state has been computed*

`OK` *successful action*

`OLDRST` *the restart file is too old, e.g. it was produced with a different version number than the current program was compiled with*

`OVERFLOW` *floating point overflow*

`PARALLEL` *tangents in the bifurcation point are parallel*

`REPPOINT` *the current point (and arc) is repeatedly computed*

`RSTDEST` *restart file is destroyed*

`SAVE` *saving the current scenario*

`SMALLSTEP` *too small step size*

`UNDEFERR` *other reason*

In the following all *long* portions of the messages are listed in alphabetic order. Most of them are not commented since it is hoped that they can be understood immediately. Comments concern *long* portions that contain variable parts.

`arc has been united to the arc at node` *n*

`arc too long`

`branch point computed`

`break by control key`

`special point computed`

`special point is the same as node` *n*

`curve cannot be initialized`

`curve cannot be continued`

`eigenvalues not calculable`

`files cannot be opened`

`first point found`

`insufficient convergence`

`limit of parameter interval`

`not all branches computed`

`not all critical points or branches computed`

k `of` *l* `branches computed`

k `of` *l* `end points computed`

special `point approximated`
actually, "special" is replaced by the type of special point

`point with constant period inserted`

`point with extreme stability approximated`

`restart file not correct`

`restart file too old`

`saving scenario no.` *n*

`simulation error`

In connection with reading or writing files of graphics formats further *long* portions occur. They are described in Sec. 11.4.5.

The same message may occur as information, warning, or error depending on the program context, also the same *SHORT* portion may be combined with different *long* portions.

9.4 Structure of several output files

9.4.1 General remarks

The structure of the files will be presented in a BACKUS-NAUR like form. This means the following. There is a set of key words (the so-called nonterminals) which, on the one hand, describe a certain complex of the file with some meaning and which, on the other hand, are declared as a sequence of other nonterminals or fixed symbols (the terminals) and where only the defining terminals are written onto the files. For reading the files by the computer, a program is recommended that includes for each nonterminal a procedure. These procedures have to read repeatedly an item, decode it by the declaration (the right hand side) of the nonterminal and, if another nonterminal results, call the corresponding procedure. The entire program itself corresponds to the first nonterminal of a file description.

The following convention will be used for the declaration of the nonterminals.

- The new nonterminal is the left hand side of an assignment whereas the right hand side is its declaration. The declaration is ended by a period, so one can recognize the end also in declarations covering more than one line.

- `Teletype letters`: nonterminals.
- **Bold letters**: terminals.
- *Italic letters*: comments.
- Consecutive expressions in a declaration mean that the corresponding values occur consecutively on the file.
- Expressions in braces with an exponent, i.e. $\{\}^e$, mean as many repetitions of the expression as the exponent says, the exponent may be zero.
- Expressions in braces without an exponent, i.e. {}, mean any number of repetitions of that expression.
- Expressions in parentheses, i.e. (), mean that exactly one of them has to be selected.

In all file structures the following elementary nonterminals are used.

`String` := *ASCII characters*

`Integer` := *Integer number*

`Real` := *Real number*

`SP` := *white space*

`EOL` := *end of line, may include white space*

`EOF` := *end of file*

9.4.2 Simulation files

During each simulation run of the model a simulation file with the extension `.s`*xx* is written where *xx* is the consecutive number of the simulation within a scenario. These files are devoted to be read by a computer for graphical presentation. Such a file simply contains a table: each line contains in the first column the simulation time and then the values of the model states. All values (even the integer time of difference equation systems) are written as a 15 digit number (ensuring at least 2 blanks between the numbers).

`SimFile` := {`Block`} `EOF` .

`Block` := `Time` { `SP` `Xi`}$^{\texttt{Dim}}$ `EOL` .

`Time` := `Real` . *Current simulation time*

`Xi` := `Real` . *Component of state vector*

`Dim` := `Integer` . *Dimension of system*

Additionally to the simulation files which are created for each simulation run there is a global simulation file for the entire scenario (with the extension `.s00`) that contains for each simulation the extrema of the state. It can be used to scale graphical output to the appropriate size.

`GlobalSimFile` := {`ExtremaBlock`} `EOF` .

`ExtremaBlock` := `SimNo` `Count` {`Minimum`}$^{\texttt{Count}}$ {`Maximum`}$^{\texttt{Count}}$.

`SimNo` := `Integer` `EOL` . *Number of the simulation whose extrema follow*

`Count` := `Integer` `EOL` . *Number of the components per point, i.e. the dimension of system*

`Minimum` := `Real` `EOL` . *Minimum of a component*

`Maximum` := `Real` `EOL` . *Maximum of a component*

9.4.3 Connection files

In the course of writing a new restart-file also so-called connection files are written. The files with the extension `.cxx` list the node connections of the graphs where *xx* is the graph number. It is devoted to visual inspection and should help the user when selecting a node.

`Graph` := {`Connection`} `EOF` . *Sequence of node connections ordered by the node number*

`Connection` := `Node EOL --> { SP Edge} EOL` . *Node description and list of the edges belonging to the node*

`Node` := `Number SP : SP Period {SP NodeSpecial}`$^{\texttt{VarParamCount}}$. *Node description*

`Number` := `Integer` . *The node number*

`Period` := `Integer` . *The period length, 0 in the case of a steady-state*

`NodeSpecial` := (**Turning Bifurc HopfBif imagHopf PerDbl TorusBif imagTori Cusp degHopf degPD degTorus ConstPer ExtrQu stable unstable**) . *Fixed names for the different special points,* `stable` *and* `unstable` *denote temporary end points (c.f. Sec. 9.2)*

`VarParamCount` := `Integer` . *Number of varied parameters*

`Edge` := `NodeNumber : EdgeSpecial { _ EdgeSpecial }`$^{\texttt{VarParamCount}-1}$. *One string containing the number of node pointed to by the edge and type of the points on the arc*

`EdgeSpecial` := (**T B H iH P To iT C dH dP dT CP Q s u**) . *Fixed symbols for the different special points listed in the same order as for* `NodeSpecial`

An example of on entry in the file (a `Node`) is given below.

```
 16 :  0   Turning   Cusp
   -->  14:0_T_s  12:0_T_s
```

This means that node `16`

1. is a steady-state (period `0`);
2. is a special point (no `stable` or `unstable` item);
3. is a cusp point (the item `Turning` results from the fact that one eigenvalue equals 0);
4. is connected to two other nodes, namely to nodes `14` and `12`, and both connecting edges consist of points that are
   (a) steady-states (item `0_`),
   (b) turning points (item `T_`),
   (c) stable (item `s`), i.e. all nontrivial eigenvalues have negative real part.

Additionally, the file with the extension `.c00` lists the connections in the meta-graph.

`MetaGraph` := { `Graph` } `EOF` .

`Graph` := `Number : EOL GraphParams {DownGraph} {UpGraph}` . *Sequence of graph connections ordered by graph number*

`Number` := `Integer` . *The graph number*

`GraphParams` := { `SP Param = ( Value ***`) }$^{\texttt{ParamNo}}$ `EOL` . *List of values of fixed parameters (indicated by a real number) and of varied parameters (indicated by* `***`*), in case of a large* `ParamNo` *additional* `EOL`*s can be included*

`Param` := String . *Model parameter*

`Value` := Real . *The parameter value*

`ParamNo` := Integer . *Number of model parameters*

`DownGraph` := `--> SP Number SP Param = Value EOL` . *Current graph originates from graph* `Number` *where parameter* `Param` *is varied additionally, this parameter has fixed value* `Value` *in graph* `Number`

`UpGraph` := `--> SP Number SP Param =*** EOL` . *From current graph originates graph* `Number` *where parameter* `Param` *is varied additionally*

An example of one entry in the file (a `Graph`) is given below (it is taken from the example of Sec. 10.1.1).

```
2:
r=***  sigma=10  b=2.66667
-->  1  r=20
-->  3  sigma=***
```

This means that graph `2`:

1. has one variable parameter, `r`;
2. has the fixed parameters `sigma` and `b` with given values;
3. it originates from graph `1` where
   (a) parameter `r` is varied additionally,
   (b) parameter `r` has fixed value `20` in graph `1`;
4. from it originates graph `3` where
   (a) parameter `sigma` is varied additionally.

Chapter 10

Example

In this section, we demonstrate the capabilities of CANDYS/QA by some examples. The user interactions and the results are presented in form of the mutually interleaved files `*.inp` and `*.out`. The following convention will be used:

- **bold letters**: the user inputs as recorded in file `*.inp`;
- sans-serif letters: the corresponding questions as recorded in file `*.inp`;
- `teletype letters`: file `*.out`, i.e. the results;
- Roman letters: comments;
- long sections of the file `*.out` without dialogue are represented by
  `...`

10.1 The main course of qualitative analysis

10.1.1 The model: Lorenz system

We examine the well known Lorenz system [30, 46]. It is an autonomous system of differential equations:

$$\begin{aligned} \dot{x} &= \sigma * (y - x) \\ \dot{y} &= r * x - x * z - y \\ \dot{z} &= x * y - b * z \end{aligned}$$

Herein x, y, and z are the state variables, r, σ, and b are the parameters. [2]

10.1.2 The methods level

The program starts with the following heading:

[2] A ready-to-use module for Lorenz comes with the package.

```
************************************************************
*                                                          *
*                    C A N D Y S / Q A                     *
*                                                          *
*     Computer Analysis of Nonlinear Dynamical Systems     *
*                                                          *
*          Qualitative Analysis of Model "Lorenz"          *
*                                                          *
************************************************************

    Version: 4.2.6.4

    Copy right:
           Max-Planck-Gesellschaft
           AG 'Nichtlineare Dynamik'
           an der Universitaet Potsdam
```

No scenarios exist so far. Thus, we start a new one (with number 1), we do not plan to compute extrema:

N	Enter scenario option
1	Enter number of scenario
0	Enter name of quantity to be extreme

```
************************************
   Computation of the initial point
************************************
```

Choice of an invariant set (steady-states):

S	invariant set type

Input of the system parameters:

Yes	Do you want to change the fixed parameters
20.000	r
10.000	sigma
2.6667	b

```
Graph 1
Parameters:
  r=20  sigma=10  b=2.66667
```

Input of the initial values of the state variables:

Yes	Do you want to change the initial vector
12.000	x
20.000	y
24.000	z

Choice of a method (Monte-Carlo search):

M	method

Input of the control parameters of the Monte-Carlo search:

Yes	Do you want to change the sizes of parallelepiped
1.0000	x
2.0000	y
2.0000	z
20	MaxSearchSteps
No	Do you want to change the integration constants
1.0000E-6	EpsF
No	Do you want to use a numerical Jacobian

The Monte-Carlo search is not successful:

```
Computed            9.8188726        16.739860        20.817469
%CANDYS-W-NOCONV, insufficient convergence
```

Choice of an option how to proceed with that point, here improve it:

I	option

Choice of another method (evolutionary search):

E	method

No change of the control parameters:

No	Do you want to change the sizes of parallelepiped

The evolutionary strategy did not lead to an acceptable result either:

```
Computed            7.2005371        11.752617        22.934904
%CANDYS-W-NOCONV, insufficient convergence
```

Once again, the point should be improved:

I	option

Choice of another method (Newton's method):

N	method

Input of the control parameters:

Yes	Do you want to change the parameters of Newton's Routine
1.0000E-06	EpsF
20	MaxIterations
No	Do you want to use a numerical Jacobian
No	Do you want to change the integration constants

Newton's method yields the following solution:

```
Node no. 1
State              7.1180522         7.1180522         19.000000
Quantity          -20.513855
Nontrivial eigen values :
Real             -0.15479177       -0.15479177       -13.357083
Imag               8.7086685        -8.7086685         0.0000000
%CANDYS-I-OK, first point found
```

This point will be the starting point of the path-following:

P	option

The program enters the process of node selection. This time we have only one node, i.e. we have the possibilities to start the path-following or to go back to the scenario level. Let's go to the scenario level and save the result:

0	Select node for next curve
S	saving option

10.1.3 The graph-manipulating level

Then we continue the scenario. We want to make path-following starting at the only node. Before, we have to create a new graph by the choice "?" at node 1:

C	scenario option
?	Enter char or node number

The program asks which parameter shall be varied, how long the initial step should be, and other numerical control parameters (not so important at this point):

r	Enter variable parameter to be varied

0.10000	Length of initial step
No	Do you want to change the parameters of Newton-Routine
No	Do you want to change the integration constants
0.01000	MaxCurveStep
1.0000E-6	MinCurveStep
1.0000	MaxArcLength
0.0010	EpsPredic

The graph 2 is created and a short curve of steady-states (more precisely: two points) is computed:

```
   ****************************************
      Creation of a new bifurcation graph
   ****************************************

Graph 2
Parameters:
  r=***  sigma=10  b=2.66667

Node no. 2
Parameters          19.930500
State               7.1050218         7.1050218         18.930500
Quantity           -20.568883
Nontrivial eigen values :
Real              -0.15725360       -0.15725360       -13.352159
Imag                8.6942942         -8.6942942         0.0000000

Node no. 3
Parameters          20.069504
State               7.1310596         7.1310596         19.069504
Quantity           -20.458585
Nontrivial eigen values :
Real              -0.15232729       -0.15232729       -13.362012
Imag                8.7230634         -8.7230634         0.0000000
%CANDYS-I-OK, 2 of 2 endpoints computed
```

The program switches once more to the node-selection. We save the results and stop.

0	Select node for next curve
S	saving option
X	scenario option

```
%CANDYS-I-SAVE, saving scenario No. 1
```

10.1.4 The path-following-level

We start the program anew, we choose to restart the former scenario:

R	scenario option

Now, the program is in the same state as before saving the results (at the end of the previous section). We want to continue the existing short curve at both endpoints, first into the direction of larger r values:

3	Select node for next curve
!	Select node for next curve

The limits of the curve have to be specified. Before, those variables have to be chosen for which limits are to be set. The single varied parameter r belongs automatically to the set of such variables. We choose that all other variable should be unlimited (by entering the name "0"). Then the program asks for the limits and numerical control parameters:

Yes	Do you want to change the set of variables to be limited
0	Enter variable to be limited
10.000	lower limit
30.000	upper limit
No	Do you want to change the parameters of Newton-Routine
No	Do you want to change the integration constants
0.5	MaxCurveStep
5.0000E-5	MinCurveStep
50.000	MaxArcLength
0.0010	EpsPredic

The program switches to the path-following:

```
   ********************************
      Tracing the selected curve
   ********************************

Graph 2
Curve type: steady-state    stable
Parameters:
  r=***  sigma=10  b=2.66667
Limits:
  r     varies within  [      10.000 ...       30.000]


Node no. 3
Parameters         20.069504
State              7.1310596        7.1310596        19.069504
Quantity          -20.458585

Node no. 3
Parameters         20.139012
```

```
State              7.1440441        7.1440441        19.139012
Quantity          -20.403075
Nontrivial eigen values :
Real             -0.14987322      -0.14987322       -13.366920
Imag               8.7374027       -8.7374027        0.0000000

 ...
```

A Hopf bifurcation point has been found:

```
Node no. 4
Parameters         24.736842
State              7.9560195        7.9560195        23.736842
Quantity          -16.261949
Nontrivial eigen values :
Real              -13.666691
Imag               0.0000000
Trivial eigen values :
Real           1.2066770E-05    1.2066770E-05
Imag               9.6246025       -9.6246025
HopfBif
%CANDYS-I-OK, special point computed
```

Next, a stump of the branching-off curve of cycles will be computed. But the Poincaré surface has to be chosen before:

E	section mode
z	Extrema of which axis
No	Do you want to change the integration constants

```
Node no. 5
Parameters         24.735984
Return time =     0.6528883C
State              8.0436799        7.9468740        23.970792
Quantity          -14.767954
Nontrivial eigen values :
Modulus            1.0000904    0.00013341231
Degrees            0.0000000        0.0000000
Trivial eigen values :
Modulus           0.99999910
Degrees            0.0000000
%CANDYS-I-OK, 1 of 1 branches computed
```

The curve of steady-states is continued beyond the Hopf bifurcation point until the limit of the parameter interval is reached:

```
Node no. 3
Parameters          24.736842
State               7.9560195         7.9560195         23.736842
Quantity           -16.261949
Nontrivial eigen values :
Real            0.0033408768      0.0033408768       -13.673348
Imag                9.6445951        -9.6445951         0.0000000

 ...

Node no. 3
Parameters          30.000000
State               8.7939373         8.7939373         29.000000
Quantity           -10.606040
Nontrivial eigen values :
Real               0.14736835        0.14736835       -13.961403
Imag                10.524252        -10.524252         0.0000000
%CANDYS-I-OK, limit of parameter interval
```

Now, we continue the curve at the other end, but no interesting things occur:

2	Select node for next curve
!	Select node for next curve
No	Do you want to change the set of variables to be limited
10.000	lower limit
30.000	upper limit
No	Do you want to change the parameters of Newton-Routine
No	Do you want to change the integration constants
0.50000	MaxCurveStep
5.0000E-5	MinCurveStep
50.000	MaxArcLength
0.0010	EpsPredic

```
   ********************************
      Tracing the selected curve
   ********************************


Graph 2
Curve type: steady-state    stable
Parameters:
  r=***  sigma=10  b=2.66667
Limits:
  r     varies within  [      10.000 ...       30.000]
```

```
Node no. 2
Parameters          19.930500
State               7.1050218         7.1050218         18.930500
Quantity           -20.568883

Node no. 2
Parameters          19.748568
State               7.0707979         7.0707979         18.748568
Quantity           -20.711796
Nontrivial eigen values :
Real               -0.16373540       -0.16373540       -13.339196
Imag                8.6564727        -8.6564727         0.0000000

 ...

Node no. 2
Parameters          10.000000
State               4.8989795         4.8989795         9.0000000
Quantity           -24.989795
Nontrivial eigen values :
Real               -0.59549806       -0.59549806       -12.475671
Imag                6.1741652        -6.1741652         0.0000000
%CANDYS-I-OK, limit of parameter interval
```

We want to continue the stump of the curve of cycles at the Hopf bifurcation. Thus, we choose node 5 (recall that it is possible to step through the list of points by repeatedly hitting the RETURN key):

5	Select node for next curve
!	Select node for next curve
No	Do you want to change the set of variables to be limited
18.000	lower limit
30.000	upper limit
No	Do you want to change the parameters of Newton-Routine
No	Do you want to change the integration constants
0.50000	MaxCurveStep
5.0000E-5	MinCurveStep
50.000	MaxArcLength
0.0010	EpsPredic

```
 ...

Node no. 5
Parameters          20.384167
Return time =      0.85222000
State               10.709513         6.8505190         27.512146
```

```
Quantity           46.188475
Nontrivial eigen values :
Modulus            1.3417274    6.5177878E-06
Degrees            0.0000000        0.0000000
Trivial eigen values :
Modulus            1.0000354
Degrees            0.0000000
```

Here, the curve crosses the value $r = 20.0$, the value of r in graph 1. The operation (3) explained in Sec. 3.3 becomes active. The program switches temporarily to graph 1 and inserts a new node there:

```
   ****************************
      Switch to another graph
   ****************************


Graph 1
Parameters:
  r=20  sigma=10  b=2.66667


Node no. 6
Return time =     0.87655293
State              10.739693        6.7827647        27.316805
Quantity           47.513367
```

Since no curves are branching-off from this new node, the program immediately switches back to the path-following in graph 2:

```
   ******************************
      Tracing the selected curve
   ******************************


Node no. 5
Parameters         19.948673
Return time =     0.87992478
State              10.742807        6.7738236        27.288706
Quantity           47.669676
Nontrivial eigen values :
Modulus            1.4107811    4.2456688E-06
Degrees            0.0000000        0.0000000
Trivial eigen values :
Modulus            1.0000356
Degrees            0.0000000
```

```
 ...

Node no. 5
Parameters          18.000000
Return time =       1.0357519
State               10.720931        6.4487004        25.926027
Quantity            50.451355
Nontrivial eigen values :
Modulus             1.9422404    3.6665796E-07
Degrees             0.0000000        0.0000000
Trivial eigen values :
Modulus             1.0000447
Degrees             0.0000000
%CANDYS-I-OK, limit of parameter interval
```

We have entered once more the process of node-selection. You can check that really one more node is contained in graph 1. To this end, choose "_" and then press several times the RETURN key. You will see that the graph has two nodes: the steady-state (our starting point) and the cycle inserted during the last path-following.

We save the results and stop the program:

0	Select node for next curve
S	saving option
X	scenario option

```
%CANDYS-I-SAVE, saving scenario No. 1
```

10.1.5 A 2-parameter graph

In this section we create and build up a 2-parameter graph by the path-following of the Hopf bifurcation in graph 2.

We restart the former scenario and select the Hopf bifurcation point (node 4) for the creation of a new graph:

4	Select node for next curve
?	Select node for next curve
sigma	Enter variable parameter to be varied
0.100000	Length of initial step

The program creates the 2-parameter graph with variable parameters r and σ.

```
   ****************************************
      Creation of a new bifurcation graph
```

```
   ****************************************


Graph 3
Parameters:
  r=***  sigma=***  b=2.66667


Node no. 7
Parameters          24.732404         9.9688798
State               7.9552757         7.9552757         23.732404
Quantity           -16.018776
Nontrivial eigen values :
Real               -13.635645
Imag                0.0000000
Real           -3.9225876E-06    -3.9225876E-06
Imag                9.6195960        -9.6195960
Hopf bifurcation

Node no. 8
Parameters          24.741508         10.030910
State               7.9568014         7.9568014         23.741508
Quantity           -16.503269
Nontrivial eigen values :
Real               -13.697870
Imag                0.0000000
Trivial eigen values :
Real           -1.1883186E-05    -1.1883186E-05
Imag                9.6294368        -9.6294368
Hopf bifurcation
%CANDYS-I-OK, 2 of 2 endpoints computed
```

The program checks whether all 1-parameter graphs with variable parameters r or σ exist. The graph for r exists but that for parameter σ does not. Thus, this graph is created additionally originating from graph 1 (no variable parameters). Since graph 1 already contains 2 nodes (node 1, a steady-state, and node 6, a cycle), two short curves are computed:

```
   ****************************************
      Creation of a new bifurcation graph
   ****************************************


Graph 4
Parameters:
  r=20  sigma=***  b=2.66667
```

```
Node no. 9
Parameters          9.9006347
Return time =      0.87694231
State               10.736412         6.7821632         27.306038
Quantity            48.122829
Nontrivial eigen values :
Modulus             1.3984229     4.8675826E-06
Degrees             0.0000000         0.0000000
Trivial eigen values :
Modulus             1.0000284
Degrees             0.0000000

  ...

Node no. 12
Parameters          10.100000
State               7.1180522         7.1180522         19.000000
Quantity           -21.225660
Nontrivial eigen values :
Real               -0.15602733       -0.15602733       -13.454612
Imag                8.7203066        -8.7203066         0.0000000
%CANDYS-I-OK, 4 of 4 endpoints computed
```

Now, we can perform the path-following of the curve of Hopf bifurcations:

7	Select node for next curve
!	Select node for next curve
No	Do you want to change the set of variables to be limited
2.0000	lower limit
30.000	upper limit
1.0000	lower limit
20.000	upper limit
No	Do you want to change the parameters of Newton-Routine
No	Do you want to change the integration constants
1.0000	MaxCurveStep
0.0001	MinCurveStep
100.00	MaxArcLength
0.0001	EpsPredic

```
   ********************************
      Tracing the selected curve
   ********************************

Graph 3
Curve type: steady-state    HopfBif   stable
Parameters:
  r=***  sigma=***  b=2.66667
Limits:
```

```
  r     varies within  [      2.0000 ...       30.000]
  sigma varies within  [      1.0000 ...       20.000]


Node no. 7
Parameters         24.732404        9.9688798
State              7.9552757        7.9552757        23.732404
Quantity          -16.018776

Node no. 7
Parameters         24.721929        9.8863557
State              7.9535198        7.9535198        23.721929
Quantity          -15.372848
Nontrivial eigen values :
Real              -13.553056
Imag               0.0000000
Trivial eigen values :
Real          -1.3554323E-06   -1.3554323E-06
Imag               9.6067013       -9.6067013

  ...

Node no. 7
Parameters         30.000000        5.9999999
State              8.7939373        8.7939373        29.000000
Quantity           24.569711
Nontrivial eigen values :
Real              -9.6666665
Imag               0.0000000
Trivial eigen values :
Real          -1.0886650E-08   -1.0886650E-08
Imag               9.7979590       -9.7979590
%CANDYS-I-OK, limit of parameter interval
```

Similarly, we can select node 8 for the path-following, and we will find that nothing interesting happens. Thus, we stop:

0	Select node for next curve
S	Enter saving option

```
%CANDYS-I-SAVE, saving scenario No. 1
```

X	Enter scenario option

Chapter 11

Graphics

11.1 Introduction

CANDYS/QA has now its own online graphics output module (called OG) that works in connection with JAVA. Besides the interface to OG, CANDYS/QA presently offers interfaces to PGPLOT (online graphics) and GNUPLOT (post-processing). Henceforth, *graphics* denotes any of them, and program stands for CANDYS/QA or the post-processor `g2gnu`.

The graphics systems differ in the amount of different colors, line types and marker types they support (only lines, markers, coordinate axes, and pure texts are plotted). Since the lines and markers plotted by the programs represent different invariant sets and/or special points these should also be plotted in a different fashion. The correspondence between invariant sets and/or special points and their graphical representation is not prescribed by the programs. Instead, the user may provide a file *graphics*`.lmf` (`lmf` stands for line-marker-format) containing a not necessarily complete list of invariant sets and/or special points and their graphical representation (default *graphics*`.lfm` files are available). In Sec. 11.4 the list and its treatment will be discussed in more detail.

11.2 Use of online graphics

The online graphics opens one or more windows and performs the following operations.

- When entering the node selection process (c.f. Sec. 8.4) then the actual bifurcation graph is plotted.
- If during the node selection process the actual graph is changed then the new bifurcation graph is plotted.
- During the node selection process the node offered for selection is indicated by a special marker.

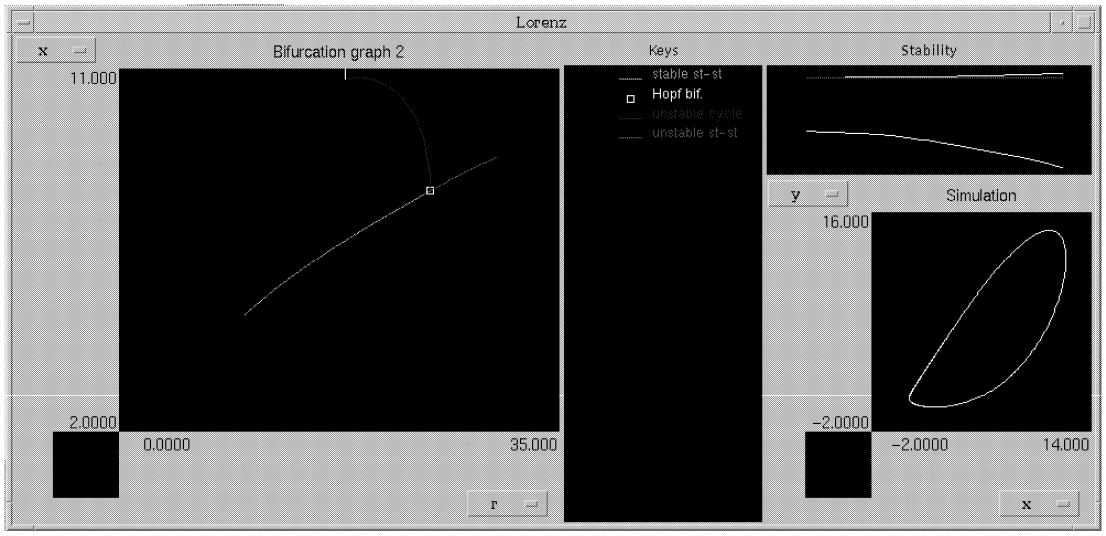


Figure 11.1: The outlook of the OG graphics window.

- During the path-following process the last plotted bifurcation graph (this is that from which a node was selected) is continued by plotting in it the newly computed arcs and special points. If the bifurcation graph is changed temporarily (in the sense of Sec. 3.3) then the plot remains unchanged.

- The OG graphics provides two more windows for showing a simulated trajectories and the variation of eigen values.

11.2.1 Graphics output with 'og'

The outlook of the graphics window is shown in Fig. 11.1. The window is divided into four subwindows. From left to right and then top down these are

- the bifurcation subwindow showing the current bifurcation diagram;

- the keys subwindow explaining the meaning of lines and markers;

- the simulation subwindow showing the current point (i.e. the point computed last during path-following or the one selected during node selection) as simulated trajectory;

- the stability subwindow showing the eigen values (more precisely, $\Re\lambda_i$ or $\log|\lambda_i|$) vs. the arclength during path-following (the red straight line indicates zero).

The subwindows are divided into several areas. The main parts are the canvas areas containig the figures. The bifurcation and simulation subwindows contain some more areas: a coordinate area and a panner area (the black square in the left bottom corner, its meaning will be explained below).

The configuration of the OG graphics can be controled by the following command line options if CANDYS/QA is run under UNIX:

`-g` *n*	:	the graphics will contain only *n* subwindows
	:	($n = \{0, 2, 4\}$, default: 4)
`-w` *width*	:	determines the width of the bifurcation subwindow, the other subwindows will be scaled proportionally (default: 400)

The simulation and stability subwindows are plotted by the special Handle module `oghdl`. [2] If another Handle module is applied then both subwindows remain empty, in that case the option `-g2` makes sense. The option `-w` allows to adapt the subwindow sizes to the screen size; the default works well in case of a 1280 by 1024 pixels screen.

The bifurcation and simulation subwindows can be manipulated interactively without influencing the computations. The following kinds of manipulations have been implemented:

- determining coordinates of a point
- resizing the window (this works for the stability subwindow, too)
- zooming the canvas area (this works for the stability subwindow, too)
- moving the contents of a canvas area
- writing a snapshot of a canvas area to a POSTSCRIPT file (this feature works for all subwindows)

Determining point coordinates When the mouse pointer is moved inside the canvas area a crosshair is drawn at the current pointer position and the position is written into the frame area. For fine tuning, the crosshair can also be moved by pressing the arrow keys.

Resizing The common resize manipulations of the window manager can be applied: the subwindows are resized proportionally and the contents of the canvas areas are rescaled.

Zooming Dragging the mouse pointer within the canvas area while pressing the 1^{st} mouse button defines the two diagonal corners of an axis parallel rectangle which replaces the crosshair. After releasing the button, the figure in the rectangle is zoomed-in to the full canvas area size (or the figure in the area is zoomed-out to the rectangle if the shift key is depressed simultaneously).

The zoom is represented in the panner area: a small rectangle within that area represents the position and size of the chosen zoom rectangle relative to the whole canvas area. The following color convention is applied. In case of a zoom-in the small rectangle is black while the panner area's background is gray; in case of a zoom-out the colors are reversed. In each case, the black region in the panner area represents the current canvas area (i.e. its black background) while the gray region represents the former canvas area which has been zoomed.

[2] Don't forget to set it in the `makefile`.

The zooming can be repeated any times. All zoomings are stored in a stack. Older zoomings can be recovered by a double clicking the 1^{st} mouse button; thereby, newer zoomings are lost. A triple click removes all zoomings. The zoomings are also removed when the bifurcation graph is changed during node selection.

Moving The contents of a zoomed canvas area can be moved. To this end, the 1^{st} button is to be dragged in the panner area. While dragging the mouse the small rectangle in the panner area is shifted and the same time the figure in the canvas area iis moved to the position indicated by the small rectangle. This way, the small rectangle acts like a two-dimensional scrollbar.

Choosing the projection The computed points and lines often contain more than two coordinates, two of them define the two-dimensional projection currently applied. The projection (i.e. the two coordinates) can easily be changed. Clicking on one of the coordinate labels pops up the list of all coordinate names from which another coordinate can be chosen.

Writing a POSTSCRIPT **file** Pressing the *F6* function key when the mouse pointer is in one canvas area pops up a file dialog window. After choosing a file the area contents is written in EPS format. The colors black and white are exchanged such that the black background of the graphics becomes the wite background of the, in general, white printing paper.

The POSTSCRIPT file written can be slightly adapted to your needs. Below the comment line "`% --- Editable entities: ---`" you can redefine the name of the headline (if the one generated automatically is meaningless), choose whether or not the clip rectangle is to be drawn, set a scaling factor for string sizes, and redefine the relative width of lines. The two last entities may be important when the figure is rescaled during printing such that strings (lines) would become too small (thin) or to large (thick) otherwise.

11.2.2 Graphics output with 'PGPLOT'

This graphics uses the online graphics system PGPLOT. [3] It is relatively poor. It provides one window only and does not allow interactive manipulations. There is only one way to control the plot: if more than two variables are available then one can choose the variables to be plotted. This is done during the node selection process by pressing the `&` key (c.f. Sec. 8.4):

& *Change the variables to be plotted*

The program then asks for the names of the two variables to be plotted. The PGPLOT graphics provides one command line option:

`-g0` : the graphics output will be suppressed

[3] PGPLOT is a public domain software, information can be obtained via WWW:
`http://astro.caltech.edu/~tjp/pgplot/` .

11.3 Post-processor for 'gnuplot'

11.3.1 Overview

One post-processing program, *g2gnu*, has been provided. It transforms the graphics files *ScenarioName*.`g`*xx* into files readable by GNUPLOT. [4] It may be used for producing 2D-graphics or 3D-graphics of the bifurcation diagrams and converting them to POSTSCRIPT files. The graphics files of several scenarios can also be merged in one run of the post-processor provided that in the corresponding graphs the same parameters were varied. In the simplest case the several graphs overlay in one graphics. It is also possible to plot several 2D-bifurcation diagrams having the same variable parameters but differing in the value of specific fixed parameter in a single 3D-graphics. Here, the 3^{rd} dimension is given by this specific parameter.

The files generated get a name defined by dialog with predescribed extensions. The extension is `.gpl` in the case of GNUPLOT control files, and `.g`*tx* in the case of GNUPLOT data files, where *t* stands for `l` (i.e. lines) or `p` (i.e. points), and *x* stands for the number of line or point type. The files are accessed in GNUPLOT by the command
`load '`*filename*`.gpl'`

The resulting graphics is formatted such a way that

- the full bifurcation diagram is shown and that it does not touch the frame;
- the *ScenarioName* becomes the title, and the names of the chosen variables become the axis labels;
- the axes are decorated with nice tic values;
- right to the figure the meaning of different line and point types is written (i.e. the keys table of the figure);
- the `output` is set to a file with the extension `.ps` and the name part of *filename*`.gpl`;
- the `terminal` is set to `x11`.

Note

The bounding box of the generated POSTSCRIPT file is not correct. To correct it, 200 has to be added to the 3^{rd} number in the line which starts with `%%BountingBox: .`

11.3.2 Dialogue

The post-processor is interactive and use the input formats as described in Sec. 8.1. After program start by the command
`g2gnu`
The Dialogue is as follows.

[4] GNUPLOT is a public domain software, information can be obtained via e-mail: `info-gnuplot@dartmouth.edu` .

1. The type of graphics:
   `Enter plot` *whether a 2D-graphics, 3D-graphics, or multiple 2D-graphics in 3D has to be produced.*
   Default: **2**

2. The model:
   `Enter model name`
   Default: **CANDYS**

3. A series of questions asking for the names of parameters that are variable in the graphs to be plotted. Entering **0** stops the series of questions.
   `Enter variable parameter`
   Default: **0**

4. If several scenarios of the chosen model exist then a series of questions follows asking for a scenario. Entering **0** stops the series of questions.
   `Enter no. of scenario`
   Default: **0**
   If a chosen scenario contains more than one graph with the variable parameters selected in step (3), then a series of questions follows asking for a graph number. Entering **0** stops the series of questions.
   `Enter graph`
   Default: **0**

5. Questions asking for the names of two or three variables to be plotted:
   `Enter state or parameter for 1st axis`
   Default: **1st variable**
   And so on.

6. `Enter maximum number of cycle points`
   Default: **1**

7. `Enter file name (without extension)`
   Default: var1_var2_var3
   where *var1* is the name of the 1st variable to be plotted etc. (in case of a 2D-graphics the part *_var3* is omitted)

8. `One more plot` *If "yes" then continuing with step 5*
   Default: **No**

11.3.3 Example

The example of Sec. 10 produced, among others, the graphics file `Lorenz_01.g02` (i.e. the graphics file of graph number 2 of the scenario 1). This graph is to be displayed as a plot of **x** vs. **r** using GNUPLOT. Applying the same convention of representation as in Sec. 10, the input is as follows:

2 Enter plot

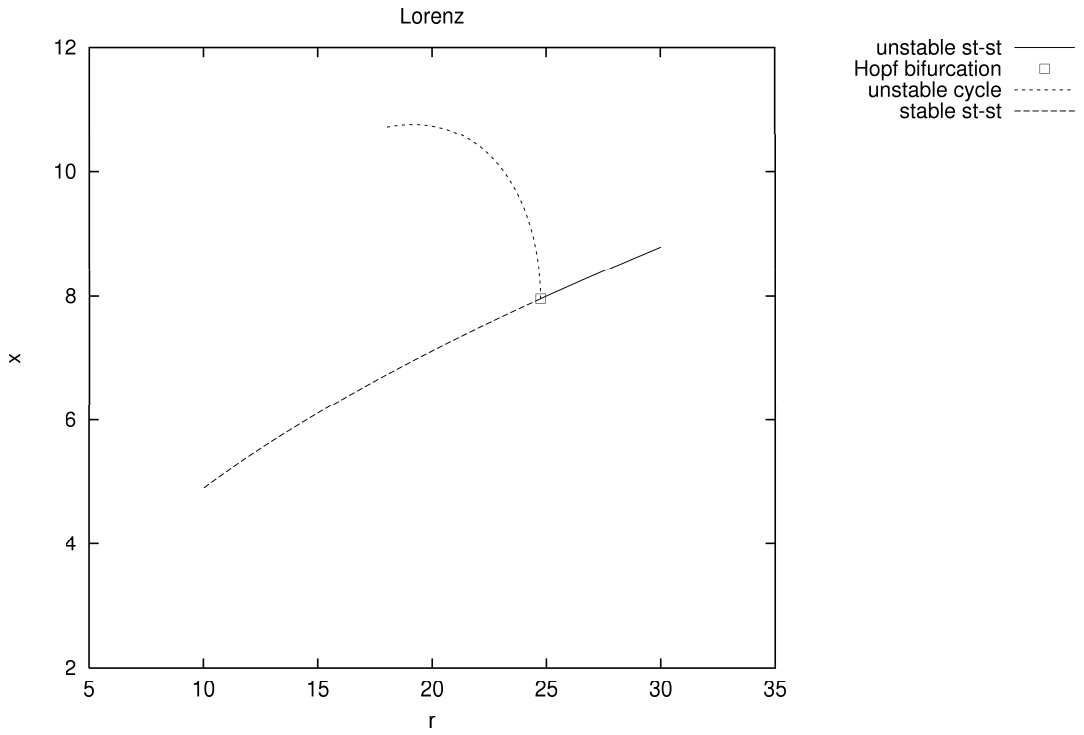


Figure 11.2: The bifurcation diagram of the example in Sec. 10

Lorenz	Enter model name
1	Enter no. of scenario
2	Enter no. of graph
r	Enter state or parameter for 1^{st} axis
x	Enter state or parameter for 2^{nd} axis
1	Enter maximum number of cycle points
r_x	Enter file name (without extension)
No	One more plot

The result of GNUPLOT (via POSTSCRIPT) is shown in Figure 11.2.

11.4 The lmf file

11.4.1 Overview

The file *graphics*.`lfm` is read in during the startup phase of the programs building up an interior list. When a line or a marker is to be plotted and if a representation is given in the list then it will be used for plotting, otherwise a default one is used. At the end of the programs a new file is written that contains additionally to the representations read in also the invariant sets and/or special points plotted with default representations. Now, you can edit the file, and the next run of the programs uses the edited representations. This way, you can maintain the list as complete as you need.

To be more precise, the file contains a table where each entry has three parts:

1. a characteristicx of the invariant set and/or special point to be plotted (e.g. a line of stable steady-states);
2. the name of its key in the graphics' keys subwindow;
3. its associated line or marker format as an *ASCII* string.

If the same characteristic occurs several times then a later entry overwrites the former ones.

Finally, during the termination phase of the programs a new *graphics*.`lmf` file is written where each new entry is marked as a comment.

11.4.2 Syntax

A version of each file *graphics*.`lmf` is shipped with the CANDYS/QA sources. Nevertheless, users may (and should) provide and update their own versions. In the following their format will be described.

1. Line structure:
   (a) each entry fills one (logical) line;
   (b) long entries can be split by closing the physical line with a \ (backslash);
   (c) empty lines and lines starting with a `#` are considered as comments;
   (d) the fields of each entry have to be separated by blanks;
   (e) the entries may be in any order while the fields in each entry have a predescribed ordering.
2. The 1^{st} part of each entry consists of the following fields:
   (a) the letter `L` or `M` defining the entry as a line or marker format, respectively;
   (b) the number `0` in the case of steady-states, or the period in the case of cycles;
   (c) the type of special point in the same format as described in Sec. 9.2;
   (d) the string `stable` or `unstable` in the case of lines.
3. The 2^{nd} part of each entry is a string enclosed in quotation marks. This string is considered as a single field: it has to be separated from the other fields by blanks but interior blanks do not separate the string into several fields. The quotation marks serve as separators of this part from the other parts.
4. The 3^{rd} part of each entry describers the outlook of the lines and markers by predefined key words like `red` or `cross` (c.f. Sec. 11.4.4).

The fields in the 1^{st} and 3^{rd} part are case insensitive.

Since there are many possibilities of invariant sets and special points (e.g. the different period length in the case of a period doubling cascade) as many entries have to be

provided. To shorten the effort of writing such a long file, one can use the wildcard `*` in each of the fields (2b), (2c), and (2d). Its effect is as follows. When an entry is searched in the internal list and no exact match is found for a specific field then the search is continued looking for an entry with a wildcard in the corresponding field position. If several entries exist matching the searched one by wildcards then that is selected matching most fields from the left.

11.4.3 Example

Typical lines of the *graphics*`.lmf` file look like

```
L 0  stable   "stable steady-states"  green
M 1  Turning  "turning point"  star
M 0  Turning HopfBif  "turning+Hopf"  cross
```

This means,

- the first entry specifies a line format while the other two specify marker formats;
- the first and last entry specify steady-states (the period length is zero) while the second entry specifies a cycle of period length 1;
- the first two entries specify a line and a marker in 1-parameter graphs (the characteristic contains one field) while the last entry specifies a marker in 2-parameter graphs.

The following demonstrates how the entry selection works if they contain fields in form of wildcards. Assume that

```
L 1 * "..." ...
L * * "..." ...
```

are given and that the entry

```
L 1 stable
```

is searched for. Then both entries match and the first entry is selected since it matches best. But if

```
L 2 stable
```

is searched then the 2^{nd} entry is selected since it is the only matching one.

11.4.4 Rendering key wordss

The `lmf` file for OG graphics is `og.lmf`. Bifurcation lines are drawn as solid lines of a specific color. The color is coded by one of the following color names:

white, lightgray, gray, darkgray, red, pink, orange, yellow, green, magenta, cyan, blue, darkred, darkgreen, darkblue, brown, khaki.
Bifurcation points are drawn as white markes of a specific shape. The shape is coded by one of the following names:
dot, circle, square, diamond, star, plus, cross, up, down, left, right.
The last four denote triangles pointing in the indicated direction; the others need no explanation.

The lmf file for PGPLOT graphics is pg.lmf. Bifurcation lines are drawn as solid lines of a specific color. The color is coded by one of the following color names:
white grey, red, darkred, orange, yellow, brown, green, darkgreen, magenta, blue, darkblue, cyan.
Bifurcation points are drawn as white markes of a specific shape. The shape is coded by one of the following names:
dot, circle, square, diamond, star, plus, cross, astroide, fife_star, david_star, delta, full_delta, david_star.

The lmf file for the post-processor is g2gnu./lmf. Just like the POSTSCRIPT files produced by the OG graphics the colors white and black are exchaged. Bifurcation lines are drawn as solid lines of a specific color. The color is coded by one of the following color names:
black, red,green, blue, magenta, cyan, brown, yellow,
Bifurcation points are drawn as black markes of a specific shape. The shape is coded by one of the following names:
dot, delta, square, diamond, star, plus, cross.

11.4.5 Error messages

If during the reading of the *graphics*.lmf file an error occurs, e.g. an unknown field or missing quotation marks, then a warning is issued in the style of Sec. 9.3:
%CANDYS-W-FORMERROR, *reason* in line *n*
Here, *n* denotes the entry's (last physical) line in the file where the error occurred, and *reason* is a verbal description of the error. In the following all reasons are listed alphabetically without any comment:

incorrect period

invalid stability

neither line nor marker

no label

not enough fields

unclosed label string

unknown field '*field*'

Moreover, the file is copied to the file *graphics*.lm0. The error messages refer to this file since during program termination a new (and correct) file *graphics*.lmf is written omitting the erreneous lines.

If during the writing of the new file entries with default representation occur then the warning
`%CANDYS-W-FORMERROR, new formats written to file` *graphics*`.lmf`
is issued. The new lines are indicated by their 2^{rd} part: "`??`".

Appendix A

Prototype of the *Model* module

```
     /* -------- */
     /* Includes */
     /* -------- */

#include "Model.h"
#include "LinAlgebra.h"
#include "Errors.h"
#include "Strings.h"
#include "math.h"


     /* --------- */
     /* Functions */
     /* --------- */

void Model::Rhs
  (double Time,const Vector& X, Vector& F)
{
  /* Right hand side of system equations:
     F is a function of X and Time */
  F[0] = /* ... */ ;
  /* and so on for the other components of F */
}

void Model::Jacobi
  (double Time,const Vector& X, Matrix& J)
{
  /* Jacobian matrix of system equations with respect
     to state variables: J is a function of X and Time */
  J(0,0) = /* ... */ ;
  /* and so on for the other components of J */
}

void Model::JacobiParam
```

```
  (double Time,const Vector& X, int Pidx, Vector& JP)
{
  /* Jacobian matrix of system equations with respect
     to parameters: JP is a function of X and Time */
  switch (Pidx) {
    case 0:
      JP[0] = /* ... */ ;
      /* and so on for the other components of JP */
      break;
    case 1:
    /* and so on for the other cases of Pidx */
    default:;
  }
}

void Model::QuantRhs
  (const Vector& X,int Qidx,double *Q)
{
  /* state dependent quantities */
  switch (Qidx) {
    case 0:
      *Q = /* ... */ ;
      break;
    /* and so on for the other cases of Qidx */
    default:;
  }
}

void Model::QuantJacobi
  (const Vector& X, int Qidx, Vector& QX)
{
  /* Derivation of quantities with respect to state variables */
  switch(Qidx) {
    case 0:
      QX[0] = /* ... */ ;
      /* and so on for the other components of QX */
      break;
    /* and so on for the other cases of Qidx */
    default:;
  }
}

void Model::QuantJacobiParam
  (const Vector& X, int Qidx, int Pidx, double *QP)
{
  /* Derivation of quantities with respect to parameters */
  switch (Pidx) {
    case 0:
      switch (Qidx)
```

```
      { case 1:
          *QP = /* ... */ ;
          break;
        /* and so on for the other cases of Qidx */
        default:;
      }
      break;
    case 1:
    /* and so on for the other cases of Pidx */
    default:;
  }
}


     /* -------------- */
     /* Initialization */
     /* -------------- */

Model::Model(int argc,char **argv)
{
  /* Model configuration: */
ModelClass =     /* autonomous, periodical, or difference */ ;
Dim =            /* number of state variables */ ;
NbOfParams =     /* number of parameters */ ;
NbOfOutputs =    /* number of state variables to be output */ ;
NbOfQuantities = /* number of implemented quantities */ ;
hasJacobi =      /* 1 if all procedures computing Jacobians
                    have been implemented */ ;
stiff =          /* 1 if the differential equations may be stiff */ ;

  /* Names: */
ModelName =      /* string for names of the files created */ ;
TName =          /* string for name of the
                    independent variable Time */ ;
PName.zero(NbOfParams);
PName[0] =       /* string for name of the parameter P[0] */ ;
                 /* and so on for the other components of P */
XName.zero(NbOfOutputs);
XName[0] =       /* string for name of the variable X[Oindex[0]] */ ;
                 /* and so on for the other components of X */
QName.zero(NbOfQuantities);
QName[0] =       /* string for name of the quantity Q[0] */ ;
                 /* and so on for the other components of Q */

  /* Scales: */
TScale =         /* scaling factor for Time,
                    normally equal to 1.0 */ ;
PScale.zero(NbOfParams);
PScale[0] =      /* scaling factor for P[0],
```

```
                   normally equal to 1.0 */ ;
                /* and so on for the other components of P  */
XScale.zero(NbOfOutputs);
XScale[0] =     /* scaling factor for X[Oindex[0]],
                   normally equal to 1.0 */ ;
                /* and so on for the other components of X  */

  /* Initial values: */
X0.zero(Dim);
X0[0] =         /* initial value for X[0] */ ;
                /* and so on for the other components of X */
P.zero(NbOfParams);
P[0] =          /* initial value for P[0] */ ;
                /* and so on for the other components of P */
}
```

Bibliography

[1] ALLGOWER, E. L.: A survey of homotopy methods for smooth mappings. In: *Numerical Solution of Nonlinear Equations* (Berlin, Heidelberg, New York, 1981), E. L. Allgower, K. Glashoff, and H.-O. Peitgen (Eds.), vol. 878 of *Lect. Notes in Math.*, Springer Verlag, pp. 1–29.

[2] ALLGOWER, E. L., AND GEORG, K.: *Numerical Continuation Methods. An Introduction*, vol. 13 of *Springer Series in Computationsl Mathematics*. Springer, Berlin, Heidelberg, New York, 1990.

[3] ARNOL'D, V. I.: *Geometrical Methods in the Theory of Ordinary Differential Equations*. Springer-Verlag, Berlin, Heidelberg, New York, 1983.

[4] ARROWSMITH, D. K., AND PLACE, C. M.: *An Introduction to Dynamical Systems*. Cambridge Univ. Press, Cambridge, 1990.

[5] BOGDANOV, R. I.: Bifurcation on the limit cycle of a family of plane vector fields. *Sel. Math. Sov.* **1** (1981), 373–388.

[6] BOGDANOV, R. I.: Versal deformation of a singularity of a vector field on the plane in the case of zero eigenvalues. *Sel. Math. Sov.* **1** (1981), 389–421.

[7] CURRY, J. H.: An algorithm for finding closed orbits. In: *Global Theory of Dynamical Systems* (Berlin, Heidelberg, New York, 1980), Z. Nitecki and C. Robinson (Eds.), vol. 819 of *Lect. Notes Math.*, Springer-Verlag, pp. 111–120.

[8] DOEDEL, E. J., AND KERNEVEZ, J. P.: *Software for Continuation Problems in Ordinary Differential Equations with Application*. California Institute of Technology, Pasadena, 1985.

[9] EIDNER, R., FEUDEL, U., AND JANSEN, W.: The interaction of selfpurification and nutrient load in water quality models. - a study of long time behaviour. Preprint, Zentralinstitut für Kybernetik und Informationsprozesse & Institut für Wasserwirtschaft, Berlin, 1988.

[10] EIDNER, R., FEUDEL, U., AND JANSEN, W.: Sensitivity analysis for the dynamics of the interaction between nitrogen compounds and bacteria in a river. *Syst. Anal. Model. Simul.* **8** (1991), 429–436.

[11] FEUDEL, U., AND JANSEN, W.: CANDYS/QA - a software system for the qualitative analysis of nonlinear dynamical systems. *Int. J. Bifurcation & Chaos* **2** (1992), 773–794.

[12] FEUDEL, U., JANSEN, W., AND KURTHS, J.: Tori and chaos in a nonlinear dynamo model for solar activity. *Int. J. Bifurcation & Chaos* **3** (1993), 131–138.

[13] GARCIA, C. B., AND ZANGWILL, W. I.: *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice-Hall Inc., Englewood Cliffs, 1981.

[14] GATERMANN, K., AND WERNER, B.: Group theoretical mode interactions with different symmetries. *Int. J. Bifurcation & Chaos* **4** (1994), 177–191.

[15] GEORG, K.: *Zur numerischen Realisierung von Kontinuitätsmethoden mit Prädiktor-Korrektor- oder simplizialen Verfahren*. Habilitationsschrift, Rheinische Friedrich-Wilhelms-Universität, Bonn, 1982.

[16] GOLUBITSKY, M., STEWART, I., AND SCHAEFFER, D. G.: *Singularities and Groups in Bifurcation Theory II*, vol. 2. Springer-Verlag, Berlin, Heidelberg, New York, 1988.

[17] GOVAERTS, W.: Computation of Takens-Bogdanov type bifurcations with arbitrary codimension. *SIAM J. Numer. Anal.* **30** (1993), 1121–1133.

[18] GRIEWANK, A., AND REDDIEN, G. W.: The calculation of Hopf points by a direct method. *IMA J. Numer. Anal.* **3** (1983), 295–303.

[19] GUCKENHEIMER, J., AND HOLMES, P.: *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer Verlag, Berlin, 1983.

[20] HASSARD, B. D., KAZARINOV, N. D., AND WAN, Y. H.: *Theory and Application of the Hopf Bifurcation*. Cambridge University Press, Cambridge, 1981.

[21] HOLODNIOK, M., AND KUBIČEK, M.: DERPER - an algorithm for the continuation of periodic solutions in ordinary differential equations. *J. Comp. Phys.* **55** (1984), 254–267.

[22] HOPF, E.: Abzweigung einer periodischen Lösung von einer stationären Lösung eines Differentialsystems. *Berichte der Mathematisch-Physiskalischen Klasse der Sächsischen Akademie der Wissenschaften zu Leipzig* **94** (1942), 15–25.

[23] JEPSON, A. D., AND KELLER, H. B.: Steady state and periodic solution paths: their bifurcation points and computations. In: *Numerical Methods for Bifurcation Problems* (Basel, 1984), T. Kuepper, H. D. Mittelmann, and H. Weber (Eds.), vol. 70 of *Internat. Ser. Numer. Math.*, Birkhäuser Verlag, pp. 219–246.

[24] KAAS-PETERSEN, C.: Computation, continuation, and bifurcation of torus solutions for dissipative maps and ordinary differential equations. *Physica* **25D** (1987), 288–306.

[25] KELLER, H. B.: Numerical solution of bifurcation and nonlinear eigenvalue problems. In: *Application of Bifurcation Theory*, P. Rabinowitz (Ed.). Academic Press, New York, 1977, pp. 359–384.

[26] KHIBNIK, A. I.: Periodical solutions of systems of n differential equations. Preprint, Akademiya Nayk SSSR, Pushchino, 1979. In Russian.

[27] KHIBNIK, A. I., AND SHNOL', E. E.: Programs for qualitative analysis of differential equations. Preprint, Akademiya Nayk SSSR, Pushchino, 1982. In Russian.

[28] KRUG, H.-J., POHLMANN, L., AND KUHNERT, L.: Analysis of the modified complete Oregonator accounting for oxigin sensitivity and photosensitivity of Belousov-Zhabotinski systems. *J. Phys. Chem.* **94** (1990), 4862–4866.

[29] KUBIČEK, M., AND MAREK, M.: *Computational Methods in Bifurcation Theory and Dissipative Structures*. Springer Series on Computational Physics. Springer-Verlag, Berlin, Heidelberg, New York, 1983.

[30] LORENZ, E. N.: The problem of deducing the climate from the governing equations. *Tellus* **16**, no. 1 (1964), 1–11.

[31] MARSDEN, J., AND MCCRACKEN, M.: *The Hopf Bifurcation and Its Applications*. Springer, Berlin, Heidelberg, New York, 1976.

[32] MOORE, G., AND SPENCE, A.: The calculation of turning points of nonlinear equations. *SIAM J. Numer. Math.* **17** (1980), 567–576.

[33] PARKER, T. S., AND CHUA, L. O.: INSITE - A software toolkit for the analysis of nonlinear dynamical systems. *Proc. IEEE* **75** (1987), 1081–1089.

[34] PARKER, T. S., AND CHUA, L. O.: *Practical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, Berlin, Heidelberg, New York, 1989.

[35] PÖNISCH, G.: Das Programmsystem TURN zur Berechnung von Rückkehrpunkten. Tech. rep., Technische Universität Dresden, Sektion Mathematik, 1979.

[36] PÖNISCH, G.: Ein implementierbares ableitungsfreies Verfahren zur Bestimmung von Rückkehrpunkten implizit definierter Raumkurven. *Beitr. Numer. Math.* **9** (1981), 147–160.

[37] RECHENBERG, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich-Fromman-Verlag, Stuttgart, 1973.

[38] RINZEL, J., AND MILLER, R. N.: Numerical calculation of stable an unstable periodic solutions to the Hodgkin-Huxley equations. *Math. Biosci.* **49** (1980), 27–59.

[39] ROOSE, D.: An algorithm for the computation of Hopf bifurcation points in comparison with other methods. *J. Comp. Appl. Math.* **12&13** (1985), 517–529.

[40] SCHWEFEL, H. P.: *Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie*. Birkhäuser-Verlag, Basel, 1977.

[41] SCHWETLICK, H., AND CLEVE, J.: Higher order predictors and adaptive steplength control in path following algorithms. *SIAM J. Numer. Anal.* **24** (1987), 1382–1393.

[42] SEYDEL, R.: Numerical computation of branch points in nonlinear equations. *Numer. Math.* **33** (1979), 339–352.

[43] SEYDEL, R.: Berechnung von periodischen Lösungen bei gewöhnlichen Differentialgleichungen. *ZAMM* **63** (1983), T98–T99.

[44] SEYDEL, R.: *From Equilibrium to Chaos*. Elsevier, New York, 1988.

[45] SEYDEL, R.: Tutorial on continuation. *Int. J. Bifurcation and Chaos* **1** (1991), 3–11.

[46] SPARROW, C.: *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*. Springer Verlag, Berlin, Heidelberg, New York, 1982.

[47] TAKENS, F.: .

[48] TIEBEL, R., SCHÜTTE, F.-J., AND MAREYEN, M.: Chaotic motion of the density operator for an exciton-photon system. *Chaos, Solitons & Fractals* **3** (1993), 177–184.

[49] WERNER, B.: Computation of Hopf bifurcation with bordered matrices. Preprint A73, Institut für Angewandte Mathematik, Universität Hamburg, 1993.