Interactive LOCal BIFurcation Analyzer

LOCBIF

version 2

A.I.Khibnik, Yu.A.Kuznetsov,
V.V.Levitin, E.V.Nikolaev

Contents

Part I
INTRODUCTION

1 Introduction to LOCBIF

In this section you'll learn about the general features of **LOCBIF**, the authors of **LOCBIF** and system requirements for using **LOCBIF**. You will be also informed about typographic conventions used in this manual.

1.1 What is LOCBIF ?

LOCBIF is a new powerful tool for bifurcation analysis of ordinary differential equations (ODEs) which depend upon parameters. It allows you to explore interactively the existence and stability of equilibria in dynamical models[1] .

LOCBIF is based on a continuation procedure for relevant local bifurcation curves up to codimension three. Projection of these curves onto the parameter space determines the boundaries of the equilibrium existence and stability.

LOCBIF combines modern results on normal forms and local bifurcations with interactive computer graphics, resulting in a unique integrated environment for ODE model analysis.

LOCBIF plots two dimensional projections of the bifurcation curves during the computation which may be stepwise or automatic.

LOCBIF maintains an archive of ODEs and allows you to specify a new system of ODEs during a session in a simple Pascal-like language. It compiles the right hand sides of ODEs by means of an on-line compiler. Computed curves may be stored on a disk. Stored curves may be plotted and used in further computations.

LOCBIF allows you to study dynamical systems with up to ten phase variables and ten parameters.

[1]Three additional versions of **LOCBIF** are also available: for analysis of fixed points and periodic orbits of iterated maps, periodic solutions of periodically forced ODEs and periodic solutions (limit cycles) of autonomous ODEs. These versions are presented in Appendix C.

LOCBIF is designed for students, teachers and scientists.

1.2 Authors of LOCBIF

LOCBIF combines numerical algorithms developed by A. Khibnik and E. Nikolaev with modern interface designed by Yu. Kuznetsov and V. Levitin.

The numerics of **LOCBIF** are based on an improved version of the LINLBF code developed by A. Khibnik. The new continuation code BEETLE used in **LOCBIF** was developed by E. Nikolaev and A. Khibnik.

The general dialog scenario and graphics window interface of **LOCBIF** were designed and implemented by Yu. Kuznetsov.

The built-in RHS editor and on-line compiler, as well as a manager for system archive, were designed and written by V. Levitin.

As in any large project, **LOCBIF** is more than simple sum of the parts mentioned above. A number of useful features of **LOCBIF** are the result of close collaboration of all the authors.

For general reference concerning **LOCBIF** see (Khibnik, Kuznetsov, Levitin, Nikolaev, 1992). An earlier presentation of the numerical algorithms implemented in **LOCBIF** can be found in (Khibnik, 1990a,b).

The permanent address of the authors of the **LOCBIF**:

Institute of Mathematical Problems of Biology,
Russian Academy of Sciences,
Pushchino, Moscow Region,
142292 Russia.

E-mail: com@impb.serpukhov.su

1.3 System requirements

LOCBIF requires an IBM PC/XT, AT, PS/2 or compatible microcomputer running DOS version 3.0 or higher. **LOCBIF** automatically supports monochrome and color EGA, VGA and Hercules monochrome graphics display adapters. It requires 640K of RAM with no less than 500K free. A math coprocessor is not required but **LOCBIF** will use it if one is present. The math coprocessor will greatly decrease computation time and is strongly recommended.

LOCBIF can be run from a double or high density floppy disk but it will work more efficiently if all its files are on a hard disk.

If your computer has more than 120K of extended memory, **LOCBIF** can use a virtual (RAM) drive installed in this memory to store working files.

1.4 Typographic conventions

The following typographic conventions are used throughout the manual:

Typeface	Description
KEY TERMS	Text in upper case bold typeface indicates a specific term, punctuation or mark that you must type in exactly as shown.
placeholders	Text in italic typeface indicates a field or a general kind of information.
[*options*]	Items inside square brackets are optional.
"Messages"	Quoted text in bold typeface represents program messages.
Key+Names	The names of key or key sequences. Two-key combination Key1+Key2 means that you must depress the first key and, while holding this key down, press the second key and then release both keys.
↩	This is the Enter key.

2 Getting started with LOCBIF

In this section you'll learn about files which come on the **LOCBIF** distribution diskette; the procedure for installing **LOCBIF** on your machine; and how to start and use **LOCBIF** in the case of a very simple problem.

2.1 Files on the LOCBIF distribution diskette

LOCBIF is distributed on a single high density floppy disk which contains a subdirectory **LBEP**[2] with the following files.

LBEP.EXE	The **LOCBIF** manager program. You should execute this program to run **LOCBIF**.
LBMEP.EXE	The system archives maintain program.
LBFEP.EXE	The main program supporting computations and user graphics interface.
LBM.HLP	The help message file for **LBMEP** program.
INIT.DAT	A standard initial data file.
RHS.DAT	A default right hand sides file.
COURB.FON	A font file.

The root directory of the diskette contains:

MSHERC.COM	A driver file for the system equipped with a Hercules monochrome graphics display adapter.
MOUSE.COM	A driver for the Microsoft mouse pointer device (see Appendix D).

2.2 Installing LOCBIF

The first thing you should do is to make a backup copy of the **LOCBIF** distribution diskette. See your DOS manual for relevant instructions. If you

[2]The other subdirectories in the diskette (namely: **LBFP**, **LBPS** and **LBLC**) contain **LOCBIF** versions for fixed points, periodic orbits and limit cycles respectively (see Appendix C).

do not have a hard disk there is no more installation - **LOCBIF** will run from the supplied diskette.

To install **LOCBIF** on a hard (fixed) disk you should create a directory where **LOCBIF**[3] will reside, for example **LBEP**. Then copy all files from the subdirectory **LBEP** of the **LOCBIF** distribution diskette to the created directory. If your hard drive is **C:** and the **LOCBIF** distribution diskette is in the high density floppy disk drive **A:**, then you may enter the following sequence of DOS commands:

C:↩
md LBEP↩
cd LBEP↩
copy A:\LBEP*.*↩

2.3 Starting LOCBIF

Select the directory with all the **LOCBIF** files (**LBEP** in the example above) as the current directory. You can invoke **LOCBIF** using the following command at the DOS command prompt:

LBEP [*drv*] ↩

The optional parameter *drv* is a pathname of the drive and the directory in which temporal **LOCBIF** working files will be located during a session. There should be a free space of approximately $120K$ on this drive. If the *drv* parameter is ommitted, **LOCBIF** places working files into the current directory (more precisely, its subdirectory related to ODE system under investigation) on the current drive.

Notes:

1. If your computer has an extended memory, it is recommended to install a virtual (RAM) drive in this memory and substitute its name for *drv* in the command line. For example, if RAM drive is **E:** use command

LBEP E: ↩

Consult your DOS manual for details on RAM drive installation.

[3] In the most of the following sections we will be dealing exclusively with the **LOCBIF** version for equilibria of autonomous ODEs (**LBEP**). The installation procedure for the other versions is the same if you substitute **EP** by **FP**, **PS**, or **LC** respectively.

2. If the Hercules monochrome graphics display adapter is installed on your computer you must load a special driver before running **LOCBIF** by the command

MSHERC $\hookleftarrow$

You should load this driver only once when you turn on your computer. To do this, add the command **MSHERC** to your AUTOEXEC.BAT file.

2.4 A simple example of LOCBIF usage

Consider a system of differential equations with two phase variables (x, y) which depends upon four parameters $(\alpha, \beta, \gamma, \delta)$:

$$\begin{aligned} \dot{x} &= x - \frac{xy}{1+\alpha x} - \beta x^2 \\ \dot{y} &= -\gamma y + \frac{xy}{1+\alpha x} - \delta y^2 \end{aligned}$$

These equations describe qualitatively the behavior of an ecological system of predator-prey type (Bazykin, 1985). Here x and y are (dimensionless) prey and predator population densities, while parameters $(\alpha, \beta, \gamma, \delta)$ represent some inter- and cross-population effects.

It is known (see Part 3) that for $\alpha = 0.3, \beta = 0.01, \gamma = 1.0$ and $\delta = 0.5$ the system has an equilibrium point with $x = 82.97.., y = 4.409\ldots$. The problem is to determine a dependence of this equilibrium upon parameter δ and to analyze its stability while δ varies from 0.2 to 0.6.

Suppose, you have invoked **LOCBIF** as described before. You should see the opening screen shown in Figure 1. Press any key. Now you can see an Initial screen with a ODEs Archive Window. The archive is empty (actually it may contain several test examples as well as systems which have been specified earlier, say by other users).

Input equations

First of all you have to input the equations into **LOCBIF**. Type a name for the system, for example **ECOL**, and press Enter. An Equation Window will appear with default right hand sides (RHS). Type in RHS of the equations in the following form:

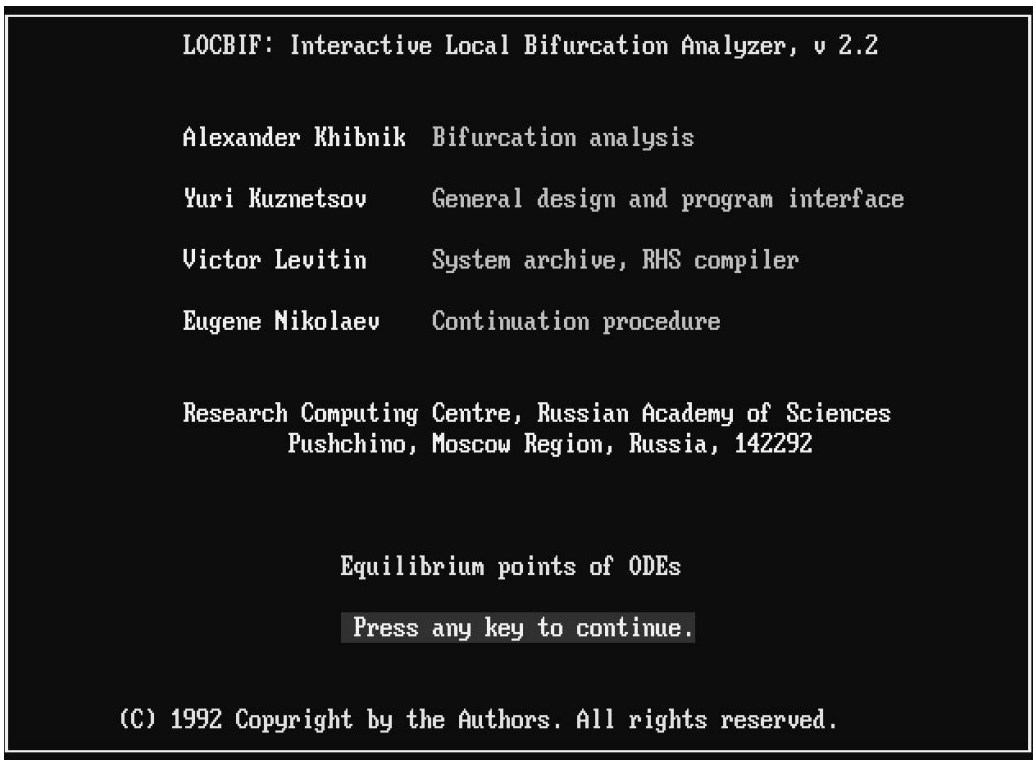

Figure 1: **LOCBIF** opening screen

PHASE X,Y
PAR ALPHA,BETA,GAMMA,DELTA
X′=X-X*Y/(1+ALPHA*X)-BETA*X^2
Y′=-GAMMA*Y+X*Y/(1+ALPHA*X)-DELTA*Y^2

and erase the rest of the default RHS.

You may use conventional keys (Arrows, Del, Backspace and so on). See Figure 2 for the resulting screen.

The first and second lines are variable and parameter declarations, while the last two lines are formulae for the RHS. Here ′ stands for time derivatives.

After you finish typing RHS, press Alt+X (press and hold down Alt key and then press X key). If errors occur you will see an error message. Otherwise you will leave the Equation Window and return to the Initial screen with the ODEs Archive Window. The archive now contains your system **ECOL**

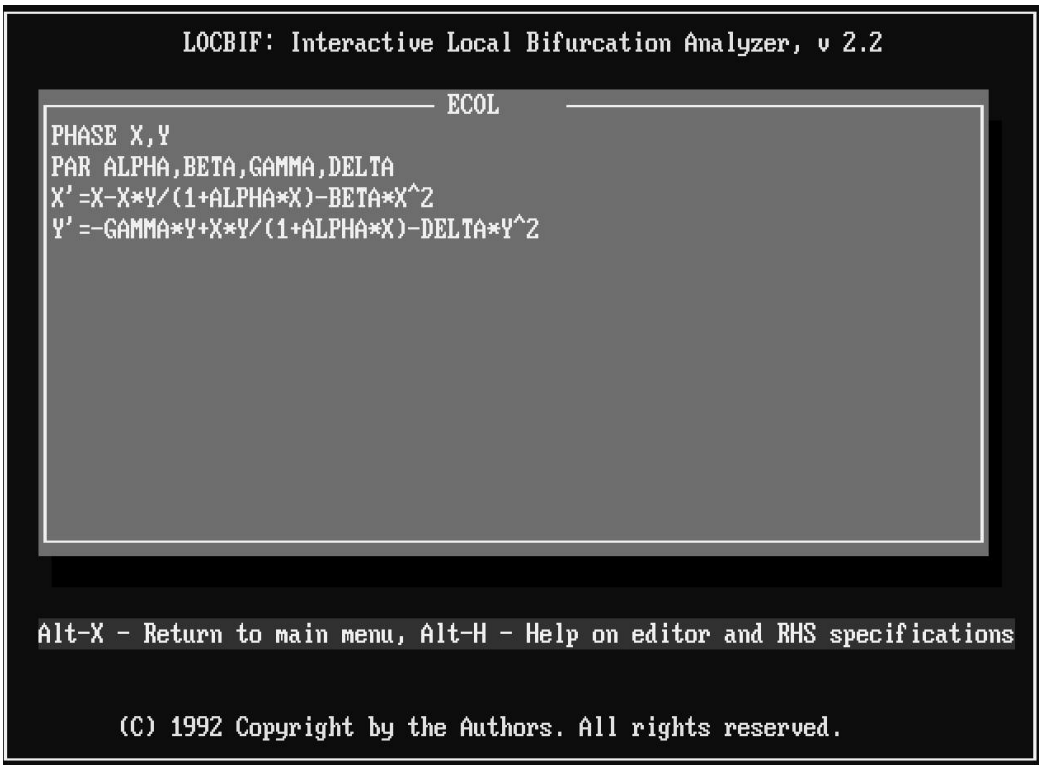

Figure 2: Equation Window of **LOCBIF**

(see Figure 3).

Selection and activation of items

For selection of an item move the cursor with directional keys (Up, Down,Left, Right). Press Enter to complete selection and activate the item.

Setting initial values

Select the system **ECOL** and press Enter. After a short delay you will see a main **LOCBIF** screen with a Main Menu line, Graphics, Value, Curve and Message Windows. All phase variable and parameter values are preset to zero as well as real and imaginary parts of the eigenvalues. The curve type is **Equilibrium** (Figure 4).

Select item **Values** within the Main Menu and press Enter. A one-symbol highlight (cursor) will appear in the Value Window. Now you can move it by directional keys and type initial values of the phase variables and parameters

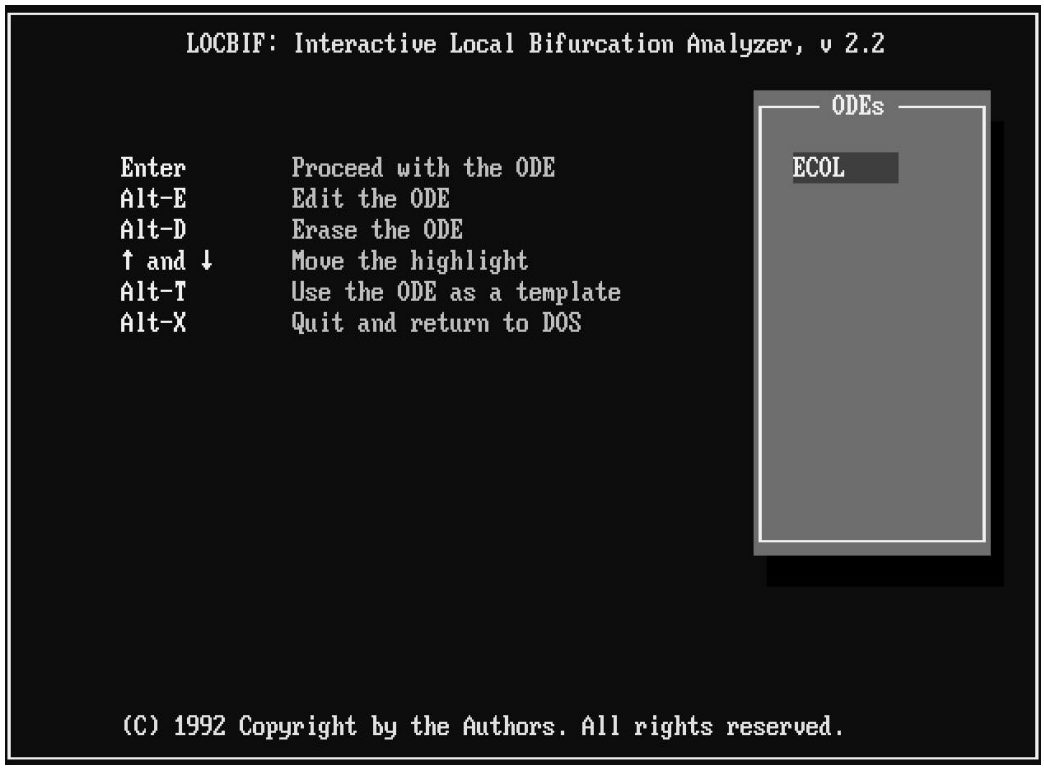

Figure 3: Initial **LOCBIF** screen with the ODEs Archive Window

specified above.

Choose an active parameter

Being inside the Value Window, move the highlight to the line with parameter **DELTA** and press Alt+F keys. This parameter is now highlighted which means that it is chosen as active and we will compute a dependence of the equilibrium upon this parameter. Leave the window by pressing the Esc key. The cursor disappears from the Value Window.

Setting drawing parameters

Select and activate **Options** item within the Main Menu. You will see an Option Window appearing over the Graphics Window (see Figure 5). Activate the item **Axis Parameters**. The Axis Parameter Window is opened over the Graphics Window. Set **DELTA** for the abscissa and **Y** for the ordinate of a plot. This action requires highlighting the abscissa line and

Figure 4: Main **LOCBIF** screen

pressing Enter key several times until the required name for abscissa is chosen. Proceed similarly with ordinate name. Then move the highlight and type axis limits. Set $0.2 <$ **DELTA** < 0.6 and $0 <$ **Y** <10. See Figure 6 for the resulting shape of the Axis Parameter Window. Then press Esc to leave the Axis Parameter Window and press Esc again to leave the Option Window.

Now you are back at the main screen with the **Options** item still highlighted. Activate **Commands** item. You will see a Command Window with highlighted command **Delete Graphics** over the Graphics Window (see Figure 7). Press Enter key to refresh the Graphics Window and to exit the Command Window. Notice, that you have now the proper axis names.

Computation

Now you are ready to start computations. Select **Compute** item and

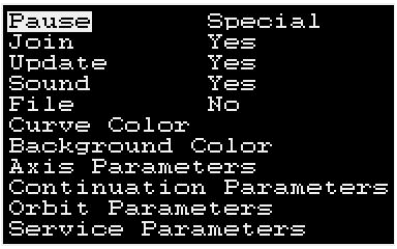

Figure 5: Option Window

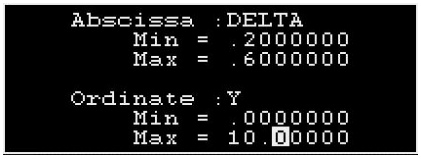

Figure 6: Axis Parameter Window

press Enter. The flashing marker has appeared in the Message Window after your pressing Enter. This indicates the computations are in progress.The first point near the initial point is found and plotted in the Graphics Window. A corresponding message is sent to the Message Window. In the Value Window you can see the current values of the equilibrium coordinates **X** and **Y**, the parameter **DELTA** and the eigenvalues. The real parts of the eigenvalues are negative, so the equilibrium is stable. The program is waiting now for your action.

Press the Enter key once more. The computation will be continued and you'll get the equilibrium curve going to the right from the first point. When it leaves the Graphics Window, you may terminate computation in this direction by pressing the Esc key. (There is no automatic termination assumed, so the computations will be continued until a special point occurs or the buffer reserved for curve storing is filled.)

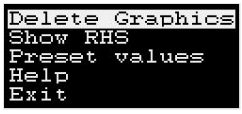

Figure 7: Command Window

To start computation of the equilibrium point curve in the opposite direction, press the Ctrl+Enter keys. The curve will be computed and some messages reporting about special points on the curve will be displayed in the Message Window. After each message the program will wait for your reaction. To continue, press Enter. When curve leaves the Graphics Window terminate the computation by pressing Esc key.

Now you have a screen like that in Figure 8. You can see that the dependence of the equilibrium with respect to parameter **DELTA** has an S-shape. There is an interval $0.263\ldots <$ **DELTA** $< 0.436\ldots$ within which the system has three equilibria: two of them are stable and one is unstable. The boundary points of the interval are critical parameter values corresponding to when two equilibria appear or disappear in the system (we refer to these cases as fold or tangent bifurcations).

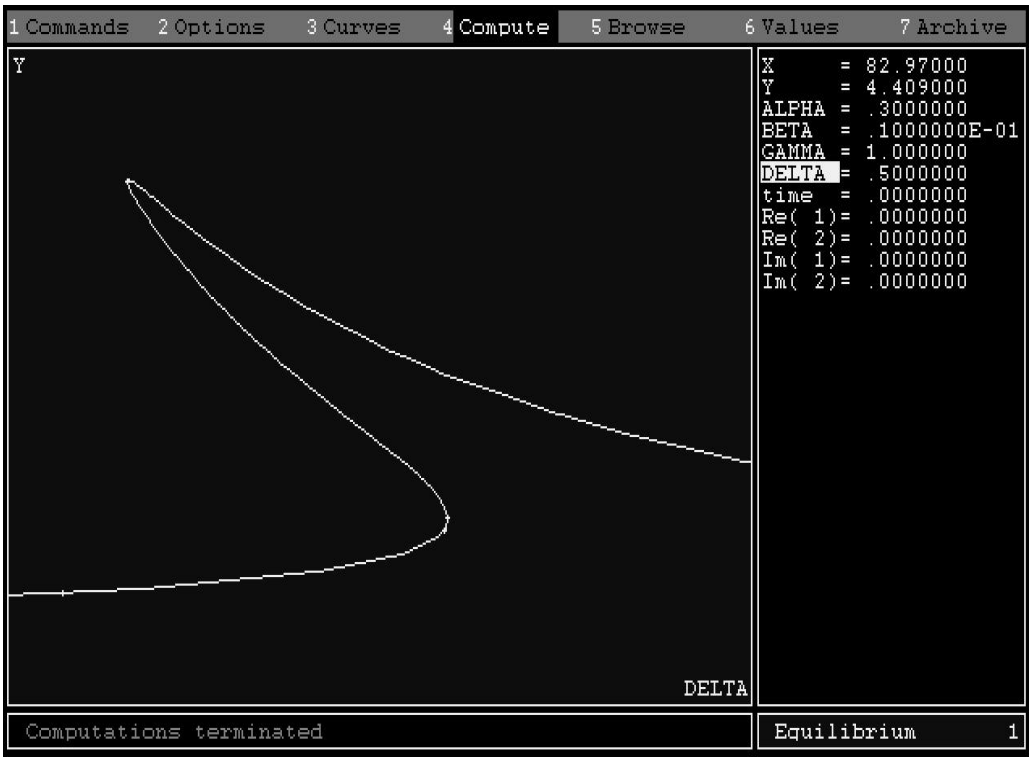

Figure 8: S-shaped equilibrium curve of the ecological model

Browsing the curve

Browse command allows to plot the curve once more and display the related values without recomputation. Let us use browsing to monitor the behaviour of eigenvalues along the computed equilibrium curve and to analyze how changes in stability occur.

First, delete the curve through the Command Window as before or select another color to redraw the curve. In order to chose another color, select **Options**. Then select the option **Curve Color** and press Enter several times. The current choice is shown by the color used for the curve name in the Curve Window. When you have found the color you want, press Esc to leave the Option Window.

Browse the curve by activating the item **Browse** within the Main Menu. Eigenvalues in the Value Window will be updated at every point, as well as the phase coordinates and parameter values. To browse the curve in the opposite direction, press the Ctrl+Enter keys. You may terminate browsing at any point just by pressing the Esc key.

Storing the curve

To store the curve in the archive, select **Archive** within the Main Menu and press Enter. You will see a Curve Archive Window in place of the Value Window. Type a name of the curve, say EQUILIB, and press Enter. The program reports that storing is in process. When it is completed, you will see the name EQUILIB in the list of the stored curves (Figure 9). Press Esc to leave the Archive Window and return to the main screen.

Visiting the option windows

Select and activate **Options** item in the Main Menu. Activate **Continuation Parameters** to open the Continuation Parameter Window (Figure 10), option **Orbit Parameters** to open the Orbit Parameter Window (Figure 11), and **Service Parameters** option to open the Service Parameter Window (Figure 12) respectively. (These windows are described in Section 5). As before, to exit from any window, press the Esc key. Press the Esc key once more to exit the Option Window.

Select and activate **Curves** item. A Curve Select Window will appear (Figure 13) with a list of all curves computed by **LOCBIF**. They are described in the next sections. Use Esc key to leave the window.

Help

Figure 9: Curve Archive Window with EQUILIB curve stored

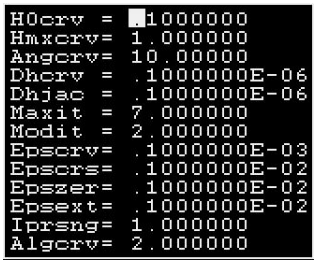

Figure 10: Continuation Parameter Window

Press Alt+H keys from within the Main Menu. A Help Window is opened and some help information is displayed. Press the PgDn key to see the next page of help and Esc to exit.

Quitting

To quit the analysis of the system, select and activate **Commands**, then select **Exit** command and press Enter. Confirm your intention by pressing the Y key. You will see the Initial screen. Press Alt+X to exit **LOCBIF**.

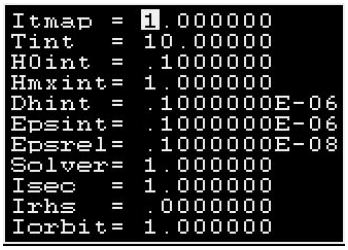

Figure 11: Orbit Parameter Window

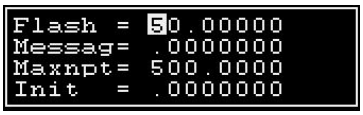

Figure 12: Service Parameter Window

3 The fundamentals of LOCBIF

In this section you'll learn about **LOCBIF** interface and file system. Some terminology used in **LOCBIF** and this manual is introduced.

3.1 Terminology

Mathematical models are often represented as systems of ordinary differential equations (ODEs) which depend upon parameters. In this manual (exept of Appendix C) the term *system of ODEs* or *dynamical system* means a system of nonlinear autonomous differential equations of the form

$$\dot{x} = F(x, p), \tag{3.1}$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^n, p = (p_1, p_2, \ldots, p_m) \in \mathbf{R}^m$ and F is a smooth vector function. Here dot $(\dot{})$ means differentiation with respect to variable t, referred to as time.

Components x_i of vector x are called *phase variables*, while components p_i of vector p are referred to as *parameters*. Spaces $\mathbf{R}^n$ and $\mathbf{R}^m$ are called *phase* and *parameter spaces* respectively.

Values of parameters may be fixed or free during a stage of the system analysis. A free parameter is called an *active parameter*. Phase variables and

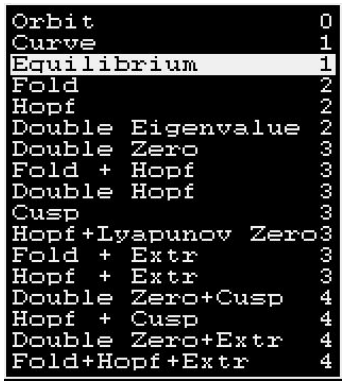

Figure 13: Curve Select Window

active parameters together will be called simply *variables.*

Let $p_{i_1}, p_{i_2}, ..., p_{i_k}$ be the active parameters. The space $\mathbf{R}^s$, with $s = n + k$ coordinates

$$(x_1, x_2, \ldots, x_n; p_{i_1}, p_{i_2}, \ldots, p_{i_k})$$

is called the *active phase-parameter space.* If $k = 0$, this is the same as the phase space.

A *curve* in $\mathbf{R}^s$ with coordinates denoted by $y = (y_1, y_2, ..., y_s)$ is a one dimensional smooth manifold defined either parametrically:

$$y = y(t)$$

or implicitly by $s - 1$ functions G_i:

$$G_1(y) = 0, G_2(y) = 0, ..., G_{s-1}(y) = 0.$$

An *initial point* is a point $y^{(0)} \in \mathbf{R}^s$ from which a curve will be computed point by point. The initial point is to be defined by the user. The initial point may not lie exactly on the curve. The first computed point on the curve is called the *first point.* There are two *directions* in which the curve can be computed starting from the initial point.

An *orbit* or *trajectory* is a curve in the phase space defined by solution of Initial Value Problem for system (3.1) forward or backward in time. There are no active parameters in this case.

An *equilibrium point* or *equilibrium* is a point x in the phase space where the right hand side of (3.1) vanishes:

$$F(x,p) = 0. \tag{3.2}$$

Eigenvalues of the equilibrium are the eigenvalues of the linearization matrix of (3.1) at the equilibrium point.

An *equilibrium curve* is a curve in active phase-parameter space $\mathbf{R}^{n+1}$ with coordinates $(x; p_{i_1})$ defined by $F_j(x; p_{i_1}) = 0, j = 1, 2, \ldots$,n. There is one active parameter in this case, the others are fixed.

A *bifurcation curve* is a curve of equilibria satisfying so called *bifurcation conditions*. It lies in an active phase-parameter space and is defined by (3.2) and $(k-1)$ *bifurcation functions* $G_{n+1}, G_{n+2}, ..., G_{n+k-1}$, where k is the number of active parameters.

There are several *types* of bifurcation curves. Each of them is defined by a different set of bifurcation functions (see Part 2 for details).

Equilibrium and bifurcation curves may have *singular points* at which some additional conditions specific for each curve hold. Singular points may serve as initial points for the computation of the other bifurcation curves. These curves may lie in the same, or in a different, active phase-parameter space.

The computation of orbits is controlled by numerical parameters called *orbit parameters*. The computation of curves given implicitly (so called *continuation)* is controlled by *continuation parameters*.

3.2 Main features of LOCBIF interface

Most of actions available for the user may be invoked independently of each other. This allows the study of different systems in different ways, *i.e.* creating a user- and problem-dependent scenario. The building blocks of the **LOCBIF** user interface arise naturally from the very idea of continuation bifurcation analysis and are as follows:

You can specify a new dynamical system to study or select an existing system from an archive in order to modify it or continue its investigation. To formulate a new dynamical system, you must specify the right hand sides of the system as a program in a Pascal-like language using a build-in editor.

You can set initial values of all phase variables and parameters, select active parameters, and choose a type of curve to be computed.

You can compute and (simultaneously) plot a curve, doing this in a forward or backward direction, stepwise or automatically. The computation can be terminated at any point.

You can browse the last computed curve without recomputation and select any point on it as an initial point for the next computation.

You can store the computed curve into an archive and extract a stored curve from the archive for browsing.

You can change current computational and plotting parameters and save them for the next session.

3.3 LOCBIF windows

A user interacts with **LOCBIF** by means of the keyboard [4] . **LOCBIF** sends all the information to the user through various windows.

A *window* is a rectangular area on the screen. All the windows in **LOCBIF** are of a fixed size and position and may have some attributes which can be changed by the user. Displayed *options* as well as *numerical values* can be directly modified by the user through a relevant window. The user can invoke a window by selecting the corresponding item and pressing the Enter key within a **Main Menu** or within another window. The user can leave the window by pressing the Esc key.

Names of ODEs are listed in an **ODEs Archive Window**. A dynamical system can be specified or modified through an **Equation Window**. These two windows appear over the **Initial screen**. See Figure 2 and Figure 3.

A curve is plotted in a **Graphics Window**. Phase variables and parameters are displayed in a **Value Window**, as well as eigenvalues and values of user-defined functions. The type of a curve to be computed is shown in a **Curve Window** and can be selected through a **Curve Select Window** (Figure 13). All messages are directed into a **Message Window**.

The **Graphics, Value, Curve** and **Message Windows** as well as the **Main Menu** are located on the **main LOCBIF screen** (see Figure 4). Some commands can be performed through a **Command Window** (see Figure 7) while computation and drawing options can be selected through

[4] Appendix C describes how to use the mouse, if present, to operate **LOCBIF**.

an **Option Window** (see Figure 5). Attributes of the **Graphics Window** can be seen and modified through an **Axis Parameter Window** (Figure 6). Continuation parameter values, orbit parameter values and additional service parameter values are displayed and can be modified within **Continuation, Orbit and Service Parameter Windows** respectively (Figures 10, 11, 12). These windows, as well as the **Curve Select Window**, appear over the **Graphics Window**. Stored curves can be seen and selected through a **Curve Archive Window**, which appears instead of the **Value Window** (Figure 9).

There is also a **Help Window** which contains a list of possible user actions.

3.4 LOCBIF file system

For each newly specified ODE system **LOCBIF** will create a separate subdirectory which contains files, both permanent and working, related to this system (all these subdirectories are listed in the **ODEs Archive Window**). In particular, the file RHS.DAT contains a program with RHS specifications which the user created during a **LOCBIF** session using the built-in editor, and file INIT.DAT is for system-dependent parameter settings. Two other permanently resident files (RS.DAT and TS.DAT) are for internal use only.

There are also several optional files with extension LIN which are related to the curve archive. Their names are assigned by the user through the **Archive Window**. The optional file RESULT contains a protocol of curve computations.

Normally the user shouldn't be concerned about these files except to preserve them in the corresponding subdirectory.

Note. The manual creation of a subdirectory in the directory **LBEP** automatically leads to the appearance of a new name in the **LOCBIF** ODEs archives. Careful copying of all the files to this subdirectory from another ODE system subdirectory makes it valid for **LOCBIF**, otherwise a program crash becomes a distinct possibility.

Part II
PROGRAM DESCRIPTION

4 Specification of a dynamical system

In this section you'll learn how to begin the study of a new dynamical system or continue investigation of an existing one. You'll learn how to manage the ODEs Archive and specify RHS of the dynamical system using the build-in editor. A description of a special language for RHS-programming will be given.

4.1 ODEs Archive management

All the dynamical systems studied are stored in the ODEs Archive. When viewing the Initial screen you'll see an *ODEs Archive Window* with a list of all stored dynamical systems. The first line in the window is highlighted. To the left of the window you can see a menu with possible actions. It includes the following actions:

- input a new ODE system;
- select one of the listed ODE systems;
- quit **LOCBIF** and exit to DOS.

To input a new ODEs system, type its name. The name will appear in the first window line. Correct the name, if necessary, using Arrow, Space and Backspace keys. When the name is correctly typed, press Enter. You'll see an *Equation Window* and may start specification of the system RHS.

For selecting one of the listed ODE systems, use Up and Down Arrow keys. The name of the chosen system is highlighted. If a list of ODEs is longer than the ODEs Archive Window you may use these keys to scroll the archive across the window.

To quit **LOCBIF** and exit to DOS, press the Alt+X keys.

When a system name is selected you may do one of the following actions which are listed on the screen:

- proceed with the selected dynamical system;
- edit the RHS of the selected system;

- delete the selected system;
- use the selected system as a template to input a new ODEs system;
- quit **LOCBIF** and exit to DOS.

To proceed with the selected system, just press Enter key. In a seconds, the main **LOCBIF** screen will appear.

To edit the RHS of the selected system, press Alt+E keys. You'll see the Equation Window and you can modify the RHS specifications.

To delete the selected system, press Alt+D keys. **LOCBIF** requests confirmation. Press Y to go ahead and make the desired deletion.

Sometimes a new dynamical system may be similar to an existing one which is stored in the archive. In this case you can use the existing system as a template to create a new system. For this, select the system to be considered as a template (i.e. highlight its name) and then press Alt+T keys. The name will be duplicated to the first window line. You have to change this name and press Enter. Then you'll see the Equation Window and can start the modification of the selected system producing a new one.

All files of a newly specified ODE system will be placed in a subdirectory of the **LOCBIF** residal directory. The subdirectory will have the same name as given to the ODE system.

4.2 Build-in RHS Editor

The system RHS is specified by a program in a Pascal-like language. This program is stored as file RHS.DAT in a subdirectory related to this system. You can enter a new ODE system specification or modify an existing system specification using a build-in editor within the Equation Window.

The following operations are supported by the Editor:

Cursor movement	Up, Down, Left, and Right Arrows
Delete character	Del
Delete character left	Backspace
Delete line	Alt+D or Atl+Y
Split line/Insert line	Enter
Join lines	Del when the cursor is at the end of line
Put cursor at the beginning/end of line	Home/End

Put cursor at the beginning/end of text	Ctrl+Home/Ctrl+End
Return to Initial screen with error checking	Alt+X
Return to Initial screen without error checking	Alt+Q

If the RHS text is longer than the Equation Window, you can use Arrow keys to scroll the text.

If after pressing Alt+X an error found in the RHS specifications, one of the error messages listed in Appendix A will appear below the Equation Window. In this case, you can press any key to continue editing.

4.3 RHS Language description

4.3.1 Structure of ODEs specification

A specification of ODEs is to be organized in accordance with the following structure:

TIME variable declaration	(may be omitted)
PHASE variable declaration	(must be present)
PARameter declaration	(must be present)
FUNction declaration	(may be omitted)
COMMON variable declaration	(may be omitted)
function definition(s)	(may be omitted)
RHS definition(s)	(must be present)
INITial values computation	(may be omitted)

The meaning of the terms *time*, *phase variables* and *parameters* was explained in Section 3.

Any text between exclamation mark "**!**" and the end of a line is considered a comment and ignored.

4.3.2 Variable and function declaration

Any declaration consists of an appropriate keyword (**TIME**, **PHASE**, **PAR**, **FUN** or **COMMON**) followed by a list of names. In the case of the **TIME** declaration the list consists of only one name. Names may have up to six characters and must be separated by commas.

FUNctions are scalar valued functions of phase variables and parameters. They may have some additional formal parameters or arguments. The functions may be called from the other functions or RHS. Note that functions may be also used in **LOCBIF** for additional purposes (e.g. tracing user-defined functions along a curve).

COMMON variables are internal variables which may be used through the number of the array elements enclosed in square brackets [ ] must follow variable name.

Note that the **LOCBIF** functions are not pure functions in the mathematical sense because they can produce *side effects*. For example, one function can assign a value to a **COMMON** variable and another can use this value although it does not call the first function at all.

4.3.3 Function definitions

The function definition has the following form:

func_name(*arg_name, arg_name,...*) = *body*

Here *func_name* is a function name, *arg_names* are names of the function arguments which should not have been declared before. If the function has no arguments the function definition is simply

func_name = *body*

The structure of the *body* will be defined later. You should define as many functions as you have declared. A function cannot be recursively defined.

4.3.4 RHS definitions

The RHS definition has the following form:

$var\_nam' = body$

Symbol " ′ " means time derivative d/dt. The number of RHS definitions in the program should be equal to the number of phase variables.

4.3.5 Simple body structure

In the simplest cases, *body* is an arithmetic expression which is build according to the usual rules. You may use phase variable and parameter names,

constants (*e.g.*,$2, 3.1415, 0.123E-1$, *etc*), signs ($+, -, *, /, **$ as well as ^ for exponentiation) and parentheses ().

You can also use the following standard mathematical functions:

ABS(X)	absolute value of **X**
SQR(X)	square root of **X**
SIN(X)	sine of **X**
COS(X)	cosine of **X**
LOG(X)	natural logarithm of **X**
EXP(X)	exponent of **X**
TAN(X)	tangent of **X**
ATN(X)	inverse tangent of **X** (or arctan **X**)
SGN(X)	if **X** >0 then **SGN(X)**= 1
	if **X** = 0 then **SGN(X)**= 0
	if **X** <0 then **SGN(X)**=-1

User-defined functions may be used in the same way as the standard functions. The number of arguments in a function call must be the same as in the corresponding function declaration.

4.3.6 Complex bodies

An arithmetic expression in the body may be preceded by a *program* enclosed in braces . It may contain local variable declarations and a sequence of statements. The arithmetic expression is evaluated after execution of the program statements.

You can use **local VARiables** in the program which have to be declared at the beginning of that program. Local variable declarations consist of the keyword **VAR** followed by a list of local variable names:

VAR *locvar_name, locvar_name,...*

Local variables may be one-dimensional arrays. In this case a number of the array elements enclosed in brackets [ ] must follow a local variable name.

Expressions following braces may use local variables declared in the preceding program.

There are six kinds of **statements**: **assignment**, **IF**, **WHILE**, **FOR**, **terminator** and **compound**.

An **assignment** statement assigns the value of an arithmetic expression to a variable. In the expression you may use **PHASE**, **PAR**, **COMMON**

and **VAR** variables as well as all declared functions. An array variable name must be followed by [*index*].

An **IF** statement makes a decision regarding program flow. It has the form:

IF *expression relation expression statement* **ELSE** *statement*

ELSE *statement* is optional, *relation* must be one of the symbols: $=, <>$ (not equal to), $<=, >=, <$ and $>$. The **IF** first computes both *expressions*, then compares the resulting values using the *relation*. If the relationship is *true* the first statement is executed, otherwise the second statement, if present, is executed. In both cases execution continues with the following statements.

The **WHILE** statement executes a statement in a loop as long as given relationship is *true*. It has the form:

WHILE *expression relation expression statement*

If the *expression relation expression*is *true* then the *statement* is executed. If it is not true, execution continues with the following statements.

The **FOR** statement executes a statement in the loop a given number of times. It has the following structure:

FOR *var* = *expression_from, expression_to statement*

Variable *var* takes values from *expression_from* to *expression_to* with step 1 and may be used within the statement.

The **terminator** statement is one of the following statements:

BREAK, **ABORT**.

An execution of either of these statements leads to generation of an error return code of the RHS computation. The **ABORT** indicates additionally that all the computations should be terminated.

The **Compound** statement is a sequence of statements surrounded by braces:

{ *statement statement* ...}

which is considered as one statement. Compound statement is useful in **FOR**, **WHILE** and **IF** statements.

4.3.7 Initial values computation

You can assign desired initial values to phase variables and parameters, as well as to **COMMON** variables, by means of the following structure:

INIT = *assign code*

Here *assign* is a compound statement which assigns some values to **PHASE**, **PAR** and **COMMON** variables, and *code* is a control parameter. You should set *code* equal to zero if the variable assignment is successful and nonzero otherwise. Execution of the **INIT** assignment depends on user initiative and is invoked normally by pressing a special key before starting the main computations (see the next Section).

4.3.8 Error messages

When you have finished the RHS specification you press the Alt+X keys. The program starts to check the syntax errors and produce internal files TS.DAT and IR.DAT. If no errors are found, the Equation Window is closed and you see the Initial screen again.

If there is an error in the RHS specifications the relevant message appears below the Equation Window and the cursor is placed at the error position. You may correct the error or quit the editor without error checking by pressing Alt+Q keys.

The list of all possible error messages is given in Appendix B.

5 Investigation of a system

In this section you'll learn how to use **LOCBIF** for an ODE system study. Among other things you'll learn how to set options and initial data, choose active parameters, select the curve type, start and terminate computation of a curve, browse the computed curve and manage a curve archive.

5.1 Main Menu

Suppose you have selected an ODE system for investigation. You should see the main **LOCBIF** screen with four windows and a *Main Menu* line on the top (Figure 4). The line lists seven *items*: **Commands**, **Options**, **Curves**, **Compute**, **Browse**, **Values** and **Archive** assigned by their numbers. The first one is highlighted. You can select an item by using Left and Right Arrows, as well as Home and End keys. When an item is selected, it can be *activated* by pressing Enter. The activation of **Compute** and **Browse** items results in an *immediate* action, while the activation of the other items allows you to work with relevant windows.

You can select and activate an item by pressing one of the function keys: F1 - F7. This convention is valid also if you are inside a window activated from the Main Menu.

5.2 Command Window

If you activate item **Commands**, the *Command Window* appears over the Graphics Window (see Figure 7). The Command Window contains a list of five *commands*: **Delete Graphics**, **Show RHS**, **Preset Values, Help** and **Exit**. The first one is marked by a highlight. You can select a command by pressing the Down and Up Arrows and Home and End keys. To perform a selected command you have to press Enter key.

5.2.1 Clearing Graphics Window

Select command **Delete Graphics** within the Command Window and press Enter. The Command Window will be closed and all the plotted curves deleted from the Graphics Window.

5.2.2 Display equations

To display the RHS of a system under investigation, select command **Show RHS** within the Command Window and press Enter. The Command Window will disappear and you will see the system specification over the Graphics Window. Notice, that some part of the specification text may be invisible due to the window frame.

5.2.3 Preset values

You may assign values to variables using an **INIT** statement defined in the RHS program. Select command **Preset Values** within the Command Window and Press Enter. The window will disappear, the values of the phase variables and parameters will be replaced by those computed in **INIT** statement, and the message

"Values initialized"

will appear. If the **INIT** statement completes with a nonzero *return code* the values will not be changed and the message

"Values not initialized"

will be displayed. No messages will appear if there is no INIT statement in the RHS specifications.

You can preset values automatically before each curve computation by nonzero setting of the **Init** service parameter (see Section 5.3.11).

5.2.4 Help

Select the **Help** command within the Command Window and press Enter. The Command Window will be closed and a *Help Window* will appear over the Graphics Window (see Section 8).

5.2.5 Return to the Initial screen

To return to the Initial screen, select the **Exit** command within the Command Window and press Enter. The Initial screen with the ODEs Archive Window will appear.

5.3 Option Window

If you activate item **Options**, the *Option Window* (Figure 5) appears over the Graphics Window. The Option Window contains a list of several *options*: **Pause**, **Join**, **Update**, **Sound**, **File**, **Curve Color**, **Background Color**, **Axis Parameters**, **Continuation Parameters**, **Orbit Parameters** and **Service Parameters**. You can select an option using the directional keys. To modify the selected option you have to press the Enter key. The first seven options are toggles: they can be modified directly, by pressing Enter one or several times, and their new settings are immediately displayed. The activation of the other options results in the appearance of a corresponding window. All values within such a window can be modified directly. You can use Space, Del and Backspace keys for corrections. To leave the window, press the Esc key. This will return you to the Option Window. To leave the Option window, press Enter once more.

All the options chosen will be current during the computation and browsing until you change them.

5.3.1 Setting the pause mode

A process of curve computation or browsing may be interrupted by pauses which can be initiated either by the program or by the user. *Pause* means stoping after calculating a new point. The **Pause** mode determines when the pauses should be made. There are three pause modes provided:

Pointwise	- with a pause after each calculating point
Special	- with pauses at special points only
No	- with no pauses

5.3.2 Setting join mode

During continuation or browsing each computed point is plotted in the Graphics Window. A "dotted" or "solid" mode is used according to the **Join** mode setting. The **Yes** option means that two sequential points will be joint together by a straight line, while the **No** option will result in plotting a sequence of disjointed points.

5.3.3 Setting update mode

Numerical entries in the Value Window may be updated during computation or browsing at each computed point or at special points only. **Update** mode settings **Yes** or **No** correspond to these two alternatives. A frame of the Value Window indicates that variables will be updated at each point. The **No** option may be particularly useful if you are not interested in exact numbers or if the graphics hardware of your computer is slow.

5.3.4 Setting sound mode

Display of a point during the computation or browse may be accomplished by a sound. **Sound** option **Yes/No** corresponds to sound on/off.

5.3.5 Setting hardcopy mode

While computing or browsing a curve, you can output it to a special file, **RESULT**, permanently stored in a subdirectory corresponding to the system under investigation. The output may be *complete* or *reduced*. The complete output contains: the curve type, values of computational parameters, initial values of phase variables and system parameters, and coordinates of all obtained points on the curve with corresponding messages at special points. The reduced output consists only of pairs of variables values currently plotted along abscissa and ordinate axis. For storing in the **RESULT** file, an append mode is used.

A **File** option determines if there will be output into the **RESULT** file or not and determines its type (**No/Complete/Reduced**).

5.3.6 Setting curve color

To change the foreground color in which a curve type is displayed in the Curve Window , select the **Curve Color** option and press Enter key. The curve type in the Curve Window will be displayed in the next color supported by your graphics hardware. You may need to press the same key repeatedly to find the desired color.

5.3.7 Setting background color

To change the background color of the Graphics Window, select the option **Background Color** and press the Enter key. The next color supported by your graphics hardware will be displayed in the Curve Window. You may need to press the same key repeatedly in order to find a desired background color. Upon exiting from the Option Window, the color of the background of the Graphics Window will be changed to the selected one and all plotted curves will disappear.

5.3.8 Setting axis variables and their limits

Select the **Axis Parameters** option and press Enter. You will see an *Axis Parameter Window* which displays variable names plotted along the abscissa and the ordinate and limits of their visibility (Figure 6). The abscissa name is highlighted. You can move the cursor by directional keys.

To select a desired abscissa name, press Enter if the abscissa line is selected. All variable, function and eigenvalue names will be consequently displayed by repeatedly pressing the key. You may press the Ctrl+Enter keys to select the names in reverse order. The same procedure should be applied to select a variable to be plotted along the ordinate.

To set a limit value, move the cursor to the corresponding line and type a relevant value.

If you have computed a curve or loaded it from the archive, you may undertake an automatic setting of the visibility limits by pressing the Alt+L keys. The curve will be analyzed to determine maximal and minimal values of the axis variables. The visibility limits will then be changed and the message

"Limits adjusted"

will appear in the Message Window. If minimum and maximum values along an axis happen to be equal, message

"Limits not adjusted: zero difference"

will be displayed and no operations on the limits will be performed.

To leave the Axis Parameter Window, press the Esc key. The window disappears and you will see the main **LOCBIF** screen with the Option Window again.

Note. If you don't clear the Graphics Window after changing the drawing parameters described above, the new settings of these parameters will remain current, even if you see "old" axis names on the screen. This feature might be helpful if one needs to overlap several pictures which are either drawn using different scales or related to projections onto different planes.

5.3.9 Setting continuation parameters

Parameters of the continuation procedure may be changed through a *Continuation Parameter Window* which is invoked by pressing Enter if the option **Continuation Parameters** is selected within the Option Window (Figure 10).

In the invoked window you can see values of the computational parameters which control the continuation procedure. You can modify them if necessary. For the meaning and possible range of the parameters, see Section 7.

5.3.10 Setting orbit parameters

In an *Orbit Parameter Window* (see Figure 11), which is invoked by selecting the **Orbit Parameters** option and pressing Enter, you can see values of computational parameters controlling the orbit computation. You can modify them if necessary. For the meaning and possible range of the parameters, see Section 7.

5.3.11 Setting service parameters

In a *Service Parameter Window* (Figure 12), which is invoked by selecting **Service Parameters** option and pressing Enter, you can see values of parameters controlling the program output. You can modify them if necessary. They have the following meaning:

Flash	cursor flash index
Messag	intermediate message index
	0 - message off
	1 - message on
Maxnpt	maximal number of points in each direction
Init	automatic initialization index

0 - preset off

5.3.12 Default option settings

The following default values of the options (except numerical ones which are discussed further) are satisfactory for most needs.

Pause	**Special**
Update	**Yes**
Join	**Yes**
Sound	**Yes**
File	**No**
Flash	50
Messag	0
Maxnpt	500
Init	0

5.4 Curve type

You can choose a curve for continuation through a *Curve Select Window* (see Figure 13) which is invoked by selecting and activating item **Curves** of the Main Menu. Names of all the curves which may be continued by **LOCBIF** will be displayed in the Curve Window together with a proper number of active parameters. You can select the curve type using the directional keys and then pressing Enter. The Curve Select Window will disappear and the selected curve type will be displayed in the Curve Window. You can leave the window without changing the curve type by pressing Esc.

For the meaning of curve types displayed in this window, see Section 6.

5.5 Value Window

The right most window (see Figure 4), called the *Value Window*, displays all phase variable and parameter names, together with their numerical values. Names and values of all functions used in the RHS, as well as real and

imaginary parts of eigenvalues are displayed. The names of variables and functions are the same as in the RHS program, with lowercases changed to uppercases. **Re**(i) and **Im**(i) stand for the real and the imaginary part of i-th eigenvalue respectively. When starting investigation, numerical values are restored from the last session. If you have just specified a new dynamical system without using template all numerical values will be preset to zero.

Some of the parameter names may be highlighted. This means that corresponding parameters are considered as "free" (or *active*) in curve continuations while the other are fixed at their initial values.

Before you will start curve computation you should set initial values of all **PHASE** and **PAR** variables and define which of the parameters will be active during curve continuation. To access the Value Window you have to select and activate item **Values** in the Command Line. Then you will see a one-symbol *cursor* in the Value Window.

When the variable list is longer then the window, you see symbols Up/Dn in the Message Window and can use the Up and Down Arrows for scrolling.

5.5.1 Setting initial values by hands

To set or change a variable value, move the cursor by directional keys and type in a number. Use the Space key to erase a character. Use also the Backspace and Del keys for corrections.

5.5.2 Taking current values as initial

While in computational or browsing mode (see Sections 5.6 and 5.8), you may take variable values of a current point as initial values for further computations. To do this, press the Ins key when a fitting point is found. At that moment a program should be in a waiting mode (pause). Then the computation/browsing will be terminated and coordinates of the last displayed point will be assigned to the initial one.

A natural way of applying such a basic option is *e.g.* to take the last point of an orbit converging to an equilibrium point as initial one for tracing the next branch of equilibrium. We will discuss some similar and even more sophisticated examples further.

5.5.3 Parameter activation

To make a parameter active ("free"), move the cursor in the Value Window to a line with this parameter name and press Alt+F. The parameter name becomes highlighted. To make the parameter inactive, press the Alt+F keys once more.

Remember that the number of active parameters should be equal to the number pointed out with the curve type in the Curve Window. If that is not the case, message

"Incorrect number of active parameters"

will appear when you attempt to start computations.

5.5.4 Variable ordering

You can exchange any two lines in the Value Window for better recognition. Press Ctrl+PgUp/PgDn keys to move a line with the cursor up/down.

5.6 Computation

5.6.1 Basic conventions

A curve is presented to the user in a form of an ordered number of points computed sequentially. Each point is processed immediately after its computation, which includes graphics representation, displaying of a relevant message, updating values, storing in a buffer, pause and termination processing, *etc.*

A newly obtained point is plotted in the Graphic Window. A "dotted" or "solid" mode is used in accordance with the **Join** option (see Section 5.3.2). Special points on a curve are marked.

A standard or special (case-dependent) message is displayed in the Message Window, depending on the type of current point (regular or special). When getting a new point, all values in the Value Window are either updated or not to their current values, depending on the option **Update** (see Section 5.3.3). Regardless to this option, if a special point is encountered, this always leads to values being updated.

Displaying a new point is accompanied by a sound. Its frequency is different for regular and special points (so you can distinguish between them

even when speaking with colleagues during curve computation). If this sound makes you nervous, just turn it off by setting **No** for the **Sound** option (see Section 5.3.4).

Curve computation may be interrupted by a pause, which can be initiated either by the program or by the user (see Section 5.3.1). Any pause leads to updating all entries in the Value Window to their current values. Some of the currently used optional modes can be changed at a pause by the user.

When computations are in process, a special marker in the Message Window is flashing; it reflects the RHS evaluations (normally not every one of them). Typically, if there is no flashing, this indicates that the program is in pause and waiting for user's action.

A pause enables the user to terminate computation if necessary. Termination is normally followed by the resetting of all values in the Value Window to their initial values. This can be changed to preserve the current point as a new initial one. In this case, the former initial point will be replaced by the current point.

Computations can be terminated asynchronously, just by pushing a special key at an arbitrary moment. Then, the computations will be stopped and all the values reset to initial ones.

Tracing a curve in either direction, starting from the first point, is considered an independent computation. The user must specify the desired direction (forward or backward) when starting a computation. The direction chosen is referred to by a sign at an ordering number of the current point (**Npt**) ("+" for forward, and "-" for backward direction). To get a whole branch (in both directions), one has to start forward and backward computation sequentially (in either order) from the same initial point. Note that the notion of direction has here a conventional and not a geometrical meaning.

During the computation, the points of a curve are stored in a buffer file. A stored curve remains available after termination of computations, for example for redisplaying or saving in a permanent file, RESULT (see Section 5.3.5). The buffer contains only the last computed curve (for each direction). When starting new computations the old curve is lost. The size of the buffer is preset by the user (within the free space available or maximal number of points to be computed). Reaching either buffer or free space limits leads to termination of the computation automatically.

Computing a curve repeatedly, using the same initial values and the same computational parameters, should provide the same results. In fact, this can

be changed by the user when implementing more sophisticated RHS program but any consequences of this are of his/her responsibility.

5.6.2 Starting computation

To start forward computation of a curve (using the selected curve type and all the selected options), choose **Compute** within the Main Menu and press Enter. To start backward computation, use the Ctrl+Enter keys. Computation and plotting of the curve will be started. If the service parameter **Init** is nonzero, an initialization procedure will be first executed and completed with a relevant message (see Section 5.5.2). Next, a flashing cursor will appear in the Message Window which means curve computation has started.

Recall that you can start computation by pressing the F4 functional key at the Main Menu or inside the Command, Option, Value or Curve Archive Windows.

After the first point on the curve is found, a message

"The first point"

is displayed, all values are updated, the point is plotted in the Graphics Window, and a pause takes place (i.e. the program waits for your reaction), unless the **No**-pause mode is currently being used. It should be noticed that searching for the first point may take a considerable time and may even fail if an initial guess isn't good enough. If the program fails, you will get one of the error messages listed in Section 7.

Having got the first point on a curve, a regular procedure for computing the next points in the chosen direction can be started. To resume from a pause , you should press the Enter key.

For any regular point on a curve, you will see a standard message

"Npt = *num*",

where *num* is a point number preceded by a sign showing direction. If a special point is be encountered, a relevant special message will be displayed; all the such messages are presented in Section 7.

5.6.3 Pause processing

Causing a pause. The program pauses at every point or only at special points depending of the pause mode you have chosen. Regardless of the mode, you

can make it pause at any point on a curve just by pressing the Space key. In this case the message

"Npt = *num*",

will appear when the next point will be computed. You can initiate such a pause at an arbitrary moment.

You will see that the program is in a pause when the flashing square from the Message Window disappears. This may be easier to recognize than watching for message updates.

Changing options. You can change some of the options through a *Pause Option Window* during a pause. To invoke the Pause Option Window, press the F2 functional key. The window appears over the Graphics Window (see Figure 14). You can set the displayed options in the same way as through the Option Window (see Section 5.3). Use Esc to leave the window. The Pause Option Window can be invoked directly by pressing the F2 functional key while computing.

Figure 14: Pause Option Window

Continuing computations. To continue computation in the selected direction after a pause, press again Enter or Ctrl+Enter, just as you did when starting the computations. You can also use F4 (Ctrl+F4) for continuation. These later keys are also valid inside the Pause Option Window.

Terminate computation. To terminate computation, press the Esc key. The initial point will be then restored at the Value Window. The message

"Computations terminated"

will be displayed.

Terminate computation with saving current point as the new initial point. You can take all variable values, corresponding to a point at which you have paused, as initial values for the next computations. To do this, press the

Ins key. This action terminates computation without restoring the previous initial point. In other words, the currently visible point will be used further as an initial point. The message displayed after termination is the same as before.

Error messages and forced termination. If the program cannot continue computations (*e.g.,* it detects a closed curve or a current step-size becomes extremely small), then it will send a message (see section 7). In this case, you should press the Esc or Ins key to terminate computation in one of the possible ways described above. There may be numerical exceptions during the RHS computation. Then you will get a warning message (see Appendix B), and the program will wait for your reaction. You may press Esc to terminate computations, or any other key to continue. In principle, the latter action may cause erroneous results.

5.6.4 Urgent (asynchronyous) termination

You can terminate computation by pressing the Esc key at any moment. The computation will be aborted, and the program will react in the same way as when it is terminated by the Esc key at a pause.

5.6.5 Curve storage

All points computed on a current curve forward (or backward) are stored in a temporal working file, called *buffer* (see Section 1). No more than **Maxnpt** (defined in the Service Parameter Window) points are stored in each direction. If this number is exceeded, the message

"Last point in buffer"

will appear. You can press the Ins key to select the last point as new initial point, or just terminate the current curve computation by pressing the Esc key.

The message

"Store writing error"

indicates an error in writing into the buffer file and usually means there is no free space on the corresponding device.

5.7 Browsing

Recall that the last computed curve is stored in the buffer (see Section 5.6.5). If one loads a curve from the curve archive (see the next section), it is also stored in the buffer, replacing a previous one. While having a curve stored in the buffer, you can browse it as many times as you want. By *browsing* the curve, we mean redisplaying it graphically in the Graphics Window and numerically in the Value Window. All the messages related to special points remains the same as for curve computation, but warning and error messages will be omitted. The message

"Last point in the direction"

is specific for browsing; it indicates that last point in the buffer has been reached.

We should emphasize that almost all the control used in computation (see 5.6) is also applicable for browsing. It is true for the continuation and termination of browsing and the option modification. Pressing the Space key enables the user to make a pause. The Esc and Ins keys are used to interrupt browsing with or without restoring an initial point. Note that the last option may be of considerable help since it provides the possibility of starting tracing some other curve from the currently browsed point.

You can change some of the options (axis, colors, etc.) before starting browsing (as well as at a pause during the browsing).

To start browsing forward, select the item **Browse** within the Main Menu and press Enter. To start browsing backward use the Ctrl+Enter keys.

The message

"Store reading error"

indicates an error in reading the buffer file. Press any key to terminate browsing. Note that automatic clearing of the buffer follows such an error, so the last stored curve is no longer available.

5.8 Curve Archive Window

The computed curve (temporarily stored in the buffer file) may be stored permanently in the archive. The curve is stored in the directory with the system name as a file *curname*.LIN, where *curname* is a name given by the

user. The stored curve can be loaded from the archives and then be used for browsing.

To work with the curve archive, select item **Archive** within the Command Line and press Enter. A *Curve Archive Window* with the list of all stored curves will appear over the Value Window (see Figure 9). The first line in the window is highlighted. If there are more stored curves than lines in the window, symbols Up/Dn appear. You can scroll the curve name list using the Up and Down Arrows.

To store a computed curve into the archive, type its name. The name will appear in the first window line. When the name is correctly typed, press Enter. The message

"Storing..."

will be send to the Message Window and after a short delay a new curve will be placed into the archive and listed. If you have typed an existing curve name, the message

"Name duplication: Press Enter to overwrite"

will be send to the Message Window. To overwrite the existing curve, press Enter, otherwise you should modify the name.

The message

"Storing error"

means an error in the writing procedure and usually corresponds to the absence of free space on the drive.

To load a curve from the archive, select the desired curve name and press Enter. The message

"Loading..."

will be displayed while the curve is loaded. After loading done, Curve Archive Window will be automatically closed and a Value Window will be updated. The new curve type will be also displayed in the Curve Window.

The message

"Loading error"

means an error in the reading procedure. It may mean the corresponding *curname*.LIN file is damaged. Such an error also occurs if a number of

currently used phase variables, parameters, and functions does not match with that for loading curve.

To delete a curve from the archive, select the name of the curve to be deleted, press the Alt+D keys, and confirm your intention by pressing the Y key.

To exit from the Curve Archive Window, use the Esc key.

5.9 Help system

You may ask for help inside the Equation Window or at the Main Menu, by pressing Alt+H keys. You will see a *Help Window* with the relevant information.

If you press Alt+H during a system specification, the Equation Window will be replaced by a Help Window. Information on the RHS Editor and RHS specification language is now available. You may use Home, End and Up/Down Arrow keys to scroll the information text within the window. Press Esc to leave the window.

If you press Alt+H at the Main Menu, another Help Window will appear over the screen. The window will contain a two-page list of possible user actions together with corresponding keys. To get the next/previous page, press the PgDn/PgUp keys. You can leave the Help Window by pressing Esc. This help window can also be opened by the command **Help** from the Command Window.

There is also another part of the help system which consists of special context-dependent *hints* displayed in the Message Window when you are in pause outside the Main Menu (for example, inside the Option Window or pausing during the computations). These hints provide useful information on possible user actions or about the meaning of optional parameters.

5.10 Hidden keys

You can perform some operations at the Main Menu or inside the Command or Option Window by directly pressing of special keys. This possibility speeds up your work with **LOCBIF**, if you become sufficiently familiar with it.

If you are inside the Command Window, you can use the following keys to perform the commands:

Alt+D	**Delete graphics**
Alt+R	**Show RHS**
Alt+I	**Preset values**
Alt+H	**Help**
Alt+X	**Exit**

If you are inside the Option Window, you can use the following keys to invoke corresponding windows:

Alt+G	**Axis Parameter Window**
Alt+C	**Continuation Parameter Window**
Alt+O	**Orbit Parameter Window**
Alt+S	**Service Parameter Window**

as well as Alt+U keys to set **Update** mode.

You can use all the above listed keys at the Main Menu line as well. After performing the corresponding operation you will be back to this line.

6 Curve descriptions

In this section you'll learn which curves can be computed by **LOCBIF**, how they are defined in terms of bifurcation theory and by means of determining systems, and how points of higher singularity may be found on these curves.

6.1 List of curves

A list of curves supported by **LOCBIF** consists of the following items [5] :

		General curves:
Equilibrium	**1**	Generic equilibrium point curve
Fold	**2**	Fold bifurcation curve
Hopf	**2**	Hopf bifurcation curve
Double Eigenvalue	**2**	Double eigenvalue curve
Double Zero	**3**	Bogdanov-Takens bifurcation curve
Fold + Hopf	**3**	Gavrilov-Guckenheimer bifurcation curve
Cusp	**3**	Cusp bifurcation curve
Hopf + Lyapunov Zero	**3**	Degenerate Hopf bifurcation curve
Double Zero + Cusp	**4**	Degenerate Bogdanov-Takens bifurcation curve (triple point case)
Hopf + Cusp	**4**	Degenerate Gavrilov-Guckenheimer bifurcation curve (triple point case)
		Curves involving nontransversality:
Fold + Extr	**3**	Nontransversal fold bifurcation
Hopf + Extr	**3**	Nontransversal Hopf bifurcation
Double Zero + Extr	**4**	Nontransversal Bogdanov-Takens bifurcation
Fold + Hopf + Extr	**4**	Nontransversal Gavrilov-Guckenheimer bifurcation
		Auxiliary curves:
Orbit	**0**	Solution to ODE system
Curve	**1**	"Simple" continuation

[5] Continuation of the curve **Double Hopf** is not implemented.

The figure next to a curve name denotes the number of active parameters required for computation of the curve.

6.2 Basic concepts

The term *curve* is used here as before for any one-dimensional manifold given explicitly or implicitly. With respect to their domains of definition, the curves in the list above are of two kinds:

1) those lying in an extended phase space including also a time variable (**Orbit** type),

2) those lying in a phase-parameter space, *i.e.* in a product of phase and parameter spaces (all other curve types).

A curve of type **Orbit** presents a solution to the Initial Value Problem for the ODE system

$$\dot{x} = F(x, p) \tag{6.1}$$

with n-dimensional phase vector x and m-dimensional parameter vector p. For such a problem all the parameter values are assumed to be fixed. The computation of the curve means numerical integration of the system either in a forward or a backward time direction (see the next section for details).

We discuss in this section the rest of the list of curves. All of these curves consist of equilibrium points, generic or degenerate. The *equilibrium manifold* M is given by the "algebraic" system

$$F(x, p) = 0 \tag{6.2}$$

This manifold is located in a phase-parameter space and has a dimension m. We assume that all the solutions of (6.2) except those lying on a submanifold of codimension one correspond to *hyperbolic* (or *generic*) equilibrium points. An equilibrium point x_0 is hyperbolic if its Jacobian matrix $A = F_x(x_0)$ has no eigenvalues with zero real part, *i.e.* $Re\lambda_i \neq 0, i = 1, 2, \ldots$,n. A hyperbolic equilibrium point is *stable* if all its eigenvalues λ_i have negative real parts: $Re\lambda_i < 0, i = 1, 2, \ldots, n$, and *unstable* if there is an eigenvalue with positive real part.

A *critical case* arises if a hyperbolicity condition $Re\lambda_i \neq 0$ is violated for one or more eigenvalues; the correspondent equilibrium point is called *nonhyperbolic* or *nongeneric* (*degenerate*). The nonhyperbolic equilibria fill

in a submanifold of codimension one mentioned above. The points of this submanifold in turn can be divided into two classes: "generic" and "critical" where the last corresponds to higher singularities. They fill in a submanifold of codimension two on M. Such a stratification of M can be continued to a certain extent giving rise to a *bifurcation structure* of M.

The important assumption we make about the system (6.1) is that this system is a *generic m-parameter family of vector fields* in $\mathbf{R}^n$. Here *generic* means that the family doesn't belong to any subset in a space H of all vector fields given by equality type conditions (*i.e.* no symmetry or other similar specific features are presumed). More specifically, we assume that for each singularity of vector fields, the corresponding submanifolds in H and the family intersect transversally (in particular they may have no common part). Several consequences follow from this assumption: 1) all equilibrium points of (6.1) are generically (*i.e.* for almost all parameter values) hyperbolic, 2) only (generic) codimension k singularities with $k \leq m$ may occur in the family, 3) a codimension of a singularity in the parameter space of the family which is measured as a codimension of the corresponding submanifold, is equal to that for a space of all vector fields, and 4) for any singularity of codimension $k \leq m$ occurring in the family, the parameters of the family allow to get its versal k-parameters unfolding. Basically, we can confine genericity assumptions in a neighbourhood of M which is enough for our purposes in this text.

Although we focus our attention in **LOCBIF** upon generic systems and their bifurcations, using the software for problems with symmetry is possible but of course there are limitations. We make special remarks about this when discussing the continuation of different singularities.

Roughly speaking, continuation methods are used to investigate the bifurcation structure of the manifold M. Given a point on M with a certain singularity type (*e.g.* with one zero eigenvalue), the idea is to follow a path of equilibrium points having the same singularity type. This results in a curve which lies in a submanifold corresponding to the given singularity.

Curves are the central object of our attention. Each curve may be characterized by the type of singularity it presents which we refer to as a curve type.

For curve specification we use two parallel languages. We first describe a singularity using terminology from bifurcation theory (eigenvalues, eigenvectors, normal forms, *etc.*). Then we specify a related curve using the concept of *determining system* which is used by the continuation method.

Using continuation for bifurcation analysis means not only tracing a curve itself but also finding special points on it. This primarily addresses points of higher singularity which one can encounter on the curve. They appear naturally as intersection points of the curve and a submanifold corresponding to higher singularity. To detect and locate such points, *bifurcation* or *test* functions are used which change sign when the higher singularity occurs. It may also happen that a higher singularity manifests itself in a more geometric fashion, either as a *self-crossing point* or an *extremum* of a system parameter along the curve. By a self-crossing point (sometimes called *bifurcation point*) we mean a point at which locally two (or more) different branches intersect each other. Notice that all these local branches may lie in the same global branch. To search for higher singularities and other isolated points of special interest along the curve, the program analyzes zero points of specified bifurcation functions, selfcrossing points and extrema with respect to system parameters; all such points are referred to as *special points*.

Searching for special points on a curve creates a basis for what we call a *continuation strategy*. This arises from techniques saying 1) how one can start to compute a curve (*i.e.* what is the "right" choice of initial point), and 2) how one can manage to switch between different curves in the course of bifurcation analysis. The continuation strategy refers primarily to the *graph of adjacency of singularities* (cf. Afrajmovich *et al.*, 1985). From a practical point of view, developing a continuation strategy also depends on a choice of determining systems used for curve definition.

We refer further to monitoring eigenvalues along a branch of equilibrium points (either generic or degenerate) as a *stability analysis*, and searching for special points on the branch as a *bifurcation analysis*. Monitoring eigenvalues means just their computation at each point of a curve and displaying them on a screen. This clearly enables the user to decide the stability of equilibrium points. The **LOCBIF** convention is such that the eigenvalues are monitored automatically for almost all the curves. The exceptional cases are **Orbit** and **Curve** which don't utilize eigenvalues at all.

6.3 Bifurcation functions

By *bifurcation function* we mean a scalar-valued function of the phase variables and parameters whose zero values correspond to simple (*codim* 1) or higher singularities of equilibrium points.

Bifurcation functions serve two purposes:
1) to find, *i.e.* to detect and locate, special points along curves;
2) to construct determining systems for the bifurcation curves under consideration.

Consider the following bifurcation functions:

$$\psi_1 = \det A \tag{6.3}$$

$$\psi_2 = \Delta_{n-1} \tag{6.4}$$

$$\psi_3 = \ Res\ (P(\lambda), P'(\lambda)) \tag{6.5}$$

$$\psi_4 = a \tag{6.6}$$

$$\psi_5 = L_1 \tag{6.7}$$

Here $A = A(x,p)$ is the Jacobian (linearization) matrix of system (6.1) at the equilibrium point (x,p).$P(\lambda)$ is the characteristic polynomial of the matrix A and $P'(\lambda)$ is its first derivative with respect to λ. Δ_{n-1} denotes the Hurwitz determinant of order $n-1$ associated with $P(\lambda)$(see Gantmacher, 1960, p.193). *Res* stands for the resultant of two polynomials (cf. Gohberg, Lancaster, Rodman, 1982). The values of a and L_1 are the coefficients of normal forms for some singularities of codimension one (see below). Given a Taylor expansion of (6.1) in a neighborhood of the equilibrium point, the values of a and L_1 are determined by the coefficients of the linear and the nonlinear terms. In other words, unlike the first three functions ψ_1, ψ_2, ψ_3, the functions ψ_4, ψ_5 involve not only first but also higher derivatives of $F(x,p)$, and in this sense they are based essentially on nonlinearity of F.

We should notice that all the functions above are initially assumed to be functions of phase variables and parameters defined on equilibrium manifold M. To use these functions in the framework of the continuation method , we need them also to be smoothly defined in a neighborhood of M.

Condition $\psi_1 = 0$ implies that Jacobian matrix A has at least one zero eigenvalue:

$(S1.1)$ $$\lambda_1 = 0$$

Generic equilibrium with this condition has multiplicity two. We refer to such singularity as a *fold point.*

Condition $\psi_2 = 0$ holds if and only if there are two eigenvalues λ_1 and λ_2 with zero sum:

$$(S1.2) \qquad \lambda_1 + \lambda_2 = 0$$

The condition $(S1.2)$ is called a *neutrality condition*, and the corresponding eigenvalues are called *neutral* ones. If the neutral eigenvalues are pure imaginary:

$$(S1.2_f) \qquad \lambda_{1,2} = \pm i\omega \ (\omega > 0)$$

we have a so called *focus case*. This corresponds to the Andronov-Hopf bifurcation (called also Hopf bifurcation) which leads to the appearance of a small-amplitude periodic solution with a period T close to $T_0 = 2\pi/\omega$. Another possible case associated with the condition $\psi_2 = 0$ is that the eigenvalues are real, hence having a different sign and the same magnitude:

$$(S1.2_s) \qquad \lambda_1 = -\lambda_2 \ (\textit{real non-zero})$$

This case is called a *neutral saddle* [6]. In what follows, we use the "Hopf" to denote case $(S1.2)$, which includes two possible variants $(S1.2_f)$ and $(S1.2_s)$.

Condition $\psi_3 = 0$ means the existence of two equal eigenvalues:

$$(S1.3) \qquad \lambda_1 = \lambda_2$$

The function $a = a(x, p)$ is defined at an equilibrium point with zero eigenvalue $\lambda_1 = 0$ and a one dimensional null-space. It reads:

$$a = \frac{1}{2}\frac{d^2}{d\xi^2}\langle e', F(x + \xi e, p)\rangle|_{\xi=0} \tag{6.8}$$

where e and e' are eigenvectors of the matrices A and its transpose respectively corresponding to zero eigenvalue, with normalizing conditions $\langle e, e\rangle = 1, \langle e', e'\rangle = 1$. Here $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in $\mathbf{R}^n$.

[6] It has no bifurcation meaning by itself. However, it may influence a change of stability of a saddle loop and nearby limit cycle if the neutral saddle does have such loop (generally this is a codimension two singularity).

The quantity a also may be characterized as the coefficient of a quadratic term in the normal form of the equations on a one dimensional *center manifold* corresponding to the zero eigenvalue (see Carr, 1981):

$$\dot{u} = au^2 + \dots \tag{6.9}$$

This manifold is tangent to the null-space of A, and one can assume a coordinate u to be chosen along the eigenvector e.

For a generic system, a typical point on a submanifold of degenerate equilibria with zero eigenvalue has a nonzero value of a. This implies that the multiple equilibrium point is double. If the coefficient a vanishes, *i.e.* if conditions $\psi_1 = 0$ and $\psi_4 = 0$ hold together, then the equilibrium, generically, has multiplicity three. This bring us to a codimension two singularity called a *cusp point*, which therefore can be specified as the zero of two functions: ψ_1 and ψ_4.

Notice that the function $a(x, p)$ can be applied not only for the case of one eigenvalue zero, but also for higher singularities such as two zero eigenvalues (with zero Jordan block) and one zero and two pure imaginary eigenvalues. In both these cases a appears as a coefficient in the corresponding normal form (see below). The crucial feature of a is that it allows to distinguish between *double* ($a \neq 0$) and *triple* ($a = 0$) equilibrium points. The terms double, triple *etc.* are related to the multiplicity of a root of the equilibrium point system (6.2): at most two equilibria may appear under perturbation of a double equilibrium point, at most three - from triple equilibrium point, *etc.*

Function $L_1 = L_1(x, p)$ is called the *first Lyapunov coefficient* (or the *first Lyapunov quantity*). It is defined for an equilibrium with a pair of pure imaginary eigenvalues: $\lambda_{1,2} = \pm i\omega$. L_1 is the real part of the coefficient c_1 of the normal form of the equation on a two-dimensional center manifold (see Hassard *et al.*, 1981):

$$\dot{z} = \lambda z + c_1 z^2 \bar{z} + \dots \tag{6.10}$$

where z is a complex variable. Typically, $L_1 \neq 0$. However for certain parameter values, it may vanish. This constitutes a codimension two singularity called a *degenerate Hopf bifurcation.* The corresponding critical boundary defined by equations $\psi_2 = 0, \psi_5 = 0$ separates *subcritical* ($L_1 > 0$) and *supercritical* ($L_1 < 0$) Hopf bifurcations. For more details about the computation

of L_1 see Section 6.11.

Note that this list of bifurcation functions is not restricted to "classical" singularities only (this refers to branching behaviour of equilibria or stability changes) . It enables the user to study some other critical cases such as the double eigenvalue.

6.4 Determining system.

By a *determining system*, we mean a nondegenerate "algebraic" system of the form

$$\mathcal{F}(y) = 0 \tag{6.11}$$

where $\mathcal{F} : \mathbf{R}^N \to \mathbf{R}^{N-1}$ is a smooth mapping. The nondegeneracy of (6.11) means that the Jacobian matrix $\mathcal{F}_y$ has a rank $N-1$ along a solution curve to (6.1) at all but some isolated points on this curve. Typically such points are self-crossing points. We include them into a set of special points (this set may contain also points of other types). We will use the concept of a determining system to define one-dimensional paths on the equilibrium manifold which correspond to different properties of the underlying equilibrium points.

A determining system for generic equilibrium points arises naturally if one uses the system (6.2) with the proper number of variables, namely $n+1$. Typically one uses for such variables all the phase variables and one of the parameters while the other parameters are assumed to be fixed.

A general way to provide a determining system for a nongeneric equilibrium point curve (bifurcation curve) is to combine the equation (6.2) with one or more suitably chosen degeneracy (bifurcation) conditions having the form $\psi_i = 0$; the resulting system is also called an *augmented system*. The number of bifurcation conditions used in the determining system is to be equal to the codimension of the singularity under consideration. Given a codimension ν singularity, the determining system contains $n + \nu$ equations and therefore requires $n + \nu + 1$ variables. For such variables one can take n phase variables and $\nu + 1$ *active* parameters. The active parameters may be arbitrarily chosen among all the system parameters while the remaining parameters are to be kept at fixed values.

We consider next equilibrium and bifurcation curves in the same order as in the list above (*i.e.* ordered by the number of active parameters). So

called nontransversal cases are discussed separately since they require an additional bifurcation function and presume some other relationship between the codimension of a singularity and the number of active parameters.

6.5 Equilibrium curve

The curve type **Equilibrium** stands for a curve of equilibrium points. It requires one active parameter. To simplify notation, choose p_1 for such a parameter. Consider an *active phase-parameter space* with $n+1$ coordinates $(x_1, x_2, \ldots, x_n, p_1)$. The curve in this space is defined by the system (6.2) where all parameters but p_1 are fixed.

We assume that all points of the curve except several isolated ones correspond to hyperbolic equilibria. This implies the nondegeneracy of the determining system (6.2).

The second important assumption is that every nonhyperbolic equilibrium point on the curve has the following features:

a) it represents a singularity of codimension one,

b) it constitutes a generic or *transversal* case for the singularity.

Based on this assumption, one can search for such points along the curve as simple zeros of properly chosen bifurcation functions. Notice that given a generic system (6.1), the second assumption doesn't hold automatically: it requires the nonactive parameters to be fixed at "generic" values. In other words, not only should the system be generic, but also a *slice* in phase-parameter space given by fixed values of nonactive parameters should be generic. Roughly speaking this means that the slice can not pass through the highly degenerate points on M. Notice that similar assumptions should be made also with respect to all other curves below.

As was mentioned before, a stability analysis is performed automatically along a branch of equilibria. A bifurcation analysis involves three bifurcation functions ψ_1, ψ_2, ψ_3, zero values of which indicate special points of type $(S1.1), (S.2), (S1.3)$, respectively. The type $(S1.1)$ is referred to in the program as "**Zero eigenvalue**". For type $(S1.2)$ the two cases, $(S1.2_f)$ and $(S1.2_s)$, are possible: $(S1.2_f)$ is referred to as "**Hopf**" and $(S1.2_s)$ is referred to as **"Neutral saddle"**. The type $(S1.3)$ is indicated as **"Double eigenvalue"**.

Notice that only two $-(S1.1)$ and $(S1.2_f)$ - of four different types of special points on the branch of equilibria show nonhyperbolic equilibria. Two

other types deal with hyperbolic equilibrium points which might be of particular interest in some applications. (See also discussion of continuation strategy in Section 6.10).

When a special point along the curve is found, additional information related to this point and the corresponding normal form is provided whenever possible. For the point of type $(S1.1)a$ value of the coefficient a in (6.9) given by formula (6.8) is computed and reported in the form "**a =** *value*". Normally this value should be nonzero. Notice that the sign of this value has no special meaning (indeed, it may be changed by changing the direction of an eigenvector e). A small or zero value of a indicates possible higher degeneracy such as a cusp singularity. (In particular a cusp singularity means that the slice chosen is not generic). However, for systems with a symmetry group, vanishing $a(x,p)$ appears to be the typical non-exceptional situation which indicates that a pitchfork bifurcation occurs (see also below).

For the point of type $(S1.2_f)$ the value of the first Lyapunov coefficient in (6.10), $L_1 = Rec_1$, is computed and reported in a form "**L1 =** value ". Normally it should be nonzero. A negative (positive) value of L_1 indicates a supercritical (subcritical) Hopf bifurcation.

Bifurcation analysis of an equilibrium curve also involves examining the condition for an extremum of active parameter p_1 along the curve, as well as the condition for a self-crossing point. Notice that any fold point on an equilibrium curve is at the same time an extremal point. So whenever a fold point is found it also will be reported as an extremal point in a form "**Maximum** (or **Minimum**) **of parameter** *name* **is** *value*".

For a generic system, a selfcrossing point can't occur on an equilibrium curve. For a system with a symmetry group or a system having an invariant subspace that doesn't depend on parameters, such points may however appear. It will be reported then as "**Selfcrossing**". Notice that at a self-crossing point one eigenvalue vanishes, $\lambda_1 = 0$, so the corresponding diagnostics might be expected. Two typical cases of selfcrossing point are: pitchfork bifurcation and transcritical bifurcation. For a transcritical bifurcation, both intersecting branches are monotone with respect to the parameter, and λ_1 changes sign along any of them. So regardless of branch, this change is detected by function ψ_1 and reported as "**Zero eigenvalue: a =** value" with a typically nonzero value of a. In the case of a pitchfork bifurcation, only one of two branches (which is non parabola-like) has a change in sign of λ_1. This is diagnosed and reported in the same fashion as above. In this case, a zero

value of a is expected. For the other (parabola-like) branch, the self-crossing point is also an extremal point, which is also reported.

Note. An item **Curve** in the list is reserved for a curve defined by the same formula (6.2) as for **Equilibrium** curve. A difference from **Equilibrium** is that for **Curve**, neither stability nor bifurcation analysis is performed. In other words, choosing the **Curve** option just means performing simple continuation for the problem defined by (6.2) without taking into account a formal meaning of $F(x,p)$ as RHS of differential equation system. In particular this option can be applied for *arbitrary* continuation problems given in a form analogous to (6.2) not only for those which come from the bifurcation analysis of dynamical systems.

6.6 Curves with two active parameters

Let us choose two active parameters, *e.g.* p_1 and p_2, and fix all the others at some particular values. Consider an active phase-parameter space with $n+2$ coordinates $(x_1, x_2, \ldots, x_n, p_1, p_2)$.

6.6.1 Fold (tangent) bifurcation curve

The curve type **Fold** stands for a fold (also called tangent, limit point, or turning point) bifurcation curve defined by the condition ($S1.1$). A determining system for the curve reads:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_1(x,p) &= 0 \end{aligned} \tag{6.12}$$

where all parameters but p_1 and p_2 are fixed.

The projection of a fold curve onto an active parameter plane defines a curve on which a pair of equilibrium points appear or disappear when parameters are changed in such a way to cross this curve *transversally*. For critical and neighboring parameter values there is a one dimensional invariant manifold in the phase space with dynamics described by (6.9).

While tracing a fold curve, the bifurcation functions ψ_2, ψ_3, and ψ_4 are monitored along it to find special points corresponding to zeros of these functions.

Condition $\psi_2 = 0$ means that the equilibrium has a pair of eigenvalues of zero sum (simultaneously with having one zero eigenvalue). We have here one of the following cases:

$(S2.1)$ $\qquad \lambda_1 = \lambda_2 = 0$ (*with zero Jordan block of order two)*

$(S2.2)$ $\qquad \lambda_1 = 0,\ \lambda_2 + \lambda_3 = 0$

$(S2.2_f)$ $\qquad \lambda_1 = 0,\ \lambda_{2,3} = \pm i\omega\ (\omega > 0)$

$(S2.2_s)$ $\qquad \lambda_1 = 0,\ \lambda_2 = -\lambda_3$ (*real non-zero*).

We refer to the case $(S2.1)$ as *Bogdanov-Takens* (BT) singularity, and to the case $(S2.2)$ as *Gavrilov-Guckenheimer* (GG) singularity (for more information about these codimension two singularities see Arnold, 1982; Guckenheimer and Holmes, 1983; Afrajmovich et al., 1985, 1991). In fact only the subcase $(S2.2_f)$ is of interest for GG-singularity; another one formally has codimension one ("degenerate neutral saddle") and doesn't attract much interest in bifurcation studies.

A special point satisfying case $(S2.1)$ may be used as an initial one to compute **Hopf, Double Eigenvalue** and **Double Zero** bifurcation curves. Analogously, the cases $(S2.2_f)$ and $(S2.2_s)$ provide initial points to compute **Hopf** and **Fold + Hopf** bifurcation curves. Notice that the continuation of **Double Zero** and **Fold + Hopf** bifurcation curves requires three active parameters (see below). Notice also that you can easily distinguish between the cases $(S2.1), (S2.2_f), (S2.2_s)$ simply by looking at the eigenvalues in the Value Window. When a special point with $\psi_2 = 0$ is found, it is reported in the form "**Neutrality : a =** *value*". Here *value* is a current value of a given by (6.8). Normally it should be nonzero.

Condition $\psi_3 = 0$ means that the equilibrium has a pair of multiple eigenvalues (simultaneously with having one zero eigenvalue). We may have here the case $(S2.1)$ or another case

$(S2.3)$ $\qquad \lambda_1 = 0,\ \lambda_2 = \lambda_3$ (*real non-zero*).

We should note that the case $(S2.1)$ can hardly be detected on a fold curve by using the function ψ_3 since being zero it doesn't change sign at such a

point. The case ($S2.3$) is more regular in this sense. However it doesn't present a higher singularity. The corresponding special point can be used to start computation of the **Double Eigenvalue** curve.

Finding a special point with the condition at which $\psi_3 = 0$ is reported in the form **"Double eigenvalue : a =** *value*". Here *value* is a current value of a as in the previous case. Typically it should be nonzero.

The last type of special point detected on a fold curve is that defined by the condition $\psi_4 = 0$:

$$(S2.4) \qquad \lambda_1 = 0, \ a = 0. \tag{S2.4}$$

It corresponds to triple equilibrium point also referred to as a cusp point. Such a point may be used as an initial point for tracing a **Cusp** curve. A message associated with the case $\psi_4 = 0$ reads simply **"Cusp"**. Notice that a cusp point is typically detected also as an extremum along the curve of one or both active parameters.

A self-crossing point can't arise on a fold curve provided the system is generic. But for a non-generic system, such a point may occur. In this case, the null-space corresponding to zero eigenvalue, has dimension two, not one as before.

A point at which one of active parameters say p_1 reaches an extremal value may be of the cusp type discussed earlier. The other (equally possible) case is a *regular extremal point*, *i.e.* a non-degenerate fold point ($a \neq 0$) at which the transversality condition for fold bifurcation is not satisfied. This transversality condition is related to the complimentary active parameter p_2 which changes monotonically at this point (see also discussion in the next section). A regular extremal point on the fold curve may be used as an initial point to compute a **Fold + Extr** curve.

6.6.2 Hopf bifurcation curve

The curve type **Hopf** stands for a Hopf bifurcation curve. Recall that by a (generalized) Hopf singularity we mean a case ($S1.2$), *i.e.* when the system has an equilibrium point with a pair of neutral eigenvalues. To define this curve in algebraic form, we use the following equations:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_2(x,p) &= 0 \end{aligned} \tag{6.13}$$

where all parameters but p_1 and p_2 are fixed.

The projection of the Hopf curve onto an active parameter plane with coordinates (p_1, p_2) defines a curve at which a (small-amplitude) limit cycle bifurcates from an equilibrium point. In fact this happens only for the (*focal*) segment of the curve which corresponds to pure imaginary eigenvalues. Still, on another (*saddle*) segment of the Hopf curve the neutral eigenvalues happen to be real, which corresponds to a neutral saddle. Appearance of a neutral saddle in the system doesn't lead to any bifurcation unless global bifurcation comes into consideration (see *e.g.* Afrajmovich et. al., 1985, 1991). The focal and saddle segments meet at a codimension two point of Bogdanov-Takens type. We should notice that since we use the general condition $\psi_2 = 0$ to define the Hopf curve, a transition between these segments doesn't cause any problems.

Three bifurcation functions ψ_1, ψ_3, and ψ_5 are monitored along the Hopf curve. Their zeros indicate the special points of the following type: $(S2.1)$, $(S2.2)$, and also

$$(S2.5) \qquad \lambda_1 + \lambda_2 = 0,\ \lambda_3 = \lambda_4\ (\textit{real non-zero})$$

$$(S2.5_f) \qquad \lambda_{1,2} = \pm i\omega\ (\omega > 0), \lambda_3 = \lambda_4\ (\textit{real non-zero})$$

$$(S2.5_s) \qquad \lambda_1 = -\lambda_2\ (\textit{real non-zero}),\ \lambda_3 = \lambda_4\ (\textit{real non-zero})$$

$$(S2.6) \qquad \lambda_{1,2} = \pm i\omega,\ L_1 = 0$$

More specifically, one has the following possibilities:

- $\psi_1 = 0$ type $(S2.1)$ or $(S2.2)$; reported as **"Zero eigenvalue: a = *value*"**.
- $\psi_3 = 0$ type $(S2.1)$ or $(S2.5)$; reported as **"Double eigenvalue"**.
- $\psi_5 = 0$ type $(S2.6)$; reported as **"Zero Lyapunov value"**.

The first Lyapunov value L_1 computed along the Hopf curve (only when critical eigenvalues are pure imaginary) provides the user with important information about the direction of limit cycle appearance, when the eigenvalues cross the imaginary axis, and its stability.

A Hopf bifurcation curve may also contain a point with two pairs of neutral eigenvalues:

$$(S2.7) \qquad \lambda_1 + \lambda_2 = 0,\ \lambda_3 + \lambda_4 = 0,$$

$$(S2.7_{ff}) \qquad \lambda_{1,2} = \pm i\omega_1,\ \lambda_{3,4} = \pm i\omega_2,$$

$$(S2.7_{fs}) \qquad \lambda_{1,2} = \pm i\omega_1,\ \lambda_3 = -\lambda_4\ (real),$$

$$(S2.7_{ss}) \qquad \lambda_1 = -\lambda_2\ (real),\ \lambda_3 = -\lambda_4\ (real).$$

Such a point is detected on the curve as a self-crossing point (*i.e.* as a standard geometric singularity on the curve). One has in this case a message **"Selfcrossing"**.

A special point of type $(S2.1)$ on Hopf curve can be used to start **Fold**, **Double Eigenvalue** or **Double Zero** curves. A special point of type $(S2.2)$ can be used to start **Fold** and **Fold + Hopf** curves. A point of type $(S2.5)$ is a starting point for a **Double Eigenvalue** curve. A point of type $(S2.6)$ can be used for starting computation of a **Hopf + Lyapunov Zero** curve. On the contrary, a point of type $(S2.7)$ can't be used directly to start tracing another branch of a **Hopf** curve since at this point the determining system (6.13) becomes degenerate.

A point at which one of active parameters reaches an extremal value indicates a degeneracy of nontransversality type. Assume for example that the parameter p_1 has maximim or minimum at a point E on the Hopf curve with $\lambda_{1,2} = \pm i\omega$.Fix the parameter p_1 at its extremal value and leave the parameter p_2 still active. Consider the equilibrium curve which pass through the point E. Then this point becomes a special point of the Hopf type on the equilibrium curve. The transversality condition for Hopf bifurcation means that $d(Re\lambda_{1,2})/dp_2 \neq 0$ along the equilibrium curve. At the point E however, this condition happens to be broken because of tangency of the equilibrium and the Hopf curves. Notice now that because of this degeneracy, the bifurcation function ψ_2 doesn't change sign at the point E along the equilibrium curve (it has a zero of order two). Therefore the corresponding special point of Hopf type can hardly be detected. Referring to the previous discussion of generic

and non-generic slices, one can interpret the nontransversality above saying that the p_2-slice corresponding to the chosen fixed value of p_1 is non-generic.

A regular extremal point on the Hopf curve may be used as an initial point to compute a **Hopf + Extr** curve.

The above discussion about mutual relation between extrema of active parameters and degeneracies of nontransversality type may be carried over to all other curve types.

6.6.3 Double eigenvalue curve

The curve type **Double Eigenvalue** refers to the double eigenvalue curve which is determined by the system of equations:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_3(x,p) &= 0 \end{aligned} \tag{6.14}$$

where all parameters but p_1, p_2 are fixed.

The projection of the computed curve onto an active parameter plane generally defines a curve on which two distinct real eigenvalues of an equilibrium point coincide and become complex. This curve is not bifurcational, however it may be important for applications. For stable equilibria of a two-dimensional system the curve defines a border on which a monotone decay to stationary solution replaces the oscillatory type decay (on the phase plane, a node changes to a focus). One can see the same phenomenon even in the case of a higher dimensional system provided that colliding eigenvalues lie closer to the imaginary axes than others.

Two bifurcation functions ψ_1 and ψ_2 are monitored along the double eigenvalue curve. A zero of the function ψ_1 indicates a special point of either type $(S2.1)$ or $(S2.3)$. The corresponding message is **"Zero eigenvalue: a** $=$ *value*". A zero of the function ψ_2 indicates a special point of type $(S2.5)$. The corresponding message is **"Neutrality"**.

A special point with $\psi_1 = 0$ may be used to start a **Fold** curve, and analagously a special point with $\psi_2 = 0$ may be used to start a **Hopf** curve. If both functions ψ_1 and ψ_2 vanish simultaneously, this means a point of $(S2.1)$ type is detected. It may be used to start **a Double Zero** curve.

6.7 Curves with three active parameters

We consider here singularities of codimension two. Tracing such singularities requires three active parameters. Notice that these and higher singularities may exhibit rather complicated bifurcation diagrams, discussion of which is far beyond the present manual. We recommend (Arnold, 1982; Guckenheimer and Holmes, 1983; Afrajmovich *et al.*, 1985, 1991) for general references concerning these singularities. We also should note that not all known cases of codimension two and three singularities are supported by the current version of **LOCBIF**.

Choose three active parameters, *e.g.* p_1, p_2, and p_3 and fix all the others at some particular values. Consider an active phase-parameter space with $n+3$ coordinates $(x_1, x_2, \ldots, x_n, p_1, p_2, p_3)$.

6.7.1 Bogdanov-Takens bifurcation curve

A Bogdanov-Takens (BT) bifurcation curve, which is referred to in the program as a **Double Zero** curve is a curve of equilibrium points satisfying the singularity condition ($S2.1$). The corresponding determining system has a form:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_1(x,p) &= 0 \\ \psi_2(x,p) &= 0 \end{aligned} \tag{6.15}$$

where all parameters but p_1, p_2 and p_3 are fixed.

Notice that the Bogdanov-Takens bifurcation curve is a common curve of two surfaces corresponding to fold and Hopf singularities respectively.

In the case of BT singularity the value a defined by formula (6.8) is essentially one of the coefficients of the corresponding normal form:

$$\begin{aligned} \dot{u} &= v + \ldots \\ \dot{v} &= au^2 + buv + \ldots \end{aligned} \tag{6.16}$$

Generically, both a and b should be nonzero. Vanishing of either a or b provides two different singularities of codimension three. The bifurcation function ψ_4 is monitored along the BT curve to search for one of these singularities, namely that related to the coefficient a:

($S3.1$) $\qquad \lambda_1 = \lambda_2 = 0$ (*zero Jordan block*), $a = 0$.

Finding such a point is reported as **"Cusp"** since a zero eigenvalue with the condition $a = 0$ yields a cusp point. See (Bazykin *et al.*, 1985, 1989; Dumortier *et al*, 1991) for more details about singularity ($S3.1$).

A special point of the type ($S3.1$) may be used to start a **Cusp** curve.

The other singularity on BT curve is related to the coefficient b in (6.16):

($S3.2$) $\qquad \lambda_1 = \lambda_2 = 0$ (*zero Jordan block*), $b = 0$.

The program doesn't search for this singularity automatically yet. Some indirect way to detect this singularity is to investigate a possibility for a Hopf curve to create a cusp point as being projected onto the active parameter plane. A closely associated phenomenon which clearly indicate such cusp to occur is the following. Suppose that one observes in the projection of a Hopf curve a loop which terminates at a self-crossing point and contains BT or GG point in its internal part. Assume now that under variation of the third parameter such a loop shrinks and disappears at BT or GG point. This means that a point of ($S3.2$) type has been passed. For more information about this singularity see (Berezovskaya and Khibnik, 1985; Basykin *et al.*, 1985, 1989; Dumortier *et al.*, 1987).

Assume that a regular extremal point occurs along a BT curve, say with respect to parameter p_1(which is correspondingly reported). This means the transversality condition for BT singularity is broken, with respect to complimentary parameters p_2 and p_3. This violation of the transversality condition may be characterized geometrically as follows. On (p_2, p_3)-plane, the fold and Hopf curves have at BT point a higher order of tangency than usual quadratic one. For close to extremal values of p_1, there are (locally) two or no BT points exist. A regular extremal point on BT curve may be used as a starting point for a **Double Zero + Extr** curve.

Notice that for two-dimensional systems a BT curve typically doesn't contain self-crossing points unless a symmetry is involved. As for higher dimensions, this becomes possible (see next section).

6.7.2 Gavrilov-Guckenheimer bifurcation curve

A Gavrilov-Guckenheimer (GG) bifurcation curve, which is referred to in the program as a **Fold + Hopf** curve, is defined by the singularity condition ($S2.2$). We use for this case exactly the same determining system (6.15) as for the Bogdanov -Takens case above.

For GG singularity, the value a defined by formula (6.8) appears just as one of the coefficients of a normal form related to $(S2.2_f)$:

$$\begin{aligned} \dot{u} &= au^2 + bz\bar{z} + \dots \\ \dot{z} &= i\omega z + cz^2\bar{z} + dzu + ez^3\bar{z}^2 + \dots \end{aligned} \tag{6.17}$$

where u is a real and z is a complex variable.

Generically, all the coefficients in the normal form should be nonzero. Vanishing of any one of them leads to a singularity of codimension three (more precisely, this refers to the coefficients of the corresponding amplitude equation which are essentially $a, b, Re\ c, Re\ d$ and $Re\ e$). The bifurcation function ψ_4 is monitored along the GG curve to search for one such singularity, namely associated with vanishing of the coefficient a:

$(S3.3_f)$ $$\lambda_1 = 0,\ \lambda_{2,3} = \pm i\omega,\ a = 0.$$

Similarly to the previous case, a zero of the function ψ_4 is reported as **"Cusp"**. Notice that such a zero may also mean a less interesting point of the type

$(S3.3_s)$ $$\lambda_1 = 0,\ \lambda_2 = -\lambda_3\ (real\ nonzero),\ a = 0.$$

Both $(S3.2_f)$ and $(S3.2_s)$ are the two subcases of the following singularity type:

$(S3.3)$ $$\lambda_1 = 0,\ \lambda_2 + \lambda_3 = 0,\ a = 0.$$

A special point of the type $(S3.3)$ on GG curve may be used to start a **Cusp** curve.

If a regular extremal point occurs along a GG curve, with respect to a certain active parameter, this means the transversality condition for GG singularity is broken with respect to two other complimentary active parameters. Such point may be used as a starting point for a **Fold + Hopf + Extr** curve.

A self-crossing point may occur generically on GG(resp. BT) curve. It simply means intersection with BT(resp. GG) curve. Recall that for both cases BT and GG we use the same defining system (6.15), and self-crossing means just a degeneration of this system. This becomes possible if three eigenvalues vanish simultaneously:

$(S3.4)$ $$\lambda_1 = \lambda_2 = \lambda_3 = 0\ (zero\ Jordan\ block\ of\ order\ 3).$$

This is a codimension three singularity not too much studied so far. Anyway such self-crossing point may be used to switch from BT to GG curve and vise versa, although due to degeneracy of (6.15) we can't perform this directly without special efforts.

6.7.3 Cusp bifurcation curve

A cusp bifurcation curve, which is referred to in the program as a **Cusp** curve, is defined by the singularity condition ($S2.4$). The corresponding determining system has the form:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_1(x,p) &= 0 \\ \psi_4(x,p) &= 0 \end{aligned} \tag{6.18}$$

where all parameters but p_1, p_2 and p_3 are fixed.

The bifurcation function ψ_2 is monitored along the cusp curve. A zero of ψ_2 indicates a codimension three point either of type ($S3.1$) or ($S3.3_f$) or ($S3.3_s$). Regardless of the case, the corresponding message is **"Neutrality"**.

A special point of the types ($S3.1$) or ($S3.3$) may be used to start a **Double Zero** or **Fold + Hopf** curves respectively.

Recall that a normal form for cusp singularity is

$$\dot{u} = bu^3 + \dots \tag{6.19}$$

where u is real, and b is assumed to be nonzero. A point with $b = 0$ also might occur on the cusp curve; such a point is called a *swallowtail* and has codimension three. This point can be typically detected on the cusp curve as an extremum with respect to an active system parameter.

6.7.4 Degenerate Hopf bifurcation curve

A degenerate Hopf bifurcation curve, which is referred to in the program as a **Hopf + Lyapunov Zero** curve, is defined by singularity condition ($S2.6$). The corresponding determining system reads:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_1(x,p) &= 0 \\ \psi_5(x,p) &= 0 \end{aligned} \tag{6.20}$$

where all parameters but p_1, p_2 and p_3 are fixed.

Notice that the condition $(S2.6)$ is limited to a Hopf point of focus type only (*i.e.* with pure imaginary eigenvalues). A consequence of this limitation is that unlike the other curves above, a degenerate Hopf bifurcation curve may terminate naturally. This happens when critical eigenvalues approach zero. One can suggest that the termination point (which belongs to a closure of the curve) has higher codimension. Two particular cases of this type are codimension three singularities of $(S3.1)$ and $(S3.2)$ types (cf. bifurcation diagrams in relevant references above).

No bifurcation functions are monitored along the degenerate Hopf bifurcation curve.

A normal form relevant to the singularity $(S2.6)$ is as follows:

$$\dot{z} = i\omega z + c_2 z^3 \bar{z}^2 + \dots \tag{6.21}$$

where z is complex, and by a nondegeneracy condition, $L_2 = Re\ c_2$ is nonzero. The value L_2 is called the *second Lyapunov coefficient (quantity).* Notice that finding its zeros along the degenerate Hopf bifurcation curve appears to be an exciting but nontrivial numerical problem even for two dimensional systems (it requires computing derivatives up to fifth order); the program doesn't deal with this problem.

If a regular extremal point occurs along a degenerate Hopf curve, with respect to a certain active parameter, this means the transversality conditions for the correspondent singularity are no longer satisfied, with respect to the other two complimentary active parameters.

6.8 Curves with four active parameters

We consider here just two cases of singularities of codimension three. Tracing such singularities requires four active parameters. Choose such parameters, *e.g.* p_1, p_2, p_3 and p_4 and fix all the others at some particular values. Consider an active phase-parameter space with $n+4$ coordinates $(x_1, x_2, \dots, x_n, p_1, p_2, p_3, p_4)$.

6.8.1 Degenerate Bogdanov-Takens bifurcation curve (triple point case)

A degenerate Bogdanov-Takens bifurcation curve (triple point case), which is referred to in the program as a **Double Zero + Cusp** curve is a curve of equilibrium points satisfying the singularity condition ($S3.1$). The corresponding determining system has a form:

$$\begin{array}{rcl} F(x,p) & = & 0 \\ \psi_1(x,p) & = & 0 \\ \psi_2(x,p) & = & 0 \\ \psi_4(x,p) & = & 0 \end{array} \tag{6.22}$$

where all parameters but p_1, p_2, p_3 and p_4 are fixed.

A normal form for this singularity reads:

$$\begin{array}{rcl} \dot{u} & = & v + \ldots \\ \dot{v} & = & cu^3 + buv + du^2 v + \ldots \end{array} \tag{6.23}$$

with nondegeneracy conditions $c \neq 0,\ b \neq 0,\ d \neq 0,\ d^2 + 8c \neq 0$.

No bifurcation functions are monitored along this curve. When a regular extremal point occurs, this corresponds to violation of the transversality condition, with respect to the complimentary active parameters.

6.8.2 Degenerate Gavrilov-Guckenchimer bifurcation curve (triple point case)

A degenerate Gavrilov-Guckenheimer bifurcation curve (triple point case), which is referred to in the program as a **Hopf + Cusp** curve is a curve of equilibrium points satisfying the singularity condition ($S3.3$). The corresponding determining system is (6.22), the same as for the previous case.

No bifurcation functions are monitored along this curve. When regular extremal point for an active parameter occurs, this corresponds to violation of the transversality condition, with respect to the complimentary active parameters.

6.9 Curves with nontransversality condition violated

We start with remark that all singularities (and related to them bifurcation functions) discussed above have nothing to do with a concrete dependence of the system (6.1) upon parameters. Such dependence first appears in a determining system for the singularity continuation when we need to choose active parameters and impose the non-degeneracy of this system. Now we arrive at the point at which the dependence upon parameters becomes really crucial: we will discuss here the violation of the transversality conditions with respect to the concrete system parameters. We need first to explain how a determining system is constructed in this case.

6.9.1 Construction of determining system

For a codimension ν singularity undergoing additionally a degeneracy of the nontransversality type, a determining system consists of three parts: 1)n equations defining equilibrium point, 2) iν equations defining singularity conditions, and 3) one equation defining nontransversality condition. This last equation is derived from the previous $n+\nu$ equations which define the singularity itself in a way described above. Since the determining system being constructed contains $n+\nu+1$ equations, it requires $n+\nu+2$ unknowns. In other words, the continuation of the codimension ν singularity with the transversality conditions broken requires $\nu+2$ active parameters not $\nu+1$ as in the standard case above.

Given a codimension ν singularity, consider the corresponding (standard) determining system. Choose a set of ν system parameters. We say that this set is non-generic at the point (x^0, p^0) of the product space if the matrix B is singular,

$$\det B = 0 \tag{6.24}$$

where B is the $(n+\nu)\times(n+\nu)$ linearization matrix at (x^0, p^0) of the determining system with respect to the phase variables and the chosen parameters. The meaning of the condition (6.24) is that the chosen parameters do not provide the generic unfolding of the singularity under consideration. This is equivalent to saying that the transversality conditions are broken with respect to the chosen set of parameters.

Now one can extend the original determining system (of $n+\nu$ equations) by the equation (6.24). This completes a new determining system which defines the nontransversal case. Notice that the matrix B depends on the choice of the set of parameters to be interpreted as non-generic, although this is not reflected explicitly in (6.24). Apparently, a different choice leads to a different matrix B. Therefore the corresponding determining system appears to be also different although constructed in the same fashion.

6.9.2 Nontransversal fold bifurcation curve

A nontransversal fold bifurcation curve referred to in the program as a **Fold + Extr** curve is defined by the condition $(S1.1)$ and the nontransversality type condition (6.24). Since a codimension of a fold singularity is one, this curve requires three active parameters, *e.g.* p_1, p_2 and p_3. Make a choice which one of them will be considered as a non-generic parameter (in general such choce can be made among all system parameters). Consider the determining system (6.12) as depending on the phase variables and one parameter p_1 and compute the linearisation matrix B at a given point in a product space. Now combine the system (6.12) and equation (6.24) with the computed matrix B. This results in a new determining system of $n+2$ equations relevant to our case:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_1(x,p) &= 0 \\ \det B(x,p) &= 0 \end{aligned} \tag{6.25}$$

where all parameters but p_1, p_2 and p_3 are fixed.

The bifurcation function ψ_2 is monitored along this curve to detect special points with condition $(S2.1)$ or $(S2.2)$ being satisfied. The corresponding message is exactly the same as for a fold curve. Since in generic case, the projection of fold and Hopf curves onto the parameter plane shows tangency at their common point (which is essentially of BT or GG type), a nontransversality phenomenon for fold bifurcation implies just the same phenomenon for Hopf bifurcation. This allows us to use the special point to switch to **Double Zero**, **Fold + Hopf**, and also to **Hopf + Extr** curves.

Reaching an extremal point is possible along a nontransversal fold bifurcation curve and may happen in several different variants. We just mention

some of them not pretending to provide a complete classification.

First possibility is a swallowtail point which has codimension three. The next two possibilities are the isola and the self-crossing of a fold curve (in fact these are two complimentary subcases of the same singularity type). To be definite, assume a maximum of the parameter p_3, namely p_3^0 is achieved. Locally, an isola means that in a (p_1, p_2) parameter slice corresponding to p_3 a fold curve forms an annulus (a small closed curve topologically equivalent to a circle), while for values of p_3 greater than p_3^0, a fold curve does not exist at all. A self-crossing of a fold curve implies that for all p_3 close to p_3^0, the fold curve has (locally) two different hyperbola-like branches which approach to one other as p_3 tends to p_3^0.

Notice that reaching an extremal point along nontransversal fold curve means that two different system parameters become non-generic simultaneously. This allows us to switch between different branches corresponding to the degeneracy of the nontransversality type. For this to be done, one needs just to switch the index of the non-generic parameter in the computation of the matrix B.

6.9.3 Nontransversal Hopf bifurcation curve

A nontransversal Hopf bifurcation curve referred to in the program as a **Hopf + Extr** curve is defined by the condition $(S1.2)$ and the nontransversality type condition (6.24). This curve also requires three active parameters, *e.g.* p_1, p_2 and p_3.

The way of constructing the corresponding determining system is exactly the same as for a nontransversal fold curve. One needs just to replace the system (6.12) by (6.13) which is the determining system for a Hopf bifurcation curve. The linearization matrix B is to be computed now from the system (6.13) and a choice is to be made for a non-generic parameter. This gives meaning to the equation (6.24) which now can be added to the system (6.13). The resulting determining system reads:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_2(x,p) &= 0 \\ \det B(x,p) &= 0 \end{aligned} \tag{6.26}$$

where all parameters but p_1, p_2 and p_3 are fixed.

The bifurcation function ψ_1 is monitored along this curve to detect special points with condition ($S2.1$) or ($S2.2$) being satisfied. The corresponding message is exactly the same as for a Hopf curve. A nature of the special point may be of that type as discussed above for nontransversal fold curve (*i.e.* generic BT or GG point with transversality conditions broken simultaneously for fold and Hopf curves). There is however another possibility, namely a codimension three singularity of ($S3.2$) type (remind that near this singularity, a loop of a Hopf curve in the corresponding parameter projection occurs, see section 6.7.4). Notice that for this latter case, the fold curve still remain generic , *i.e.* no transversality conditions are violated.

Switches to **Double Zero**, **Fold + Hopf**, and **Fold + Extr** curves are possible at the special point regarding particular type of singularity it presents.

An extremal point on a nontransversal Hopf curve may easily be of isola or of self-crossing type.

6.9.4 Nontransversal Bogdanov-Takens bifurcation curve

A nontransversal Bogdanov-Takens bifurcation curve referred to in the program as a **Double Zero + Extr** curve is defined by the condition ($S2.1$) and the nontransversality type condition (6.24). This curve requires four active parameters, *e.g.* p_1, p_2, p_3 and p_4.

According to the above scheme, consider the determining system (6.15) for the BT curve, choose two active parameters to be of non-generic type, and compute the linearization matrix B of (6.15) with respect to the phase variables and chosen parameters. Now extend (6.15) with (6.24) where B is the computed matrix. This yields a new determining system relevant for our case:

$$\begin{aligned} F(x,p) &= 0 \\ \psi_3(x,p) &= 0 \\ \psi_2(x,p) &= 0 \\ \det B(x,p) &= 0 \end{aligned} \tag{6.27}$$

where all parameters but p_1, p_2, p_3 and p_4 are fixed.

No bifurcation functions are monitored along a nontransversal BT curve. As usual extremal points may occur along the curve. The switching to an-

other branch of the same type characterized by a different choice of non-generic parameters, might be possible at such a point regarding a case.

6.9.5 Nontransversal Gavrilov-Guckenheimer bifurcation curve

A nontransversal Gavrilov-Guckenheimer bifurcation curve referred to in the program as a **Fold + Hopf + Extr** curve is defined by the condition ($S2.2$) and the nontransversality type condition (6.24). This curve requires four active parameters, *e.g.* p_1, p_2, p_3 and p_4.

A determining system used in this case is exactly the same as for nontransversal BT singularity, namely (6.27).

No bifurcation functions are monitored along a nontransversal GG curve. As usual extremal points may occur along the curve. The switching to another branch of the same type characterized by a different choice of non-generic parameters, might be possible at such a point regarding a case.

6.10 Continuation strategy

We summarize here all information related to curves, special points on them and possible switches to other curves, in a way suitable for further references.

A key question we have in mind is how one can use the continuation of all singularities above to create a kind of route along the equilibrium manifold M. Such a route aims to investigate a bifurcation structure of M. It consists of a number of smooth paths which essentially are segments of curves above. Being considered geometrically, the route presents a connected network on M which ideally should cover all strata of the bifurcation structure. We do not intend to discuss here the algorithmic aspects of constructing an arbitrary or in a certain sense minimal network of this kind (see *e.g.* Khibnik, 1990). We address only the "elementary operations" used in constructing the route. They are: 1) the detecting and the locating of special points on curves, and 2) the switching to another curves whenever possible.

This reference material is organized as follows. For each curve we enumerate all possible messages. The explanation concerns the related bifurcation function, the type of singularity, and other curves (of the same or higher codimension) which pass through the special point under consideration. A convenient way to present (part of) this information graphically is a graph of adjacency (cf. Khibnik *et al.*, 1992, Fig.1).

Curve	Message	Comment
Equilibrium(1)	Zero eigenvalue	ψ_1, fold point ($S1.1$), **Fold (2)**
	Hopf	ψ_2, Hopf point ($S1.2_f$), **Hopf(2)**
	Neutral saddle	ψ_2, Hopf point ($S1.2_s$), **Hopf(2)**
	Double eigenvalue	ψ_3, double eigenvalue point (S1.3), **Double Eigenvalue(2)**
	Extremum	fold point ($S1.1$), **Fold(2)**
	Selfcrossing	pitchfork or transcritical bifurcation (only for system with symmetry)
Fold (2) ($S1.1$)	Neutrality	ψ_2, BT point ($S2.1$) or GG point ($S2.2$), **Hopf(2)**, **Double Eigenvalue(2)**, **Double Zero(3)**, **Fold + Hopf(3)**
	Double eigenvalue	ψ_3, BT point ($S2.1$), or ($S2.3$) type, **Double Eigenvalue (2)**, **Double Zero(3)**
	Cusp	ψ_4, cusp point ($S2.4$), **Cusp(3)**
	Extremum	cusp point ($S2.4$) or nontransversal fold point, **Cusp(3)**, **Fold + Cusp(3)**
	Selfcrossing	does not occur for generic systems
Hopf(2) ($S1.2$)	Zero eigenvalue	ψ_1, BT point ($S2.1$) or GG point ($S2.2$), **Fold(2)** **Double Eigenvalue(2)** **Double Zero(3)**
	Double eigenvalue	ψ_3, BT point ($S2.1$), or ($S2.5$) type,

		Double Eigenvalue(2), **Double Zero(3)**
	Zero Lyapunov	ψ_5, degenerate Hopf point ($S2.6$), **Hopf + Lyapunov Zero (3)**
	Extremum	nontransversal Hopf point, **Hopf + Cusp(3)**
	Selfcrossing	double Hopf point ($S2.7$), **Hopf(2)** (the other branch) (*)
Double Eigenvalue(2) ($S1.3$)	Zero eigenvalue	ψ_1, BT point ($S2.1$), or ($S2.3$) type, **Fold(2)**, **Double Zero(3)**
	Neutrality	ψ_2, BT point ($S2.1$), or ($S2.5$) type, **Hopf(2)**, **Double Zero(3)**
	Extremum	nontransversal double eigenvalue point
	Selfcrossing	two pairs of colliding eigenvalues, **Double Eigenvalue(2)** (the other branch) (*)
Double Zero (3) ($S2.1$)	Cusp	ψ_4, degenerate BT point ($S3.1$) **Cusp(3)**, **Double Zero + Cusp(4)**
	Extremum	nontransversal BT point, **Double Zero + Extr(4)**,
	Selfcrossing	triple zero eigenvalue ($S3.4$), **Fold + Hopf(3)** (*)
Fold + Hopf (3) ($S2.2$)	Cusp	ψ_4, degenerate GG point ($S3.3$), **Cusp (3)**, **Fold + Hopf + Cusp(4)**

	Extremum	nontransversal GG point, **Fold + Hopf + Extr(4)**,
	Selfcrossing	triple zero eigenvalue ($S3.4$), **Double Zero(3)** (*)
Cusp (3)	Neutrality	ψ_2, degenerate BT point ($S3.1$) or degenerate GG point ($S3.3$),
($S2.4$)		**Double Zero(3)**, **Fold + Hopf(3)**, **Double Zero + Cusp(4)**, **Fold + Hopf + Cusp(4)**
	Extremum	swallowtail point or nontransversal cusp point
	Selfcrossing	does not occur for generic systems
Hopf + Lyapunov Zero(3)	Extremum	nontransversal degenerate Hopf point
($S2.6$)	Selfcrossing	does not occur for generic systems
Double Zero + Cusp(4)	Extremum	nontransversality conditions violated
($S3.1$)	Selfcrossing	triple equilibrium point with triple zero eigenvalue, **Fold + Hopf + Cusp(4)** (*)
Fold + Hopf + Cusp(4)	Extremum	nontransversality conditions violated
($S3.3$)	Selfcrossing	triple equilibrium point with triple zero eigenvalue, **Double Zero + Cusp(4)** (*)
Fold + Extr (3)	Neutrality	ψ_2, BT point ($S2.1$) or GG point ($S2.2$), **Double Zero(3)**, **Fold + Hopf(3)** **Hopf + Extr(3)**

	Extremum	swallowtail point or fold isola or fold self-crossing, **Cusp(3)**, **Fold + Extr (3)** (the other branch)
	Selfcrossing	does not occur for generic systems
Hopf + Extr (3)	Zero eigenvalue	ψ_1, BT point ($S2.1$) or GG point ($S2.2$) or degenerate BT point of ($S3.2$) type, **Double Zero(3)**, **Fold + Hopf(3)**, **Fold + Extr(3)**
	Extremum	Hopf isola or Hopf self-crossing, **Hopf + Extr(3)** (the other branch)
	Selfcrossing	does not occur for generic systems
Double Zero + Extr(4)	Extremum	
	Selfcrossing	does not occur for generic systems
Fold + Hopf + Extr(4)	Extremum	
	Selfcrossing	does not occur for generic systems

The figure in parenthesis denotes the number of active parameters required. A star (*) near a curve name indicates a degeneracy of the corresponding determining system which means one can not switch directly to such curve but probably some small perturbation would be sufficient.

6.11 More about bifurcation functions

This section aims to discuss briefly what we mean by a smooth extension of bifurcation functions in a neighbourhood of M(see Section 6.3), and also to present a computational algorithm for the first Lyapunov coefficient.

Bifurcation functions are used to construct determining systems described

above. Recall that such a system is assumed to have sense in some neighbourhood of the curve it defines and to be non-degenerate on the curve. That is why the bifurcation functions have to be suitably extended.

For functions ψ_1, ψ_2, and ψ_3 such an extension is trivial since they use only the linearization matrix A of (6.1) at an equilibrium point. Naturally one can assume that the matrix A is computed at any point close to the equilibrium and apply function definitions (6.3)-(6.5) to this matrix.

As far as the function ψ_4 is concerned, let the linearization matrix A be computed at a given point (x^0, p^0) of a product space close to a submanifold of equilibrium points with zero eigenvalue and one-dimensional null space. Let $\tilde{A}$ be some singular rank $n-1$ matrix close to A (in fact we get such matrix by LU-decomposition of A with complete pivoting and replacing then the last small pivot by zero), with e and e' be the corresponding eigenvectors of $\tilde{A}$ and its transpose. Then the formula (6.8) may be applied as before.

Now we give a constructive definition of the first Lyapunov coefficient associated with the Hopf bifurcation. We will assume from the very beginning that the Hopf condition $Re\ \lambda_{1,2} = 0$ is satisfied approximately. Namely, let an equilibrium point (x^0, p^0) be given, at which the linearization matrix A has a pair of complex-conjugate eigenvalues $\lambda_{1,2} = \alpha \pm i\omega$ close to the imaginary axis while having other eigenvalues located relatively far from it. Let vectors e_1, e_2 and e'_1, e'_2 lie in the corresponding two-dimension invariant subspaces of A and its transpose, respectively, and the following conditions are satisfied: $Ae_1 = \alpha e_1 - \omega e_2, Ae_2 = \omega e_1 + \alpha e_2, \langle e_1, e_1\rangle\langle e_2, e_2\rangle - [\langle e_1, e_2\rangle]^2 = 1, \langle e'_1, e_1\rangle = 1, \langle e'_1, e_2\rangle = 0, \langle e'_2, e_1\rangle = 0, \langle e'_2, e_2\rangle = 1$.

Extend these two pairs of vectors to two bi-orthogonal sets of vectors

$$(e_1, e_2, \ldots, e_n),\ (e'_1, e'_2, \ldots, e'_n)$$

where $\langle e'_i, e_j\rangle = \delta_{ij}$, and δ_{ij} is the Kronecker's symbol. Denote by $\Sigma = (\sigma_{ij})$ an $(n-2)\times(n-2)$ matrix with components $\sigma_{ij} = \langle e'_{i+2}, Ae_{j+2}\rangle, i, j = 1, \ldots, n-2$.

A number of direction derivatives of second and third order has to be computed at point (x^0, p^0):

$$a^k_{ij} = \frac{1}{i!j!}\frac{\partial^2}{\partial\xi^i\partial\eta^j}\langle e'_k, F(x^0 + \xi e_1 + \eta e_2, p^0)\rangle, i = 0, 1, 2, j = 2 - i,$$

$$k = 1, \ldots, n,$$

$$b_{ij}^k = \frac{\partial^2}{\partial\xi\partial\eta}\langle e_k', F(x^0 + \xi e_i + \eta e_j, p^0)\rangle, i, k = 1, 2, j = 3, \ldots, n,$$

$$c_{ij}^k = \frac{1}{i!j!}\frac{\partial^3}{\partial\xi^i\partial\eta^j}\langle e_k', F(x^0 + \xi e_1 + \eta e_2, p^0)\rangle, k = 1, 2, j = k-1, k+1, i = 3-j$$

The following evaluations provide a value of the first Lyapunov coefficient, essentially only for two-dimensional system being a projection of the original system onto a plane spanned on e_1, e_2:

$$r_1 = a_{20}^1 + a_{02}^1,\ r_2 = a_{20}^2 + a_{02}^2,$$
$$s_1 = a_{20}^1 + a_{11}^2 - a_{02}^1,\ s_2 = a_{20}^2 - a_{11}^1 - a_{02}^2$$
$$r_3 = \frac{3}{8}(s_1 r_1 - s_2 r_2) + \frac{1}{4}(r_1^2 + r_2^2)$$
$$r_4 = s_1^2 + s_2^2,\ r_5 = s_1 r_2 + s_2 r_1,$$
$$\tilde{L}_1 = \alpha\left[\frac{r_3}{(\alpha^2+\omega^2)} + \frac{1}{8}\frac{r_4}{(\alpha^2+9\omega^2)}\right] - \frac{1}{8}\frac{\omega r_5}{(\alpha^2+\omega^2)} +$$
$$+\frac{1}{8}(3c_{30}^1 + c_{21}^2 + c_{12}^1 + 3c_{03}^2) \tag{6.28}$$

Introduce now $(n-2)$-vectors $b_i^k = (b_{ij}^k), j = 3, \ldots, n$, and $d_{ij} = (a_{ij}^k), k = 3, \ldots, n$. The last set of formulas completing the computations of the first Lyapunov coefficient in the n-dimensional case reads:

$$u_1 = b_1^1 - b_2^2,\ u_2 = b_2^1 - b_1^2,\ u_3 = b_1^1 + b_2^2,$$
$$v_1 = \Sigma^T u_1 - 2\omega u_2,\ v_2 = \Sigma^T u_2 + 2\omega u_1,$$
$$T = \left(\Sigma^2 + 4\omega^2 I\right)^T,$$
$$h_1 = T^{-1}v_1,\ h_2 = T^{-1}v_2,\ h_3 = (\Sigma^T)^{-1}u_3,$$
$$L_1 = \tilde{L}_1 - \frac{1}{8}\left[\langle h_1, d_{20} - d_{02}\rangle + \langle h_2, d_{11}\rangle + 2\langle h_3, d_{20} + d_{02}\rangle\right] \tag{6.29}$$

Here I denotes the identity matrix, and A^T stands for A transpose. We conclude with the remark that all computations above may be accomplished succesfully (without regard to the meaning of a resulting value) without necessarily the point (x^0, p^0) being an equilibrium point. It just should be close to the equilibrium manifold and satisfy the conditions above on the spectrum of A.

7 Curve computation

In this section you will learn about a general continuation algorithm customized in **LOCBIF** and about particular techniques used to evaluate bifurcation functions and to search for special points on a curve. You will be given some information about ODE solvers available in **LOCBIF** for orbit computations. A list of all computational parameters will be presented and discussed here.

7.1 Continuation algorithms

7.1.1 Basic continuation scheme

A continuation problem for equilibrium and all bifurcation curves may be formulated as follows. We have to continue a curve $\gamma = \gamma(\tau)$ in space $\mathbf{R}^s$ with coordinates $y = (y_1, y_2, \ldots, y_s)$ starting from a given initial point $y^{(0)} \in \mathbf{R}^s$; here $\tau \in \mathbf{R}$ is some parametrization of the curve. The curve is defined by $s-1$ smooth scalar functions $G_1(y), G_2(y), \ldots, G_{s-1}(y)$ as their zero manifold:

$$\begin{aligned} G_1(y) &= 0 \\ G_2(y) &= 0 \\ &\ldots \\ G_{s-1}(y) &= 0 \end{aligned} \tag{7.1}$$

Continuation of the curve γ given by (7.1) means computing a sequence of points $y^{(1)}, y^{(2)}, y^{(3)}, \ldots$, which satisfy (7.1). Moreover, we assume that the sequence provides a "reasonable" approximation of γ as far as its continuity, curvature and location of special points on it are concerned.

The first point $y^{(1)}$ is expected to be close to the initial point $y^{(0)}$. Notice that the initial point $y^{(0)}$ plays two roles: (1) it serves as an initial guess for searching the first point on the curve; (2) it indicates implicitly the branch to be computed (in case there are several branches defined by system (7.1)).

The basic assumption for a (smooth) continuation method is that the system (7.1) is non-degenerate on curve γ, *i.e.* the linearisation matrix $\mathcal{A} = G_y, G = (G_1, \ldots, G_{s-1})$, is nonsingular, of rank $s-1$. Theoretically, this allows us to apply the Implicit Function Theorem for constructing a branch of solutions to (7.1), and practically, to implement the related ideas in a numerical algorithm. We should outline that the non-degeneracy conditions

may be violated at some (isolated) points on curve γ. Typically, such points, which are called *self-crossing* (or bifurcation) points, reveal the existence of other branches.

Basically, the continuation scheme involves the following stages which are performed repeatedly, except the first one:

1) Searching the first point on the curve.
2) Guessing the next point on the curve (*predictor*).
3) Computing the next point (*corrector*).
4) Testing the computed point.
5) Choosing the new step size.
6) Processing the computed point.

Stages 1)-5) deal with so called regular points and represent a basic (problem-independent) part of a continuation scheme. The stage 6) is problem-dependent. In our approach, the idea behind this stage is to analyze the behaviour of some (test) functions defined along the curve, with particular interest to their roots. This results in computing some more points on the curve; these additional points are referred to as *special* points. Each new sequence of special points fits to the curve between the two last regular points. Finally, one obtains a (mixed) sequence of both regular and special points, naturally ordered along the curve.

In **LOCBIF** the curve continuation is supported by a general code **BEETLE** which incorporates the finding of special points.

7.1.2 Computing regular points

Assume that regular points $y^{(1)}, \ldots, y^{(k-1)}$ are already computed. To find the next regular point on the curve, we use a tangent prediction and Newton iteration as a correction scheme.

Predictor. A step of size h is made from the previous point $y^{(k-1)}$, in the tangent direction along the curve: $\tilde{y}^{(k)} = y^{(k-1)} + hu^{(k-1)}$, where $u^{(k-1)}$ is a normalized tangent vector, $G_y(y^{(k-1)})u^{(k-1)} = 0, \| \ u^{(k-1)} \| = 1, \langle u^{(k-1)}, y^{(k-1)} - y^{(k-2)} \rangle > 0$.

Corrector. We fix a plane $y_j = const$, where $const = \tilde{y}_j^{(k)}$ and y_j is (locally) the most rapidly changing variable called *leading variable*. The leading variable y_j corresponds to the largest component of the tangent vector $u^{(k-1)}$ and is used for local parametrization of the curve. Newton iterations are

performed within the plane to find a root of equations (7.1). When the iterations are successful (see stopping criteria below) the result of the iterations is considered to be the next point $y^{(k)}$ on the curve. More details about corrector iterations will be given in Section 7.1.5.

Testing the computed point. The computed point $y^{(k)}$ still has to pass some tests to be finally accepted as the next regular point on the curve. There are three tests: a tangent, a curvature and (optionally) a special point test. The tangent test simply checks whether tangent vector $u^{(k)}$ to the curve can be computed at the point $y^{(k)}$. The curvature test checks whether the angle between the tangent vector $u^{(k-1)}$ and the secant vector $\tilde{u}^{(k-1)} = y^{(k)} - y^{(k-1)}$ passing through two last points, isn't "too large" (and similarly for the tangent vector $u^{(k)}$ at the point $y^{(k)}$). If it is large, a jump to another branch or different part of the same branch may be suspected, which should be prevented, of course. The curvature test also aims at linking the "density" of the computed points to the (local) curvature of a branch: a larger curvature should lead to the computation of a more dense set of regular points. Specifically, for the curvature test we check the following condition: $\rho_1 = \cos(u^{(k-1)}, \tilde{u}^{(k-1)}) \geq \rho_0 > 0$, where ρ_0 is a given value (it is derived from the user-defined parameter **Angcrv**). If the current point $y^{(k)}$ satisfies the curvature test, we start to search for special points on the curve located between points $y^{(k-1)}$ and $y^{(k)}$(see further). In case of certain failures of searching procedure, the point $y^{(k)}$ may be also rejected.

Choosing step size. Based on results of the Newton corrector iterations and of the above tests, the point $y^{(k)}$ is accepted or rejected. This also means that the current step size h is considered as acceptable or not. If not, the step size is reduced by a factor 2, and the computation of the next point $y^{(k)}$ starts from stage 2 again. If point $y^{(k)}$ is accepted, then the program adjusts the step size using one of the implemented step size control algorithms, regarding user's choice (parameter **Algcrv**). In any case, after making a successful step, a choice of the next step size *depends* on the local curvature (such dependence is controlled by user defined parameter **Angcrv**), which may cause either increasing or decreasing of the step size. Lower and upper limits for step size restrict its variation. If (due to reductions)h becomes smaller than the minimally allowed value the continuation is terminated with the corresponding message. The initial, minimal and maximal step sizes (**H0**, **Hcrmin** and **Hcrmax** respectively) are to be defined by the user. We address to particular strategies of the step size control in Section 7.1.8.

Other numerical aspects. The linearization matrix of (7.1) is required for computing the tangent vector and for the corrector iterations. We calculate it by numerical differentiation of functions $G_i, i = 1, 2, \ldots s-1$. Typically, in the course of Newton iterations, only a few matrix reevaluations are necessary. We assume that the matrix is evaluated at initial iterations and then is kept fixed. A number of such initial iterations as well as maximally allowed number of corrector iterations (**Modit** and **Maxit** respectively) are to be given by the user. Notice that the Jacobian matrix of the ODE system which is required for evaluating bifurcation functions, is also computed numerically. The user is supposed to provide increments for computing both matrices (**Dhcrv** and **Dhjac**). For treating numerical linear algebra problems arising in the continuation code and in the computation of bifurcation functions, we mostly use standard routines from LINPACK (Dongarra, *et al.*, 1978) or similar to them.

7.1.3 Special points

The continuation code locates three types of special points: self-crossing points, local extremum points and zeros of external (called also "user-defined") functions defined on the curve. In some sense the first two types of special points characterize the curve itself (its geometry, branching properties *etc.*), while the third type describes additional (external) features of the curve. Despite these differences, we use the same technique to compute all special points. It is based on the assumption that every special point may be defined as a root of a properly chosen continuous function defined along the curve, called *test* function. If the function changes sign at a special point, a secant method may be used to find its root.

Assume that a curve segment is given bounded by regular points $y^{(k-1)}$ and $y^{(k)}$, and that y_j is a local leading variable. By definition, at a self-crossing point the Jacobian matrix $\mathcal{A}$ of (7.1) is singular and has rank $< s-1$. Consider the determinant of the $(s-1) \times (s-1)$ submatrix $\tilde{\mathcal{A}}$ of $\mathcal{A}$ obtained by removing the j-th column, where j is the index of locally leading variable. At a self-crossing point, determinant $\det(\tilde{\mathcal{A}})$ changes sign along the branch. Therefore, it can be used as the test function related to self-crossing points.

Assume that a component y_i with index $i, i \neq j$, reaches a local extremum along the branch between regular points $y^{(k-1)}$ and $y^{(k)}$, *i.e.* that a turning point with respect to y_i is reached. This implies that the i-th components

of the related tangent vectors $u^{(k-1)}$ and $u^{(k)}$ have different sign. We use the i-th component of the tangent vector u to the curve, normalized such that $\| u \| = 1, \langle u^{(k-1)}, u \rangle > 0$, as the test function related to a local extremum of variable y_i. To analyze local extrema for all variables (except the leading one which is supposed to behave monotonically on the segment), one clearly needs $s-1$ different test functions defined similarly as above. In **LOCBIF**, only those components of the continuation problem (7.1) which represent system parameters, are examined with respect to extrema.

To locate roots of external functions defined on the curve, we use these functions directly as the corresponding test functions. It is required therefore that these functions are continuous and change sign at their roots.

Notice that for general usage of the continuation code, external functions are to be defined by the user. In **LOCBIF** however, these functions are defined inside the program. They are treated as bifurcation functions which depend on the curve type (see Section 6).

The user defined parameters **Epscrs**, **Epsext**, **Epszer** provide tolerances for computing the three types of special points described above. The user can suppress the calculation of special points (for each type independently).

7.1.4 Determining systems and bifurcation functions

The continuation code requires an (external) procedure for the evaluation of the functions G_i in (7.1). Recall (see Section 6) that system (7.1) represents a determining system for equilibrium and bifurcation curves. Such a system is built automatically from the ODE system (6.1), which is provided by the user. This procedure depends on a curve type and involves the right-hand-sides of (6.1) and bifurcation functions corresponding to the curve type.

Numerical computation of the bifurcation functions $\psi_1 - \psi_5$, defined by (6.3)-(6.7), is implemented in a more or less straightforward way (see also additional remarks in Section 6.11). The functions ψ_1, ψ_2, ψ_3 require the linearization matrix which is computed by numerical differentiation (see above) and some standard operations from linear algebra. In addition, the functions ψ_4, ψ_5 require higher derivatives. Numerical differentiation is used in this case as well. The number of higher derivatives required for the evaluation of ψ_4, ψ_5 is not high (of order n) due the use of directional derivatives in (6.6) and (6.28). (The evident drawback of this approach is that the accuracy of the computations might decrease when higher derivatives are involved.

However, our experience in this respect is quite good.)

The calculation of nontransversal bifurcation curves requires special consideration since it involves additional information. This concerns the so called nongeneric parameters. One can have one or two such parameters depending on the curve type. The parameter **Iprsng** which is to be set by the user deals with the indices of nongeneric parameters in the list of all system parameters.

7.1.5 Corrector iterations

Let $\tilde{y}^{(k)}$ be a point obtained by the predictor, and y_j is the local leading variable. We use the following standard Newton iteration procedure:

$$\tilde{y}^{(k,l+1)} = \tilde{y}^{(k,l)} + \Delta^{(l+1)},\ l = 0, 1, \ldots$$

where $\Delta^{(l+1)} = -(H_y)^{-1} H(\tilde{y}^{(k,l)})$, $H(y) = (G(y), y_j - \tilde{y}_j^{(k)})$, and $\tilde{y}^{(k,0)} = \tilde{y}^{(k)}$.

The stopping criterion involves two conditions:

$$\max_i \left| \frac{\Delta_i^{(l+1)}}{1 + \mid \tilde{y}_i^{(k,l)} \mid} \right| \leq \epsilon_c$$

and

$$\max_i \left| G(\tilde{y}_i^{(k,l+1)}) \right| \leq \epsilon_c$$

Here ϵ_c is the given tolerance (parameter **Epscrv**). If the maximal number of iterations **Maxit** is exceeded, or $\Delta^{(l+1)}$ "substantially" grows with l, or functions $G(\tilde{y}_i^{(k,l+1)})$ become "large enough", the corrector iterations terminate without accepting the current point.

The Jacobian matrix H_y of map $H = H(y)$ is reevaluated during the first **Modit** corrector steps; it is kept fixed during the following iterations.

7.1.6 Locating root of test function

Let $g = g(y)$ denotes a test function defined along the curve. Suppose that a root of g is detected between the two regular points $y^{(k-1)}$ and $y^{(k)}$. This means that g has different sign at these points. Moreover, for detection of the root we impose the following additional conditions:

$$|g(y^{(k-1)})| > \epsilon_r \text{ or } |g(y^{(k)})| > \epsilon_r, \text{ and}$$

$$|(g(y^{(k)}) - g(y^{(k-1)}))/(y_j^{(k)} - y_j^{(k-1)})| > 0.1 \cdot \epsilon_r$$

Here ϵ_r is the given tolerance for root finding (parameter **Epscrs** or **Epszer** or **Epsext**), and y_j is the local leading variable. Notice that if g changes slowly along the curve or if g is "badly scaled", *i.e.* it always has small values, the above conditions do not allow us to detect the root. From the other side, we are saved from finding false roots which may occur due to the computational error in the evaluation of g.

Once the root is detected, the standard secant method is used to find the root. By this, we assume the curve is locally parametrized by $y_j, y = y(y_j)$, and therefore $g = g(y_j)$. Then, one iteration of the secant method involves three steps:

(1) calculating initial guess for Newton corrector iterations:

$$\tilde{y}^{(k,l+1)} = y^{(k,l)} - g(y^{(k,l)}) \frac{y^{(k,l)} - y^{(k,l-1)}}{g(y^{(k,l)}) - g(y^{(k,l-1)})}$$

where $y^{(k,l)}$ and $y^{(k,l-1)}$ are points on the curve obtained in the previous step of the secant method;
(2) the computation of point $y^{(k,l+1)}$ on the curve using the corrector iterations described above;
(3) the evaluation of the function g at point $y^{(k,l+1)}$.

Assume that g changes sign between points $y^{(k,l)}$ and $y^{(k,l+1)}$, otherwise set $y^{(k,l)} = y^{(k,l-1)}$. The stopping criterion requires that at least one of the two conditions is satisfied:

$$| \, y_j^{(k,l+1)} - y_j^{(k,l)} \, | \leq \epsilon_r,$$

$$| \, g(y^{(k,l+1)}) \cdot [(g(y^{(k,l+1)}) - g(y^{(k,l)}))/(y_j^{(k,l+1)} - y_j^{(k,l)})] \, | \leq 0.1 \cdot \epsilon_r$$

In this case $y^{(k,l+1)}$ is treated as the found root of the test function.

The secant method may fail due to the following reasons:
(a) the corrector iterations fail;
(b) function g can not be evaluated at the current point;
(c) the maximal allowed number of secant iterations is exceeded.
In all these cases the procedure for locating roots terminates. However,

it returns the best found approximation to the root and a warning indicating that the desired accuracy hasn't been achieved. The corresponding message contains in such a case a question mark, for example:

Zero eigenvalue (?): a = *value*

The maximal number of secant iterations is equal to 10. For corrector iterations used in the root finding the standard maximal value of corrector iterations (**Maxit**) is used. However, in case of locating a self-crossing point this value is increased by factor 5 due to slow convergence near such a point.

7.1.7 Searching first point

To search for the first point, the corrector iterations are used. The given initial point $y^{(0)}$ serves as the initial guess. Recall that corrector iterations use the leading variable y_j. When searching the first point the leading variable is computed through the tangent vector at $y^{(0)}$ as usual. By this, we neglect the fact that $G(y^{(0)})$ may not be equal to zero (formally we compute the tangent vector to the curve given by $G(y) = G(y^{(0)})$).

If the corrector iterations do not succeed starting from the given initial point and using the leading variable chosen as described above, the program retrieves corrector iterations with all other possible choices of the leading variable (*i.e.* all other variables are tested sequentially as leading variables).

7.1.8 Step size control

The concept of the step size control in **LOCBIF** has been already discussed above. Here we present formulas used for choosing the next step size $h^{(k+1)}$ if the current step, with the step size $h^{(k)}$, is successful (recall that in case of rejecting of the current point the step size is always reduced by factor 2).

Let $\alpha^{(k)}$ be the angle between the secant and the tangent vectors at the current point $y^{(k)}$ and α be the desired angle. Introduce two values $Angmin = 0.6 \cdot \alpha$ and $Angmax = 2 \cdot \alpha$ which have the meaning of the lower and upper limits, respectively, for the angle between the secant and the tangent vectors. The value $Angmax$ is used in the curvature test: if $\alpha^{(k)} > Angmax$ the current point is rejected. Consider now three strategies of the step size control.

Doubling/halving. If $\alpha^{(k)} < Angmin$, the step size is increased by factor $2, h^{(k+1)} = 2 \cdot h^{(k)}$.

Explicit dependence on local curvature. Let $\tilde{u}^{(k)}$ be the secant vector. Then $h^{(k+1)} = \sin\alpha \cdot \parallel \tilde{u}^{(k)} \parallel /(2\sin\alpha^{(k)})$.

Implicit dependence on local curvature. Let $\gamma = \max(1-\kappa\cdot(\alpha^{(k)}-\alpha), 0.3)$. Then $h^{(k+1)} = \gamma \cdot h^{(k)}$. Here κ is a parameter of the algorithm. We use the value $\kappa = 0.1$.

We should notice that there are some additional factors which may increase or decrease the computed value of $h^{(k+1)}$. This depends, in particular, on that whether the step rejection has occured during the computation of the current point.

7.2 Using the continuation code

7.2.1 Initial data

The continuation code is used in **LOCBIF** to compute curves listed in Section 6.1 (except of **Orbit** type). To start the curve computations, the following initial data have to be provided:

a) curve type,
b) list of active parameters $p_{i_1}, \ldots, p_{i_k}$, where the number of active parameters k should be in agreement with the chosen curve type (see Section 6.1),
c) the initial point in a product space (*i.e.* initial values for all phase variables and system parameters),
d) continuation parameters.

Recall that the values of the non-active parameters are kept fixed during the computation. The role of these values it to define a slice in a product space which is considered as the (continuation) state space of problem (7.1). The continuation code deals with the phase variables $x_1, \ldots, x_n$ and the active parameters $p_{i_1}, \ldots, p_{i_k}$ only. The state vector y for the continuation is thus given by $(y_1, \ldots, y_s) = (x_1, \ldots, x_n, p_{i_1}, \ldots, p_{i_k})$. The dimension of y is equal to $s = n + k$. Initial values for all components of the state vector y must be given.

All data a)-d) can be specified in **LOCBIF** by using the interactive window interface (see Section 5). We should emphasize the necessity of a careful choice of the initial point. Otherwise the continuation code may fail to find the first point on the curve, or may start tracing another branch than that is actually desired. Some related issues are discussed in Section 7.2.5 below.

7.2.2 Continuation parameters

The list of continuation parameters includes five groups of parameters regarding the following aspects: (1) step size control, (2) tolerances, (3) control of corrector iterations, (4) computing derivatives, (5) evaluating determining system.

A. Step size control

H0crv	initial step size
Hmxcrv	maximal step size
Angcrv	parameter used in the step size control algorithm
Algcrv	type of step size control algorithm
	± 1 - doubling/halving
	± 2 - explicit dependence on local curvature
	± 3 - implicit dependence on local curvature

The initial step size **H0crv** is used in the first predictor step. Sign of **H0crv** indicates (implicitly) in which direction a curve will be traced. Recall that the **LOCBIF** interface provides an easy way to choose one of two possible directions which have conventional names *forward* and *backward* (see Section 5). Computing a curve forward or backward means using a given step size **H0crv** or $-$**H0crv** respectively.

The parameter **Hmxcrv** restricts a possible growth of a continuation step size. Evidently it should be not smaller than the initial step size.

The parameter **Angcrv** measures (in degrees) a desired angle between the tangent vector to the curve at a current point and a secant vector passing through the previous and the current points. One can get the initial guess for a value of **Angcrv** by trying to estimate a desirable number of continuation points on unit cycle. If q is such number, than **Angcrv**$\approx 180/q$. For better tuning of **Angcrv** one might need a few experiments. The internal parameters *Angmin* and *Angmax* which define maximal and minimal angles respectively (see Section 7.1.8) are derived from **Angcrv** according to the following formulas: $Angmax = 2.0 \cdot$**Angcrv**, $Angmin = 0.6 \cdot$**Angcrv**.

The parameter **Algcrv** allows the user to choose a type of step size control algorithm (see Section 7.1.8). Besides, it also has to do with enabling or disabling a curvature test. The curvature test is enabled (disabled) if **Algcrv** is positive (negative).

Notice that a minimal step size isn't included in the list above. We

assume its value is linked with the tolerance parameter **Epscrv** (see below). In particular we assign a minimal step size equals to $10^{-2}\cdot$**Epscrv**.

B. Tolerances

Epscrv tolerance for Newton corrector iterations
Epscrs tolerance for selfcrossing point location
Epszer tolerance for location of roots of bifurcation functions
Epsext tolerance for location of extremal values of the parameters

Note that a zero value of the parameters **Epscrs**, **Epszer**, **Epsext** has a conventional meaning: it suppresses detecting and locating the corresponding special points on a curve.

C. Control of corrector iterations

Maxit maximal number of Newton corrections
Modit number of Newton corrector steps with Jacobian reevaluation

D. Computing derivatives

Dhcrv increment for numerical evaluation of Jacobian matrix of the system defining a curve
Dhjac increment for numerical evaluation of Jacobian matrix of the right hand sides of ODE system with respect to phase variables

E. Evaluating determining system

Iprsng ordering number(s) of a non-generic parameter(s) for the continuation of nontransversal curve types

It indicates active parameters (by their ordering numbers in the current list of active parameters) which are considered as non-generic (nontransversal) parameters for singularity under consideration. If one has two such parameters, then **Iprsng** is a two-digit number each digit of which is interpreted as the ordering number above.

The following list shows recommended parameter settings used as defaults, along with their range allowed:

H0crv	= 0.1	(**H0crv** $\neq 0$)
Hmxcrv	= 1.0	(**Hmxcrv** $>\mid$ **H0crv** $\mid$)
Angcrv	= 10.0	($0 <$ **Angcrv** < 90)
Dhcrv	= 1.0e-7	($\mid$ **Dhcrv** $\mid > 0$)
Dhjac	= 1.0e-7	($\mid$ **Dhjac** $\mid > 0$)
Maxit	= 7	(**Maxit** > 0)
Modit	= 2	($0 \leq$ **Modit** $\leq$ **Maxit**)
Epscrv	= 1.0e-4	(**Epscrv** > 0)
Epscrs	= 1.0e-3	(**Epscrs** ≥ 0)
Epszer	= 1.0e-3	(**Epszer** ≥ 0)
Epsext	= 1.0e-3	(**Epsext** ≥ 0)
Iprsng	= 1	(**Iprsng** = 1,2,3,12,13,14,23,24,34)
Algcrv	= 2	(**Algcrv** $= \pm 1, \pm 2, \pm 3$)

7.2.3 Standard output

The standard output of the continuation code has a form of a sequence of regular and special points ordered along a curve. Each point may have additional attributes (values and/or texts). Such attributes are in particular values of user-defined functions on a curve, values of test functions, eigenvalues, type of a special point (including possibly some related values serving to characterize it more completely), arclenght, current step size, *etc.*

In **LOCBIF**, the coordinates of a current point and other "standard" numerical values are available via the Value Window, while texts and some additional numerical values are interpreted as messages and forwarded to the Message Window (see Section 5 for discussion of interface, and Section 6 for messages related to special points).

Graphic output goes in parallel with the numerical and the text output. This proceeds in the same point-by-point fashion: each newly computed point appears on the Graphics Window. For drawing curves, points may be sequantially connected by line since they are ordered already along the curve. Still, the user has the choice to use "solid" or "dotted" mode (see section 5).

Ordering points needs some remarks. This addresses also to the actual procedure of their computations. The continuation code proceeds in such a way that it first computes a next regular point $y^{(k)}$ and only then starts searching for special points on a curve's segment between the last and the

preceeding regular points, $y^{(k)}$ and $y^{(k-1)}$. In particular this means that the output of the last point is delayed until all intermediate special points will be computed. After this is done, the last regular and all special points found on the segment are ordered using the leading variable as a local parameter. And only then the output actually starts. This makes clear that several newly computed (ordered) points may appear more or less at once, sequentially though. The latest in this sequence is the last regular point $y^{(k)}$.

7.2.4 Control of the continuation flow

The standard output from the continuation code produces the most massive data which describe the curve itself. Above this, the program sends so called control messages. They report about that how the continuation code is proceeding along in time. The most important part of this information concerns the program interruptions which may arise due to several reasons.

After the curve computation is invoked, the message

"The first point"

says that the first point on a curve is found successfully. This is a remarkable moment since only after that one can actually start curve tracing.

Alternatively, one of the following messages appears:

(1) **"Incorrect number of active parameters"**
(2) **"Error in initial data"**
(3) **"Undefined function values at the initial point"**
(4) **"Cannot find Jacobian matrix at the in initial point"**
(5) **"Cannot find tangent vector at the initial point"**
(6) **"Cannot find the first point"**
(7) **"Cannot find tangent vector at the first point"**

They report about failures which occur at the initialization stage. In any case, the computations are terminated.

Message (1) means the number of parameters activated by the user doesn't satisfy the current curve type (see Section 6.1).

Message (2) indicates possible inconsistency in the continuation parameters settings. A particular though typical mistake of this sort is making an initial step size larger than maximal.

Message (3) is much more serious. It means the determining system for a curve type specified by the user, can not be evaluated at the initial point.

In such a case, it is advisable to recall constraints related to a domain of definition of the determining system. Indeed, the appearance of the message means that the point does not lie in this domain. Remark that the constraints may originate either from the RHS program or from particular algorithms for evaluating bifurcation functions. In practice, it is recommended to examine first the ODE system specifications, the curve type chosen and the coordinates of initial point.

Messages (4) and (5) are both related to the linearization matrix of the determining system at the initial point. The message (4) may have the same reason as (3) except that a point at which a "non-evaluating" situation arises is not the initial one but close to it (due to numerical differentiation). Once the linearization matrix is computed successfully, the program searches for its null (tangent) vector, and if it fails, the message (5) appears. Three typical reasons may be relevant to the case: (a) bad initial point, (b) the ODE system is non-generic (in particular, symmetry or first integral could be involved), and (c) the linearization matrix itself is not reliable (this reveals substantial errors in numerical differentiation; the related increment should be checked anyway).

Message (6) indicates the most typical problem: the program fails to find the first point on the curve. Searching the first points involves some iterations which use the initial point given by the user for the initial guess. Exceeding a maximal number of iterations allowed on this stage (typically 10), is a common reason for this error. It is worthwile to stress once more the importance of the proper choice of the initial point. However, revision of all data (initial point, curve type, system specifications, computational parameters) might be needed in difficult cases, especially when starting the analysis of a new system.

The last message (7) reports that after the first curve has been found successfully, the programs fails to compute a tangent vector to the curve at this point. Hence the continuation can not be started. This failure indicates a degeneracy of the system under consideration or problems with computation of the linearization matrix (see above).

Assume now that the first point has been found, and then the continuation procedure has been started. The question we address now is completing or terminating computations. Basically, there are three ways the computations could be finalized:

a) The computations are interrupted by the user. This can be done

explicitly, by pressing the corresponding key (see Section 5), or implicitly, by means of the **ABORT** statement in RHS specifications (see section 4).

b) The computations complete since the specified number **Maxnpt** of continuation points is computed (see Section 5), or the curve turns out to close on itself (*i.e.* it comes back to the first point). The first and second cases respectively are reported as

"Last point in buffer"
"Closed curve"

c) The computations are terminated because of the program can not find the next point on the curve. Recall that the step size is halved after each unsuccessful step. When the step size becomes smaller than the minimally allowed value, the message appears

"Current step size is too small"

and the computations get terminated.

Three typical reasons for this failure are: approaching the curve end point, approaching singular or nearly singular point on a curve, accuracy or step size control problems. It might be expected that the program fails near an end point. Indeed, typically end point lies on a boundary of a domain of definition of the determining system for the curve. Intermediate computations (at predictor or corrector steps) may produce a point which lies outside this domain, and each time this occurs, the program reduces the step size.

Another possibility is that continuation points approach a singular point on a curve such that the branch can not be extended smoothly through this point. This is the case when a geometric singularity occurs. It is also possible that because of large curvature (which may be interpreted as "near singularity") the curve can not be continued further. The parameters controlling maximal curvature allowed and the accuracy of computations should be examined in the last case.

If two reasons above do not provide a key to the problem, it might make sense to consider more thoroughly the current settings for continuation parameters. Attention should be paid first to the tolerance, the maximal number of corrector iterations and the increment for numerical differentiation. Note that parameter settings which are well suited for one part of a curve, may become unsatisfactory or even wrong for another part! A "quality" of

continuation parameters may decrease gradually while the computations proceed. This is a hidden process, which create a feeling that the computations interrupt "suddenly".

We should outline that the proper handling problems causing program interruptions reveals a great deal of practical experience.

If one selects nonzero value of service parameter **Messng** (see Section 5) then more details about the continuation process become available. In this case each corrector iteration is reported (namely, the ordering number, accuracy achieved and a norm of the determining system functions at the current corrector point. Moreover, the user is informed about accepting or rejecting a continuation step and a value of the step size chosen.

7.3 Orbit computation

7.3.1 Conventions

Orbit computation is organized in **LOCBIF** similarly to tracing curve defined implicitly by means of determining systems. The evident difference is in the numerical algorithms used in both cases. For orbit computation we mostly use standard integration routines RADAU5 and DOPRI5 from (Hairer, *et al.*, 1987; Hairer and Wanner, 1991).

The integration proceeds from $t = 0$ without limitation of time interval in forward or backward directions, which means forward and backward time respectively. The integration may be terminated by the user explicitly, by pressing the corresponding key, or implicitly, via RHS specifications (see Section 7.2). If not interrupted by the user, the integration will be completed when a given number of points on the orbit (**Maxnpt**) is computed. The integration may be also terminated if the integration routine cannot find the next point. Then the message appears:

"Cannot continue integration".

The evident possible reason for this failure is that given accuracy cannot be achieved. For more details, see references cited above.

Points obtained by integration are processed in the same way as continuation points. For orbits, a notion of special points is different, however. These are just points $t = j*Tmax$, where j is integer and $Tmax$ is a time interval defined by the user (see below). As it has been already explained

above, there is no limit on total time interval for integration except maximal number of points which is also defined by the user. Continuing orbit computation after a pause at special point means continuing integration for the same time interval *Tmax*.

7.3.2 Orbit parameters

Orbit parameters may be displayed and modified through the Orbit Window (see Section 5). These parameters have the following meaning:

Itmap iteration number
Tint length of the sample time interval for integration

Time interval *Tmax* between standard pauses in orbit computation is *Tmax*=**Itmap*Tint**.The Initial Value Problem for ODE system is assumed to be solved initially on interval $[0, Tmax]$ or on interval $[-Tmax, 0]$ (backward in time) and then continued in the same direction if necessary. There is a special convention for assigning time intervals linked with π: if a value $6.28\ldots$ is assigned to **Tint**, the program automatically reassigns it to the value 2π up to a machine accuracy.

H0int initial step size for integration
Hmxint maximal step size for integration
Dhint increment for numerical evaluation of Jacobian matrix of the RHS used by a stiff ODEs solver
Epsint absolute step tolerance for integration
Epsrel relative step tolerance for integration
Solver type of ODE solver
1,2,3 - non-stiff fifths-order solver (DOPRI5)
4 - non-stiff forth-order solver (RKGS-type)
-1 - stiff fifths-order solver (RADAU5)
Isec integer parameter (not used in this version, see Appendix C)
Irhs parameter used for integration of differential-algebraic systems

The last parameter indicates the number of algebraic equations in the system. Positive value of **Irhs** means that first **Irhs** equations are algebraic. Negative **Irhs** means that last |**Irhs**| equations are algebraic. The parameter is meaningful only for RADAU5 solver (*i.e.* if **Solver** = -1).

Iorbit orbit type (not used in this version, see Appendix C)

The following values of orbit parameters are recommended; they are used as defaults:

Itmap = 1
Tint = 10.0
H0int = 0.1
Hmxint = 1.0
Dhint = 1.0e-7
Epsint = 1.0e-7
Epsrel = 1.0e-9
Solver = 1
Isec = 1
Irhs = 0
Iorbit = 1

7.4 Problems and hints

7.4.1 Initial point

It has been stressed repeatedly above that the choice of the initial point for continuation is a rather delicate problem for which no general and evident solution exists. In this respect, using a continuation strategy as explained in Section 6) is one useful idea to help in this choice. Another idea is to provide an initialization procedure whenever possible.

This procedure attached to the RHS specifications can contain calculations required to set up the initial point. For example, if one has an explicit formula allowing to compute coordinates of an equilibrium point provided that parameter values are given (or vice versa), it may be included in the RHS specifications as an **INIT** procedure. See description of **INIT** statement in Section 4.The actual execution of this procedure is controlled by the user. Namely, this procedure may be invoked manually, by pressing the corresponding key, or it may be executed each time the computations start. In the latter case the appropriate value of service parameter **Init** must be set. See Section 5 for details.

7.4.2 Using homotopy

The homotopy method provides another possibility to search for the initial point. This method suggests "to connect" by a smooth path the unknown point that we are searching for and some given point being a solution of another "similar" problem. This idea may be implemented manually, by creating a relevant artificial continuation problem and by using the **LOCBIF** continuation code to compute the connecting path. However, this may be arranged also by using certain **LOCBIF** conventions, as explained below.

Let $y^{(0)}$ be a given initial point. Set the service parameter **Init** to -1. This means that function $G(7.1)$ will be replaced by the function

$$\tilde{G}(y) = \delta \cdot G(y) + (1 - \delta) \cdot (G(y) - G(y^{(0)})) \tag{7.2}$$

where δ is a so called homotopy parameter. For $\delta = 0$ the point $y^{(0)}$ is clearly a solution of the modified continuation problem. If, by varying δ, and tracing a solution to (7.2), one can achieve the target value $\delta = 1$, this provides an initial point for the original problem. To implement this idea, include δ in the list of active parameters (see below). Make sure that you have the proper number of active parameters corresponding to the selected curve. Start the computation of the chosen curve and try to reach the target $\delta = 1$. Computations in both directions might be needed. Remark that even when this method succeeds, it may provide a wrong solution, in the sense that the target point might lie on another branch than expected.

To have the homotopy parameter available for the required manipulations (setting values, activating/desactivating *etc.*), it has to be included *formally* in the list of system parameters under the *fixed* name **HOMPAR**. However, it is not required and even not recommended to involve this parameter into the system specifications. Provided the parameter **Init** has the appropriate value which enables homotopy continuation, the transformation from (7.1) to (7.2) proceeds automatically, so that the continuation code will use the modified function $\tilde{G}(y)$, instead of $G(y)$. One should keep in mind however, that the above modification makes the meaning of the computed curve different from the original one (although formally this name remains the same). This requires the careful interpretation of results.

By **LOCBIF** conventions, all parameters declared in system specifications may be monitored by means of the user interface (actually some of them may not be used in the specifications). This allows the user to arrange

computations in which the homotopy parameter is involved explicitly and even to use it as a plotting coordinate if needed.

As a side effect, the modification (7.2) provides the possibility to compute "isolines".

7.4.3 Isolines

Assume that modification (7.2) is enabled as explained before, but now the homotopy parameter δ is non-active. Set, for example, $\delta = 0$. Then, instead of computing the original curve with determining system $G(y) = 0$, another curve will be computed, which provides solutions to the equation

$$G(y) - G(y^{(0)}) = 0 \tag{7.3}$$

Provided $G(y^{(0)})$ is small enough, this new curve should be close to the previous one. Notice that the initial point $y^{(0)}$ lies exactly on this new curve.

One can suggest different ways how to exploit this possibility. A trivial but useful application is the computation and the plotting of isolines in the plane. We will mention also two other approaches, less evident.

Sometimes the initial point is assumed to be good enough, but nevertheless the program fails when seeks for the first point. Then it might be recommended to compute one or several curves of a family of curves given by (7.3), where $y^{(0)}$ plays the role of the parameterization parameter. Since $y^{(0)}$ lies exactly on the curve it defines, all difficulties are shifted from the computation of the first point on the curve to the computation of another curve. Experiments of this kind may help to visualize the geometry of the original curve and to understand why the search of the initial point fails. The failures may be related in particular to the fact that the underlying determining system is degenerate (nearly or exactly) along the curve or just near the initial point.

Another application which uses to the same modified determining system (7.3), is inspired by numerical problems which may be caused by self-crossing points. One could be interested in the analysis of the branching behaviour of solutions to the determining system near a self-crossing point, or one may like to switch from one branch to another. Given an initial point in the neighbourhood of the self-crossing point (but not exactly on the branches), one can start the continuation from this point and use the determining system (7.3). In most cases, this simple device unfolds the original system (7.1) such

that the self-crossing point disappears. In particular, if symmetry is the key for the branching behaviour, using (7.3) with the proper choice of the initial point enables us to destroy the symmetry.

The branch started from the chosen point and defined by (7.3) with small $G(y^{(0)})$, will first follow one branch of the original problem, and after passing close to the "ghost" of the self-crossing point, it "switches" smoothly to the other branch. With several different choices of $y^{(0)}$, one can expect to reveal all branches and choose those of them which appear of particular interest. Disabling the usage of (7.3) brings us back to the original problem and to the computation of "standard" curves, but probably with a good enough starting point for the new branch. This trick may be particularly recommended for handling self-crossing points on a Hopf bifurcation curve.

7.4.4 Testing the current point

If the curvature test causes problems, *e.g.* leads to substantial reduction of the step size, the user can suppress this test by setting **Algcrv** = -i where i=1,2,3 has the same meaning as before (see Section 7.2.2).

Part III
APPLICATIONS

8 Example 1: Normal form for cusp bifurcation

In this section we will illustrate how to use **LOCBIF** for existence and stability analysis of equilibria in a simple dynamical system.

8.1 System description

Consider the following simple ODE with one phase variable x and two parameters α and β:

$$\dot{x} = \alpha + \beta x - x^3 \tag{8.1}$$

The problem is to analyze the number of equilibria in (8.1) for various parameter values (α, β) and to determine their stability.

Equation (8.1) is a normal form for cusp bifurcation (cf. Arnold, 1982).

8.2 System specification

Invoke **LOCBIF**, press a key to leave the Initial screen, type a name for the system, for example **CUBIC**, and press Enter. The Equation Window appears with dummy right hand sides. Type in the system specification, for example:

PHASE X
PAR ALPHA,BETA
X′ = ALPHA + BETA * X − X^3

and erase all the rest of the dummy specification.

Press the Alt+X keys. You will leave the Equation Window and return to the Archive Window. The archive now contains the system **CUBIC**.

8.3 Setting of computational parameters

Select the **CUBIC** system and press the Enter key. After a short delay you will see the main **LOCBIF** screen. Select item **Options** and press Enter, then select option **Continuation Parameters** and press Enter again. The Continuation Parameter Window appears. Type in new parameter values: **H0crv** = 0.01, **Hmxcrv** = 0.05 (see Figure 15).

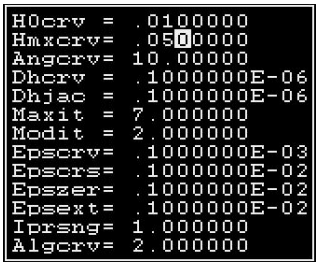

Figure 15: Continuation Parameter Window after input of new values of **H0crv** and **Hmxcrv**

Press Esc to leave the window and Esc again to return to the Command Line.

8.4 Equilibrium continuation

Let us find an equilibrium for (8.1) for several fixed values of β and compute its dependence upon parameter α. Make sure that curve type is **Equilibrium**. Enter into the Value Window by selecting and activating item **Values** in the Command Line. Move the cursor to parameter **ALPHA** and press the Alt+F keys. Now **ALPHA** is activated. Input the following initial values: **X** = 1.0, **ALPHA** = 1.0, **BETA** = -1.0 (Figure 16).

Leave the Value Window by pressing the Esc key and activate the **Options** item (you can do this directly by pressing F2 key inside the Value Window). Select option **Axis Parameters** inside the Option Window (you can also do this immediately by pressing Alt+G). The Axis Parameter Window will appear. Select **ALPHA** as abscissa and **X** as ordinata (press Enter/Ctrl+Enter to list variables forward/backward). Type in minimal and maximal values of **ALPHA** and **X**:

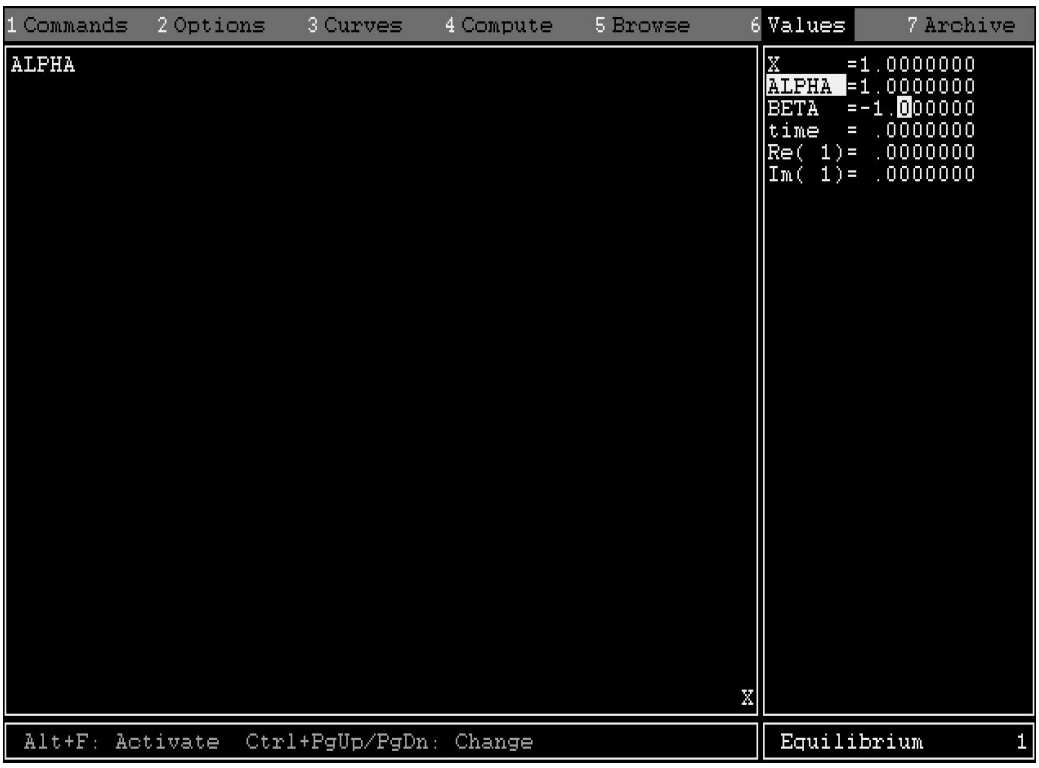

Figure 16: New initial values. Notice, parameter ALPHA is activated

$$\mathbf{ALPHA}_{min} = -1.0,\ \mathbf{ALPHA}_{max} = 1.0, \mathbf{X}_{min} = -2.0, \mathbf{X}_{max} = 2.0.$$

(see Figure 17).

Press Esc twice to leave the window and return to the main **LOCBIF** screen. Select and activate item **Commands** and press Enter to refresh the axis names (you can also do it directly by pressing Alt+D). Now you are ready to start the equilibrium continuation.

Select item **Compute** and press the Ctrl+Enter keys to start computations backward. The first point will appear on the right boundary of the Graphics Window. Press the same keys to continue computations. A monotonic equilibrium curve will be computed. When the curve cross the left boundary, press the Esc key to terminate calculations. The corresponding message will be displayed in the Message Window.

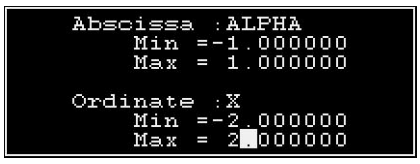

Figure 17: Visibility limits

Change the **BETA** value to **BETA** = 0 through the Value Window and repeat the calculations (you can use the same initial point as before). The equilibrium curve shape will change.

Repeat calculations once more for **BETA** = 1. Several messages appear in the Message Window which report about special points found on the curve. After each message press Ctrl+Enter to continue the curve computation. Each of the fold points on the curve is reported as an extremum with respect to **ALPHA**, and as non-generic equilibrium with eigenvalue zero. By monitoring eigenvalues the stability change may be detected at these points. Each of them may be used as an initial point to start the continuation of **Fold** curve. The equilibrium curve has S-shape. The resulting Graphics Window is presented in Figure 18.

This computation reveals that for **BETA**=1.0, the interval [-1, 1] of parameter **ALPHA** is divided by critical parameter values into three subintervals. For the parameter values within two of them equation (8.1) has one stable equilibrium, while for the parameter values lying between critical values there are three equilibria: two stable and one unstable.

8.5 Bifurcation analysis

The critical parameter values correspond to bifurcations in which two equilibria appear or disappear. Let us compute a bifurcation curve on the (**ALPHA**, **BETA**)-plane on which this bifurcation takes place. More precisely, we compute **Fold** curve in (**X, ALPHA, BETA**) active phase-parameter space. The locus of critical points is a projection of this curve onto (**ALPHA**, **BETA**)-plane.

Take one of the special points computed as the initial point for **Fold** curve continuation. For this, clean the Graphics Window and initialize browsing backward by selecting **Browse** item and pressing the Ctrl+Enter keys. To

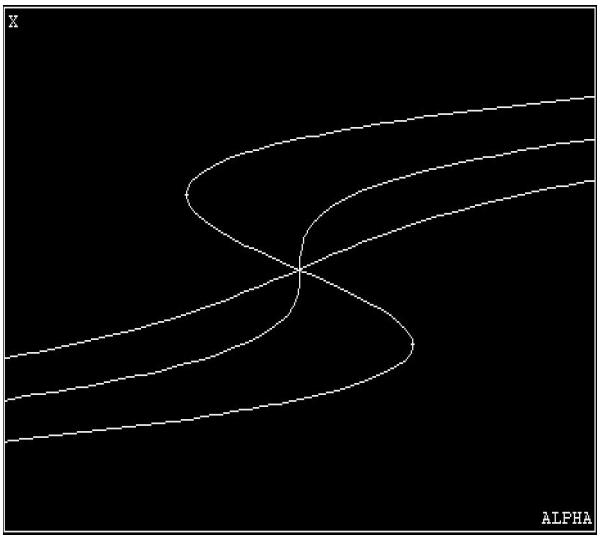

Figure 18: Equilibrium curve for BETA=-1.0 (monotonous), BETA=0.0 (with a vertical tangency) and BETA=1.0 (S-shaped)

continue browsing, press the same keys after each message. When the message

"Zero eigenvalue: a = -3.46405"

appears, press the Ins key. The browsing is terminated and an initial point for **Fold** curve is now selected.

Activate parameter **BETA** through the Value Window. Also invoke the Axis Parameter Window through the Option Window and select parameter **BETA** as the ordinate. Type in

$$\mathbf{ALPHA}_{min} = -2.0,\ \mathbf{ALPHA}_{max} = 2.0$$

and return to the Command Line by pressing Esc twice. Clean the Graphics Window and see the new axis names. Finally, change the curve type by selecting type **Fold** inside the Curve Select Window (see Figure 19).

Now you are ready to start **Fold** curve computations. Press F4/Ctrl+F4 to initialize computations forward/backward and to continue after a message. The computed curve will have another special point: a **Cusp** bifurcation point. At this point, the **Fold** curve projection onto parameter plane has a geometric singularity of cusp type (see Figure 20). The (**ALPHA**, **BETA**)

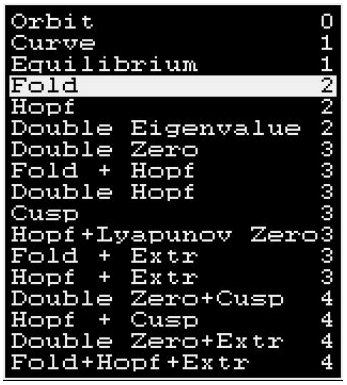

Figure 19: Curve Select Window with **Fold** curve selected

plane is divided into two regions by the **Fold** curve projection. In the first (larger) region, equation (8.1) has only one and stable equilibrium point, while in the second (smaller region) it has three equilibria, one unstable and two stable.

It is interesting to notice that bifurcation curve **Fold** in the corresponding active phase-parameter space with coordinates (**X**, **ALPHA**, **BETA**) is smooth and has no geometric singularities. To see this, browse the computed curve in the (**BETA**,**X**) coordinates. Select **BETA** as the abscissa and **X** as the ordinate and check if

$$\mathbf{BETA}_{min} = -2.0,\ \mathbf{BETA}_{max} = 2.0, \mathbf{X}_{min} = -2.0, \mathbf{X}_{max} = 2.0$$

through the Axis Parameter Window.

Clean the Graphics Window and browse the curve forward and backward by pressing the F5 and the Ctrl+F5 keys. You will see a simple parabola (Figure 21).

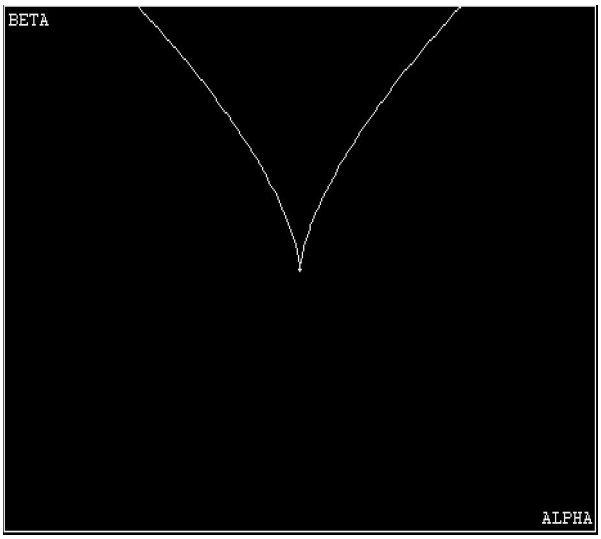

Figure 20: Cusp singularity

9 Example 2: Ecological modelling

In this section we will illustrate how to use **LOCBIF** for existence, stability and bifurcation analysis of equilibria in a model with two phase variables.

9.1 System description

Consider a system of differential equations with two phase variables (x, y) depending upon four parameters $(\alpha, \beta, \gamma, \delta)$(see Bazykin, 1985):

$$\begin{aligned} \dot{x} &= x - xy/(1+\alpha x) - \beta x^2 \\ \dot{y} &= -\gamma y + xy/(1+\alpha x) - \delta y^2 \end{aligned} \tag{9.1}$$

Here x and y are scaled prey and predator population densities respectively. Parameter α determines the saturation of predator, β and δ are the prey and predator competition rates and γ is the predator natural mortality rate. For $\alpha = \beta = \delta = 0$ we obtain the original Lotka-Volterra model. We shall study equilibria of (9.1) with the help of **LOCBIF.**

Input system (9.1) into **LOCBIF**, for example, in the following form:

PHASE X,Y

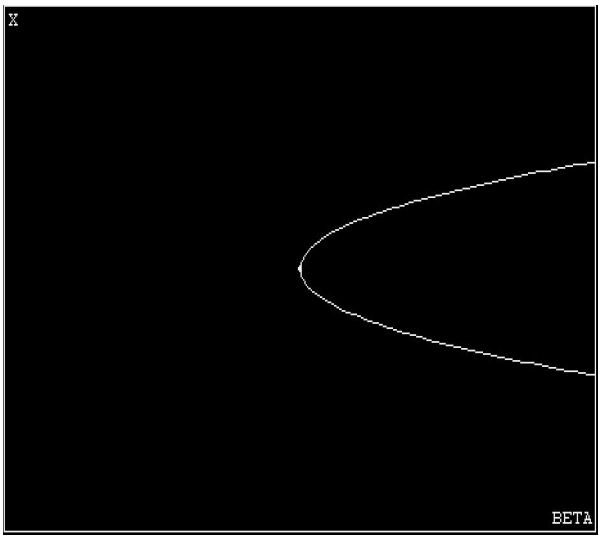

Figure 21: Parabola in (BETA,X)-plane

PAR ALPHA,BETA,GAMMA,DELTA
FUN F,F1,F2
F=X*Y/(1.0+ ALPHA*X)
F1=X-F- BETA*X^2
F2=-GAMMA*Y+F- DELTA*Y^2
$\mathbf{X}' = \mathbf{F1}$
$\mathbf{Y}' = \mathbf{F2}$

9.2 Finding of an equiliblum by integration

Set **ALPHA** = 0.3, **BETA** = 0.01, **GAMMA** = 1.0 and **DELTA** = 0.5. Note that we have no *apriori* information on equilibrium coordinates. In such a case we may try to find an equilibrium of (9.1) by integration from arbitrary chosen initial conditions, *e.g.* $\mathbf{X} = 100.0, \mathbf{Y} = 10.0$. Only a stable equilibrium may be approached in this way.

Input the initial values into the Value Window, check all parameters are inactive. Set the curve type **Orbit**. Select **X** as the abscissa and **Y** as ordinate in the Axis Parameter Window and type in limits

$$\mathbf{X}_{min} = 0.0, \mathbf{X}_{max} = 100.0, \mathbf{Y}_{min} = 0.0, \mathbf{Y}_{max} = 10.0.$$

If you have a color monitor, select the desired background and curve colors through the Option Window. Being inside the same Option Window, set the **No**-pause mode.

Compute the orbit forward (F4). You may see that the trajectory tends to a limit position which is an equilibrium point. You can press the Space key to pause and notice that the **F1**, **F2** function absolute values become less and less (they are equal to zeros in equilibrium). Press F4 to continue computations.

When the trajectory converges to the equilibrium, pause by hitting the Space key and press the Ins key to accept the current point as an initial point for equilibrium continuation. You can see that the point has coordinates: $\mathbf{X} = 82.97\ldots, \mathbf{Y} = 4.409\ldots$. These values were used in Section 2.

9.3 Equilibrium continuation

We have found an equilibrium. Let's compute the dependence of the equilibrium upon parameter δ and analyse its stability.

Activate parameter **DELTA** through the Value Window and set the curve type to be **Equilibrium** through the Curve Select Window. Select **DELTA** to be plotted as the abscissa and input

$$\mathbf{DELTA}_{min} = 0.0, \mathbf{DELTA}_{max} = 1.0.$$

inside the Axis Parameter Window.

Clean the Graphics Window and set the pause mode to **Special** through the Option Window. Now you are ready to start computations.

Press F4 to start computations forward. When the first point is found, it will be displyed in the Value and Graphics Windows. The equilibrium is stable. Press F4 for continuation and Esc to terminate computations when the curve leaves the Graphics Window.

Use Ctrl+F4 keys to compute the curve in the opposite direction. Terminate computations when the curve leavs the window by pressing the Ecs key. You should have a screen like presented in Figure 22.

There is an interval of **DELTA** within which the system (9.1) has three equilibria: two stable and one unstable. The limit points of the interval,

$$\mathbf{DELTA}_1 = 0.263\ldots, \mathbf{DELTA}_2 = 0.436\ldots,$$

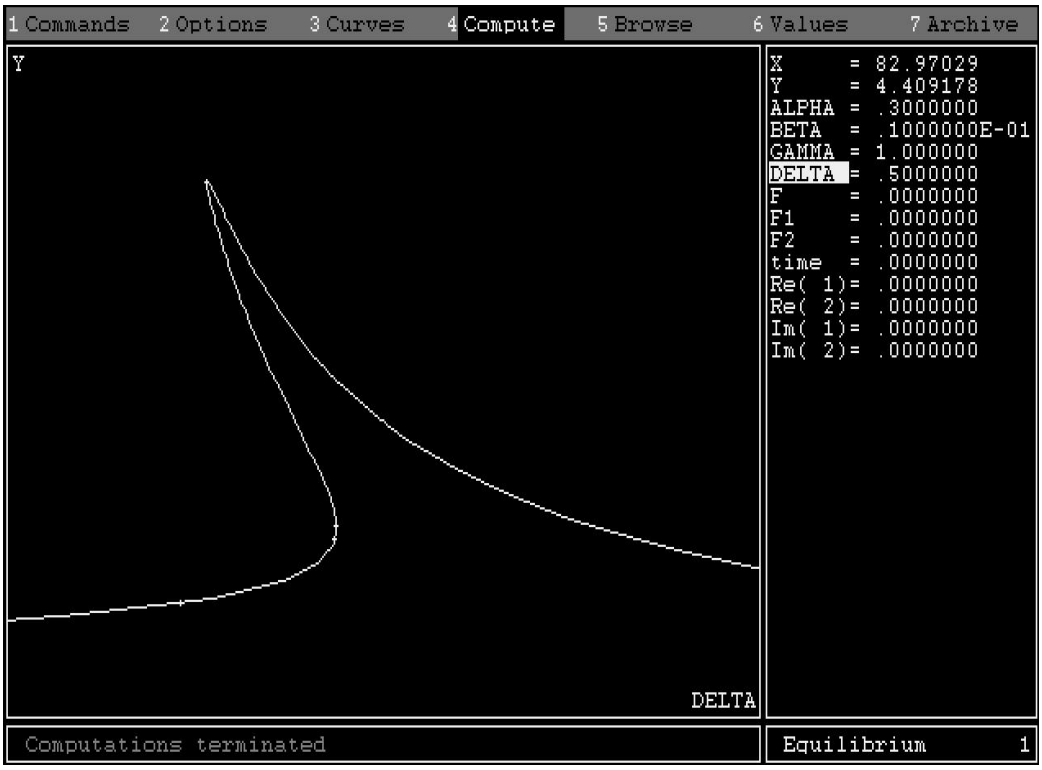

Figure 22: Equilibrium curve

are tangent bifurcation points corresponding to equilibria appearance or disappearance. These points may be used as initial points for **Fold** bifurcation curve continuation.

9.4 Fold curve continuation

Clean the Graphics Window by pressing the Alt+D keys and start backward browsing of the computed equilibrium curve by pressing the Ctrl+F5 keys. Press the same keys until the message

"Zero eigenvalue : a = -.918... E-02"

appears in the Message Window. Terminate browsing and take the last point as the initial point for **Fold** curve continuation by pressing the Ins key.

Activate parameter **ALPHA** and select **Fold** as the curve type. Select

parameter **ALPHA** as the abscissa and parameter **DELTA** as the ordinate through the Axis Parameter Window and set

$$\mathbf{ALPHA}_{min} = 0.0,\ \mathbf{ALPHA}_{max} = 1.0,$$
$$\mathbf{DELTA}_{min} = 0.0,\ \mathbf{DELTA}_{max} = 3.0.$$

Clean the Graphics Window. Press F4 to start computations and to continue after each message. Note that the **Fold** curve is closed. Press Esc after the final message.

The projection of the **Fold** curve onto the (**ALPHA**,**DELTA**)-plane has two geometrically singular points of a **Cusp** type and two singular points where the **Neutrality** condition holds. Within a region bounded by the projection, system (9.1) has three equilibria: two stable and one unstable (saddle type). Outside the boundary system (10.1) has one stable equilibrum. You may erase the curve by pressing the Alt+D keys and browse it again by pressing the F5 key.

Let us store the computed curve on a disk. To invoke a Curve Archive Window press F7. Type a name for the stored curve, for example FOLD, and press Enter. The curve is stored on a disk and will appear in the archive list. To leave the Curve Archive Window, press Esc.

9.5 Hopf curve continuation

There were two **Neutrality** points on the **Fold** curve. For corresponding parameter values, system (9.1) has an equilibrium with two zero eigenvalues. Each of these points may be used as the initial point for a **Hopf** curve continuation. Let us compute this curve.

Clean the Graphics Window. Browse the **Fold** curve (F5) and select the first **Neutrality** point as the initial point using the Ins key. Clean the window again and select **Hopf** as the curve type.

Compute the **Hopf** curve forward and backward using the F4 and Ctrl+F4 keys until the curve leaves the Graphics Window. You can see that the **Hopf** curve goes through both the **Neutrality** points. Between the points, it corresponds to the existence of an equilibrium with two real eigenvalues $\lambda_1 = -\lambda_2$. It is not a bifurcation branch. The other parts of **Hopf** curve correspond to the appearence of a limit cycle from the equilibrium with a pair of pure imaginary eigenvalues $\lambda_{1,2} = \pm i\omega$(Hopf bifurcation). On the right Hopf bifurcation branch there is an additional singular point: **Zero Lyapunov value**.

At this point the direction of appearence and the stability of the limit cycle changes.

Store the computed curve into the archive, for example, under the name HOPF.

Note:

There are some other (global) bifurcation curves which cannot be computed by the equilibrium point version of **LOCBIF**. In system (9.1) there are four curves of this kind: three homoclinic orbit curves and one double cycle curve. Althougth, we have found initial points for some of these curves. Two homoclinic bifurcation curves originate at **Double Zero** points. The double cycle (fold) curve starts from a **Zero Lyapunov value** point.

9.6 Joined picture

Let us plot on the screen a final picture with two computed curves: **Fold** and **Hopf**. These curves have already been stored in the archive.

Invoke the Curve Archive Window by the F7 key, move the highlight to the FOLD curve name and press Enter. The message

"Loading..."

will appear and after a short delay the curve becomes available for browsing. Note that the Value Window is updated and the relevant curve type (**Fold**) is displayed in the Curve Window. Invoke the Axis Parameter Window by pressing Alt+G and adjust the limits of visibility by pressing the Alt+L keys. After the message

"Limits adjusted"

leave the window by pressing the Esc key, clean the Graphics Window by hitting Alt+D and press F5 to browse the curve forward.

Invoke the Curve Archive Window again and load the HOPF curve by the same procedure. Browse the curve forward and backward.

Now you have a screen like that shown in Figure 23 which you may copy to your printer using a standard software.

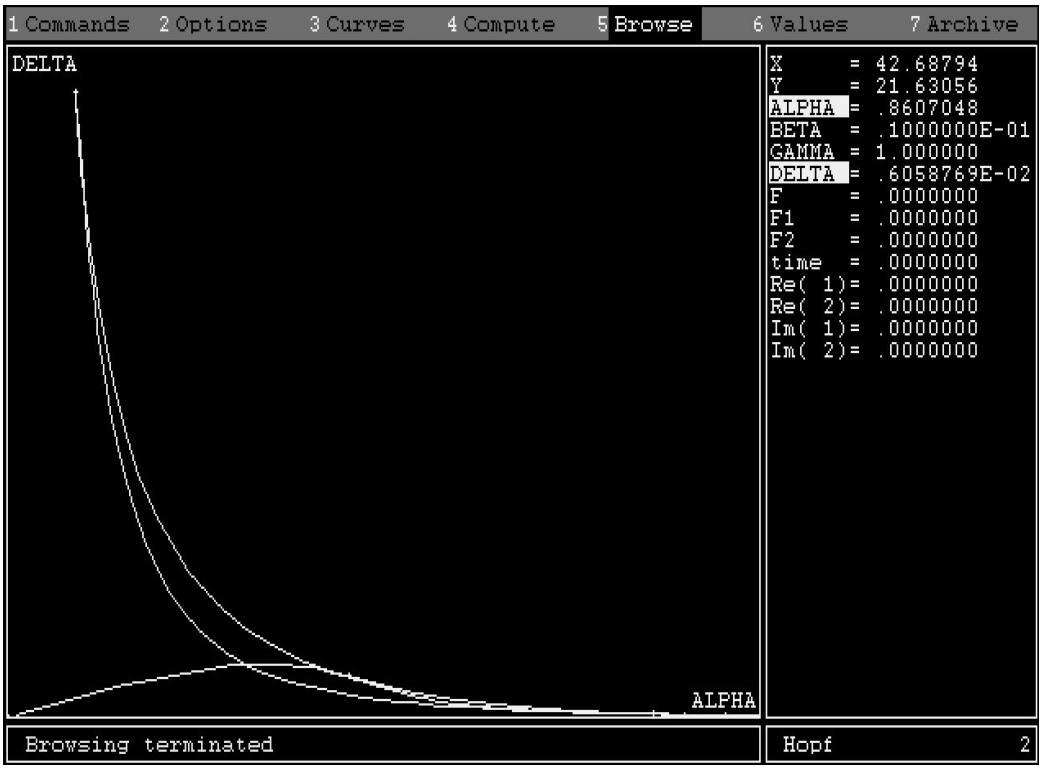

Figure 23: Fold and Hopf bifurcation curves

10 Example 3: Chemical kinetic model

In this section we will illustrate **LOCBIF** usage with a chemical kinetic model.

10.1 System description

Consider the following ODE system with three phase variables and seven parameters (Bykov *et al.*, 1978, Khibnik *et al.*, 1987):

$$\begin{aligned} \dot{x} &= 2k_1z^2 - 2k_{-1}x^2 - k_3xy \\ \dot{y} &= k_2z - k_{-2}y - k_3xy \\ \dot{s} &= k_4z - k_{-4}s \end{aligned} \tag{10.1}$$

where $z = 1 - x - y - s$. The model describes CO oxidation on platinum:

1) $O_2 + 2Pt \longleftrightarrow 2PtO$
2) $CO + Pt \longleftrightarrow PtCO$
3) $PtCO + PtO \longleftrightarrow 2Pt + CO_2$
4) $CO + Pt \longrightarrow (PtCO)$

In system (10.1) z, x, y, s are (scaled) concentrations of Pt, PtO, PtCO and a nonreactable form (PtCO) respectively, while k_i stand for the corresponding reaction rate constants. Several equilibrium and global bifurcations were found analytically and numerically in the system, which has 23 topologically nonequivalent phase portraits. The analysis has shown that the chemical system can behave as either an oscillator or a trigger.

In this Section we will use the **LOCBIF** program to investigate equilibrium existence, stability and bifurcations in system (10.1) and compute relevant bifurcation curves.

Ratio $K = k_{-4}/k_4$ will be used in the result presentation. The parameters $k_1, k_2, k_3, k_4, k_{-1}, k_{-2}$ and K will be denoted as $Q1, Q2, Q3, Q4, Q5, Q6$ and K respectively.

Let us input equations (10.1) into **LOCBIF** in the following form:

PHASE X,Y,S
PAR Q1,Q2,Q3,Q4,Q5,Q6,K
FUN Z
Z=1-X-Y-S
$\mathbf{X' = 2 * Q1 * Z\hat{}2 - 2 * Q5 * X\hat{}2 - Q3 * X * Y}$
$\mathbf{Y' = Q2 * Z - Q6 * Y - Q3 * X * Y}$
$\mathbf{S' = Q4 * Z - K * Q4 * S}$

10.2 The results

Some results of the investigation are shown in Figure 24 and Figure 25 as they were produced by **LOCBIF**.

Figure 24 shows projections A_1, A_2 and A_3 of three **Equilibrium** curves onto the $(\mathbf{Q2}, \mathbf{X})$-plane for different values of $\mathbf{K}$:

$$\mathbf{K} = 0.3, 1.0, 3.0.$$

The values of the other parameters are fixed:

$$\mathbf{Q1} = 2.5, \mathbf{Q3} = 10, \mathbf{Q4} = 0.0675, \mathbf{Q5} = 1, \mathbf{Q6} = 0.1.$$

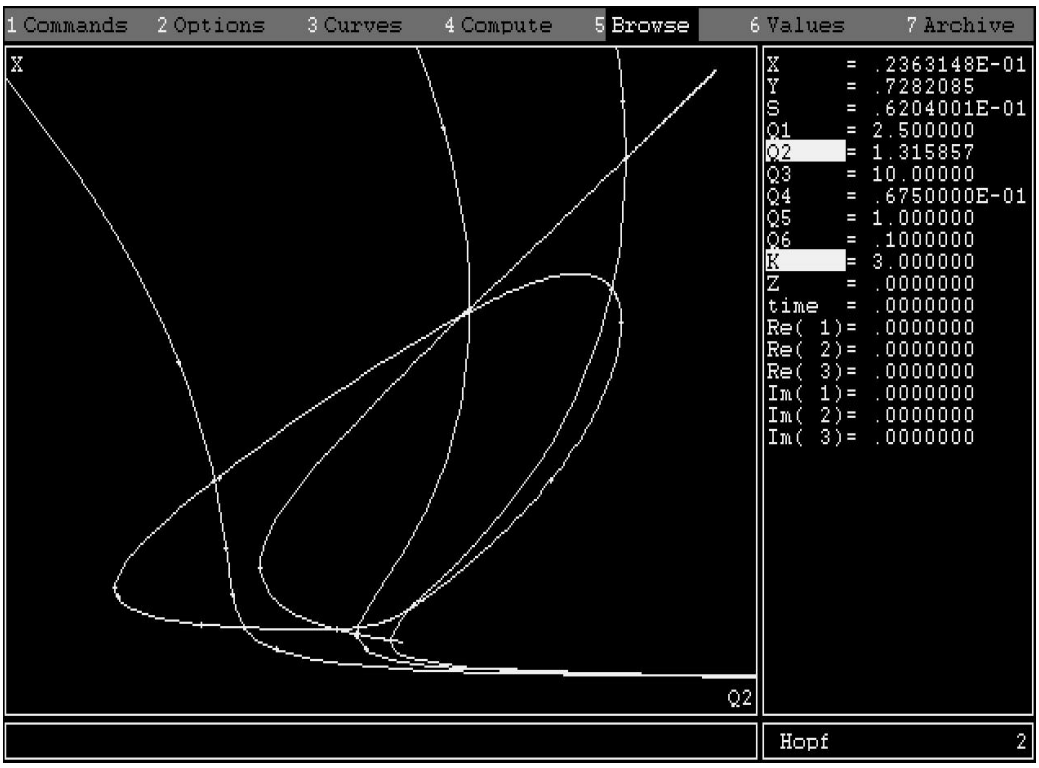

Figure 24: Equilibrium and bifurcation curves in model (10.1):A_1, A_2 and A_3 - **Equilibrium**; B - **Fold**; C - **Hopf**

Initial values of the phase variables and parameter $Q2$ for these equilibrium curves are

$$\mathbf{X}_0 = 0.00154, \mathbf{Y}_0 = 0.927, \mathbf{S}_0 = 0.0178, \mathbf{Q2}_0 = 2.0.$$

The limits of visibility are

$$\mathbf{Q2}_{min} = 0.5, \mathbf{Q2}_{max} = 2.0, \mathbf{X}_{min} = -0.01, \mathbf{X}_{max} = 0.2.$$

The projections B and C of the **Fold** and **Hopf** bifurcation curves respectively are also plotted in Figure 24; they correspond to active parameters **Q2** and **K**. The relevant singular points (**Zero eigenvalue** and **Neutral saddle**) on the **Equilibrium** curve A_3 may be chosen as initial for **Fold** and **Hopf** curve continuations. Computation of **Fold** curve was terminated at

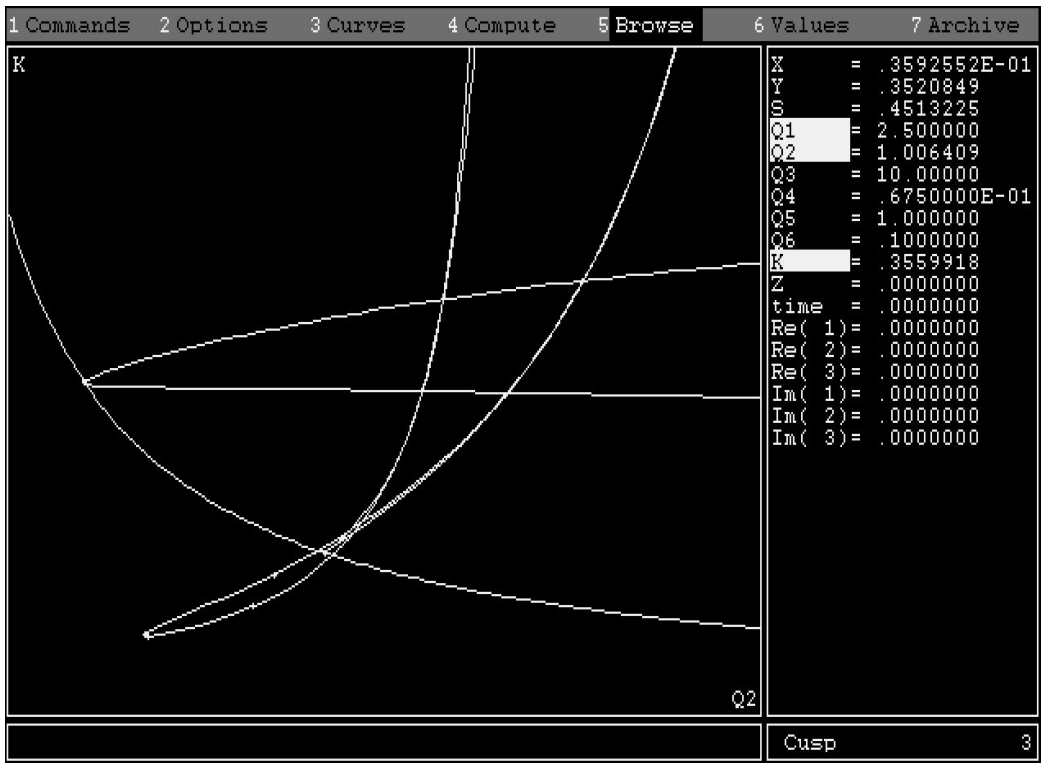

Figure 25: Bifurcation curves in model (10.1): B - **Fold**; C - **Hopf**; G - **Cusp**; E - **Double Zero**

some points corresponding to high values of K.

The **Hopf** bifurcation curve is closed; it consists of two connected parts corresponding to Hopf and neutral saddle cases. These parts are separated by two Bogdanov-Takens bifurcation points. The Hopf part contains also two degenerate Hopf bifurcation points.

The **Fold** bifurcation curve contains two Bogdanov-Takens bifurcation points mentioned before and one Cusp bifurcation point.

You will get corresponding messages while computing or browsing the curves. We recommend you store all the computed curves into the archive.

In Figure 25 several bifurcation curves are represented as projections onto the parameter plane $(\mathbf{Q2}, \mathbf{K})$. These are **Fold** and **Hopf** curves denoted by B and C, as well as **Cusp** and **Double Zero** bifurcation curves denoted by G and E respectively.

Limits of visibility for parameters **Q2** and **K** are

$$\mathbf{Q2}_{min} = 0.5, \mathbf{Q2}_{max} = 1.7, \mathbf{K}_{min} = -0.01, \mathbf{K}_{max} = 1.5.$$

To start computation of **Cusp** and **Double Zero** curves use relevant points on **Fold** and **Hopf** bifurcation curves and activate the additional parameter **Q1**.

Note that a degenerate Bogdanov-Takens bifurcation point has been located on the **Cusp** and **Double Zero** curves. It is marked by *I* in the figure.

Part IV
APPENDICES

A LOCBIF restrictions

LOCBIF allows you to study dynamical systems with no more than 10 phase variables and 10 parameters.

The name of a system studied cannot be longer than 8 characters.

The names of phase variables, parameters, functions, local and common variables in equation specifications cannot exceed 6 characters.

The names of curves cannot exceed 8 characters.

You cannot store more than 100 systems in the ODEs Archive and more than 100 curves in the Curve Archive for each system.

System specifications cannot be too complex (see next Appendix B).

B Error messages

B.1 RHS Editor error messages

"RHS file not found"

File RHS.DAT containing RHS specifications was not found (it may have been accidentally deleted by the user).

"Can't open output file"

"Can't write output file"

These two messages mean that there may be not enough space on the disk for the RHS.DAT file.

"Too many characters in file"

Number of characters in RHS specification exceeds 20 000.

"Too many lines in file"

Number of characters in RHS specification exceeds 500.

B.2 RHS Compiler error messages

"TIME redeclaration"

"PHASE redeclaration"

"PAR redeclaration"

"COMMON redeclaration"

"FUN redeclaration"

Any of above four messages indicates that the corresponding keyword appears more than once in the RHS specification.

"TIME declaration is empty"

"PHASE declaration is empty"

"PAR declaration is empty"

"FUN declaration is empty"

"COMMON declaration is empty"

Any of above four messages means that the corresponding keyword is not followed by a name.

"Redeclaration of identifier"

Each name must be declared only once in the RHS specification.

"PHASE declaration is absent"

"Too many equations (more than 10)"

"Too many parameters (more than 10)"

"Too many functions (more than 10)"

"Too many common variables (more than 10)"

"Too many local variables (more than 10)"

"Too many arguments (more than 10)"

"Function name is expected"

Number of function declared in **FUN** statement differs from number of defined functions.

"')' or ',' is expected"

"Identifier is expected"

"'=' is expected"

"Too complex RHS specification"

RHS Compiler can not process your equations due to internal limitations.

"Operator can not be recognized"

An operator is expected; the next symbol should be a variable name, **IF**, **WHILE**, **FOR** or {.

"Natural number is expected"

"']' is expected"

"Relation sign is expected (<, <=, >, >=, =)**"**

"Undeclared identifier is used"

"FOR cycle parameter must be a variable"

The parameter should be either **COMMON** or **VAR** variable.

"',' is expected"

"This variable can not be changed"

You must not change a phase variable or parameter values except inside the **INIT** statement.

"'[' is expected"

"' ' is expected"

"Phase name is expected"

"Identifier, number or '(' is expected"

"')' is expected"

"You should use neither '[' nor '('"

The variable is neither an array nor a function.

"'(' is expected"

"Too many elements in array"

"Open failure (TS.DAT)"

"Open failure (IR.DAT)"

"Write error (TS.DAT)"

"Write error (IR.DAT)"
"Close failure (TS.DAT)"
"Close failure (IR.DAT)"
"Function is defined twice"
"Invalid number of arguments in function call"
"Unnecessary character (delete up to EOF)"
"'{' is expected"
"INIT redefinition"

This message indicates that the corresponding keyword appears more than once in the RHS specification.

B.3 RHS Computation error messages

"Error: square root of negative number"
"Error: zerodivide"
"Error: overflow"
"Error: underflow"
"Error: logarithm of negative number"
"Error: real degree of negative number"
"Error: index out of range"
"Error: invalid operation"

A coprocessor invalid exception was detected, for example an attempt to compute 0/0 will result in the second message.

"Error: unknown type"

Internal error; please inform the authors.

"Cannot open file IR.DAT"
"File IR.DAT has invalid structure"

The former two messages mean that you have changed or deleted the IR.DAT file.

"Run-time data memory size is too small"

Total size of **COMMON** and **VAR** variables exceeds 2K.

"Run-time code memory size is too small"

Total size of generated machine instructions exceeds 4K.

C LOCBIF versions for fixed points and periodic solutions

In this Appendix we briefly explain how to use new versions of the LOCBIF program for fixed and periodic points of iterated maps and time-periodic solutions of periodic and autonomous ODEs.

C.1 General information

Now three additional versions of the **LOCBIF** program are available: for continuation and bifurcation analysis of the fixed points and periodic orbits of maps (**LBFP**), for continuation and bifurcation analysis of periodic solutions of periodically forced ODEs (**LBPS**) and for continuation and bifurcation analysis of limit cycles of autonomous ODEs (**LBLC**). All these programs have the same user interface as the **LBEP** version of **LOCBIF** described in this manual. These programs use the same basic continuation code and support a unified continuation strategy. The difference lies in the bifurcation functions and in the meaning of the corresponding bifurcations.

The installation of these versions is similat to that for **LBEP** version (see Section 2).To invoke a relevant version, enter one of the following commands: **LBFP**, **LBPS** or **LBLC**.

C.2 LBFP version of LOCBIF

C.2.1 Basic definitions

In this Appendix the term *iterated map* or *discrete time dynamical system* means a nonlinear mapping:

$$x' = F(x, p) \qquad (C.1)$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^n$ are *phase variables*, $p = (p_1, p_2, \ldots, p_m) \in \mathbf{R}^m$ are *parameters* and F is a smooth vector function. Here $'$ means the result of the action on point x of map $F(\cdot, p)$.

A *fixed point* is a point x in the phase space which is not affected by the map:

$$x - F(x,p) = 0 \qquad (C.2)$$

Multipliers of the fixed point are the eigenvalues of the linearization matrix of $(C.1)$ with respect to x at the fixed point.

An *orbit* or *trajectory* is a sequence of points in the phase space defined by the recurrent formulas:

$$x^{(k+1)} = F(x^{(k)}, p),\ k = 1, 2, 3, \ldots \qquad (C.3)$$

An orbit is called *periodic with period* K if $x^{(K)} = x^{(1)}$. Multipliers of the periodic orbit are eigenvalues of the Jacobian matrix of the K-th iterate of map F. A fixed point or periodic orbit is stable if all its multipliers $\mu_i, i = 1, 2, \ldots, n$, lie inside the unit circle $\mid \mu \mid = 1$. The fixed points bifurcate if there are multipliers on the unit circle.

C.2.2 Bifurcation functions

Bifurcation functions are scalar-valued functions which are used to define bifurcation curves and to find special points on them. The following bifurcation functions are involved in **LBFP** computations:

$$\begin{aligned}
\phi_1 &= \det(A - I) \\
\phi_2 &= D_{n-1} \\
\phi_3 &= \det(A + I) \\
\phi_4 &= Res\ (P(\mu), P'(\mu)) \\
\phi_5 &= a
\end{aligned}$$

Here $A = A(p)$ is the linearization (Jacobian) matrix of $(C.1)$ at the fixed (periodic) point (x, p). $P(\mu)$ is the characteristic polynomial of matrix A and $P'(\mu)$ is its first derivative with respect to μ. D_{n-1} denotes the Hurwitz determinant of the order $n-1$ of the polynomial

$$Q(\mu) = (1-\mu)^n P\left(\frac{1+\mu}{1-\mu}\right),$$

while *Res* stands for the resultant of two polynomials. Function a is determined by nonlinear (quadratic) terms of $(C.1)$. I denotes the unit matrix.

Condition $\phi_1 = 0$ implies that the Jacobian matrix A has at least one eigenvalue $\mu_1 = 1$. A fixed point which satisfies this condition generically has multiplicity two and bifurcates into two simple fixed points (fold bifurcation).

Function ϕ_2 is equal to zero if there are two multipliers μ_1 and μ_2 with unity product: $\mu_1\mu_2 = 1$. If a fixed point with this condition has two complex multipliers, *e.g.* $\mu_{1,2} = \exp(\pm i\omega)$, we have the Hopf (Neimark-Sacker) bifurcation and the appearance of a closed invariant curve. On the contrary, if the multipliers are real, we have a saddle point which does not bifurcate.

Condition $\phi_3 = 0$ implies the existence of a multiplier -1 which gives rise to the period doubling (flip) bifurcation.

Condition $\phi_4 = 0$ means the existence of equal multipliers $\mu_1 = \mu_2$.

The function *a(p)* is defined for a fixed point with $\mu_1 = 1$, if the corresponding eigenspace is one-dimensional. It is given by the expression

$$a(p) = \frac{1}{2}\frac{d^2}{d\xi^2}\langle e', F(x + \xi e, p)\rangle|_{\xi=0}$$

where e and e' are the eigenvectors of matrices A and A^T corresponding to the unity eigenvalue, with conditions $\langle e, e\rangle = 1, \langle e', e'\rangle = 1$. Here $\langle\cdot,\cdot\rangle$ denotes the standard scalar product in $\mathbf{R}^n$, and T stands for transposition.

C.2.3 Curve definitions

Bifurcation analysis in **LBFP** version of **LOCBIF** is performed by continuation of a curve (one dimensional manifolds) in an appropriate phase-parameter space. The number of non-fixed (*active*) parameters depends on the selected curve type. Continuation of the following curves is automatically supported by **LBFP**:

Curves with one active parameter:

Fixed point $(x - F(x,p) = 0)$
Curve $(x - F(x,p) = 0$ without bifurcation analysis)

Curves with two active parameters:

Fold $(x - F(x,p) = 0, \phi_1 = 0)$
Hopf $(x - F(x,p) = 0, \phi_2 = 0)$
Flip $(x - F(x,p) = 0, \phi_3 = 0)$
Double Multiplier $(x - F(x,p) = 0, \phi_4 = 0)$

The last curve is not a bifurcation curve but is useful in applications. The bifurcations of the dynamical system near the first three curves are described in (Arnold, 1982; Guckenheimer and Holmes, 1983).

Curves with three active parameters:

Double Fold $(x - F(x,p) = 0, \phi_1 = 0, \phi_2 = 0)$
Double Flip $(x - F(x,p) = 0, \phi_3 = 0, \phi_2 = 0)$

On these curves the system has a fixed point with double multiplier +1 (-1). In the literature these bifurcation are also known as *strong resonances* 1:1 and 1:2 (Arnold, 1982).

Fold + Hopf $(x - F(x,p) = 0, \phi_1 = 0, \phi_2 = 0)$
Flip + Hopf $(x - F(x,p) = 0, \phi_3 = 0, \phi_2 = 0)$
Fold + Flip $(x - F(x,p) = 0, \phi_1 = 0, \phi_3 = 0)$

These curves are the result of the superposition of two of the simplest bifurcation conditions.

Cusp $(x - F(x,p) = 0, \phi_1 = 0, \phi_5 = 0)$

On this curve we generically have a fixed point of multiplicity three.

Fold + Extr
Hopf + Extr
Flip + Extr

These curves of nontransversal bifurcations arise from **Fold**, **Hopf** and **Flip** curves in the same way as the corresponding curves in Section 6.

Note. The current version of **LBFP** does not support all possible two-parameter bifurcations of fixed points. It does not deal with the following bifurcations: two pairs of complex multipliers on the unit circle; degenerate flip and degenerate Hopf bifurcations, strong resonances 1:3 and 1:4.

C.2.4 Specific features of LBFP

As was pointed out before, the user interface of the **LBFP** version is the same as in the main part of the manual. The obvious difference is in the meaning of the **Orbit** curve which now is a sequence of points and can be computed only forward (we do not assume the invertibility of F). Therefore, almost all orbit parameters used in the numerical integration of ODEs have no meaning in **LBFP**.

The orbit parameter **Itmap** is the period of the studied orbit and should be equal to 1 for fixed points, 2 for period two orbits, *etc.* For discrete orbits the parameter **Itmap** plays almost the same role as the parameter **Tmax** for continuous orbits of ODEs.

C.2.5 Example: Periodic orbits of a discrete time population growth model

The model. Consider the following reccurence

$$x_{t+1} = rx_t(1 - x_{t-1}) + \epsilon \qquad (C.4)$$

where x_t is the density of a population at time t and r, ϵ are the growth and immigration rates.

If we introduce $y_t = x_{t-1}$, the equation $(C.4)$ can be rewritten as

$$\begin{aligned} x_{t+1} &= rx_t(1 - y_t) + \epsilon \\ y_{t+1} &= x_t \end{aligned}$$

which, in turn, defines a two-dimensional discrete time dynamical system:

$$\begin{aligned} x' &= rx(1 - y) + \epsilon \\ y' &= x \end{aligned} \qquad (C.5)$$

Input system $(C.5)$ into **LBFP** version of **LOCBIF**, for example, in the following form:

PHASE X,Y
PAR R,EPS
X′=R*X*(1-Y)+EPS
Y′=X

Find a fixed point. Set $r = 1.9, \epsilon = 0$. Select x and y as the abscissa and the ordinate with the visibility limits from 0 to 1. Input initial values $x_0 = 0.5$ and $y_0 = 0.2$. Set options: Pause=Update=Join=**No**. Select the curve type **Orbit**, iterate the system and check if the orbit converges to a fixed point. Try some other initial data. Select the last point as intial.

Continue the fixed point. Activate r, select the curve type **Fixed point**, set Update = **Yes**, Pause = **Special** and Join = **Yes**. Input **H0crv**=0.01

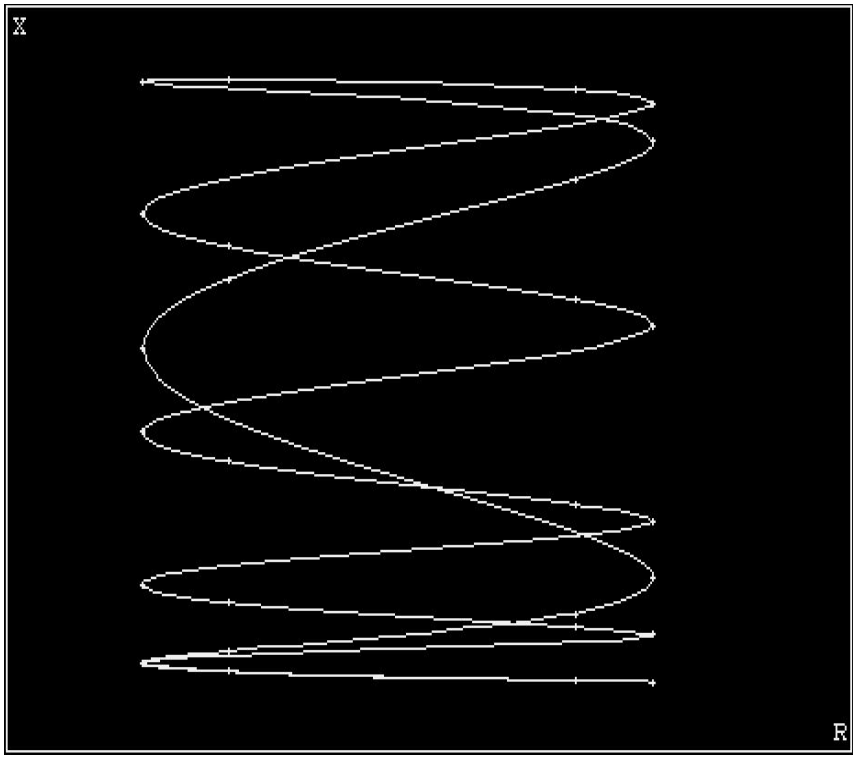

Figure 26: A closed curve of fixed points of the 7th-iteration of the map $(C.5)$

and **Hmxcrv**=0.1. (Do not change the projection!). Compute the curve and monitor the multipliers of the fixed point. Check, if message **"Hopf"** is reported which indicates the Neimark-Sacker bifurcation at $r = 2.0$. Find the arguments of the critical multipliers: $\omega = \frac{2\pi}{6}$ (therefore, we have so called *weak resonance*).

Find a closed invariant curve. Set **No** option for Pause, Update and Join modes. Compute **Orbit** curves for $r = 2.01, 2.05, 2.1, 2.15, 2.17$ using various intial data, and check if they converge to closed invariant curves giving rise to quasi-periodic reccurences.

Find and continue a period-7 cycle. Set $r = 2.177$ and compute **Orbit**. Select the last point, clean the screen and compute the orbit again. Reveal

by the **Pointwise** browsing that a period-7 cycle is found. Select a point on the cycle with maximal x.

Activate r, set Update=**Yes**, Join=**Yes** and Pause=**Special**. Put **Hmx-crv**=0.01 and select r as the abscissa (with $r_{min} = 2.17, r_{max} = 2.21$) and x as the ordinate (with $x_{min} = 0, x_{max} = 1.0$). Set **Itmap**=7, clean the Graphics Window and compute a **Fixed point** curve. Interpret the resulting closed curve (see Figure 26) and store it as **P7** in the archive. (*Hint*: Each point of a period-7 orbit is a fixed point of the 7-th iteration of the map ($C.5$).)

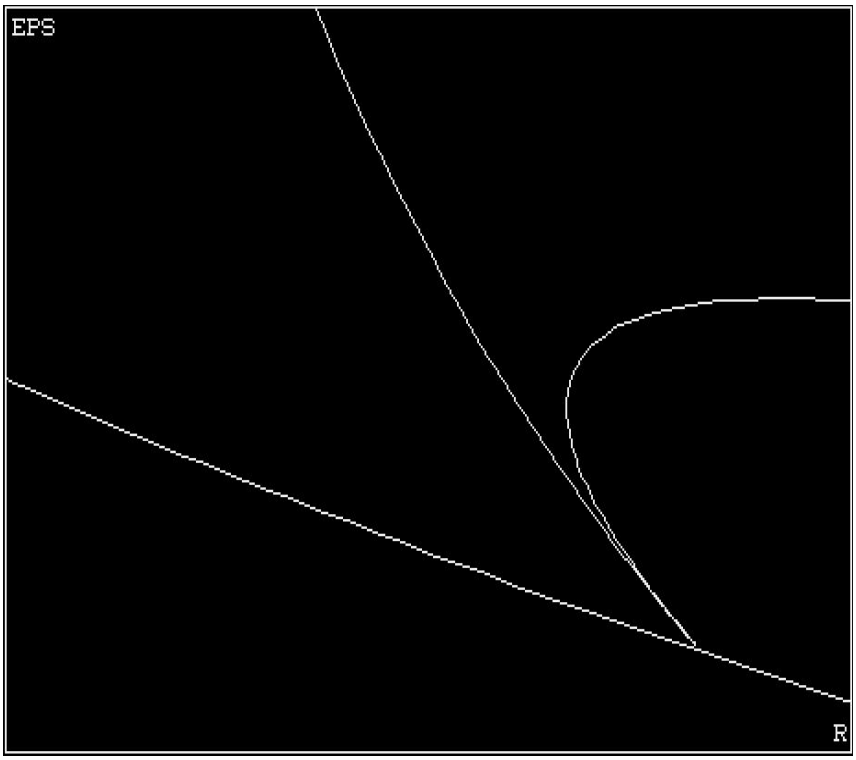

Figure 27: Arnold's tongue (the fold bifurcation curves for period-7 cycle - $t^{(7)}_{1,2}$) approaching a point of resonance 1:7 on the Hopf bifurcation curve $h^{(1)}$

Hopf continuation. Select r and ϵ as the abscissa and the ordinate with the visibility limits from 1.5 to 3.0 and from -1.15 to 1.15 respectively. Load **P1** curve from the archive and select the Hopf bifurcation point as initial.

Clean the Graphics Window. Activate ϵ, select the curve type **Hopf** and compute the Neimark-Sacker bifurcation curve within the visibility limits. Store it.

Tangent bifurcation continuation for period-7 cycle. Load **P7** curve and select a **"Multiplier = 1"** point corresponding to a higher value of r, set the increments **Dhcrv=Dhjac**=0.1E-06. Activate ϵ and compute the **Fold** curve in the both directions and store it. Load the **P7** curve once more and select a fold bifurcation point with the lower value of r, input the same values of the increments as before. Activate ϵ and compute **Fold** curve starting at this point. Store the curve. Clean the Graphics Window and browse the stored bifurcation curves (see Figure 27).

Notice, the fold curves form the so called *Arnold's tongue* which approaches a point on the Hopf curve. Find the corresponding argument value of the critical Hopf multipliers by browsing the Hopf curve using a different color ($\omega = \frac{2\pi}{7}$). Try to explain where are the period-6 cycles.

C.3 LBPS version of LOCBIF

C.3.1 General features of LBPS

This version of i**LOCBIF** is used for the analysis of periodically forced ODEs:

$$\dot{x} = F(t, x, p) \tag{C.6}$$

where F is assumed to be 2π-periodic in t. (This can be easily achived by time rescaling. You can also choose another sample period.) For system $(C.6)$ the *period return* or *Poincaré* map is defined (see Guckenheimer and Holmes, 1983) by the solution of the Initial Value Problem in the interval $[0, 2\pi]$. This map defines a discrete time dynamical system for which continuation and bifurcation analysis can be performed as described above in this Appendix. Periodic orbits of period K of this dynamical system correspond to time-periodic solutions (so called *subharmonics*) with period $2\pi K$ of $(C.6)$ they are called **Periodic Solutions** in the curve type list.

Using **LBPS** you have to specify 2π-periodic RHS of your ODEs and set the orbit parameter **Tmax** equal to 6.28... .While selecting **Orbit** type for computations, parameter **Iorbit** should be chosen +1 or -1 regarding

whether you would like to see the complete trajectory of $(C.6)$ in the interval [0, **Tmax*Itmap**] or only discrete orbit of the Poincaré map.

C.3.2 Example: Periodic solutions of a periodically forced predator-prey model

The model. Let us analyze the following time-periodic system of ODEs (Kuznetsov *et al.*, 1992):

$$\begin{aligned} \dot{x} &= rx(1-x) - \frac{cxy}{a(t)+x} \\ \dot{y} &= -dy + \frac{cxy}{a(t)+x} \end{aligned}$$

where $a(t) = b(1 + \epsilon \sin t)$, and parameters r, b, c, d are positive while $0 \leq \epsilon \leq 1$. If $\epsilon = 0$, the system becomes a classical predator-prey model. The periodic function $a(t)$ describes seasonal variation in the predator searching time.

To analyze periodic solutions of the system, input the equations into **LBPS** version of **LOCBIF**, for example, in the following form:

TIME T
PHASE X,Y
PAR R,B,C,D,EPS
FUN A,F
A=B*(1+EPS*sin(T))
F=C*X*Y/(A+X)
X′=R*X*(1-X)-F
Y′=-D*Y+F

Find an equilibrium in the unperturbed system. Set $r = 1, b = 0.4, c = 2, d = 1, \epsilon = 0$. Input $x_0 = 0.1$ and $y_0 = 0.4$ as initial data. Select x and y as the abscissa and the ordinate with the visibility limits from 0 to 1 and from 0 to 0.4 respectively. Set Update=**No** and compute **Orbit** forward. The orbit tends to a stable equilibrium point with coordinates: $(x^{(0)}, y^{(0)}) = (0.4..., 0.24...)$. Select this point as the initial one.

Continue the equilibrium in the unperturbed system. Activate parameter b, set **Messag**=1, **Epsint** $= 10^{-6}$ and check if **Tint** $= 6.28$. Modify the

continuation parameters by setting **H0crv**=0.01 and **Hmxcrv**=0.05. Select ϵ and b as the abscissa and the ordinate with the visibility limits from 0 to 1 and from 0 to 0.55 respectively. Continue the **Periodic solution** curve forward and backward and find a Hopf (Neimark-Sacker) bifurcation point. Select the Hopf point.

Continue the Neimark-Sacker bifurcation. Activate b and ϵ, select the curve type **Hopf**, set **Epsext**=0 (to suppress extremum location) and continue the bifurcation curve $h^{(1)}$ for $\epsilon > 0$ until a codimention two point A_1 with a double multiplier -1 will be approached (*strong resonance* 1:2 (see, Arnold, 1982)). Store the curve as **H1**.

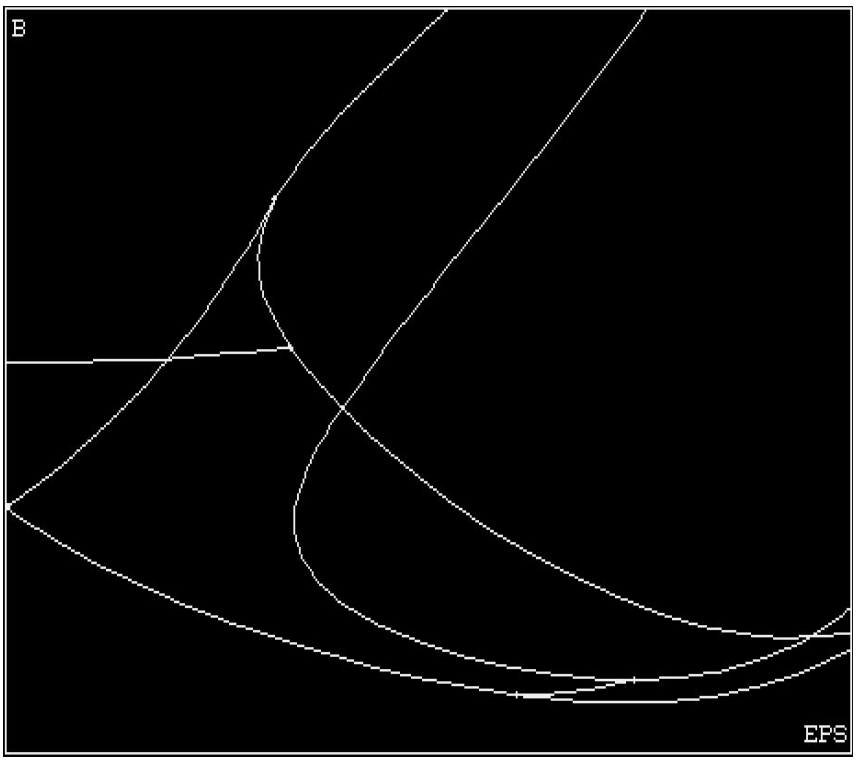

Figure 28: Bifurcation curves of the periodically forced predator-prey system: $h^{(1,2)}$ - **Hopf** for period-2π and -4π solutions; $f^{(1,2)}$- **Flip** for period-2π and -4π solutions; $t^{(2)}_{1,2}$ - **Fold** for period-2π solutions

Continuation of the flip bifurcation. Browse the Hopf curve and select the point of 1:2 resonance. Select the curve type **Flip** and continue the period doubling bifurcation curve $f^{(1)}$ in the both direction within the visibility limits (see Figure 28). Store the curve as **F1** in the archive.

Find a period-$2 \cdot 2\pi$ solution near the flip bifurcation. Browse the flip curve and stop at the upper point with $\epsilon = 0.50....$ Select the point. Select the curve type back to **Periodic solution**, deactivate ϵ and continue the period-2π solution curve into the unstable region. Take a point near the flip bifurcation and find a period-$2{\cdot}2\pi$ solution by integration (deactivate all the parameters, select the curve type **Orbit** and set **Iorbit**=-1 to look at an orbit generated by the 2π-map). Select a point on the found periodic orbit.

Continue the period-$2 \cdot 2\pi$ solution. Activate b and select the curve type **Periodic solution**. Set **Itmap**=2. Continue the period-$2{\cdot}2\pi$ solution with respect to b until the next period doubling point will be found and reported as **"Multiplier = -1"** (the message **"Selfcrossing"** obtained while computing the curve forward means approaching the flip bifurcation point for the period-2π solutions). Store the curve as **P21** into the archive.

Take the last point as the initial and activate now only the parameter ϵ. Continue the periodic solution and monitor the messages. Notice, there were no messages when the curve crossed the flip curve since this part of the flip curve corresponds to the appearence of an unstable period-$2 \cdot 2\pi$ cycle under decrease of the parameter ϵ. Continue until message: **"Multiplier = 1"** will appear which corresponds to the fold bifurcation of the period-$2 \cdot 2\pi$ orbits. Store the curve as **P22**. Select the last found point as initial, activate again only parameter b and continue the periodic solution until another tangent bifurcation point will be found at relatively low value of b. Store the curve as **P23**.

Continue flip, fold and Hopf bifurcation curves for the period-$2 \cdot 2\pi$ solutions. Load the curve **P21** from the archive and select the flip point for continuation of the **Flip** curve $f^{(2)}$. Activate the parameters b and ϵ, compute the curve and store it as **F2** into the archive. Notice, another point of codim 2 was found, namely, the point A_2 of resonance 1:2 for period-$2 \cdot 2\pi$ cycle.

Load in the same way the curves **P22** and **P23** and compute the **Fold** curves for the period-$2 \cdot 2\pi$ solutions. The upper branch $t_1^{(2)}$ of the fold curve

terminates at a codim 2 point D (*degenerate flip*, see: Afrajmovich *et al.*, 1985; Kuznetsov & Rinaldi, 1992) on the flip bifurcation curve $f^{(1)}$ and on the axis $\epsilon = 0$ at a point where the limit cycle in the unperturbed system has period 4π. Store this branch under name **T21**. The lower branch $t_2^{(2)}$ of the **Fold** curve (for its computation, set **Epsint**=0.1E-06) terminates at the same point on the vertical axis and passes through another codim 2 point B with a double +1 multiplier (*strong resonance* 1:1, Arnold(1982)) of the period-$2 \cdot 2\pi$ cycle. Store the curve as **T22**.

Select the resonance 1:1 point B as the initial one for **Hopf** bifurcation curve $h^{(2)}$ continuation and compute it. Notice, it terminates at the point A_2 in the flip curve $f^{(2)}$. Store the curve as **H2**. Clean the Graphics Window and plot all the bifurcation curves in one figure (see Figure 28).

C.4 LBLC version of LOCBIF

C.4.1 General features of LBLC

This version is designed for the analysis of isolated periodic solutions (*limit cycles*) of autonomous ODEs. For this problem the *Poincaré map* can also be defined. Consider the system of differential equations

$$\dot{x} = F(x, p) \tag{C.7}$$

Finding out periodic solutions with period T_0 of this system is equivalent to finding out periodic solutions with period 1 for the following *extended* system:

$$\dot{x} = T_0 F(x, p) \tag{C.8}$$

where T_0 is an extra parameter. A fixed point of time-one transformation of the phase space defined by $(C.8)$ corresponds to a limit cycle with period T_0 of the original system $(C.7)$ (they are called **Limit Cycle**s in the curve type list). Specifying RHS of your system, you have to reserve one more parameter in your parameter list for this extra parameter T_0 (you can use an arbitrary name for it but must place it as the *last* parameter in the list). Recall, that the total number of parameters (including T_0) should not exceed ten.

To avoid a phase shift along the cycle, you have to specify a *Poincaré section* ($(n-1)$-dimensional manifold $H = 0$ transversal to the limit cycle).

For this, orbit parameter **Isec** is used. If positive, **Isec** is a number of an equation in RHS which zero defines the secant surface ($H = F_{\mathrm{Isec}}$). If **Isec** is negative, |**Isec**| is an ordering number of a user-defined function zero of which defines the secant surface. Finally, **Isec** $= 0$ means that a secant plane will be initially chosen and then traced automatically to keep it orthogonal to the limit cycle.

Except one multiplier of a limit cycle which is always equals one (numerically it may be close to one), all other multipliers determine stability and bifurcations of the limit cycle. The bifurcation functions described above are used for definition and continuation of bifurcation curves for limit cycles as well.

The meaning of parameters **Itmap** and **Iorbit** is similar to that for **LBPS**.

Note. The period T_0 of the limit cycle is generically varying under parameter variations. Therefore, in the continuation of a limit cycle or while tracing its bifurcations, parameter T_0 must be active.

You can also compute an *isochrone* of constant cycle period by fixing T_0 and activating an extra system parameter.

C.4.2 Example: Limit cycle bifurcations in a Lur'e type feedback control system

The model. Consider the following nonlinear differential equation:

$$x''' + ax'' + bx' + x(1 - x) = 0$$

which is a Lur'e type feedback control model (see, Šiljak (1969)). Rewrite this system as the third order system of differential equations:

$$\begin{aligned} x' &= y \\ y' &= z \\ z' &= -az - by - x + x^2 \end{aligned} \qquad (C.9)$$

Input system $(C.9)$ into **LBLC** version of **LOCBIF** (do not forget to include the time scaling factor t_0 into the parameter list as the last parameter and multiply all the equations by t_0):

PHASE X,Y,Z

PAR A,B,T0
X′=Y*T0
Y′=Z*T0
Z′=(-A*Z-B*Y-X+X*X)*T0

Limit cycle location and determination of its period. Input $a = 0.8, b = 1$ and find a stable limit cycle in the model starting from the inital point $x_0 = 0.1, y_0 = 0, z_0 = 0.1$. Select the last point in the buffer as inital, clean the Graphics Window and continue **Orbit** computation several times. Use **No** Pause and **No** Update options.

Find the period of the cycle. For this, select a point on the cycle and recompute it with the Pause option **Pointwise**. Stop the computation when the orbit becomes (approximately) closed and estimate the period by reading the value of **time** in the Value Window. Set now **Tint**=1 but fix the parameter t_0 at the estimated value (6.34...). Set Pause to **Special** and recompute the cycle. Adjust t_0 to get the most exact closure. Browse the cycle once more and make a pause at the maximal value of x. Select the pause point.

Limit cycle continuation. Select the curve type **Limit cycle** and activate parameters a and t_0. Set **Flash**=500, **Messag**=1 and check if **Isec**=1 (the plane $y = 0$ will be used as a secant plane to construct the Poincaré map). Input **H0crv**=0.01 and **Hmxcrv**=0.1. Select a as the abscissa (with $a_{min} = 0$ and $a_{max} = 1.5$) and b as the ordinate (with $b_{min} = 0$ and $b_{max} = 2$). Clean the screen. Set Update option **Yes** and continue the cycle curve in the both directions until the massges " **Minimum of parameter a=...**" and "**Multiplier = -1**" will be found. The first message, actually, corresponds to the disappearence of the limit cycle through the Hopf bifurcation (check this!), while the second reveals the flip bifurcation. Compute a point on the curve corresponding to the unstable cycle (for this, change Pause option to **Poinwise** and let **LOCBIF** to find the next point after the flip). Store the cycle curve into the archive as **LC1**.

Flip continuation. Browse the cycle curve and select the flip point. Set **Epsext**=0 to supperss extremum location and select the curve type **Flip**. Acticate parameter b and continue the flip curve within the visibility limits. Store it as **F1**.

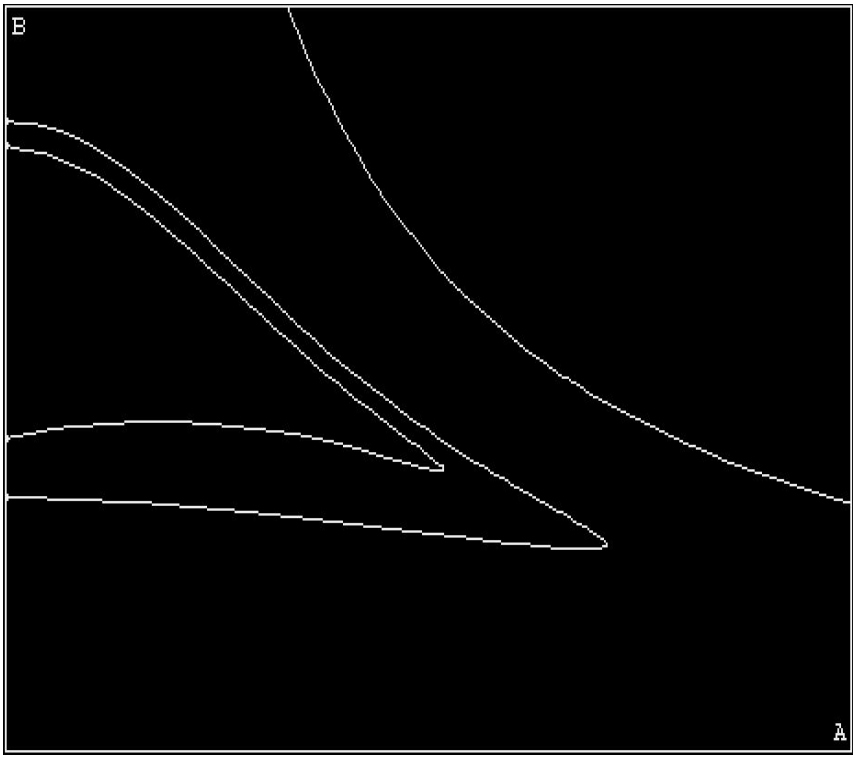

Figure 29: Bifurcation curves in model $(C.8)$: H_0 - the Hopf bifurcation for the equilibrium; $F_{1,2}$ - the flip bifurcation curves

Continue the Hopf bifurcation of an equilibrium. Load **LC1** curve and select the minimum of the parameter a point. Select the curve type **Fold** (explain, why), activate b and compute an approximation of the Hopf bifurcation curve for equilibrium $(0,0,0)$ in system $(C.8)$. Store it as **HOPF**.

Find and continue a double period cycle. Load **LC1** curve and select the last point corresponding to the unstable cycle. Set **Iorbit**=-1 and integrate **Orbit** with all the parameters deactivated. Make a pause when a double period cycle will be found. Set **Iorbit**=1 and repeate computation. Select a point with $y = 0$ (approximately).

Select the curve type **Limit cycle**, **Itmap**=2 and start continuation of the double period cycle with active a. Select a point reported as "**Multiplier**

= -1" as the initial for the flip continuation.

Continue the flip bifurcation for the double period cycle. Activate a and b, select the curve type **Flip**, set **Epsext**=0 to supperss extremum location and continue the flip curve within the visibility limits. Store it as **F2**. Clean the Graphics Window and plot the curves **HOPF, F1** and **F2** together (Figure 29).

D Mouse support

In this Appendix you will be informed how to use the mouse, if present, to operate **LOCBIF**.

D.1 Mouse installation

If your computer has a mouse and the Micrisoft mouse driver, you can use this pointer device to operate **LOCBIF** (all four versions: **LBEP**, **LBFP**, **LBLC** and **LBPS**). Before starting **LOCBIF**, you have to install the mouse driver by performing the command

MOUSE

*Note.*If your computer uses some other Microsoft-compatible mouse driver, the installation command may be slightly different (see your mouse manual).

Once the mouse driver is present and installed, **LOCBIF** will automatically detect it and allow you to operate with the mouse, as well as with the keyboard. When LOCBIF is invoked in the standard way, a mouse *pointer* appears over the Opening screen: it is a one-symbol rectangle which you can move within the screen while moving the mouse on the desk. Over the Main **LOCBIF** screen the pointer becomes a small arrow.

D.2 General conventions

The mouse has several *buttons* which can be pressed, or *clicked on.* In **LOCBIF** only the *left* and *right* buttons are used. Although your mouse might have a central button, it is ignored. In most situations only the left button is used. However, there are cases where clicking on the right button is meaningful.

LOCBIF detects the pointer position only at the moment when a mouse button is clicked on. Users can *select* various objects on the screen by moving the pointer to an appropiate position and clicking on a mouse button. These objects are the following:

- items of the Main Menu;
- entries in windows appearing over the main screen (i.e., commands, options, names and values);

- parts of hints (i.e., texts in the Message Window or some symbols);
- positions in the Graphics Window.

Pointing and clicking an object can result either in relocation of the highlight, or in an immediate action. In the first case, to perform an action you have to click on the button once more at the same mouse position.

If you are working in a window, you can close it either by clicking on the left mouse button when the pointer is *outside* the window, or by clicking on the right button *anywhere*. Notice, that in the first case clicking on some of the objects can cause another window to be opened or an action to be performed. (Rigorously speaking, in the second case you can also start some actions; see below.)

During the computation or curve browsing the pointer is disabled (invisible on the screen). It becomes visible again during pauses or if the computation or browsing are terminated. Nevertheless, during the computation/browsing the mouse reacts to button pressing.

D.3 LOCBIF management through mouse

Using the mouse you can perform many of the same operations as with the keyboard. It should be noted that there are text operations that can be done only using the keyboard. Nevertheless, the mouse allows you to operate quickly and more effectively.

Initial screen

You can use the mouse to select a dynamical system within the ODEs Archive Window. To do this, move the pointer to the name of the system and click on the left button. The highlight will be placed at this system name. If there are more systems than lines in the window, you can scroll the name list in the window by clicking on the up/down arrow symbols.

When a system is selected, you can proceed with its analysis by clicking on the button once more keeping the pointer at the same position. You can also perform one of the listed actions (edit, delete, *etc.*) by putting the pointer at a relevant text and clicking on the mouse button.

You cannot use the mouse inside the Equation Window to edit the system specification.

Main menu

Using the mouse to select and activate items of the Main Menu is straightforward and equivalent to using function keys F1-F7. But you can access the Value Window by simply moving the pointer inside the window and click on the left mouse button.

Command Window

If you have clicked on the **Commands** item of the Main Menu, the Command Window appears and you can perform directly one of the listed commands by pointing at it and pressing the left mouse button. The only exception is the Exit command. In this case, you have to click twice to confirm your intention to return to the Initial screen.

In accordance with the general conventions, you can leave the window by pressing the right button or the left button (outside the window). If you press the left button keeping the pointer at an item of the Main Menu, you will leave the Command Window and immediately activate the pointed item as if you had used function keys F2-F7.

If you point out a position inside the Value Window, the window will be activated (see section *Value Window* below).

Option and related Windows

If you have clicked on the **Options** item of the Main Menu, the Option Window appears and you can set one of the listed options directly or invoke one of the option parameter windows by selecting it with the left mouse button.

When the Axis Window is invoked, you can change the visibility limits by pointing at corresponding numerical values and using the keyboard. You can select a variable to be plotted along the abscissa(ordinata) axis in two ways. You have to point at the abscissa(ordinata) entry within the Axis Window, and either press the left mouse button several times until the desired name appears, or point at the desired name directly in the Value Window and click on the left button. To adjust the visibility limits automatically, select the corresponding part of the hint text using the left mouse button.

When the Continuation (Orbit or Service) Parameter Window is invoked, you can change the numerical values of the parameters by selecting them with the mouse and then using the keyboard.

You can leave the window as described in the previous section.

Curve Select Window

If you have clicked on the **Curves** item of the Main Menu, the Curve Select Window appears and you can select one of the listed curves using the left mouse button.

You can leave the window as usual. In this case the old curve selection will be preserved.

Value Window

If you have clicked on the **Values** item of the Main Menu, the Value Window becomes active and the cursor will appear inside it. If you select a position within the numerical value field the cursor will relocate to the selected position. Now you can modify the corresponding value immediately.

If you click at a name of a parameter its highlight status changes. It becomes highlighted if it was not highlighted previously, and not highlighted if it was. Therefore, this action is equivalent to (de)activation of a parameter with the Alt+F keys. It is possible to (de)activate a parameter by pointing at the corresponding part of the hint in the Message Window and pressing the left button. In the same way you can change lines Up/Down in the Value Window.

Note, that you can access the Value Window directly from the Main Menu or from the Command, Option and Curve Windows.

Curve Archive Window

If you have clicked on the **Archive** item of the Main Menu, the Curve Archive Window appears at the place of the Value Window. You can select one of the stored curves by pointing at it and pressing the left button. When the curve is selected, you can load it back into the memory for browsing by clicking on the left button once more. If a stored curve is selected, you can delete it by clicking on the corresponding part of the hint in the Message Window.

If there are more stored curves than lines in the Curve Archive Window, you can scroll the curve name list by selecting the corresponding Up/*Dn* symbol in the Message Window.

To store a curve, you can input its name using the keyboard and press the left mouse button keeping the pointer at the first window line. Clicking on this line with the void name results in the leaving the window.

Computation and browsing control

Note that clicking on the left button, if the **Compute** item in the Main Menu has been selected, results in starting the computation *forward*, while

clicking on the right button starts computation *backward*. The same is true for the **Browse** item.

During the computation/browsing the pointer is invisible, but you can click on the left button to make a pause, and click on it once more to continue computation/browsing. Clicking on the right button terminates the computation/browsing process.

Once a pause is initialized, the pointer appears again. You can change some of the selected options by pointing at the **Options** item of the Main Menu and clicking on the left mouse button. The Pause Option Window will appear and you can proceed as described before.

The current point can be selected as initial ("inserted") by pointing at the **Values** item of the Main Menu and pressing the left mouse button.

To continue computation/browsing, put the pointer at the relevant item in the Main Menu line and click on the left button if you compute/browse forward and the right button to compute/browse backward.

Setting initial point for orbit computation

If the selected curve is **Orbit** and some of the phase variables are used as the abscissa and the ordinata, it is possible to use the mouse to assign initial values to these variables and start the orbit computation. While working within the Main Menu or inside the Value Window, you can point at a point in the Graphics Window and click on the left/right mouse button to start the computation forward/backward. You can then proceed as usual.

Part V
References

V.S. Afrajmovich, V.I. Arnold, Yu.S. Il'yashenko and L.P.Shil'nikov, *Theory of Bifurcations*, in: Dynamical Systems V, V.I. Arnold (ed.), Encyclopaedia of Mathematical Sciences, Springer-Verlag, to appear [Russian original: VINITI, Moscow, 1985].

V.I. Arnold, *Geometrical Methods in the Theory of Ordinary Differential Equations*, Springer-Verlag, 1982.

A.D. Bazykin, *Mathematical Biophysics of Interacting Populations*, Nauka, Moscow, 1985 [in Russian]

A.D. Bazykin, Yu.A. Kuznetsov and A.I. Khibnik, *Bifurcation Diagrams of Planar Dynamical Systems*, USSR Academy of Sciences, Pushchino, 1985 [in Russian].

A.D. Bazykin, Yu.A. Kuznetsov and A.I. Khibnik, *Portraits of Bifurcations: Bifurcation Diagrams of Planar Dynamical Systems*, Znanie, Moscow, 1989 [in Russian].

F.S. Berezovskaya and A.I. Khibnik, *Bifurcations in second order dynamical system with double zero eigenvalue and addinitional degeneracy*, in: Methods of Qualitative Theory of Differential Equations, Gorkii, 1985, pp. 128-138 [in Russian].

V.I. Bykov, G.S. Yablonskii, V.F. Kim, *On the simple model of kinetic self-oscillations in catalytic reaction of CO oxidation*, Dokl. Sov. Math., pp. 637-639, 1978.

J. Carr, *Application of Centre Manifold Theory*, Applied Mathematical Sciences, 35, Springer-Verlag, 1981.

J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart. *LINPACK Users Guide*. SIAM Publications, Philadelphia, PA, 1978.

F. Dumortier, R. Roussarie and J. Sotomayor, *Generic 3-parameter families of vector fields on the plane, unfolding a singularity with nilpotent linear part. The cusp case of codimension 3*, Ergod. Theory and Dynam. Systems, v. 7, pp. 375-413, 1987.

F. Dumortier, R. Roussarie , J. Sotomayor, *Generic 3-parameter families of planar vector fields, unfolding of saddle, focus and elliptic singularities with with nilpotent linear parts,* Lecture Notes in Mathematics, 1480, Springer-Verlag, 1991.

I. Gohberg, P. Lancaster and L. Rodman, *Matrix Polinomials*, Academic Press, New-York, 1982.

J. Guckenhimer and Ph. Holmes, *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields*, Springer-Verlag, 1983

E. Hairer, S.P. Norsett, G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems*, Springer Series in Computational Mathematics 8, Springer-Verlag, 1987.

E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics 14, Springer-Verlag, 1991.

B. Hassard, N.D. Kazarinoff and Y.-H. Wan, *Theory and Applications of Hopf Bifurcations*, Cambridge University Press, Cambridge, 1981.

A.I. Khibnik, V.I. Bykov, G.S. Yablonskii, *23 phase portraits of the symplest catalytic oscillator*, J. Phys. Khim. **61**, pp. 1388-1390, 1987.

A.I. Khibnik, *LINLBF: A program for continuation and bifurcation analysis of equilibria up to codimension three*, in: Continuation and Bifurcations: Numerical Techniques and Applications, D. Roose *et al.* (eds.), Kluwer, The Netherlands, pp. 283-296, 1990.

A.I.Khibnik, *Numerical methods in bifurcation analysis of dynamical systems: a continuation approach*, in: Mathematics and Modelling, A.D. Bazykin and Yu.G. Zarhin (eds.), USSR Academy of Sciences, Pushchino, 1990, pp. 162-197 [in Russian].

A.I. Khibnik, Yu.A. Kuznetsov, V.V. Levitin, E.V. Nikolaev, *Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps*, Physica D, 1992.

Yu.A. Kuznetsov, S. Rinaldi, *Numerical analysis of the flip bifurcation of maps.* Appl.Math & Comp. **43**, 1991, pp. 231-236.

Yu.A. Kuznetsov, S. Muratori, S. Rinaldi, *Bifurcations and chaos in a periodic predator-prey model.* Int.J.Bifurcation&Chaos **2**, 1992, pp.117- 128.

D.D Šiljak, *Nonlinear Systems*. Wiley, New York, 1969.