AUTO 97 :
CONTINUATION AND BIFURCATION SOFTWARE FOR ORDINARY DIFFERENTIAL EQUATIONS (with HomCont)

Eusebius J. Doedel
Concordia University
Montreal, Canada

Alan R. Champneys
University of Bristol
United Kingdom

Thomas F. Fairgrieve
Ryerson Polytechnic University
Toronto, Canada

Yuri A. Kuznetsov
CWI, Amsterdam
The Netherlands

Björn Sandstede
Weierstraß-Institut
Berlin, Germany

Xianjun Wang
Concordia University
Montreal, Canada

July 1997

Contents

Preface

This is a guide to the software package AUTO for continuation and bifurcation problems in ordinary differential equations. Earlier versions of AUTO were described in Doedel (1981), Doedel & Kernévez (1986*a*), Doedel & Wang (1995), Wang & Doedel (1995). For a description of the basic algorithms see Doedel, Keller & Kernévez (1991*a*), Doedel, Keller & Kernévez (1991*b*). This version of AUTO incorporates the HomCont algorithms of Champneys & Kuznetsov (1994), Champneys, Kuznetsov & Sandstede (1996) for the bifurcation analysis of homoclinic orbits. The graphical user interface was written by Wang (1994). The Floquet multiplier algorithms were written by Fairgrieve (1994), Fairgrieve & Jepson (1991).

Acknowledgments

The first author is much indebted to H. B. Keller of the California Institute of Technology for his inspiration, encouragement and support. He is also thankful to AUTO users and research collaborators who have directly or indirectly contributed to its development, in particular, Jean Pierre Kernévez, UTC, Compiègne, France; Don Aronson, University of Minnesota, Minneapolis; and Hans Othmer, University of Utah. Material in this document related to the computation of connecting orbits was developed with Mark Friedman, University of Alabama, Huntsville. Also acknowledged is the work of Nguyen Thanh Long, Concordia University, Montreal, on the graphics program PLAUT and the pendula animation program. An earlier graphical user interface for AUTO on SGI machines was written by Taylor & Kevrekidis (1989). Special thanks are due to Sheila Shull, California Institute of Technology, for her cheerful assistance in the distribution of AUTO over a long period of time. Over the years, the development of AUTO has been supported by various agencies through the California Institute of Technology. Work on this updated version was supported by a general research grant from NSERC (Canada).

The development of HomCont has much benefitted from various pieces of help and advice from, among others, W.-J. Beyn, Universität Bielefeld, M. J. Friedman, University of Alabama, A. Rucklidge, University of Cambridge, M. Koper, University of Utrecht and C. J. Budd, University of Bristol. Financial support for collaboration was received from the U.K. Engineering and Physical Science Research Council and the Nuffield Foundation.

Chapter 1

Installing AUTO.

1.1 Installation.

The AUTO files `auto.ps.Z, auto.tar.Z` and `README` are available via FTP from directory `pub/doedel/auto` at *ftp.cs.concordia.ca.* The `README` file contains the instructions for printing this manual. Below it is assumed that you are using the Unix shell *csh* and that the file `auto.tar.Z` is in your main directory.

While in your main directory, enter the commands *uncompress auto.tar.Z*, followed by *tar xvfo auto.tar.* This will result in a directory `auto`, with one subdirectory, `auto/97`. Type *cd auto/97* to change directory to `auto/97`. Then type either *make sgi* , to compile AUTO on Silicon Graphics machines, or *make* on SUNs and, in principle, on other Unix systems. Upon compilation, type *make clean* to remove unnecessary files. Also enter the command *source $HOME/auto/97/cmds/auto.env* and add this command to your `.login` or `.cshrc` file.

The Graphical User Interface (GUI) requires the X-Window system and Motif. It may be necessary to enter their correct pathname in the appropriate makefile in `auto/97/gui`. Note that AUTO can be very effectively run in Command Mode, i.e., the GUI is not strictly necessary. To compile AUTO without GUI, type *make cmd* in directory `auto/97`.

For timing purposes, the file `auto/97/src/autlib1.f` contains references to the function *etime.* If this function is not automatically supplied by your f77 compiler then it can be replaced by an appropriate alternative call, or it can be disabled by replacing the two occurrences of the string *T=etime(timaray)* with *T=0.* To recompile `autlib1.f`, type *@C 1* in directory `auto/97/src`.

To enable the PostScript conversion command *@ps*, make the changes indicted in the `README` file in directory `auto/97/tek2ps` and recompile by typing *make* in that directory. Moreover, to enable the *@pr* command you may have to enter the correct printer name in `auto/97/cmds/@pr`. To generate the on-line manual, type *make* in `auto/97/doc`.

To prepare AUTO for transfer to another machine, type *make superclean* in directory `auto/97` before creating the *tar*-file. This will remove all executable, object, and other non-essential files, and thereby reduce the size of the package.

AUTO can be tested by typing *make > TEST &* in directory `auto/97/test`. This will execute a selection of demos from `auto/97/demos` and write a summary of the computations in the file `TEST`. The contents of `TEST` can then be compared to other test result files in directory `auto/97/test`. Note that minor differences are to be expected due to architecture and compiler differences.

Some EISPACK routines used by AUTO for computing eigenvalues and Floquet multipliers are included in the package (Smith, Boyle, Dongarra, Garbow, Ikebe, Klema & Moler (1976)).

1.2 Restrictions on Problem Size.

There are size restrictions in the file `auto/97/include/auto.h` on the following AUTO-constants : the effective problem dimension `NDIM`, the number of collocation points `NCOL`, the number of mesh intervals `NTST`, the effective number of boundary conditions `NBC`, the effective number of integral conditions `NINT`, the effective number of equation parameters `NPAR`, the number of stored branch points NBIF for algebraic problems, and the number of user output points `NUZR`. See Chapter 6 for the significance these constants. Their maxima are denoted by the corresponding constant followed by an X. For example, `NDIMX` in `auto.h` denotes the maximum value of `NDIM`. If any of these maxima is exceeded in an AUTO-run then a message will be printed. The exception is the the maximum value of `NPAR`, which, if exceeded, may lead to unreported errors. Upon installation `NPARX`=36; it should never be decreased below that value; see also Section 7.1. Size restrictions can be changed by editing `auto.h`. This must be followed by recompilation by typing *make* in directory `auto/97/src`. *It is strongly recommended that* `NCOLX=4` *be used, and that the value of* `NDIMX` *and* `NTSTX` *be chosen as small as possible for the intended application of* AUTO.

Note that in certain cases the *effective dimension* may be greater than the user dimension. For example, for the continuation of folds, the effective dimension is 2`NDIM`+1 for algebraic equations, and 2`NDIM` for ordinary differential equations, respectively. Similarly, for the continuation of Hopf bifurcations, the effective dimension is 3`NDIM`+2.

1.3 Compatibility with Older Versions.

There are two changes compared to early versions of AUTO94: The user-supplied equations-files must contain the subroutine `PVLS`. For an example of use of `PVLS` see the demo `pvl` in Section 15.1. There is also a small change in the `q.xxx` data-file. If necessary, older AUTO94 files can be converted using the *@94to97* command; see Section 3.5.

Chapter 2

Overview of Capabilities.

2.1 Summary.

AUTO can do a limited bifurcation analysis of algebraic systems

$$f(u,p) = 0, \qquad f(\cdot,\cdot), u \in \mathrm{R}^n, \tag{2.1}$$

and of systems of ordinary differential equation (ODEs) of the form

$$u'(t) = f(u(t),p), \qquad f(\cdot,\cdot), u(\cdot) \in \mathrm{R}^n, \tag{2.2}$$

Here p denotes one or more free parameters.

It can also do certain stationary solution and wave calculations for the partial differential equation (PDE)

$$u_t = Du_{xx} + f(u,p), \qquad f(\cdot,\cdot), u(\cdot) \in \mathrm{R}^n, \tag{2.3}$$

where D denotes a diagonal matrix of diffusion constants. The basic algorithms used in the package, as well as related algorithms, can be found in Keller (1977), Keller (1986), Doedel, Keller & Kernévez (1991*a*), Doedel, Keller & Kernévez (1991*b*).

Below, the basic capabilities of AUTO are specified in more detail. Some representative demos are also indicated.

2.2 Algebraic Systems.

Specifically, for (2.1) the program can :

- Compute solution branches.
  (Demo `ab`; Run 1.)

- Locate branch points and automatically compute bifurcating branches.
  (Demo `pp2`; Run 1.)

- Locate Hopf bifurcation points and continue these in two parameters.
  (Demo `ab`; Runs 1 and 5.)

- Locate folds (limit points) and continue these in two parameters.
  (Demo `ab`; Runs 1,3,4.)

- Do each of the above for fixed points of the discrete dynamical system $u^{(k+1)} = f(u^{(k)}, p)$
  (Demo `dd2`.)

- Find extrema of an objective function along solution branches and successively continue such extrema in more parameters.
  (Demo `opt`.)

2.3 Ordinary Differential Equations.

For the ODE (2.2) the program can :

- Compute branches of stable and unstable periodic solutions and compute the Floquet multipliers, that determine stability, along these branches. Starting data for the computation of periodic orbits are generated automatically at Hopf bifurcation points.
  (Demo `ab`; Run 2.)

- Locate folds, branch points, period doubling bifurcations, and bifurcations to tori, along branches of periodic solutions. Branch switching is possible at branch points and at period doubling bifurcations.
  (Demos `tor`, `lor`.)

- Continue folds and period-doubling bifurcations, in two parameters.
  (Demos `plp`, `pp3`.) The continuation of orbits of fixed period is also possible. This is the simplest way to compute curves of homoclinic orbits, if the period is sufficiently large.
  (Demo `pp2`.)

- Do each of the above for *rotations*, i.e., when some of the solution components are periodic modulo a phase gain of a multiple of 2π.
  (Demo `pen`.)

- Follow curves of homoclinic orbits and detect and continue various codimension-2 bifurcations, using the HomCont algorithms of Champneys & Kuznetsov (1994), Champneys, Kuznetsov & Sandstede (1996).
  (Demos `san`, `mnt`, `kpr`, `cir`, `she`, `rev`.)

- Locate extrema of an integral objective functional along a branch of periodic solutions and successively continue such extrema in more parameters.
  (Demo `ops`.)

- Compute curves of solutions to (2.2) on $[0,1]$, subject to general nonlinear boundary and integral conditions. The boundary conditions need not be separated, i.e., they may involve both $u(0)$ and $u(1)$ simultaneously. The side conditions may also depend on parameters. The number of boundary conditions plus the number of integral conditions need not equal the dimension of the ODE, provided there is a corresponding number of additional parameter variables.
  (Demos `exp`, `int`.)

- Determine folds and branch points along solution branches to the above boundary value problem. Branch switching is possible at branch points. Curves of folds can be computed in two parameters.
  (Demos `bvp`, `int`.)

2.4 Parabolic PDEs.

For (2.3) the program can :

- Trace out branches of spatially homogeneous solutions. This amounts to a bifurcation analysis of the algebraic system (2.1). However, AUTO uses a related system instead, in order to enable the detection of bifurcations to wave train solutions of given wave speed. More precisely, bifurcations to wave trains are detected as Hopf bifurcations along fixed point branches of the related ODE

$$\begin{aligned} u'(z) &= v(z), \\ v'(z) &= -D^{-1}[c\, v(z) + f(u(z), p)], \end{aligned} \tag{2.4}$$

  where $z = x - ct$, with the wave speed c specified by the user.
  (Demo `wav`; Run 2.)

- Trace out branches of periodic wave solutions to (2.3) that emanate from a Hopf bifurcation point of Equation 2.4. The wave speed c is fixed along such a branch, but the wave length L, i.e., the period of periodic solutions to (2.4), will normally vary. If the wave length L becomes large, i.e., if a homoclinic orbit of Equation 2.4 is approached, then the wave tends to a solitary wave solution of (2.3).
  (Demo `wav`; Run 3.)

- Trace out branches of waves of fixed wave length L in two parameters. The wave speed c may be chosen as one of these parameters. If L is large then such a continuation gives a branch of approximate solitary wave solutions to (2.3).
  (Demo `wav`; Run 4.)

- Do time evolution calculations for (2.3), given periodic initial data on the interval $[0, L]$. The initial data must be specified on $[0,1]$ and L must be set separately because of internal

scaling. The initial data may be given analytically or obtained from a previous computation of wave trains, solitary waves, or from a previous evolution calculation. Conversely, if an evolution calculation results in a stationary wave then this wave can be used as starting data for a wave continuation calculation.
(Demo `wav`; Run 5.)

- Do time evolution calculations for (2.3) subject to user-specified boundary conditions. As above, the initial data must be specified on $[0, 1]$ and the space interval length L must be specified separately. Time evolution computations of (2.3) are adaptive in space and in time. Discretization in time is not very accurate : only implicit Euler. Indeed, time integration of (2.3) has only been included as a convenience and it is not very efficient. (Demos `pd1`, `pd2`.)

- Compute curves of stationary solutions to (2.3) subject to user-specified boundary conditions. The initial data may be given analytically, obtained from a previous stationary solution computation, or from a time evolution calculation.
(Demos `pd1`, `pd2`.)

In connection with periodic waves, note that (2.4) is just a special case of (2.2) and that its fixed point analysis is a special case of (2.1). One advantage of the built-in capacity of AUTO to deal with problem (2.3) is that the user need only specify f, D, and c. Another advantage is the compatibility of output data for restart purposes. This allows switching back and forth between evolution calculations and wave computations.

2.5 Discretization.

AUTO discretizes ODE boundary value problems (which includes periodic solutions) by the method of orthogonal collocation using piecewise polynomials with 2-7 collocation points per mesh interval (de Boor & Swartz (1973)). The mesh automatically adapts to the solution to equidistribute the local discretization error (Russell & Christiansen (1978)). The number of mesh intervals and the number of collocation points remain constant during any given run, although they may be changed at restart points. The implementation is AUTO-specific. In particular, the choice of local polynomial basis and the algorithm for solving the linearized collocation systems were specifically designed for use in numerical bifurcation analysis.

Chapter 3

How to Run AUTO.

3.1 User-Supplied Files.

The user must prepare the two files described below. This can be done with the GUI described in Chapter 5, or independently.

3.1.1 The equations-file `xxx.f`

A source file `xxx.f` containing the Fortran subroutines `FUNC`, `STPNT`, `BCND`, `ICND`, `FOPT`, and `PVLS`. Here `xxx` stands for a user-selected name. If any of these subroutines is irrelevant to the problem then its body need not be completed. Examples are in `auto/97/demos`, where, e.g., the file `ab/ab.f` defines a two-dimensional dynamical system, and the file `exp/exp.f` defines a boundary value problem. The simplest way to create a new equations-file is to copy an appropriate demo file. For a fully documented equations-file see `auto/97/gui/aut.f`. In GUI mode, this file can be directly loaded with the GUI-button *Equations/New*; see Section 5.2.

3.1.2 The constants-file `r.xxx`

AUTO-constants for `xxx.f` are normally expected in a corresponding file `r.xxx`. Specific examples include `ab/r.ab` and `exp/r.exp` in `auto/97/demos`. See Chapter 6 for the significance of each constant.

3.2 User-Supplied Subroutines.

The purpose of each of the user-supplied subroutines in the file `xxx.f` is described below.

- `FUNC` : defines the function $f(u, p)$ in (2.1), (2.2), or (2.3).
- `STPNT` : This subroutine is called only if `IRS`=0 (see Section 6.8.5 for `IRS`), which typically is the case for the first run. It defines a starting solution (u, p) of (2.1) or (2.2). The starting solution should not be a branch point.
  (Demos `ab`, `exp`, `frc`, `lor`.)
- `BCND` : A subroutine `BCND` that defines the boundary conditions.
  (Demo `exp`, `kar`.)
- `ICND` : A subroutine `ICND` that defines the integral conditions.
  (Demos `int`, `lin`.)
- `FOPT` : A subroutine `FOPT` that defines the objective functional.
  (Demos `opt`, `ops`.)
- `PVLS` : A subroutine `PVLS` for defining "solution measures".
  (Demo `pvl`.)

3.3 Arguments of `STPNT`.

Note that the arguments of `STPNT` depend on the solution type :

- When starting from a fixed point, `STPNT` must have three arguments, namely, (`NDIM,U,PAR`).
  (See demo `ab`).
- When starting from an analytically or numerically known space-dependent solution, `STPNT` must have four arguments, namely, (`NDIM,U,PAR,T`). Here T is the independent space variable which takes values in the interval $[0, 1]$.
  (Demo `exp`.)
- Similarly, when starting from an analytically known time-periodic solution or rotation, the arguments of `STPNT` are (`NDIM,U,PAR,T`), where T denotes the independent time variable which takes values in the interval $[0, 1]$. In this case one must also specify the period in `PAR(11)`.
  (Demos `frc`, `lor`, `pen`.)
- When using the *@fc* command (Section 3.5) for conversion of numerical data, `STPNT` must have three arguments, namely, (`NDIM,U,PAR`). In this case only the parameter values need to be defined in `STPNT`. (Demos `lor` and `pen`.)

3.4 User-Supplied Derivatives.

If AUTO-constant `JAC` equals 0 then derivatives need not be specified in `FUNC`, `BCND`, `ICND`, and `FOPT`; see Section sec:JAC. If `JAC=1` then derivatives must be given. This may be necessary for sensitive problems, and is recommended for computations in which AUTO generates an extended system. Examples of user-supplied derivatives can be found in demos `dd2`, `int`, `plp`, `opt`, and `ops`.

3.5 Running AUTO using Command Mode.

AUTO can be run with the GUI described in Chapter 5 or with the commands described below. The AUTO aliases must have been activated; see Section 1.1; and an equations-file `xxx.f` and a corresponding constants-file `r.xxx` (see Section 3.1) must be in the current user directory.
Do not run AUTO *in the directory* `auto/97` *or in any of its subdirectories.*

`@r` : Type *@r xxx* to run AUTO. Restart data, if needed, are expected in `q.xxx`, and AUTO-constants in `r.xxx`. This is the simplest way to run AUTO.

- Type *@r xxx yyy* to run AUTO with equations-file `xxx.f` and restart data-file `q.yyy`. AUTO-constants must be in `r.xxx`.

- Type *@r xxx yyy zzz* to run AUTO with equations-file `xxx.f`, restart data-file `q.yyy` and constants-file `r.zzz`.

`@R` : The command *@R xxx* is equivalent to the command *@r xxx* above.

- Type *@R xxx i* to run AUTO with equations-file `xxx.f`, constants-file `r.xxx.i` and, if needed, restart data-file `q.xxx`.

- Type *@R xxx i yyy* to run AUTO with equations-file `xxx.f`, constants-file `r.xxx.i` and restart data-file `q.yyy`.

`@sv` : Type *@sv xxx* to save the output-files `fort.7`, `fort.8`, `fort.9`, as `p.xxx`, `q.xxx`, `d.xxx`, respectively. Existing files by these names will be deleted.

`@ap` : Type *@ap xxx* to append the output-files `fort.7`, `fort.8`, `fort.9`, to existing data-files `p.xxx`, `q.xxx`, `d.xxx`, resp.

- Type *@ap xxx yyy* to append `p.xxx`, `q.xxx`, `d.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, resp.

`@p` : Type *@p xxx* to run the graphics program PLAUT (See Chapter 4) for the graphical inspection of the data-files `p.xxx` and `q.xxx`.

- Type *@p* to run the graphics program PLAUT for the graphical inspection of the output-files `fort.7` and `fort.8`.

`@ps` : Type *@ps fig.x* to convert a saved PLAUT figure `fig.x` from compact PLOT10 format to PostScript format. The converted file is called `fig.x.ps`. The original file is left unchanged.

`@pr` : Type *@pr fig.x* to convert a saved PLAUT figure `fig.x` from compact PLOT10 format to PostScript format and send it to the printer. The converted file is called `fig.x.ps`. The original file is left unchanged.

`@cp` : Type *@cp xxx yyy* to copy the data-files `p.xxx`, `q.xxx`, `d.xxx`, `r.xxx` to `p.yyy`, `q.yyy`, `d.yyy`, `r.yyy`, respectively.

`@mv` : Type *@mv xxx yyy* to move the data-files `p.xxx`, `q.xxx`, `d.xxx`, `r.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, `r.yyy`, respectively.

`@df` : Type *@df* to delete the output-files `fort.7`, `fort.8`, `fort.9`.

`@cl` : Type *@cl* to clean the current directory. This command will delete all files of the form `fort.*`, `*.o`, and `*.exe`.

`@dl` : Type *@dl xxx* to delete the data-files `p.xxx`, `q.xxx`, `d.xxx`.

`@dm` : Type *@dm xxx* to copy all files from `auto/97/demos/xxx` to the current user directory. Here `xxx` denotes a demo name; e.g., `abc`. Note that the *@dm* command also copies a *Makefile* to the current user directory. To avoid the overwriting of existing files, always run demos in a clean work directory.

`@lb` : Type *@lb* to run an interactive utility program for listing, deleting and relabeling solutions in the AUTO output-files `fort.7` and `fort.8`. The original files are backed up as ∼`fort.7` and ∼`fort.8`.

- Type *@lb xxx* to list, delete and relabel solutions in the AUTO data-files `p.xxx` and `q.xxx`. The original files are backed up as ∼`p.xxx` and ∼`q.xxx`.
- Type *@lb xxx yyy* to list, delete and relabel solutions in the AUTO data-files `p.xxx` and `q.xxx`. The modified files are written as `p.yyy` and `q.yyy`.

`@fc` : Type *@fc xxx* to convert a user-supplied data file `xxx.dat` to AUTO format. The converted file is called `q.dat`. The original file is left unchanged. AUTO automatically sets the period in `PAR(11)`. Other parameter values must be set in `STPNT`. (When necessary, PAR(11) may also be redefined there.) The constants-file file `r.xxx` must be present, as the AUTO-constants `NTST` and `NCOL` (Sections 6.3.1 and 6.3.2) are used to define the new mesh. For examples of using the *@fc* command see demos `lor` and `pen`.

`@pn` : Type *@pn xxx* to run the pendula animation program with data-file `q.xxx`. (On SGI machine only; see demo `pen` in Section 10.10 and the file `auto/97/pendula/README`.)

`@94to97` : Type *@94to97 xxx* to convert an old AUTO94 data-file `q.xxx` to new AUTO97 format. The original file is backed up as ∼`q.xxx`. This conversion is only necessary for files from early versions of AUTO94.

`@mn` : Use GHOSTVIEW to view the PostScript version of this manual.

`@h` : Use `@h` instead of `@r` when using HOMCONT, i.e., when `IPS`=9 (see Chapter 16). Type *@h xxx* to run AUTO/HOMCONT. Restart data, if needed, are expected in `q.xxx`, AUTO-constants in `r.xxx` and HOMCONT-constants in `s.xxx`.

- Type *@h xxx yyy* to run AUTO/HOMCONT with equations-file `xxx.f` and restart data-file `q.yyy`. AUTO-constants must be in `r.xxx` and HOMCONT-constants in `s.xxx`.

- Type *@h xxx yyy zzz* to run AUTO/HOMCONT with equations-file `xxx.f`, restart data-file `q.yyy` and constants-files `r.zzz` and `s.zzz`.

`@H` : The command *@H xxx* is equivalent to the command *@h xxx* above.

- Type *@H xxx i* to run AUTO/HOMCONT with equations-file `xxx.f` and constants-files `r.xxx.i` and `s.xxx.i` and, if needed, restart data-file `q.xxx`.

- Type *@H xxx i yyy* to run AUTO/HOMCONT with equations-file `xxx.f`, constants-files `r.xxx.i` and `s.xxx.i`, and restart data-file `q.yyy`.

3.6 Output Files.

AUTO writes four output-files :

- `fort.6` : A summary of the computation is written in `fort.6`, which usually corresponds to the window in which AUTO is run. Only special, labeled solution points are noted, namely those listed in Table 3.1. The letter codes in the Table are used in the screen output. The numerical codes are used internally and in the `fort.7` and `fort.8` output-files described below.

- `fort.7` : The `fort.7` output-file contains the bifurcation diagram. Its format is the same as the `fort.6` (screen) output, but the `fort.7` output is more extensive, as every solution point has an output line printed.

- `fort.8` : The `fort.8` output-file contains complete graphics and restart data for selected, labeled solutions. The information per solution is generally much more extensive than that in `fort.7`. The `fort.8` output should normally be kept to a minimum.

- `fort.9` : Diagnostic messages, convergence history, eigenvalues, and Floquet multipliers are written in `fort.9`. It is strongly recommended that this output be habitually inspected. The amount of diagnostic data can be controlled via the AUTO-constant `IID`; see Section 6.9.2.

BP	(1)	Branch point (algebraic systems)
LP	(2)	Fold (algebraic systems)
HB	(3)	Hopf bifurcation
	(4)	User-specified regular output point
UZ	(-4)	Output at user-specified parameter value
LP	(5)	Fold (differential equations)
BP	(6)	Branch point (differential equations)
PD	(7)	Period doubling bifurcation
TR	(8)	Torus bifurcation
EP	(9)	End point of branch; normal termination
MX	(-9)	Abnormal termination; no convergence

Table 3.1: Solution Types.

The user has some control over the `fort.6` (screen) and `fort.7` output via the AUTO-constant `IPLT` (Section 6.9.3). Furthermore, the subroutine `PVLS` can be used to define "solution measures" which can then be printed by "parameter overspecification"; see Section 6.7.10. For an example see demo `pvl`.

The AUTO-commands *@sv*, *@ap*, and *@df* can be used to manipulate the output-files `fort.7`, `fort.8`, and `fort.9`. Furthermore, the AUTO-command *@lb* can be used to delete and relabel solutions simultaneously in `fort.7` and `fort.8`. For details see Section 3.5.

The graphics program PLAUT can be used to graphically inspect the data in `fort.7` and `fort.8`; see Chapter 4.

Chapter 4

The Graphics Program PLAUT.

PLAUT can be used to extract graphical information from the AUTO output-files `fort.7` and `fort.8`, or from the corresponding data-files `p.xxx` and `q.xxx`. To invoke PLAUT, use the the *@p* command defined in Section 3.5. The PLAUT window (a Tektronix window) will appear, in which PLAUT commands can be entered. For examples of using PLAUT see the tutorial demo `ab`, in particular, Sections 8.7 and 8.10. See also demo `pp2` in Section 10.3.

4.1 Basic PLAUT-Commands.

The principal PLAUT-commands are

`bd0` : This command is useful for an initial overview of the bifurcation diagram as stored in `fort.7`. If you have not previously selected one of the default options *d0, d1, d2, d3*, or *d4* described below then you will be asked whether you want solution labels, grid lines, titles, or labeled axes.

`bd` : This command is the same as the *bd0* command, except that you will be asked to enter the minimum and the maximum of the horizontal and vertical axes. This is useful for blowing up portions of a previously displayed bifurcation diagram.

`ax` : With the *ax* command you can select any pair of columns of real numbers from `fort.7` as horizontal and vertical axis in the bifurcation diagram. (The default is columns 1 and 2). To determine what these columns represent, one can look at the screen ouput of the corresponding AUTO run, or one can inspect the column headings in `fort.7`.

`2d` : Upon entering the *2d* command, the labels of all solutions stored in `fort.8` will be listed and you can select one or more of these for display. The number of solution components is also listed and you will be prompted to select two of these as horizontal and vertical axis in the display. Note that the first component is typically the independent time or space variable scaled to the interval [0,1].

`sav` : To save the displayed plot in a file. You will be asked to enter a file name. Each plot must be stored in a separate new file. The plot is stored in compact PLOT10 format, which can be converted to PostScript format with the AUTO-commands `@ps` and `@pr`; see Section 4.4.

`cl` : To clear the graphics window.

`lab` : To list the labels of all solutions stored in `fort.8`. Note that PLAUT requires all labels to be distinct. In case of multiple labels you can use the AUTO command *@lb* to relabel solutions in `fort.7` and `fort.8`.

`end` : To end execution of PLAUT.

4.2 Default Options.

After entering the commands *bd0, bd*, or *2d*, you will be asked whether you want solution labels, grid lines, titles, or axes labels. For quick plotting it is convenient to bypass these selections. This can be done by the default commands *d0, d1, d2, d3*, or *d4* below. These can be entered as a single command or they can be entered as prefixes in the *bd0* and *bd* commands. Thus, for example, one can enter the command *d1bd0.*

`d0` : Use solid curves, showing solution labels and symbols.

`d1` : Use solid curves, except use dashed curves for unstable solutions and for solutions of unknown stability. Show solution labels and symbols.

`d2` : As *d1*, but with grid lines.

`d3` : As *d1*, except for periodic solutions use solid circles if stable, and open circles if unstable or if the stability is unknown.

`d4` : Use solid curves, without labels and symbols.

If no default option *d0, d1, d2, d3*, or *d4* has been selected or if you want to override a default feature, then the the following commands can be used. These can be entered as individual commands or as prefixes. For example, one can enter the command *sydpbd0.*

`sy` : Use symbols for special solution points, for example, open square = branch point, solid square = Hopf bifurcation.

`dp` : "Differential Plot", i.e., show stability of the solutions. Solid curves represent stable solutions. Dashed curves are used for unstable solutions and for solutions of unknown stability. For periodic solutions use solid/open circles to indicate stability/instability (or unknown stability).

`st` : Set up titles and axes labels.

`nu` : Normal usage (reset special options).

4.3 Other PLAUT-Commands.

The full PLAUT program has several other capabilities, for example,

`scr` : To change the diagram size.

`rss` : To change the size of special solution point symbols.

4.4 Printing PLAUT Files.

`@ps` : Type *@ps fig.1* to convert a saved PLAUT file `fig.1` to PostScript format in `fig.1.ps`.

`@pr` : Type *@pr fig.1* to convert a PLAUT file `fig.1` to PostScript format and to print the resulting file `fig.1.ps`.

Chapter 5

Graphical User Interface.

5.1 General Overview.

The AUTO97 graphical user interface (GUI) is a tool for creating and editing equations-files and constants-files; see Section 3.1 for a description of these files. The GUI can also be used to run AUTO and to manipulate and plot output-files and data-files; see Section 3.5 for corresponding commands. To use the GUI for a new equation, change to an empty work directory. For an existing equations-file, change to its directory. (*Do not activate the GUI in the directory* `auto/97` *or in any of its subdirectories.*) Then type

$$@auto,$$

or its abbreviation @*a*. Here we assume that the AUTO aliases have been activated; see Section 1.1. The GUI includes a window for editing the equations-file, and four groups of buttons, namely, the *Menu Bar* at the top of the GUI, the *Define Constants*-buttons at the center-left, the *Load Constants*-buttons at the lower left, and the *Stop- and Exit*-buttons.

Note : Most GUI buttons are activated by point-and-click action with the *left* mouse button. If a beep sound results then the *right* mouse button must be used.

5.1.1 The Menu bar.

It contains the main buttons for running AUTO and for manipulating the equations-file, the constants-file, the output-files, and the data-files. In a typical application, these buttons are used from left to right. First the *Equations* are defined and, if necessary, *Edited*, before being *Written*. Then the AUTO-constants are *Defined*. This is followed by the actual *Run* of AUTO. The resulting output-files can be *Saved* as data-files, or they can be *Appended* to existing data-files. Data-files can be *Plotted* with the graphics program PLAUT, and various file operations can be done with the *Files*-button. Auxiliary functions are provided by the *Demos*-, *Misc*-, and *Help*-buttons. The Menu Bar buttons are described in more detail in Section 5.2.

5.1.2 The Define-Constants-buttons.

These have the same function as the *Define*-button on the Menu Bar, namely to set and change AUTO-constants. However, for the *Define*-button all constants appear in one panel, while for the Define Constants-buttons they are grouped by function, as in Chapter 6, namely *Problem* definition constants, *Discretization* constants, convergence *Tolerances*, continuation *Step Size*, diagram *Limits*, designation of free *Parameters*, constants defining the *Computation*, and constants that specify *Output* options.

5.1.3 The Load-Constants-buttons.

The *Previous*-button can be used to load an existing AUTO-constants file. Such a file is also loaded, if it exists, by the *Equations*-button on the *Menu Bar*. The *Default*-button can be used to load default values of all AUTO-constants. Custom editing is normally necessary.

5.1.4 The Stop- and Exit-buttons.

The *Stop*-button can be used to abort execution of an AUTO-run. This should be done only in exceptional circumstances. Output-files, if any, will normally be incomplete and should be deleted. Use the *Exit*-button to end a session.

5.2 The Menu Bar.

5.2.1 Equations-button.

This pull-down menu contains the items *Old*, to load an existing equations-file, *New*, to load a model equations-file, and *Demo*, to load a selected demo equations-file. Equations-file names are of the form `xxx.f`. The corresponding constants-file `r.xxx` is also loaded if it exists. The equation name `xxx` remains active until redefined.

5.2.2 Edit-button.

This pull-down menu contains the items *Cut* and *Copy*, to be performed on text in the GUI window highlighted by click-and-drag action of the mouse, and the item *Paste*, which places editor buffer text at the location of the cursor.

5.2.3 Write-button.

This pull-down menu contains the item *Write*, to write the loaded files `xxx.f` and `r.xxx`, by the active equation name, and the item *Write As* to write these files by a selected new name, which then becomes the active name.

5.2.4 Define-button.

Clicking this button will display the full AUTO-constants panel. Most of its text fields can be edited, but some have restricted input values that can be selected with the right mouse button. Some text fields will display a subpanel for entering data. To actually apply changes made in the panel, click the *OK*- or *Apply*-button at the bottom of the panel.

5.2.5 Run-button.

Clicking this button will write the constants-file `r.xxx` and run AUTO. If the equations-file has been edited then it should first be rewritten with the *Write*-button.

5.2.6 Save-button.

This pull-down menu contains the item *Save*, to save the output-files `fort.7`, `fort.8`, `fort.9`, as `p.xxx`, `q.xxx`, `d.xxx`, respectively. Here `xxx` is the active equation name. It also contains the item *Save As*, to save the output-files under another name. Existing data-files with the selected name, if any, will be overwritten.

5.2.7 Append-button.

This pull-down menu contains the item *Append*, to append the output-files `fort.7`, `fort.8`, `fort.9`, to existing data-files `p.xxx`, `q.xxx`, `d.xxx`, respectively. Here `xxx` is the active equation name. It also contains the item *Append To*, to append the output-files to other existing data-files.

5.2.8 Plot-button.

This pull-down menu contains the items *Plot*, to run the plotting program PLAUT for the data-files `p.xxx` and `q.xxx`, where `xxx` is the active equation name, and the item *Name*, to run PLAUT with other data-files.

5.2.9 Files-button.

This pull-down menu contains the item *Restart*, to redefine the restart file. Normally, when restarting from a previously computed solution, the restart data is expected in the file `q.xxx`, where `xxx` is the active equation name. Use the *Restart*-button to read the restart data from another data-file in the immediately following run. The pull-down menu also contains the following items :

- *Copy*, to copy `p.xxx`, `q.xxx`, `d.xxx`, `r.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, `r.yyy`, resp.;
- *Append*, to append data-files `p.xxx`, `q.xxx`, `d.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, resp.;
- *Move*, to move `p.xxx`, `q.xxx`, `d.xxx`, `r.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, `r.yyy`, resp.;
- *Delete*, to delete data-files `p.xxx`, `q.xxx`, `d.xxx`;
- *Clean*, to delete all files of the form `fort.*`, `*.o`, and `*.exe`.

5.2.10 Demos-button.

This pulldown menu contains the items *Select*, to view and run a selected AUTO demo in the demo directory, and *Reset*, to restore the demo directory to its original state. Note that demo files can be copied to the user work directory with the *Equations/Demo*-button.

5.2.11 Misc.-button.

This pulldown menu contains the items *Tek Window* and *VT102 Window*, for opening windows; *Emacs* and *Xedit*, for editing files, and *Print*, for printing the active equations-file `xxx.f`.

5.2.12 Help-button.

This pulldown menu contains the items AUTO-*constants*, for a description of AUTO-constants, and *User Manual*, for viewing the user manual; i.e., this document.

5.3 Using the GUI.

AUTO-commands are described in Section 3.5 and illustrated in the demos. In Table 5.1 we list the main AUTO-commands together with the corresponding GUI button.

The AUTO-command *@r xxx yyy* is given in the GUI as follows : click *Files/Restart* and enter *yyy* as data. Then click *Run*. As noted in Section 3.5, this will run AUTO with the current equations-file `xxx.f` and the current constants-file `r.xxx`, while expecting restart data in `q.yyy`. The AUTO-command *@ap xxx yyy* is given in the GUI by clicking *Files/Append*.

5.4 Customizing the GUI.

5.4.1 Print-button.

The *Misc/Print*-button on the Menu Bar can be customized by editing the file `GuiConsts.h` in directory `auto/97/include`.

@r	*Run*
@sv	*Save*
@ap	*Append*
@p	*Plot*
@cp	*Files/Copy*
@mv	*Files/Move*
@cl	*Files/Clean*
@dl	*Files/Delete*
@dm	*Equations/Demo*

Table 5.1: Command Mode - GUI correspondences.

5.4.2 GUI colors.

GUI colors can be customized by creating an X resource file. Two model files can be found in directory `auto/97/gui`, namely, `Xdefaults.1` and `Xdefaults.2`. To become effective, edit one of these, if desired, and copy it to `.Xdefaults` in your home directory. Color names can often be found in the system file `/usr/lib/X11/rgb.txt`.

5.4.3 On-line help.

The file `auto/97/include/GuiGlobal.h` contains on-line help on AUTO-constants and demos. The text can be updated, subject to a modifiable maximum length. On SGI machines this is 10240 bytes, which can be increased, for example, to 20480 bytes, by replacing the line *CC = cc -Wf, -XNl10240 -O* in `auto/97/gui/Makefile` by *CC = cc -Wf, -XNl20480 -O* On other machines, the maximum message length is the system defined maximum string literal length.

Chapter 6

Description of AUTO-Constants.

6.1 The AUTO-Constants File.

As described in Section 3.1, if the equations-file is `xxx.f` then the constants that define the computation are normally expected in the file `r.xxx`. The general format of this file is the same for all AUTO runs. For example, the file `r.ab` in directory `auto/97/demos/ab` is listed below. (The tutorial demo `ab` is described in detail in Chapter 8.)

```
2 1 0 1                 NDIM,IPS,IRS,ILP
1   1                   NICP,(ICP(I),I=1,NICP)
50 4 3 1 1 0 0 0        NTST,NCOL,IAD,ISP,ISW,IPLT,NBC,NINT
100 0. 0.15 0. 100.     NMX,RL0,RL1,A0,A1
100 10 2 8 5 3 0        NPR,MXBF,IID,ITMX,ITNW,NWTN,JAC
1.e-6 1.e-6 0.0001      EPSL,EPSU,EPSS
0.01 0.005 0.05 1       DS,DSMIN,DSMAX,IADS
1                       NTHL,((I,THL(I)),I=1,NTHL)
11 0.
0                       NTHU,((I,THU(I)),I=1,NTHU)
0                       NUZR,((I,UZR(I)),I=1,NUZR)
```

The significance of the AUTO-constants, grouped by function, is described in the sections below. Representative demos that illustrate use of the AUTO-constants are also mentioned.

6.2 Problem Constants.

6.2.1 NDIM

Dimension of the system of equations as specified in the user-supplied subroutine `FUNC`.

6.2.2 NBC

The number of boundary conditions as specified in the user-supplied subroutine BCND.
(Demos exp, kar.)

6.2.3 NINT

The number of integral conditions as specified in the user-supplied subroutine ICND.
(Demos int, lin, obv.)

6.2.4 JAC

Used to indicate whether derivatives are supplied by the user or to be obtained by differencing :

- JAC=0 : No derivatives are given by the user. (Most demos use JAC=0.)
- JAC=1 : Derivatives with respect to state- and problem-parameters are given in the user-supplied subroutines FUNC, BCND, ICND and FOPT, where applicable. This may be necessary for sensitive problems. It is also recommended for computations in which AUTO generates an extended system, for example, when ISW=2. (For ISW see Section 6.8.3.)
  (Demos int, dd2, obt, plp, ops.)

6.3 Discretization Constants.

6.3.1 NTST

The number of mesh intervals used for discretization. NTST remains fixed during any particular run, but can be changed when restarting. (For mesh adaption see IAD in Section 6.3.3.) Recommended value of NTST : As small as possible to maintain convergence.
(Demos exp, ab, spb.)

6.3.2 NCOL

The number of Gauss collocation points per mesh interval, ($2 \leq$ NCOL ≤ 7). NCOL remains fixed during any given run, but can be changed when restarting at a previously computed solution. The choice NCOL=4, used in most demos, is recommended. If NDIM is "large" and the solutions "very smooth" then NCOL=2 may be appropriate.

6.3.3 IAD

This constant controls the mesh adaption :

- `IAD=0` : Fixed mesh. This choice should never be used, as it may result in spurious solutions. (Demo `ext`.)
- `IAD>0` : Adapt the mesh every `IAD` steps along the branch. Most demos use `IAD=3`, which is the strongly recommended value.

6.4 Tolerances.

6.4.1 EPSL

Relative convergence criterion for equation parameters in the Newton/Chord method. Most demos use `EPSL`=10^{-6} or `EPSL`=10^{-7}, which is the recommended value range.

6.4.2 EPSU

Relative convergence criterion for solution components in the Newton/Chord method. Most demos use `EPSU`=10^{-6} or `EPSU`=10^{-7}, which is the recommended value range.

6.4.3 EPSS

Relative arclength convergence criterion for detecting special solutions. Most demos use `EPSS`=10^{-4} or `EPSS`=10^{-5}, which is the recommended value range. Generally, `EPSS` should be approximately 100 to 1000 times the value of `EPSL`, `EPSU`.

6.4.4 ITMX

The maximum number of iterations allowed in the accurate location of special solutions, such as bifurcations, folds, and user output points, by Müller's method with bracketing. The recommended value is `ITMX`=8, used in most demos.

6.4.5 NWTN

After `NWTN` Newton iterations the Jacobian is frozen, i.e., AUTO uses full Newton for the first `NWTN` iterations and the Chord method for iterations `NWTN`+1 to `ITNW`. The choice `NWTN`=3 is strongly recommended and used in most demos. Note that this constant is only effective for ODEs, i.e., for solving the piecewise polynomial collocation equations. For algebraic systems AUTO always uses full Newton.

6.4.6 ITNW

The maximum number of combined Newton-Chord iterations. When this maximum is reached, the step will be retried with half the stepsize. This is repeated until convergence, or until the minimum stepsize is reached. In the latter case the computation of the branch is discontinued and a message printed in `fort.9`. The recommended value is ITNW=5, but ITNW=7 may be used for “difficult” problems, for example, demos `spb`, `chu`, `plp`, etc.

6.5 Continuation Step Size.

6.5.1 DS

AUTO uses pseudo-arclength continuation for following solution branches. The pseudo-arclength stepsize is the distance between the current solution and the next solution on a branch. By default, this distance includes all state variables (or state functions) and all free parameters. The constant DS defines the pseudo-arclength stepsize to be used for the first attempted step along any branch. (Note that if IADS>0 then DS will automatically be adapted for subsequent steps and for failed steps.) DS may be chosen positive or negative; changing its sign reverses the direction of computation. The relation DSMIN $\leq$ | DS | $\leq$ DSMAX must be satisfied. The precise choice of DS is problem-dependent; the demos use a value that was found appropriate after some experimentation.

6.5.2 DSMIN

This is minimum allowable absolute value of the pseudo-arclength stepsize. DSMIN must be positive. It is only effective if the pseudo-arclength step is adaptive, i.e., if IADS>0. The choice of DSMIN is highly problem-dependent; most demos use a value that was found appropriate after some experimentation. See also the discussion in Section 7.2.

6.5.3 DSMAX

The maximum allowable absolute value of the pseudo-arclength stepsize. DSMAX must be positive. It is only effective if the pseudo-arclength step is adaptive, i.e., if IADS>0. The choice of DSMAX is highly problem-dependent; most demos use a value that was found appropriate after some experimentation. See also the discussion in Section 7.2.

6.5.4 IADS

This constant controls the frequency of adaption of the pseudo-arclength stepsize.

- `IADS=0` : Use fixed pseudo-arclength stepsize, i.e., the stepsize will be equal to the specified value of `DS` for every step. The computation of a branch will be discontinued as soon as the maximum number of iterations `ITNW` is reached. This choice is not recommended. (Demo `tim`.)

- `IADS>0` : Adapt the pseudo-arclength stepsize after every `IADS` steps. If the Newton/Chord iteration converges rapidly then | `DS` | will be increased, but never beyond `DSMAX`. If a step fails then it will be retried with half the stepsize. This will be done repeatedly until the step is successful or until | `DS` | reaches `DSMIN`. In the latter case nonconvergence will be signalled. The strongly recommended value is `IADS`=1, which is used in almost all demos.

6.5.5 NTHL

By default, the pseudo-arclength stepsize includes all state variables (or state functions) and all free parameters. Under certain circumstances one may want to modify the weight accorded to individual parameters in the definition of stepsize. For this purpose, `NTHL` defines the number of parameters whose weight is to be modified. If `NTHL`=0 then all weights will have default value 1.0 . If `NTHL`>0 then one must enter `NTHL` pairs, *Parameter Index Weight* , with each pair on a separate line.

For example, for the computation of periodic solutions it is recommended that the period not be included in the pseudo-arclength continuation stepsize, in order to avoid period-induced limitations on the stepsize near orbits of infinite period. This exclusion can be accomplished by setting `NTHL=1`, with, on a separate line, the pair 11 0.0 . Most demos that compute periodic solutions use this option; see for example demo `ab`.

6.5.6 NTHU

Under certain circumstances one may want to modify the weight accorded to individual state variables (or state functions) in the definition of stepsize. For this purpose, `NTHU` defines the number of states whose weight is to be modified. If `NTHU`=0 then all weights will have default value 1.0 . If `NTHU`>0 then one must enter `NTHU` pairs, *State Index Weight* , with each pair on a separate line. At present none of the demos use this option.

6.6 Diagram Limits.

There are three ways to limit the computation of a branch :

- By appropriate choice of the computational window defined by the constants `RL0`, `RL1`, `A0`, and `A1`. One should always check that the starting solution lies within this computational window, otherwise the computation will stop immediately at the starting point.

- By specifying the maximum number of steps, `NMX`.
- By specifying a negative parameter index in the list associated with the constant `NUZR`; see Section 6.9.4.

6.6.1 `NMX`

The maximum number of steps to be taken along any branch.

6.6.2 `RL0`

The lower bound on the principal continuation parameter. (This is the parameter which appears first in the `ICP` list; see Section 6.7.1.).

6.6.3 `RL1`

The upper bound on the principal continuation parameter.

6.6.4 `A0`

The lower bound on the principal solution measure. (By default, if `IPLT`=0, the principal solution measure is the L_2-norm of the state vector or state vector function. See the AUTO-constant `IPLT` in Section 6.9.3 for choosing another principal solution measure.)

6.6.5 `A1`

The upper bound on the principal solution measure.

6.7 Free Parameters.

6.7.1 `NICP, ICP`

For each equation type and for each continuation calculation there is a typical ("generic") number of problem parameters that must be allowed to vary, in order for the calculations to be properly posed. The constant `NICP` indicates how many free parameters have been specified, while the array `ICP` actually designates these free parameters. The parameter that appears first in the `ICP` list is called the "principal continuation parameter". Specific examples and special cases are described below.

6.7.2 Fixed points.

The simplest case is the continuation of a solution branch to the system $f(u,p) = 0$, where $f(\cdot,\cdot), u \in \mathrm{R}^n$, cf. Equation (2.1). Such a system arises in the continuation of ODE stationary solutions and in the continuation of fixed points of discrete dynamical systems. There is only one free parameter here, so `NICP`=1.

As a concrete example, consider Run 1 of demo `ab`, where `NICP=1`, with `ICP(1)=1`. Thus, in this run `PAR(1)` is designated as the free parameter.

6.7.3 Periodic solutions and rotations.

The continuation of periodic solutions and rotations generically requires two parameters, namely, one problem parameter and the period. Thus, in this case `NICP`=2. For example, in Run 2 of demo `ab` we have `NICP`=2, with `ICP`(1)=1 and `ICP`(2)=11. Thus, in this run, the free parameters are `PAR(1)` and `PAR(11)`. (Note that AUTO reserves `PAR(11)` for the period.)

Actually, for periodic solutions, one can set `NICP`=1 and only specify the index of the free problem parameter, as AUTO will automatically addd `PAR(11)`. However, in this case the period will not appear in the screen output and in the `fort.7` output-file.

For fixed period orbits one must set `NICP`=2 and specify two free problem parameters. For example, in Run 7 of demo `pp2`, we have `NICP`=2, with `PAR(1)` and `PAR(2)` specified as free problem parameters. The period `PAR(11)` is fixed in this run. If the period is large then such a continuation provides a simple and effective method for computing a locus of homoclinic orbits.

6.7.4 Folds and Hopf bifurcations.

The continuation of folds for algebraic problems and the continuation of Hopf bifurcations requires two free problem parameters, i.e., `NICP`=2. For example, to continue a fold in Run 3 of demo `ab`, we have `NICP`=2, with `PAR(1)` and `PAR(3)` specified as free parameters. Note that one must set `ISW`=2 for computing such loci of special solutions. Also note that in the continuation of folds the principal continuation parameter must be the one with respect to which the fold was located.

6.7.5 Folds and period-doublings.

The continuation of folds, for periodic orbits and rotations, and the continuation of period-doubling bifurcations require two free problem parameters plus the free period. Thus, one would normally set `NICP`=3. For example, in Run 6 of demo `pen`, where a locus of period-doubling bifurcations is computed for rotations, we have `NICP`=3, with `PAR(2)`, `PAR(3)`, and `PAR(11)` specified as free parameters. Note that one must set `ISW`=2 for computing such loci of special solutions. Also note that in the continuation of folds the principal continuation parameter must be the one with respect to which the fold was located.

Actually, one may set `NICP`=2, and only specify the problem parameters, as AUTO will automatically add the period. For example, in Run 3 of demo `plp`, where a locus of folds is computed for periodic orbits, we have `NICP`=2, with `PAR(4)` and `PAR(1)` specified as free parameters. However, in this case the period will not appear in the screen output and in the `fort.7` output-file.

To continue a locus of folds or period-doublings with fixed period, simply set `NICP`=3 and specify three problem parameters, not including `PAR(11)`.

6.7.6 Boundary value problems.

The simplest case is that of boundary value problems where `NDIM`=`NBC` and where `NINT`=0. Then, generically, one free problem parameter is required for computing a solution branch. For example, in demo `exp`, we have `NDIM`=`NBC`=2, `NINT`=0. Thus `NICP`=1. Indeed, in this demo one free parameter is designated, namely `PAR(1)`.

More generally, for boundary value problems with integral constraints, the generic number of free parameters is `NBC` + `NINT`−`NDIM` +1. For example, in demo `lin`, we have `NDIM`=2, `NBC`=2, and `NINT`=1. Thus `NICP`=2. Indeed, in this demo two free parameters are designated, namely `PAR(1)` and `PAR(3)`.

6.7.7 Boundary value folds.

To continue a locus of folds for a general boundary value problem with integral constraints, set `NICP`=`NBC`+`NINT`−`NDIM`+2, and specify this number of parameter indices to designate the free parameters.

6.7.8 Optimization problems.

In algebraic optimization problems one must set `ICP`(1)=10, as AUTO uses `PAR(10)` as principal continuation parameter to monitor the value of the objective function. Furthermore, one must designate one free equation parameter in `ICP`(2). Thus, `NICP`=2 in the first run.

Folds with respect to `PAR(10)` correspond to extrema of the objective function. In a second run one can restart at such a fold, with an additional free equation parameter specified in `ICP`(3). Thus, `NICP`=3 in the second run.

The above procedure can be repeated. For example, folds from the second run can be continued in a third run with three equation parameters specified in addition to `PAR(10)`. Thus, `NICP`=4 in the third run.

For a simple example see demo `opt`, where a four-parameter extremum is located. Note that `NICP`=5 in each of the four constants-files of this demo, with the indices of `PAR(10)` and `PAR(1)-PAR(4)` specified in `ICP`. Thus, in the first three runs, there are overspecified parameters.

However, AUTO will always use the correct number of parameters. Although the overspecified parameters will be printed, their values will remain fixed.

6.7.9 Internal free parameters.

The actual continuation scheme in AUTO may use additional free parameters that are automatically added. The simplest example is the computation of periodic solutions and rotations, where AUTO automatically adds the period, if not specified. The computation of loci of folds, Hopf bifurcations, and period-doublings also requires additional internal continuation parameters. These will be automatically added, and their indices will be greater than 10.

6.7.10 Parameter overspecification.

The number of specified parameter indices is allowed to be be greater than the generic number. In such case there will be "overspecified" parameters, whose values will appear in the screen and `fort.7` output, but which are not part of the continuation process. A simple example is provided by demo `opt`, where the first three runs have overspecified parameters whose values, although constant, are printed.

There is, however, a more useful application of parameter overspecification. In the user-supplied subroutine `PVLS` one can define solution measures and assign these to otherwise unused parameters. Such parameters can then be overspecified, in order to print them on the screen and in the `fort.7` output. It is important to note that such overspecified parameters must appear at the end of the `ICP` list, as they cannot be used as true continuation parameters.

For an example of using parameter overspecification for printing user-defined solution measures, see demo `pvl`. This is a boundary value problem (Bratu's equation) which has only one true continuation parameter, namely `PAR(1)`. Three solution measures are defined in the subroutine `PVLS`, namely, the L_2-norm of the first solution component, the minimum of the second component, and the left boundary value of the second component. These solution measures are assigned to `PAR(2)`, `PAR(3)`, and `PAR(4)`, respectively. In the constants-file `r.pvl` we have `NICP`=4, with `PAR(1)-PAR(4)` specified as parameters. Thus, in this example, `PAR(2)-PAR(4)` are overspecified. Note that `PAR(1)` must appear first in the `ICP` list; the other parameters cannot be used as true continuation parameters.

6.8 Computation Constants.

6.8.1 `ILP`

- `ILP=0` : No detection of folds. This choice is recommended.

- `ILP=1` : Detection of folds. To be used if subsequent fold continuation is intended.

6.8.2 `ISP`

This constant controls the detection of branch points, period-doubling bifurcations, and torus bifurcations.

- `ISP=0` : This setting disables the detection of branch points, period-doubling bifurcations, and torus bifurcations and the computation of Floquet multipliers.

- `ISP=1` : Branch points are detected for algebraic equations, but not for periodic solutions and boundary value problems. Period-doubling bifurcations and torus bifurcations are not located either. However, Floquet multipliers are computed.

- `ISP=2` : This setting enables the detection of all special solutions. For periodic solutions and rotations, the choice `ISP`=2 should be used with care, due to potential inaccuracy in the computation of the linearized Poincaré map and possible rapid variation of the Floquet multipliers. The linearized Poincaré map always has a multiplier $z = 1$. If this multiplier becomes inaccurate then the automatic detection of secondary periodic bifurcations will be discontinued and a warning message will be printed in `fort.9`. See also Section 7.4.

- `ISP=3` : Branch points will be detected, but AUTO will not monitor the Floquet multipliers. Period-doubling and torus bifurcations will go undetected. This option is useful for certain problems with non-generic Floquet behavior.

6.8.3 `ISW`

This constant controls branch switching at branch points for the case of differential equations. Note that branch switching is automatic for algebraic equations.

- `ISW=1` : This is the normal value of `ISW`.

- `ISW=`−1 : If `IRS` is the label of a branch point or a period-doubling bifurcation then branch switching will be done. For period doubling bifurcations it is recommended that `NTST` be increased. For examples see Run 2 and Run 3 of demo `lor`, where branch switching is done at period-doubling bifurcations, and Run 2 and Run 3 of demo `bvp`, where branch switching is done at a transcritical branch point.

- `ISW=2` : If `IRS` is the label of a fold, a Hopf bifurcation point, or a period-doubling bifurcation then a locus of such points will be computed. An additional free parameter must be specified for such continuations; see also Section 6.7.

6.8.4 `MXBF`

This constant, which is effective for algebraic problems only, sets the maximum number of bifurcations to be treated. Additional branch points will be noted, but the corresponding bifurcating branches will not be computed. If MXBF is positive then the bifurcating branches of the first MXBF branch points will be traced out in both directions. If MXBF is negative then the bifurcating branches of the first | MXBF | branch points will be traced out in only one direction.

6.8.5 IRS

This constant sets the label of the solution where the computation is to be restarted.

- IRS=0 : This setting is typically used in the first run of a new problem. In this case a starting solution must be defined in the user-supplied subroutine STPNT; see also Section 3.3. For representative examples of analytical starting solutions see demos ab and frc. For starting from unlabeled numerical data see the *@fc* command (Section 3.5) and demos lor and pen.

- IRS>0 : Restart the computation at the previously computed solution with label IRS. This solution is normally expected to be in the current data-file q.xxx; see also the *@r* and *@R* commands in Section 3.5. Various AUTO-constants can be modified when restarting.

6.8.6 IPS

This constant defines the problem type :

- IPS=0 : An algebraic bifurcation problem. Hopf bifurcations will not be detected and stability properties will not be indicated in the fort.7 output-file.

- IPS=1 : Stationary solutions of ODEs with detection of Hopf bifurcations. The sign of PT, the point number, in fort.7 is used to indicate stability : $-$ is stable , $+$ is unstable. (Demo ab.)

- IPS=$-$1 : Fixed points of the discrete dynamical system $u^{(k+1)} = f(u^{(k)}, p)$, with detection of Hopf bifurcations. The sign of PT in fort.7 indicates stability : $-$ is stable , $+$ is unstable.
  (Demo dd2.)

- IPS=$-$2 : Time integration using implicit Euler. The AUTO-constants DS, DSMIN, DSMAX, and ITNW, NWTN control the stepsize. In fact, pseudo-arclength is used for "continuation in time". Note that the time discretization is only first order accurate, so that results should be carefully interpreted. Indeed, this option has been included primarily for the detection of stationary solutions, which can then be entered in the user-supplied subroutine STPNT.
  (Demo ivp.)

- IPS=2 : Computation of periodic solutions. Starting data can be a Hopf bifurcation point (Run 2 of demo ab), a periodic orbit from a previous run (Run 4 of demo pp2), an analytically known periodic orbit (Run 1 of demo frc), or a numerically known periodic orbit (Demo `lor`). The sign of PT in `fort.7` is used to indicate stability : $-$ is stable , $+$ is unstable or unknown.

- `IPS=4` : A boundary value problem. Boundary conditions must be specified in the user-supplied subroutine `BCND` and integral constraints in `ICND`. The AUTO-constants `NBC` and `NINT` must be given correct values.
  (Demos `exp`, `int`, `kar`.)

- `IPS=5` : Algebraic optimization problems. The objective function must be specified in the user-supplied subroutine `FOPT`.
  (Demo `opt`.)

- `IPS=9` : This option is used in connection with the HomCont algorithms described in Chapters 16-22 for the detection and continuation of homoclinic bifurcations.
  (Demos `san`, `mtn`, `kpr`, `cir`, `she`, `rev`.)

- `IPS=11` : Spatially uniform solutions of a system of parabolic PDEs, with detection of bifurcations to traveling waves. The user need only define the nonlinearity (in the user-supplied subroutine `FUNC`), initialize the wave speed in `PAR(10)`, initialize the diffusion constants in `PAR(15,16,···)`, and set a free equation parameter in `ICP`(1). (Run 2 of demo `wav`.)

- `IPS=12` : Continuation of traveling wave solutions to a system of parabolic PDEs. Starting data can be a Hopf bifurcation point from a previous run with `IPS`=11, or a traveling wave from a previous run with `IPS`=12. (Run 3 and Run 4 of demo `wav`.)

- `IPS=14` : Time evolution for a system of parabolic PDEs subject to periodic boundary conditions. Starting data may be solutions from a previous run with `IPS`=12 or 14. Starting data can also be specified in `STPNT`, in which case the wave length must be specified in `PAR(11)`, and the diffusion constants in `PAR(15,16,···)`. AUTO uses `PAR(14)` for the time variable. `DS`, `DSMIN`, and `DSMAX` govern the pseudo-arclength continuation in the space-time variables. Note that the time discretization is only first order accurate, so that results should be carefully interpreted. Indeed, this option is mainly intended for the detection of stationary waves. (Run 5 of demo `wav`.)

- `IPS=15` : Optimization of periodic solutions. The integrand of the objective functional must be specified in the user supplied subroutine `FOPT`. Only `PAR(1-9)` should be used for problem parameters. `PAR(10)` is the value of the objective functional, `PAR(11)` the period, `PAR(12)` the norm of the adjoint variables, `PAR(14)` and `PAR(15)` are internal optimality variables. `PAR(21-29)` and `PAR(31)` are used to monitor the optimality functionals associated with the problem parameters and the period. Computations can be started at a solution computed with `IPS`=2 or `IPS`=15. For a detailed example see demo `ops`.

- `IPS=16` : This option is similar to `IPS`=14, except that the user supplies the boundary conditions. Thus this option can be used for time-integration of parabolic systems subject to user-defined boundary conditions. For examples see the first runs of demos `pd1`, `pd2`, and `bru`. Note that the space-derivatives of the initial conditions must also be supplied in the user supplied subroutine `STPNT`. The initial conditions must satisfy the boundary conditions. This option is mainly intended for the detecting stationary solutions.

- `IPS=17` : This option can be used to continue stationary solutions of parabolic systems obtained from an evolution run with `IPS`=16. For examples see the second runs of demos `pd1` and `pd2`.

6.9 Output Control.

6.9.1 NPR

This constant can be used to regularly write `fort.8` plotting and restart data. IF `NPR`>0 then such output is written every `NPR` steps. IF `NPR`=0 or if `NPR`≥`NMX` then no such output is written. Note that special solutions, such as branch points, folds, end points, etc., are always written in `fort.8`. Furthermore, one can specify parameter values where plotting and restart data is to be written; see Section 6.9.4. For these reasons, and to limit the output volume, it is recommended that `NPR` output be kept to a minimum.

6.9.2 IID

This constant controls the amount of diagnostic output printed in `fort.9` : the greater `IID` the more detailed the diagnostic output.

- `IID=0` : Minimal diagnostic output. This setting is not recommended.
- `IID=2` : Regular diagnostic output. This is the recommended value of `IID`.
- `IID=3` : This setting gives additional diagnostic output for algebraic equations, namely the Jacobian and the residual vector at the starting point. This information, which is printed at the beginning of `fort.9`, is useful for verifying whether the starting solution in `STPNT` is indeed a solution.
- `IID=4` : This setting gives additional diagnostic output for differential equations, namely the reduced system and the associated residual vector. This information is printed for every step and for every Newton iteration, and should normally be suppressed. In particular it can be used to verify whether the starting solution is indeed a solution. For this purpose, the stepsize `DS` should be small, and one should look at the residuals printed in the `fort.9` output-file. (Note that the first residual vector printed in `fort.9` may be identically zero, as it may correspond to the computation of the starting direction. Look at the second residual vector in such case.) This residual vector has dimension `NDIM`+`NBC`+`NINT`+1, which

accounts for the NDIM differential equations, the NBC boundary conditions, the NINT user-defined integral constraints, and the pseudo-arclength equation. For proper interpretations of these data one may want to refer to the solution algorithm for solving the collocation system, as described in Doedel, Keller & Kernévez (1991*b*).

- IID=5 : This setting gives very extensive diagnostic output for differential equations, namely, debug output from the linear equation solver. This setting should not normally be used as it may result in a huge fort.9 file.

6.9.3 IPLT

This constant allows redefinition of the principal solution measure, which is printed as the second (real) column in the screen output and in the fort.7 output-file :

- If IPLT $= 0$ then the L_2-norm is printed. Most demos use this setting.
- If $0 <$ IPLT $\leq$ NDIM then the maximum of the IPLT'th solution component is printed.
- If $-$NDIM $\leq$ IPLT <0 then the minimum of the IPLT'th solution component is printed. (Demo fsh.)
- If NDIM $<$ IPLT $\leq$ 2*NDIM then the integral of the (IPLT$-$NDIM)'th solution component is printed.
  (Demos exp, lor.)
- If 2*NDIM $<$ IPLT $\leq$ 3*NDIM then the L_2-norm of the (IPLT$-$NDIM)'th solution component is printed.
  (Demo frc.)

Note that for algebraic problems the maximum and the minimum are identical. Also, for ODEs the maximum and the minimum of a solution component are generally much less accurate than the L_2-norm and component integrals. Note also that the subroutine PVLS provides a second, more general way of defining solution measures; see Section 6.7.10.

6.9.4 NUZR

This constant allows the setting of parameter values at which labeled plotting and restart information is to be written in the fort.8 output-file. Optionally, it also allows the computation to terminate at such a point.

- Set NUZR$=0$ if no such output is needed. Many demos use this setting.

- If `NUZR`>0 then one must enter `NUZR` pairs, *Parameter-Index   Parameter-Value* , with each pair on a separate line, to designate the parameters and the parameter values at which output is to be written. For examples see demos `exp`, `int`, and `fsh`.

- If such a parameter index is preceded by a minus sign then the computation will terminate at such a solution point.
  (Demos `pen` and `bru`.)

Note that `fort.8` output can also be written at selected values of overspecified parameters. For an example see demo `pvl`. For details on overspecified parameters see Section 6.7.10.

Chapter 7

Notes on Using AUTO.

7.1 Restrictions on the Use of `PAR`.

The array `PAR` in the user-supplied subroutines is available for equation parameters that the user wants to vary at some point in the computations. In any particular computation the free parameter(s) must be designated in `ICP`; see Section 6.7. The following restrictions apply :

- The maximum number of parameters, `NPARX` in `auto/97/include/auto.h`, has pre-defined value `NPARX`=36. `NPARX` should not normally be increased and it should never be decreased. Any increase of `NPARX` must be followed by recompilation of AUTO.

- Generally one should only use `PAR(1)-PAR(9)` for equation parameters, as AUTO may need the other components internally.

7.2 Efficiency.

In AUTO, efficiency has at times been sacrificed for generality of programming. This applies in particular to computations in which AUTO generates an extended system, for example, computations with `ISW`=2. However, the user has significant control over computational efficiency, in particular through judicious choice of the AUTO-constants `DS`, `DSMIN`, and `DSMAX`, and, for ODEs, `NTST` and `NCOL`. Initial experimentation normally suggests appropriate values.

Slowly varying solutions to ODEs can often be computed with remarkably small values of `NTST` and `NCOL`, for example, `NTST`=5, `NCOL`=2. Generally, however, it is recommended to set `NCOL`=4, and then to use the "smallest" value of `NTST` that maintains convergence.

The choice of the pseudo-arclength stepsize parameters `DS`, `DSMIN`, and `DSMAX` is highly problem dependent. Generally, `DSMIN` should not be taken too small, in order to prevent excessive step refinement in case of non-convergence. It should also not be too large, in order to avoid instant non-convergence. `DSMAX` should be sufficiently large, in order to reduce computation time and amount of output data. On the other hand, it should be sufficiently small, in order to prevent stepping over bifurcations without detecting them. For a given equation, appropriate values of these constants can normally be found after some initial experimentation.

The constants `ITNW`, `NWTN`, `THL`, `EPSU`, `EPSL`, `EPSS` also affect efficiency. Understanding their significance is therefore useful; see Section 6.4 and Section 6.5. Finally, it is recommended that initial computations be done with `ILP=0`; no fold detection; and `ISP`=1; no bifurcation detection for ODEs.

7.3 Correctness of Results.

AUTO-computed solutions to ODEs are almost always structurally correct, because the mesh adaption strategy, if `IAD`>0, safeguards to some extent against spurious solutions. If these do occur, possibly near infinite-period orbits, the unusual appearance of the solution branch typically serves as a warning. Repeating the computation with increased `NTST` is then recommended.

7.4 Bifurcation Points and Folds.

It is recommended that the detection of folds and bifurcation points be initially disabled. For example, if an equation has a "vertical" solution branch then AUTO may try to locate one fold after another.

Generally, degenerate bifurcations cannot be detected. Furthermore, bifurcations that are close to each other may not be noticed when the pseudo-arclength step size is not sufficiently small. Hopf bifurcation points may go unnoticed if no clear crossing of the imaginary axis takes place. This may happen when there are other real or complex eigenvalues near the imaginary axis and when the pseudo-arclength step is large compared to the rate of change of the critical eigenvalue pair. A typical case is a Hopf bifurcation close to a fold. Similarly, Hopf bifurcations may go undetected if switching from real to complex conjugate, followed by crossing of the imaginary axis, occurs rapidly with respect to the pseudo-arclength step size. Secondary periodic bifurcations may not be detected for similar reasons. In case of doubt, carefully inspect the contents of the diagnostics file `fort.9`.

7.5 Floquet Multipliers.

AUTO extracts an approximation to the linearized Poincaré map from the Jacobian of the linearized collocation system that arises in Newton's method. This procedure is very efficient; the map is computed at negligible extra cost. The linear equations solver of AUTO is described in Doedel, Keller & Kernévez (1991*b*). The actual Floquet multiplier solver was written by Fairgrieve (1994). For a detailed description of the algorithm see Fairgrieve & Jepson (1991).

For periodic solutions, the exact linearized Poincaré map always has a multiplier $z = 1$. A good accuracy check is to inspect this multiplier in the diagnostics output-file `fort.9`. If this multiplier becomes inaccurate then the automatic detection of potential secondary periodic bifurcations (if `ISP`=2) is discontinued and a warning is printed in `fort.9`. It is strongly recommended that the contents of this file be habitually inspected, in particular to verify whether solutions labeled as BP or TR (cf. Table 3.1) have indeed been correctly classified.

7.6 Memory Requirements.

Pre-defined maximum values of certain AUTO-constants are in `auto/97/include/auto.h`; see also Section 1.2. These maxima affect the run-time memory requirements and should not be set to unnecessarily large values. If an application only solves algebraic systems and if `NDIM` is "large" then memory requirements can be much reduced by setting each of `NTSTX`, `NCOLX`, `NBCX`, `NINTX`, equal to 1 in `auto/97/include/auto.h`, followed by recompilation of the AUTO libraries.

Chapter 8

AUTO **Demos : Tutorial.**

8.1 Introduction.

The directory `auto/97/demos` has a large number of subdirectories, for example `ab`, `pp2`, `exp`, etc., each containing all necessary files for certain illustrative calculations. Each subdirectory, say `xxx`, corresponds to a particular equation and contains one equations-file `xxx.f` and one or more constants-files `r.xxx.i`, one for each successive run of the demo. To see how the equations have been programmed, inspect the equations-file. To understand in detail how AUTO is instructed to carry out a particular task, inspect the appropriate constants-file. In this chapter we describe the tutorial demo `ab` in detail. A brief description of other demos is given in later chapters.

8.2 ab : A Tutorial Demo.

This demo illustrates the computation of stationary solutions, Hopf bifurcations and periodic solutions, and the computation loci of folds and Hopf bifurcation points. The equations, that model an A $\rightarrow$ B reaction, are those from Uppal, Ray & Poore (1974), namely

$$\begin{aligned} u_1' &= -u_1 + p_1(1-u_1)e^{u_2}, \\ u_2' &= -u_2 + p_1 p_2 (1-u_1)e^{u_2} - p_3 u_2. \end{aligned} \tag{8.1}$$

8.3 Copying the Demo Files.

The commands listed in Table 8.1 will copy the demo files to your work directory.

UNIX-COMMAND	ACTION
cd	go to your main directory (or other directory)
mkdir ab	create an empty work directory
cd ab	change to the work directory
AUTO-COMMAND	ACTION
@dm ab	copy the demo files to the work directory

Table 8.1: Copying the demo `ab` files.

At this point you may want to see what files have been copied to the work directory. In particular, you may want to edit the equations-file `ab.f` to see how the equations have been entered (in subroutine `FUNC`) and how the starting solution has been set (in subroutine `STPNT`). Note that, initially, $p_1 = 0$ $p_2 = 14$, and $p_3 = 2$, for which $u_1 = u_2 = 0$ is a stationary solution.

8.4 Executing all Runs Automatically.

To execute all prepared runs of demo `ab`, simply type the command given in Table 8.2.

UNIX-COMMAND	ACTION
make	execute all runs of demo `ab`

Table 8.2: Executing all runs of demo `ab`.

The resulting screen output is given on the following page in somewhat abbreviated form. Some differences in output are to be expected on different machines. This does not mean that the results have different accuracy, but simply that arithmetic differences have accumulated from step to step, possibly leading to different step size decisions.

Note that there are five separate runs. In the first run, a branch of stationary solutions is traced out. Along it, two folds (LP) and one Hopf bifurcation (HB) are located. The free parameter is p_1. The other parameters remain fixed in this run. Note also that only special, labeled solution points are printed on the screen. More detailed results are saved in the data-files `p.ab`, `q.ab`, and `d.ab`.

The second run traces out the branch of periodic solutions that emanates from the Hopf bifurcation. The free parameters are p_1 and the period. The detailed results are appended to the existing data-files `p.ab`, `q.ab`, and `d.ab`.

In the third run, one of the folds detected in the first run is followed in the two parameters p_1 and p_3, while p_2 remains fixed. The fourth run continues this branch in opposite direction. Similarly, in the fifth run, the Hopf bifurcation located in the first run is followed in the two parameters p_1 and p_3. (In this example this is done in one direction only.) The detailed results of these continuations are accumulated in the data-files `p.2p`, `q.2p`, and `d.2p`.

One could now use PLAUT to graphically inspect the contents of the data-files, but we shall do this later. However, it may be useful to edit these files to view their contents.

Next, reset the work directory, by typing the command given in Table 8.3.

UNIX-COMMAND	ACTION
make clean	remove data-files and temporary files of demo `ab`

Table 8.3: Cleaning the demo `ab` work directory.

```
ab : first run : stationary solutions

BR    PT  TY LAB    PAR(1)        L2-NORM        U(1)          U(2)
 1     1  EP   1  0.000000E+00  0.000000E+00  0.000000E+00  0.000000E+00
 1    33  LP   2  1.057390E-01  1.484391E+00  3.110230E-01  1.451441E+00
 1    70  LP   3  8.893185E-02  3.288241E+00  6.889822E-01  3.215250E+00
 1    90  HB   4  1.308998E-01  4.271867E+00  8.950803E-01  4.177042E+00
 1    92  EP   5  1.512417E-01  4.369748E+00  9.155894E-01  4.272750E+00
 Saved as *.ab

ab : second run : periodic solutions

BR    PT  TY LAB    PAR(1)        L2-NORM       MAX U(1)      MAX U(2)       PERIOD
 4    30       6  1.198815E-01  3.987129E+00  9.919113E-01  7.020342E+00  2.721044E+00
 4    60       7  1.153033E-01  3.146307E+00  9.995775E-01  9.957649E+00  6.147449E+00
 4    90       8  1.056503E-01  2.219179E+00  9.991661E-01  9.366097E+00  1.399772E+01
 4   120       9  1.055071E-01  1.696847E+00  9.990869E-01  9.296297E+00  9.956218E+01
 4   150  EP  10  1.055071E-01  1.603885E+00  9.997894E-01  9.281460E+00  1.867100E+03
 Appended to *.ab

ab : third run : a 2-parameter locus of folds

BR    PT  TY LAB    PAR(1)        L2-NORM        U(1)          U(2)         PAR(3)
 2    27  LP  11  1.353352E-01  2.060123E+00  4.996530E-01  1.998613E+00  2.499998E+00
 2   100  EP  12  1.093816E-08  2.136506E+01  9.531479E-01  2.134378E+01 -3.748029E-01
 Saved as *.2p

ab : fourth run : the locus of folds in reverse direction

BR    PT  TY LAB    PAR(1)        L2-NORM        U(1)          U(2)         PAR(3)
 2    35  EP  11 -1.319397E-03  9.964328E-01 -3.586515E-03  9.964263E-01 -1.050391E+00
 Appended to *.2p

ab : fifth run : a 2-parameter locus of Hopf points

BR    PT  TY LAB    PAR(1)        L2-NORM        U(1)          U(2)         PAR(3)
 4   100  EP  11  8.809407E-05  1.174404E+01  9.146095E-01  1.170837E+01  9.362239E-02
 Appended to *.2p
```

8.5 Executing Selected Runs Automatically.

As illustrated by the commands in Table 8.4, one can also execute selected runs of demo `ab`. In general, this cannot be done in arbitrary order, as any given run may need restart data from a previous run. Run 3 only requires the results of Run 1, so that the displayed command sequence is indeed appropriate. The screen output of these runs will be identical to that of the corresponding earlier runs, except for a change in solution labels in Run 3.

Unix-COMMAND	ACTION
make first	execute the first run of demo `ab`
make third	execute the third run of demo `ab`

Table 8.4: Selected runs of demo `ab`.

Of course, in real use, one must prepare a constants-file for each run. In the illustrative runs above, the constants-files were carefully prepared in advance. For example, the file `r.ab.1` contains the AUTO-constants for Run 1, `r.ab.3` contains the AUTO-constants for Run 3, etc.

8.6 Using AUTO-Commands.

Next, with the commands in Table 8.5, we execute the first two runs of demo `ab` again, but now using the commands that one would normally use in an actual application. We still use the demo constants-files that were prepared in advance.

COMMAND	ACTION
make clean	reset the work directory
cp r.ab.1 r.ab	get the first constants-file
@r ab	compute a stationary solution branch with folds and Hopf bifurcation
@sv ab	save output-files as `p.ab, q.ab, d.ab`
cp r.ab.2 r.ab	get the second constants-file
@r ab	compute a branch of periodic solutions from the Hopf point
@ap ab	append the output-files to `p.ab, q.ab, d.ab`

Table 8.5: Commands for Run 1 and Run 2 of demo `ab`.

It is instructive to look at the constants-files `r.ab.1` and `r.ab.2` used in the two runs above. The significance of each AUTO-constant set in these files can be found in Chapter 6. Note in particular the AUTO-constants that were changed between the two runs; see Table 8.6.

Constant	Run 1	Run 2	Reason for Change
`IPS`	1	2	To compute periodic solutions in Run 2
`IRS`	0	4	To specify the Hopf bifurcation restart label
`NICP`	1	2	The second run has two free parameters
`ICP`	1	1, 11	To use and print `PAR(1)` and `PAR(11)` in Run 2
`NMX`	100	150	To allow more continuation steps in Run 2
`NPR`	100	30	To print output every 30 steps in Run 2

Table 8.6: Differences in AUTO-constants between `r.ab.1` and `r.ab.2`.

Actually, for periodic solutions, AUTO automatically adds `PAR(11)` (the period) as second parameter. However, for the period to be printed, one must specify the index 11 in the `ICP` list, as shown in Table 8.6.

8.7 Plotting the Results with PLAUT.

The bifurcation diagram computed in the runs above is stored in the file `p.ab`, while each labeled solution is fully stored in `q.ab`. To use PLAUT to graphically inspect these data-files, type the AUTO-command given in Table 8.7. The PLAUT window (a Tektronix window) will appear, in which one can enter the PLAUT-commands given in Table 8.8. The saved plots are shown in Figure 8.1 and in Figure 8.2.

AUTO-COMMAND	ACTION
@p ab	run PLAUT to graph the contents of `p.ab` and `q.ab`;

Table 8.7: Command for plotting the files `p.ab` and `q.ab`.

8.8 Following Folds and Hopf Bifurcations.

The commands in Table 8.9 will execute the remaining runs of demo `ab`. Here, as in later demos, some of the AUTO-constants that have been changed between runs are indicated in the Table.

PLAUT-COMMAND	ACTION
d1	choose one of the default settings
bd0	plot the default bifurcation diagram; L_2-norm versus p_1
bd	make a blow-up of current bifurcation diagram
.08 .14 .5 4.5	enter diagram limits
sav	save the current plot
fig.1	upon prompt, enter a new file name, e.g., `fig.1`
2d	enter 2D mode, for plotting labeled solutions
6 7 10	select labeled orbits 6, 7, and 10 in `q.ab`
d	default orbit display; u_1 versus scaled time
1 3	select columns 1 and 3 in `q.ab`
d	display the orbits; u_2 versus scaled time
2 3	select columns 2 and 3 in `q.ab`
d	phase plane display; u_2 versus u_1
sav	save the current plot
fig.2	upon prompt, enter a new file name
ex	exit from 2D mode
end	exit from PLAUT

Table 8.8: Commands to be typed in the PLAUT window.

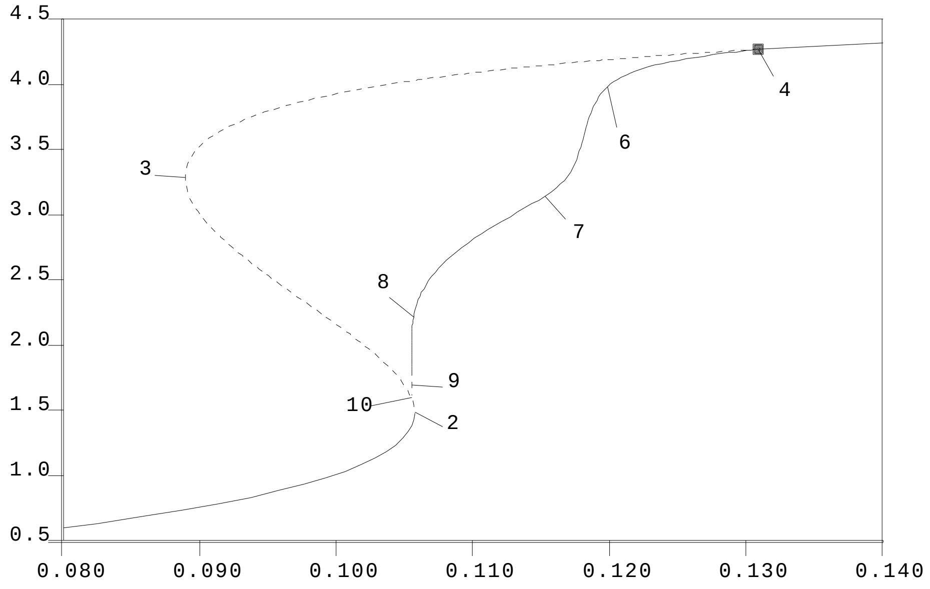

Figure 8.1: The bifurcation diagram of demo `ab`.

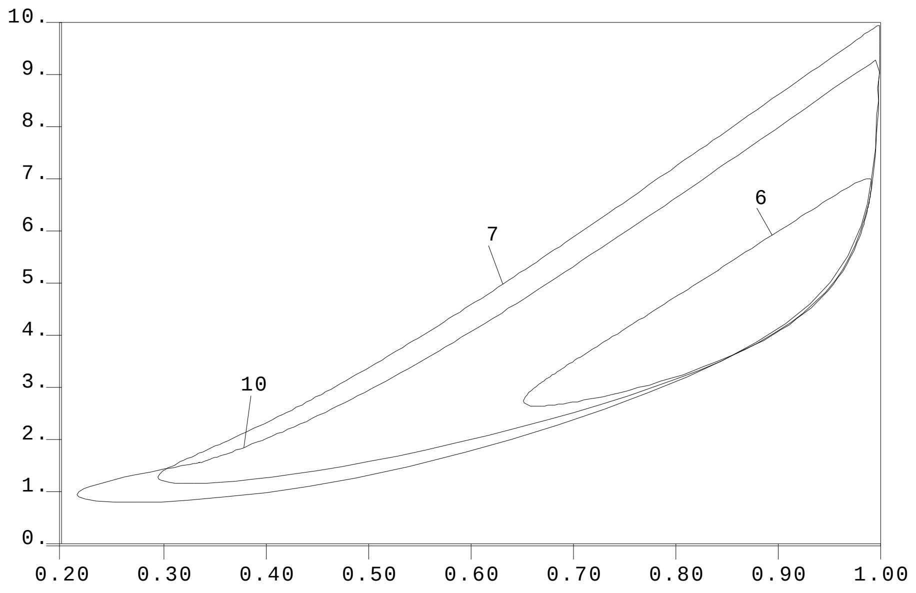

Figure 8.2: The phase plot of solutions 6, 7, and 10 in demo `ab`.

COMMAND	ACTION
cp r.ab.3 r.ab	changes (from `r.ab.1`) : IRS, NICP, ICP, ISW, DSMAX
@r ab	compute a locus of folds
@sv 2p	save output-files as `p.2p, q.2p, d.2p`
cp r.ab.4 r.ab	changes (from `r.ab.3`) : DS (sign)
@r ab	compute the locus of folds in reverse direction
@ap 2p	append the output-files to `p.2p, q.2p, d.2p`
cp r.ab.5 r.ab	changes (from `r.ab.4`) : IRS
@r ab	compute a locus of Hopf points
@ap 2p	append the output-files to `p.2p, q.2p, d.2p`

Table 8.9: Commands for Runs 3, 4, and 5 of demo `ab`.

8.9 Relabeling Solutions in the Data-Files.

Next we want to plot the two-parameter diagram computed in the last three runs. However, the solution labels in these runs are not distinct. This is due to the fact that in each of these three runs the restart solution was read from `q.ab`, while the computed solutions were stored in `q.2p`. Consequently, these runs were unaware of each other's results, which led to non-unique labels. For relabeling purpose, and more generally for file maintenance, there is a utility program that can be invoked as indicated in Table 8.10. Its use is illustrated in Table 8.11.

AUTO-COMMAND	ACTION
@lb 2p	run the relabeling program on `p.2p` and `q.2p`

Table 8.10: Command to run the relabeling program on `p.2p` and `q.2p`.

RELABELING COMMAND	ACTION
l	list the labeled solutions in `q.2p`
r	relabel the solutions
l	list the new solution labeling
w	rewrite `p.2p` and `q.2p`

Table 8.11: Relabeling commands for the files `p.2p` and `q.2p`.

8.10 Plotting the 2-Parameter Diagram.

To run PLAUT on the files `p.2p` and `q.2p`, enter the command listed in Table 8.12. The PLAUT-commands for plotting the two-parameter diagram are then as given in Table 8.13. The saved plot is shown in Figure 8.3.

AUTO-COMMAND	ACTION
@p 2p	run PLAUT to graph the contents of `p.2p` and `q.2p`;

Table 8.12: Command to run PLAUT for files `p.2p` and `q.2p`.

PLAUT-COMMAND	ACTION
d0	set default option
ax	select axes
1 5	select real columns 1 (p_1) and 5 (p_3) in `p.2p`
bd0	plot the 2-parameter diagram; p_3 versus p_1
cl	clear the screen
d2	set other default option
bd0	plot the 2-parameter diagram; p_3 versus p_1
bd	make a blow-up of the current diagram
0 .15 0 2.5	enter diagram limits
sav	save plot
fig.3	upon prompt, enter a new file name, e.g., `fig.3`
end	exit from PLAUT

Table 8.13: PLAUT-commands for files `p.2p` and `q.2p`.

8.11 Converting Saved PLAUT Files to PostScript.

Plots are saved in compact Tektronix PLOT10 format. In Table 8.14 it is shown how such files can be converted to PostScript format. Note that the latter files are much bigger.

AUTO-COMMAND	ACTION
@ps fig.1	convert file `fig.1` into PostScript file `fig.1.ps`
lpr fig.1.ps	system dependent : print `fig.1.ps` on your printer
@ps fig.2	convert file `fig.2` into PostScript file `fig.2.ps`
lpr fig.2.ps	system dependent : print `fig.2.ps` on your printer
@ps fig.3	convert file `fig.3` into PostScript file `fig.3.ps`
lpr fig.3.ps	system dependent : print `fig.3.ps` on your printer

Table 8.14: Printing commands for the saved Figures in demo `ab`.

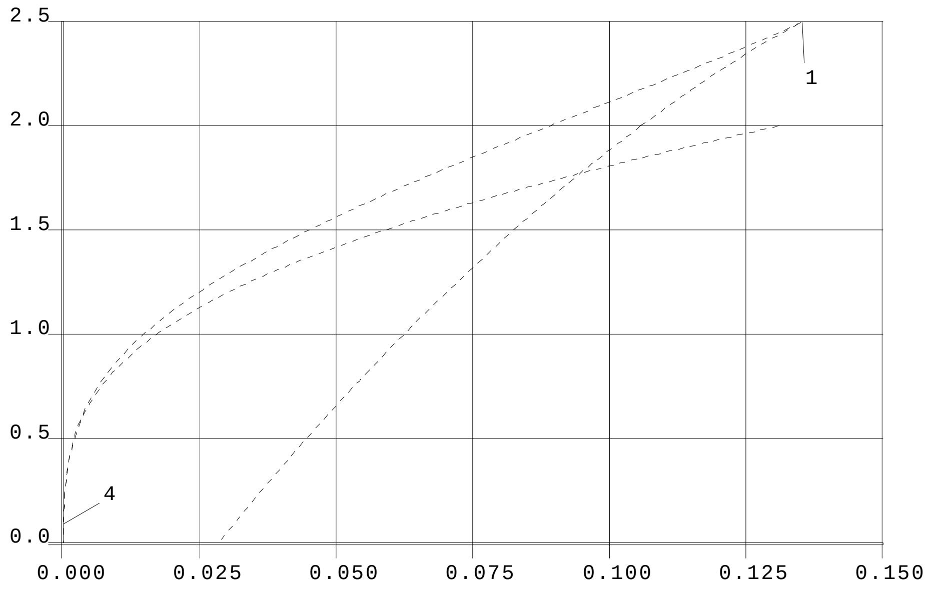

Figure 8.3: Loci of folds and Hopf bifurcations for demo `ab`.

8.12 Using the GUI.

Demos can also be run using the GUI. See Table 5.1 for the correspondence between Command Mode and GUI actions. To activate the GUI, type the command in Table 8.15. The GUI actions to execute the first two runs of demo `ab` are given in Table 8.16. In GUI Mode one can copy demo files to the user work directory using the *Equations/Demo*-button. To load a selected constants-file, use the *Previous*-button in the *LoadConsts* area of the GUI window. Press the *Filter*-button in the pop-up window to update the displayed list of files, and then select the appropriate constants-file.

AUTO-COMMAND	ACTION
@auto	Activate the Graphical User Interface

Table 8.15: Command to activate the GUI.

To execute all runs of a selected demo with the GUI, click *Demos/Select*, select a demo, and click the *Run*-button in the pop-up window. This will actually run the demo in the corresponding subdirectory of `auto/97/demos`, which is only possible if you have write access to this directory. Make sure to click the *Demos/Reset*-button afterwards. Do not otherwise run AUTO in the directory `auto/97` or in any of its subdirectories.

GUI-button	ACTION
Equations/Demo	Select demo `ab`, then press *OK*
Previous	Push *Filter*, select file `r.ab.1`, then press *OK*
Run	This will execute Run 1 of demo `ab`
Save/Save	Save the output files as `p.ab`, `q.ab`, `d.ab`
Previous	Select file `r.ab.2`, then press *OK*
Run	This will execute Run 2 of demo `ab`
Append/Append	Append the output-files to `p.ab`, `q.ab`, `d.ab`

Table 8.16: GUI actions for Run 1 and Run 2 of demo `ab`.

8.13 Abbreviated AUTO-Commands.

The AUTO-commands given in, for example, Table 8.5 can be simplified by using the *@R* command. For Table 8.5 the equivalent command sequence is given in Table 8.17.

COMMAND	ACTION
make clean	reset the work directory
@R ab 1	(reads AUTO-constants from `r.ab.1`)
@sv ab	save output-files as `p.ab, q.ab, d.ab`
@R ab 2	(reads AUTO-constants from `r.ab.2`)
@ap ab	append the output-files to `p.ab, q.ab, d.ab`

Table 8.17: Abbreviated AUTO-commands.

Chapter 9

AUTO Demos : Fixed points.

9.1 enz : Stationary Solutions of an Enzyme Model.

The equations, that model a two-compartment enzyme system (Kernévez (1980)), are given by

$$\begin{aligned} s_1' &= (s_0 - s_1) + (s_2 - s_1) - \rho R(s_1), \\ s_2' &= (s_0 + \mu - s_2) + (s_1 - s_2) - \rho R(s_2), \end{aligned} \tag{9.1}$$

where

$$R(s) = \frac{s}{1 + s + \kappa s^2}.$$

The free parameter is s_0. Other parameters are fixed. This equation is also considered in Doedel, Keller & Kernévez (1991*a*).

COMMAND	ACTION
mkdir enz	create an empty work directory
cd enz	change directory
@dm enz	copy the demo files to the work directory
cp r.enz.1 r.enz	get the constants-file
@r enz	compute stationary solution branches
@sv enz	save output-files as `p.enz, q.enz, d.enz`

Table 9.1: Commands for running demo `enz`.

9.2 dd2 : Fixed Points of a Discrete Dynamical System.

This demo illustrates the computation of a solution branch and its bifurcating branches for a discrete dynamical system. Also illustrated is the continuation of Naimark-Sacker (or Hopf) bifurcations The equations, a discrete predator-prey system, are

$$\begin{aligned} u_1^{k+1} &= p_1 u_1^k (1 - u_1^k) - p_2 u_1^k u_2^k, \\ u_2^{k+1} &= (1 - p_3) u_2^k + p_2 u_1^k u_2^k. \end{aligned} \tag{9.2}$$

In the first run p_1 is free. In the second run, both p_1 and p_2 are free. The remaining equation parameter, p_3, is fixed in both runs.

AUTO-COMMAND	ACTION
mkdir dd2	create an empty work directory
cd dd2	change directory
@dm dd2	copy the demo files to the work directory
cp r.dd2.1 r.dd2	get the first constants-file
@r dd2	1st run; fixed point solution branches
@sv dd2	save output-files as `p.dd2, q.dd2, d.dd2`
cp r.dd2.2 r.dd2	constants changed : `IRS, ISW`
@r dd2	2nd run; a locus of Naimark-Sacker bifurcations
@sv ns	save output-files as `p.ns, q.ns, d.ns`

Table 9.2: Commands for running demo `dd2`.

Chapter 10

AUTO **Demos : Periodic solutions.**

10.1 lrz : The Lorenz Equations.

This demo computes two symmetric homoclinic orbits in the Lorenz equations

$$\begin{aligned} u_1' &= p_3(u_2 - u_1), \\ u_2' &= p_1 u_1 - u_2 - u_1 u_3, \\ u_3' &= u_1 u_2 - p_2 u_3. \end{aligned} \tag{10.1}$$

Here p_1 is the free parameter, and $p_2 = 8/3$, $p_3 = 10$. The two homoclinic orbits correspond to the final, large period orbits on the two periodic solution branches.

COMMAND	ACTION
mkdir lrz	create an empty work directory
cd lrz	change directory
@dm lrz	copy the demo files to the work directory
cp r.lrz.1 r.lrz	get the first constants-file
@r lrz	compute stationary solutions
@sv lrz	save output-files as `p.lrz, q.lrz, d.lrz`
cp r.lrz.2 r.lrz	constants changed : `IPS, IRS, NICP, ICP, NMX, NPR, DS`
@r lrz	compute periodic solutions; the final orbit is near-homoclinic
@ap lrz	append the output-files to `p.lrz, q.lrz, d.lrz`
cp r.lrz.3 r.lrz	constants changed : `IRS`
@r lrz	compute the symmetric periodic solution branch
@ap lrz	append the output-files to `p.lrz, q.lrz, d.lrz`

Table 10.1: Commands for running demo `lrz`.

10.2 abc : The A $\rightarrow$ B $\rightarrow$ C Reaction.

This demo illustrates the computation of stationary solutions, Hopf bifurcations and periodic solutions in the A $\rightarrow$ B $\rightarrow$ C reaction (Doedel & Heinemann (1983)).

$$\begin{aligned} u_1' &= -u_1 + p_1(1-u_1)e^{u_3}, \\ u_2' &= -u_2 + p_1 e^{u_3}(1-u_1-p_5 u_2), \\ u_3' &= -u_3 - p_3 u_3 + p_1 p_4 e^{u_3}(1-u_1+p_2 p_5 u_2), \end{aligned} \tag{10.2}$$

with $p_2 = 1$, $p_3 = 1.55$, $p_4 = 8$, and $p_5 = 0.04$. The free parameter is p_1.

COMMAND	ACTION
mkdir abc	create an empty work directory
cd abc	change directory
@dm abc	copy the demo files to the work directory
cp r.abc.1 r.abc	get the first constants-file
@r abc	compute the stationary solution branch with Hopf bifurcations
@sv abc	save output-files as `p.abc, q.abc, d.abc`
cp r.abc.2 r.abc	constants changed : `IRS, IPS, NICP, ICP`
@r abc	compute a branch of periodic solutions from the first Hopf point
@ap abc	append the output-files to `p.abc, q.abc, d.abc`
cp r.abc.3 r.abc	constants changed : `IRS, NMX`
@r abc	compute a branch of periodic solutions from the second Hopf point
@ap abc	append the output-files to `p.abc, q.abc, d.abc`

Table 10.2: Commands for running demo `abc`.

10.3 pp2 : Bifurcations in a 2D Predator-Prey Model.

This demo illustrates a variety of calculations. The equations, which model a predator-prey system with harvesting, are

$$\begin{aligned} u_1' &= p_2 u_1(1-u_1) - u_1 u_2 - p_1(1-e^{-p_3 u_1}), \\ u_2' &= -u_2 + p_4 u_1 u_2. \end{aligned} \tag{10.3}$$

Here p_1 is the principal continuation parameter, $p_3 = 5$, $p_4 = 3$, and, initially, $p_2 = 3$. For two-parameter computations p_2 is also free. The use of PLAUT is also illustrated. The saved plots are shown in Figure 10.1 and Figure 10.2.

COMMAND	ACTION
mkdir pp2	create an empty work directory
cd pp2	change directory
@dm pp2	copy the demo files to the work directory
cp r.pp2.1 r.pp2	get the first constants-file
@r pp2	1st run; stationary solutions
@sv pp2	save output-files as `p.pp2, q.pp2, d.pp2`
cp r.pp2.2 r.pp2	constants changed : `IRS, RL1`
@r pp2	2nd run; restart at a labeled solution
@ap pp2	append output-files to `p.pp2, q.pp2, d.pp2`
cp r.pp2.3 r.pp2	constants changed : `IRS, IPS, ILP`
@r pp2	3rd run; periodic solutions
@ap pp2	append output-files to `p.pp2, q.pp2, d.pp2`
cp r.pp2.4 r.pp2	constants changed : `IRS, NTST`
@r pp2	4th run; restart at a labeled periodic solution
@ap pp2	append output-files to `p.pp2, q.pp2, d.pp2`
cp r.pp2.5 r.pp2	constants changed : `IRS, IPS, ISW, ICP`
@r pp2	5th run; continuation of folds
@sv lp	save output-files as `p.lp, q.lp, d.lp`
cp r.pp2.6 r.pp2	constants changed : `IRS`
@r pp2	6th run; continuation of Hopf bifurcations
@sv hb	save output-files as `p.hb, q.hb, d.hb`
cp r.pp2.7 r.pp2	constants changed : `IRS, IPS, ISP`
@r pp2	7th run; continuation of homoclinic orbits
@sv hom	save output-files as `p.hom, q.hom, d.hom`

Table 10.3: Commands for running demo pp2.

AUTO-COMMAND	ACTION
@p pp2	run PLAUT to graph the contents of `p.pp2` and `q.pp2`;
PLAUT-COMMAND	ACTION
d1	set convenient defaults
bd0	plot the default bifurcation diagram; L_2-norm versus p_1
cl	clear the screen
ax	select axes
1 3	select real columns 1 and 3 in `p.pp2`
bd0	plot the bifurcation diagram; *max* u_1 versus p_1
cl	clear the screen
d3	choose other default settings
bd0	bifurcation diagram
bd	get blow-up of current bifurcation diagram
0 1 -0.25 1	enter diagram limits
sav	save plot (see Figure 10.1)
fig.1	upon prompt, enter a new file name, e.g., `fig.1`
cl	clear the screen
2d	enter 2D mode, for plotting labeled solutions
12 13 14	select labeled orbits 12, 13, and 14 in `q.pp2`
d	default orbit display; u_1 versus time
1 3	select columns 1 and 3 in `q.pp2`
d	display the orbits; u_2 versus time
2 3	select columns 2 and 3 in `q.pp2`
d	phase plane display; u_2 versus u_1
sav	save plot (see Figure 10.2)
fig.2	upon prompt, enter a new file name
ex	exit from 2D mode
end	exit from PLAUT

Table 10.4: Plotting commands for demo `pp2`.

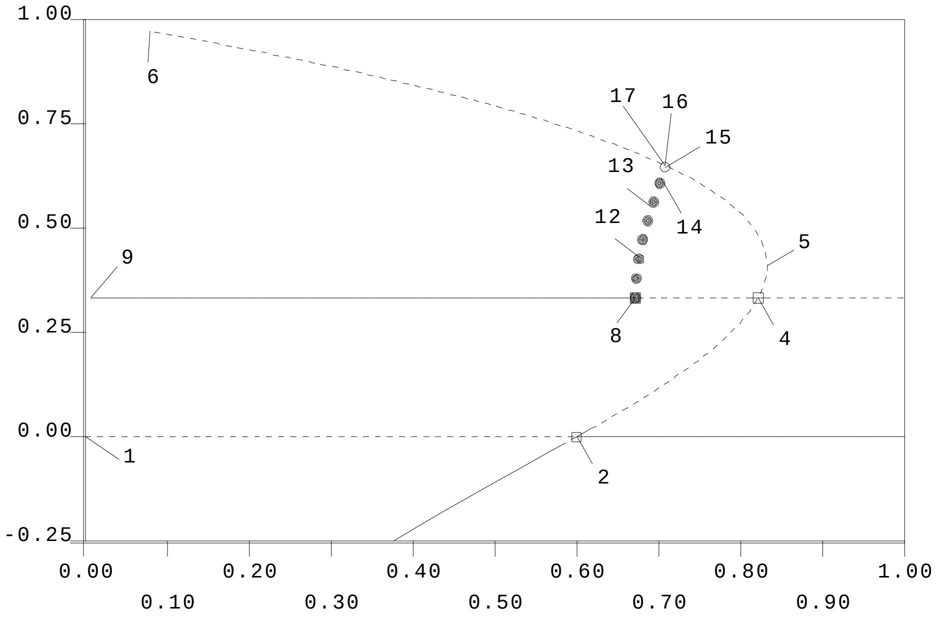

Figure 10.1: The bifurcation diagram of demo pp2.

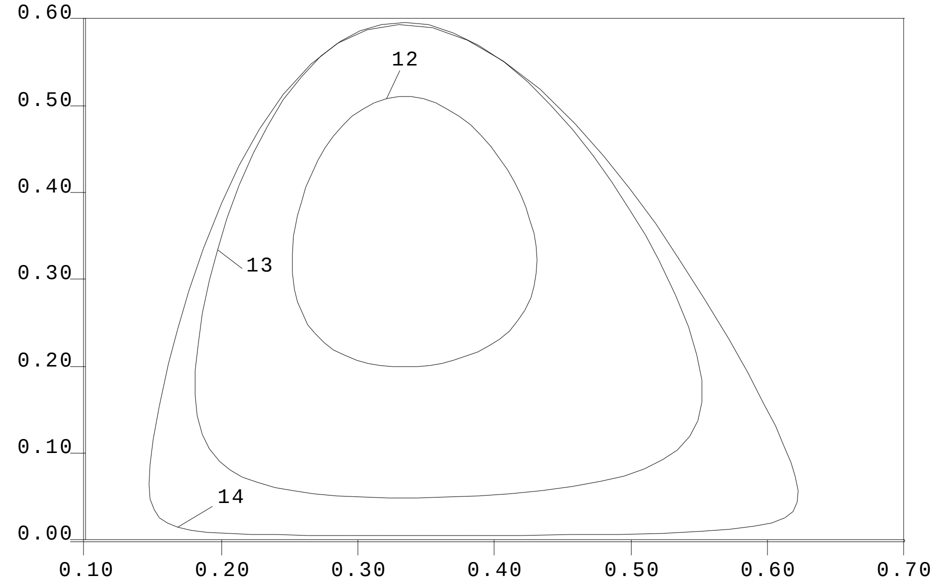

Figure 10.2: The phase plot of solutions 12, 13, and 14 in demo pp2.

10.4 lor : Starting an Orbit from Numerical Data.

This demo illustrates how to start the computation of a branch of periodic solutions from numerical data obtained, for example, from an initial value solver. As an illustrative application we consider the Lorenz equations

$$\begin{aligned} u_1' &= p_3(u_2 - u_1), \\ u_2' &= p_1 u_1 - u_2 - u_1 u_3, \\ u_3' &= u_1 u_2 - p_2 u_3. \end{aligned} \tag{10.4}$$

Numerical simulations with a simple initial value solver show the existence of a stable periodic orbit when $p_1 = 280$, $p_2 = 8/3$, $p_3 = 10$. Numerical data representing one complete periodic oscillation are contained in the file `lor.dat`. Each row in `lor.dat` contains four real numbers, namely, the time variable t, u_1, u_2 and u_3. The correponding parameter values are defined in the user-supplied subroutine `STPNT`. The AUTO-command *@fc lor* then converts the data in `lor.dat` to a labeled AUTO solution (with label 1) in a new file `q.dat`. The mesh will be suitably adapted to the solution, using the number of mesh intervals `NTST` and the number of collocation point per mesh interval `NCOL` specified in the constants-file `r.lor`. (Note that the file `q.dat` should be used for restart only. Do not append new output-files to `q.dat`, as the command *@fc lor* only creates `q.dat`, with no corresponding `p.dat`.)

COMMAND	ACTION
mkdir lor	create an empty work directory
cd lor	change directory
@dm lor	copy the demo files to the work directory
cp r.lor.1 r.lor	get the first constants-file
@fc lor	convert `lor.dat` to AUTO format in `q.dat`
@r lor dat	compute a solution branch, restart from `q.dat`
@sv lor	save output-files as `p.lor, q.lor, d.lor`
cp r.lor.2 r.lor	constants changed : `IRS, ISW, NTST`
@r lor	switch branches at a period-doubling detected in the first run
@ap lor	append the output-files to `p.lor, q.lor, d.lor`

Table 10.5: Commands for running demo `lor`.

10.5 frc : A Periodically Forced System.

This demo illustrates the computation of periodic solutions to a periodically forced system. In AUTO this can be done by adding a nonlinear oscillator with the desired periodic forcing as one of the solution components. An example of such an oscillator is

$$\begin{aligned} x' &= x + \beta y - x(x^2 + y^2), \\ y' &= -\beta x + y - y(x^2 + y^2), \end{aligned} \tag{10.5}$$

which has the asymptotically stable solution $x = sin(\beta t)$, $y = cos(\beta t)$. We couple this oscillator to the Fitzhugh-Nagumo equations :

$$\begin{aligned} v' &= (F(v) - w)/\epsilon, \\ w' &= v - dw - (b + r\sin(\beta t)), \end{aligned} \tag{10.6}$$

by replacing $\sin(\beta t)$ by x. Above, $F(v) = v(v - a)(1 - v)$ and a, b, ϵ and d are fixed. The first run is a homotopy from $r = 0$, where a solution is known analytically, to $r = 0.2$. Part of the solution branch with $r = 0.2$ and varying β is computed in the second run. For detailed results see Alexander, Doedel & Othmer (1990).

COMMAND	ACTION
mkdir frc	create an empty work directory
cd frc	change directory
@dm frc	copy the demo files to the work directory
cp r.frc.1 r.frc	get the first constants-file
@r frc	homotopy to $r = 0.2$
@sv 0	save output-files as `p.0, q.0, d.0`
cp r.frc.2 r.frc	constants changed : `IRS, ICP(1), NTST, NMX, DS, DSMAX`
@r frc 0	compute solution branch; restart from `q.0`
@sv frc	save output-files as `p.frc, q.frc, d.frc`

Table 10.6: Commands for running demo `frc`.

10.6 ppp : Continuation of Hopf Bifurcations.

This demo illustrates the continuation of Hopf bifurcations in a 3-dimensional predator prey model (Doedel (1984)). This curve contain branch points, where one locus of Hopf points bifurcates from another locus of Hopf points. The equations are

$$\begin{aligned} u_1' &= u_1(1-u_1) - p_4 u_1 u_2, \\ u_2' &= -p_2 u_2 + p_4 u_1 u_2 - p_5 u_2 u_3 - p_1(1-e^{-p_6 u_2}) \\ u_3' &= -p_3 u_3 + p_5 u_2 u_3. \end{aligned} \tag{10.7}$$

Here $p_2 = 1/4$, $p_3 = 1/2$, $p_4 = 3$, $p_5 = 3$, $p_6 = 5$, and p_1 is the free parameter. In the continuation of Hopf points the parameter p_4 is also free.

COMMAND	ACTION
mkdir ppp	create an empty work directory
cd ppp	change directory
@dm ppp	copy the demo files to the work directory
cp r.ppp.1 r.ppp	get the first constants-file
@r ppp	compute stationary solutions; detect Hopf bifurcations
@sv ppp	save output-files as `p.ppp, q.ppp, d.ppp`
cp r.ppp.2 r.ppp	constants changed : `IPS, IRS, ICP`, etc.
@r ppp	compute a branch of periodic solutions
@ap ppp	append the output-files to `p.ppp, q.ppp, d.ppp`
cp r.ppp.3 r.ppp	constants changed :
@r ppp	compute Hopf bifurcation curves
@sv hb	save the output-files as `p.hb, q.hb, d.hb`

Table 10.7: Commands for running demo `ppp`.

10.7 plp : Fold Continuation for Periodic Solutions.

This demo, which corresponds to computations in Doedel, Keller & Kernévez (1991*a*), shows how one can continue a fold on a branch of periodic solution in two parameters. The equations, that model a one-compartment activator-inhibitor system (Kernévez (1980)), are given by

$$\begin{aligned} s' &= (s_0 - s) - \rho R(s,a), \\ a' &= \alpha(a_0 - a) - \rho R(s,a), \end{aligned} \tag{10.8}$$

where

$$R(s,a) = \frac{sa}{1 + s + \kappa s^2}, \qquad \kappa > 0.$$

The free parameter is ρ. In the fold continuation s_0 is also free.

COMMAND	ACTION
mkdir plp	create an empty work directory
cd plp	change directory
@dm plp	copy the demo files to the work directory
cp r.plp.1 r.plp	get the first constants-file
@r plp	1st run; compute a stationary solution branch
@sv plp	save output-files as `p.plp, q.plp, d.plp`
cp r.plp.2 r.plp	constants changed : `IPS, IRS, NMX`
@r plp	compute a branch of periodic solutions and locate a fold
@ap plp	append output-files to `p.plp, q.plp, d.plp`
cp r.plp.3 r.plp	constants changed : `IRS, ISW, NMX`
@r plp	generate starting data for the fold continuation
@sv tmp	save output-files as `p.tmp, q.tmp, d.tmp`
cp r.plp.4 r.plp	constants changed : `IRS, NUZR`
@r plp tmp	fold continuation; restart data from `q.tmp`
@sv 2p	save output-files as `p.2p, q.2p, d.2p`
cp r.plp.5 r.plp	constants changed : `IRS, ISW, NMX, NUZR`
@r plp 2p	compute an isola of periodic solutions; restart data from `q.2p`
@sv iso	save output-files as `p.iso, q.iso, d.iso`

Table 10.8: Commands for running demo `plp`.

10.8 pp3 : Period-Doubling Continuation.

This demo illustrates the computation of stationary solutions, Hopf bifurcations, and periodic solutions, branch switching at a period-doubling bifurcation, and the computation of a locus of period-doubling bifurcations. The equations model a 3D predator-prey system with harvesting (Doedel (1984)).

$$\begin{aligned} u_1' &= u_1(1-u_1) - p_4 u_1 u_2, \\ u_2' &= -p_2 u_2 + p_4 u_1 u_2 - p_5 u_2 u_3 - p_1(1-e^{-p_6 u_2}) \\ u_3' &= -p_3 u_3 + p_5 u_2 u_3. \end{aligned} \tag{10.9}$$

The free parameter is p_1, except in the period-doubling continuation, where both p_1 and p_2 are free.

COMMAND	ACTION
mkdir pp3	create an empty work directory
cd pp3	change directory
@dm pp3	copy the demo files to the work directory
cp r.pp3.1 r.pp3	get the first constants-file
@r pp3	1st run; stationary solutions
@sv pp3	save output-files as `p.pp3, q.pp3, d.pp3`
cp r.pp3.2 r.pp3	constants changed : `IRS, IPS, NMX`
@r pp3	compute a branch of periodic solutions
@ap pp3	append output-files to `p.pp3, q.pp3, d.pp3`
cp r.pp3.3 r.pp3	constants changed : `IRS, ISW, NTST`
@r pp3	compute the branch bifurcating at the period-doubling
@ap pp3	append output-files to `p.pp3, q.pp3, d.pp3`
cp r.pp3.4 r.pp3	constants changed : `ISW`
@r pp3	generate starting data for the period-doubling continuation
@sv tmp	save output-files as `p.tmp, q.tmp, d.tmp`
cp r.pp3.5 r.pp3	constants changed : `IRS`
@r pp3 tmp	period-doubling continuation; restart from `q.tmp`
@sv 2p	save output-files as `p.2p, q.2p, d.2p`

Table 10.9: Commands for running demo pp3.

10.9 tor : Detection of Torus Bifurcations.

This demo uses a model in Freire, Rodríguez-Luis, Gamero & Ponce (1993) to illustrate the detection of a torus bifurcation. It also illustrates branch switching at a secondary periodic bifurcation with double Floquet multiplier at $z = 1$. The computational results also include folds, homoclinic orbits, and period-doubling bifurcations. Their continuation is not illustrated here; see instead the demos `plp`, `pp2`, and `pp3`, respectively. The equations are

$$\begin{aligned} x'(t) &= [-(\beta+\nu)x + \beta y - a_3 x^3 + b_3(y-x)^3]/r, \\ y'(t) &= \beta x - (\beta+\gamma)y - z - b_3(y-x)^3, \\ z'(t) &= y, \end{aligned} \tag{10.10}$$

where $\gamma = -0.6$, $r = 0.6$, $a_3 = 0.328578$, and $b_3 = 0.933578$. Initially $\nu = -0.9$ and $\beta = 0.5$.

COMMAND	ACTION
mkdir tor	create an empty work directory
cd tor	change directory
@dm tor	copy the demo files to the work directory
cp r.tor.1 r.tor	get the first constants-file
@r tor	1st run; compute a stationary solution branch with Hopf bifurcation
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.tor.2 r.tor	constants changed : `IPS, IRS`
@r tor 1	compute a branch of periodic solutions; restart from `q.1`
@ap 1	append output-files to `p.1, q.1, d.1`
cp r.tor.3 r.tor	constants changed : `IRS, ISW, NMX`
@r tor 1	compute a bifurcating branch of periodic solutions; restart from `q.1`
@ap 1	append output-files to `p.1, q.1, d.1`

Table 10.10: Commands for running demo `tor`.

10.10 pen : Rotations of Coupled Pendula.

This demo illustrates the computation of rotations, i.e., solutions that are periodic, modulo a phase gain of an even multiple of π. AUTO checks the starting data for components with such a phase gain and, if present, it will automatically adjust the computations accordingly. The model equations, a system of two coupled pendula, (Doedel, Aronson & Othmer (1991)), are given by

$$\begin{aligned} \phi_1'' + \epsilon\phi_1' + \sin\phi_1 &= I + \gamma(\phi_2 - \phi_1), \\ \phi_2'' + \epsilon\phi_2' + \sin\phi_2 &= I + \gamma(\phi_1 - \phi_2), \end{aligned} \tag{10.11}$$

or, in equivalent first order form,

$$\begin{aligned} \phi_1' &= \psi_1, \\ \phi_2' &= \psi_2, \\ \psi_1' &= -\epsilon\psi_1 - \sin\phi_1 + I + \gamma(\phi_2 - \phi_1), \\ \psi_2' &= -\epsilon\psi_2 - \sin\phi_2 + I + \gamma(\phi_1 - \phi_2). \end{aligned} \tag{10.12}$$

Throughout $\gamma = 0.175$. Initially, $\epsilon = 0.1$ and $I = 0.4$.

Numerical data representing one complete rotation are contained in the file `pen.dat`. Each row in `pen.dat` contains five real numbers, namely, the time variable t, ϕ_1, ϕ_2, ψ_1 and ψ_2. The correponding parameter values are defined in the user-supplied subroutine `STPNT`.

Actually, in this example, a scaled time variable t is given in `pen.dat`. For this reason the period (`PAR(11)`) is also set in `STPNT`. Normally AUTO would automatically set the period according to the data in `pen.dat`.

The AUTO-command *@fc pen* converts the data in `pen.dat` to a labeled AUTO solution (with label 1) in a new file `q.dat`. The mesh will be suitably adapted to the solution, using the number of mesh intervals `NTST` and the number of collocation point per mesh interval `NCOL` specified in the constants-file `r.pen`. (Note that the file `q.dat` should be used for restart only. Do not append new output-files to `q.dat`, as the command *@fc pen* only creates `q.dat`, with no corresponding `p.dat`.)

The first run, with I as free problem parameter, starts from the converted solution with label 1 in `pen.dat`. A period-doubling bifurcation is located, and the period-doubled branch is computed in the second run. Two branch points are located, and the bifurcating branches are traced out in the third and fourth run, respectively. The fifth run generates starting data for the subsequent computation of a locus of period-doubling bifurcations. The actual computation is done in the sixth run, with ϵ and I as free problem parameters.

COMMAND	ACTION
mkdir pen	create an empty work directory
cd pen	change directory
@dm pen	copy the demo files to the work directory
cp r.pen.1 r.pen	get the first constants-file
@fc pen	convert `pen.dat` to AUTO format in `q.dat`
@r pen dat	locate a period doubling bifurcation; restart from `q.dat`
@sv pen	save output-files as `p.pen, q.pen, d.pen`
cp r.pen.2 r.pen	constants changed : `IPS, NTST, ISW, NMX`
@r pen	a branch of period-doubled (and out-of-phase) rotations
@ap pen	append output-files tp `p.pen, q.pen, d.pen`
cp r.pen.3 r.pen	constants changed : `IRS, ISP`
@r pen	a secondary bifurcating branch (without bifurcation detection)
@ap pen	append output-files to `p.pen, q.pen, d.pen`
cp r.pen.4 r.pen	constants changed : `IRS`
@r pen	another secondary bifurcating branch (without bifurcation detection)
@ap pen	append output-files to `p.pen, q.pen, d.pen`
cp r.pen.5 r.pen	constants changed : `IRS, ICP, ICP, ISW, NMX`
@r pen	generate starting data for period doubling continuation
@sv t	save output-files as `p.t, q.t, d.t`
cp r.pen.6 r.pen	constants changed : `IRS`
@r pen t	compute a locus of period doubling bifurcations; restart from `q.t`
@sv pd	save output-files as `p.pd, q.pd, d.pd`
@pn pen	run an animation program to view the solutions in `q.pen`
	(on SGI machines only; see also the file `auto/97/pendula/README`).

Table 10.11: Commands for running demo `pen`.

10.11 chu : A Non-Smooth System (Chua's Circuit).

Chua's circuit is one of the simplest electronic devices to exhibit complex behavior. For related calculations see Khibnik, Roose & Chua (1993). The equations modeling the circuit are

$$\begin{aligned} u_1' &= \alpha[\ u_2 - h(u_1)\ ]\ , \\ u_2' &= u_1 - u_2 + u_3\ , \\ u_3' &= -\beta\ u_2\ , \end{aligned} \tag{10.13}$$

where

$$h(x) = a_1 x + \frac{1}{2}\ (a_0 - a_1)\ \{|\ x+1\ | - |\ x-1\ |\}\ ,$$

and where we take $\beta = 14.3$, $a_0 = -1/7$, $a_1 = 2/7$.

Note that $h(x)$ is not a smooth function, and hence the solution to the equations may have non-smooth derivatives. However, for the orthogonal collocation method to attain its optimal accuracy, it is necessary that the solution be sufficiently smooth. Moreover, the adaptive mesh selection strategy will fail if the solution or one of its lower order derivatives has discontinuities. For these reasons we use the smooth approximation

$$|\ x\ | \approx \frac{2x}{\pi}\ \arctan(Kx),$$

which get better as K increases. In the numerical calculations below we use $K = 10$. The free parameter is α.

COMMAND	ACTION
mkdir chu	create an empty work directory
cd chu	change directory
@dm chu	copy the demo files to the work directory
cp r.chu.1 r.chu	get the first constants-file
@r chu	1st run; stationary solutions
@sv chu	save output-files as `p.chu, q.chu, d.chu`
cp r.chu.2 r.chu	constants changed : `IPS, IRS, ICP, ICP`
@r chu	2nd run; periodic solutions, with detection of period-doubling
@ap chu	append the output-files to `p.chu, q.chu, d.chu`

Table 10.12: Commands for running demo `chu`.

10.12 phs : Effect of the Phase Condition.

This demo illustrates the effect of the phase condition on the computation of periodic solutions. We consider the differential equation

$$\begin{aligned} u_1' &= \lambda u_1 - u_2, \\ u_2' &= u_1(1-u_1). \end{aligned} \tag{10.14}$$

This equation has a Hopf bifurcation from the trivial solution at $\lambda = 0$. The bifurcating branch of periodic solutions is vertical and along it the period increases monotonically. The branch terminates in a homoclinic orbit containing the saddle point $(u_1, u_2) = (1, 0)$. Graphical inspection of the computed periodic orbits, for example u_1 versus the scaled time variable t, shows how the phase condition has the effect of keeping the "peak" in the solution in the same location.

COMMAND	ACTION
mkdir phs	create an empty work directory
cd phs	change directory
@dm phs	copy the demo files to the work directory
cp r.phs.1 r.phs	get the first constants-file
@r phs	detect Hopf bifurcation
@sv phs	save output-files as `p.phs, q.phs, d.phs`
cp r.phs.2 r.phs	constants changed : `IRS, IPS, NPR`
@r phs	compute periodic solutions
@ap phs	append output-files to `p.phs, q.phs, d.phs`

Table 10.13: Commands for running demo `phs`.

10.13 ivp : Time Integration with Euler's Method.

This demo uses Euler's method to locate a stationary solution of the following predator-prey system with harvesting :

$$\begin{aligned} u_1' &= p_2 u_1(1-u_1) - u_1 u_2 - p_1(1-e^{-p_3 u_1}), \\ u_2' &= -u_2 + p_4 u_1 u_2, \end{aligned} \tag{10.15}$$

where all problem parameters have a fixed value. The equations are the same as those in demo `pp2`. The continuation parameter is the independent time variable, namely `PAR(14)`.

Note that Euler time integration is only first order accurate, so that the time step must be sufficiently small to ensure correct results. Indeed, this option has been added only as a convenience, and should generally be used only to locate stationary states. Note that the AUTO-constants `DS`, `DSMIN`, and `DSMAX` control the step size in the space consisting of time, here `PAR(14)`, and the state vector, here (u_1, u_2).

COMMAND	ACTION
mkdir ivp	create an empty work directory
cd ivp	change directory
@dm ivp	copy the demo files to the work directory
cp r.ivp.1 r.ivp	get the constants-file
@r ivp	time integration
@sv ivp	save output-files as `p.ivp, q.ivp, d.ivp`

Table 10.14: Commands for running demo `ivp`.

Chapter 11

AUTO **Demos : BVP.**

11.1 exp : Bratu's Equation.

This demo illustrates the computation of a solution branch to the boundary value problem

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{u_1}, \end{aligned} \tag{11.1}$$

with boundary conditions $u_1(0) = 0, \quad u_1(1) = 0$. This equation is also considered in Doedel, Keller & Kernévez (1991*a*).

COMMAND	ACTION
mkdir exp	create an empty work directory
cd exp	change directory
@dm exp	copy the demo files to the work directory
cp r.exp.1 r.exp	get the first constants-file
@r exp	1st run; compute solution branch containing fold
@sv exp	save output-files as `p.exp, q.exp, d.exp`
cp r.exp.2 r.exp	constants changed : `IRS, NTST, A1, DSMAX`
@r exp	2nd run; restart at a labeled solution, using increased accuracy
@ap exp	append output-files to `p.exp, q.exp, d.exp`

Table 11.1: Commands for running demo `exp`.

11.2 int : Boundary and Integral Constraints.

This demo illustrates the computation of a solution branch to the equation

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{u_1}, \end{aligned} \tag{11.2}$$

with a non-separated boundary condition and an integral constraint:

$$u_1(0) - u_1(1) - p_2 = 0, \qquad \int_0^1 u(t)dt - p_3 = 0.$$

The solution branch contains a fold, which, in the second run, is continued in two equation parameters.

COMMAND	ACTION
mkdir int	create an empty work directory
cd int	change directory
@dm int	copy the demo files to the work directory
cp r.int.1 r.int	get the first constants-file
@r int	1st run; detection of a fold
@sv int	save output-files as `p.int, q.int, d.int`
cp r.int.2 r.int	constants changed : `IRS, ISW`
@r int	2nd run; generate starting data for a curve of folds
@sv t	save the output-files as `p.t, q.t, d.t`
cp r.int.3 r.int	constants changed : `IRS`
@r int t	2nd run; compute a curve of folds; restart from `q.t`
@sv lp	save the output-files as `p.lp, q.lp, d.lp`

Table 11.2: Commands for running demo `int`.

11.3 bvp : A Nonlinear ODE Eigenvalue Problem.

This demo illustrates the location of eigenvalues of a nonlinear ODE boundary value problem as bifurcations from the trivial solution branch. The branch of solutions that bifurcates at the first eigenvalue is computed in both directions. The equations are

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -(p_1\pi)^2 u_1 + u_1^2, \end{aligned} \tag{11.3}$$

with boundary conditions $u_1(0) = 0, \quad u_1(1) = 0$.

COMMAND	ACTION
mkdir bvp	create an empty work directory
cd bvp	change directory
@dm bvp	copy the demo files to the work directory
cp r.bvp.1 r.bvp	get the first constants-file
@r bvp	compute the trivial solution branch and locate eigenvalues
@sv bvp	save output-files as `p.bvp, q.bvp, d.bvp`
cp r.bvp.2 r.bvp	constants changed : `IRS, ISW, NPR, DSMAX`
@r bvp	compute the first bifurcating branch
@ap bvp	append output-files to `p.bvp, q.bvp, d.bvp`
cp r.bvp.3 r.bvp	constants changed : `DS`
@r bvp	compute the first bifurcating branch in opposite direction
@ap bvp	append output-files to `p.bvp, q.bvp, d.bvp`

Table 11.3: Commands for running demo `bvp`.

11.4 lin : A Linear ODE Eigenvalue Problem.

This demo illustrates the location of eigenvalues of a linear ODE boundary value problem as bifurcations from the trivial solution branch. By means of branch switching an eigenfunction is computed, as is illustrated for the first eigenvalue. This eigenvalue is then continued in two parameters by fixing the L_2-norm of the first solution component. The eigenvalue problem is given by the equations

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= (p_1 \pi)^2 u_1, \end{aligned} \tag{11.4}$$

with boundary conditions $u_1(0) - p_2 = 0$ and $u_1(1) = 0$. We add the integral constraint $\int_0^1 u_1(t)^2 dt - p_3 = 0$. Then p_3 is simply the L_2-norm of the first solution component. In the first two runs p_2 is fixed, while p_1 and p_3 are free. In the third run p_3 is fixed, while p_1 and p_2 are free.

COMMAND	ACTION
mkdir lin	create an empty work directory
cd lin	change directory
@dm lin	copy the demo files to the work directory
cp r.lin.1 r.lin	get the first constants-file
@r lin	1st run; compute the trivial solution branch and locate eigenvalues
@sv lin	save output-files as `p.lin, q.lin, d.lin`
cp r.lin.2 r.lin	constants changed : `IRS, ISW, DSMAX`
@r lin	2nd run; compute a few steps along the bifurcating branch
@ap lin	append output-files to `p.lin, q.lin, d.lin`
cp r.lin.3 r.lin	constants changed : `IRS, ISW, ICP(2)`
@r lin	3rd run; compute a two-parameter curve of eigenvalues
@sv 2p	save the output-files as `p.2p, q.2p, d.2p`

Table 11.4: Commands for running demo `lin`.

11.5 non : A Non-Autonomous BVP.

This demo illustrates the continuation of solutions to the non-autonomous boundary value problem

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{x^3 u_1}, \end{aligned} \tag{11.5}$$

with boundary conditions $u_1(0) = 0, \quad u_1(1) = 0$. Here x is the independent variable. This system is first converted to the following equivalent autonomous system :

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{u_3^3 u_1}, \\ u_3' &= 1, \end{aligned} \tag{11.6}$$

with boundary conditions $u_1(0) = 0, \quad u_1(1) = 0, \quad u_3(0) = 0$. (For a periodically forced system see demo `frc`).

COMMAND	ACTION
mkdir non	create an empty work directory
cd non	change directory
@dm non	copy the demo files to the work directory
cp r.non.1 r.non	get the constants-file
@r non	compute the solution branch
@sv non	save output-files as `p.non, q.non, d.non`

Table 11.5: Commands for running demo `non`.

11.6 kar : The Von Karman Swirling Flows.

The steady axi-symmetric flow of a viscous incompressible fluid above an infinite rotating disk is modeled by the following ODE boundary value problem (Equation (11) in Lentini & Keller (1980) :

$$\begin{aligned} u_1' &= Tu_2, \\ u_2' &= Tu_3, \\ u_3' &= T[-2\gamma u_4 + u_2^2 - 2u_1u_3 - u_4^2], \\ u_4' &= Tu_5, \\ u_5' &= T[2\gamma u_2 + 2u_2u_4 - 2u_1u_5], \end{aligned} \tag{11.7}$$

with left boundary conditions

$$u_1(0) = 0, \qquad u_2(0) = 0, \qquad u_4(0) = 1 - \gamma,$$

and (asymptotic) right boundary conditions

$$\begin{aligned} &[f_\infty + a(f_\infty,\gamma)]\ u_2(1) + u_3(1) - \gamma\ \tfrac{u_4(1)}{a(f_\infty,\gamma)} = 0, \\ &a(f_\infty,\gamma)\ \tfrac{b^2(f_\infty,\gamma)}{\gamma}\ u_2(1) + [f_\infty + a(f_\infty,\gamma)]\ u_4(1) + u_5(1) = 0, \\ &u_1(1) = f_\infty, \end{aligned} \tag{11.8}$$

where

$$\begin{aligned} a(f_\infty,\gamma) &= \tfrac{1}{\sqrt{2}}[(f_\infty^4 + 4\gamma^2)^{1/2} + f_\infty^2]^{1/2}, \\ b(f_\infty,\gamma) &= \tfrac{1}{\sqrt{2}}[(f_\infty^4 + 4\gamma^2)^{1/2} - f_\infty^2]^{1/2}. \end{aligned} \tag{11.9}$$

Note that there are five differential equations and six boundary conditions. Correspondingly, there are two free parameters in the computation of a solution branch, namely γ and f_∞. The "period" T is fixed; $T = 500$. The starting solution is $u_i = 0$, $i = 1, \cdots, 5$, at $\gamma = 1$, $f_\infty = 0$.

COMMAND	ACTION
mkdir kar	create an empty work directory
cd kar	change directory
@dm kar	copy the demo files to the work directory
cp r.kar.1 r.kar	get the constants-file
@r kar	computation of the solution branch
@sv kar	save output-files as `p.kar, q.kar, d.kar`

Table 11.6: Commands for running demo `kar`.

11.7 spb : A Singularly-Perturbed BVP.

This demo illustrates the use of continuation to compute solutions to the singularly perturbed boundary value problem

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= \frac{\lambda}{\epsilon}(u_1 u_2 (u_1^2 - 1) + u_1), \end{aligned} \tag{11.10}$$

with boundary conditions $u_1(0) = 3/2$, $u_1(1) = \gamma$. The parameter λ has been introduced into the equations in order to allow a homotopy from a simple equation with known exact solution to the actual equation. This is done in the first run. In the second run ϵ is decreased by continuation. In the third run ϵ is fixed at $\epsilon = .001$ and the solution is continued in γ. This run takes more than 1500 continuation steps. For a detailed analysis of the solution behavior see Lorenz (1982).

COMMAND	ACTION
mkdir spb	create an empty work directory
cd spb	change directory
@dm spb	copy the demo files to the work directory
cp r.spb.1 r.spb	get the first constants-file
@r spb	1st run; homotopy from $\lambda = 0$ to $\lambda = 1$
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.spb.2 r.spb	constants changed : `IRS, ICP(1), NTST, DS`
@r spb 1	2nd run; let ϵ tend to zero; restart from `q.1`
@sv 2	save the output-files as `p.2, q.2, d.2`
cp r.spb.3 r.spb	constants changed : `IRS, ICP(1), RL0, ITNW, EPSL, EPSU, NUZR`
@r spb 2	3rd run; continuation in γ; $\epsilon = 0.001$; restart from `q.2`
@sv 3	save the output-files as `p.3, q.3, d.3`

Table 11.7: Commands for running demo `spb`.

11.8 ezp : Complex Bifurcation in a BVP.

This demo illustrates the computation of a solution branch to the the complex boundary value problem

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{u_1}, \end{aligned} \tag{11.11}$$

with boundary conditions $u_1(0) = 0,\ u_1(1) = 0$. Here u_1 and u_2 are allowed to be complex, while the parameter p_1 can only take real values. In the real case, this is Bratu's equation, whose solution branch contains a fold; see the demo `exp`. It is known (Henderson & Keller (1990)) that a simple quadratic fold gives rise to a pitch fork bifurcation in the complex equation. This bifurcation is located in the first computation below. In the second and third run, both legs of the bifurcating solution branch are computed. On it, both solution components u_1 and u_2 have nontrivial imaginary part.

COMMAND	ACTION
mkdir ezp	create an empty work directory
cd ezp	change directory
@dm ezp	copy the demo files to the work directory
cp r.ezp.1 r.ezp	get the first constants-file
@r ezp	1st run; compute solution branch containing fold
@sv ezp	save output-files as `p.ezp, q.ezp, d.ezp`
cp r.ezp.2 r.ezp	constants changed : `IRS, ISW`
@r ezp	2nd run; compute bifurcating complex solution branch
@ap ezp	append output-files to `p.ezp, q.ezp, d.ezp`
cp r.ezp.3 r.ezp	constant changed : `DS`
@r ezp	3rd run; compute 2nd leg of bifurcating branch
@ap ezp	append output-files to `p.ezp, q.ezp, d.ezp`

Table 11.8: Commands for running demo `ezp`.

Chapter 12

AUTO **Demos : Parabolic PDEs.**

12.1 pd1 : Stationary States (1D Problem).

This demo uses Euler's method to locate a stationary solution of a nonlinear parabolic PDE, followed by continuation of this stationary state in a free problem parameter. The equation is

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + p_1 \, u \, (1 - u),$$

on the space interval $[0, L]$, where $L =$ `PAR(11)` $= 10$ is fixed throughout, as is the diffusion constant $D =$ `PAR(15)` $= 0.1$. The boundary conditions are $u(0) = u(L) = 0$ for all time.

In the first run the continuation parameter is the independent time variable, namely `PAR(14)`, while $p_1 = 1$ is fixed. The AUTO-constants `DS`, `DSMIN`, and `DSMAX` then control the step size in space-time, here consisting of `PAR(14)` and $u(x)$. Initial data are $u(x) = \sin(\pi x/L)$ at time zero. Note that in the subroutine `STPNT` the initial data must be scaled to the unit interval, and that the scaled derivative must also be provided; see the equations-file `pv1.f`. In the second run the continuation parameter is p_1.

Euler time integration is only first order accurate, so that the time step must be sufficiently small to ensure correct results. Indeed, this option has been added only as a convenience, and should generally be used only to locate stationary states.

COMMAND	ACTION
mkdir pd1	create an empty work directory
cd pd1	change directory
@dm pd1	copy the demo files to the work directory
cp r.pd1.1 r.pd1	get the first constants-file
@r pd1	time integration towards stationary state
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.pd1.2 r.pd1	constants changed : `IPS, IRS, ICP`, etc.
@r pd1 1	continuation of stationary states; read restart data from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`

Table 12.1: Commands for running demo `pd1`.

12.2 pd2 : Stationary States (2D Problem).

This demo uses Euler's method to locate a stationary solution of a nonlinear parabolic PDE, followed by continuation of this stationary state in a free problem parameter. The equations are

$$\begin{aligned} \partial u_1/\partial t &= D_1\ \partial^2 u_1/\partial x^2\ +\ p_1\ u\ (1-u)\ -\ u_1 u_2, \\ \partial u_2/\partial t &= D_2\ \partial^2 u_2/\partial x^2\ -\ u_2\ +\ u_1 u_2, \end{aligned} \tag{12.1}$$

on the space interval $[0, L]$, where $L =$ `PAR(11)` $= 1$ is fixed throughout, as are the diffusion constants $D_1 =$ `PAR(15)` $= 1$ and $D_2 =$ `PAR(16)` $= 1$. The boundary conditions are $u_1(0) = u_1(L) = 0$ and $u_2(0) = u_2(L) = 1$, for all time.

In the first run the continuation parameter is the independent time variable, namely `PAR(14)`, while $p_1 = 12$ is fixed. The AUTO-constants `DS`, `DSMIN`, and `DSMAX` then control the step size in space-time, here consisting of `PAR(14)` and $(u_1(x), u_2(x))$. Initial data at time zero are $u_1(x) = \sin(\pi x/L)$ and $u_2(x) = 1$. Note that in the subroutine `STPNT` the initial data must be scaled to the unit interval, and that the scaled derivatives must also be provided; see the equations-file `pv2.f`. In the second run the continuation parameter is p_1. A branch point is located during this run.

Euler time integration is only first order accurate, so that the time step must be sufficiently small to ensure correct results. Indeed, this option has been added only as a convenience, and should generally be used only to locate stationary states.

COMMAND	ACTION
mkdir pd2	create an empty work directory
cd pd2	change directory
@dm pd2	copy the demo files to the work directory
cp r.pd2.1 r.pd2	get the first constants-file
@r pd2	time integration towards stationary state
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.pd2.2 r.pd2	constants changed : `IPS, IRS, ICP`, etc.
@r pd2 1	continuation of stationary states; read restart data from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`

Table 12.2: Commands for running demo `pd2`.

12.3 wav : Periodic Waves.

This demo illustrates the computation of various periodic wave solutions to a system of coupled parabolic partial differential equations on the spatial interval $[0,1]$. The equations, that model an enzyme catalyzed reaction (Doedel & Kernévez (1986*b*)) are :

$$\begin{aligned} \partial u_1/\partial t &= \partial^2 u_1/\partial x^2 - p_1[p_4 R(u_1,u_2) - (p_2 - u_1)], \\ \partial u_2/\partial t &= \beta\partial^2 u_2/\partial x^2 - p_1[p_4 R(u_1,u_2) - p_7(p_3 - u_2)]. \end{aligned} \tag{12.2}$$

All equation parameters, except p_3, are fixed throughout.

COMMAND	ACTION
mkdir wav	create an empty work directory
cd wav	change directory
@dm wav	copy the demo files to the work directory
cp r.wav.1 r.wav	get the first constants-file
@r wav	1st run; stationary solutions of the system without diffusion
@sv ode	save output-files as `p.ode, q.ode, d.ode`
cp r.wav.2 r.wav	constants changed : `IPS`
@r wav	2nd run; detect bifurcations to wave train solutions
@sv wav	save output-files as `p.wav, q.wav, d.wav`
cp r.wav.3 r.wav	constants changed : `IRS, IPS, NUZR, ILP`
@r wav	3rd run; wave train solutions of fixed wave speed
@ap wav	append output-files to `p.wav, q.wav, d.wav`
cp r.wav.4 r.wav	constants changed : `IRS, IPS, NMX, ICP, NUZR`
@r wav	4th run; wave train solutions of fixed wave length
@sv rng	save output-files as `p.rng, q.rng, d.rng`
cp r.wav.5 r.wav	constants changed : `IPS, NMX, NPR, ICP`
@r wav	5th run; time evolution computation
@sv tim	save output-files as `p.tim, q.tim, d.tim`

Table 12.3: Commands for running demo `wav`.

12.4 brc : Chebyshev Collocation in Space.

This demo illustrates the computation of stationary solutions and periodic solutions to systems of parabolic PDEs in one space variable, using Chebyshev collocation in space. More precisely, the approximate solution is assumed of the form $u(x,t) = \sum_{k=0}^{n+1} u_k(t)\ell_k(x)$. Here $u_k(t)$ corresponds to $u(x_k,t)$ at the Chebyshev points $\{x_k\}_{k=1}^{n}$ with respect to the interval $[0,1]$. The polynomials $\{\ell_k(x)\}_{k=0}^{n+1}$ are the Lagrange interpolating coefficients with respect to points $\{x_k\}_{k=0}^{n+1}$, where $x_0 = 0$ and $x_{n+1} = 1$. The number of Chebyshev points in $[0,1]$, as well as the number of equations in the PDE system, can be set by the user in the file `brc.inc`.

As an illustrative application we consider the Brusselator (Holodniok, Knedlik & Kubíček (1987))

$$\begin{aligned} u_t &= D_x/L^2 u_{xx} + u^2 v - (B+1)u + A, \\ v_t &= D_y/L^2 v_{xx} - u^2 v + Bu, \end{aligned} \tag{12.3}$$

with boundary conditions $u(0,t) = u(1,t) = A$ and $v(0,t) = v(1,t) = B/A$.

Note that, given the non-adaptive spatial discretization, the computational procedure here is not appropriate for PDEs with solutions that rapidly vary in space, and care must be taken to recognize spurious solutions and bifurcations.

COMMAND	ACTION
mkdir brc	create an empty work directory
cd brc	change directory
@dm brc	copy the demo files to the work directory
cp r.brc.1 r.brc	get the first constants-file
@r brc	compute the stationary solution branch with Hopf bifurcations
@sv brc	save output-files as `p.brc, q.brc, d.brc`
cp r.brc.2 r.brc	constants changed : `IRS, IPS`
@r brc	compute a branch of periodic solutions from the first Hopf point
@ap brc	append the output-files to `p.brc, q.brc, d.brc`
cp r.brc.3 r.brc	constants changed : `IRS, ISW`
@r brc	compute a solution branch from a secondary periodic bifurcation
@ap brc	append the output-files to `p.brc, q.brc, d.brc`

Table 12.4: Commands for running demo `brc`.

12.5 brf : Finite Differences in Space.

This demo illustrates the computation of stationary solutions and periodic solutions to systems of parabolic PDEs in one space variable. A fourth order accurate finite difference approximation is used to approximate the second order space derivatives. This reduces the PDE to an autonomous ODE of fixed dimension which AUTO is capable of treating. The spatial mesh is uniform; the number of mesh intervals, as well as the number of equations in the PDE system, can be set by the user in the file `brf.inc`.

As an illustrative application we consider the Brusselator (Holodniok, Knedlik & Kubíček (1987))

$$\begin{aligned} u_t &= D_x/L^2 u_{xx} + u^2 v - (B+1)u + A, \\ v_t &= D_y/L^2 v_{xx} - u^2 v + Bu, \end{aligned} \tag{12.4}$$

with boundary conditions $u(0,t) = u(1,t) = A$ and $v(0,t) = v(1,t) = B/A$.

Note that, given the non-adaptive spatial discretization, the computational procedure here is not appropriate for PDEs with solutions that rapidly vary in space, and care must be taken to recognize spurious solutions and bifurcations.

COMMAND	ACTION
mkdir brf	create an empty work directory
cd brf	change directory
@dm brf	copy the demo files to the work directory
cp r.brf.1 r.brf	get the first constants-file
@r brf	compute the stationary solution branch with Hopf bifurcations
@sv brf	save output-files as `p.brf, q.brf, d.brf`
cp r.brf.2 r.brf	constants changed : `IRS, IPS`
@r brf	compute a branch of periodic solutions from the first Hopf point
@ap brf	append the output-files to `p.brf, q.brf, d.brf`
cp r.brf.3 r.brf	constants changed : `IRS, ISW`
@r brf	compute a solution branch from a secondary periodic bifurcation
@ap brf	append the output-files to `p.brf, q.brf, d.brf`

Table 12.5: Commands for running demo `brf`.

12.6 bru : Euler Time Integration (the Brusselator).

This demo illustrates the use of Euler's method for time integration of a nonlinear parabolic PDE. The example is the Brusselator (Holodniok, Knedlik & Kubíček (1987)), given by

$$\begin{aligned} u_t &= D_x/L^2 u_{xx} + u^2 v - (B+1)u + A, \\ v_t &= D_y/L^2 v_{xx} - u^2 v + Bu, \end{aligned} \tag{12.5}$$

with boundary conditions $u(0,t) = u(1,t) = A$ and $v(0,t) = v(1,t) = B/A$. All parameters are given fixed values for which a stable periodic solution is known to exist.

The continuation parameter is the independent time variable, namely `PAR(14)`. The AUTO-constants `DS`, `DSMIN`, and `DSMAX` then control the step size in space-time, here consisting of `PAR(14)` and $(u(x), v(x))$. Initial data at time zero are $u(x) = A - 0.5\sin(\pi x)$ and $v(x) = B/A + 0.7\sin(\pi x)$. Note that in the subroutine `STPNT` the space derivatives of u and v must also be provided; see the equations-file `bru.f`.

Euler time integration is only first order accurate, so that the time step must be sufficiently small to ensure correct results. This option has been added only as a convenience, and should generally be used only to locate stationary states. Indeed, in the case of the asymptotic periodic state of this demo, the number of required steps is very large and use of a better time integrator is advisable.

COMMAND	ACTION
mkdir bru	create an empty work directory
cd bru	change directory
@dm bru	copy the demo files to the work directory
cp r.bru.1 r.bru	get the constants-file
@r bru	time integration
@sv bru	save output-files as `p.bru, q.bru, d.bru`

Table 12.6: Commands for running demo `bru`.

Chapter 13

AUTO **Demos : Optimization.**

13.1 opt : A Model Algebraic Optimization Problem.

This demo illustrates the method of successive continuation for constrained optimization problems by applying it to the following simple problem : Find the maximum sum of coordinates on the unit sphere in R^5. Coordinate 1 is treated as the state variable. Coordinates 2-5 are treated as control parameters. For details on the successive continuation procedure see Doedel, Keller & Kernévez (1991*a*), Doedel, Keller & Kernévez (1991*b*).

COMMAND	ACTION
mkdir opt	create an empty work directory
cd opt	change directory
@dm opt	copy the demo files to the work directory
cp r.opt.1 r.opt	get the first constants-file
@r opt	one free equation parameter
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.opt.2 r.opt	constants changed : `IRS`
@r opt 1	two free equation parameters; read restart data from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`
cp r.opt.3 r.opt	constants changed : `IRS`
@r opt 2	three free equation parameters; read restart data from `q.2`
@sv 3	save output-files as `p.3, q.3, d.3`
cp r.opt.4 r.opt	constants changed : `IRS`
@r opt 3	four free equation parameters; read restart data from `q.3`
@sv 4	save output-files as `p.4, q.4, d.4`

Table 13.1: Commands for running demo `opt`.

13.2 ops : Optimization of Periodic Solutions.

This demo illustrates the method of successive continuation for the optimization of periodic solutions. For a detailed description of the basic method see Doedel, Keller & Kernévez (1991*b*). The illustrative system of autonomous ODEs, taken from Rodríguez-Luis (1991), is

$$\begin{aligned} x'(t) &= [-\lambda_4(x^3/3 - x) + (z - x)/\lambda_2 - y]/\lambda_1, \\ y'(t) &= x - \lambda_3, \\ z'(t) &= -(z - x)/\lambda_2, \end{aligned} \tag{13.1}$$

with objective functional

$$\omega = \int_0^1 g(x, y, z; \lambda_1, \lambda_2, \lambda_3, \lambda_4)\ dt,$$

where $g(x, y, z; \lambda_1, \lambda_2, \lambda_3, \lambda_4) \equiv \lambda_3$. Thus, in this application, a one-parameter extremum of g corresponds to a fold with respect to the problem parameter λ_3, and multi-parameter extrema correspond to generalized folds. Note that, in general, the objective functional is an integral along the periodic orbit, so that a variety of optimization problems can be addressed.

For the case of periodic solutions, the extended optimality system can be generated automatically, i.e., one need only define the vector field and the objective functional, as in done in the file `ops.f`. For reference purpose it is convenient here to write down the full extended system in its general form :

$$\begin{aligned}
& u'(t) = Tf(u(t), \lambda), \qquad T \in \mathrm{R}\ \text{(period)},\ u(\cdot), f(\cdot,\cdot) \in \mathrm{R}^n,\ \lambda \in \mathrm{R}^{n_\lambda}, \\
& w'(t) = -Tf_u(u(t), \lambda)^* w(t) + \kappa u_0'(t) + \gamma g_u(u(t), \lambda)^*, \qquad w(\cdot) \in \mathrm{R}^n,\ \kappa, \gamma \in \mathrm{R}, \\
& u(1) - u(0) = 0, \qquad w(1) - w(0) = 0, \\
& \textstyle\int_0^1 u(t)^* u_0'(t)\ dt = 0, \\
& \textstyle\int_0^1 \omega - g(u(t), \lambda)\ dt = 0, \\
& \textstyle\int_0^1 w(t)^* w(t) + \kappa^2 + \gamma^2 - \alpha\ dt = 0, \qquad \alpha \in \mathrm{R}, \\
& \textstyle\int_0^1 f(u(t), \lambda)^* w(t) - \gamma g_T(u(t), \lambda) - \tau_0\ dt = 0, \qquad \tau_0 \in \mathrm{R}, \\
& \textstyle\int_0^1 Tf_{\lambda_i}(u(t), \lambda)^* w(t) - \gamma g_{\lambda_i}(u(t), \lambda) - \tau_i\ dt = 0, \qquad \tau_i \in \mathrm{R}, \quad i = 1, \cdots, n_\lambda.
\end{aligned} \tag{13.2}$$

Above u_0 is a reference solution, namely, the previous solution along a solution branch.

In the computations below, the two preliminary runs, with IPS=1 and IPS=2, respectively, locate periodic solutions. The subsequent runs are with IPS=15 and hence use the automatically generated extended system.

- *Run 1.* Locate a Hopf bifurcation. The free system parameter is λ_3.
- *Run 2.* Compute a branch of periodic solutions from the Hopf bifurcation.
- *Run 3.* This run retraces part of the periodic solution branch, using the full optimality system, but with all adjoint variables, $w(\cdot), \kappa, \gamma$, and hence α, equal to zero. The optimality parameters τ_0 and τ_3 are zero throughout. An extremum of the objective functional with respect to λ_3 is located. Such a point corresponds to a branch point of the extended system. Given the choice of objective functional in this demo, this extremum is also a fold with respect to λ_3.
- *Run 4.* Branch switching at the above-found branch point yields nonzero values of the adjoint variables. Any point on the bifurcating branch away from the branch point can serve as starting solution for the next run. In fact, the branch-switching can be viewed as generating a nonzero eigenvector in an eigenvalue-eigenvector relation. Apart from the adjoint variables, all other variables remain unchanged along the bifurcating branch.
- *Run 5.* The above-found starting solution is continued in two system parameters, here λ_3 and λ_2; i.e., a two-parameter branch of extrema with respect to λ_3 is computed. Along this branch the value of the optimality parameter τ_2 is monitored, i.e., the value of the functional that vanishes at an extremum with respect to the system parameter λ_2. Such a zero of τ_2 is, in fact, located, and hence an extremum of the objective functional with respect to both λ_2 and λ_3 has been found. Note that, in general, τ_i is the value of the functional that vanishes at an extremum with respect to the system parameter λ_i.
- *Run 6.* In the final run, the above-found two-parameter extremum is continued in three system parameters, here λ_1, λ_2, and λ_3, toward $\lambda_1 = 0$. Again, given the particular choice of objective functional, this final continuation has an alternate significance here : it also represents a three-parameter branch of transcritical secondary periodic bifurcations points.

Although not illustrated here, one can restart an ordinary continuation of periodic solutions, using IPS=2 or IPS=3, from a labeled solution point on a branch computed with IPS=15.

The free scalar variables specified in the AUTO constants-files for Run 3 and Run 4 are shown in Table 13.2.

Index	3	11	12	22	-22	-23	-31
Variable	λ_3	T	α	τ_2	$[\lambda_2]$	$[\lambda_3]$	$[T]$

Table 13.2: *Runs 3 and 4* (files `r.ops.3` and `r.ops.4`).

The parameter α, which is the norm of the adjoint variables, becomes nonzero after branch switching in Run 4. The negative indices (-22, -23, and -31) set the active optimality functionals, namely for λ_2, λ_3, and T, respectively, with corresponding variables τ_2, τ_3, and τ_0, respectively. These should be set in the first run with `IPS`=15 and remain unchanged in all subsequent runs.

Index	3	2	11	22	-22	-23	-31
Variable	λ_3	λ_2	T	τ_2	$[\lambda_2]$	$[\lambda_3]$	$[T]$

Table 13.3: *Run 5* (file `r.ops.5`).

In Run 5 the parameter α, which has been replaced by λ_2, remains fixed and nonzero. The variable τ_2 monitors the value of the optimality functional associated with λ_2. The zero of τ_2 located in this run signals an extremum with respect to λ_2.

Index	3	2	1	11	-22	-23	-31
Variable	λ_3	λ_2	λ_1	T	$[\lambda_2]$	$[\lambda_3]$	$[T]$

Table 13.4: *Run 6* (file `r.ops.6`).

In Run 6 τ_2, which has been replaced by λ_1, remains zero.

Note that τ_0 and τ_3 are not used as variables in any of the runs; in fact, their values remain zero throughout. Also note that the optimality functionals corresponding to τ_0 and τ_3 (or, equivalently, to T and λ_3) *are* active in all runs. This set-up allows the detection of the extremum of the objective functional, with T and λ_3 as scalar equation parameters, as a bifurcation in the third run.

The parameter λ_4, and its corresponding optimality variable τ_4, are not used in this demo. Also, λ_1 is used in the last run only, and its corresponding optimality variable τ_1 is never used.

COMMAND	ACTION
mkdir ops	create an empty work directory
cd ops	change directory
@dm ops	copy the demo files to the work directory
cp r.ops.1 r.ops	get the first constants-file
@r ops	locate a Hopf bifurcation
@sv 0	save output-files as `p.0, q.0, d.0`
cp r.ops.2 r.ops	constants changed : `IPS, IRS, NMX, NUZR`
@r ops 0	compute a branch of periodic solutions; restart from `q.0`
@ap 0	append the output-files to `p.0, q.0, d.0`
cp r.ops.3 r.ops	constants changed : `IPS, IRS, ICP`, $\cdots$
@r ops 0	locate a 1-parameter extremum as a bifurcation; restart from `q.0`
@sv 1	save the output-files as `p.1, q.1, d.1`
cp r.ops.4 r.ops	constants changed : `IRS, ISP, ISW, NMX`
@r ops 1	switch branches to generate optimality starting data; restart from `q.1`
@ap 1	append the output-files to `p.1, q.1, d.1`
cp r.ops.5 r.ops	constants changed : `IRS, ISW, ICP, ISW`, $\cdots$
@r ops 1	compute 2-parameter branch of 1-parameter extrema; restart from `q.1`
@sv 2	save the output-files as `p.2, q.2, d.2`
cp r.ops.6 r.ops	constants changed : `IRS, ICP, EPSL, EPSU, NUZR`
@r ops 2	compute 3-parameter branch of 2-parameter extrema; restart from `q.2`
@sv 3	save the output-files as `p.3, q.3, d.3`

Table 13.5: Commands for running demo `ops`.

13.3 obv : Optimization for a BVP.

This demo illustrates use of the method of successive continuation for a boundary value optimization problem. A detailed description of the basic method, as well as a discussion of the specific application considered here, is given in Doedel, Keller & Kernévez (1991*b*). The required extended system is fully programmed here in the user-supplied subroutines in `obv.f`. For the case of periodic solutions the optimality system can be generated automatically; see the demo `ops`.

Consider the system

$$\begin{aligned} u_1'(t) &= u_2(t), \\ u_2'(t) &= -\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)}, \end{aligned} \tag{13.3}$$

where $p(u_1,\lambda_2,\lambda_3) \equiv u_1 + \lambda_2 u_1^2 + \lambda_3 u_1^4$, with boundary conditions

$$\begin{aligned} u_1(0) &= 0, \\ u_1(1) &= 0. \end{aligned} \tag{13.4}$$

The objective functional is

$$\omega = \int_0^1 (u_1(t)-1)^2 \, dt + \frac{1}{10}\sum_{k=1}^{3} \lambda_k^2.$$

The successive continuation equations are given by

$$\begin{aligned} u_1'(t) &= u_2(t), \\ u_2'(t) &= -\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)}, \\ w_1'(t) &= \lambda_1 e^{p(u_1,\lambda_2,\lambda_3)} p_{u_1} w_2(t) + 2\gamma(u_1(t)-1), \\ w_2'(t) &= -w_1(t), \end{aligned} \tag{13.5}$$

where

$$p_{u_1} \equiv \frac{\partial p}{\partial u_1} = 1 + 2\lambda_2 u_1 + 4\lambda_3 u_1^3,$$

with

$$\begin{aligned} u_1(0) = 0, \qquad & w_1(0) - \beta_1 = 0, \qquad & w_2(0) = 0, \\ u_1(1) = 0, \qquad & w_1(1) + \beta_2 = 0, \qquad & w_2(1) = 0, \end{aligned} \tag{13.6}$$

$$\int_0^1 [\omega - (u_1(t)-1)^2 - \frac{1}{10}\sum_{k=1}^{3}\lambda_k^2] \, dt = 0,$$

$$\int_0^1 [w_1^2(t) - \alpha_0] \, dt = 0,$$

$$\begin{aligned} &\textstyle\int_0^1 [-e^{p(u_1,\lambda_2,\lambda_3)} w_2(t) - \frac{1}{5}\gamma\lambda_1] \, dt = 0, \\ &\textstyle\int_0^1 [-\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)} u_1(t)^2 w_2(t) - \frac{1}{5}\gamma\lambda_2 - \tau_2] \, dt = 0, \\ &\textstyle\int_0^1 [-\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)} u_1(t)^4 w_2(t) - \frac{1}{5}\gamma\lambda_3 - \tau_3] \, dt = 0. \end{aligned} \tag{13.7}$$

In the first run the free equation parameter is λ_1. All adjoint variables are zero. Three extrema of the objective function are located. These correspond to branch points and, in the second run, branch switching is done at one of these. Along the bifurcating branch the adjoint variables become nonzero, while state variables and λ_1 remain constant. Any such non-trivial solution point can be used for continuation in two equation parameters, after fixing the L_2-norm of one of the adjoint variables. This is done in the third run. Along the resulting branch several two-parameter extrema are located by monotoring certain inner products. One of these is further continued in three equation parameters in the final run, where a three-parameter extremum is located.

COMMAND	ACTION
mkdir obv	create an empty work directory
cd obv	change directory
@dm obv	copy the demo files to the work directory
cp r.obv.1 r.obv	get the first constants-file
@r obv	locate 1-parameter extrema as branch points
@sv obv	save output-files as `p.obv, q.obv, d.obv`
cp r.obv.2 r.obv	constants changed : `IRS, ISW, NMX`
@r obv	compute a few step on the first bifurcating branch
@sv 1	save the output-files as `p.1, q.1, d.1`
cp r.obv.3 r.obv	constants changed : `IRS, ISW, NMX, ICP(3)`
@r obv 1	locate 2-parameter extremum; restart from `q.1`
@sv 2	save the output-files as `p.2, q.2, d.2`
cp r.obv.4 r.obv	constants changed : `IRS, ICP(4)`
@r obv 2	locate 3-parameter extremum; restart from `q.2`
@sv 3	save the output-files as `p.3, q.3, d.3`

Table 13.6: Commands for running demo `obv`.

Chapter 14

AUTO **Demos : Connecting orbits.**

14.1 fsh : A Saddle-Node Connection.

This demo illustrates the computation of travelling wave front solutions to the Fisher equation,

$$\begin{aligned} &w_t = w_{xx} + f(w), \qquad -\infty < x < \infty, \quad t > 0, \\ &f(w) \equiv w(1-w). \end{aligned} \tag{14.1}$$

We look for solutions of the form $w(x,t) = u(x+ct)$, where c is the wave speed. This gives the first order system

$$\begin{aligned} u_1'(z) &= u_2(z), \\ u_2'(z) &= cu_2(z) - f(u_1(z)). \end{aligned} \tag{14.2}$$

Its fixed point $(0,0)$ has two positive eigenvalues when $c > 2$. The other fixed point, $(1,0)$, is a saddle point. A branch of orbits connecting the two fixed points requires one free parameter; see Friedman & Doedel (1991). Here we take this parameter to be the wave speed c.

In the first run a starting connecting orbit is computed by continuation in the period T. This procedure can be used generally for time integration of an ODE with AUTO. Starting data in `STPNT` correspond to a point on the approximate stable manifold of $(1,0)$, with T small. In this demo the "free" end point of the orbit necessary approaches the unstable fixed point $(0,0)$. A computed orbit with sufficiently large T is then chosen as restart orbit in the second run, where, typically, one replaces T by c as continuation parameter. However, in the second run below, we also add a phase condition, and both c and T remain free.

COMMAND	ACTION
mkdir fsh	create an empty work directory
cd fsh	change directory
@dm fsh	copy the demo files to the work directory
cp r.fsh.1 r.fsh	get the first constants-file
@r fsh	continuation in the period T, with c fixed; no phase condition
@sv 0	save output-files as `p.0, q.0, d.0`
cp r.fsh.2 r.fsh	constants changed : `IRS, ICP, NINT, DS`
@r fsh 0	continuation in c and T, with active phase condition
@sv fsh	save output-files as `p.fsh, q.fsh, d.fsh`

Table 14.1: Commands for running demo `fsh`.

14.2 nag : A Saddle-Saddle Connection.

This demo illustrates the computation of traveling wave front solutions to Nagumo's equation,

$$\begin{aligned} &w_t = w_{xx} + f(w,a), \qquad -\infty < x < \infty, \quad t > 0, \\ &f(w,a) \equiv w(1-w)(w-a), \qquad 0 < a < 1. \end{aligned} \tag{14.3}$$

We look for solutions of the form $w(x,t) = u(x+ct)$, where c is the wave speed. This gives the first order system

$$\begin{aligned} u_1'(z) &= u_2(z), \\ u_2'(z) &= cu_2(z) - f(u_1(z),a), \end{aligned} \tag{14.4}$$

where $z = x + ct$, and $' = d/dz$. If $a = 1/2$ and $c = 0$ then there are two analytically known heteroclinic connections, one of which is given by

$$u_1(z) = \frac{e^{\frac{1}{2}\sqrt{2}z}}{1+e^{\frac{1}{2}\sqrt{2}z}}, \qquad u_2(z) = u_1'(z), \qquad -\infty < z < \infty.$$

The second heteroclinic connection is obtained by reflecting the phase plane representation of the first with respect to the u_1-axis. In fact, the two connections together constitute a heteroclinic cycle. One of the exact solutions is used below as starting orbit. To start from the second exact solution, change SIGN=-1 in the subroutine `STPNT` in `nag.f` and repeat the computations below; see also Friedman & Doedel (1991).

COMMAND	ACTION
mkdir nag	create an empty work directory
cd nag	change directory
@dm nag	copy the demo files to the work directory
cp r.nag.1 r.nag	get the first constants-file
@r nag	compute part of first branch of heteroclinic orbits
@sv nag	save output-files as `p.nag, q.nag, d.nag`
cp r.nag.2 r.nag	constants changed : `DS`
@r nag	compute first branch in opposite direction
@ap nag	append output-files to `p.nag, q.nag, d.nag`

Table 14.2: Commands for running demo `nag`.

14.3 stw : Continuation of Sharp Traveling Waves.

This demo illustrates the computation of sharp traveling wave front solutions to nonlinear diffusion problems of the form

$$w_t = A(w)w_{xx} + B(w)w_x^2 + C(w),$$

with $A(w) = a_1 w + a_2 w^2$, $B(w) = b_0 + b_1 w + b_2 w^2$, and $C(w) = c_0 + c_1 w + c_2 w^2$. Such equations can have *sharp traveling wave fronts* as solutions, i.e., solutions of the form $w(x,t) = u(x+ct)$ for which there is a z_0 such that $u(z) = 0$ for $z \geq z_0$, $u(z) \neq 0$ for $z < z_0$, and $u(z) \to$ *constant* as $z \to -\infty$. These solutions are actually generalized solutions, since they need not be differentiable at z_0.

Specifically, in this demo a homotopy path will be computed from an analytically known exact sharp traveling wave solution of

$$w_t = 2ww_{xx} + 2w_x^2 + w(1-w), \tag{1}$$

to a corresponding sharp traveling wave of

$$w_t = (2w + w^2)w_{xx} + ww_x^2 + w(1-w). \tag{2}$$

This problem is also considered in Doedel, Keller & Kernévez (1991*b*). For these two special cases the functions A, B, C are defined by the coefficients in Table 14.3.

	a_1	a_2	b_0	b_1	b_2	c_0	c_1	c_2
Case (1)	2	0	2	0	0	0	1	-1
Case (2)	2	1	0	1	0	0	1	-1

Table 14.3: Problem coefficients in demo `stw`.

With $w(x,t) = u(x+ct)$, $z = x + ct$, one obtains the reduced system

$$\begin{aligned} u_1'(z) &= u_2, \\ u_2'(z) &= [cu_2 - B(u_1)u_2^2 - C(u_1)]/A(u_1). \end{aligned} \tag{14.5}$$

To remove the singularity when $u_1 = 0$, we apply a nonlinear transformation of the independent variable (see Aronson (1980)), viz., $d/d\tilde{z} = A(u_1)d/dz$, which changes the above equation into

$$\begin{aligned} u_1'(\tilde{z}) &= A(u_1)u_2, \\ u_2'(\tilde{z}) &= cu_2 - B(u_1)u_2^2 - C(u_1). \end{aligned} \tag{14.6}$$

Sharp traveling waves then correspond to heteroclinic connections in this transformed system.

Finally, we map $[0, T] \to [0, 1]$ by the transformation $\xi = \tilde{z}/T$. With this scaling of the independent variable, the reduced system becomes

$$\begin{aligned} u_1'(\xi) &= TA(u_1)u_2, \\ u_2'(\xi) &= T[cu_2 - B(u_1)u_2^2 - C(u_1)]. \end{aligned} \tag{14.7}$$

For Case 1 this equation has a known exact solution, namely,

$$u(\xi) = \frac{1}{1 + exp(T\xi)}, \qquad v(\xi) = \frac{-\frac{1}{2}}{1 + exp(-T\xi)}.$$

This solution has wave speed $c = 1$. In the limit as $T \to \infty$ its phase plane trajectory connects the stationary points $(1, 0)$ and $(0, -\frac{1}{2})$.

The sharp traveling wave in Case 2 can now be obtained using the following homotopy. Let $(a_1, a_2, b_0, b_1, b_2) = (1 - \lambda)(2, 0, 2, 0, 0) + \lambda(2, 1, 0, 1, 0)$. Then as λ varies continuously from 0 to 1, the parameters $(a_1, a_2, b_0, b_1, b_2)$ vary continously from the values for Case 1 to the values for Case 2.

COMMAND	ACTION
mkdir stw	create an empty work directory
cd stw	change directory
@dm stw	copy the demo files to the work directory
cp r.stw.1 r.stw	get the constants-file
@r stw	continuation of the sharp traveling wave
@sv stw	save output-files as `p.stw, q.stw, d.stw`

Table 14.4: Commands for running demo `stw`.

Chapter 15

AUTO **Demos : Miscellaneous.**

15.1 pvl : Use of the Subroutine `PVLS`.

Consider Bratu's equation

$$\begin{aligned} u_1' &= u_2, \\ u_2' &= -p_1 e^{u_1}, \end{aligned} \tag{15.1}$$

with boundary conditions $u_1(0) = 0$, $u_1(1) = 0$. As in demo `exp`, a solution curve requires one free parameter; here p_1.

Note that additional parameters are specified in the user-supplied subroutine `PVLS` in file `pvls.f`, namely, p_2 (the L_2-norm of u_1), p_3 (the minimum of u_2 on the space-interval $[0, 1]$), p_4 (the boundary value $u_2(0)$). These additional parameters should be considered as "solution measures" for output purposes; they should not be treated as true continuation parameters.

Note also that four free parameters are specified in the AUTO-constants file `r.pvl.1`, namely, p_1, p_2, p_3, and p_4. The first one in this list, p_1, is the true continuation parameter. The parameters p_2, p_3, and p_4 are *overspecified* so that their values will appear in the output. However, *it is essential that the true continuation parameter appear first.* For example, it would be an error to specify the parameters in the following order : p_2, p_1, p_3, p_4.

In general, true continuation parameters must appear first in the parameter-specification in the AUTO constants-file. Overspecified parameters will be printed, and can be defined in `PVLS`, but they are not part of the intrinsic continuation procedure.

As this demo also illustrates (see the `UZR` values in `r.pvl.1`), labeled solutions can also be output at selected values of the overspecified parameters.

COMMAND	ACTION
mkdir pvl	create an empty work directory
cd pvl	change directory
@dm pvl	copy the demo files to the work directory
cp r.pvl.1 r.pvl	get the constants-file
@r pvl	compute a solution branch
@sv pvl	save output-files as `p.pvl, q.pvl, d.pvl`

Table 15.1: Commands for running demo `pvl`.

15.2 ext : Spurious Solutions to BVP.

This demo illustrates the computation of spurious solutions to the boundary value problem

$$\begin{aligned} &u_1' - u_2 = 0, \\ &u_2' + \lambda^2\pi^2 \sin(u_1 + u_1^2 + u_1^3) = 0, \qquad t \in [0,1], \\ &u_1(0) = 0, \quad u_1(1) = 0. \end{aligned} \tag{15.2}$$

Here the differential equation is discretized using a fixed uniform mesh. This results in spurious solutions that disappear when an adaptive mesh is used. See the AUTO-constant `IAD` in Section 6.3. This example is also considered in Beyn & Doedel (1981) and Doedel, Keller & Kernévez (1991*b*).

COMMAND	ACTION
mkdir ext	create an empty work directory
cd ext	change directory
@dm ext	copy the demo files to the work directory
cp r.ext.1 r.ext	get the first constants-file
@r ext	detect bifurcations from the trivial solution branch
@sv ext	save output-files as `p.ext, q.ext, d.ext`
cp r.ext.2 r.ext	constants changed : `IRS, ISW, NUZR`
@r ext	compute a bifurcating branch containing spurious bifurcations
@ap ext	append output-files to `p.ext, q.ext, d.ext`

Table 15.2: Commands for running demo `ext`.

15.3 tim : A Test Problem for Timing AUTO.

This demo is a boundary value problem with variable dimension `NDIM`. It can be used to time the performance of AUTO for various choices of `NDIM` (which must be even), `NTST`, and `NCOL`. The equations are

$$\begin{aligned} u_i' &= u_i, \\ v_i' &= -p_1 \ e(u_i), \end{aligned} \tag{15.3}$$

$i = 1, \cdots,$`NDIM`$/2$, with boundary conditions $u_i(0) = 0$, $u_i(1) = 0$. Here

$$e(u) = \sum_{k=0}^{n} \frac{u^k}{k!} \ ,$$

with $n = 25$. The computation requires 10 full LU-decompositions of the linearized system that arises from Newton's method for solving the collocation equations. The commands for running the timing problem for a particular choice of `NDIM`, `NTST`, and `NCOL` are given below. (Note that if `NDIM` is changed then `NBC` must be changed accordingly.)

COMMAND	ACTION
mkdir tim	create an empty work directory
cd tim	change directory
@dm tim	copy the demo files to the work directory
cp r.tim.1 r.tim	get the first constants-file
@r tim	Timing run
@sv tim	save output-files as `p.tim, q.tim, d.tim`

Table 15.3: Commands for running demo `tim`.

Chapter 16

HomCont.

16.1 Introduction.

HomCont is a collection of subroutines for the continuation of homoclinic solutions to ODEs in two or more parameters. The accurate detection and multi-parameter continuation of certain codimension-two singularities is allowed for, including all known cases that involve a unique homoclinic orbit at the singular point. Homoclinic connections to hyperbolic and non-hyperbolic equilibria are allowed as are certain heteroclinic orbits. Homoclinic orbits in reversible systems can also be computed. The theory behind the methods used is explained in Champneys & Kuznetsov (1994), Bai & Champneys (1996), Sandstede (1995*b*, 1995*c*), Champneys, Kuznetsov & Sandstede (1996) and references therein. The final cited paper contains a concise description of the present version.

The current implementation of HomCont must be considered as experimental, and updates are anticipated. The HomCont subroutines are in the file `auto/97/src/autlib5.f`. Expert users wishing to modify the routines may look there. Note also that at present, HomCont can be run only in AUTO Command Mode and not with the GUI.

16.2 HomCont Files and Subroutines.

In order to run HomCont one must prepare an equations file `xxx.f`, where `xxx` is the name of the example, and two constants-files `r.xxx` and `s.xxx`. The first two of these files are in the standard AUTO format, whereas the `s.xxx` file contains constants that are specific to homoclinic continuation. The choice `IPS`=9 in `r.xxx` specifies the problem as being homoclinic continuation, in which case `s.xxx` is required.

The equation-file `kpr.f` serves as a sample for new equation files. It contains the Fortran subroutines `FUNC`, `STPNT`, `PVLS`, `BCND`, `ICND` and `FOPT`. The final three are dummy subroutines which are never needed for homoclinic continuation. Note a minor difference in `STPNT` and `PVLS` with other AUTO equation-files, in that the common block `/BLHOM/` is required.

The constants-file `r.xxx` is identical in format to other AUTO constants-files. Note that the values of the constants `NBC` and `NINT` are irrelevant, as these are set automatically by the choice `IPS`=9. Also, the choice `JAC=1` is strongly recommended, because the Jacobian is used extensively for calculating the linearization at the equilibria and hence for evaluating boundary conditions and certain test functions. However, note that `JAC=1` does not necessarily mean that AUTO will use the analytically specified Jacobian for continuation.

16.3 HomCont-Constants.

An example for the additional file `s.xxx` is listed below:

```
1 2 1 1 1    NUNSTAB,NSTAB,IEQUIB,ITWIST,ISTART
0            NREV,(/,I,IREV(I)),I=1,NREV)
1            NFIXED,(/,I,IFIXED(I)),I=1,NFIXED)
  13
1            NPSI,(/,I,IPSI(I)),I=1,NPSI)
  9 10 13
```

The constants specified in `s.xxx` have the following meaning.

16.3.1 NUNSTAB

Number of unstable eigenvalues of the left-hand equilibrium (the equilibrium approached by the orbit as $t \to -\infty$).

16.3.2 NSTAB

Number of stable eigenvalues of the right-hand equilibrium (the equilibrium approached by the orbit as $t \to +\infty$).

16.3.3 IEQUIB

- `IEQUIB=0` : Homoclinic orbits to hyperbolic equilibria; the equilibrium is specified explicitly in `PVLS` and stored in `PAR(11+I)`, `I=1,NDIM`.

- `IEQUIB=1` : Homoclinic orbits to hyperbolic equilibria; the equilibrium is solved for during continuation. Initial values for the equilibrium are stored in `PAR(11+I)`, `I=1,NDIM` in `STPNT`.

- `IEQUIB=2` : Homoclinic orbits to a saddle-node; initial values for the equilibrium are stored in `PAR(11+I)`, `I=1,NDIM` in `STPNT`.

- `IEQUIB=-1` : Heteroclinic orbits to hyperbolic equilibria; the equilibria are specified explicitly in `PVLS` and stored in `PAR(11+I)`, `I=1,NDIM` (left-hand equilibrium) and `PAR(11+I)`, `I=NDIM+1,2*NDIM` (right-hand equilibrium).
- `IEQUIB=-2` : Heteroclinic orbits to hyperbolic equilibria; the equilibria are solved for during continuation. Initial values are specified in `STPNT` and stored in `PAR(11+I)`, `I=1,NDIM` (left-hand equilibrium), `PAR(11+I)`, `I=NDIM+1,2*NDIM` (right-hand equilibrium).

16.3.4 ITWIST

- `ITWIST=0` : the orientation of the homoclinic orbit is not computed.
- `ITWIST=1` : the orientation of the homoclinic orbit is computed. For this purpose, the adjoint variational equation is solved for the unique bounded solution. If `IRS = 0`, an initial solution to the adjoint equation must be specified as well. However, if `IRS>0` and `ITWIST` has just been increased from zero, then AUTO will automatically generate the initial solution to the adjoint. In this case, a dummy Newton-step should be performed, see Section 16.7 for more details.

16.3.5 ISTART

- `ISTART=1` : This option is no obsolete in the current version. It may be used as a flag that a solution is to be restarted from a previously computed point or from numerical data converted into AUTO format using `@fc`. In this case `IRS>0`.
- `ISTART=2` : If `IRS=0`, an explicit solution must be specified in the subroutine `STPNT` in the usual format.
- `ISTART=3` : The "homotopy" approach is used for starting, see Section 16.7 for more details. Note that this is not available with the choice `IEQUIB=2`.

16.3.6 NREV, IREV

If `NREV=1` then it is assumed that the system is reversible under the transformation $t \to -t$ and $U(i) \to -U(i)$ for all i with `IREV(i)>0`. Then only half the homoclinic solution is solved for with right-hand boundary conditions specifying that the solution is symmetric under the reversibility (see Champneys & Spence (1993)). The number of free parameters is then reduced by one. Otherwise `IREV=0`.

16.3.7 NFIXED, IFIXED

Number and labels of test functions that are held fixed. E.g., with `NFIXED=1` one can compute a locus in one extra parameter of a singularity defined by test function `PSI(IFIXED(1))=0`.

16.3.8 `NPSI, IPSI`

Number and labels of activated test functions for detecting homoclinic bifurcations, see Section 16.6 for a list. If a test function is activated then the corresponding parameter (`IPSI(I)+20`) must be added to the list of continuation parameters `NICP,(ICP(I),I=1 NICP)` and zero of this parameter added to the list of user-defined output points `NUZR, (/,I,PAR(I)),I=1, NUZR` in `r.xxx`.

16.4 Restrictions on HomCont Constants.

Note that certain combinations of these constants are not allowed in the present implementation. In particular,

- The computation of orientation `ITWIST=1` is not implemented for `IEQUIB<0` (heteroclinic orbits), `IEQUIB=2` (saddle-node homoclinics), `IREV=1` (reversible systems), `ISTART=3` (homotopy method for starting), or if the equilibrium contains complex eigenvalues in its linearization.
- The homotopy method `ISTART=3` is not fully implemented for heteroclinic orbits `IEQUIB<0`, saddle-node homoclinic orbits `IEQUIB=2` or reversible systems `IREV=1`.
- Certain test functions are not valid for certain forms of continuation (see Section 16.6 below); for example `PSI(13)` and `PSI(14)` only make sense if `ITWIST=1` and `PSI(15)` and `PSI(16)` only apply to `IEQUIB=2`.

16.5 Restrictions on the Use of `PAR`.

The parameters `PAR(1)` – `PAR(9)` can be used freely by the user. The other parameters are used as follows :

- `PAR(11)` : The value of `PAR(11)` equals the length of the time interval over which a homoclinic solution is computed. Also referred to as "period". This must be specified in `STPNT`.
- `PAR(10)` : If `ITWIST=1` then `PAR(10)` is used internally as a dummy parameter so that the adjoint equation is well-posed.
- `PAR(12)-PAR(20)` : These are used for specifying the equilibria and (if `ISTART=3`) the artificial parameters of the homotopy method (see Section 16.7 below).
- `PAR(21)-PAR(36)` : These parameters are used for storing the test functions (see Section 16.6).

The output is in an identical format to AUTO except that additional information at each computed point is written in `fort.9`. This information comprises the eigenvalues of the (left-hand) equilibrium, the values of each activated test function and, if `ITWIST=1`, whether the saddle homoclinic loop is orientable or not. Note that the statement about orientability is only meaningful if the leading eigenvalues are not complex and the homoclinic solution is not in a flip configuration, that is, none of the test functions ψ_i for $i = 11, 12, 13, 14$ is zero (or close to zero), see Section 16.6. Finally, the values of the `NPSI` activated test functions are written.

16.6 Test Functions.

Codimension-two homoclinic orbits are detected along branches of codim 1 homoclinics by locating zeroes of certain test functions ψ_i. The test functions that are "switched on" during any continuation are given by the choice of the labels i, and are specified by the parameters `NPSI,(/,I,IPSI(I)),I=1,NPSI)` in `s.xxx`. Here `NPSI` gives the number of activated test functions and `IPSI(1),...,IPSI(NPSI)` give the labels of the test functions (numbers between 1 and 16). A zero of each labeled test function defines a certain codimension-two homoclinic singularity, specified as follows. The notation used for eigenvalues is the same as that in Champneys & Kuznetsov (1994) or Champneys et al. (1996).

- `i=1` : Resonant eigenvalues (neutral saddle); $\mu_1 = -\lambda_1$.
- `i=2` : Double real leading stable eigenvalues (saddle to saddle-focus transition); $\mu_1 = \mu_2$.
- `i=3` : Double real leading unstable eigenvalues (saddle to saddle-focus transition); $\lambda_1 = \lambda_2$.
- `i=4` : Neutral saddle, saddle-focus or bi-focus (includes `i=1`); $\mathrm{Re}(\mu_1) = -\mathrm{Re}(\lambda_1)$.
- `i=5` : Neutrally-divergent saddle-focus (stable eigenvalues complex); $\mathrm{Re}(\lambda_1) = -\mathrm{Re}(\mu_1) - \mathrm{Re}(\mu_2)$.
- `i=6` : Neutrally-divergent saddle-focus (unstable eigenvalues complex); $\mathrm{Re}(\mu_1) = -\mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_2)$.
- `i=7` : Three leading eigenvalues (stable); $\mathrm{Re}(\lambda_1) = -\mathrm{Re}(\mu_1) - \mathrm{Re}(\mu_2)$.
- `i=8` : Three leading eigenvalues (unstable); $\mathrm{Re}(\mu_1) = -\mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_2)$.
- `i=9` : Local bifurcation (zero eigenvalue or Hopf): number of stable eigenvalues decreases; $\mathrm{Re}(\mu_1) = 0$.
- `i=10` : Local bifurcation (zero eigenvalue or Hopf): number of unstable eigenvalues decreases; $\mathrm{Re}(\lambda_1) = 0$.
- `i=11` : Orbit flip with respect to leading stable direction (e.g., 1D unstable manifold).

- `i=12` : Orbit flip with respect to leading unstable direction, (e.g., 1D stable manifold).
- `i=13` : Inclination flip with respect to stable manifold (e.g., 1D unstable manifold).
- `i=14` : Inclination flip with respect to unstable manifold (e.g., 1D stable manifold).
- `i=15` : Non-central homoclinic to saddle-node (in stable manifold).
- `i=16` : Non-central homoclinic to saddle-node (in unstable manifold).

Expert users may wish to add their own test functions by editing the function `PSIHO` in `autlib5.f`. *It is important to remember that, in order to specify activated test functions, it is required to also add the corresponding label* $+20$ *to the list of continuation parameters and a zero of this parameter to the list of user-defined output points. Having done this, the corresponding parameters are output to the screen and zeros are accurately located.*

16.7 Starting Strategies.

There are four possible starting procedures for continuation.

(i) Data can be read from a previously-obtained output point from AUTO (e.g., from continuation of a periodic orbit up to large period; note that the end-point of the data stored must be close to the equilibrium). These data can be read from fort.8 (saved to `q.xxx`) by making `IRS` correspond to the label of the data point in question.

(ii) Data from numerical integration (e.g., computation of a stable periodic orbit, or an approximate homoclinic obtained by shooting) can be read in from a data file using the general AUTO utility `@fc` (see earlier in the manual). The numerical data should be stored in a file `xxx.dat`, in multi-column format according to the read statement

```
READ(...,*) T(J),(U(I,J),I=1,NDIM)
```

where `T` runs in the interval `[0,1]`. After running `@fc` the restart data is stored in the format of a previously computed solution in `q.dat`. When starting from this solution `IRS` should be set to 1 and the value of `ISTART` is irrelevant.

(iii) By setting `ISTART=2`, an explicit homoclinic solution can be specified in the routine `STPNT` in the usual AUTO format, that is `U=...(T)` where `T` is scaled to lie in the interval`[0,1]`.

(iv) The choice `ISTART=3`, allows for a homotopy method to be used to approach a homoclinic orbit starting from a small approximation to a solution to the linear problem in the unstable manifold (Doedel, Friedman & Monteiro 1993). For details of implementation, the reader is referred to Section 5.1.2. of Champneys & Kuznetsov (1994), under the simplification that we do not solve for the adjoint $u(t)$ here. The basic idea is to start with a small solution in the unstable manifold, and perform continuation in `PAR(11)`$=2T$ and dummy initial-condition parameters ξ_i in order to satisfy the correct right-hand boundary conditions, which are defined by zeros of other dummy parameters ω_i. More precisely, the left-hand end point is placed in the tangent space to the unstable manifold of the saddle and is characterized by `NUNSTAB` coordinates ξ_i satisfying the condition

$$\xi_1^2 + \xi_2^2 + \ldots + \xi_{\texttt{NUNSTAB}}^2 = \epsilon_0^2,$$

where ϵ_0 is a user-defined small number. At the right-hand end point, `NUNSTUB` values ω_i measure the deviation of this point from the tangent space to the stable manifold of the saddle.

Suppose that `IEQUIB=0,1` and set `IP=12+IEQUIB*NDIM`. Then

`PAR(IP)`	:	ϵ_0
`PAR(IP+i)`	:	$\xi_{\texttt{i}}$, `i=1,2,...,NUNSTAB`
`PAR(IP+NUNSTAB+i)`	:	$\omega_{\texttt{i}}$, `i=1,2,...,NUNSTAB`

Notice that in order to avoid interference with the test functions (i.e. `PAR(21)-PAR(36)`*), one must have* `IP+2*NUNSTAB < 21`*.*

If an ω_i is vanished, it can be frozen while another dummy or system parameter is allowed to vary in order to make consequently all $\omega_i = 0$. The resulting final solution gives the initial homoclinic orbit provided the right-hand end point is sufficiently close to the saddle. See Chapter 19 for an example, however, we recommend the homotopy method only for "expert users".

To compute the orientation of a homoclinic orbit (in order to detect inclination-flip bifurcations) it is necessary to compute, in tandem, a solution to the modified adjoint variational equation, by setting `ITWIST=1`. In order to obtain starting data for such a computation when restarting from a point where just the homoclinic is computed, upon increasing `ITWIST` to 1, AUTO generates trivial data for the adjoint. Because the adjoint equations are linear, only a single step of Newton's method is required to enable these trivial data to converge to the correct unique bounded solution. This can be achieved by making a single continuation step in a trivial parameter (i.e. a parameter that does not appear in the problem).

Decreasing `ITWIST` to 0 automatically deletes the data for the adjoint from the continuation problem.

16.8 Notes on Running HomCont Demos.

HomCont demos are given in the following chapters. To copy all files of a demo `xxx` (for example, `san`), move to a clean directory and type *@dm xxx*. Simply typing *make* or *make all* will then automatically execute all runs of the demo. To automatically run a demo in "step-by-step" mode, type *make first*, *make second*, etc., to run each separate computation of the demo. At each step, the user is encouraged to plot the data saved by using the command *@p* (e.g., *@p 1* plots the data saved in `p.1` and `q.1`).

Of course, in a real application, the runs will not have been prepared in advance, and AUTO-commands must be used. Such commands can be found in a table at the end of each chapter. Note that the sequence of detailed AUTO-commands given in these tables can be abbreviated, as illustrated in Table 16.1 and Table 16.2 for two representative runs of HomCont demo `san`.

The user is encouraged to copy the format of one of these demos when constructing new examples.

The output of the HomCont demos reproduced in the following chapters is somewhat machine dependent, as already noted in Section 8.4. In exceptional circumstances, AUTO may reach its maximum number of steps `NMX` before a certain output point, or the label of an output point may change. In such case the user may have to make appropriate changes in the AUTO constants-files.

COMMAND	ACTION
cp r.san.1 r.san	get the AUTO constants-file
cp s.san.1 s.san	get the HomCont constants-file
@h san	run AUTO/HomCont
@sv 6	save output-files as `p.6, q.6, d.6`
@H san 1	
@sv 6	

Table 16.1: These two sets of AUTO-Commands are equivalent.

COMMAND	ACTION
cp r.san.9 r.san	get the AUTO constants-file
cp s.san.9 s.san	get the HomCont constants-file
@h san 6	run AUTO/HomCont; restart solution read from `q.6`
@ap 6	append output-files to `p.6, q.6, d.6`
@H san 9 6	
@ap 6	

Table 16.2: These two sets of AUTO-Commands are equivalent.

Chapter 17

HomCont **Demo : san.**

17.1 Sandstede's Model.

Consider the system (Sandstede 1995*a*)

$$\begin{array}{rcl} \dot{x} & = & a\,x + b\,y - a\,x^2 + (\tilde{\mu} - \alpha\,z)\,x\,(2 - 3x) \\ \dot{y} & = & b\,x + a\,y - \frac{3}{2}\,b\,x^2 - \frac{3}{2}\,a\,x\,y - (\tilde{\mu} - \alpha\,z)\,2\,y \\ \dot{z} & = & c\,z + \mu\,x + \gamma\,x\,y + \alpha\,\beta\,(x^2\,(1-x) - y^2) \end{array} \tag{17.1}$$

as given in the file `san.f`. Choosing the constants appearing in (17.1) appropriately allows for computing inclination and orbit flips as well as non-orientable resonant bifurcations, see (Sandstede 1995*a*) for details and proofs. The starting point for all calculations is $a = 0$, $b = 1$ where there exists an explicit solution given by

$$(x(t), y(t), z(t)) = \left(1 - \left(\frac{1-e^t}{1+e^t}\right)^2, 4\,e^t\,\frac{1-e^t}{(1+e^t)^3}, 0\right).$$

This solution is specified in the routine `STPNT`.

17.2 Inclination Flip.

We start by copying the demo to the current work directory and running the first step

@dm san
make first

This computation starts from the analytic solution above with $a = 0$, $b = 1$, $c = -2$, $\alpha = 0$, $\beta = 1$ and $\gamma = \mu = \tilde{\mu} = 0$. The homoclinic solution is followed in the parameters $(a, \tilde{\mu})$ `=(PAR(1), PAR(8))` up to $a = 0.25$. The output is summarised on the screen as

```
   BR  PT  TY LAB    PAR(1)        L2-NORM           PAR(8)
    1   1  EP   1  0.000000E+00  4.000000E-01 ...  0.000000E+00
    1   5  UZ   2  2.500000E-01  4.030545E-01 ... -3.620329E-11
    1  10  EP   3  7.384434E-01  4.339575E-01 ... -9.038826E-09
```

and saved in more detail as `p.1`, `q.1` and `d.1`.

Next we want to add a solution to the adjoint equation to the solution obtained at $a = 0.25$. This is achieved by making the change `ITWIST = 1` saved in `s.san.2`, and `IRS = 2`, `NMX = 2` and `ICP(1) = 9` saved in `r.san.2`. We also disable any user-defined functions `NUZR=0`. The computation so-defined is a single step in a trivial parameter `PAR(9)` (namely a parameter that does not appear in the problem). The effect is to perform a Newton step to enable AUTO to converge to a solution of the adjoint equation.

make second

The output is stored in `p.2`, `q.2` and `d.2`.

We are now ready to perform continuation of the homoclinic plus adjoint in $(\alpha, \tilde{\mu})$ `=(PAR(4), PAR(8))` by changing the constants (stored in `r.san.3`) to read `IRS = 4`, `NMX = 50` and `ICP(1) = 4`. We also add `PAR(10)` to the list of continuation parameters `NICP,(ICP(I),I=1 NICP)`. Here `PAR(10)` is a dummy parameter used in order to make the continuation of the adjoint well posed. Theoretically, it should be zero if the computation of the adjoint is successful (Sandstede 1995*a*). The test functions for detecting resonant bifurcations (`ISPI(1)=1`) and inclination flips (`ISPI(1)=13`) are also activated. Recall that this should be specified in three ways. First we add `PAR(21)` and `PAR(33)` to the list of continuation parameters in `r.san.3`, second we set up user defined output at zeros of these parameters in the same file, and finally we set `NPSI=2 (IPSI(1),IPSI(2))=1,13` in `s.san.3`. We also add to `r.san.3` another user zero for detecting when `PAR(4)=1.0`. Running

make third

reads starting data from `q.2` and outputs to the screen

```
 BR  PT  TY LAB    PAR(4)      ...     PAR(8)         PAR(10)     ...     PAR(33)
  1  20         5  7.847219E-01 ... -3.001440E-11 -4.268884E-09 ... -1.441124E+01
  1  27  UZ     6  1.000000E+00 ... -3.844872E-11 -4.460769E-09 ... -5.701675E+00
  1  35  UZ     7  1.230857E+00 ... -5.833977E-11 -4.530541E-09 ...  9.434843E-06
  1  40         8  1.383969E+00 ... -8.133899E-11 -4.671817E-09 ...  1.348810E+00
  1  50  EP     9  1.695209E+00 ... -1.386324E-10 -5.098460E-09 ...  5.311065E-01
```

Full output is stored in `p.3`, `q.3` and `d.3`. Note that the artificial parameter $\epsilon =$`PAR(10)` is zero within the allowed tolerance. At label `7`, a zero of test function ψ_{13} has been detected which corresponds to an inclination flip with respect to the stable manifold. That the orientation of the homoclinic loop changes as the branch passes through this point can be read from the information in `d.3`. However in `d.3`, the line

```
ORIENTABLE (    0.2982090775D-03)
```

at PT=35 would seems to contradict the detection of the inclination flip at this point. Nonetheless, the important fact is the zero of the test function; and note that the value of the variable indicating the orientation is small compared to its value at the other regular points. Data for the adjoint equation at LAB= 5, 7 and 9 at and on either side of the inclination flip are presented in Fig. 17.1. The switching of the solution between components of the leading unstable left eigenvector is apparent. Finally, we remark that the Newton step in the dummy parameter PAR(20) performed above is crucial to obtain convergence. Indeed, if instead we try to continue the homoclinic orbit and the solution of the adjoint equation directly by setting

```
  ITWIST = 1   IRS = 2   NMX = 50   ICP(1) = 4   NPUSZR = 0
```

(as saved in r.san.4) and running

make fourth

we obtain a no convergence error.

17.3 Non-orientable Resonant Eigenvalues.

Inspecting the output saved in the third run, we observe the existence of a non-orientable homoclinic orbit at label 7 corresponding to N=40. We restart at this label, with the first continuation parameter being once again a =PAR(1), by changing constants and storing them in r.san.5 according to

```
   IRS = 7     DS = -0.05D0    NMX = 20    ICP(1) = 1
```

Running,

make fifth

the output at label 10

```
  BR     PT  TY LAB    PAR(1)            PAR(8)         PAR(10)        PAR(21)
  1      8  UZ  10 -1.304570E-07  ... 3.874816E-12 -1.468457E-09 -2.609139E-07
```

indicates that AUTO has detected a zero of PAR(21), implying that a non-orientable resonant bifurcation occurred at that point.

17.4 Orbit Flip.

In this section we compute an orbit flip. To this end we restart from the original explicit solution, without computing the orientation. We begin by separately performing continuation in $(\alpha, \tilde{\mu})$, $(\beta, \tilde{\mu})$, $(a, \tilde{\mu})$, $(b, \tilde{\mu})$ and $(\mu, \tilde{\mu})$ in order to reach the parameter values $(a, b, \alpha, \beta, \mu) = (0.5, 3, 1, 0, 0.25)$. The sequence of continuations up to the desired parameter values are run via

make sixth
make seventh
make eighth
make ninth
make tenth

with appropriate continuation parameters and user output values set in the corresponding `r.san.xx`. All the output is saved to `q.6`.

The final saved point `LAB=10` contains a homoclinic solution at the desired parameter values. From here we perform continuation in the negative direction of $(\mu, \tilde{\mu}) =$ (`PAR(7)`,`PAR(8)`) with the test function ψ_{11} for orbit flips with respect to the stable manifold activated.

make eleventh

The output detects an inclination flip (by a zero of `PAR(31)`) at `PAR(7)=0`

```
 BR    PT  TY LAB    PAR(7)       ...    PAR(8)        PAR(31)
 1     5  UZ  12  2.394737E-07    ...  6.434492E-08 -4.133994E-06
```

at which parameter value the homoclinic orbit is contained in the (x, y)-plane (see Fig. 17.2).

Finally, we demonstrate that the orbit flip can be continued as three parameters (`PAR(6)`, `PAR(7)`, `PAR(8)`) are varied.

make twelfth

```
BR    PT  TY LAB    PAR(7)        ...    PAR(8)        PAR(6)
  1     5       14 -5.374538E-19  ... -1.831991E-10 -3.250000E-01
  1    10       15 -6.145911E-19  ... -2.628607E-10 -8.250001E-01
  1    15       16 -4.947133E-19  ... -2.361151E-10 -1.325000E+00
  1    20  EP   17 -5.792940E-19  ... -3.075527E-10 -1.825000E+00
```

The orbit flip continues to be defined by a planar homoclinic orbit at `PAR(7)=PAR(8)=0`.

17.5 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir san	create an empty work directory
cd san	change directory
@dm san	copy the demo files to the work directory
cp r.san.1 r.san	get the AUTO constants-file
cp s.san.1 s.san	get the HomCont constants-file
@h san	continuation in `PAR(1)`
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.san.2 r.san	get the AUTO constants-file
cp s.san.2 s.san	get the HomCont constants-file
@h san 1	generate adjoint variables; restart from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`
cp r.san.3 r.san	get the AUTO constants-file
cp s.san.3 s.san	get the HomCont constants-file
@h san 2	continue homoclinic orbit and adjoint; restart from `q.2`
@sv 3	save output-files as `p.3, q.3, d.3`
cp r.san.4 r.san	get the AUTO constants-file
cp s.san.4 s.san	get the HomCont constants-file
@h san 1	no convergence without dummy step; restart from `q.1`
@sv 4	save output-files as `p.4, q.4, d.4`
cp r.san.5 r.san	get the AUTO constants-file
cp s.san.5 s.san	get the HomCont constants-file
@h san 3	continue non-orientable orbit; restart from `q.3`
@sv 5	save output-files as `p.5, q.5, d.5`

Table 17.1: Detailed AUTO-Commands for running demo `san`.

COMMAND	ACTION
cp r.san.6 r.san	get the AUTO constants-file
cp s.san.6 s.san	get the HomCont constants-file
@h san	restart and homotopy to `PAR(4)`=1.0
@sv 6	save output-files as `p.6, q.6, d.6`
cp r.san.7 r.san	get the AUTO constants-file
cp s.san.7 s.san	get the HomCont constants-file
@h san 6	homotopy to `PAR(5)`=0.0; restart from `q.6`
@ap 6	append output-files to `p.6, q.6, d.6`
cp r.san.8 r.san	get the AUTO constants-file
cp s.san.8 s.san	get the HomCont constants-file
@h san 6	homotopy to `PAR(1)`=0.5; restart from `q.6`
@ap 6	append output-files to `p.6, q.6, d.6`
cp r.san.9 r.san	get the AUTO constants-file
cp s.san.9 s.san	get the HomCont constants-file
@h san 6	homotopy to `PAR(2)`=3.0; restart from `q.6`
@ap 6	append output-files to `p.6, q.6, d.6`
cp r.san.10 r.san	get the AUTO constants-file
cp s.san.10 s.san	get the HomCont constants-file
@h san 6	homotopy to `PAR(7)`=0.25; restart from `q.6`
@ap 6	append output-files to `p.6, q.6, d.6`
cp r.san.11 r.san	get the AUTO constants-file
cp s.san.11 s.san	get the HomCont constants-file
@h san 6	continue in `PAR(7)` to detect orbit flip; restart from `q.6`
@sv 11	save output-files as `p.11, q.11, d.11`
cp r.san.12 r.san	get the AUTO constants-file
cp s.san.12 s.san	get the HomCont constants-file
@h san 11	three-parameter continuation of orbit flip; restart from `q.11`
@sv 12	save output-files as `p.12, q.12, d.12`

Table 17.2: Detailed AUTO-Commands for running demo `san`.

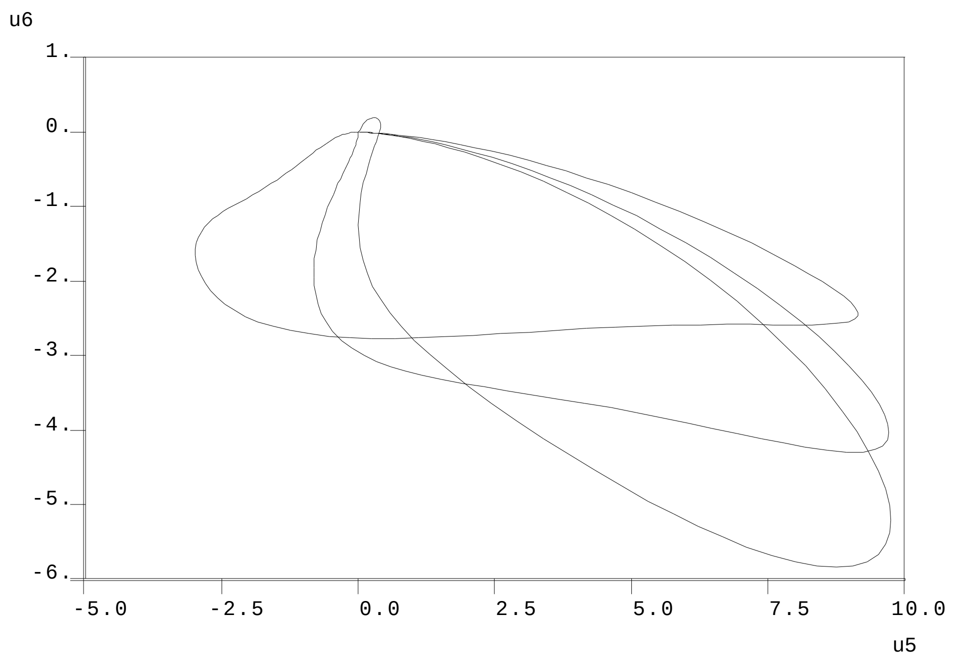

Figure 17.1: Second versus third component of the solution to the adjoint equation at labels `5`, `7` and `9`

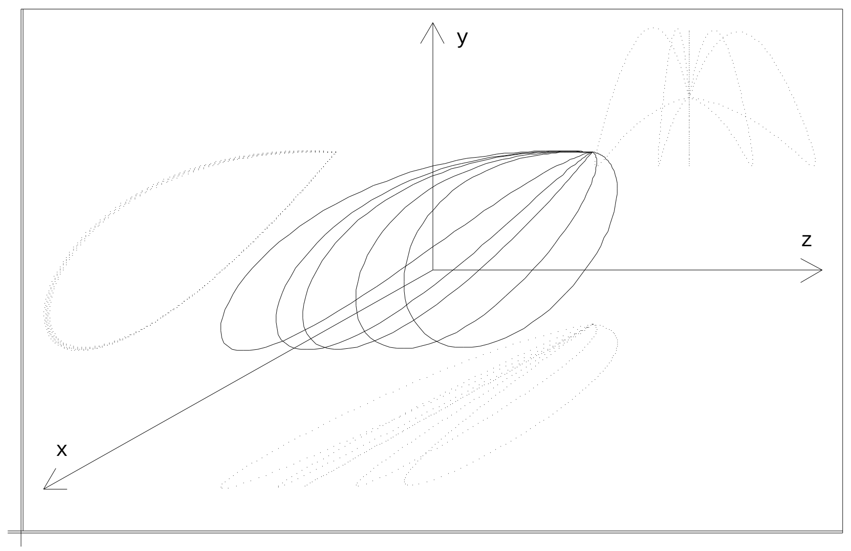

Figure 17.2: Orbits on either side of the orbit flip bifurcation. The critical orbit is contained in the (x, y)-plane

Chapter 18

HomCont **Demo : mtn.**

18.1 A Predator-Prey Model with Immigration.

Consider the following system of two equations (Scheffer 1995)

$$\begin{aligned} \dot{X} &= RX\left(1-\frac{X}{K}\right)-\frac{A_1XY}{B_1+X}+D_0K \\ \dot{Y} &= E_1\frac{A_1XY}{B_1+X}-D_1Y-\frac{A_2ZY^2}{B_2^2+Y^2}. \end{aligned} \tag{18.1}$$

The values of all parameters except (K, Z) are set as follows :

$$R=0.5,\ A_1=0.4,\ B_1=0.6,\ D_0=0.01,\ E_1=0.6,\ A_2=1.0,\ B_2=0.5,\ D_1=0.15.$$

The parametric portrait of the system (18.1) on the (Z, K)-plane is presented in Figure 18.1. It contains fold ($t_{1,2}$) and Hopf (H) bifurcation curves, as well as a homoclinic bifurcation curve P. The fold curves meet at a cusp singular point C, while the Hopf and the homoclinic curves originate at a Bogdanov-Takens point BT. Only the homoclinic curve P will be considered here, the other bifurcation curves can be computed using `AUTO` or, for example, LOCBIF (Khibnik, Kuznetsov, Levitin & Nikolaev 1993).

18.2 Continuation of Central Saddle-Node Homoclinics.

Local bifurcation analysis shows that at $K = 6.0,\ Z = 0.06729762\ldots$, the system has a saddle-node equilibrium

$$(X^0, Y^0) = (5.738626\ldots, 0.5108401\ldots),$$

with one zero and one negative eigenvalue. Direct simulations reveal a homoclinic orbit to this saddle-node, departing and returning along its central direction (i.e., tangent to the null-vector).

Starting from this solution, stored in the file `mtn.dat`, we continue the saddle-node central homoclinic orbit with respect to the parameters K and Z by copying the demo and running it

@dm mtn
make first

The file `mtn.f` contains approximate parameter values

$$K = \texttt{PAR(1)} = 6.0,\ Z = \texttt{PAR(2)} = 0.06729762,$$

as well as the coordinates of the saddle-node

$$X^0 = \texttt{PAR(12)} = 5.738626,\ Y^0 = \texttt{PAR(13)} = 0.5108401,$$

and the length of the truncated time-interval

$$T_0 = \texttt{PAR(11)} = 1046.178\ .$$

Since a homoclinic orbit to a saddle-node is being followed, we have also made the choices

$$\texttt{IEQUIB} = 2 \quad \texttt{NUNSTAB} = 0 \quad \texttt{NSTAB} = 1$$

in `s.mtn.1`. The two test-functions, ψ_{15} and ψ_{16}, to detect non-central saddle-node homoclinic orbits are also activated, which must be specified in three ways. Firstly, in `s.mtn.1`, `NPSI` is set to 2 and the active test functions `IPSI(I),I=1,2` are chosen as 15 and 16. This sets up the monitoring of these test functions. Secondly, in `r.mtn.1` user-defined functions (`NUZR=2`) are set up to look for zeros of the parameters corresponding to these test functions. Recall that the parameters to be zeroed are always the test functions plus 20. Finally, these parameters are included in the list of continuation parameters (`NICP,(ICP(I),I=1 NICP)`).
Among the output there is a line

```
 BR    PT  TY LAB    PAR(1)     ...     PAR(2)        PAR(35)       PAR(36)
  1    27  UZ   5  6.10437E+00 ...   6.932475E-02 -6.782898E-07  8.203437E-02
```

indicating that a zero of the test function `IPSI(1)=15` This means that at

$$D_1 = (K^1, Z^1) = (6.6104\ldots, 0.069325\ldots)$$

the homoclinic orbit to the saddle-node becomes *non-central*, namely, it returns to the equilibrium along the stable eigenvector, forming a non-smooth loop. The output is saved in `p.1`, `q.1` and `d.1`. Repeating computations in the opposite direction along the curve, `IRS=1, DS=-0.01` in `r.mtn.2`,

make second

one obtains

```
 BR    PT  TY LAB    PAR(1)      ...     PAR(2)        PAR(35)       PAR(36)
  1    34  UZ   9  5.180323E+00 ...  6.385506E-02  3.349720E-09  9.361957E-02
```

which means another non-central saddle-node homoclinic bifurcation occurs at

$$D_2 = (K^2, Z^2) = (5.1803\ldots, 0.063855\ldots).$$

Note that these data were obtained using a smaller value of `NTST` than the original computation (compare `r.mtn.1` with `r.mtn.2`). The high original value of `NTST` was only necessary for the first few steps because the original solution is specified on a uniform mesh.

18.3 Switching between Saddle-Node and Saddle Homoclinic Orbits.

Now we can switch to continuation of saddle homoclinic orbits at the located codim 2 points D_1 and D_2.

make third

starts from D_1. Note that now

```
NUNSTAB = 1   IEQUIB = 1
```

has been specified in `s.mtn.3`. Also, test functions ψ_9 and ψ_{10} have been activated in order to monitor for non-hyperbolic equilibria along the homoclinic locus. We get the following output

```
BR    PT  TY LAB    PAR(1)      ...    PAR(2)        PAR(29)       PAR(30)
 1    10       11  7.114523E+00 ...  7.081751E-02 -4.649861E-01  3.183429E-03
 1    20       12  9.176810E+00 ...  7.678731E-02 -4.684912E-01  1.609294E-02
 1    30       13  1.210834E+01 ...  8.543468E-02 -4.718871E-01  3.069638E-02
 1    40  EP   14  1.503788E+01 ...  9.428036E-02 -4.743794E-01  4.144558E-02
```

The fact that `PAR(29)` and `PAR(30)` do not change sign indicates that there are no further non-hyperbolic equilibria along this branch. Note that restarting in the opposite direction with `IRS=11, DS=-0.02`

make fourth

will detect the same codim 2 point D_1 but now as a zero of the test-function ψ_{10}

```
BR    PT  TY LAB    PAR(1)      ...    PAR(2)        PAR(29)       PAR(30)
1    10  UZ  15  6.610459E+00  ...  6.932482E-02 -4.636603E-01  1.725013E-09
```

Note that the values of `PAR(1)` and `PAR(2)` differ from that at label `4` only in the sixth significant figure.

Actually, the program runs further and eventually computes the point D_2 and the whole lower branch of P emanating from it, however, the solutions between D_1 and D_2 should be considered as spurious[1], therefore we do not save these data. The reliable way to compute the lower branch of P is to restart computation of saddle homoclinic orbits in the other direction from the point D_2

make fifth

This gives the lower branch of P approaching the BT point (see Figure 18.1)

```
 BR    PT  TY LAB    PAR(1)      ...    PAR(2)        PAR(29)       PAR(30)
  1    10       15  4.966429E+00 ... 6.298418E-02 -4.382426E-01  4.946824E-03
  1    20       16  4.925379E+00 ... 7.961214E-02 -3.399102E-01  3.288447E-02
  1    30       17  7.092267E+00 ... 1.587114E-01 -1.692842E-01  3.876291E-02
  1    40  EP   18  1.101819E+01 ... 2.809825E-01 -3.482651E-02  2.104384E-02
```

The data are appended to the stored results in `p.1`, `q.1` and `d.1`. One could now display all data using the AUTO command *@p 1* to reproduce the curve P shown in Figure 18.1.

It is worthwhile to compare the homoclinic curves computed above with a curve $T_0 = const$ along which the system has a limit cycle of constant large period $T_0 = 1046.178$, which can easily be computed using AUTO or LOCBIF. Such a curve is plotted in Figure 18.2. It obviously approximates well the saddle homoclinic loci of P, but demonstrates much bigger deviation from the saddle-node homoclinic segment D_1D_2. This happens because the period of the limit cycle grows to infinity while approaching both types of homoclinic orbit, but with *different asymptotics*: as $-\ln\|\alpha-\alpha^*\|$, in the saddle homoclinic case, and as $\|\alpha-\alpha^*\|^{-1}$ in the saddle-node case.

18.4 Three-Parameter Continuation.

Finally, we can follow the curve of non-central saddle-node homoclinic orbits in three parameters. The extra continuation parameter is D_0=`PAR(3)`. To achieve this we restart at label 4, corresponding to the codim 2 point D_1. We return to continuation of saddle-node homoclinics, `NUNSTAB=0`,`IEQUIB=2`, but append the defining equation $\psi_{15} = 0$ to the continuation problem (via `NFIXED=1, IFIXED(1)=15`). The new continuation problem is specified in `r.mtn.6` and `s.mtn.6`.

make sixth

Notice that we set `ILP=1` and choose `PAR(3)` as the first continuation parameter so that AUTO can detect limit points with respect to this parameter. We also make a user-defined function (`NUZR=1`) to detect intersections with the plane $D_0 = 0.01$. We get among other output

[1] The program actually computes the saddle-saddle heteroclinic orbit bifurcating from the non-central saddle-node homoclinic at the point D_1, see Champneys et al. (1996, Fig. 2), and continues it to the one emanating from D_2.

```
 BR    PT  TY LAB    PAR(3)        L2-NORM    ...    PAR(1)        PAR(2)
  1    22  LP  19  1.081212E-02  5.325894E+00 ...  5.673631E+00  6.608184E-02
  1    31  UZ  20  1.000000E-02  4.819681E+00 ...  5.180317E+00  6.385503E-02
```

the first line of which represents the D_0 value at which the homoclinic curve P has a tangency with the branch t_2 of fold bifurcations. Beyond this value of D_0, P consists entirely of saddle homoclinic orbits. The data at label `20` reproduce the coordinates of the point D_2. The results of this computation and a similar one starting from D_1 in the opposite direction (with `DS=-0.01`) are displayed in Figure 18.3.

18.5 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir mtn	create an empty work directory
cd mtn	change directory
@dm mtn	copy the demo files to the work directory
cp r.mtn.1 r.mtn	get the AUTO constants-file
cp s.mtn.1 s.mtn	get the HomCont constants-file
@fc mtn	use the starting data in `mtn.dat` to create `q.dat`
@h mtn dat	continue saddle-node homoclinic orbit
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.mtn.2 r.mtn	get the AUTO constants-file
cp s.mtn.2 s.mtn	get the HomCont constants-file
@h mtn 1	continue in opposite direction; restart from `q.1`
@ap 1	append output-files to `p.1, q.1, d.1`
cp r.mtn.3 r.mtn	get the AUTO constants-file
cp s.mtn.3 s.mtn	get the HomCont constants-file
@h mtn 1	switch to saddle homoclinic orbit ; restart from `q.1`
@ap 1	append output-files to `p.1, q.1, d.1`
cp r.mtn.4 r.mtn	get the AUTO constants-file
cp s.mtn.4 s.mtn	get the HomCont constants-file
@h mtn 1	continue in reverse direction; restart from `q.1`
@sv 4	save output-files as `p.4, q.4, d.4`
cp r.mtn.5 r.mtn	get the AUTO constants-file
cp s.mtn.5 s.mtn	get the HomCont constants-file
@h mtn 1	other saddle homoclinic orbit branch; restart from `q.1`
@ap 1	append output-files to `p., q.1, d.1`
cp r.mtn.6 r.mtn	get the AUTO constants-file
cp s.mtn.6 s.mtn	get the HomCont constants-file
@h mtn 1	3-parameter non-central saddle-node homoclinic.
@sv 6	save output-files as `p.6, q.6, d.6`

Table 18.1: Detailed AUTO-Commands for running demo `mtn`.

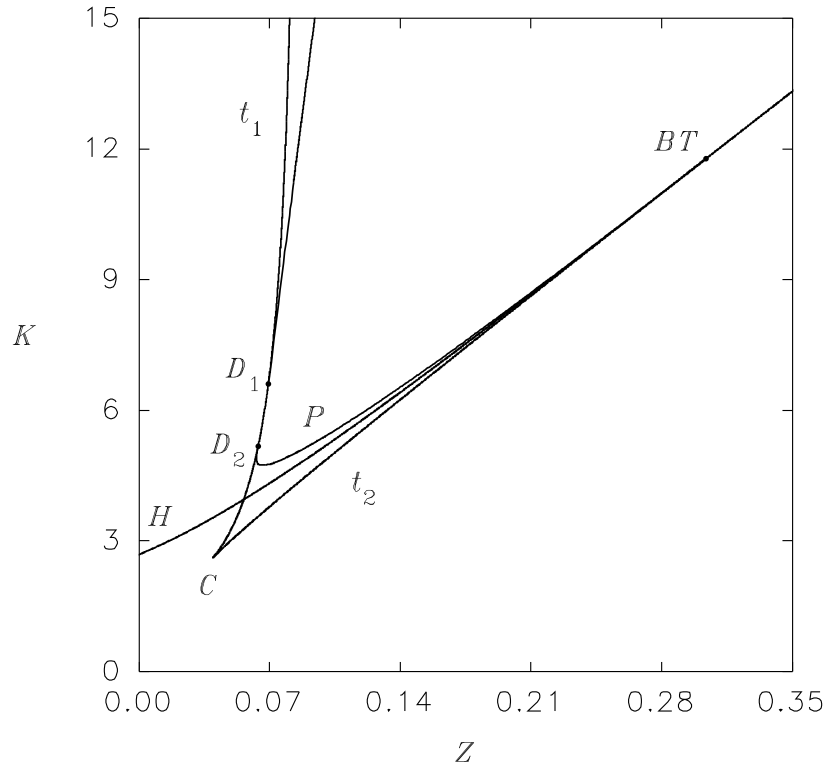

Figure 18.1: Parametric portrait of the predator-prey system

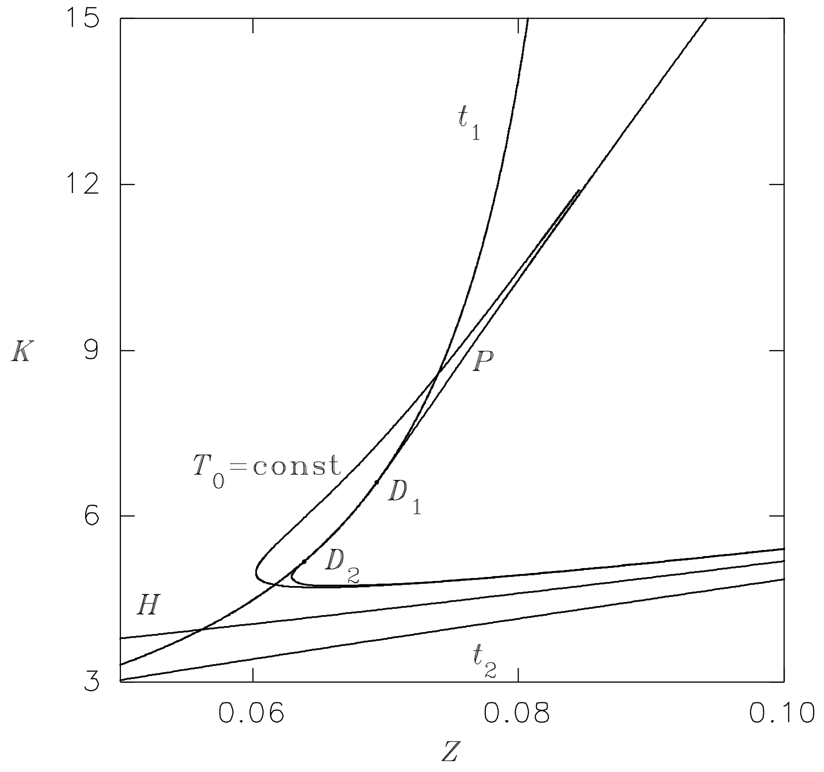

Figure 18.2: Approximation by a large-period cycle

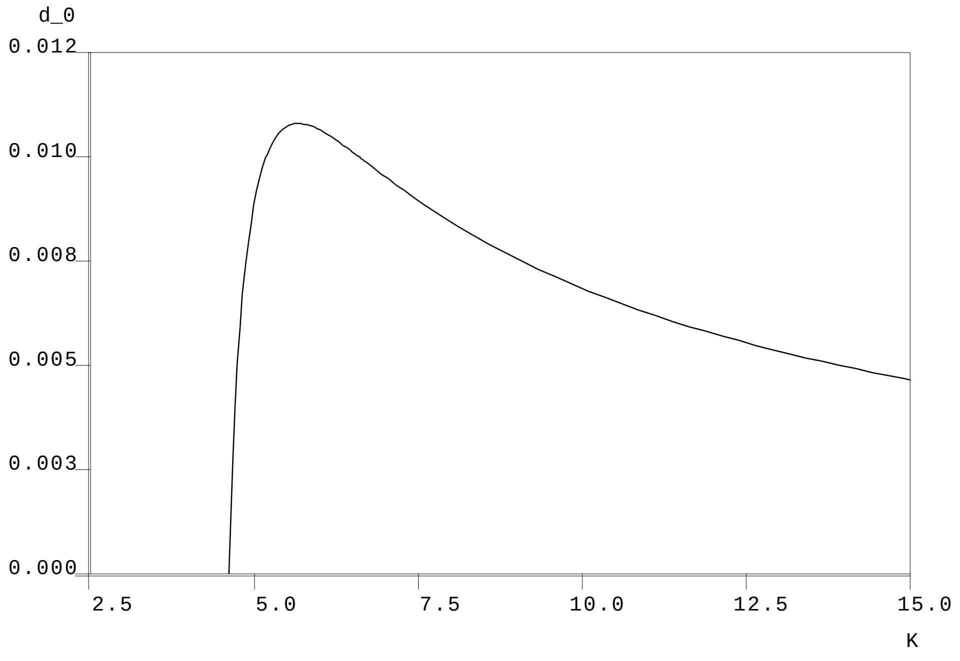

Figure 18.3: Projection onto the (K, D_0)-plane of the three-parameter curve of non-central saddle-node homoclinic orbit

Chapter 19

HomCont **Demo : kpr.**

19.1 Koper's Extended Van der Pol Model.

The equation-file `kpr.f` contains the equations

$$\begin{aligned} \dot{x} &= \epsilon_1^{-1}\,(k\,y - x^3 + 3\,x - \lambda) \\ \dot{y} &= x - 2\,y + z \\ \dot{z} &= \epsilon_2(y - z), \end{aligned} \tag{19.1}$$

with $\epsilon_1 = 0.1$ and $\epsilon_2 = 1$ (Koper 1995).
To copy across the demo `kpr` and compile we type

@dm kpr

19.2 The Primary Branch of Homoclinics.

First, we locate a homoclinic orbit using the homotopy method. The file `kpr.f` already contains approximate parameter values for a homoclinic orbit, namely λ =`PAR(1)=-1.851185`, k =`PAR(2)=-0.15`. The files `r.kpr.1` and `s.kpr.1` specify the appropriate constants for continuation in $2T$=`PAR(11)` (also referred to as `PERIOD`) and the dummy parameter ω_1=`PAR(17)` starting from a small solution in the local unstable manifold;

make first

Among the output there is the line

```
BR    PT  TY LAB    PERIOD        L2-NORM     ...    PAR(17)    ...
 1    29  UZ   2  1.900184E+01  1.693817E+00  ...  4.433433E-09 ...
```

which indicates that a zero of the artificial parameter ω_1 has been located. This means that the right-hand end point of the solution belongs to the plane that is tangent to the stable manifold at the saddle. The output is stored in files `p.1, q.1, d.1`. Upon plotting the data at label 2 (see Figure 19.1) it can be noted that although the right-hand projection boundary condition is satisfied, the solution is still quite away from the equilibrium.

The right-hand endpoint can be made to approach the equilibrium by performing a further continuation in T with the right-hand projection condition satisfied (`PAR(17)` fixed) but with λ allowed to vary.

make second

the output at label 4, stored in `kpr.2`,

```
BR   PT TY   LAB    PERIOD        L2-NORM       ...     PAR(1)       ...
1    35  UZ    4  6.000000E+01  1.672806E+00   ... -1.851185E+00 ...
```

provides a good approximation to a homoclinic solution (see Figure 19.2).

The second stage to obtain a starting solution is to add a solution to the modified adjoint variational equation. This is achieved by setting both `ITWIST` and `ISTART` to 1 (in `s.kpr.3`), which generates a trivial guess for the adjoint equations. Because the adjoint equations are linear, only a single Newton step (by continuation in a trivial parameter) is required to provide a solution. Rather than choose a parameter that might be used internally by AUTO, in `r.kpr.3` we take the continuation parameter to be `PAR(11)`, which is not quite a trivial parameter but whose affect upon the solution is mild.

make third

The output at the second point (label `6`) contains the converged homoclinic solution (variables (`U(1), U(2), U(3)`) and the adjoint (`U(4), U(5), U(6)`)). We now have a starting solution and are ready to perform two-parameter continuation.

The fourth run

make fourth

continues the homoclinic orbit in `PAR(1)` and `PAR(2)`. Note that several other parameters appear in the output. `PAR(10)` is a dummy parameter that should be zero when the adjoint is being computed correctly; `PAR(29)`, `PAR(30)`, `PAR(33)` correspond to the test functions ψ_9,ψ_{10} and ψ_{13}. That these test functions were activated is specified in three places in `r.kpr.4` and `s.kpr.4` as described in Section 16.6.

Note that at the end-point of the branch (reached when after `NMX=50` steps) `PAR(29)` is approximately zero which corresponds to a zero of ψ_9, a non-central saddle-node homoclinic orbit. We shall return to the computation of this codimension-two point later. Before reaching this point, among the output we find two zeroes of `PAR(33)` (test function ψ_{13}) which gives the accurate location of two inclination-flip bifurcations,

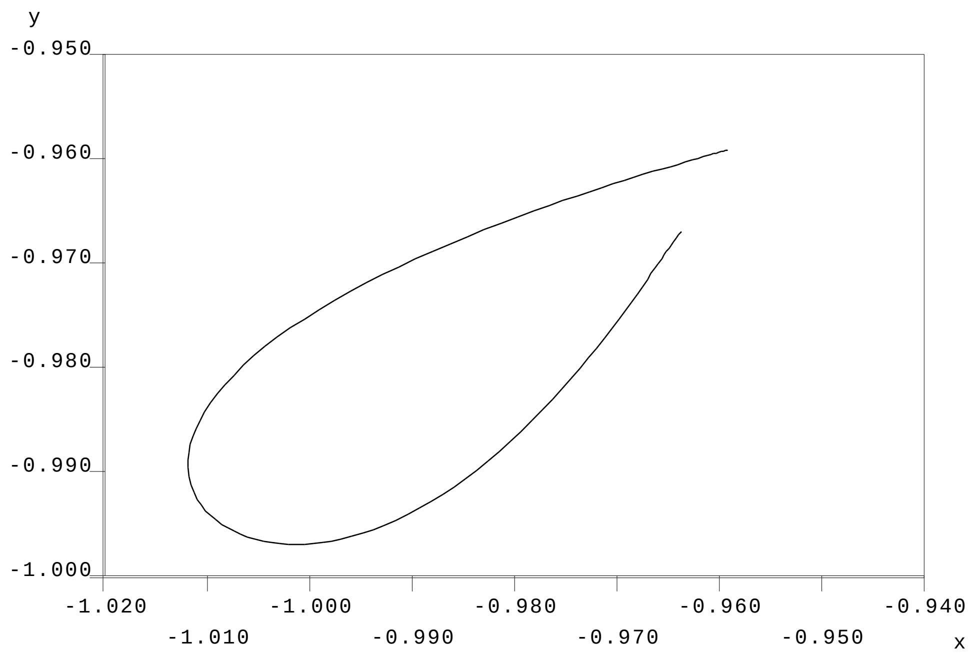

Figure 19.1: Projection on the (x, y)-plane of solutions of the boundary value problem with $2T = 19.08778$.

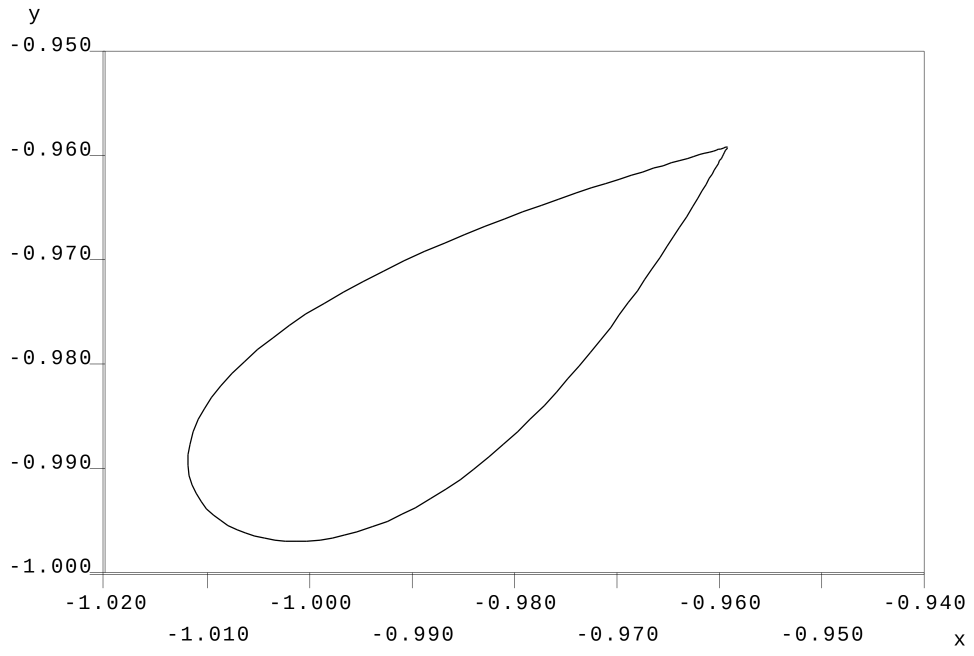

Figure 19.2: Projection on the (x, y)-plane of solutions of the boundary value problem with $2T = 60.0$.

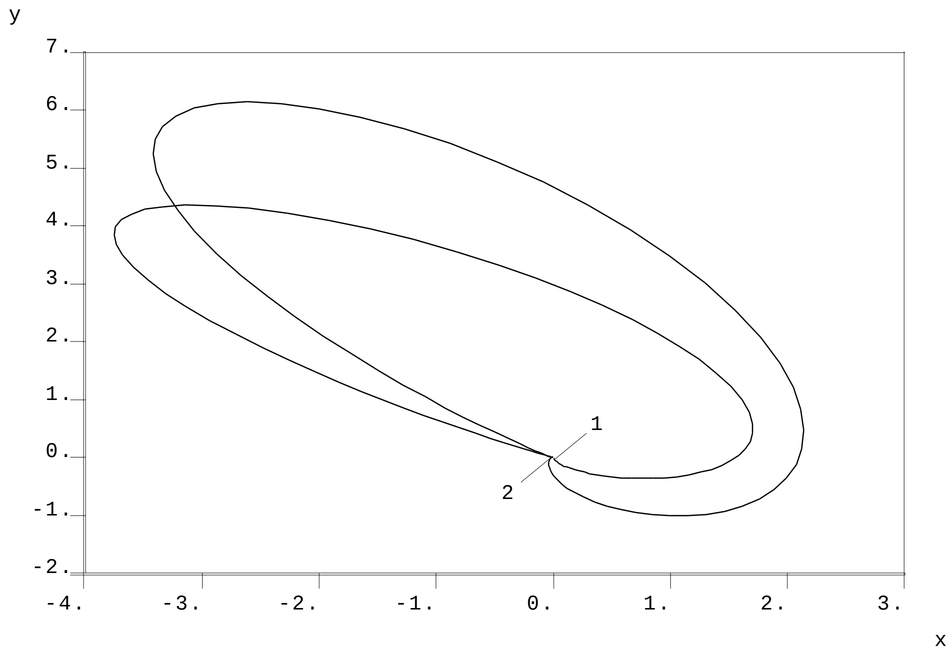

Figure 19.3: Projection on the (x, y)-plane of solutions $\phi(t)$ at `1` ($\lambda = -1.825470, k = -0.1760749$) and `2` ($\lambda = -1.686154, k = -0.3183548$).

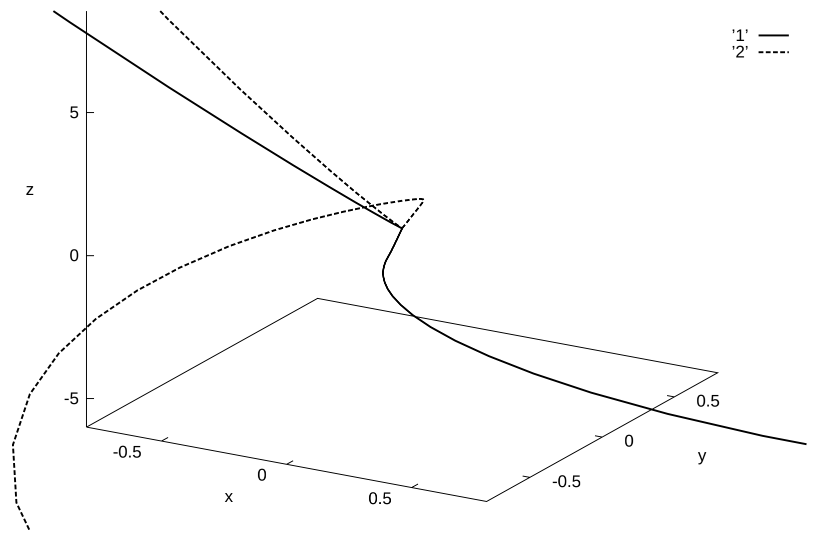

Figure 19.4: Three-dimensional blow-up of the solution curves $\phi(t)$ at labels `1` (dotted) and `2` (solid line) from Figure 3.8.

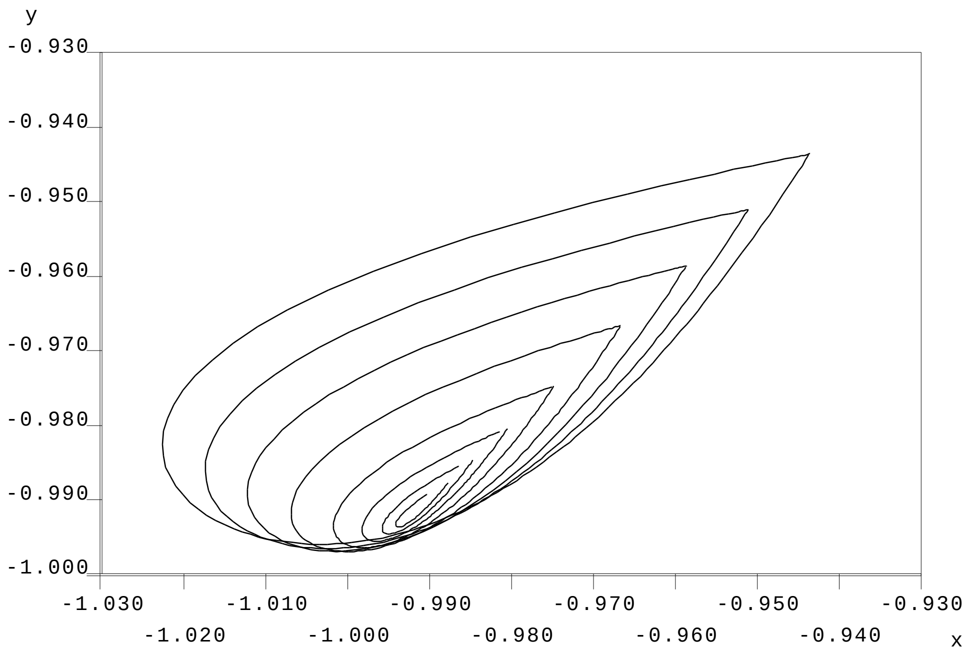

Figure 19.5: Computed homoclinic orbits approaching the BT point

```
BR  PT  TY LAB    PAR(1)      ...     PAR(2)         PAR(10)    ...    PAR(33)
 1   6  UZ  10 -1.801662E+00 ... -2.002660E-01 -7.255434E-07 ... -1.425714E-04
 1  12  UZ  11 -1.568756E+00 ... -4.395468E-01 -2.156353E-07 ...  4.514073E-07
```

That the test function really does have a regular zero at this point can be checked from the data saved in `p.3`, plotting `PAR(33)` as a function of `PAR(1)` or `PAR(2)`. Figure 19.3 presents solutions $\phi(t)$ of the modified adjoint variational equation (for details see Champneys et al. (1996)) at parameter values on the homoclinic branch before and after the first detected inclination flip. Note that these solutions were obtained by choosing a smaller step `DS` and more output (smaller `NPR`) in `r.kpr.4`. A blow-up of the region close to the origin of this figure is shown in Figure 19.4. It illustrates the flip of the solutions of the adjoint equation while moving through the bifurcation point. Note that the data in this figure were plotted after first performing an additional continuation of the solutions with respect to `PAR(11)`.

Continuing in the other direction

make fifth

we approach a Bogdanov-Takens point

```
BR    PT  TY LAB    PAR(1)      ...    PAR(10)     ...    PAR(33)
 1    50  EP  13 -1.938276E+00 ... -7.523344E+00 ...  6.310810E+01
```

Note that the numerical approximation has ceased to become reliable, since `PAR(10)` has now become large. Phase portraits of homoclinic orbits between the BT point and the first inclination flip are depicted in Figure 19.5. Note how the computed homoclinic orbits approaching the BT point have their endpoints well away from the equilibrium. To follow the homoclinic orbit to the BT point with more precision, we would need to first perform continuation in T (`PAR(11)`) to obtain a more accurate homoclinic solution.

19.3 More Accuracy and Saddle-Node Homoclinic Orbits.

Continuation in T in order to obtain an approximation of the homoclinic orbit over a longer interval is necessary for parameter values near a non-hyperbolic equilibrium (either a saddle-node or BT) where the convergence to the equilibrium is slower. First, we start from the original homoclinic orbit computed via the homotopy method, label `4`, which is well away from the non-hyperbolic equilibrium. Also, we shall no longer be interested in in inclination flips so we set `ITWIST=0` in `r.kpr.6`, and in order to compute up to `PAR(11)=1000`, we set up a user-defined function for this. Running AUTO with `PAR(11)` and `PAR(2)` as free parameters

make sixth

we obtain among the output

```
 BR    PT  TY LAB     PERIOD       L2-NORM    ...     PAR(2)
  1    35  UZ   6  1.000000E+03  1.661910E+00 ... -1.500000E-01
```

We can now repeat the computation of the branch of saddle homoclinic orbits in `PAR(1)` and `PAR(2)` from this point with the test functions ψ_9 and ψ_{10} for non-central saddle-node homoclinic orbits activated

make seventh

The saddle-node point is now detected at

```
 BR    PT  TY LAB    PAR(1)      ...     PAR(2)         PAR(29)        PAR(30)
  1    30  UZ   8  1.765003E-01 ... -2.405345E+00  2.743361E-06  2.309317E+01
```

which is stored in `q.7`. That `PAR(29)` (ψ_9) is zeroed shows that this is a non-central saddle-node connecting the centre manifold to the strong stable manifold. Note that all output beyond this point, although a well-posed solution to the boundary-value problem, is spurious in that it no longer represents a homoclinic orbit to a saddle equilibrium (see Champneys et al. (1996)). If we had chosen to, we could continue in the other direction in order to approach the BT point more accurately by reversing the sign of `DS` in `r.kpr.7`.

The files `r.kpr.9` and `s.kpr.9` contain the constants necessary for switching to continuation of the central saddle-node homoclinic curve in two parameters starting from the non-central saddle-node homoclinic orbit stored as label `8` in `q.7`.

make eighth

In this run we have activated the test functions for saddle to saddle-node transition points along curves of saddle homoclinic orbits (ψ_{15} and ψ_{16}). Among the output we find

```
 BR    PT  TY LAB    PAR(1)      ...     PAR(2)         PAR(35)        PAR(36)
  1    38  UZ  13  1.765274E-01 ... -2.405284E+00  9.705426E-03 -5.464784E-07
```

which corresponds to the branch of homoclinic orbits leaving the locus of saddle-nodes in a second non-central saddle-node homoclinic bifurcation (a zero of ψ_{16}).

Note that the parameter values do not vary much between the two codimension-two non-central saddle-node points (labels `8` and `13`). However, Figure 19.6 shows clearly that between the two codimension-two points the homoclinic orbit rotates between the two components of the 1D stable manifold, i.e. between the two boundaries of the center-stable manifold of the saddle node. The overall effect of this process is the transformation of a nearby "small" saddle homoclinic orbit to a "big" saddle homoclinic orbit (i.e. with two extra turning points in phase space).

Finally, we can switch to continuation of the big saddle homoclinic orbit from the new codim 2 point at label `13`.

make ninth

Note that AUTO takes a large number of steps near the line `PAR(1)=0`, while `PAR(2)` approaches $-2.189\ldots$ (which is why we chose such a large value `NMX=500` in `r.kpr.9`). This particular computation ends at

```
 BR    PT  TY LAB    PAR(1)        L2-NORM    ...     PAR(2)
  1   500  EP  24 -1.218988E-05  2.181205E-01 ... -2.189666E+00
```

By plotting phase portraits of orbits approaching this end point (see Figure 19.7) we see a "canard-like" like transformation of the big homoclinic orbit to a pair of homoclinic orbits in a figure-of-eight configuration. That we get a figure-of-eight is not a surprise because `PAR(1)=0` corresponds to a symmetry in the differential equations (Koper 1994); note also that the equilibrium, stored as (`PAR(12), PAR(13), PAR(14)`) in `d.9`, approaches the origin as we approach the figure-of-eight homoclinic.

19.4 Three-Parameter Continuation.

We now consider curves in three parameters of each of the codimension-two points encountered in this model, by freeing the parameter $\epsilon =$ `PAR(3)`. First we continue the first inclination flip stored at label `7` in `q.3`

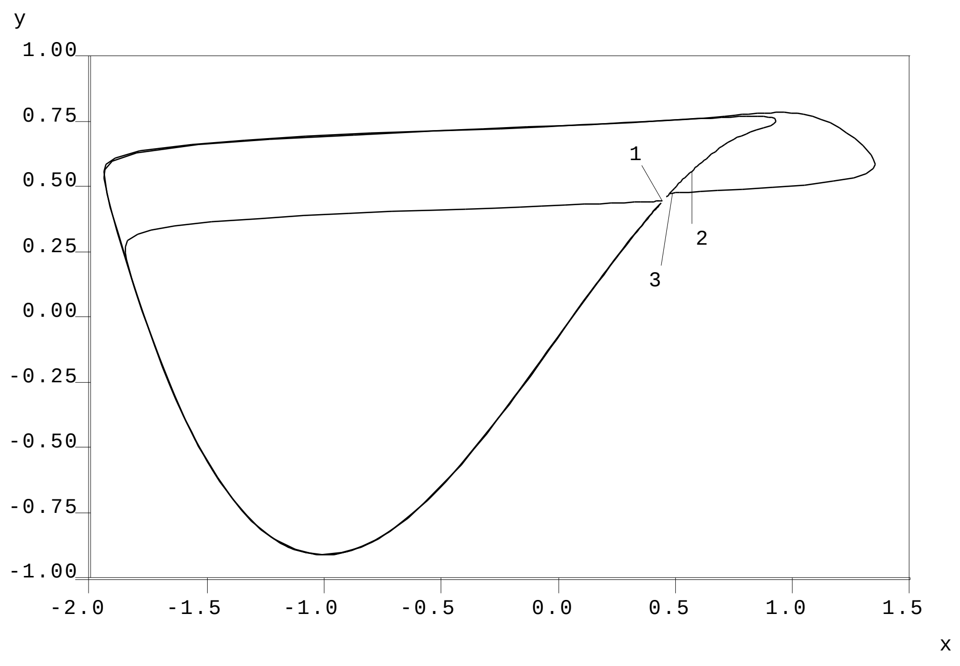

Figure 19.6: Two non-central saddle-node homoclinic orbits, 1 and 3; and, 2, a central saddle-node homoclinic orbit between these two points

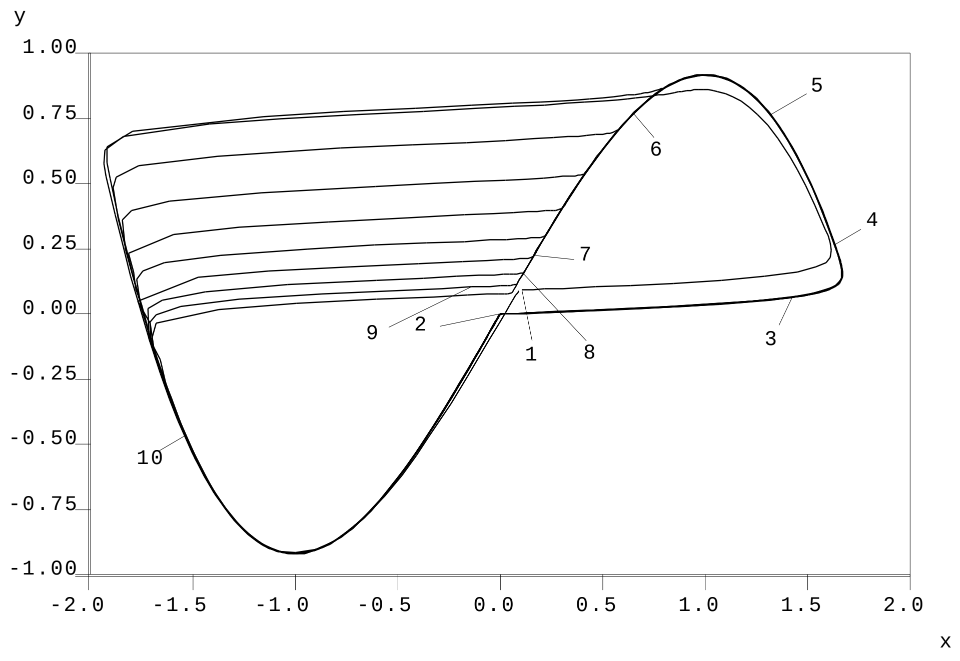

Figure 19.7: The big homoclinic orbit approaching a figure-of-eight

make tenth

Note that `ITWIST=1` in `s.kpr.10`, so that the adjoint is also continued, and there is one fixed condition `IFIXED(1)=13` so that test function ψ_{13} has been frozen. Among the output there is a codimension-three point (zero of ψ_9) where the neutrally twisted homoclinic orbit collides with the saddle-node curve

```
 BR  PT  TY LAB    PAR(1)     ...   PAR(2)        PAR(3)        PAR(29)     ...
  1  28  UZ  14  1.282702E-01 ... -2.519325E+00 5.744770E-01 -4.347113E-09 ...
```

The other detected inclination flip (at label `8` in `q.3`) is continued similarly

make eleventh

giving among its output another codim 3 saddle-node inclination-flip point

```
 BR  PT  TY LAB    PAR(1)     ...   PAR(2)        PAR(3)        PAR(29)     ...
  1  27  UZ  14  1.535420E-01 ... -2.458100E+00 1.171705E+00 -1.933188E-07 ...
```

Output beyond both of these codim 3 points is spurious and both computations end in an `MX` point (no convergence).

To continue the non-central saddle-node homoclinic orbits it is necessary to work on the data without the solution $\phi(t)$. We restart from the data saved at `LAB=8` and `LAB=13` in `q.7` and `q.8` respectively. We could continue these codim 2 points in two ways, either by appending the defining condition $\psi_{16} = 0$ to the continuation of saddle-node homoclinic orbits (with `IEQUIB=2`, etc.), or by appending $\psi_9 = 0$ to the continuation of a saddle homoclinic orbit (with `IEQUIB=1`. The first approach is used in the example `mtn`, for contrast we shall adopt the second approach here.

make twelfth
make thirteenth

The projection onto the (ϵ, k)-plane of all four of these codimension-two curves is given in Figure 19.8. The intersection of the inclination-flip lines with one of the non-central saddle-node homoclinic lines is apparent. Note that the two non-central saddle-node homoclinic orbit curves are almost overlaid, but that as in Figure 19.6 the orbits look quite distinct in phase space.

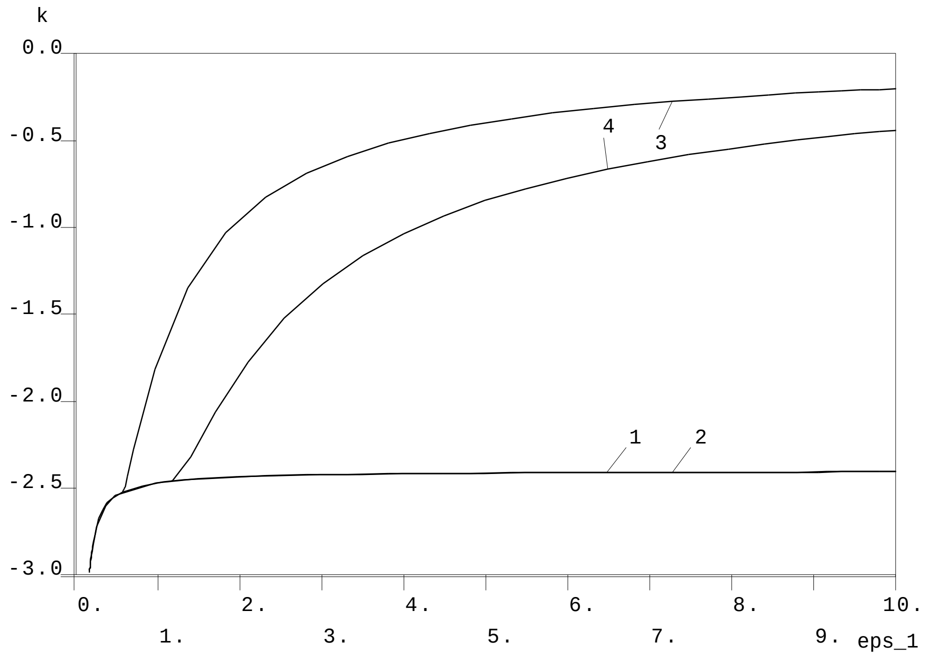

Figure 19.8: Projection onto the `(PAR(3),PAR(2))`-plane of the non-central saddle-node homoclinic orbit curves (labeled `1` and `2`) and the inclination-flip curves (labeled `3` and `4`)
.

19.5 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir kpr	create an empty work directory
cd kpr	change directory
@dm kpr	copy the demo files to the work directory
cp r.kpr.1 r.kpr	get the AUTO constants-file
cp s.kpr.1 s.kpr	get the HomCont constants-file
@h kpr	continuation in the time-length parameter `PAR(11)`
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.kpr.2 r.kpr	get the AUTO constants-file
cp s.kpr.2 s.kpr	get the HomCont constants-file
@h kpr 1	locate the homoclinic orbit; restart from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`
cp r.kpr.3 r.kpr	get the AUTO constants-file
cp s.kpr.3 s.kpr	get the HomCont constants-file
@h kpr 2	generate adjoint variables ; restart from `q.2`
@sv 3	save output-files as `p.3, q.3, d.3`
cp r.kpr.4 r.kpr	get the AUTO constants-file
cp s.kpr.4 s.kpr	get the HomCont constants-file
@h kpr 3	continue the homoclinic orbit; restart from `q.3`
@ap 3	append output-files to `p.3, q.3, d.3`
cp r.kpr.5 r.kpr	get the AUTO constants-file
cp s.kpr.5 s.kpr	get the HomCont constants-file
@h kpr 3	continue in reverse direction; restart from `q.3`
@ap 3	append output-files to `p.3, q.3, d.3`
cp r.kpr.6 r.kpr	get the AUTO constants-file
cp s.kpr.6 s.kpr	get the HomCont constants-file
@h kpr 2	increase the period; restart from `q.2`
@sv 6	save output-files as `p.6, q.6, d.6`

Table 19.1: Detailed AUTO-Commands for running demo `kpr`.

COMMAND	ACTION
cp r.kpr.7 r.kpr	get the AUTO constants-file
cp s.kpr.7 s.kpr	get the HomCont constants-file
@h kpr 6	recompute the branch of homoclinic orbits; restart from `q.6`
@sv 7	save output-files as `p.7, q.7, d.7`
cp r.kpr.8 r.kpr	get the AUTO constants-file
cp s.kpr.8 s.kpr	get the HomCont constants-file
@h kpr 7	continue central saddle-node homoclinics; restart from `q.7`
@sv 8	save output-files as `p.8, q.8, d.8`
cp r.kpr.9 r.kpr	get the AUTO constants-file
cp s.kpr.9 s.kpr	get the HomCont constants-file
@h kpr 8	continue homoclinics from codim-2 point; restart from `q.8`
@sv 9	save output-files as `p.9, q.9, d.9`
cp r.kpr.10 r.kpr	get the AUTO constants-file
cp s.kpr.10 s.kpr	get the HomCont constants-file
@h kpr 3	3-parameter curve of inclination-flips; restart from `q.3`
@sv 10	save output-files as `p.10, q.10, d.10`
cp r.kpr.11 r.kpr	get the AUTO constants-file
cp s.kpr.11 s.kpr	get the HomCont constants-file
@h kpr 3	another curve of inclination-flips; restart from `q.3`
@sv 11	save output-files as `p.11, q.11, d.11`
cp r.kpr.12 r.kpr	get the AUTO constants-file
cp s.kpr.12 s.kpr	get the HomCont constants-file
@h kpr 7	continue non-central saddle-node homoclinics; restart from `q.7`
@sv 12	save output-files as `p.12, q.12, d.12`
cp r.kpr.13 r.kpr	get the AUTO constants-file
cp s.kpr.13 s.kpr	get the HomCont constants-file
@h kpr 8	continue non-central saddle-node homoclinics; restart from `q.8`
@ap 12	append output-files to `p.12, q.12, d.12`

Table 19.2: Detailed AUTO-Commands for running demo `kpr`.

Chapter 20

HomCont **Demo : cir.**

20.1 Electronic Circuit of Freire *et al.*

Consider the following model of a three-variable electronic circuit (Freire, Rodríguez-Luis, Gamero & Ponce 1993)

$$\left\{ \begin{array}{rcl} \dot{x} & = & \left[-(\beta+\nu)x + \beta y - a_3 x^3 + b_3 (y-x)^3\right]/r, \\ \dot{y} & = & \beta x - (\beta+\gamma) y - z - b_3 (y-x)^3, \\ \dot{z} & = & y. \end{array} \right. \tag{20.1}$$

These autonomous equations are also considered in the AUTO demo `tor`.

First, we copy the demo into a new directory and compile

@dm cir

The system is contained in the equation-file `cir.f` and the initial run-time constants are stored in `r.cir.1` and `s.cir.1`. We begin by starting from the data from `cir.dat` for a saddle-focus homoclinic orbit at $\nu = -0.721309$, $\beta = 0.6$, $\gamma = 0$, $r = 0.6$, $A_3 = 0.328578$ and $B_3 = 0.933578$, which was obtained by shooting over the time interval $2T =$`PAR(11)`$= 36.13$. We wish to follow the branch in the (β, ν)-plane, but first we perform continuation in (T, ν) to obtain a better approximation to a homoclinic orbit.

make first

yields the output

```
BR  PT  TY LAB     PERIOD       L2-NORM    ...   PAR(1)
 1  21  UZ   2  1.000000E+02  1.286637E-01 ... -7.213093E-01
 1  42  UZ   3  2.000000E+02  9.097899E-02 ... -7.213093E-01
 1  50  EP   4  2.400000E+02  8.305208E-02 ... -7.213093E-01
```

Note that $\nu =$`PAR(1)` remains constant during the continuation as the parameter values do not change, only the the length of the interval over which the approximate homoclinic solution is computed. Note from the eigenvalues, stored in `d.1` that this is a homoclinic orbit to a saddle-focus with a one-dimensional unstable manifold.

We now restart at `LAB=3`, corresponding to a time interval $2T = 200$, and change the principal continuation parameters to be (ν, β). The new constants defining the continuation are given in `r.cir.2` and `s.cir.2`. We also activate the test functions pertinent to codimension-two singularities which may be encountered along a branch of saddle-focus homoclinic orbits, viz. ψ_2, ψ_4, ψ_5, ψ_9 and ψ_{10}. This must be specified in three ways: by choosing `NPSI=5` and appropriate `IPSI(I)` in `s.cir.2`, by adding the corresponding parameter labels to the list of continuation parameters `ICP(I)` in `r.cir.2` (recall that these parameter indices are 20 more than the corresponding ψ indices), and finally adding USZR functions defining zeros of these parameters in `r.cir.2`. Running

make second

results in

```
BR  PT  TY LAB    PAR(1)      ...    PAR(2)      ...    PAR(25)       PAR(29)
1   17  UZ   5 -7.256925E-01 ...  4.535645E-01 ... -1.765251E-05 -2.888436E-01
1   75  UZ   6 -1.014704E+00 ...  9.998966E-03 ...  1.664509E+00 -5.035997E-03
1   78  UZ   7 -1.026445E+00 ... -2.330391E-05 ...  1.710804E+00  1.165176E-05
1   81  UZ   8 -1.038012E+00 ... -1.000144E-02 ...  1.756690E+00  4.964621E-03
1  100  EP   9 -1.164160E+00 ... -1.087732E-01 ...  2.230329E+00  5.042736E-02
```

with results saved in `p.2, q.2, d.2`. Upon inspection of the output, note that label 5, where `PAR(25)`≈ 0, corresponds to a neutrally-divergent saddle-focus, $\psi_5 = 0$. Label 7, where `PAR(29)`≈ 0 corresponds to a local bifurcation, $\psi_9 = 0$, which we note from the eigenvalues stored in `d.2` corresponds to a *Shil'nikov-Hopf* bifurcation. Note that `PAR(2)` is also approximately zero at label 7, which accords with the analytical observation that the origin of (20.1) undergoes a Hopf bifurcation when $\beta = 0$. Labels 6 and 8 are the user-defined output points, the solutions at which are plotted in Fig. 20.1. Note that solutions beyond label 7 (e.g., the plotted solution at label 8) do not correspond to homoclinic orbits, but to *point-to-cycle* heteroclinic orbits (c.f. Section 2.2.1 of Champneys et al. (1996)).

We now continue in the other direction along the branch. It turns out that starting from the initial point in the other direction results in missing a codim 2 point which is close to the starting point. Instead we start from the first saved point from the previous computation (label 5 in `q.2`):

make third

The output

```
 BR  PT  TY LAB    PAR(1)      ...    PAR(2)        PAR(22)       PAR(24)
  1   9  UZ  10 -7.204001E-01 ...  5.912315E-01 -1.725669E+00 -3.295862E-05
  1  18  UZ  11 -7.590583E-01 ...  7.428734E-01  3.432139E-05 -2.822988E-01
  1  26  UZ  12 -7.746686E-01 ...  7.746147E-01  5.833163E-01  1.637611E-07
  1  28  EP  13 -7.746628E-01 ...  7.746453E-01  5.908902E-01  1.426214E-04
```

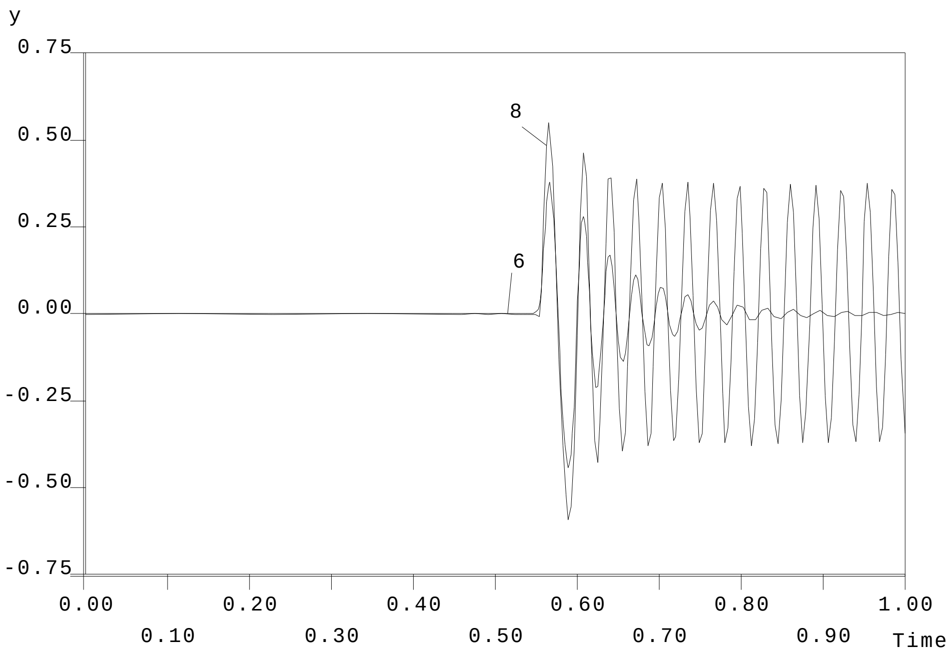

Figure 20.1: Solutions of the boundary value problem at labels 6 and 8, either side of the Shil'nikov-Hopf bifurcation

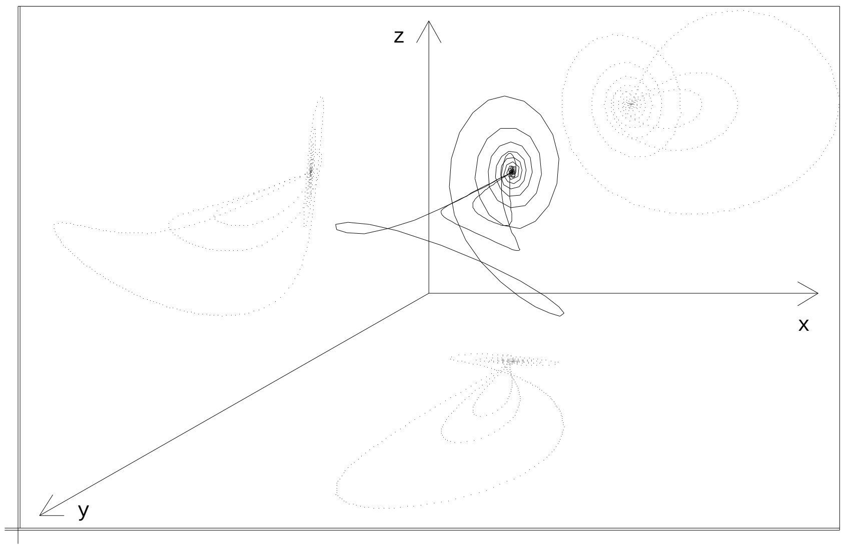

Figure 20.2: Phase portraits of three homoclinic orbits on the branch, showing the saddle-focus to saddle transition

contains a neutral saddle-focus (a *Belyakov* transition) at `LAB=10` ($\psi_4 = 0$), a double real leading eigenvalue (saddle-focus to saddle transition) at `LAB =11` ($\psi_2 = 0$) and a neutral saddle at `LAB=12` ($\psi_4 = 0$). Data at several points on the complete branch are plotted in Fig. 20.2. If we had continued further (by increasing `NMX`), the computation would end at a no convergence error `TY=MX` owing to the homoclinic branch approaching a Bogdanov-Takens singularity at small amplitude. To compute further towards the BT point we would first need to continue to a higher value of `PAR(11)`.

20.2 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir cir	create an empty work directory
cd cir	change directory
@dm cir	copy the demo files to the work directory
cp r.cir.1 r.cir	get the AUTO constants-file
cp s.cir.1 s.cir	get the HomCont constants-file
@fc cir	use the starting data in `cir.dat` to create `q.dat`
@h cir dat	increase the truncation interval; restart from `q.dat`
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.cir.2 r.cir	get the AUTO constants-file
cp s.cir.2 s.cir	get the HomCont constants-file
@h cir 1	continue saddle-focus homoclinic orbit; restart from `q.1`
@sv 2	save output-files as `p.2, q.2, d.2`
cp r.cir.3 r.cir	get the AUTO constants-file
cp s.cir.3 s.cir	get the HomCont constants-file
@h cir 2	generate adjoint variables ; restart from `q.2`
@ap 2	append output-files as `p.2, q.2, d.2`

Table 20.1: Detailed AUTO-Commands for running demo `cir`.

Chapter 21

HomCont **Demo : she.**

21.1 A Heteroclinic Example.

The following system of five equations Rucklidge & Mathews (1995)

$$\begin{array}{rcl}\dot{x} &=& \mu\,x + x\,y - z\,u, \\ \dot{y} &=& -y - x^2, \\ \dot{z} &=& (4\sigma\,x\,u + 4\sigma\,\mu\,z - 9\sigma\,z + 4x\,u + 4\mu\,z)/4(1+\sigma) \\ \dot{u} &=& -\sigma u/4 - \sigma Q v/4\pi^2 + 3(1+\sigma)xz/4\sigma \\ \dot{v} &=& \zeta u/4 - \zeta v/4\end{array} \qquad (21.1)$$

has been used to describe shearing instabilities in fluid convection. The equations possess a rich structure of local and global bifurcations. Here we shall reproduce a single curve in the (σ, μ)-plane of codimension-one heteroclinic orbits connecting a non-trivial equilibrium to the origin for $Q = 0$ and $\zeta = 4$. The defining problem is contained in equation-file `she.f`[1], and starting data for the orbit at $(\sigma, \mu) = (0.5, 0.163875)$ are stored in `she.dat`, with a truncation interval of `PAR(11)=85.07`.

We begin by computing towards $\mu = 0$ with the option `IEQUIB=-2` which means that both equilibria are solved for as part of the continuation process.

@dm she
make first

This yields the output

```
BR    PT  TY LAB    PAR(3)         L2-NORM     ...     PAR(1)
 1     5        2  4.528332E-01  3.726787E-01 ...  1.364973E-01
 1    10        3  3.943370E-01  3.303798E-01 ...  1.044119E-01
```

[1]The last parameter used to store the equilibria (`PAR(21)`) is overlaped here with the first test-function. In this example, it is harmless since the test functions are irrelevant for heteroclinic continuation.

```
  1    15       4  3.358942E-01  2.873213E-01 ...  7.515570E-02
  1    20       5  2.772726E-01  2.433403E-01 ...  4.952636E-02
  1    25       6  2.181955E-01  1.981358E-01 ...  2.845849E-02
  1    30  EP   7  1.581633E-01  1.512340E-01 ...  1.292975E-02
```

Alternatively, for this problem there exists an analytic expression for the two equilibria. This is specified in the subroutine `PVLS` of `she.f`. Re-running with `IEQUIB=-1`

make second

we obtain the output

```
 BR    PT  TY LAB    PAR(3)        L2-NORM    ...    PAR(1)
  1     5       2  4.432015E-01  3.657716E-01 ...  1.310559E-01
  1    10       3  3.723085E-01  3.142439E-01 ...  9.300982E-02
  1    15       4  3.008842E-01  2.611556E-01 ...  5.933966E-02
  1    20       5  2.286652E-01  2.062194E-01 ...  3.179939E-02
  1    25       6  1.555409E-01  1.491652E-01 ...  1.239897E-02
  1    30  EP   7  8.107462E-02  9.143108E-02 ...  2.386616E-03
```

This output is similar to that above, but note that it is obtained slightly more efficiently because the extra parameters `PAR(12-21)` representing the coordinates of the equilibria are no longer part of the continuation problem. Also note that AUTO has chosen to take slightly larger steps along the branch. Finally, we can continue in the opposite direction along the branch from the original starting point (again with `IEQUIB=-1`).

make third

```
 BR    PT  TY LAB    PAR(3)        L2-NORM    ...    PAR(1)
  1     5       8  4.997590E-01  4.060153E-01 ...  1.637322E-01
  1    10       9  5.705299E-01  4.551872E-01 ...  2.065264E-01
  1    15      10  6.416439E-01  5.031844E-01 ...  2.507829E-01
  1    20      11  7.133301E-01  5.500668E-01 ...  2.959336E-01
  1    25      12  7.857688E-01  5.958712E-01 ...  3.415492E-01
  1    30      13  8.590970E-01  6.406182E-01 ...  3.872997E-01
  1    35  EP  14  9.334159E-01  6.843173E-01 ...  4.329270E-01
```

The results of both computations are presented in Figure 21.1, from which we see that the orbit shrinks to zero as `PAR(1)`$=\mu \to 0$.

21.2 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir she	create an empty work directory
cd she	change directory
@dm she	copy the demo files to the work directory
cp r.she.1 r.she	get the AUTO constants-file
cp s.she.1 s.she	get the HomCont constants-file
@fc she	use the starting data in `she.dat` to create `q.dat`
@h she dat	continue heteroclinic orbit; restart from `q.dat`
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.she.2 r.she	get the AUTO constants-file
cp s.she.2 s.she	get the HomCont constants-file
@h she dat	repeat with IEQUIB=-1
@sv 2	save output-files as `p.2, q.2, d.2`
cp r.she.3 r.she	get the AUTO constants-file
cp s.she.3 s.she	get the HomCont constants-file
@h she 2	continue in reverse direction ; restart from `q.2`
@ap 2	append output-files to `p.2, q.2, d.2`

Table 21.1: Detailed AUTO-Commands for running demo `she`.

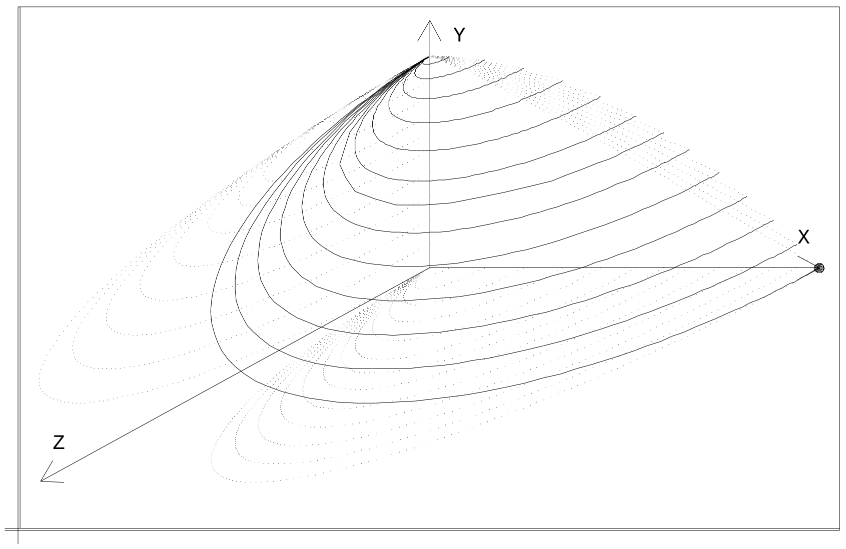

Figure 21.1: Projections into (x, y, z)-space of the family of heteroclinic orbits.

Chapter 22

HomCont **Demo : rev.**

22.1 A Reversible System.

The fourth-order differential equation

$$u'''' + Pu'' + u - u^3 = 0$$

arises in a number of contexts, including as the travelling-wave equation for a nonlinear-Schrödinger equation with fourth-order dissipation (Buryak & Akhmediev 1995) and as a model of a strut on a symmetric nonlinear elastic foundation (Hunt, Bolt & Thompson 1989). It may be expressed as a system

$$\begin{cases} \dot{u_1} &= u_2 \\ \dot{u_2} &= u_3 \\ \dot{u_3} &= u_4 \\ \dot{u_4} &= -Pu_3 - u_1 + u_1^3 \end{cases} \tag{22.1}$$

Note that (22.1) is invariant under two separate reversibilities

$$R_1 : (u_1, u_2, u_3, u_4, t) \mapsto (u_1, -u_2, u_3, -u_4, -t) \tag{22.2}$$

and

$$R_2 : (u_1, u_2, u_3, u_4, t) \mapsto (-u_1, u_2, -u_3, u_4, -t) \tag{22.3}$$

First, we copy the demo into a new directory

@dm rev

For this example, we shall make two separate starts from data stored in equation and data files `rev.f.1, rev.dat.1` and `rev.f.3, rev.dat.3` respectively. The first of these contains initial data for a solution that is reversible under R_1 and the second for data that is reversible under R_2.

22.2 An R_1-Reversible Homoclinic Solution.

The first run

make first

starts by copying the files `rev.f.1` and `rev.dat.1` to `rev.f` and `rev.dat`. The orbit contained in the data file is a "primary" homoclinic solution for $P = 1.6$, with truncation (half-)interval `PAR(11) = 39.0448429`. which is reversible under R_1. Note that this reversibility is specified in `s.rev.1` via `NREV=1, (IREV(I), I=1,NDIM) = 0 1 0 1`. Note also, from `r.rev.1` that we only have one free parameter `PAR(1)` because symmetric homoclinic orbits in reversible systems are generic rather than of codimension one. The first run results in the output

```
 BR    PT  TY LAB    PAR(1)        L2-NORM       MAX U(1)    ...
  1     7  UZ   2  1.700002E+00  2.633353E-01  4.179794E-01
  1    12  UZ   3  1.800000E+00  2.682659E-01  4.806063E-01
  1    15  UZ   4  1.900006E+00  2.493415E-01  4.429364E-01
  1    20  EP   5  1.996247E+00  1.111306E-01  1.007111E-01
```

which is consistent with the theoretical result that the solution tends uniformly to zero as $P \to 0$. Note, by plotting the data saved in `q.1` that only "half" of the homoclinic orbit is computed up to its point of symmetry. See Figure 22.1.

The second run continues in the other direction of `PAR(1)`, with the test function ψ_2 activated for the detection of saddle to saddle-focus transition points

make second

The output

```
BR  PT  TY LAB    PAR(1)        L2-NORM       MAX U(1)    ...    PAR(22)
 1  11  UZ   6  1.000005E+00  2.555446E-01  1.767149E-01 ... -3.000005E+00
 1  22  UZ   7 -1.198325E-07  2.625491E-01  4.697314E-02 ... -2.000000E+00
 1  33  UZ   8 -1.000000E+00  2.741483E-01  4.316007E-03 ... -1.000000E+00
 1  44  UZ   9 -2.000000E+00  2.873838E-01  1.245735E-11 ...  2.318248E-08
 1  55  EP  10 -3.099341E+00  3.020172E-01 -2.749454E-11 ...  1.099341E+00
```

shows a saddle to saddle-focus transition (indicated by a zero of `PAR(22)`) at `PAR(1)=-2`. Beyond that label the first component of the solution is negative and (up to the point of symmetry) monotone decreasing. See Figure 22.2.

22.3 An R_2-Reversible Homoclinic Solution.

make third

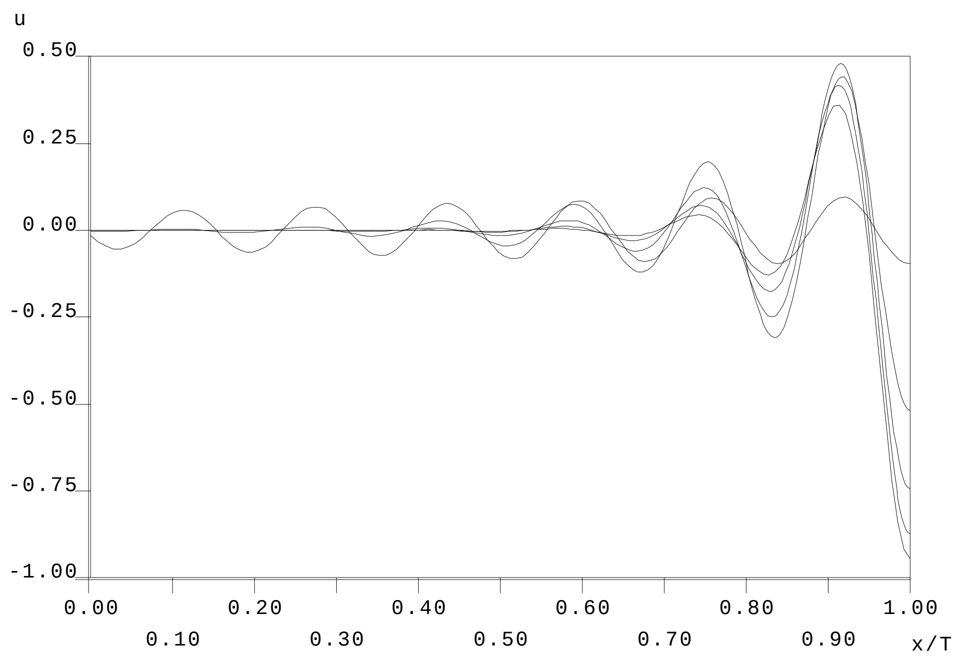

Figure 22.1: R_1-Reversible homoclinic solutions on the half-interval $x/T \in [0, 1]$ where $T = 39.0448429$ for P approaching 2 (solutions with labels `1-5` respectively have decreasing amplitude)

Copies the files `rev.f.3` and `rev.dat.3` to `rev.f` and `rev.dat`, and runs them with the constants stored in `r.rev.3` and `s.rev.3`. The orbit contained in the data file is a "multi-pulse" homoclinic solution for $P = 1.6$, with truncation (half-)interval `PAR(11) = 47.4464189`. which is reversible under R_2. This reversibility is specified in `s.rev.1` via `NREV=1`, `(IREV(I), I=1,NDIM) = 1 0 1 0`. The output

```
 BR    PT  TY LAB    PAR(1)        L2-NORM       MAX U(1)    ...
  1    15  UZ   2  1.700000E+00  3.836401E-01  4.890015E-01
  1    16  LP   3  1.711574E+00  3.922135E-01  5.442385E-01
  1    19  UZ   4  1.600000E+00  4.329404E-01  7.769491E-01
  1    31  UZ   5  1.000000E+00  4.808488E-01  1.083298E+00
  1    86  UZ   6 -9.664802E-10  5.158463E-01  1.258650E+00
```

contains the label of a limit point (`ILP` was set to `1` in `r.rev.3`, which corresponds to a "coalescence" of two reversible homoclinic orbits. The two solutions on either side of this limit point are displayed in Figure 22.3. The computation ends in a no-convergence point. The solution here is depicted in Figure 22.4. The lack of convergence is due to the large peak and trough of the solution rapidly moving to the left as $P \to -2$ (cf. Champneys & Spence (1993)).

Continuing from the initial solution in the other parameter direction

make fourth

we obtain the output

```
 BR    PT  TY LAB    PAR(1)        L2-NORM       MAX U(1)    ...
  1     7  UZ   8  1.600000E+00  3.701709E-01  3.836833E-01
  1    33  UZ   9  9.999980E-01  3.614405E-01  1.775035E-01
  1    93  UZ  10 -7.819855E-06  3.713007E-01  4.698309E-02
```

which again ends at a no convergence error for similar reasons.

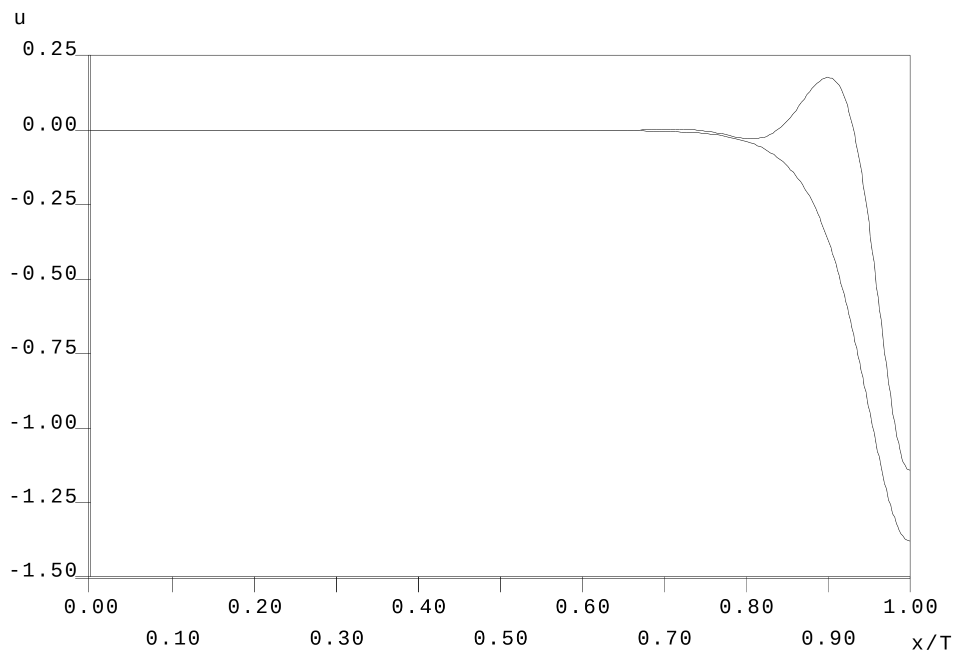

Figure 22.2: R_1-reversible homoclinic orbits with oscillatory decay as $x \to -\infty$ (corresponding to label `6`) and monotone decay (at label `10`)

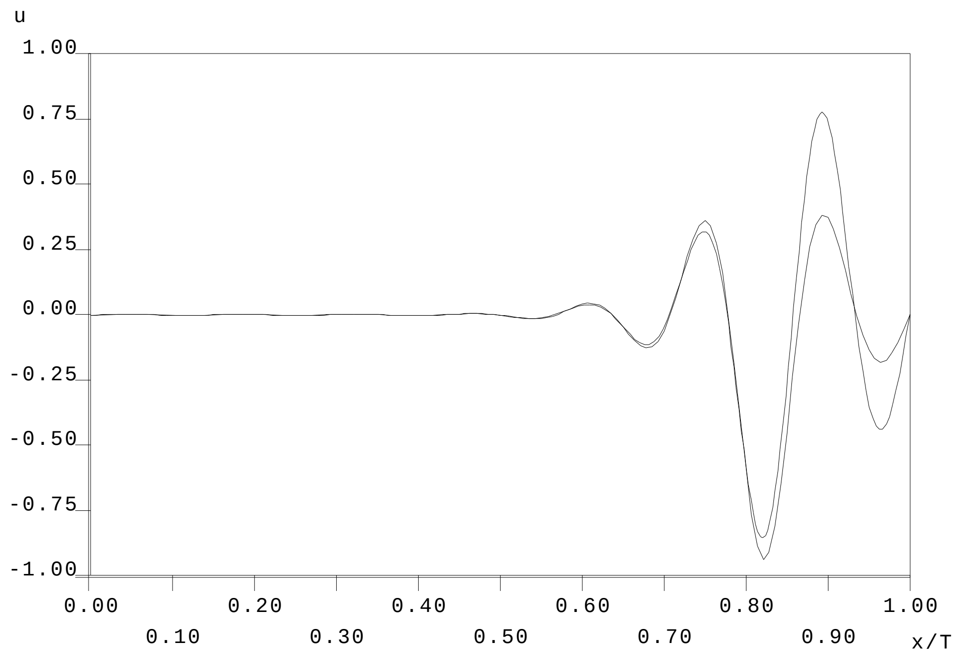

Figure 22.3: Two R_2-reversible homoclinic orbits at $P = 1.6$ corresponding to labels `1` (smaller amplitude) and `5` (larger amplitude)

22.4 Detailed AUTO-Commands.

COMMAND	ACTION
mkdir rev	create an empty work directory
cd rev	change directory
@dm rev	copy the demo files to the work directory
cp rev.f.1 rev.f	get equations file to `rev.f`
cp rev.dat.1 rev.dat	get the starting data to `rev.dat`
cp r.rev.1 r.rev	get the AUTO constants-file
cp s.rev.1 s.rev	get the HomCont constants-file
@fc rev	use the starting data in `rev.dat` to create `q.dat`
@h rev dat	increase `PAR(1)`
@sv 1	save output-files as `p.1, q.1, d.1`
cp r.rev.2 r.rev	get the AUTO constants-file
cp s.rev.2 s.rev	get the HomCont constants-file
@h rev 1	continue in reverse direction; restart from `q.1`
@ap 1	append output-files to `p.1, q.1, d.1`
cp rev.f.3 rev.f	get equations file with new value of `PAR(11)`
cp rev.dat.3 rev.dat	get starting data with different reversibility
cp r.rev.3 r.rev	get the AUTO constants-file
cp s.rev.3 s.rev	get the HomCont constants-file
@fc rev	use the starting data in `rev.dat` to create `q.dat`
@h rev dat	restart with different reversibility
@sv 3	save output-files as `p.3, q.3, d.3`
cp r.rev.4 r.rev	get the AUTO constants-file
cp s.rev.4 s.rev	get the HomCont constants-file
@h rev 3	continue in reverse direction; restart from `q.3`
@ap 3	append output-files to `p.3, q.3, d.3`

Table 22.1: Detailed AUTO-Commands for running demo `rev`.

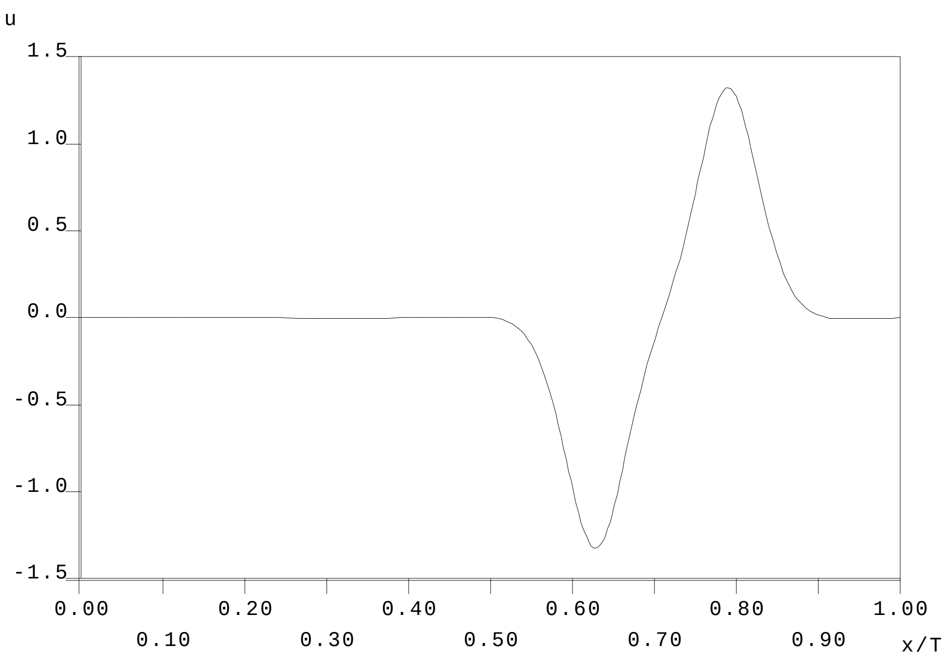

Figure 22.4: An R_2-reversible homoclinic orbit at label 8

Bibliography

Alexander, J. C., Doedel, E. J. & Othmer, H. G. (1990), 'On the resonance structure in a forced excitable system', *SIAM J. Appl. Math.* **50, No. 5**, 1373–1418.

Aronson, D. G. (1980), Density dependent reaction-diffusion systems, *in* 'Dynamics and Modelling of Reactive Systems', Academic Press, pp. 161–176.

Bai, F. & Champneys, A. (1996), 'Numerical detection and continuation of saddle-node homoclinic orbits of codimension one and codimension two', *J. Dyn. Stab. Sys.* **11**, 327–348.

Beyn, W.-J. & Doedel, E. J. (1981), 'Stability and multiplicity of solutions to discretizations of nonlinear ordinary differential equations', *SIAM J. Sci. Stat. Comput.* **2**(1), 107–120.

Buryak, A. & Akhmediev, N. (1995), 'Stability criterion for stationary bound states of solitons with radiationless oscillating tails', *Physical Review E* **51**, 3572–3578.

Champneys, A. & Kuznetsov, Y. (1994), 'Numerical detection and continuation of codimension-two homoclinic bifurcations', *Int. J. Bifurcation & Chaos* **4**, 795–822.

Champneys, A. & Spence, A. (1993), 'Hunting for homoclinic orbits in reversible systems: a shooting technique', *Adv. Comp. Math.* **1**, 81–108.

Champneys, A., Kuznetsov, Y. & Sandstede, B. (1996), 'A numerical toolbox for homoclinic bifurcation analysis'.

de Boor, C. & Swartz, B. (1973), 'Collocation at gaussian points', *SIAM J. Numer. Anal.* **10**, 582–606.

Doedel, E. J. (1981), 'AUTO, a program for the automatic bifurcation analysis of autonomous systems', *Cong. Numer.* **30**, 265–384.

Doedel, E. J. (1984), 'The computer-aided bifurcation analysis of predator-prey models', *J. Math. Biol.* **20**, 1–14.

Doedel, E. J. & Heinemann, R. F. (1983), 'Numerical computation of periodic solution branches and oscillatory dynamics of the stirred tank reactor with $a \to b \to c$ reactions', *Chem. Eng. Sci.* **38, No. 9**, 1493–1499.

Doedel, E. J. & Kernévez, J. P. (1986*a*), AUTO: Software for continuation problems in ordinary differential equations with applications, Technical report, California Institute of Technology. Applied Mathematics.

Doedel, E. J. & Kernévez, J. P. (1986*b*), A numerical analysis of wave phenomena in a reaction diffusion model, *in* H. G. Othmer, ed., 'Nonlinear Oscillations in Biology and Chemistry', Vol. 66, Springer Verlag, pp. 261–273.

Doedel, E. J. & Wang, X. J. (1995), AUTO94 : Software for continuation and bifurcation problems in ordinary differential equations, Technical report, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125. CRPC-95-2.

Doedel, E. J., Aronson, D. G. & Othmer, H. G. (1991), 'The dynamics of coupled current-biased Josephson junctions II', *Int. J. Bifurcation and Chaos* **1, No. 1**, 51–66.

Doedel, E. J., Friedman, M. & Monteiro, A. (1993), On locating homoclinic and heteroclinic orbits, Technical report, Cornell Theory Center; Center for Applied Mathematics, Cornell University.

Doedel, E. J., Keller, H. B. & Kernévez, J. P. (1991*a*), 'Numerical analysis and control of bifurcation problems: (I) Bifurcation in finite dimensions', *Int. J. Bifurcation and Chaos* **1**(3), 493–520.

Doedel, E. J., Keller, H. B. & Kernévez, J. P. (1991*b*), 'Numerical analysis and control of bifurcation problems: (II) Bifurcation in infinite dimensions', *Int. J. Bifurcation and Chaos* **1**(4), 745–772.

Fairgrieve, T. F. (1994), The computation and use of Floquet multipliers for bifurcation analysis, PhD thesis, University of Toronto.

Fairgrieve, T. F. & Jepson, A. D. (1991), 'O.K. Floquet multipliers', *SIAM J. Numer. Anal.* **28, No. 5**, 1446–1462.

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

Friedman, M. J. & Doedel, E. J. (1991), 'Numerical computation and continuation of invariant manifolds connecting fixed points', SIAM *J. Numer. Anal.* **28**, 789–808.

Henderson, M. E. & Keller, H. B. (1990), 'Complex bifurcation from real paths', *SIAM J. Appl. Math.* **50, No. 2**, 460–482.

Holodniok, M., Knedlik, P. & Kubíček, M. (1987), Continuation of periodic solutions in parabolic differential equations, *in* T. Küpper, R. Seydel & H. Troger, eds, 'Bifurcation: Analysis, Algorithms, Applications', Vol. INSM 79, Birkhäuser, Basel, pp. 122–130.

Hunt, G. W., Bolt, H. M. & Thompson, J. M. T. (1989), 'Structural localization phenomena and the dynamical phase-space analogy', *Proc. Roy. Soc. Lond. A* **425**, 245–267.

Keller, H. B. (1977), Numerical solution of bifurcation and nonlinear eigenvalue problems, *in* P. H. Rabinowitz, ed., 'Applications of Bifurcation Theory', Academic Press, pp. 359–384.

Keller, H. B. (1986), *Lectures on Numerical Methods in Bifurcation Problems*, Springer Verlag. Notes by A. K. Nandakumaran and Mythily Ramaswamy, Indian Institute of Science, Bangalore.

Kernévez, J. P. (1980), *Enzyme Mathematics*, North-Holland Press, Amsterdam.

Khibnik, A. I., Roose, D. & Chua, L. O. (1993), 'On periodic orbits and homoclinic bifurcations in Chua's circuit with a smooth nonlinearity', *Int. J. Bifurcation and Chaos* **3, No. 2**, 363–384.

Khibnik, A., Kuznetsov, Y., Levitin, V. & Nikolaev, E. (1993), 'Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps', *Physica D* **62**, 360–371.

Koper, M. (1994), Far-from-equilibrium phenomena in electrochemical systems, PhD thesis, Universiteit Utrecht, The Netherlands.

Koper, M. (1995), 'Bifurcations of mixed-mode oscillations in a three-variable autonomous Van der Pol-Duffing model with a cross-shaped phase diagram', *Physica D* **80**, 72–94.

Lentini, M. & Keller, H. B. (1980), 'The Von Karman swirling flows', *SIAM J. Appl. Math.* **38**, 52–64.

Lorenz, J. (1982), Nonlinear boundary value problems with turning points and properties of difference schemes, *in* W. Eckhaus & E. M. de Jager, eds, 'Singular Perturbation Theory and Applications', Springer Verlag.

Rodríguez-Luis, A. J. (1991), Bifurcaciones multiparamétricas en osciladores autónomos, PhD thesis, Department of Applied Mathematics, University of Seville, Spain.

Rucklidge, A. & Mathews, P. (1995), 'Analysis of the shearing instability in nonlinear convection and magnetoconvection'. Submitted to *Nonlinearity.*

Russell, R. D. & Christiansen, J. (1978), 'Adaptive mesh selection strategies for solving boundary value problems', *SIAM J. Numer. Anal.* **15**, 59–80.

Sandstede, B. (1995*a*), Constructing dynamical systems possessing homoclinic bifurcation points of codimension two, In preparation.

Sandstede, B. (1995*b*), Convergence estimates for the numerical approximation of homoclinic solutions, In preparation.

Sandstede, B. (1995*c*), Numerical computation of homoclinic flip-bifurcations, In preparation.

Scheffer, M. (1995), 'Personal communication'.

Smith, B., Boyle, J., Dongarra, J., Garbow, B., Ikebe, Y., Klema, X. & Moler, C. (1976), *Matrix Eigensystem Routines : EISPACK Guide*, Vol. 6, Springer Verlag.

Taylor, M. A. & Kevrekidis, I. G. (1989), Interactive AUTO : A graphical interface for AUTO86, Technical report, Department of Chemical Engineering, Princeton University.

Uppal, A., Ray, W. H. & Poore, A. B. (1974), 'On the dynamic behaviour of continuous stirred tank reactors', *Chem. Eng. Sci.* **29**, 967–985.

Wang, X. J. (1994), 'Parallelization and graphical user interface of AUTO94'. M. Comp. Sci. Thesis, Concordia University, Montreal, Canada.

Wang, X. J. & Doedel, E. J. (1995), AUTO94P : An experimental parallel version of AUTO, Technical report, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125. CRPC-95-3.