AUTO94

SOFTWARE FOR
CONTINUATION AND BIFURCATION PROBLEMS
IN ORDINARY DIFFERENTIAL EQUATIONS

Eusebius Doedel and Xianjun Wang
Computer Science Department, Concordia University, Montreal

Thomas Fairgrieve
Computer Science Department, University of Toronto

APPLIED MATHEMATICS REPORT
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA 91125

September 1994

This is a guide to the software package AUTO for continuation and bifurcation problems in ordinary differential equations (ODEs). A companion paper can be found in [1,2]. Earlier versions are described in [3,4].

The software described in this document, AUTO94, is the sequential version of AUTO94P [5], an experimental parallel program that currently runs on the Intel Delta. Both were developed in the Applied Mathematics Department at the California Institute of Technology as one of the projects of the Differential Equations Group in the Center for Research on Parallel Computation.

Compared to previous versions, AUTO94 has significant internal structural changes that are not apparent to the user. In addition it is somewhat more convenient to use, and it has a graphical user interface (GUI) written by Xianjun Wang. An earlier GUI for AUTO on SGI machines was written by Mark Taylor and Ioannis Kevrekidis [6]. AUTO94 also incorporates an improved Floquet multiplier algorithm written by Thomas Fairgrieve [7,8].

The first author is much indebted to H. B. Keller for his inspiration, encouragement and support. He is also thankful to AUTO users who have 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. Victor Burnley, California Institute of Technology contributed to one of the demos. 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. Who said that the AUTO interface is not user friendly ... ?

Pasadena, California
July 1994

TABLE OF CONTENTS

1. Installing AUTO.

1.1 Installation. To obtain a copy of AUTO, send email to *doedel@cs.concordia.ca*. The software is transported as a compressed, encoded file, called `auto.tar.Z.uu` . Put this file in your main directory. Remove the first few lines put there by the mailer, if applicable. Type

uudecode auto.tar.Z.uu

followed by

uncompress auto.tar.Z

and

tar xvfo auto.tar

This will result in the creation of a directory, `auto`, with one subdirectory, `94`, containing the AUTO files. Type

cd auto/94

to change directory to `auto/94`. Then type either

make sgi

to compile AUTO on Silicon Graphics machines, or just

make

on SUNs and, in principle, on other Unix systems. Upon compilation, type

make clean

to remove unnecessary files. Also enter the following command :

source $HOME/auto/94/cmds/auto.env

and add the above line to your `.cshrc` file.

EISPACK [9] routines used by AUTO for computing eigenvalues and Floquet multipliers are included in the package. The Graphical User Interface (GUI) requires X Windows and Motif [10] to be on your machine. It may be necessary to enter their pathname in the appropriate makefile in `auto/94/gui`. AUTO can also be run in Command Mode, independently of the GUI. To compile the AUTO library files without compiling the GUI, type *make cmd* in directory `auto/94`.

For timing purposes, the file `auto/94/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 prepare AUTO for transfer to another machine, type *make superclean* in directory `auto/94`. This will remove all executable, object, and other non-essential files, and thereby minimize the size of the package.

AUTO can be tested by typing *make > TEST &* in directory `auto/94/test`. This will execute all runs of all demos in `auto/94/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/94/test`. In case of suspected portability problems send `TEST` to *doedel@cs.concordia.ca*.

1.2 Restrictions on problem size. There are pre-defined size restrictions in the file `auto/94/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 bifurcation points NBIF (algebraic problems), and the number of user output points NUZR. See Section 5 for the significance of each of the above 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=20; it should never be decreased below that value. See also Section 3.4. Size restrictions can be changed by editing `auto.h`. This must be followed by recompilation by typing *make* in directory `auto/94/src`.

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

1.3 Compatibility with AUTO86. AUTO94 is compatible with AUTO86, except :

- There is no preprocessor.
- There are adjustable size limitations on certain AUTO-constants (see Section 1.2 above).
- The subroutine INIT is no longer required. Instead, all AUTO-constants are read from a file.
- The subroutine USZR is no longer required. Output parameter values are read from the AUTO-constants file.

2. Overview of Capabilities.

2.1 Summary. AUTO can do a limited bifurcation analysis of algebraic systems of the form

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

and of systems of ODEs of the form

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

Here p denotes one or more free parameters.
It can also do certain wave computations for the 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 [1,2] and [11,12].

2.2 Algebraic systems. Specifically, for (2.1) the program can :

- Compute solution branches.
- Locate bifurcation points and automatically compute bifurcating branches.
- Locate Hopf bifurcation points and continue these in two parameters.
- Locate folds (limit points) and continue these in two parameters.
- Do the above for fixed points of the discrete dynamical system

$$u^{(k+1)} = f(u^{(k)},p).$$

- Find extrema of an objective function along solution branches and successively continue such extrema in more parameters.

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.
- Locate folds, regular bifurcations, period doubling bifurcations, and bifurcations to tori, along branches of periodic solutions. Branch switching is possible at regular and period doubling bifurcations.
- Continue folds in two parameters. The two-parameter continuation of orbits of fixed period is also possible. This allows the approximate computation of curves of homoclinic orbits, if the period is sufficiently large.
- 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.
- Determine folds and bifurcation points along solution branches to the above boundary value problem. Branch switching is possible at bifurcation points. Curves of folds can be computed in two parameters.

2.4 Periodic waves in diffusive systems. 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

$$(2.2') \qquad \begin{aligned} u'(z) &= v(z), \\ v'(z) &= -D^{-1}\big[c\ v(z) + f\big(u(z),p\big)\big], \end{aligned}$$

  where $z = x - ct$, with the wave speed c specified by the user.
- Trace out branches of periodic wave solutions to (2.3) that emanate from a Hopf bifurcation point of (2.2′). The wave speed c is fixed along such a branch, but the wave length L, i.e., the period of periodic solutions to (2.2′), will normally vary. If the wave length L becomes large, i.e., if a homoclinic orbit of (2.2′) is approached, then the wave tends to a solitary wave solution of (2.3).

- 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).
- 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.

Note that the system (2.2′) 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 ODEs by the method of orthogonal collocation using piecewise polynomials with 2-7 collocation points per mesh interval [13]. The mesh automatically adapts to the solution to equidistribute the local discretization error [14]. 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. 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.

2.6 Output files. AUTO writes four output files :

`fort.6` : A summary of the computation is written in Fortran unit 6; usually the screen. Only special solution points are noted, including :

- EP : End point of branch; normal termination.
- MX : Abnormal termination; no convergence.
- UZ : Output at a user-specified parameter value.
- BP : Bifurcation point.
- LP : Fold (limit point).
- HB : Hopf bifurcation.
- PD : Period doubling bifurcation.
- TR : Torus bifurcation.

`fort.7` : The output file `fort.7` contains the bifurcation diagram. This information is written by the subroutines STHD, HEADNG, and STPLAE in the section 'Output (Algebraic Problems)', and by STPLBV in the section 'Output (Boundary Value Problems)' of the file `auto/94/src/autlib1.f` . The user has some control over this output via the AUTO-constant IPLT; see Section 5.9. The graphics program PLAUT can be used to graphically inspect this file.

`fort.8` : The output file `fort.8` contains complete graphics and restart data for selected solutions. The information per solution is much more extensive than that in `fort.7` and should normally be written only for a limited number of solutions. This file is written by WRTSP8 in the section 'Output (Algebraic Problems)' and by WRTBV8 in the section 'Output (Boundary Value Problems)' in `auto/94/src/autlib1.f` . The graphics program PLAUT can be used to graphically inspect this output.

`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.

3. How to Run AUTO.

3.1 Command mode. It is convenient to run AUTO with the graphical user interface described in Section 4. However, AUTO can also be run using the commands described in this section. We assume that the AUTO aliases have been activated; see Section 1.1; and that the equations-file `xxx.f` and the constants-file `r.xxx` have been prepared in the current user directory; see Section 3.2.

`@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 most common 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`.

A message will be printed if certain AUTO-constants exceed pre-defined maxima; see Section 1.2 for appropriate action.

`@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 files `p.xxx`, `q.xxx`, `d.xxx`, respectively.
- Type *@ap xxx yyy* to append the data files `p.xxx`, `q.xxx`, `d.xxx`, to `p.yyy`, `q.yyy`, `d.yyy`, respectively.

`@p` : - Type *@p xxx* to run the graphics program PLAUT 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`.

`@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 the demo files `xxx.f` and `r.xxx.*` from the demo directory `auto/94/demos/xxx` to the current user directory. Here `xxx` denotes a demo name; e.g., `pp2`; see Section 7.

`@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`.

`@pn` : Type *@pn xxx* to run the pendula animation program with data file `q.xxx`. (On SGI machine only; see the pen demo in Section 7, as well as the file `auto/94/pendula/README` and reference [15].)

3.2 User supplied files. The user must prepare the two files described below. This can be done with the graphical user interface described in Section 4. They can also be prepared independently.

`xxx.f` : A source file `xxx.f` containing the Fortran subroutines FUNC, STPNT, BCND, ICND, and FOPT. 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. For their precise form see the model equations-file `auto/94/gui/aut.f`, which can be directly loaded with the GUI button *Equations/New*; see Section 4.2. Examples are in directory `auto/94/demos`, where, e.g., the file `pp2/pp2.f` defines a two-dimensional dynamical system, and the file `bvp/bvp.f` defines a boundary value problem. The simplest way to create a new equations-file is to copy and edit the model equations-file or an appropriate demo file.

`r.xxx` : AUTO-constants for `xxx.f` are normally expected in a corresponding file `r.xxx`. Specific examples include the files `bvp/r.bvp` and `pp2/r.pp2` in `auto/94/demos`. See Section 5 for the significance of each constant.

The purpose 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 typically called only during the first AUTO run, when IRS=0; see Section 5.8. It defines a solution (u, p) of (2.1) or (2.2). This starting solution should not be a bifurcation point. When starting from a fixed point, the arguments of STPNT are (NDIM,U,PAR). When starting from an analytically known time- or space-dependent solution, the arguments of STPNT are (NDIM,U,PAR,T), where T denotes the independent time or space variable which takes values in the interval $[0, 1]$. When restarting from an analytically known periodic orbit, one must in addition specify the period in PAR(11).

BCND : A subroutine BCND that defines the boundary conditions, if any.

ICND : A subroutine ICND that defines the integral conditions, if any.

FOPT : A subroutine FOPT that defines the objective function, if any, for algebraic optimization problems.

3.3 User supplied derivatives. The recommended value of the AUTO-constant JAC is JAC=0; see Section 5.2; in which case derivatives need not be specified in FUNC, BCND, ICND, and FOPT. However, if one chooses to compute with JAC=1 then derivatives must be given. Examples of user-supplied derivatives can be found in directory `auto/94/demos`, namely in the equations-files `dd2/dd2.f`, `int/int.f`, and `opt/opt.f`.

3.4 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 5.7. Note the following restrictions on the use of PAR :

- The maximum number of parameters, NPARX in `auto/94/include/auto.h`, has pre-defined value NPARX=20. It should not be decreased below that value. Any increase of NPARX must be followed by recompilation of the AUTO libraries.
- For computations with ISW=2, see Section 5.8, only PAR(1)-PAR(9) should be used, as AUTO may need the remaining components internally.
- For computations with ISW=1 or ISW=-1, all entries of PAR are available to the user, except :
  - AUTO uses PAR(11) for the detection of Hopf bifurcations, while for periodic solutions PAR(11) contains the period.
  - For wave calculations PAR(10) contains the wave speed, PAR(11) the wave length, PAR(14) the independent time variable, and the NDIM diffusion constants are defined starting at PAR(15).
  - For algebraic optimization problems, AUTO uses PAR(11) to monitor the value of the objective function.

4. Graphical User Interface.

4.1 General overview. The AUTO94 graphical user interface (GUI) is a tool for creating and editing equations-files and constants-files; see Section 3.2 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 3.1 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. Then type

@*auto* or just @*a*

Here we assume that the AUTO aliases have been activated; see Section 1.1. The GUI includes a window for editing equations-files, 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. GUI buttons are normally activated by point-and-click action with the left mouse button. If a beep sound results then use the right mouse button.

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 4.2 below.

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 Section 5, 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.

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.

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.

4.2 The Menu Bar.

Equations. 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.

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

Write. 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.

Define. 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.

Run. 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.

Save. 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.

Append. 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.

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

Files. 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 data files `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`, respectively;
Move, to move data files `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`; and *Clean*, to delete files of the form `fort.*`, `*.o`, and `*.exe`.

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

Misc. 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`.

Help. 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.

4.3 Using the GUI. AUTO-commands are described in Section 3.1 and illustrated in the demos of Section 7. Below we list the main AUTO commands together with the corresponding GUI button.

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

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 3.1, 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*.

4.4 User notes.

Print. The *Misc/Print* button on the Menu Bar can be customized by editing `auto/94/include/GuiConsts.h`.

Color. GUI colors can be customized by creating an X resource file. Two model files can be found in directory `auto/94/gui`, namely, `Xdefaults.1` and `Xdefaults.2`. To become effective, edit one of these and copy it to `.Xdefaults` in your home directory. Color names can often be found in the file `/usr/lib/X11/rgb.txt`.

Help. The file `auto/94/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/94/gui/Makefile` by

CC = cc -Wf, -XNl20480 -O

On other machines, the maximum message length is the system defined maximum string literal length.

5. Description of AUTO-constants.

5.1 The AUTO-constants file. As described in Section 3.2, 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.exp` for the boundary value problem `exp.f` in directory `auto/94/demos/exp` is listed below.

```
2 4 0 1                      NDIM,IPS,IRS,ILP
1   1                        NICP,(ICP(I),I=1,NICP)
5 4 3 1 1 0 2 0              NTST,NCOL,IAD,ISP,ISW,IPLT,NBC,NINT
50  0.0 4.0 0.0 50.0         NMX,RL0,RL1,A0,A1
50 10 2 8 5 3 0              NPR,MXBF,IID,ITMX,ITNW,NWTN,JAC
1.d-4 1.d-4 1.d-4            EPSL,EPSU,EPSS
0.01 0.001 1.0 1             DS,DSMIN,DSMAX,IADS
0                            NTHL,(/,I,THL(I)),I=1,NTHL)
0                            NTHU,(/,I,THU(I)),I=1,NTHU)
2                            NUZR,(/,I,PAR(I)),I=1,NUZR)
1  1.0
1  3.0
```

The significance of the AUTO-constants, grouped by function, is described below.

5.2 Problem constants.

NDIM : Dimension of the system of algebraic equations or ODEs.

NBC : The number of boundary conditions. Must be specified if IPS=4 or 6.

NINT : The number of integral conditions. Must be specified if IPS=4 or 6.

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

- JAC=0 : Recommended value; no derivatives are given by the user.
- JAC=1 : Derivatives are given in the user-supplied subroutines FUNC, BCND, ICND and FOPT. Only necessary for extremely sensitive problems.

5.3 Discretization constants.

NTST : The number of mesh intervals to be used for discretization. NTST remains fixed during any particular run, but can be changed when restarting at a previously computed solution. However, the location of the mesh points can be made to adapt to the solution by setting the constant IAD. Recommended value of NTST : As small as possible, while maintaining convergence.

NCOL : The number of Gauss collocation points per mesh interval, ($2 \le$ NCOL ≤ 7). NCOL remains fixed during any given run, but can be changed when restarting at a previously computed solution. Strongly recommended value : NCOL=4.

IAD :

- IAD=0: Fixed mesh. This choice is not recommended, as it may result in the computation of spurious solutions.
- IAD>0 : Adapt the mesh every IAD steps along the branch. Strongly recommended value : IAD=3.

5.4 Tolerances.

EPSL : Relative convergence criterion for equation parameters in the Newton/Chord method. Recommended value EPSL=1.D-6 or EPSL=1.D-8.

EPSU : Relative convergence criterion for solution components in the Newton/Chord method. Recommended value EPSU=1.D-6 or EPSU=1.D-8.

EPSS : Relative arclength convergence criterion for detecting bifurcations and user output points. Recommended value EPSS=1.D-4 or EPSS=1.D-6, i.e., approximately 100 times the value of EPSL, EPSU.

ITMX : The maximum number of iterations allowed in the accurate location of bifurcations, folds, and user output points. Recommended value : ITMX=8.

NWTN : For ODEs only : After NWTN Newton iterations the Jacobian is frozen, i.e., the Chord method is used if NWTN $<$ iteration index $\le$ ITNW. Strongly recommended value : NWTN=3.

ITNW : The maximum number of combined Newton-Chord iterations. When 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`. Recommended value : Usually ITNW=5, but ITNW=7 for 'difficult' problems.

5.5 Continuation step size.

DS : Pseudo-arclength stepsize for the first step along any branch, including the first step after restart. DS may be chosen positive or negative; its sign determines the direction of computation. The relation DSMIN $\leq$ | DS | $\leq$ DSMAX must be satisfied. Recommended value of | DS | : somewhere between DSMIN and DSMAX, but typically closest to DSMIN.

DSMIN : The minimum allowable absolute value of the pseudo-arclength stepsize. Only relevant if IADS>0. DSMIN must be positive. Recommended value : highly problem-dependent; see Section 6.1.

DSMAX : The maximum allowable absolute value of the pseudo-arclength stepsize. Only relevant if IADS>0. DSMAX must be positive. Recommended value : highly problem-dependent; see Section 6.1.

IADS :

IADS=0 : Use fixed pseudo-arclength stepsize. Not recommended.

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. Recommended value : IADS=1.

NTHL : The number of parameter weights to be modified in the definition of the pseudo-arclength continuation algorithm. If NTHL=0 then all weights will have default value 1.0 . Under certain circumstances one may want to modify some weights. In such case NTHL>0 and one must enter NTHL pairs (*Parameter Index*, *Weight*), each pair on a separate line. In particular, for the computation of periodic solutions it is recommended to set NTHL=1, with, on a separate line, the pair (11   0.0), without brackets. This removes PAR(11); the period; from the pseudo-arclength continuation stepsize, which avoids period-induced limitations on the stepsize near orbits of infinite period. See the file `r.pp2.3` in `auto/94/demos/pp2` for an example.

NTHU : The number of solution-component weights to be modified in the definition of the pseudo-arclength continuation algorithm. Normally NTHU=0 and all weights will have default value 1.0 . Under exceptional circumstances one may want to modify some weights. In such case NTHU>0 and one must enter NTHU pairs (*Solution-Component-Index*, *Weight*), each pair on a separate line.

5.6 Diagram limits.

NMX : Maximum number of steps to be taken along any branch.

RL0 : Lower bound on the principal continuation parameter; i.e., on PAR(ICP(1)).

RL1 : Upper bound on the principal continuation parameter.

A0 : Lower bound on the principal solution measure, i.e., on the L_2-norm or other measure selected through IPLT; see Section 5.9.

A1 : Upper bound on the principal solution measure.

5.7 Free parameters.

NICP : The number of free equation parameters specified, i.e., the number of indices (ICP(I),I=1 NICP). This number should be greater than or equal to the generic number required by the computation.

ICP : Designates the free parameter(s) :

- Often there is only a single free equation parameter. In this case NICP=1 and ICP(1) specifies the free parameter's index, i.e., PAR(ICP(1)) is free.
- For the two-parameter continuation of folds and Hopf bifurcations, set NICP=2 and use ICP(1) and ICP(2) to indicate the free parameters.
- For boundary value problems set NICP = NBC + NINT−NDIM +1, and use ICP(1)$\cdots$ICP(NICP) to specify the free equation parameters. A simple case is when NBC=NDIM and NINT=0. Then NICP=1, i.e., only one free parameter needs to be specified.
- For the continuation of folds in a boundary value problem, set NICP = NBC + NINT−NDIM +2. A simple case is when NBC=NDIM and NINT=0. Then NICP=2, i.e., two free parameters must be specified.
- For the first run of an optimization problems one must set ICP(1)=11, as AUTO uses PAR(11) as principal parameter to monitor the value of the objective function. Furthermore, one must designate one free equation parameter in ICP(2). Thus NICP=2 in such a first run. Folds with respect to the principal parameter then correspond to local one-parameter extrema of the objective function. In a second run, with NICP=3, one can restart at such an extremum and specify an additional equation parameter in ICP(3). Folds located in a second run then correspond to local two-parameter extrema of the objective function. This procedure can be repeated indefinitely.

5.8 Computation constants.

ILP :

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

ILP=1 : Detection of folds. Use if subsequent two-parameter fold continuation is intended.

ISP :

ISP=0 : For all equations : No detection of bifurcation points. For periodic solutions : No Floquet multipliers.

ISP=1 : For algebraic equations : Detection of bifurcations. For ODEs : No detection of bifurcations. For periodic solutions : Floquet multipliers computed.

ISP=2 : For all equations : Detection of bifurcations. For periodic solutions : Floquet multipliers computed.

ISP=3 : If IPS=2,3,6 then the choice ISP=3 is similar to ISP=2, except that the first two, rather than one, closest Floquet multipliers to $z = 1$ are excluded from stability and bifurcation considerations. Furthermore, folds will be determined with respect to the period. This option can be useful in energy preserving systems.

If IPS=2,3, or 6, then the choice ISP=2,3 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 potential secondary periodic bifurcations, if ISP=2,3, will be discontinued and a warning message will be printed in `fort.9`.

ISW : Normally ISW=1.

If ISW=−1 and IPS= 2,3,4, or 6, and if IRS is the label of a bifurcation point, then the restart procedure will attempt to switch branches, rather than continue computation of the given branch. For period doubling bifurcations it is recommended that NTST be increased to twice the value used in the computation that detected the bifurcation.

If ISW=2 and IPS=−1,0,1,2,3 or 4, and if IRS is the label of a fold or a Hopf bifurcation point, then the program will try to trace out a curve of such points in two parameters. The choice of second parameter must be specified in ICP(2), while ICP(1) remains the index of the first parameter. For boundary value problems with more than one free parameter one should use ICP(NBC+NINT−NDIM+2) to indicate the additional parameter necessary for the computation of the curve of folds.

MXBF : For algebraic problems only : MXBF sets the maximum number of bifurcating branches to be traced out. Additional bifurcations will be noted, but the corresponding bifurcating branches will not be computed. If MXBF is negative then bifurcations will only be traced out in one direction. If MXBF is positive then both directions are traced out.

IRS :

IRS=0 : A new problem : No previously computed restart data. A starting solution must be specified in STPNT; see Section 3.2.

IRS>0 : Computation to be restarted at a previously computed solution with label IRS. This solution is normally expected to be present in the current solution file `q.xxx`. Most AUTO-constants can be modified at a restart point.

IPS : This constant defines the problem type :

IPS=0 : Algebraic bifurcation problem. Hopf bifurcations will not be detected and stability properties will not be indicated in `fort.7`.

IPS=1 : Algebraic bifurcation problem with detection of Hopf bifurcations. The sign of PT, the point number, in `fort.7` is used to indicate stability : $-=$ stable , $+ =$ unstable.

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 : $-=$ stable , $+ =$ unstable.

IPS= 2 : Computation of periodic solutions. Starting data can be a Hopf bifurcation point from a previous run with IPS=1, or a periodic orbit from a previous run with IPS=2 or 3. One can also specify an analytically known periodic orbit in the user subroutine STPNT. One free equation parameter must be designated in ICP(1); AUTO will automatically add the period, PAR(11), as second free parameter. The sign of PT in `fort.7` is used to indicate stability : $-=$ stable , $+ =$ unstable or unknown.

IPS= 3 : Two-parameter continuation of orbits of fixed period. Starting data may be a periodic orbit from a previous run with IPS=2 or 3, or an analytically known periodic orbit specified in STPNT. Two free equation parameters must be designated in ICP(1) and ICP(2). The sign of PT in `fort.7` is used to indicate stability : $-=$ stable , $+ =$ 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.

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

IPS= 6 : A boundary value problem. This option is equivalent to IPS=4, except that AUTO will also compute Floquet multipliers and check for associated bifurcations. This option is useful for computing periodic solutions of the second kind; rotations; where the user has to define the periodicity conditions in BCND. It is also useful for computing periodic solutions to problems where integral quantities; defined in ICND; are to be monitored.

IPS=11 : Spatially uniform solutions of the system (2.3) with detection of bifurcations to traveling waves, i.e., continuation of fixed points of the system (2.2'). The advantage of this option over IPS=1 is that AUTO automatically generates (2.2'). The user need only define f, initialize the wave speed in PAR(10), the NDIM diffusion constants in PAR(15,16,...), and a free equation parameter in ICP(1).

IPS=12 : Continuation of traveling waves, i.e., continuation of periodic solutions of the system (2.2'). This is similar to the choice IPS=2, except that AUTO will automatically use (2.2') rather than (2.2). The user must designate one free equation parameter in ICP(1). Note that AUTO uses PAR(10) for the fixed wave speed and PAR(11) for the variable wave length. 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 or 13.

IPS=13 : Continuation of traveling waves of fixed wave length, i.e., continuation of fixed period solutions of the system (2.2'). This is similar to the choice IPS=3, except that AUTO will automatically use the equations (2.2') rather than (2.2). Two free parameters must be designated, for example, the wave speed PAR(10) and a equation parameter. Starting data can be a traveling wave from a previous run with IPS=12 or 13.

IPS=14 : Time evolution computation for (2.3) on a periodic space interval. The initial data must be specified in the interval [0,1]. The actual length of the interval must be specified in PAR(11). AUTO uses PAR(14) for the independent time variable and PAR(15,16,...) for the diffusion constants. DS, DSMIN, and DSMAX govern the stepsize in time : Pseudo-arclength in the (U,Time)-variables is used. Starting data may be solutions from a previous run with IPS=12, 13 or 14, or given analytically in STPNT.

5.9 Output control.

NPR : Write plotting and restart information in `fort.8` every NPR steps along solution branches.

IID : Controls diagnostic output printed in `fort.9`. Higher values of IID result in more output.

IID=0 : Minimal output.

IID=2 : Recommended value of IID.

IID=3 : Algebraic problems : Jacobian and residual vector printed at the initial starting point.

IID=4 : ODEs : Reduced system and residual vector printed at each iteration. This setting should not normally be used.

IID=5 : ODEs : Very extensive output from the linear equation solver. This setting should not normally be used.

IPLT : Gives the user some control over the `fort.7` output, namely the choice of principal solution measure; the second real number written per output line.

- If IPLT $= 0$: L_2 - norm.
- If $0 <$ IPLT $\leq$ NDIM : Maximum of the IPLT'th component.
- If $-$NDIM $\leq$ IPLT <0 : Minimum of the IPLT'th component.
- If NDIM $<$ IPLT $\leq$ 2*NDIM : Integral of the (IPLT$-$NDIM)'th component.

Note that, for algebraic problems, maximum and 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.

NUZR : Allows the setting of parameter values at which labelled graphics or restart information is wanted. Set NUZR=0 if no such output is needed. If NUZR>0 then one must enter NUZR pairs (*Parameter-Index,Parameter-Value*), each pair on a separate line. For an example, see the file `r.exp` at the beginning of this section.

6. Notes on Using AUTO.

6.1 Efficiency. In AUTO, efficiency has at times been sacrificed for generality of programming. This applies in particular to the two-parameter continuation of folds and Hopf bifurcations. 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, e.g., 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, THU, THL, EPSU, EPSL, EPSS also affect efficiency. Understanding their significance is therefore useful; see Section 5.4 and Section 5.5. Finally, it is recommended that initial computations be done with ILP=0; no fold detection; and ISP=1; no bifurcation detection for ODEs.

6.2 Correctness of results. AUTO-computed solutions to ODEs are almost always structurally correct, because the mesh adaption strategy, if IAD>0, safeguards 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.

6.3 Bifurcation points and folds. As noted above, in Section 6.1, 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 singular point after another; namely folds.

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 output file `fort.9`.

6.4 Floquet multipliers. The Floquet multiplier solver of AUTO94 has been written by Thomas Fairgrieve [7]. For a detailed description of the algorithm see Fairgrieve and Jepson [8].

If IPS=2,3,6,12,13, and ISP=1,2, then AUTO extracts approximations to the Floquet multipliers from the Jacobian of the linearized system in Newton's method. This procedure is very efficient; the multipliers are computed at negligible extra cost. 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`. Note that if this multiplier becomes inaccurate, then the automatic detection of potential secondary periodic bifurcations (if ISP=2,3) will be discontinued and a warning message printed in `fort.9`. It is strongly recommended that the contents of this file be habitually inspected, in particular to verify whether solutions labelled as LP, BP, PD, or TR have indeed been correctly classified.

The current linear equations solver of AUTO is described in [2]. A more robust linear systems solver for AUTO is presented in [16], but at this time it has not yet been incorporated.

6.5 Memory requirements. Pre-defined maximum values of certain AUTO-constants are defined in in `auto/94/include/auto.h`; see also Section 1.2. These maxima effect the run-time memory requirements and should not be set to unnecessarily large values. If an application does not require the differential equations options, i.e., if only IPS=0,1,-1,5, or 11 are used, and if NDIM is "large", then memory requirements can be much reduced by setting NTSTX=NCOLX=NBCX=NINTX=1 in `auto/94/include/auto.h`, followed by recompilation of the AUTO libraries.

7. AUTO Demos.

The directory `auto/94/demos` has a number of subdirectories containing the necessary files for certain illustrative bifurcation calculations. Each subdirectory `xxx`; e.g., `pp2`; corresponds to a particular equation and contains an equations-file `xxx.f`; e.g., `pp2.f`; and one or more constants-files `r.xxx.i`; e.g., `r.pp2.1`, `r.pp2.2`, $\cdots$.

For each demo, the Command Mode actions for performing various calculations and file maintenance operations are listed in this section; see Section 4.3 for corresponding GUI actions. The first action is always to copy the demo files to a user work directory. In Command Mode this is done with the *@dm xxx* command. In GUI Mode use the *Equations/Demo* button. Commands for plotting the data files are not listed in this section; use of the graphics program PLAUT is described in Section 8. To understand in detail how AUTO is instructed to carry out a particular task, inspect the equations-file `xxx.f`, as well as the appropriate constants-file `r.xxx`. For second and subsequent runs, it is indicated below which AUTO-constants in `r.xxx` have been changed with respect to the preceding run.

It is also possible to execute all runs of a selected demo automatically. This is mainly useful for testing purposes. To do this in GUI Mode, click *Demos/Select*, select a demo, and click the *Run* button in the pop-up window. In Command Mode type *make* in the selected demo directory.

New demos that illustrate a new type of computation are welcome for inclusion in this manual. Contact the first author for more information.

pp2 : Basic computations for continuous dynamical systems.

This demo illustrates the computation of stationary solutions, periodic solutions, and the two-parameter continuation of folds, Hopf bifurcation points, and homoclinic orbits, for an autonomous ODE. The equations, which model a predator-prey system with harvesting, are

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

Here p_1 is used as the principal continuation parameter, $p_3 = 5$, $p_4 = 3$, and, initially, $p_2 = 3$. For two-parameter computations p_2 is also free.

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 labelled 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 labelled periodic solutions
@ap pp2	append output files to `p.pp2, q.pp2, d.pp2`
cp r.pp2.5 r.pp2	constants changed : IRS, IPS, ISW, NICP
@r pp2	5th run; two-parameter 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; two-parameter 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; two-parameter continuation of homoclinic orbits
@sv hom	save output files as `p.hom, q.hom, d.hom`
cp r.pp2.8 r.pp2	constants changed : IRS, RL1
@r pp2 hom	8th run; restart homoclinic orbits; read restart data from `q.hom`
@ap hom	append the output files to `p.hom, q.hom, d.hom`
@cl	clean directory of unnecessary files
@dl lp	as an example, delete `p.lp, q.lp, d.lp`

exp : A boundary value problem (Bratu's equation).

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

$$u_1' = u_2,$$
$$u_2' = -p_1 e^{u_1},$$

with boundary conditions $u_1(0) = 0, \quad u_1(1) = 0$. This equation is also considered in [1].

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 labelled solution with increased accuracy
@ap exp	append output files to `p.exp, q.exp, d.exp`

int : An ODE with boundary and integral constraints.

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

$$u_1' = u_2,$$
$$u_2' = -p_1 e^{u_1},$$

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; compute a curve of folds
@sv lp	save the output files as `p.lp`, `q.lp`, `d.lp`

dd2 : Basic computations for discrete dynamical systems.

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

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

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.

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 two-parameter curve of Naimark-Sacker bifurcations
@sv ns	save output files as `p.ns, q.ns, d.ns`

opt : A model algebraic optimization problem.

This demo illustrates the method of successive continuation for constrained optimization problems [1,2], 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.

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`

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

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

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_3 is free. In the third run p_2 is free, and p_3 is fixed.

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`

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

$$u_1' = u_2,$$
$$u_2' = -(p_1\pi)^2 u_1 + u_1^2,$$

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`

pp3 : A continuous dynamical system with period-doubling.

This demo illustrates the computation of stationary solutions, Hopf bifurcations, and periodic solutions, including the computation of a branch from a period-doubling bifurcation. The equations model a 3D predator-prey system with harvesting [17].

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

All equation parameters, except p_1, are fixed.

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`

wav : Periodic waves in a nonlinear parabolic PDE.

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 [18], are :

$$\frac{\partial u_1}{\partial t} = \frac{\partial^2 u_1}{\partial x^2} - p_1\big[p_4 R(u_1,u_2) - (p_2 - u_1)\big],$$
$$\frac{\partial u_2}{\partial t} = \beta\frac{\partial^2 u_2}{\partial x^2} - p_1\big[p_4 R(u_1,u_2) - p_7(p_3 - u_2)\big],$$

$$R(u_1,u_2) = \frac{u_2}{p_5 + u_2}\ \frac{u_1}{1 + u_1 + p_6 u_1^2}\ .$$

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, NICP, NUZR
@r wav	4th run; computation of fixed wave length waves (on a ring)
@sv rng	save output files as `p.rng, q.rng, d.rng`
cp r.wav.5 r.wav	constants changed : IPS, NMX, NPR, NICP
@r wav	5th run; time evolution computation
@sv tim	save output files as `p.tim, q.tim, d.tim`

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

$$\begin{aligned} u_i' &= u_i, \\ v_i' &= -p_1 \; e(u_i), \end{aligned} \qquad i = 1, \cdots, NDIM/2,$$

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

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	do the computations
@sv tim	save output files as `p.tim`, `q.tim`, `d.tim`

ezp : Complex bifurcation in a boundary value problem.

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

$$u_1' = u_2,$$
$$u_2' = -p_1 e^{u_1},$$

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 exp demo. It is known [19] 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`

non : A non-autonomous boundary value problem.

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

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

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

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

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

1cl : A one-cell, two-substrate enzyme model.

This demo, which corresponds to computations in [1], shows how one can restart one-parameter curves from a two-parameter curve. The equations, that model a one-compartment activator-inhibitor system [20], are given by

$$\frac{ds}{dt} = (s_0 - s) - \rho R(s,a),$$
$$\frac{da}{dt} = \alpha(a_0 - a) - \rho R(s,a),$$

where

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

The free parameter is ρ. In the two-parameter Hopf continuation s_0 is also free.

COMMAND	ACTION
mkdir 1cl	create an empty work directory
cd 1cl	change directory
@dm 1cl	copy the demo files to the work directory
cp r.1cl.1 r.1cl	get the first constants-file
@r 1cl	1st run; compute a stationary solution branch
@sv 1	save output files as `p.1, q.1, d.1`
cp r.1cl.2 r.1cl	constants changed : IRS, ISW, NMX
@r 1cl 1	two-parameter curve of Hopf bifurcations; restart data from `q.1`
@sv hb	save output files as `p.hb, q.hb, d.hb`
cp r.1cl.3 r.1cl	constants changed : IRS, ISW, NMX
@r 1cl hb	compute a second stationary solution branch; restart data from `q.hb`
@sv 2	save output files as `p.2, q.2, d.2`
cp r.1cl.4 r.1cl	constants changed : IRS, NMX, DS
@r 1cl 2	restart solution branch in above run; restart data from `q.2`
@ap 2	append output files to `p.2, q.2, d.2`

2cl : A two-cell, one-substrate enzyme model.

The equations, that model a two-compartment enzyme system [20], are given by

$$s_1' = (s_0 - s_1) + (s_2 - s_1) - \rho R(s_1),$$
$$s_2' = (s_0 + \mu - s_2) + (s_1 - s_2) - \rho R(s_2),$$

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 [1].

COMMAND	ACTION
mkdir 2cl	create an empty work directory
cd 2cl	change directory
@dm 2cl	copy the demo files to the work directory
cp r.2cl.1 r.2cl	get the constants-file
@r 2cl	compute stationary solution branches
@sv 2cl	save output files as `p.2cl, q.2cl, d.2cl`

cbv : Optimization in a boundary value problem.

This demo illustrates use of the method of successive continuation for a boundary value optimization problem. For a detailed description see [2]. Consider the system

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

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

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

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

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

where

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

with

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

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

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

$$
\begin{aligned}
&\int_0^1 \Big[-e^{p(u_1,\lambda_2,\lambda_3)} w_2(t) - \frac{1}{5}\gamma\lambda_1\Big] \; dt = 0,\\
&\int_0^1 \Big[-\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)} u_1(t)^2 w_2(t) - \frac{1}{5}\gamma\lambda_2 - \alpha_2\Big] \; dt = 0,\\
&\int_0^1 \Big[-\lambda_1 e^{p(u_1,\lambda_2,\lambda_3)} u_1(t)^4 w_2(t) - \frac{1}{5}\gamma\lambda_3 - \alpha_3\Big] \; dt = 0.
\end{aligned}
$$

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

ext : Spurious solutions to a boundary value problem.

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

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

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 5.3. This example is also considered in [2].

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`

nag : Heteroclinic orbits : a saddle-saddle connection.

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

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

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

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

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 [21].

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`

fsh : Heteroclinic orbits : a saddle-node connection.

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

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

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

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

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 [21] for details. 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, NICP, 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`

phs : Effect of the phase condition on periodic orbits.

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

$$u_1' = \lambda u_1 - u_2,$$
$$u_2' = u_1(1 - u_1).$$

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, e.g., 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 branch
@ap phs	append output files to `p.phs`, `q.phs`, `d.phs`

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. In this case one can not make use of the built-in capability of AUTO to compute periodic solutions; instead the boundary conditions and the phase condition must be defined by the user in the subroutines BCND and ICND. The model equations, a system of two coupled Josephson junctions, or equivalently, two coupled pendula [15], are given by

$$\phi_1'' + \epsilon\phi_1' + \delta \ \sin\phi_1 = I + \gamma(\phi_2 - \phi_1)$$
$$\phi_2'' + \epsilon\phi_2' + \delta \ \sin\phi_2 = I + \gamma(\phi_1 - \phi_2).$$

The first run is a homotopy from $\delta = 0$, for which there is an analytical solution, to $\delta = 1$. In the second run, part of the in-phase solution branch is computed and a bifurcation to out-of phase solutions is located. Part of the bifurcating branch is computed in the third run. Throughout, the forcing $I = 0$, and ϵ is a free parameter, even though its computed value is zero. In fact, an appropriate free equation parameter is necessary for computing energy preserving solutions. The phase gain per period is 4π, rather than 2π, even in the first two runs. As a result, the bifurcation, which is actually a period-doubling, is located as a pitch fork.

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
@r pen	homotopy to $\delta = 1$
@sv 0	save output files as `p.0, q.0, d.0`
cp r.pen.2 r.pen	constants changed : IRS, ICP(1), NMX
@r pen 0	branch of in-phase rotations; restart from `q.0`
@sav pen	save output files as `p.pen, q.pen, d.pen`
cp r.pen.3 r.pen	constants changed : IRS, ISW, ISP, IPS, NMX, NPR, DSMAX
@r pen	compute bifurcating branch of out-of-phase rotations;
@ap pen	append output files to `p.pen, q.pen, d.pen`
@pn pen	run an animation program to view the solutions in `q.pen`
	(on SGI machines only; see also [15] and `auto/94/pendula/README`)

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

$$x' = x + \beta y - x(x^2 + y^2),$$
$$y' = -\beta x + y - y(x^2 + y^2),$$

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

$$v' = \big(F(v) - w\big)/\epsilon,$$
$$w' = v - dw - \big(b + r\sin(\beta t)\big),$$

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. It contains a fold and a bifurcation point. For detailed results see [22].

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`

spb : A singularly perturbed boundary value problem.

This demo illustrates the use of continuation to compute a small ϵ solution to the singularly perturbed boundary value problem

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

with boundary conditions $u_1(0) = 3/2, \quad u_1(1) = \gamma$. 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 requires more than 1500 continuation steps. For a detailed analysis of the solution behavior see [23].

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`

lor : Starting a periodic orbit from numerical data.

The special purpose subroutines used in this demo were written by Victor Burnley, California Institute of Technology.

The demo illustrates how to start the computation of a branch of periodic solutions from numerical data obtained, for example, from an initial value simulation. The data, which must correspond to a complete periodic orbit, is expected in `fort.4`. In this demo, the user-supplied subroutine STPNT contains a call to the library subroutine STDAT in `auto/94/src/stdat.f`

As an illustrative application we consider the Lorenz equations

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

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 DATA. This file must be copied to fort.4 before the start of the AUTO calculations.

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
cp lor.dat fort.4	copy the periodic orbit data to `fort.4`
@r lor	compute a solution branch starting from the numerical data
@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`

8. Using the Graphics Program PLAUT.

In command mode, type *@p xxx* to use the graphics program PLAUT to inspect the contents of the AUTO data files `p.xxx` and `q.xxx`. In GUI mode, click the *Plot* button on the Menu Bar. More precisely, click *Plot/Plot* for plotting the active data files, and *Plot/Name* for other data files. In all cases a graphics window will appear in which PLAUT commands can be entered. The PLAUT *Help* command lists the available commands.

To run PLAUT directly, e.g., on a non-window Tektronix compatible screen, do the following : First copy the data files `p.xxx` and `q.xxx` to `fort.7` and `fort.8`, respectively. Then run `auto/94/bin/plaut`. It is convenient to put these commands in a file. Conversion of PLAUT to other graphics environments is possible by making appropriate changes in `auto/94/plaut/plautint.f`.

To illustrate the use of PLAUT, we assume that the first three runs of the pp2-demo have been completed in a user work directory; see Section 7. The files `p.pp2` and `q.pp2` will then exist in this directory. To interactively plot their contents, run PLAUT, as indicated above, by typing *@p pp2*, or by the corresponding GUI action. Then enter the commands below in the PLAUT window.

PLAUT-COMMAND	ACTION
d3	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
d2	choose other default settings
bd0	bifurcation diagram
bd	get blow-up of current bifurcation diagram
0 1 -0.25 1	enter limits
cl	clear the screen

PLAUT-COMMAND	ACTION
2d	enter '2D' mode, for plotting labelled solutions
12 13 14	select labelled orbits 12, 13, and 14
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
cl	clear the screen
ex	exit from 2D mode
3d	enter '3D' mode, for plotting labelled solutions
15	select labelled orbit 15
cz	put coordinate axes at $(0, 0, 0)$
d	display (default) columns 1,2,3 in `q.pp2`
sc	scale the diagram
2 1.8 2	enter scaling factors
d	display; as above, $x =$time, $y = u_1$, $z = u_2$
pj	select projections
d	display
sav	save plot
fig.1	upon prompt, enter file name, e.g., `fig.1`
cl	clear the screen
ex	exit from 3D mode
end	exit from PLAUT

Each saved plot must be written in a separate file, e.g., `fig.1` above. These files are in Tektronix PLOT10 format. There are several programs available for converting such files to PostScript format. (Typical names are *tek2ps* and *ps4014*).

References.

[1] E. J. Doedel, H. B. Keller, J. P. Kernévez, Numerical Analysis and Control Of Bifurcation Problems, Part I : Bifurcation in Finite Dimensions, Int. J. Bifurcation and Chaos, Vol. 1, No. 3. 1991, 493-520.

[2] E. J. Doedel, H. B. Keller, J. P. Kernévez, Numerical Analysis and Control Of Bifurcation Problems, Part II : Bifurcation in Infinite Dimensions, Int. J. Bifurcation and Chaos, Vol. 1, No. 4. 1991, 745-772.

[3] E. J. Doedel, AUTO: A program for the automatic bifurcation analysis of autonomous systems, Cong. Num. 30, 1981, 265-284, (Proc. 10th Manitoba Conf. on Num. Math. and Comp., Univ. of Manitoba, Winnipeg, Canada).

[4] E. J. Doedel, J. P. Kernévez, AUTO: Software for continuation and bifurcation problems in ordinary differential equations, Applied Mathematics Report, California Institute of Technology, 1986, 226 pages.

[5] X. J. Wang, Parallelization and graphical user interface of AUTO94, M. Comp. Sci. Thesis, Concordia University, 1994.

[6] M. A. Taylor, I. G. Kevrekidis, Interactive AUTO : A graphical interface for AUTO86, Technical Report, Department of Chemical Engineering, Princeton University, 1989.

[7] T. F. Fairgrieve, PhD Thesis, University of Toronto, 1994.

[8] T. F. Fairgrieve, A. D. Jepson, O.K. Floquet multipliers, SIAM J. Numer. Anal. 28. No. 5, 1991, 1446-1462.

[9] B. Smith, J. Boyle, J. Dongarra, B. Garbow, Y. Ikebe, Klema, C. Moler, Matrix Eigensystem Routines : EISPACK Guide, Lecture Notes in Computer Science 6, Springer Verlag, 1976.

[10] A. Nye, R. O'Reilly, X Volume 6, Motif Programming Manual, O'Reilly & Associates Inc., 1993.

[11] H. B. Keller, Numerical solution of bifurcation and nonlinear eigenvalue problems, in: Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, 1977, 359-384.

[12] H. B. Keller, Lectures on Numerical Methods in Bifurcation Problems, Notes by A. K. Nandakumaran and Mythily Ramaswamy, Indian Institute of Science, Bangalore, 1986.

[13] C. de Boor, B. Swartz, Collocation at Gaussian points, SIAM J. Numer. Anal. 10, 1973, 582-606.

[14] R. D. Russell, J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, SIAM J. Numer. Anal. 15, 1978, 59-80.

[15] E. J. Doedel, D. G. Aronson, H. G. Othmer, The dynamics of coupled current-biased Josephson Junctions II, Int. J. Bifurcation and Chaos, Vol. 1, No. 1, 1991, 51-66.

[16] L. Liu, PhD Thesis, Simon Fraser University, 1993.

[17] E. J. Doedel, The computer-aided bifurcation analysis of predator-prey models, J. Math. Biol. 20, 1984, 1-14.

[18] E. J. Doedel, J. P. Kernévez, A numerical analysis of wave phenomena in a reaction diffusion model, in: Nonlinear Oscillations in Biology and Chemistry, H. G. Othmer, ed., Lecture Notes in Biomathematics 66, Springer Verlag, 1986, 261-273.

[19] M. E. Henderson, H. B. Keller, Complex bifurcation from real paths, SIAM J. Appl. Math. 50, No. 2, 1990, 460-482.

[20] J. P. Kernévez, Enzyme Mathematics, North-Holland Press, Amsterdam, 1980.

[21] M. J. Friedman, E. J. Doedel, Numerical computation and continuation of invariant manifolds connecting fixed points, SIAM J. Numer. Anal., Vol. 28, No. 3, 1991, 789-808.

[22] J. C. Alexander, E. J. Doedel, H. G. Othmer, On the resonance structure in a forced excitable system, SIAM J. Appl. Math. 50, No. 5, 1990, 1373-1418.

[23] J. Lorenz, Nonlinear boundary value problems with turning points and properties of difference schemes, in : Singular Perturbation Theory and Applications, W. Eckhaus, E. M. de Jager, eds., Lecture Notes in Mathematics 942, Springer Verlag 1982.