1 Running AUTO using Unix Commands

AUTO can be run with the Unix commands described below. These Unix commands run both directly in the shell.

`@dm` : Type `@dm xxx` to copy all files from `auto/07p/demos/xxx` to the current user directory. Here `xxx` denotes a demo name; e.g., `abc`. Note that the `@dm` command also copies `.auto` files to the current user directory. To avoid the overwriting of existing files, always run demos in a clean work directory.

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

- Type `@r xxx yyy` to run AUTO with equations-file `xxx.f90` and restart data-file `s.yyy`. AUTO-constants must be in `c.xxx`.
- Type `@r xxx yyy zzz` to run AUTO with equations-file `xxx.f90`, restart data-file `s.yyy` and constants-file `c.zzz`.

`@sv` : Type `@sv xxx` to save the output-files `fort.7`, `fort.8`, `fort.9`, as `b.xxx`, `s.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 `b.xxx`, `s.xxx`, `d.xxx`, resp.

- Type `@ap xxx yyy` to append `b.xxx`, `s.xxx`, `d.xxx`, to `b.yyy`, `s.yyy`, `d.yyy`, resp.

`@pp` : Type `@pp xxx` to run the graphics program PyPLAUT for the graphical inspection of the data-files `b.xxx` and `s.xxx`.

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

`@p` : The command `@p` is equivalent to `@pp` but runs the graphics program PLAUT instead.

`@cp` : Type `@cp name1 name2`, or `@cp name1 name2 name3`, or `@cp name1 name2 name3 name4` to copy the data-files dir1/c.xxx, dir1/b.xxx, dir1/s.xxx, and dir1/d.xxx, to dir2/c.yyy, dir2/b.yyy, dir2/s.yyy, and dir2/d.yyy.

The values of dir1/?.xxx and dir2/?.yyy are as follows, depending on whether name1 is a directory or name2 is a directory:

`@cp name1 name2`
no directory names: ./?.name1 and ./?.name2
name1 is a directory: name1/?.name2 and ./?.name2
name2 is a directory: ./?.name1 and name2/?.name1

`@cp name1 name2 name3`
name1 is a directory: name1/?.name2 and ./?.name3
name2 is a directory: ./?.name1 and name2/?.name3

`@cp name1 name2 name3 name4`
name1/?.name2 and name3/?.name4

`@mv` : Type `@mv name1 name2`, or `@mv name1 name2 name3)`, or `@mv name1 name2 name3 name4` to move the data-files dir1/b.xxx, dir1/s.xxx, and dir1/d.xxx, to dir2/b.yyy, dir2/s.yyy, and dir2/d.yyy, and copy the constants file dir1/c.xxx to dir2/c.yyy.

The values of dir1/?.xxx and dir2/?.yyy are determined in the same way as for `@cp` above.

`@dl` : Type `@dl xxx` to delete the data-files `b.xxx`, `s.xxx`, `d.xxx`.

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

`@ll` : Type `@ll` to list all solutions in fort.8.
Type `@ll xxx` to list all solutions in s.xxx.

`@lb` : Type `@lb` to run an interactive utility program for listing, deleting and relabeling solutions and branches in the 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 and branches in the data-files `b.xxx` and `s.xxx`. The original files are backed up as `b.xxx`∼ and `s.xxx`∼.

- Type `@lb xxx yyy` to list, delete and relabel solutions in the data-files `b.xxx` and `s.xxx`. The modified files are written as `b.yyy` and `s.yyy`.

2 Output Files

AUTO writes to standard output and three output-files :

- standard output: A summary of the computation is written to standard output, which usually corresponds to the window in which AUTO is run. Only special, labeled solution points are noted 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.

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 family; normal termination
MX	(-9)	Abnormal termination; no convergence

Table 1: Solution Types.

- `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`.

3 Description of AUTO constants

If the equations-file is `xxx.f90` then the constants that define the computation are normally expected in the file `c.xxx`. The format of this file is free, with constant=value pairs separated by commas or spaces. Comments start with one of the characters "#" and "!", and run to the end of a line. An example, with default values, is listed below. In real constant files you only need to specify those values that are different from these, but listing all of them allows for easier editing.

```
# Default AUTO Constants file
e = '', s='', dat='', sv=''
unames = {}, parnames = {}
U = {}, PAR = {}
NDIM=   2, IPS =   1, IRS =   0, ILP =   1
ICP =  [1]
NTST=  20, NCOL=   4, IAD =   3, ISP =   2, ISW = 1, IPLT= 0, NBC= 0, NINT= 0
NMX=    0, NPR=    0, MXBF=  10, IID =   2, ITMX= 9, ITNW= 5, NWTN= 3, JAC= 0
EPSL= 1e-07, EPSU = 1e-07, EPSS = 1e-05
DS  =  0.01, DSMIN= 0.005, DSMAX=   0.1, IADS=   1
NPAR=  36, THL =  {}, THU =  {}
RL0=-1.7976e+308, RL1=1.7976e+308, A0=-1.7976e+308, A1=1.7976e+308,
UZR = {}, UZSTOP = {}, SP = [], STOP = []
IIS = 3, IBR=0, LAB=0, TY=''
NUNSTAB = -1, NSTAB = -1, IEQUIB = 1, ITWIST = 0, ISTART = 5
IREV = [], IFIXED = [], IPSI = []
```

3.1 Problem Constants.

`NDIM:` Dimension of the system of equations as specified in the user-supplied routine `FUNC`.

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

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

`NPAR:` Maximum parameter number that can be used in all user-supplied routines.

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 routines `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. (Demos `int`, `dd2`, `obt`, `plp`, `ops`.)
- `JAC=-1` : As for `JAC=1`, but derivatives with respect to problem-parameters may be omitted in `FUNC`. (Demo `san`.)

3.2 Discretization Constants.

`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`.) Recommended value of `NTST` : As small as possible to maintain convergence.
(Demos `exp`, `ab`, `spb`.)

`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.

`IAD:` This constant controls the mesh adaption :

- `IAD=0` : Fixed mesh. Normally, 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 family. Most demos use `IAD=3`, which is the strongly recommended value.

When computing "trivial" solutions to a boundary value problem, for example, when all solution components are constant, then the mesh adaption may fail under certain circumstances, and overflow may occur. In such case, try recomputing the solution family with a fixed mesh `(IAD=0)`. Be sure to set `IAD` back to `IAD=3` for computing eventual non-trivial bifurcating solution families.

3.3 Tolerances.

`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.

`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.

`EPSS:` Relative arclength convergence criterion for the detection of 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`.

`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.

`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.

`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 family 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.

3.4 Continuation Step Size.

`DS`: AUTO uses pseudo-arclength continuation for following solution families. The pseudo-arclength stepsize is the distance between the current solution and the next solution on a family. 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 family. (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` ≤ | `DS` | ≤ `DSMAX` must be satisfied. The precise choice of `DS` is problem-dependent; the demos use a value that was found appropriate after some experimentation.

`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.

`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.

`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 family 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.

`THL`: 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, `THL` defines the parameters whose weight is to be modified. If `THL=`{} then all weights will have default value 1.0, else one must enter pairs, {*Parameter Index* : *Weight*, ...} .

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 `THL=`{`11:0.0`}. If `THL` is not specified this is the default for computing periodic solutions (`IPS=2`). Most demos that compute periodic solutions use this option; see for example demo `ab`.

`THU:` 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, `THU` defines the number of states whose weight is to be modified. If `THU`={} then all weights will have default value 1.0, else one must enter pairs, {*State Index* : *Weight*, ...} . At present none of the demos use this option.

3.5 Diagram Limits.

There are five ways to limit the computation of a family :

- By specifying a stopping condition in the list associated with the constant `STOP`.
- By specifying parameters and parameter values in the list associated with the constant `UZSTOP`.
- By specifying the maximum number of steps, `NMX`.
- By specifying a negative parameter index in the list associated with the constant `UZR`.
- 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. Most demos do not specify these constants, and use an unbounded window.

`STOP:` This constant adds stopping conditions. It is specified as a list of bifurcation type strings followed by a number n greater than zero. These strings specify that the contination should stop as soon as the nth bifurcation of the associated type has been reached. Example:

`STOP=['HB3','UZ3']` Stop at the third Hopf bifurcation or third user defined point, whichever comes first.

`NMX:` The maximum number of steps to be taken along any family.

`RL0` The lower bound on the principal continuation parameter. (This is the parameter which appears first in the `ICP` list.).

`RL1` The upper bound on the principal continuation parameter.

`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` for choosing another principal solution measure.)

`A1` The upper bound on the principal solution measure.

3.6 Free Parameters.

`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 array `ICP` 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.

Fixed points. The simplest case is the continuation of a solution family to the system $f(u, p) = 0$, where $f(\cdot, \cdot), u \in \mathrm{R}^n$. 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.

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

Periodic solutions and rotations. The continuation of periodic solutions and rotations generically requires two parameters, namely, one problem parameter and the period. For example, in Run 2 of demo `ab` we have `ICP=[1,11]`. 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, it is sufficient to only specify the index of the free problem parameter, as AUTO will automatically add `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 specify two free problem parameters. For example, in Run 7 of demo `pp2`, we have `ICP=[1,2]`, 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.

Folds and Hopf bifurcations. The continuation of folds for algebraic problems and the continuation of Hopf bifurcations requires two free problem parameters. For example, to continue a fold in Run 3 of demo `ab`, we have `ICP=[1,3]`, 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.

Folds and period-doublings. The continuation of folds, for periodic orbits and rotations, and the continuation of period-doubling and torus bifurcations require two free problem parameters plus the free period. Thus, one would normally specify three parameters. For example, in Run 6 of demo `pen`, where a locus of period-doubling bifurcations is computed for rotations, we have `ICP=[2,3,11]`, 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 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 `ICP=[4,1]`, 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, period-doubling or torus bifurcations with fixed period, simply specify three problem parameters, not including `PAR(11)`. For torus bifurcations it is also possible to specify four problem parameters (possibly including `PAR(11)`). In that case the angle of the torus (an internal parameter within AUTO) stays fixed.

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 family. For example, in demo `exp`, we have `NDIM`=`NBC`=2, `NINT`=0. Thus, 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 `ICP=[1,3]`. Indeed, in this demo two free parameters are designated, namely `PAR(1)` and `PAR(3)`.

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

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, `ICP=[10,2]` 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, `ICP=[10,2,3]` 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, `#ICP=4` in the third run.

For a simple example see demo `opt`, where a four-parameter extremum is located. Note that `#ICP=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.

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 puts the period, if not specified, in `PAR(11)`. 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 `NPAR`. Other use depends on `IPS`: see Section on Restrictions on Parameters.

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 routine `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 routine `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)`, `PAR(4)`, and `PAR(5)`, respectively. In the constants-file `c.pvl` we have `#ICP=5`, with `PAR(1)-PAR(5)` specified as parameters. Thus, in this example, `PAR(2)-PAR(5)` are overspecified. Note that `PAR(1)` must appear first in the `ICP` list; the other parameters cannot be used as true continuation parameters.

3.7 Computation Constants.

`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.

`SP`: This constant controls the detection of bifurcations and adds stopping conditions. It is specified as a list of bifurcation type strings followed by an optional number. If this number is 0, then the detection of this bifurcation is turned off, and if it is missing then the detection is turned on. A number n greater than zero specifies that the contination should stop as soon as the nth bifurcation of this type has been reached. Examples:

- `SP=['LP0']`
  turn off detection of folds.
- `SP=['LP','HB3','BP0','UZ3']`
  turn on the detection of folds and Hopf bifurcations, turn off detection of branch points and stop at the third Hopf bifurcation or third user defined point, whichever comes first.

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

- `ISP=0` : This setting disables the detection of Hopf bifurcations, branch points, period-doubling bifurcations, and torus bifurcations and the computation of Floquet multipliers.
- `ISP=1` : Branch points and Hopf bifurcations are detected for algebraic equations. Branch points, period-doubling bifurcations and torus bifurcations are not detected for periodic solutions and boundary value problems. 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`.
- `ISP=3` : Hopf bifurcations will not be detected. Branch points will be detected, and AUTO will monitor the Floquet multipliers. Period-doubling and torus bifurcations will go undetected. This option is useful for certain problems with non-generic Floquet behavior.
- `ISP=4` : Branch points and Hopf bifurcations are detected for algebraic equations. Branch points are not detected for periodic solutions and boundary value problems. AUTO will monitor the Floquet multipliers, and period-doubling and torus bifurcations will be detected.

`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, a period-doubling, a torus bifurcation, or, in a non-generic (symmetric) system, a branch point then a locus of such points will be computed. An additional free parameter must be specified for such continuations.
- ISW=3 : If IRS is the label of a branch point in a generic (non-symmetric) system then a locus of such points will be computed. Two additional free parameters must be specified for such continuations.

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 families will not be computed. If MXBF is positive then the bifurcating families of the first MXBF branch points will be traced out in both directions. If MXBF is negative then the bifurcating families of the first | MXBF | branch points will be traced out in only one direction.

s: This constant sets the name of the solution file from which the computation is to be restarted, instead of fort.3: if s='xxx' then the name of the restart file is s.xxx.

dat: This constant, where dat='xxx', sets the name of a user-supplied ASCII data file xxx.dat, from which the contination is to be restarted. 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 first column in the data file denotes the time, which does *not* need to be rescaled to the interval $[0, 1]$, and further columns the coordinates of the solution. The constant IRS must be set to 0.

(Demos lor, pen.)

U: This constant, where U={i1:x1,i2:x2}, changes the value of U(i1) to x1, U(i2) to x2, and so on, with respect to the solution to start from. This is only valid for restarting from algebraic problems or a constant-in-time solution.

PAR: This constant, where PAR={i1:x1,i2:x2}, changes the value of PAR(i1) to x1, PAR(i2) to x2, and so on, with respect to the solution to start from.

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 routine STPNT. For representative examples of analytical starting solutions see demos ab and frc. For starting from unlabeled numerical data see the dat command above, 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 s.xxx; see also the @r and @R commands. Various AUTO-constants can be modified when restarting.

- `IRS<0` : Restart the computation at the -IRSth computed solution in the restart file. This is especially useful if you do not want to look up label numbers and know for sure that the solution to continue from is at a fixed position.

- `IRS='XYn'` : Restart the computation at the nth label of type XY in the restart file, for instance 'HB12' to restart at the twelfth Hopf bifurcation.

`TY:` This constant modifies the type from the restart solution. This is sometimes useful in conservative or extended systems, declaring a regular point to be a Hopf bifurcation point (`TY='HB'`) or a branch point (`TY='BP'`). Use `TY='HB4'` to copy the period of the emanating periodic orbit from `PAR(4)` (for example set in the routine `PVLS` in the equations file) to `PAR(11)`. (Demo `r3b`.)

`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 routine `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 routine `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 routine `FOPT`. (Demo `opt`.)

- `IPS=7` : A boundary value problem with computation of Floquet multipliers. This is a very special option; for most boundary value problems one should use `IPS=4`. Boundary conditions must be specified in the user-supplied routine `BCND` and integral constraints in `ICND`. The AUTO-constants `NBC` and `NINT` must be given correct values.

- `IPS=9` : This option is used in connection with the HomCont algorithms described in the HomCont section 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 traveling wave bifurcations. The user need only define the nonlinearity (in routine `FUNC`), initialize the wave speed in `PAR(10)`, initialize the diffusion constants in `PAR(15,16,`$\cdots$`)`, 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,`$\cdots$`)`. 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 routine `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 routine `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`.

3.8 Output Control.

`unames:` This constant, where `unames=`{`i1:s1,i2:s2`}, changes the names in all output from `U(i1)` to `s1`, from `U(i2)` to `s2`, and so on. You can also refer to these strings, instead of the corresponding indices, in the constants `U` and `THU`.

`parnames:` This constant, where `parnames=`{`i1:s1,i2:s2`}, changes the names in all output from `PAR(i1)` to `s1`, from `PAR(i2)` to `s2`, and so on. You can also refer to these strings, instead of the corresponding indices, in the constants `ICP`, `THL`, and `UZR`.

`e:` This constant, where `e='xxx'`, is only for use by post-processors: it denotes the name of the equations file and is stored in the bifurcation diagram file (fort.7), so restarts in the Python interface are possible without needing to specify the equations file.

`sv:` This constant specifies a string to write the output to instead of `fort.7`, `fort.8`, and `fort.9`: if `sv='xxx'`, then the output files are `b.xxx`, `s.xxx`, and `d.xxx`.

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. For these reasons, and to limit the output volume, it is recommended that NPR output be kept to a minimum.

IBR: This constant specifies the initial branch number BR that is used. The default IBR=0 means that that this number is determined automatically.

LAB: This constant specifies the initial label number LAB that is used. The default LAB=0 means that that this number is determined automatically. Using LAB=1 means you do not need to relabel after a non-appended continuation if this is desired.

IIS: This constant controls the amount of information printed in fort.8 : the greater IIS the more solutions contain the corresponding vector giving the direction of the branch. The direction of the branch is necessary for restart points when switching branches, but make the solution file almost two times bigger than necessary when switching branches is never performed from solutions in this file.

- IIS=0 : The direction of the branch is never provided.
- IIS=1 : The direction of the branch is only provided at special points from which branch switching can be performed (types LP (boundary value problems only), BP, PD, TR).
- IIS=2 : The direction of the branch is provided at all special points but not at regular points without a type label.
- IIS=3 : The direction of the branch is always provided. This is the default setting.

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

- IID=0 : No diagnostic output.
- IID=1 : 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.

- `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. It gives incomplete results when used in combination with MPI parallellization.

`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. For algebraic problems, the standard definition of L_2-norm is used. For differential equations, the L_2-norm is defined as

$$\sqrt{\int_0^1 \sum_{k=1}^{NDIM} U_k(x)^2 \, dx} \ .$$

  Note that the interval of integration is $[0,1]$, the standard interval used by AUTO. For periodic solutions the independent variable is transformed to range from 0 to 1, before the norm is computed. The AUTO-constants THL(*) and THU(*) affect the definition of the L_2-norm.

- 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 routine `PVLS` provides a second, more general way of defining solution measures.

`UZR`: 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 `UZR=`{} if no such output is needed. Many demos use this setting.

- Else one must enter pairs, {*Parameter-Index* : *Parameter-Value*, ...} ,
  or indices with lists of values, {*Parameter-Index* :

$$\mathit{Parameter} - \mathit{Value}, ...$$

  , ...} , 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. See also `STOP` above and `UZSTOP` in below for alternative termination methods. (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`.

`UZSTOP`: This constant specifies parameter values in the same way as `UZR` above, but the computation will always terminate if any solution point that is specified is encountered.

3.9 Quick reference

`e`, `s`, `dat`, `sv`	Define file names: equation prefix (.f,.f90,.c), restart solution suffix (s.), user data prefix (.dat), output suffix (b.,s.,d.)
`unames`, `parnames`	Dictionary (mapping) of U(*) and PAR(*) to user-defined names
`NDIM`	Problem dimension
`IPS`	Problem type; 0=AE, 1=FP(ODEs), -1=FP(maps), 2=PO, -2=IVP, 4=BVP, 7=BVP with Floquet multipliers, 5=algebraic optimization problem, 15=optimization of periodic solutions
`IRS`, `TY`	Start solution label, start solution type
`ILP`	Fold detection; 1=on, 0=off
`ICP`	Continuation parameters
`NTST`	# mesh intervals
`NCOL`	# collocation points
`IAD`	Mesh adaption every IAD steps; 0=off
`ISP`	Bifurcation detection; 0=off, 1=BP(FP), 3=BP(PO,BVP), 2=all
`ISW`	Branch switching; 1=normal, -1=switch branch (BP, HB, PD), 2=switch to two-parameter continuation (LP, BP, HB, TR) 3=switch to three-parameter continuation (BP)
`IPLT`	Select principal solution measure
`NBC`	# boundary conditions
`NINT`	# integral conditions
`NMX`	Maximum number of steps
`RL0, RL1`	Parameter interval $RL0 \le \lambda \le RL1$
`A0, A1`	Interval of principal solution measure $A0 \le \|\cdot\| \le A1$
`NPR`	Print and save restart data every NPR steps
`MXBF`	Automatic branch switching for the first MXBF bifurcation points if IPS=0, 1
`IBR`, `LAB`	Set initial branch and label number; 0=automatic
`IIS`	Control solution output of branch direction vector; 0=never, 3=always
`IID`	Control diagnostic output; 0=none, 1=little, 2=normal, 4=extensive
`ITMX`	Maximum # of iterations for locating special solutions/points
`ITNW`	Maximum # of correction steps
`NWTN`	Corrector uses full newton for NWTN steps
`JAC`	User defines derivatives; 0=no, 1=yes
`EPSL`, `EPSU`, `EPSS`	Convergence criterion: parameters, solution components, special points
`DS`	Start step size
`DSMIN, DSMAX`	Step size interval `DSMIN` $\le h \le$ `DSMAX`
`IADS`	Step size adaption every IADS steps; 0=off
`NPAR`	Maximum number of parameters
`THL, THU`	list of parameter and solution weights
`UZR`, `UZSTOP`	list of values for user defined output
`SP`, `STOP`	list of bifurcations to check and bifurcation stop conditions
`NUNSTAB`, `NSTAB`	HomCont: unstable and stable manifold dimensions
`IEQUIB`, `ITWIST`, `ISTART`	HomCont: control solution types adjoint, starting
`IREV`, `IFIXED`, `IPSI`	HomCont: control reversibility, fixed parameters, test functions