URL: http://cxc.harvard.edu/sherpa3.4/methods/methods.html
Last modified: 11 January 2007

Available Optimization Methods

Return to: Optimization Methods Index

The primary task of *Sherpa* is to fit a model $M(\vec{p})$ to a set of observed data, where $\vec{p}$ denotes the model parameters. An *optimization method* is one that is used to determine the vector of model parameter values, $\vec{p}_o$, for which the chosen fit statistic is minimized.

Optimization is complicated because there is no guarantee that, for a given model and dataset, there will not be multiple local fit−statistic minima. Thus *Sherpa* provides a number of optimization methods, each of which has different advantages and disadvantages. The default optimization method is `POWELL`, which is a robust method for finding the nearest local fit−statistic minimum (but not necessarily the global minimum). The user can change the method using the `METHOD` command.

The Advice on Minimization Methods page provides a fuller discussion of the relative merits of each of the different optimization methods.

The following optimization methods are available in *Sherpa*:

- GRID
- GRID−POWELL
- LEVENBERG−MARQUARDT | LEV−MAR
- MONTECARLO
- MONTE−LM
- MONTE−POWELL
- POWELL
- SIGMA−REJECTION | SIG−REJ
- SIMPLEX
- SIMUL−ANN−1
- SIMUL−ANN−2
- SIMUL−POW−1
- SIMUL−POW−2
- USERMETHOD

GRID

A grid search of parameter space, with no minimization.

The `GRID` method samples the parameter space bounded by the lower and upper limits for each *thawed* parameter. At each grid point, the fit statistic is evaluated. The advantage of `GRID` is that it can provide a thorough sampling of parameter space. This is good for situations where the best−fit parameter values are not easily guessed *a priori*, and where there is a high probability that false minima would be found if one−shot techniques such as `POWELL` are used instead. Its disadvantages are that it can be slow, and that because of the discrete nature of the search, the global fit−statistic minimum can easily be missed. (The latter disadvantage may be alleviated by combining a grid search with Powell minimization; see `GRID−POWELL`.)

The user should change the parameter `grid.totdim` so that its value matches the number of thawed parameters in the fit. When this change is made, the total number of `GRID` parameters changes, e.g. if the user first changes `grid.totdim` to 6, the parameters `grid.nloop05` and `grid.nloop06` will appear, each with default values of 10.

If the *Sherpa* command `FIT` is given without changing `grid.totdim` to match the number of thawed parameters, *Sherpa* will make the change automatically. However, one cannot change the value of any newly created `grid.nloopi` parameters until the fit is complete.

If one is interested in running the grid only over a subset of the thawed parameter space, there are two options: first, one can freeze the unimportant parameters at specified values, or second, one can set `grid.nloopi` for each of the unimportant parameters to 1, while also specifying their (constant) values.

The `GRID` algorithm uses the specified minimum and maximum values for each thawed parameter to determine the grid points at which the fit statistic is to be evaluated. If `grid.nloopi` = 1, the grid point is assumed to be the current value of the associated parameter, as indicated above. Otherwise, the grid points for parameter x_i are given by

$$x_{i,\min}\ ,\ x_{i,\min} + \frac{(x_{i,\max} - x_{i,\min})}{(\mathbf{grid.nloopi} - 1)}\ ,\ x_{i,\min} + 2\frac{(x_{i,\max} - x_{i,\min})}{(\mathbf{grid.nloopi} - 1)}\ ,\ \cdots\ ,\ x_{i,\max}$$

Note that the current parameter value will not be sampled unless the minimum and maximum values, and the number of grid points, are chosen appropriately.

The maximum number of grid points that may be sampled during one fit is 1.e+7. The total number of grid points is found by taking the product of all values of `grid.nloopi`.

Parameters

Number	Name	Default	Min	Max	Description
totdim	integer	4	1	24	Number of free parameters.
nloopi	integer	10	1	1.e+7	Number of grid points along axis *i*. The number of `nloopi` parameters which the user will actually see after issuing the command `SHOW GRID` is equal to `totdim`; the parameter names are `nloop01`, `nloop02`, ...

GRID−POWELL

A grid search utilizing the Powell method at each grid point.

The `GRID−POWELL` method samples the parameter space bounded by the lower and upper limits for each thawed parameter. At each grid point, the `POWELL` optimization method is used to determine the local fit−statistic minimum. The smallest of all observed minima is then adopted as the global fit−statistic minimum. The advantage of `GRID−POWELL` is that it can provide a thorough sampling of parameter space. This is good for situations where the best−fit parameter values are not easily guessed *a priori*, and where there is a high probability that false minima would be found if one−shot techniques such as `POWELL` are used instead. Its disadvantage are that it can be very slow.

Note that `GRID−POWELL` is similar in nature to `MONTE−POWELL`; in the latter method, the initial parameter value guesses in each cycle are chosen randomly, rather than being determined from a grid.

The `GRID−POWELL` method parameters are a superset of those for the `GRID` and `POWELL` methods.

LEVENBERG−MARQUARDT | LEV−MAR

The Levenberg−Marquardt optimization method.

An abbreviated equivalent is `LEV−MAR`.

The `LEVENBERG−MARQUARDT` method is a single−shot method which attempts to find the local fit−statistic minimum nearest to the starting point. Its principal advantage is that it uses information about the first and second derivatives of the fit−statistic as a function of the thawed parameter values to guess the location of the fit−statistic minimum. Thus this method works well (and fast) if the statistic surface is well−behaved. Its principal disadvantages are that it will not work as well with pathological statistic surfaces, and there is no guarantee it will find the global fit−statistic minimum.

The code for this method is derived from the implementation in Bevington (1992).

The `eps` parameter controls when the optimization will cease; for `LEVENBERG−MARQUARDT`, this will occur when

$$|S_i - S_{i-1}| < \mathbf{eps}\ ,$$

where S_{i-1} and S_i are the observed statistic values for the $(i-1)$ th and i th iteration, respectively.

The `smplx` parameter controls whether the `LEVENBERG−MARQUARDT` fit is refined with a `SIMPLEX` fit. `SIMPLEX` refinement can be useful for complicated fitting problems where straight `LEVENBERG−MARQUARDT` does not provide a quick solution. Switchover from `LEVENBERG−MARQUARDT` to `SIMPLEX` occurs when ”*S*, the change in statistic value from one iteration to the next, is less than `LEVENBERG−MARQUARDT.smplxep`, for `LEVENBERG−MARQUARDT.smplxit` iterations in a row. For example, the default is for switchover to occur when ”Ç^2 < 1 for 3 iterations in a row.

Parameters

Number	Name	Default	Min	Max	Description
iters	integer	2000	1	10000	Maximum number of iterations.
eps	real	1.e−3	1.e−9	1	Criterion to stop fit.
smplx	real	1	0	1	Refine fit with simplex (0=no)
smplxep	real	1	0.0001	1000	Switch−to−simplex eps factor
smplxit	real	3	1	20	Switch−to−simplex iters factor

MONTECARLO

A Monte Carlo search of parameter space.

The `MONTECARLO` method randomly samples the parameter space bounded by the lower and upper limits of each *thawed* parameter. At each chosen point, the fit statistic is evaluated. The advantage of `MONTECARLO` is that it can provide a good sampling of parameter space. This is good for situations where the best−fit parameter values are not easily guessed *a priori*, and where there is a high probability that false minima would be found if one−shot techniques such as `POWELL` are used instead. Its disadvantages are that it can be slow (if many points are selected), and that because of the random, discrete nature of the search, the global fit−statistic minimum can easily be missed. (The latter disadvantage may be alleviated by combining a Monte Carlo search with Powell minimization; see `MONTE−POWELL`.)

If the number of thawed parameters is larger than 3, one should increase the value of `nloop` from its default value. Otherwise the sampling may be too sparse to estimate the global fit−statistic minimum well.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	500	1	1.6777e+7	Number of parameter space samples.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.

MONTE−LM

A Monte Carlo search utilizing the Powell method at each selected point.

The `MONTE−LM` method randomly samples the parameter space bounded by the lower and upper limits for each *thawed* parameter. At each grid point, the `LEVENBERG−MARQUARDT` optimization method is used to determine the local fit−statistic minimum. The smallest of all observed minima is then adopted as the global fit−statistic minimum. The advantage of `MONTE−LM` is that it can provide a good sampling of parameter space. This is good for situations where the best−fit parameter values are not easily guessed *a priori*, and where there is a high probability that false minima would be found if one−shot techniques such as `LEVENBERG−MARQUARDT` are used instead. Its disadvantage is that it can be slow.

The `MONTE−LM` method parameters are a superset of those listed for the `LEVENBERG−MARQUARDT` method and the ones listed below.

If the number of thawed parameters is larger than 2, one should increase the value of `nloop` from its default value. Otherwise the sampling may be too sparse to estimate the global fit−statistic minimum well.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	128	1	16384	Number of parameter space samples.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.

MONTE−POWELL

A Monte Carlo search utilizing the Powell method at each selected point.

The `MONTE−POWELL` method randomly samples the parameter space bounded by the lower and upper limits for each *thawed* parameter. At each grid point, the `POWELL` optimization method is used to determine the local fit−statistic minimum. The smallest of all observed minima is then adopted as the global fit−statistic minimum. The advantage of `MONTE−POWELL` is that it can provide a good sampling of parameter space. This is good for situations where the best−fit parameter values are not easily guessed *a priori*, and where there is a high probability that false minima would be found if one−shot techniques such as `POWELL` are used instead. Its disadvantage is that it can be very slow.

Note that `MONTE−POWELL` is similar in nature to `GRID−POWELL`; in the latter method, the initial parameter values in each cycle are determined from a grid, rather than being chosen randomly.

The `MONTE−POWELL` method parameters are a superset of those listed for the `POWELL` method and the ones listed below.

If the number of thawed parameters is larger than 2, one should increase the value of `nloop` from its default value. Otherwise the sampling may be too sparse to estimate the global fit−statistic minimum well.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	128	1	16384	Number of parameter space samples.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.

POWELL

The Powell optimization method.

The `POWELL` method is a single−shot method which attempts to find the local fit−statistic minimum nearest to the starting point. Its principal advantage is that it is a robust *direction−set* method. A set of directions (e.g., unit vectors) are defined; the method moves along one direction until a minimum is reached, then from there moves along the next direction until a minimum is reached, and so on, cycling through the whole set of directions until the fit statistic is minimized for a particular iteration. The set of directions is then updated and the algorithm proceeds. Its principal disadvantages are that it will not find the local minimum as quickly as `LEVENBERG−MARQUARDT` if the statistic surface is well−behaved, and there is no guarantee it will find the global fit−statistic minimum.

The `eps` parameter controls when the optimization will cease; for `POWELL`, this will occur when

$$|S_i - S_{i-1}| < \frac{1}{2} * \mathbf{eps} * (|S_i| + |S_{i-1}|) \,,$$

where S_{i-1} and S_i are the observed statistic values for the $(i-1)^{\text{th}}$ and i^{th} iteration, respectively.

Parameters

Number	Name	Default	Min	Max	Description
iters	integer	2000	1	10000	Maximum number of iterations.
eps	real	1.e−6	1.e−9	0.001	Criterion to stop fit.
tol	real	1.e−6	1.e−8	0.1	Tolerance in lnmnop
huge	real	1.e+10	1000	1.e+12	Vestigial.

SIGMA−REJECTION | SIG−REJ

The SIGMA−REJECTION optimization method for fits to 1−D data.

Abbreviated equivalents are SIG−REJ and SR.

The `SIGMA−REJECTION` optimization method is based upon the IRAF method `SFIT`. It iteratively fits data: a best−fit to the data is determined using a "regular" optimization method (e.g. `LEVENBERG−MARQUARDT`), then outliers data points are filtered out of the dataset, and the data refit, etc. Iterations cease when there is no change in the filter from one iteration to the next, or when the fit has iterated a user−specified maximum number of times.

Parameters

Number	Name	Default	Min	Max	Description
niter	integer	5	0	128	The maximum number of iterations in the fit; if 0, the fit will run to convergence (i.e., until there is no change in the filter).
lrej	real	3	0	10	Data point rejection criterion in units of Ã, for data points *below* the model (e.g., absorption line data); using `lrej` is equivalent to doing `LPLOT DELCHI` and filtering out all data points lying below *y*=−lrej on that plot.
hrej	real	3	0	10	Data point rejection criterion in units of Ã, for data points *above* the model (e.g., absorption line data); using `hrej` is equivalent to doing `LPLOT DELCHI` and filtering out all data points lying above *y*=hrej on that plot.
grow	integer	0	0	1024	When a given data point is to be filtered out, this parameter sets the number of pixels adjacent to that pixel which are also to be filtered out, i.e., if 0, only the data point itself is filtered out; if 1, the data point and its two immediate neighbors are filtered out, etc.
omethod	string	lm			The optimization method to use to perform fits at each iteration. Note that quotes are required around the name!

SIMPLEX

A simplex optimization method.

The `SIMPLEX` method is a single−shot method which attempts to find the local fit−statistic minimum nearest to the starting point. Its principal advantage is that it *can* work well with complicated statistic surfaces (more so than `LEVENBERG−MARQUARDT`), while also working quickly (more so than `POWELL`). Its principal disadvantages are that it has a tendency to "get stuck" in regions with complicated topology before reaching the local fit−statistic minimum, and that there is no guarantee it will find the global fit−statistic minimum. Its tendency to stick means that the user may be best−served by repeating fits until the best−fit point does not change.

A simplex is geometrical form in *N*−dimensional in parameter space which has *N* + 1 vertices (e.g., in 3−D it is a tetrahedron). The fit statistic is evaluated for each vertex, and one or more points of the simplex are moved, so that the simplex moves towards the nearest local fit−statistic minimum. When a minimum is reached, the simplex may also contract itself, as an amoeba might; hence, the routine is also sometimes called "amoeba." Convergence is reached when the simplex settles into a minimum and all the vertices are within some value `eps` of each other.

The `eps` parameter controls when the optimization will cease; for `SIMPLEX`, this will occur when

$$|S_i - S_{i-1}| < \mathbf{eps}\,,$$

where S_{i-1} and S_i are the observed statistic values for the $(i-1)^{\text{th}}$ and i^{th} iteration, respectively.

Parameters

Number	Name	Default	Min	Max	Description
iters	integer	2000	1	10000	Maximum number of iterations.
eps	real	1.e−6	1.e−9	0.001	Criterion to stop fit.
alpha	real	1	0.1	2	Algorithm convergence factor.
beta	real	0.5	0.05	1	Algorithm convergence factor.
gamma	real	2	1.1	20	Algorithm convergence factor.

SIMUL−ANN−1

A simulated annealing search, with one parameter varied at each step.

The `SIMUL−ANN−1` method is a simulated annealing method. Such a method is said to be useful when a desired global minimum is among many other local minima. It has been derived via an analogy to how metals cool and anneal; as the liquid metal is cooled slowly, the atoms are able to settle into their minimum energy state, i.e. form a crystal. In simulated annealing, the function to be optimized is held to be the analog of energy, and the system slowly "cooled." At each temperature tried, the parameters are randomly varied; when there is no further improvement, the temperature is lowered, and again the parameters are varied some number of times. With sufficiently slow cooling, and sufficient random tries, the algorithm "freezes" near the global minimum. (At each randomization, only one parameter is varied.) This is quite different from the other techniques implemented here, which can be said to head downhill, fast, for a nearby minimum. The great advantage of simulated annealing is that it sometimes goes uphill, thus escaping undesirable local minima the other techniques tend to get stuck in. The great disadvantage, of course, is that it is a slow technique.

A simple view of simulated annealing is that a new guess at the location of the minimum of the objective function is acceptable if it causes the value of the function to decrease, and also with probability

$$P = \exp\left(-\frac{\Delta f}{T}\right)$$

where ”*f* is the *increase* in the value of the objective function, and T is a "temperature." After a large number of random points in parameter space have been sampled, and no improvement in the minimum value of the objective function has occurred, the temperature is decreased (increasing the probability penalty for positive ”*f*, and the process is repeated. The expectation is that with a sufficient number of sampled points at each temperature, and a sufficiently slow "cooling" of the system, the result will "freeze out" at the true minimum. The analogy is supposed to be with a cooling gas, where the Boltzmann statistics of molecular motions causes the molecules of the gas to explore both the minimum energy state and adjacent energy states at any one temperature, and then freeze in a crystalline state of minimum energy as the temperature approaches zero.

`SIMUL−ANN−1` starts with a temperature which is about twice the initial value of the objective function, and reduces it slowly towards zero, changing a single parameter value at each randomization.

Parameter `tchn` is a multiplying factor for the temperature variable between cooling steps. Its default value of 0.98 means that a large number of cooling steps is needed to reduce the temperature sufficiently that the function "freezes out," but fast cooling is not recommended (as it inhibits the ergodic occupation of the search space). Parameter `nloop` specifies the maximum number of temperature values to try, but may be overridden if the code finds that the solution has "frozen out" early. Finally, parameter `nsamp` specifies the number of movements at each temperature, i.e. the number of random changes to the parameter values that will be tried.

`SIMUL−ANN−1` tends to be slow, but is a possible way of making significant progress in traveling−salesman type problems. One of its strengths is that even if it doesn't find the absolutely best solution, it often converges to a solution that is close (in the ”*f* sense) to the true minimum solution. Somewhat better results are often obtained by `SIMUL−POW−1` and `SIMUL−POW−2`, where the simulated annealing method is combined with the Powell method.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	512	32	8192	Maximum number of temperatures.
tchn	real	0.98	0.1	0.9999	Factor for temperature reduction.
nsamp	integer	256	16	4096	Number of movements at each temperature.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.
tiny	real	1.e−12	1.e−20	1.e−6	Smallest temperature allowed.

SIMUL−ANN−2

A simulated annealing search, with all parameters varied at each step.

The `SIMUL−ANN−2` method is a simulated annealing method, analogous to `SIMUL−ANN−1` in every way except that at each randomization, all the parameters are varied, not just one.

Note that the default temperature cycle change factor, and the number of temperature cycles, correspond to more and faster changes of temperature than `SIMUL−ANN−1`. This is because of the increased mobility within parameter space, since all values are changed with each randomization.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	1024	64	16384	Maximum number of temperatures.
tchn	real	0.95	0.1	0.9999	Factor for temperature reduction.
nsamp	integer	512	32	8192	Number of movements at each temperature.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.
tiny	real	1.e−12	1.e−20	1.e−6	Smallest temperature allowed.

SIMUL−POW−1

A combination of `SIMUL−ANN−1` with `POWELL`.

This method packages together `SIMUL−ANN−1` and the `POWELL` routine; at the end of each of the cooling sequences, or annealing cycles, the `POWELL` method is invoked. Probably one of the best choices where one "best" answer is to be found, but at the expense of a lot of computer time.

Note that the parameters of `SIMUL−POW−1` are those of `SIMUL−POW−1` itself (which have the same meaning as in routine `SIMUL−ANN−1`), *plus* those of `POWELL`.

All of the `SIMUL−POW−1` parameters, with the exception of `nanne`, are explained under `SIMUL−ANN−1`. Parameter `nanne` specifies the number of annealing cycles to be used; successive anneal cycles start from cooler temperatures and have slower cooling. The pattern of anneals that has been chosen (the "annealing history") is not magic in any way. A different annealing history may be better for any particular objective function, but the pattern chosen seems to serve well.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	256	16	4096	Maximum number of temperatures.
tchn	real	0.95	0.1	0.9999	Factor for temperature reduction.
nanne	integer	16	1	256	Number of anneals.
nsamp	integer	128	16	1024	Number of movements at each temperature.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.
tiny	real	1.e−12	1.e−20	1.e−6	Smallest temperature allowed.

SIMUL−POW−2

A combination of SIMUL−ANN−2 with POWELL.

This method packages together `SIMUL−ANN−2` and the `POWELL` routine; at the end of each of the cooling sequences, or annealing cycles, the `POWELL` method is invoked. The rate of cooling in each anneal loop may be much faster than in simulated annealing alone. Probably one of the best choices where one "best" answer is to be found, but at the expense of a lot of computer time.

Note that the parameters of `SIMUL−POW−2` are those of `SIMUL−POW−2` itself (which have the same meaning as in routine `SIMUL−ANN−2`), *plus* those of `POWELL`.

All of the `SIMUL−POW−2` parameters, with the exception of `nanne`, are explained under `SIMUL−ANN−2`. Parameter `nanne` specifies the number of annealing cycles to be used; successive anneal cycles start from cooler temperatures and have slower cooling. The pattern of anneals that has been chosen (the "annealing history") is not magic in any way. A different annealing history may be better for any particular objective function, but the pattern chosen seems to serve well.

Parameters

Number	Name	Default	Min	Max	Description
nloop	integer	256	16	4096	Maximum number of temperatures.
tchn	real	0.95	0.1	0.9999	Factor for temperature reduction.
nanne	integer	16	1	256	Number of anneals.
nsamp	integer	128	16	1024	Number of movements at each temperature.
iseed	integer	14391	−1.e+20	1.e+20	Seed for random number generator.
tiny	real	1.e−12	1.e−20	1.e−6	Smallest temperature allowed.

USERMETHOD

A user−defined method.

It is possible for the user to create and implement his or her own model, own optimization method, and own statistic function within *Sherpa*. The User Models, Statistics, and Methods Within *Sherpa* chapter of the *Sherpa* Reference Manual has more information on this topic.

The tarfile `sherpa_user.tar.gz` contains the files needed to define the `usermethod`, e.g Makefiles and Implementation files, plus example files, and it is available from the *Sherpa* threads page: Data for *Sherpa* Threads.