LA-UR-15-21424
Approved for public release; distribution is unlimited.

Title: Four-Dimensional Golden Search

Author(s): Fenimore, Edward E.

Intended for: Report

Issued: 2015-02-25

Four-Dimensional Golden Search

Ed Fenimore

National Security Education Center

LA-UR 15-xxxxx

Abstract

The Golden search technique is a method to search a multiple-dimension space to find the minimum. It basically subdivides the possible ranges of parameters until it brackets, to within an arbitrarily small distance, the minimum. It has the advantages that (1) the function to be minimized can be non-linear, (2) it does not require derivatives of the function, (3) the convergence criterion does not depend on the magnitude of the function. Thus, if the function is a goodness of fit parameter such as chi-square, the convergence does not depend on the noise being correctly estimated or the function correctly following the chi-square statistic. And, (4) the convergence criterion does not depend on the shape of the function. Thus, long shallow surfaces can be searched without the problem of premature convergence. As with many methods, the Golden search technique can be confused by surfaces with multiple minima.

The Golden Search package uses a nested set of routines, one to do each parameter, up to four dimensions. It is based on a 1-Dimensional Golden Search described in Numerical Recipes (Press 1986). The technique takes a range (say, *a* to *c*) and picks a point between them (say, *b*). Under the assumption that the minimum is between *a* and *c*, the function evaluated at *b* (i. e., *f(b)*) will be less than *f(a)* and *f(c)*. Thus, *a* and *c* brackets the minimum. The point *b* is selected to split the *a* to *c* range into two segments that have relative lengths of 0.618 and 0.382 (the Golden Ratio). A new point (*d*) is selected (also following the Golden Ratio) within the range *a* to *b* (if *f(a) > f(c)*) or *b* to *c* (if *f(a) < f(c)*). The three points with the smallest function evaluations become the new a, b, and c and the process is repeated until the distance between *a* and *c* is less than a specified convergence value.

Using the Golden Ratio is somewhat more efficient than a bisection method that splits each range into half, then quarters, the eights, etc.

Assume that the function to minimize is similar to chi-square. That is, we seek to minimize

$$\chi^2 = \sum_N \frac{(O_i - M(i, P_1, P_2, P_3, \ldots P_m)^2}{\sigma_i^2} \quad \text{Eq 1}$$

where there are *N* observations (O_i) with uncertainties of σ_i. They are fit with a model (*M*) that is defined by *m* parameters.

There are three types of parameters: (1) parameters whose values are known, (2) parameters whose values are not known but linearly scale the model, and (3) parameters whose values are

not known and their range must be searched to find the value that gives the lowest chi-square. For example, the model might have the form

$$M(i,P_1,P_2,P_3,...P_m) = P_1 M_1(i,P_2,P_3,P_4,P_5) + P_6 M_2(i,P_3,P_4,P_7,P_8) \quad \text{Eq } 2$$

There are 8 parameters. Parameters P_1 and P_6 linearly scale the model. Assume parameters P_2 and P_3 have values which are set and known while the parameters P_4, P_5, P_7, and P_8 need to be searched to find the values that minimize chi-square.

When finding the best fit parameters, it is crucial that one solves analytically for any parameter that linearly scales the model. This reduces the dimensionality of the search which can drastically reduce computational time. However, the more important reason to solve analytically for linear parameters is that chi-square is extremely sensitive to linear parameters. Linear parameters balance the fit so that there are some points above the fit function and some points below the fit function. That must be true when chi-square is at a minimum. Attempting to search for the best value of a linear parameter (rather than solving for it analytically) can be unstable because chi-square changes very rapidly for linear parameters. Methods that depend on gradients (such as Levenberg-Marquardt) have a hard time correctly estimating the gradients because the function is so strong.

Thus, the best strategy for finding the minimum of Equation 1 is to identify the parameters that are linear scaling factors and those that are not. In the example of equation 2, P_1 and P_6 are linear scaling factors. The chi-square minimum is where

$$\frac{\partial \chi^2}{\partial P_1} = 0 \quad \text{Eq } 3$$

and

$$\frac{\partial \chi^2}{\partial P_2} = 0 \quad \text{Eq } 4 .$$

From equation 3 and 4, one can make two equations with two unknowns for P_1 and P_6:

$$\sum \frac{O_i M_1}{\sigma_i^2} = P_1 \sum \frac{M_1^2}{\sigma_i^2} + P_6 \sum \frac{M_1 M_2}{\sigma_i^2} \quad \text{Eq } 5$$

$$\sum \frac{O_i M_2}{\sigma_i^2} = P_1 \sum \frac{M_1 M_2}{\sigma_i^2} + P_6 \sum \frac{M_2^2}{\sigma_i^2} \quad \text{Eq 6}$$

In the example of equation 2, the values of parameters P_2 and P_3 are known. Thus, the chi-square minimization needs to search the four-dimensional space of P_4, P_5, P_7, and P_8. For each point evaluated in the four-dimensional space, equations 5 and 6 are first solved to find P_1 and P_6 and then equation 2 is used in equation 1.

The Golden Search package consists of four routines (golden1d_ds, golden2d_ds, golden3d_ds, and golden4d_ds). The "1d", "2d", "3d", and "4d" indicates what dimension it does. So, if the problem involves three searchable parameters, the user would call golden3d_ds. Golden3d_ds will call golden2d_ds which will call golden1d_ds which will call fun_to_min. The user needs to only add the routine fun_to_min which calculates chi-square.

All of the golden*d_ds routines have the same interface (case insensitive):

Variable	Type	Size	I//O	Description
Fun_to_min				Subroutine for chi-square, should be in an external statement. The spelling of the name of the routine does not have to be fun_to_min. Use the appropriate spelling when calling the Golden*d_ds routine.
chisq	Real*8	1	output	The best fit chi-square
Max_golden_parameter = M	int	1	input	Dimension of arrays, must be as least as large as the number of parameters
Golden_para	Real*8	M	Input and output	The set of parameters that give the best chi-square. Some are set by the code that calls golden*d_ds, some could be linear scale factors found by fun_to_min, some are set by the nested golden*d_ds routines
Golden_para_min	Real*8	M	input	The start of the search range. Only those elements that occur in ivar_place_golden are used.
Golden_para_max	Real*8	M	input	The end of the search range. Only those elements that occur in ivar_place_golden are used.
Data_structure			Input and ouput	A data structure that will be sent to the fun_to_min routine. The first element of the data structure should be an integer.
Golden_chisq	Real*8	M		A scratch area used to communicate best chi-square between search dimensions

Hit_boundary	int	M	output	Is 0 if the golden_para is not at either golden_para_min or golden_para_min. It is 1 if the golden_para is near golden_para_min and it is equal to 2 if golden_para is near golden_para_max. Only those elements that occur in ivar_place_golden are used.
Ivar_place_golden	int	4	input	For an N dimensional search, the first N elements define which parameter is searched. See below.
Max_golden_iterations	int	M	input	The maximum number of steps a search can take for a parameter before an error is declared. Only those elements that occur in ivar_place_golden are used.
Num_total_evaluation	int	4	output	For an N dimensional search, the first N elements give how many calls to the next lower level were made.
Golden_tol	Real*8	M	input	The accuracy to determine a searched parameter. Only those elements that occur in ivar_place_golden are used.

Where M is max_golden_parameter. Note ivar_place_golden and num_total_evaluation need only to be dimensioned 4 because the largest dimension the package can search is 4. We often dimension it with max_golden_parameter.

The interface to fun_to_min must be:

Variable	Type	Size	I//O	Description
chisq	Real*8	1	output	The chi-square for the specified parameters
Max_golden_parameter = M	int	1	input	Dimension of arrays, must be as least as large as the number of parameters
Golden_para	Real*8	M	Input and output	The set of parameters to calculate chi-square at. Some could be linear scale factors found by fun_to_min.
Data_structure			Input and ouput	A data structure that contain information to calculate chi-square. The first element of the data structure should be an integer.

First we will describe how to set up the call to the Golden Search package and then describe the output.

To set up the case in equation 2, one calls golden4d_ds with:

(a) Max_golden_parameter needs to be set to at least 8 since we have 8 parameters in equation 2
(b) According to the description of equation 2, golden_para(2) and golden_para(3) must be set by the routine that calls the top level golden*d_ds routine.
(c) Ivar_place_golden tells which parameter will be searched at which dimension. In this case, we will do a 4 dimensional search. Ivar_place_golden(1) gives what parameter will be searched by the inner most search, that is, golden1d_ds. Ivar_place_golden(2) gives what parameter will be searched by golden2d_ds. Ivar_place_golden(3) gives what parameter will be searched by golden3d_ds and ivar_place_golden(4) gives what parameter will be searched by golden4d_ds. If we want the inner most search to be parameter 7, the next search to be parameter 5, the next search to be parameter 8, and the outermost search to be parameter 4, we set ivar_place_golden(1) to (4) to be 7, 5, 8, and 4. The innermost search will call fun_to_min.
(d) Parameters 4, 5, 7, and 8 will be searched. Thus we must set the 4th, 5th, 7th, and 8^{th} elements of golden_para_min and golden_para_max. For each, golden_para_min and golden_para_max is the range that will be searched and should be made large enough to include the suspected answer.
(e) The data_structure is a method to transmit to fun_to_min all the information needed to calculate χ^2. This includes, for example, N, O_i and σ_i.
(f) Max_golden_iterations is a safety feature to prevent infinite loops. Since the search scales roughly as 2^n where n is the number of iterations, setting max_golden_iterations to, say, 50 is very safe. Max_golden_iterations needs to be set for those elements which are being searched, that is, the 4^{th}, 5^{th}, 7^{th}, and 8^{th} elements.
(g) Golden_tol is how accurately we want to know each of the search parameters. For example, parameter 4 might be a time that we want to determine to within 0.001 seconds so we set golden_tol(4)=0.0001. Parameter 5 might be the water vapor that we want to determine to within 0.01, so golden_tol(5) = 0.01. And so on for golden_tol(7) and golden_tol(8).

Fun_to_min needs to:

(a) Solve equations 5 and 6 to set golden_para(1) and golden_para(6).
(b) Use golden_para(1) to golden_para(8) with equation 1 and 2 to set chisq.
(c) If useful, set diagnostic information in data_structure.

After golden4d_ds finishes:

(a) One should check if an error condition happened by checking if error_exist is true.

(b) Chisq will be the chi-square at the best fit set of parameters.
(c) Golden_para will be the best fit set of parameters
(d) Hit_boundary should be checked to see if the best fit parameters were near either golden_para_min or golden_para_max. Only elements 4, 5, 7, and 8 are valid. For example, if hit_boundary(7) is not equal to 0, then golden_para_min(7) should be made smaller (if hit_boundary(7) is 1) or golden_para_max(7) should be made larger (if hit_boundary(7) is 2). This should be repeated until all valid elements of hit_boundary are zero.
(e) Num_total_evaluations(1) tells how many times fun_to_min was called. Num_total_evaluations(2) tells how many times golden1d_ds was called. Num_total_evaluations(3) tells how many times golden2d_ds was called. Num_total_evaluations(4) tells how many times golden3d_ds was called. This information can be used to optimize ivar_place_golden.

At the end of the process, the true minimum is located within +/- golden_tol(k) of golden_para(k) for k = 4, 5, 7, and 8.