

A Brief Guide for Using CGO

© 2009-2010 Cayuga Research Associates, LLC

This guide provides a brief introduction to the use of the Cayuga Global Optimizer (CGO). Please refer to our “CGO User’s Guide” for more detailed information.

Currently CGO addresses the following global optimization problems:

$$\min \{ f(x) : l \leq x \leq u \}, \quad (0.1)$$

$$\min \left\{ \|F(x)\|_2^2 : l \leq x \leq u \right\}, \quad (0.2)$$

$$\min \left\{ g^T x + \frac{1}{2} x^T H x : l \leq x \leq u \right\}. \quad (0.3)$$

In (0.1) f is a general differentiable function mapping vectors x to scalars $f(x)$; in (0.2) F is vector-valued function; in (0.3) H is a symmetric indefinite matrix.

There are two ways to call CGO. One is to use the graphical user interface (GUI); the other is to use one of three function calls, i.e., **cgobndmin**, **cgobndnls** or **cgobndquad**. Regardless of the chosen access method, users are required to write the objective function in either MATLAB or C. In addition, the MATLAB Optimization Toolbox is required. To use the Cayuga Research automatic differentiator ADMAT within CGO, objective functions must be written in MATLAB.

1. Installation of CGO for Windows Users

CGO is provided in a zip file CGO.zip. Unzip the file CGO.zip and obtain a folder named CGO that contains the CGO package. You may place CGO in a folder of your choice.

The steps for installing CGO are as follows:

1. Click “File” in the MATLAB window.
2. Choose the “Set Path” option.
3. Click the “Add with Subfolders” button.
4. Find the folder for CGO in the “Browse for Folders” window and click “OK”.
5. Click the “Save” button to save the paths for CGO and click the “Close” button.
6. Set the CGO folder to be the current folder of the MATLAB command window.
7. Type “startup” to activate CGO.
8. Type “Y” in the MATLAB command window to accept the end-user licensing agreement.

The message ‘CGO installed successfully’ should appear in the MATLAB command window if CGO is properly installed. The steps 1-8 described above need to be performed only once during the installation of CGO.

In order to use the full spectrum of functionalities offered by CGO, the **Automatic Differentiation Toolbox ADMAT 2.0** from Cayuga Research must be installed as well. Please consult the User’s Guide for ADMAT 2.0 for the appropriate installation of ADMAT 2.0.

2. Installation of CGO for Unix (Linux) Users

1. Unzip CGO.zip using “unzip CGO.zip” in the Unix (Linux) prompt.
2. Access the CGO directory.
3. Edit **startup.m** file. Add ALL subdirectories of CGO search paths in the file manually.
4. Save the file and type **startup** at the MATLAB prompt to set up the paths for the package.
5. Type ‘Y’ in the command window of MATLAB to accept the end-user licensing agreement.

3. CGO Activation

After the proper installation of CGO, activation of CGO may be needed in a newly-launched MATLAB session.

If the message ‘CGO installed successfully’ appears in the MATLAB command window at the start of a new MATLAB session, no activation of CGO is needed. CGO is ready for use.

Otherwise, first make sure that CGO has been properly installed. Then set the CGO folder to be the current folder of the MATLAB command window; type “startup” to activate CGO. The message ‘CGO installed successfully’ should appear in the MATLAB command window, which indicates that CGO is now activated and ready for use.

4. Write your objective function

The minimum requirement that CGO imposes on the objective function is to use the form: $f = \text{myfun}(x, \text{varargin})$, where the minimization of f is over x . Additional requirements depend on the chosen method of solution and are outlined below.

Algorithm SA stands for simulated annealing; SA+Q, SA+num avg, and SA>true avg are variants of SA with different types of smoothing method. Other user-defined functions may be needed depending on the algorithm chosen. See the table below for details.

Possible Phase 1 Algorithms	Applicable to problems	Objective function format and other functions needed
SA	Available for all dimensions	$f = \text{myfun}(x, \text{varargin})$
SA+Q	Available for all dimensions	$[f, g, H] = \text{myfun}(x, \text{varargin})$ where g is the gradient and H is the Hessian. Or, $f = \text{myfun}(x, \text{varargin})$ and use ADMAT
SA+num avg	1D	$f = \text{myfun}(x, \text{varargin})$
	2D	Write the objective function in form: $f = \text{myfunint}(x_1, x_2)$ Dot operations such as $.*$ and $./$ should be used in myfunint.m to calculate objective function f instead of $*$ and $/$.
SA>true avg	1D	$[f, g, H, \text{Itrue}] = \text{myfun}(x, \text{varargin})$ where $\text{Itrue} = \int f(x) dx$
	2D	Additional user-defined function needed besides objective function $f = \text{myfun}(x, \text{varargin})$: $\text{ftrueint} = \text{functionnametrueint}(a, b, c, d, \text{varargin})$ where $\text{ftrueint} = \int_a^b \int_c^d f(x_1, x_2) dx_2 dx_1$

In Phase 2, depending on how the objective function is defined, different methods will be used to calculate derivatives.

Objective function given	Use ADMAT	Methods used in Phase 2 to calculate derivatives
$f = \text{myfun}(x, \text{varargin})$	NO	Finite differencing
	YES	ADMAT (automatic differentiation)
$[f, g, H] = \text{myfun}(x, \text{varargin})$	NO	User-supplied derivatives
	YES	ADMAT

5. Using the CGO graphical user interface

To use the GUI, change the MATLAB current directory to the ‘CGO’ folder. Type “CGO” in the MATLAB command window. A window to choose a solver will pop up on your screen.

1. Determine whether the user-defined objective function is differentiable or not and whether the problem is special, i.e., nonlinear least squares problem or quadratic programming.
2. Choose an appropriate solver and click ‘Open Solver’.
3. Once a Cayuga Global Optimizer solver is opened, choose algorithms, set parameters, enter user-defined function name and number of variables, and provide lower bounds and upper bounds.
4. Click ‘Solve’ when all are set. In each solver, some sample objective functions are provided.

6. Using the CGO function calls

CGO function calls `cgobndmin`, `cgobndnls`, and `cgobndquad` are available to users in the style of MATLAB function `fmincon`. The syntax of each of these three functions is given below,

To solve $\min_{lb \leq x \leq ub} f(x)$:

`[xbest, fbest, elapsedtime] = cgobndmin(fun, x0, lb, ub, phase1options, phase2options, otheroptions, varargin)`

To solve $\min_{lb \leq x \leq ub} \|F(x)\|_2^2$:

`[xbest, fbest, elapsedtime] = cgobndnls(fun, x0, lb, ub, phase1options, phase2options, otheroptions, varargin)`

To solve $\min_{lb \leq x \leq ub} \frac{1}{2} x^T H x + g^T x$:

`[xbest, fbest, elapsedtime] = cgobndquad(quad_H, quad_g, lb, ub, x0, phase1options, phase2options, otheroptions, varargin)`

Set the options for Phase 1, Phase 2, (and other options) as indicated below. Available options for each function are listed and explained (see the CGO Users Guide for more details). Default settings of these options can be found in each of the solvers.

Phase 1 options	• <code>TrialeachTem</code>	Number of trials at each temperature level in simulated annealing algorithm (SA) –type method. A positive integer is required.
	• <code>SAalgorithm</code>	Choice of SA variant. Choices include ‘sa’, ‘saq’, ‘sanumavg’, and ‘satruavg’.
	• <code>CoolingSchedule</code>	A vector with entries decreasing to zero for the temperatures used in simulated annealing algorithm or its variant.
	• <code>SmoothingSchedule</code>	A vector with entries decreasing to zero for the smoothing coefficients used in the SA variants ‘saq’, ‘sanumavg’, and ‘satruavg’ algorithm.

	<ul style="list-style-type: none"> • UseApproximation • ApproximationFcn • Numofeig 	<p>Choose an approximate objective function in Phase 1 if 'UseApproximation' is set to 'on'.</p> <p>If 'UseApproximation' is set to 'on', provide function handle of the approximating function in this field.</p> <p>Approximating function is written for the indefinite quadratic programming problem. A positive integer is required. See the User Guide for more details.</p>
Phase 2 options	In cgobndmin, cgobndnls, and cgobndquad, Phase 2 options are the same as the options in MATLAB functions fmincon, lsqnonlin, and quadprog respectively.	
Other options	<ul style="list-style-type: none"> • SaveDir • NumofSoltoSave • UseAdmat • MaxTimeinPhaseOne • Phase1checked • Phase2checked 	<p>Results in .mat format will be saved in this directory.</p> <p>A set of optimizers and their function values can be saved.</p> <p>If UseAdmat is set to 'on', the CGO package ADMAT will be used to get derivatives.</p> <p>A time bound for the application of Phase 1 (simulated annealing). A positive real number is required.</p> <p>If Phase1checked is set to 'on', Phase 1 will be executed.</p> <p>If Phase2checked is set to 'on', Phase 2 (independent local minimizations) will be conducted</p>

Note that phase 1 option 'Numofeig' is only applicable in cgobndquad; phase 1 option 'SmoothingSchedule' is not applicable in cgobndnls; phase 1 option 'ApproximationFcn' and 'ApproximationFcn' are not applicable in cgobndquad. 'SmoothingSchedule' should always have the same length as 'CoolingSchedule'.

Examples. A detailed illustration of the use of these three function calls can be found in demo_cgofunctioncall.m in the folder CGO\Demos. Here we provide two brief examples.

Example 1: Solve $\min_{-10 \leq x, y \leq 10} 1 + \frac{x^2}{200} + \frac{y^2}{200} - \frac{\cos x \cos y}{\sqrt{2}}$ using cgobndmin with default options.

The objective function is written in griewank.m as follows.

```
function [f, g, H] = griewank(x, varargin)

f = (1 + x(1)*x(1)/200 + x(2)*x(2)/200 - cos(x(1))*cos(x(2))/sqrt(2));
if (nargout >= 2)
    g(1,1) = x(1)/100 + sin(x(1))*cos(x(2))/sqrt(2);
    g(2,1) = x(2)/100 + cos(x(1))*sin(x(2))/sqrt(2);
```

```

end
if (nargout == 3)
    H(1,1) = 1/100 + cos(x(1))*cos(x(2)/sqrt(2));
    H(2,1) = -sin(x(1))*sin(x(2)/sqrt(2))/sqrt(2);
    H(1,2) = H(2,1);
    H(2,2) = 1/100 + cos(x(1))*cos(x(2)/sqrt(2))/2;
end

```

Solve this problem from starting points x_0 . Each column of x_0 is a starting point. Call `cgobndmin` with default options as follows.

```

x0 = 5*rand(2,10);
[xbest, fbest, elapsedtime] = cgobndmin(@griewank, x0, [-10;-10], [10;10])

```

Call `cgobndmin` with options set as follows.

```

phasesoptions = optimsetphase1('cgophase1');
phasesoptions = optimsetphase1('TrialeachTem', 50, 'SAalgorithm', 'saq');
phase2options = optimset('fmincon');
phase2options = optimset('display', 'iter');
otheroptions = optimsetother('cgootheroption');
otheroptions = optimsetother('SaveDir', 'C:\Resultsfolder', 'UseAdmat',
'on');
x0 = 5*rand(2,10);
[xbest, fbest, elapsedtime] = cgobndmin(@griewank, x0, [-10;-10], [10;10],
phasesoptions, phase2options, otheroptions)

```

To use `'sanumavg'` in `'SAalgorithm'`, additional function `griewankint.m` is needed.

```

function f = griewankint(x1,x2, varargin)

f = (1 + x1.*x1./200 + x2.*x2./200 - cos(x1).*cos(x2./sqrt(2)));

```

To use `'satruavg'` in `'SAalgorithm'`, additional function `griewanktrueint.m` is needed.

```

function [f] = griewanktrueint(a,b,c,d, varargin)

f = a*c - a*d - b*c + b*d + (a*c^3)/600 + (a^3*c)/600 - (a*d^3)/600 -
(b*c^3)/600 - (a^3*d)/600 - (b^3*c)/600 + (b*d^3)/600 + (b^3*d)/600 -
2^(1/2)*sin(a)*(sin((2^(1/2)*c)/2) - sin(1/2*2^(1/2)*d)) +
2^(1/2)*sin(b)*(sin((2^(1/2)*c)/2) - sin(1/2*2^(1/2)*d));

```

`cgobndnls` can be called in a similar way to the above example that uses `cgobndmin`. Please refer to `demo_cgofunctioncall.m` for details.

Example 2. An illustration of the use of `cgobndquad` is as follows.

Solve $\min_{\substack{-10 \\ [-10] \leq x \leq [10]}} \frac{1}{2} x^T H x + g^T x$ using `cgobndquad`, setting phase one options as indicated, and using

30 random starting points where H is a 6-by-6 matrix and g is a 6-by-1 vector.

```

H = (magic(6)+magic(6)')/2; g = rand(6, 1);
phasesoptions = optimsetphase1('cgophase1');
phasesoptions = optimsetphase1(phasesoptions, 'numofeig', 2, 'TrialeachTem',
20);
[xbest, fbest, elapsedtime] = cgobndquad(H, g, -10*ones(6, 1), 10*ones(6, 1),
-10+20*rand(6, 30), phasesoptions)

```