RcppGO User Guide

Peter Kehler Jr
Altenstadt, Germany
Email: peter.kehler.jr@googlemail.com

25 November 2012
Version 1.0

Contents

1 Introduction

Different kinds of optimization problems require a dedicated algorithm. "[...], optimization algorithms are guided by objective functions. A function is difficult from a mathematical perspective in this context if it is not continuous, not differentiable, or if it has multiple maxima and minima."[1] The aim of the `RcppGO` package is to provide an algorithm to tackle difficult otimization problems as defined before. The algorithm utilizes Newton's laws of gravity and motion and is loosely based on Kaveh and Talatahari (2010b). In the interest of brevity the user guide provides only the steps to get startet with the package. A detailed article on the algorithm is in preparation. The `RcppGO` package relies on the `Rcpp`[2] and the `RcppArmadillo`[3] packages.

The guide is organized as follows. Section 2 describes how to install the `RcppGO` package. Section 3 presents two examples of application and goes into the differences for uni- and multidimensional optimization problems.

2 Installation

The `RcppGO` package is available on GitHub. As of today a manual installation process is required. This section describes the steps needed to install the `RcppGO` package after downloading it. For further convention we assume that you saved the package in your download folder.

2.1 Requirements

`RcppGO` depens on the `R` packages `Rcpp`[4] and `RcppArmadillo`[5]. `R` can be downloaded at the CRAN homepage. The installation of `R` is documented in R Development Core Team (2012a).

Package	Version
`R`	$\geq$ 2.14.1
`Rcpp`	$\geq$ 0.9.11
`RcppArmadillo`	$\geq$ 0.3.2.0
`RcppDE`	$\geq$ 0.1.0

Table 1: `RcppGO` installation requirements. The `RcppDE`[7] package is optional and will be needed for performance comparison.

After installing `R` you can invoke the installation of the other packages via the command in listing 1.

```
1 install.packages(c("Rcpp", "RcppArmadillo", "RcppDE"))
```

Listing 1: Installing required packages within `R`

2.2 Installation procedure

There are two options to install `RcppGO` on your maschine. You can either use the `shell` commands provided below or modify and use a prefabricated `shell` script. A prefrabricated self-executing[8] `shell` script can be found in the `RcppGO` package. The steps are:

(a) define `Rcpp` flags [9]

(b) export the defined `Rcpp` flags

[1] (Weise, 2009, p. 56)
[2] Eddelbuettel and Francois (2011)
[3] Eddelbuettel et al. (2010); Sanderson (2010)
[4] Eddelbuettel and Francois (2011)
[5] ?Sanderson (2010)
[8] In order to execute the script the user may need to change the mode of the script. This can be done via a `shell` command provided in the script.
[9] See the (R Development Core Team, 2012c, chapter 5) manual for further information.

(c) call the R command to install the RcppGO package

The following listing shows the commands that have to be executed in the shell. First open your shell and go to the folder where the RcppGO package is located, e.g. your download folder.

```
cd "~/Downloads/"
```

Listing 2: Open terminal and go to the RcppGO folder

Following the installation steps outlined above, enter the following commands in the shell window. The "#" sign is for comments only.

```
# define flags
RCPP_CXXFLAGS=`Rscript -e 'Rcpp:::CxxFlags()'`
RCPP_LIBS=`Rscript -e 'Rcpp:::LdFlags()'`
RCPP_ARMADILLO=`Rscript -e 'RcppArmadillo:::CxxFlags()'`

# export flags
export PKG_CPPFLAGS="${RCPP_ARMADILLO} ${RCPP_CXXFLAGS}"
export PKG_LIBS="-larmadillo -llapack  ${RCPP_LIBS}"

## call the R install command
R CMD INSTALL RcppGO*
```

Listing 3: Commands for the installation via the shell command line

If the procedure worked correct, the output on your console should look like this:

```
** R
** preparing package for lazy loading
** help
*** installing help indices
** building package indices ...
** testing if installed package can be loaded
*** arch - i386
*** arch - x86_64

* DONE (RcppGO)
```

Listing 4: Terminal output

After installing the package it can be loaded into your R session.

```
library(RcppGO)
```

Listing 5: Loading packages into R

Loading the RcppGO package loads also the packages RcppGO dependens on. After loading the library into R the help page of the RcppGO package is available. Type the following command in R to open it:

```
help("RcppGO")
```

Listing 6: Opening help page

The help page gives a short description of the package, its functions, their parameters and two introductory examples.

In the future the RcppGO package will be availble via CRAN. This will simplify the installation process on the user side to:

```
1 install.packages{"RcppGO"}
```

Listing 7: Installation via CRAN

3 Examples

The RcppGO package contains two functions. RcppGO() for calling the optimization routine and RcppGO.Plot() for various plots of two dimensional functions. In this section introductory examples of the RcppGO() function will be given.

Remark 3.1. *The following conventions are used in this guide.*

1. *The examples assume that the objective function is to be minimized.*
2. *The capitalized letters in the R code indicate a matrix notation instead of a vector notation.*

3.1 The RcppGO() function

The two optimization problems included in the help pages utilize variants of the Aluffi-Pentiny[10] function as objective functions. It is important for the user to understand, that one and multi-dimensional optimization problems are treated differently in RcppGO. They differ in their definition. Before two detailed examples are given, the general steps of how to handle optimization problems with RcppGO are defined:

(a) Define the objective function (optional: add constraints)[11],

(b) define search space, and the RcppGO() specific parameters[12],

(c) store the calculations in an R object,

(d) and print the results.

3.1.1 A one dimensional optimization problem

The first example in the help pages contains a one dimensional Aluffi-Pentiny objective function. The objective function is given by:

$$f(x) = \frac{1}{4}x^4 - \frac{1}{2}x^2 + \frac{1}{10}x + \frac{1}{2}x^2 \tag{1}$$

The minimum is located at approximately $x = -0.46415$ with $f(x) = -0.03481$. As outlined above the objective function has to be defined in R.

[10] The Aluffi-Pentiny function is a standard function used to benchmark optimization algorithms. See Yang (2010b), Yang (2010a), Molga and Smutnicki (2005) for further details.
[11] See section 3.1.3.
[12] See A.1

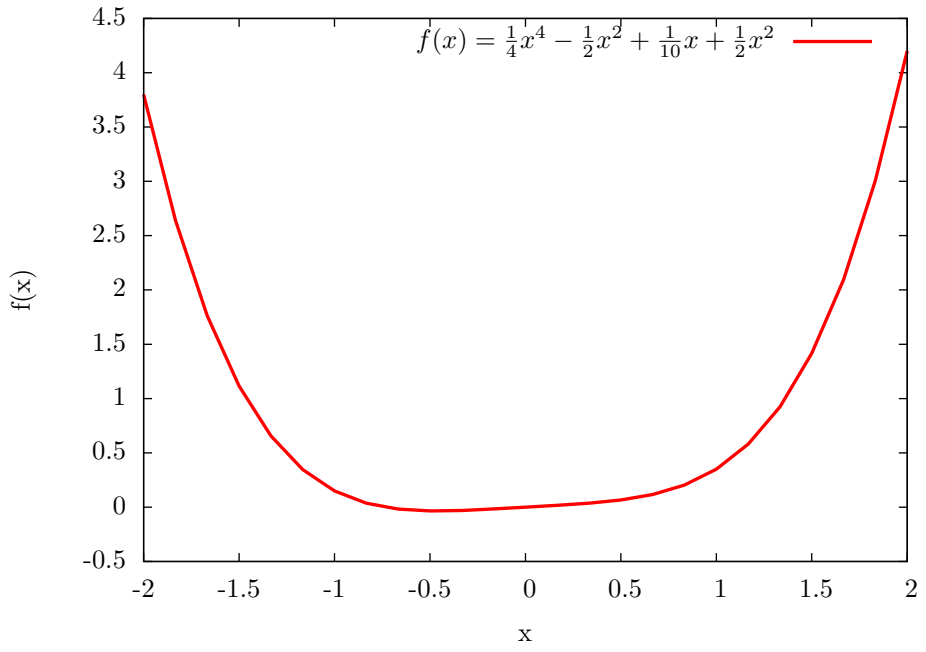

Figure 1: The Aluffi-Pentiny function

```r
# Defining f(x) = 1/4 x^4 - 1/2 x^2 + 1/10 x + 1/2 x^2 in R

AluffiPentiny01 <- function(X)
{
  1/4*X^4 - 1/2*X^2 + 1/10*X + 1/2*X^2
}
```

Listing 8: Defining objective function (1) in R

Following the steps outlined before, the search space and the RcppGO() function specific parameters have to be defined.

```r
  Example01_Output <- RcppGO(          # store calculations in an R object
  ObjectiveFunction=AluffiPentiny01,   # passing the objective function
  Args=1,                              # define the number of arguments
  Lower = -10,                         # define lower limit of search space
  Upper = 10                           # define upper limit of search space
  )
```

Listing 9: defining the parameters of RcppGO()

See section A.1 for a detailed description on all parameters of RcppGO. Before the results are accessed, the structure of Example01_Output is explained. Typing str(Example01_Output) into the R console reveals the composition of Example01_Output.

```
1 >  str(Example01_Output)               # structure of the results
2 List of 12
3  $ GravityParticles : num [1:20, 1:4, 1:200] -0.2349 -0.2246 0.0854 1.2793 -1.5062 ...
4   ..- attr(*, "dimnames")=List of 3
5   .. ..$ : chr [1:20] "1" "2" "3" "4" ...
6   .. ..$ : chr [1:4] "x1" "fn_x" "v_x1" "F_x1"
7   .. ..$ : NULL
8  $ ObjectiveFunction:function (X)
9   ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 1 20 4 1 20 1 1 4
10  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x20f7ae0>
11 $ GMemory          : num [1:20, 1:2] -0.464 -0.464 -0.464 -0.464 -0.464 ...
12  ..- attr(*, "dimnames")=List of 2
13  .. ..$ : chr [1:20] "1" "2" "3" "4" ...
14  .. ..$ : chr [1:2] "x1" "fn_x"
15 $ Iterations       : num 200
16 $ Args             : int 1
17 $ n                : int 20
18 $ g                : int 20
19 $ Lower            : num -10
20 $ Upper            : num 10
21 $ Scale            : num 0.1
22 $ User             : logi FALSE
23 $ Maximize         : logi FALSE
```

Listing 10: Understanding the results

Line 2 of listing 10 shows that `str(Example01_Output)` is a `list` of 12 objects. The results of the optimization process are stored in `GMemory`. The acces of single objects is done via the listname followed by a $ sign and the list object, as shown in the following listing.

```
1 > Example01_Output$GMemory            # print the results
2            x1        fn_x
3 1  -0.4641558 -0.03481192
4 2  -0.4641725 -0.03481192
5 3  -0.4641759 -0.03481192
6 4  -0.4641796 -0.03481192
7 5  -0.4641825 -0.03481192
8 6  -0.4641857 -0.03481192
9 7  -0.4641860 -0.03481192
10 8  -0.4641264 -0.03481192
11 9  -0.4641917 -0.03481192
12 10 -0.4641940 -0.03481192
```

Listing 11: Accessing the results

The stochastic results match the theoretical results.

3.1.2 A multi-dimensional optimization problem

In order to enable the user o define complex functions with an arbitrary amount of variables, the `RcppGO` objective function has to be written in matrix notation.[13] Here is an example for the two dimensional AluffiPentiny function.

```
1 # min at -0.352386, X in [-10,10]^2
2 AluffiPentiny02 <- function(X)
3 {
4   1/4*X[,1]^4 - 1/2*X[,1]^2 + 1/10*X[,1] + 1/2*X[,2]^2
5 }
```

Listing 12: Defining equation (1) in R using matrix notation

[13] The matrix notation `X[1,2,]` means: Access the the element in row 1 in coloumn 2. Therefore `X[ ,2]` means: Access all elements in coloumn 2.

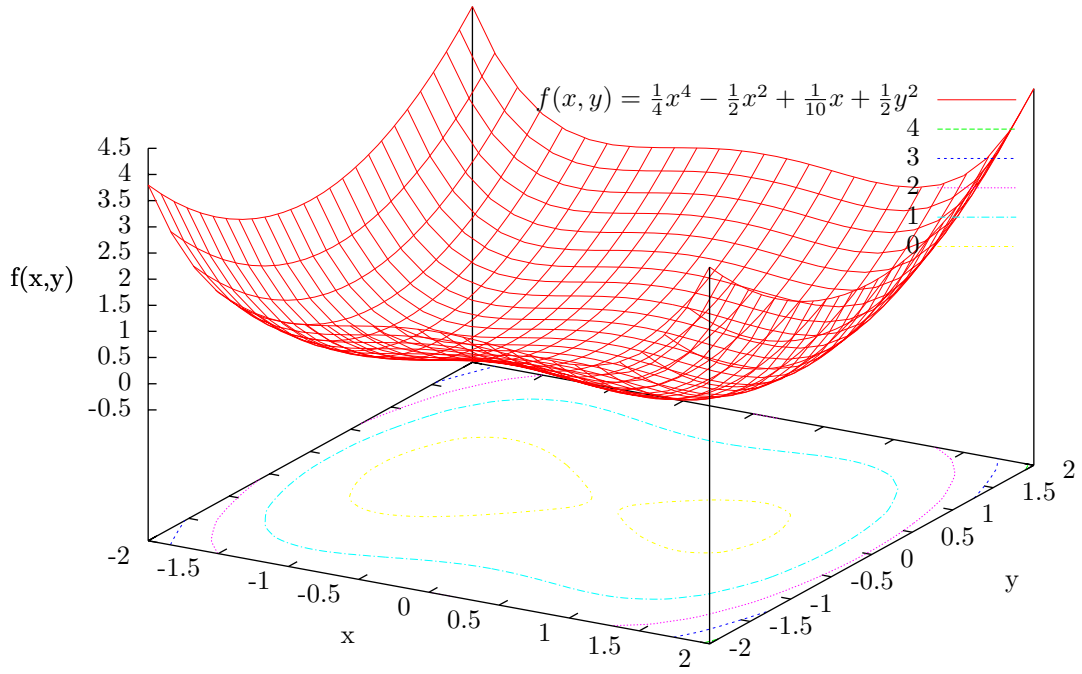

Figure 2: The two dimensional AluffiPentiny function.

Asumming the user wants to compute an approximate of the best solutions in the lower bounds of $\mathbf{x}_{lower} = \begin{pmatrix} -10 \\ -10 \end{pmatrix}$ and upper bounds $\mathbf{x}_{upper} = \begin{pmatrix} 10 \\ 10 \end{pmatrix}$, the optimization process can be initialized as follows:

```
> Example02 <- RcppGO(ObjectiveFunction=AluffiPentiny, Args=2, Lower = -10, Upper = 10)
```

Listing 13: Initializing the optimization process

Here the output of the optimization process is assigned to the variable `Test00`.In its current form the `RcppGO` contains additional information needed e.g. for the `RcppGO.Plot` function. Typing > `Test00` would result in a long chain of output.[14] To access the results only the user has to access an element of the list, called `GMemory` in `R`, as shown below:

```
> Example02$GMemory
          x1            x2       fn_x
1  -1.045695 -0.0008089336 -0.3523846
2  -1.043545  0.0085946639 -0.3523379
3  -1.049650 -0.0112856804 -0.3523123
4  -1.036312 -0.0061996451 -0.3522451
5  -1.050161  0.0163825537 -0.3522380
6  -1.059847 -0.0025767241 -0.3521822
7  -1.033795  0.0109659926 -0.3521384
8  -1.031220 -0.0060378656 -0.3520984
9  -1.072199 -0.0038773126 -0.3516166
10 -1.062366 -0.0334229455 -0.3515422
```

Listing 14: R output

The output rows show the top ten solutions generated during the optimization process. The first two coloumns represent the coordinates where the solutions were found and the third coloumn exhibits the corresponding function values.

[14] The structure of `Test00` is a list. You can see taht by typing > `str(Test00)` in the `R` console.

3.1.3 Constrained optimization

(Todo: An approach for contraint optimization will be added in the future.)

3.2 The RcppGO.Plot() function

The RcppGO.Plot function is a wrapper around a few lattice and R functions and enables the user to visualize the structure of two dimensional optimization problems. Following the example from section 3.1.2 listing 15 shows how to call the plot function.[15]

```
1 RcppGO.Plot(RcppGO.Data=Example02)
```

Listing 15: RcppGO.Plot function

Figure 3 shows the result in R. The image consists of two panels. The left one exhibits a relief-like view of the objective function and the right one shows a contourplot. In both plots the function values are coloured in topographical colours.

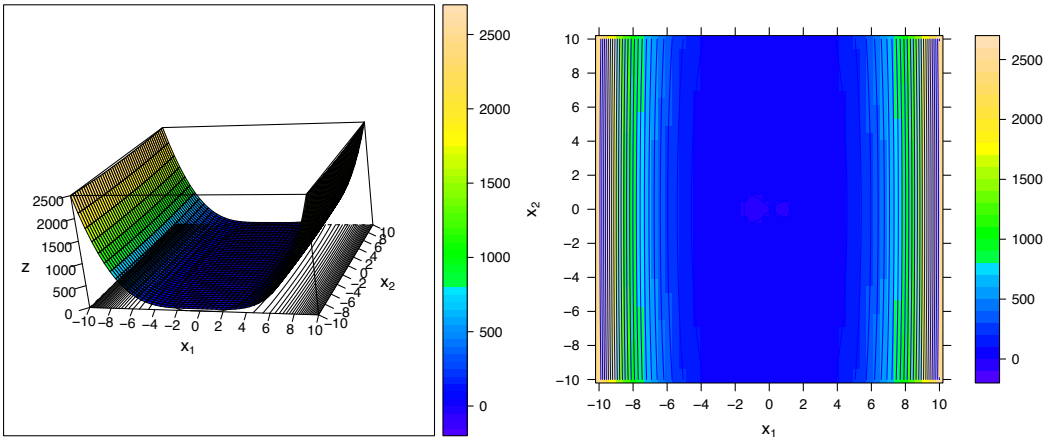

Figure 3: The RcppGO.Plot function with Particle_Sim=FALSE

Invoking the RcppGO.Plot function with Particle_Sim=TRUE visualizes the search process in an particle movement animation. The graph is updated in intervalls controlled via the Delayed option. If Physics=TRUE, the resulting force on any particle is visualized, as well as the current velocity and the position of the particles in $t + 1$.[16]

[15] Further details of the plot function are presented in section A.2.
[16] See the legend of 4.

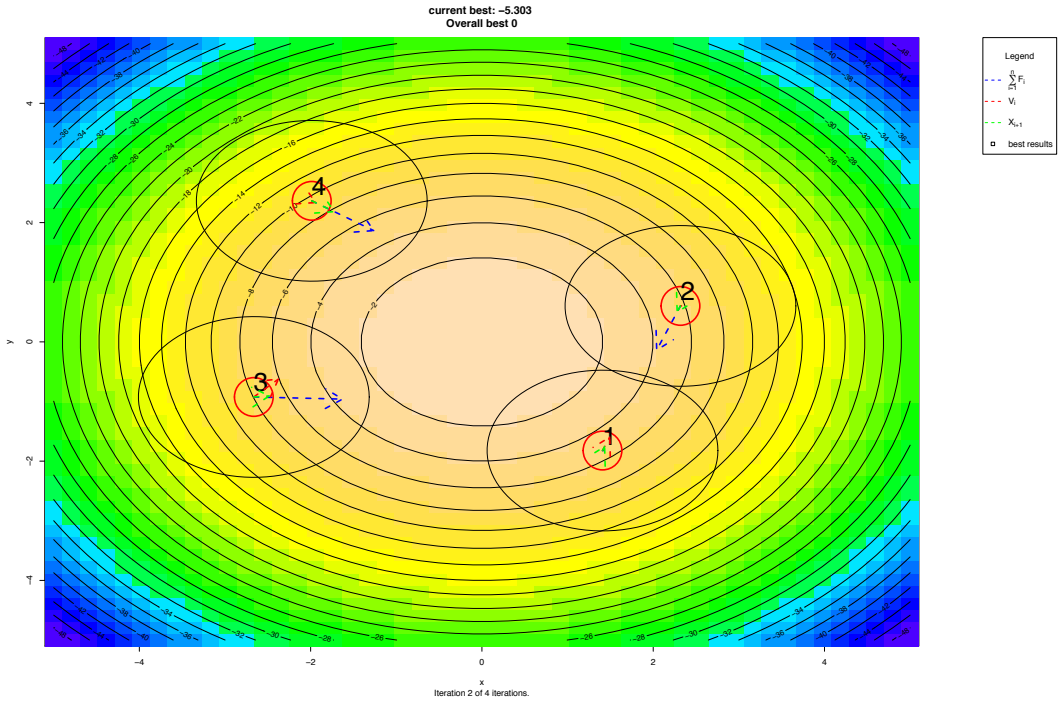

Figure 4: The RcppGO.Plot function with Particle_Sim=TRUE of $f(x, y) = x^2 + y^2$.

References

Eddelbuetel, D. and Francois, R. (2011). Extending Rcpp.

Eddelbuettel, D. (2011). From DEoptim to RcppDE: A case study in porting from C to C++ using Rcpp and RcppArmadillo.

Eddelbuettel, D. and Francois, R. (2011). Writing a package that uses Rcpp.

Eddelbuettel, D., Francois, R., and Bates, D. (2010). RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library.

Kaveh, A. and Talatahari, S. (2010a). A Charged System Search With A Fly To Boundary Method For Discrete Optimum Design Of Truss Structures. *Asian Journal Of Civil Engineering (Building And Housing)*, 11(3):277–293.

Kaveh, A. and Talatahari, S. (2010b). A Novel Heuristic Optimization Method: Charged System Search. *Acta Mechanica*, 213(3-4):267–289.

Kaveh, A. and Talatahari, S. (2010c). An Enhanced Charged System Search For Configuration Optimization Using The Concept Of Fields Of Forces. *Structural and Multidisciplinary Optimization*, pages 1–13.

Kaveh, A. and Talatahari, S. (2010d). Geometry And Topology Optimization Of Geodesic Domes Using Charged System Search. *Structural and Multidisciplinary Optimization*, pages 1–15.

Kaveh, A. and Talatahari, S. (2010e). Optimal Design Of Skeletal Structures Via The Charged System Search Algorithm. *Structural and Multidisciplinary Optimization*, 41(6):893–911.

Kaveh, A. and Talatahari, S. (2011). A general model for meta-heuristic algorithms using the concept of fields of forces. *Acta Mechanica*, 221:99–118.

Kaveh, A., Talatahari, S., and Sheikholeslami, R. (2011). An efficient charged system search using chaos for global optimization problems. *International Journal of Optimization in Civil Engineering*, 2:305–325.

Molga, M. and Smutnicki, C. (2005). Test functions for optimization needs.

Mullen, K., Ardia, D., Gil, D., Windover, D., and Cline, J. (2011). DEoptim: An R Package for Global Optimization by Differential Evolution. *Journal of Statistical Software*, 40(6):1–26.

Pedersen, M. E. H. (2010). *Tuning & Simplifying Heuristical Optimization*. PhD thesis, University of Southampton Computational Engineering and Design Group School of Engineering Sciences.

R Development Core Team (2011). *R: A Language And Environment For Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

R Development Core Team (2012a). *R Installation and Administration*.

R Development Core Team (2012b). *The R language definition*.

R Development Core Team (2012c). *Writing R Extensions*.

Sanderson, C. (2010). Armadillo: An open source C++ Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical report, NICTA.

Tipler, P. A. and Mosca, G. (2008). *Physics for scientists and engineers*, volume 1. Macmillan.

Weise, T. (2009). *Global Optimization Algorithms - Theory and Application*.

Yang, X.-S. (2010a). *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons.

Yang, X.-S. (2010b). Test problems in optimization. *Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK*.

A Appendix

This section serves as a brief overview of the RcppGO functions. For further details read the upcoming article corresponding to the package.

A.1 The parameters of RcppGO()

The RcppGO function is the core of the RcppGO package. At its heart lies the call to the optimization algorithm written in C++. The RcppGO function has ten parameters as of today.

- ObjectiveFunction: The objective function to be optimized. Be aware of the matrix notation.
- Args: Defines the number of objective function arguments.
- Lower, Upper: Vectors specifying the lower and upper bounds of the optimization routine.
- n: The number of particles to be used in the optimization process. The default is 20.
- g: The number of solutions to be saved. g should be less or equal to n. The default value is 20.
- Iterations: Specifying the number of iterations after the algorithm terminates. Default is 200.
- Scale: A parameter for definig the attraction radius of the search particles. Default is 0.1.
- User: If the user wants to controll the Scale parameter manually, set User=TRUE. Default is FALSE.
- Maximize: Control parameter for maximizing (Maximize=TRUE) or minimizing (Maximize=FALSE) the objective function.

As mentioned before the RcppGO-function returns a list. **(Todo: insert reference)**

A.2 The parameters of RcppGO.Plot()

The RcppGO.Plot function is an attempt to visualize the optimization process. In its current state only two dimensional problems can be visualized. Support for one dimensional problems might be added in the future.

As mentioned before the RcppGO function returns a list. The results of the RcppGO function are sufficient for the plot function.

- RcppGO.Data: Requires the output from the RcppGO function.
- Delayed: Controls the delay of the moving particles in seconds. Default value is 0.3.
- Particle_Sim: If FALSE a static image with a contourplot and a 3D plot of the objective function will be printed to the screen. If TRUE a contourplot with the animated movement of the particles will be displayed.
- Physics: Physics=TRUE turns on the the resultant force on a particle (blue), the resultant velocity (red) and the position in $t + 1$ (green).
- ...: The ... parameter allows the user to put parameters through to a user defined function.