complexpyDocs Documentation

Release 0.2

Varun Gopinath

Feb 17, 2022

Contents

Contents:

CHAPTER 1

ComplexRF Package

The complexpy.py implements the complexrf optimization algorithm. A theoretical description of the method is provided in chapter **Description**. The software prerequisite are descibed in chapter **Software Prerequisites**.

The complexrf package is logically divided into three:

1.1 Source Files

The source files can be found under ComplexMeethod/complexrf/ and ComplexMeethod/complexrf/src/.

Complexpy.py - This file contains the implemenation of the complexrf method. Typically, the user need not edit this file, unless to play around with speciifc parameter values such as tolerance limits, reflection distance etc.

Sampling.py - This file contains couple of sampling methods that are used in Complexpy.py. Generally, a user need not edit this file.

setup.py - This python file is used to run the optimiation. You can think of it as the glue between the complexpy.py and an objective function. Currently, there are several ways to run setup.py which are descibed in chapter **Complex Method – Python**.

1.2 Test functions

The source files can be found under ComplexMeethod/complexrf/src/testfunctions

This folder contains several testfunctions that can used to test the complexrf with. They are named objfunc*.py. The user can simply use the existing file as a template. There is an excellant wiki where mathematical test functions are listed – search for *optimization test function*.

1.3 Documentation

The source files can be found under ComplexMeethod/complexrf/docs/

Restructured text is used for documentation along with sphinx engine. The site is hosted on readthedocs, which is where you are probably reading this text.

CHAPTER 2

Software Prerequisites

There are two version of the ComplexMethod available. An implementation in python as well as in Matlab. To use the python code, Python 3.8.x (or higher) along with numpy are needed. The matlab source files require Matlab from MathWorks.

The terminal based installation given here are for ubuntu based systems. Nonetheless, the source code for python and matlab should work in mac and windows systems, provided the necessary packages are installed.

2.1 Git

Git is optional and is not required to run the optimizer. To clone the complexmethod repository, git must be installed in your system. The following links will help you understand the basics of git, which is prerequisite if you would like to use the code and contribute to this project.

http://git-scm.com/book/en/v2

https://guides.github.com/introduction/flow/index.html

To install git from a terminal run:

```
$ sudo apt-get install git
```

Once you have git installed, clone a copy of the repository from github.

In a terminal window:

```
$ mkdir complexmethod
$ cd complexmethod
$ git clone https://github.com/Robbie025/ComplexMethod.git
```

2.2 Python

A version of complex method written in python is available in the git repository in the folder *python*.

Links to install python, numpy and matplotlib:

> https://www.python.org/downloads/
> http://www.numpy.org/

2.3 Sphinx

Sphinx is used for documentation of the project and is not needed to run the complexmethod. If you commit the changes properly, readthedocs will update the documentaiton automatically. To install sphinx, in a ubuntu terminal window run:

```
$ sudo apt-get install python-sphinx
```

For installation on other platforms see :

> https://www.sphinx-doc.org

The source files can be found in the *source* folder. To build the documentation, in a terminal window run:

```
make html
```

Remember that latex must be installed for building the math equations. For ubuntu pcs:

```
$ sudo apt-get install texlive-full
```

2.4 More Information

If you are interested in more information, please check out this site. It has tons of information to get you started on the software

> http://toolbox.readthedocs.org/en/latest/

CHAPTER 3

Complex Method – Python

The complex-rf package is logically divided into three:

1. Complexpy.py - This source file contains the implemenation of the complex method in python. Typically, the user need not edit this file, unless there is a need to change certain parameter values such as tolerance limits, reflection distance etc. A theoritical decription of the complex-rf method is provided in chapter **Description**.
2. objfunc*.py - This source file contains the implementation of the objective function that you would like to minimize. Currently, there are multiple test functions provided. The user can simply use the existing file as a template. There is an excellant wiki where mathematical test functions are listed – search for *optimization test function*.
3. start.py - This python file is used to run the optimiation. You can think of it as the glue between the complexpy.py and objfunc.py. Alternatively, you can start the optimiztion from a python interpreter. For example, to run an optimization run with complexmethod on objfunc, run the following commands in a terminal window:

```
$ python
$ import objfunc
$ import complexpy
$ import numpy as np
$ xlow=np.array([-15,-15])
$ xup=np.array([15,15])

$ samplingmethod="LHS"
$ xmin,fmin,funcvector,allf,Iterations=complexpy.complexpy_(objfunc.install,
↪xlow,xup,samplingmethod)
```

After you have installed python and numpy, to get stated run start.py at the CompleMethod/python folder.

3.1 Arguments

```
python3 -h
```

(continues on next page)

(continued from previous page)

```
optional arguments:
-h, --help            show this help message and exit
-l XLOW, --xlow XLOW  Lower bound of the test function. This is a List; e.g.:-5.0 -5.0
-u XUP, --xup XUP     Upper bound of the test function. This is a List; e.g.:5.0 5.0
-n N                  Number of Runs of the complexRF
--sample {uniform,LHS}
                      --sample -lhc or --sample -uniform will tell complexpy which␣
↪start vector to use.
-o OBJF, --objf OBJF  Define where you have the objective function. Usually in the␣
↪testfunction folder. e.g.
                      src.testfunctions.objfunc
-d, --detailed        Switch on this flag to show all the results. Could be␣
↪interesting.
-p, --process         Add this flag if you want to run many runs (>1000s) in␣
↪multithreaded mode.
```

You can using the above flags to customize your run. Exmaple of using setup.py is shown below.

```
python3 setup.py --xup '512 512' --xlow '-512 -511'
python3 setup.py --xup '512 512' --xlow '-512 -511' --objf 'src.testfunctions.objfunc5
↪' -p
```

Note: The objfunc5 is the objective function. The function install() from objfunc5 is called from setup.py

3.2 Running direclty

You can run setup.py directly where the default values can be modified as in the example below.

:: default_xlow = '-5.0 -5.0' default_xup = '5.0 5.0' default_objfunc = "src.testfunctions.objfunc2" default_n = 100

to run:

```
python3 setup.py
```

3.3 Complexpy.py

You can run complexpy.py directly as well. The default values can be modified as in the example below.

to run:

```
python3 complexpy.py
```

CHAPTER 4

The Complex Method

The Complex method was first presented by Box [1], and later improved by Guin [2]. The method is a constraint simplex method, hence the name Complex, developed from the Simplex method by Spendley et al [3] and Nelder Mead, [4]. Similar related methods go under names such as Nelder-Mead Simplex. The main difference between the Simplex method and the complex method is that the Complex method uses more points during the search process.

The first part of this chapter describes the original Complex Method followed by a description of the Complex-rf, which is development of the Complex Method.

4.1 The Complex Method

In the Complex method, the word complex refers to a geometric shape with $k \geq n+1$, points in an n-dimensional space. These k points are known as vertices of the complex. To make the explanation of the algorithm simple we wiill focus on a two-dimensional space and a complex consisting of four vertices, i.e. n = 2 and k = 4.

Typically the number of points in the complex (k), is twice as many as the number of design variables (n). The starting points are generated using random numbers. Each of the k points in the complex could be expressed according to [5] where x^l and x^u are the upper and lower variable limits and R a random number in the interval [0, 1].

$$\begin{aligned} x_i = x_j^l + R(x_j^u - x_j^l) \quad & i = 1, 2 \ldots k \\ & j = 1, 2 \ldots n \end{aligned} \tag{4.1}$$

The objective is to minimize an objective function $f(x)$. The main idea of the algorithm is to replace the worst point by a new point obtained by reflecting the worst point through the centroid of the remaining points in the complex, as illustrated in Figure 1. The worst point corresponds to the maximum value of the function vector $f(x)$. The centroid, x_c, of the points in the complex excluding the worst point x_w, could be calculated according to:

$$x_{c,j} = \frac{1}{k-1}\left[\left(\sum_{i=1}^{k} x_{i,j}\right) - x_{w,j}\right] \quad i = 1, 2 \ldots n \tag{4.2}$$

The new point is now calculated as the reflection of the worst point through the centroid by a factor α

$$x_{new} = x_c + \alpha(x_c - x_w) \tag{4.3}$$

The reflection coefficient α should equal 1.3 according to Box. If the new point is better than the worst, x_w is replaced by x_{new} and the procedure starts over by reflecting the point that is worst in the new complex. If the new point is still the worst it is moved halfway towards the centroid according to:

$$x_{new}^{'} = x_c + \frac{\alpha}{2}(x_c - x_w) \tag{4.4}$$

This Equation (4.4) ould be rearranged by substituting $\alpha * (x_c - x_w) = x_{new} - x_c$ from equation (45) yielding

$$x_{new}^{'} = \frac{1}{2}(x_c + x_{new}) \tag{4.5}$$

The procedure of moving the worst point towards the centroid is repeated until the new points stop repeating as the worst.

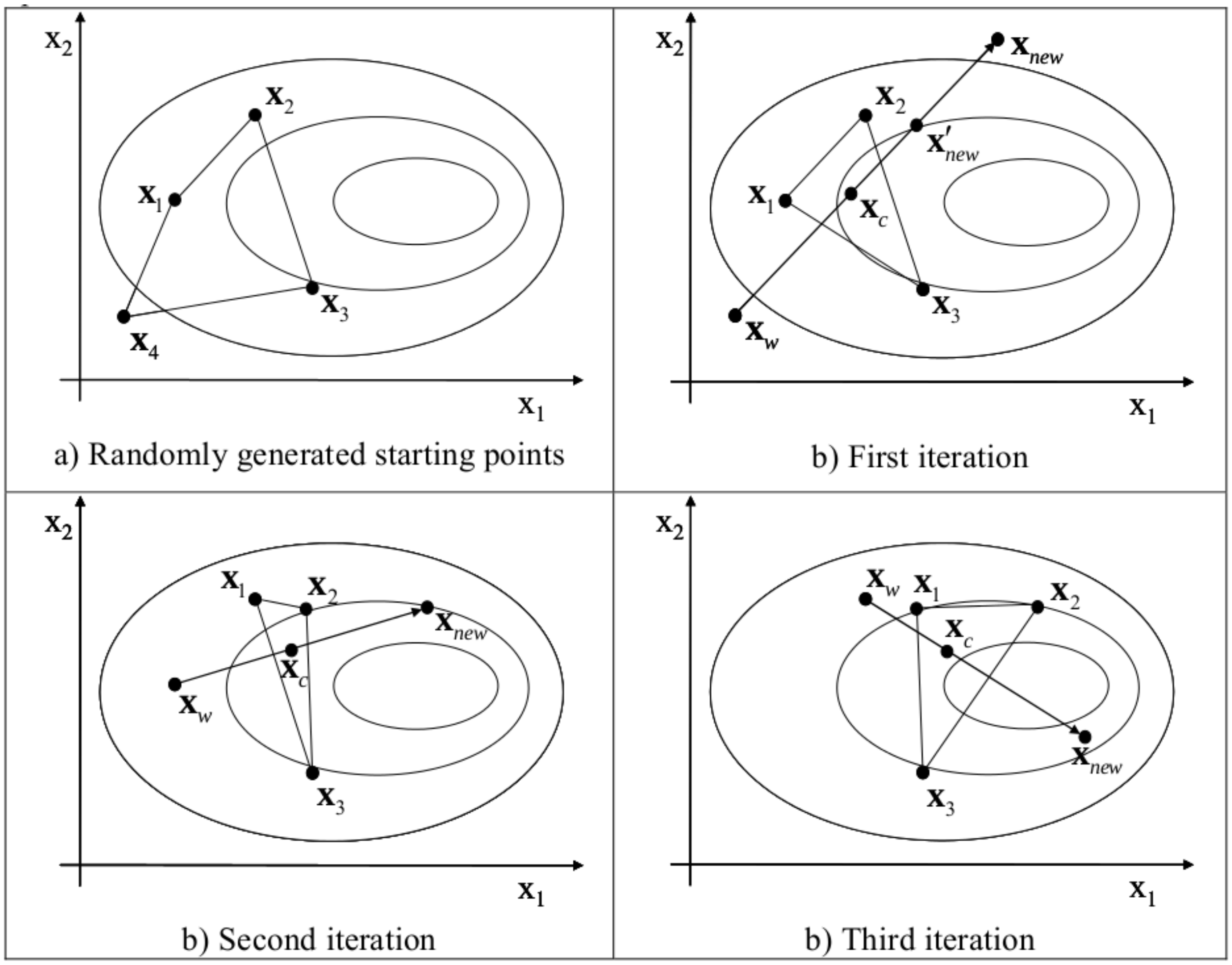

Fig. 1: **Figure 1 :Working principle of the Complex method for a problem with 2 design variables and 4 vertices in the complex. The curves represent contour lines of the objective function with the optimum to the right.**

The procedure outlined is carried out until the complex has converged or until a predescribed number of evaluations is reached. Convergence could be measured either in the function space or in variable space. In the function space the complex is considered converged if the difference between the maximum and minimum function values of all the points in the complex is less then a predescribed measure ϵ_f. Likewise, the complex has converged in the variable space if the maximum difference in all dimensions is less that a certain value ϵ_v, Thus ϵ_f and ϵ_v constitutes a measure of the spread of the complex in function space and parameter space respectively.

$$max(f(x_i)) - min(f(x_i)) \leq \epsilon_f \quad i = 1, 2 \ldots k \tag{4.6}$$

$$max[max(x_{i,j}) - min(x_{i,j})] \leq \epsilon_v \quad \begin{array}{l} i = 1, 2 \ldots k \\ j = 1, 2 \ldots n \end{array} \tag{4.7}$$

As has been stated earlier, the complex is designed to handle constraints. Constraints in the form of limits on the design variables is handle by checking if the new point is within the variable limits. If not it is move to the feasible side of the limit. If the new points is violating any other constraint it is moved halfway towards the centroid.

4.2 Pseudo Code

The working principle of the complex method is here outlined using pseudo code.

Pseudo Code

```
| Generate Starting points
| Calculate objective function
| Evaluate constraints
| Identify the worst point
| While stop criteria is not met
|       Calculate centroid
|       Reflect worst point through centroid
|       Make sure the new point is within the variable limits
|       Calculate objective function for the new point
|       Evaluate constraints
|       Identify the worst point
|       While the new point is the worst or a constraint is violated
|               Move the new point towards the centroid
|               Calculate the objective function
|               Evaluate constraints
|       end while
|       Identify the worst point in the new complex
|       Check stop criteria
| end while
| Output the optimal point
```

4.3 The Complex-RF Method

The pseudo code above describes the original Complex method as it was presented by Box. The method has some weaknesses. For one thing, if a local minimum is located at the centroid the method will keep on moving new points towards the centroid where the whole complex will collapse in one point [2]. In order to avoid this, the new point could gradually be moved towards the best point. Based on the equation (4.5) , the new point could now be expressed as

$$x_{new}^{'} = (((1-a)x_c + ax_{min}) + x_{new})\frac{1}{2} \tag{4.8}$$

where x_{min} is the best point and

$$a = 1 - e^{-k_f \frac{1}{b}} \tag{4.9}$$

where k_r is the number of times the point has repeated as the worst and b is a constant that here equals to 4. Thus the more times the point repeats as the worst the smaller a gets and the new points is moved towards the best point since x min will have a large importance in the calculation of $x_{new}^{'}$. Another feature that could be added in order to make the method less prone to collapse and lose dimensionality and to avoid getting stuck in local minima is to add some randomness to it. This is accomplished by introducing a random noise vector r to the new point in accordance to equation eq:*ten*

$$x_{new}^{'} = ((1-a)x_{min} + ax_c + x_{new})\,\frac{1}{2} + r \tag{4.10}$$

The random noise is calculated according to

$$r_f = r_{fac} * max\left(\frac{\triangle x_i}{x_i{}^u - x_i{}^l}\right)\left(x_i{}^u - x_i{}^l\right)(R - 0.5) \tag{4.11}$$

where r_{fac} is a randomization factor, x^l and x^u the variable limits and $\triangle x_i$ represent the spread in the i:th variable of the complex, and R is a random variable in the interval [0.1]. This formulation implies that the noise added is a function of the convergence $(\triangle x_i)$, and the shape of the original design space, i.e. the variable limits. Furthermore, the formulation makes it possible for the complex to maintain diversity and also regain lost dimensionality.

Since the noise is a function of the maximum spread, perturbations could be added to dimensions in which the complex has already converged. This facilitates avoidance of local optima. The randomization factor thus makes the method more robust in finding the global optima to the cost of somewhat slower convergence. Experiments have shown that a randomization factor of 0.3 is a good compromise between convergence speed and performance.

It is also possible to include a forgetting factor, γ, which ensures that the Complex is made up predominantly with recent points. This is necessary if the objective function varies over time. In that case, old objective function values become increasingly unreliable and should be replaced by new ones. This is particularly true if the optimization is to be used to optimize parameters in a real process. In this case there may be drift in the parameters of the physical system. Introducing a forgetting factor has also been found to improve the success rate in other situations as well.

One such situation is if the objective function is noisy, i.e. there are local variations in the objective function between points close to each other in parameter space, or if the objective function has a discrete nature with flat plateaus. The basic principal of the forgetting factor is to continuously deteriorate objective function values The underlying mathematics of the forgetting factor is described in detail in [6]. In [6], all parameters of the Complex algorithm are optimized in order to find the parameter set that gives the best possible performance of the algorithm. It is then concluded that $\alpha = 1.5$, $r_{fac} = 0.3$ and $\gamma = 0.3$ give a good performance of the algorithm. The complex method has been applied to a wide range of problems such as physics, structural engineering, fluid power system design and aerospace engineering.

4.4 References

1. Box, M.J., “A new method of constrained optimization and a comparison with other method”, Computer Journal, Volume 8, No. 1, pages 42-52, 1965.
2. Guin J. A., “Modification of the Complex method of constraint optimization”, Computer Journal, Volume 10, pages 416-417, 1968.
3. Spendley W., Hext G. R., and Himsworth F. R., “Sequential application of Simplex designs in optimisation and evolutionary operation”, Technometrics, Volume 4, pages 441-462, 1962
4. Nelder J. A. and Mead R., “A simplex method for function minimization”, Computer Journal, Volume 7, pages 308-313, 1965
5. Holland H. J., “Adaptation in Natural and Artificial Systems, an introductory analysis with application to biology, control and artificial intelligence”. Ann Arbor, The university of Michigan Press, 1975.
6. Krus P. and Andersson J., “Optimizing Optimization for Design Optimization”, in Proceedings of ASME Design Automation Conference, Chicago, USA, September 2-6, 2003.
7. Andersson J., “Multiobjective Optimization in Engineering Design - Application to Fluid Power Systems”, Doctoral thesis; Division of Fluid and Mechanical Engineering Systems, Department of Mechanical Engineering, Linköping University, Sweden, 2001

8. Krus P., Ölvander J., "Performance Index and Meta Optimization of a Direct Search Optimization Method", Engineering Optimization, Volume 45, Issue 10, pages 1167-1185, 2013.

CHAPTER 5

Contact

Mailing Address

Division of Machine Design
Department of Management and Engineering
Linköping University
Sweden - 581 83

5.1 Links:

Division of Machine Design

http://www.iei.liu.se/machine?l=en

Github Repository https://github.com/Robbie025/ComplexMethod

- genindex
- modindex
- search