## Contents

1 Introduction 3

2 Overview 5

3 Software Prerequisites 7
   3.1 Git 7
   3.2 Python 7
   3.3 Sphinx 7
   3.4 Matlab 8
   3.5 More Information 8

4 Complex Method – Python 9

5 Complex Method – Matlab 11

6 Complex Method – Description 13
   6.1 The Complex Method 13
   6.2 Pseudo Code 15
   6.3 The Complex-RF Method 15
   6.4 References 16

7 Contact 19
   7.1 Links: 19
The Complex Method is an optimization algorithm and is based on the Simplex method. The complex method has been applied to a wide range of problems such as physics, structural engineering, fluid power system design and aerospace engineering. The aim of the project is to make the complex method source code available to the general public. At the moment, the code is available in two implementations: python and matlab.
CHAPTER 2

Overview

The documentation for the complex method is arranged as follows:

3. **Software Prerequisites** will detail the software tools that are required to run the python and matlab codes. These include git, python with numpy, sphinx for documentation and Matlab. 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 available.

4. **Complex Method – Python** will describe the files that are included in the repository. Using examples for illustration, the included files can be used as templates for using the complex method in your own projects.

5. **Complex Method – Matlab** is a work in progress and will be updated in the near future.

6. **Description** will describe the Complex optimization algorithm along with its variant, The complex-rf.
3.1 Git

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.


To install git from a terminal run:

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

Once you have git installed, then it is quite easy to download the code onto your local drive. This process is called cloning and it will create a clone of the ComplexMethod repository.

**In a terminal window:**

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

3.2 Python

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

Links to install python and numpy:

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

3.3 Sphinx

Sphinx is used for documentation of the project. To install sphinx, in a terminal window run:

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

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

You need Matlab from mathworks to run the code. The intention is to have code compatible with both matlab and Octave.

https://www.mathworks.com/

3.5 More Information

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

http://toolbox.readthedocs.org/en/latest/
The python version of the complex method is divided into four.

1. Complexpy.py - This source file contains the implementation 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. If you would like to read a theoretical description of the complex method, please read the next section titled Description.

2. objfunc*.py - This source file contains the implementation of the objective function that you would like to minimize. Currently, the file contains one function definition (install) that takes a numpy value and returns the objective function value. The user can try to add your own objective function. The user can simply use the existing file as a template. There is an excellent wiki where mathematical test functions are listed – search for optimization test function.

3. start.py - This python file is used to run the optimization. You can think of it as the glue between the complexpy.py and objfunc.py. Of course there is no need to use this file as you can run the optimization process from a python interpreter. For example, to run the optimization with complexmethod on objfunc, run the following commands in a terminal window:

```bash
$ python
$ import objfunc
$ import complexpy
$ import numpy as np
$ xlow=np.array([[-5,-5]])
$ xup=np.array([[5,5]])
$ samplingmethod="LHS"
$ xmin,fmin,funcvector,allf,Iterations=complexpy.complexpy_(objfunc.install,xlow,xup,samplingmethod)
```

4. sampling.py - The complex method requires a set of starting values which should lie within variable limits. The easiest way is to have the user start the optimization with a user-defined starting points. However, there are other strategies that can be employed. This python file has the following: a. uniform distribution (Sample_Uniform) b. Latin-hypercube distribution (Sample_LHC) and c. user defined starting point for debugging the code (Sample_Debug). Sampling.py contains more information regarding these methods.

After you have installed python and numpy, to get started run start.py after you have cd to the python folder.

```bash
cd ComplexMethod/python
python start.py
```

start.py has been setup to optimize the first four objective functions given in repo. The function description are given in the file objfun.py, objfun2.py, objfunc3.py and objfunc4.py. You can also see the help from the python interpreter. For example:
import objfunc4
help(objfunc4)
CHAPTER 5

Complex Method – Matlab

Note: More information coming soon ....

This code is part of the TMKT48 - Design Optimization given at the Division of Machine Design, Linköping University, Sweden. Documentation can also be found in the source files.
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.

6.1 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 will 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 \([0, 1]\).

\[
    x_i = x_j^l + R(x_j^u - x_j^l) \quad i = 1, 2 \ldots k \\
    j = 1, 2 \ldots n
\]  
(6.1)

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:

\[
    x_{c,j} = \frac{1}{k-1} \left( \sum_{i=1}^{k} x_{i,j} \right) - x_{w,j} \quad i = 1, 2 \ldots n
\]  
(6.2)

The new point is now calculated as the reflection of the worst point through the centroid by a factor \( \alpha \)

\[
    x_{new} = x_c + \alpha (x_c - x_w)
\]  
(6.3)

The reflection coefficient \( \alpha \) 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:

\[
    x'_{new} = x_c + \frac{\alpha}{2} (x_c - x_w)
\]  
(6.4)

This Equation (6.4) could be rearranged by substituting \( \alpha * (x_c - x_w) = x_{new} - x_c \) from equation (45) yielding

\[
    x'_{new} = \frac{1}{2} (x_c + x_{new})
\]  
(6.5)
Fig. 6.1: 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 of moving the worst point towards the centroid is repeated until the new points stop repeating as the worst.

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 than a predescribed measure $\epsilon_f$. Likewise, the complex has converged in the variable space if the maximum difference in all dimensions is less than a certain value $\epsilon_v$. Thus $\epsilon_f$ and $\epsilon_v$ constitutes a measure of the spread of the complex in function space and parameter space respectively.

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

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

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.

### 6.2 Pseudo Code

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

#### Pseudo Code

```plaintext
| 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 |
```

### 6.3 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 (6.5), the new point could now be expressed as

\[ x'_{\text{new}} = \left( (1 - a)x_{c} + ax_{\text{min}} \right) + x_{\text{new}} \frac{1}{2} \]  

(6.8)

where \( x_{\text{min}} \) is the best point and

\[ a = 1 - e^{-k_{r} \frac{b}{4}} \]  

(6.9)

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 \( x_{\text{min}} \) will have a large importance in the calculation of \( x'_{\text{new}} \). 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

\[ x'_{\text{new}} = \left( (1 - a)x_{\text{min}} + ax_{c} + x_{\text{new}} \right) \frac{1}{2} + r \]  

(6.10)

The random noise is calculated according to

\[ r_{f} = r_{f\text{ac}} \times \text{max} \left( \frac{\Delta x_{i}}{x_{i}^{u} - x_{i}^{l}} \right) \left( x_{i}^{u} - x_{i}^{l} \right) \left( R - 0.5 \right) \]  

(6.11)

where \( r_{f\text{ac}} \) is a randomization factor, \( x^{l} \) and \( x^{u} \) the variable limits and \( \Delta x_{i} \) 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 (\( \Delta x_{i} \)), 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, \( \gamma \), 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 \( \alpha = 1.5, r_{f\text{ac}} = 0.3 \) 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.

### 6.4 References


Mailing Address

Johan Ölvander
Division of Machine Design
Department of Management and Engineering
Linköping University
Sweden -581 83
email: johan.olvander@liu.se

7.1 Links:

Division of Machine Design

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

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

• genindex
• modindex
• search