# Constrained Optimization

**ConstrainedOptimization** is a UG4 Plugin that provides various algorithms used in a constrained optimization setting.

This document should give a thorough overview on ConstrainedOptimization combined with some usage examples. For first time users, using the plugin is not necessarily straightforward, as many things have to be considered. Especially the required data format of the model output and the optimization solver setup itself may present issues. But usually all these issues can be resolved after the user has grasped the gist of how the plugin is supposed to work.
Therefore, this manual is focused more on the usage of ConstrainedOptimization from a user perspective. For an explanation of how the plugin works internally, you may refer to the doxygen file supplied with the plugin.

## Table of Contents

Installation....2
- Adding ConstrainedOptimization to UG4....2
  - GCC....2
  - Clang....3

Descripton....3
The following algorithms are currently implemented:....3
The four files needed for a successful optimization....4
- subset_target.lua....4
  - Scalar input variable....4
  - Multidimensional input variable....5
- subset_sim.lua....6
  - Scalar input variable....7
  - Multidimensional input variable....8
- evaluate.lua....9
- Running the optimizer....11
  - Call the optimizer from Lua....11
  - Call the optimizer from C++....13

Working example: Fitting disease data to an epidemics model....13
- Dataset....14
- Epidemiological Model....14
- Parameter Calibration with ConstrainedOptimization....15

Folder Structure of the Output....18
Common Errors....19

# Installation

The newest version of the plugin can be downloaded from
https://github.com/CowFreedom/ConstrainedOptimization

While compilation of ConstrainedOptimization has no dependencies, it assumes that the terminal command

```
$ ugshell
```

exists and starts a Lua compiler. This is usually satisfied by installing UG4 along with the corresponding environment variables. Experienced users could theoretically not install UG4 and set the environment variable *ugshell* to refer to any Lua (or even e.g. Python compiler). The rest of this manual assumes, however, that UG4 has been properly installed.

## Adding ConstrainedOptimization to UG4

The installation equals the standard process for adding UG4 plugins described on the ughub GitHub page. Due to the usage of OS specific fgunctions for process generation, the plugin is not necessarily multiplatform. In its current form, the package has been shown to work on Windows 10, Linux Ubuntu, Rasperry Pi OS, Arch Linux, and macOS Mojave 10.14.4. bciosyOn Windows 10, two compilers were tested, GCC 9.3.0 and Visual Studio v16.3.10. It is assumed that the package also works for older compiler versions that support C++17. On Linux Ubuntu, only GCC was tested. On macOS Mojave the Clang compiler was used. Depending on your version of GCC and Clang, special flags might need to be set during the build process. This is explained below. The utilized Visual Studio compiler has not exhibited such necessities but older versions might.

### GCC

Follow the steps on the ughub GitHub page. If errors occur, proceed with this text. The plugin makes use of C++ std::threads. This might necessitates activating the -pthread flag to the build process for ug4. Within your UG4 install library, go to

```
cd ug4/ugcore/cmake/
```

and open

```
ug_includes.cmake
```

Now search for

```
elseif("${CMAKE_CXX_COMPILER_ID}" STREQUAL "GNU")
  add_cxx_flag("-Wall")
  add_cxx_flag("-Wno-multichar")
  add_cxx_flag("-Wno-unused-local-typedefs")
  add_cxx_flag("-Wno-maybe-uninitialized")
```

and add

```
add_cxx_flag("-pthread")
```

to the GNU include statements. Now rebuild UG4 as described on the ughub GitHub page. The plugin should now be installed without any issues.

### Clang

Follow the steps on the ughub GitHub page.. If errors occur, proceed with this text. The plugin makes use of C++11 features, like std::threads and constexpr. This might necessitates activating the -std=c++11 flag to the build process for ug4. Within your UG4 install library, go to

```
cd ug4/ugcore/cmake/
```

and open

```
ug_includes.cmake
```

Now search for

```
elseif("${CMAKE_CXX_COMPILER_ID}" STREQUAL "Clang")
    add_cxx_flag("-Wall")
    add_cxx_flag("-Wno-multichar")
```

and add

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
```

to the Clang include statements. This ensures that the -std=c++11 flag is only added to the C++ compiler and not the C compiler in the build process. Now rebuild UG4 as described on the ughub GitHub page. The plugin should now be installed without any issues.

## Descripton

The following algorithms are currently implemented:

| Name | Description |
|---|---|
| Newton-Gauss | The Gauss Newton procedure minimizes functions that can be represented as a sum of squares. |
| Particle Swarm Optimization | The Particle Swarm Optimization (PSO) algorithm minimizes any continuous function on a bounded domain. |
| Geometry Sampler | Takes random samples from an arbitrary loss function. |

The inner workings of each of these algorithms are either self explaining or can be found in publically available lecture. The main feature of **ConstrainedOptimization** is the compartmentalization of problem formulation, solution finding and the underlying computation mechanism. A problem is formulated via child instantiations of the Evaluation class type, which includes details such as the objective function to be optimized or problem specific parsing details. Solution finding algorithms are represented as classes such as NewtonOptimizer. Computational interaction with UG4 is abstracted away in instances of ComputationMode. Combined, this setup gives a flexible and modular platform that can be adapted for many problems at hand. For a descripton of the inner workings of this plugin, please refer to the doxygen file supplied with the plugin.

## The four files needed for a successful optimization

Assume you have real life data and a model explaining the real life phenomena. For a successful model parameter calibration subject to the real life dataset, two steps have to be conducted:

1) You have to correspond the output of the simulation to your real life dataset.
2) You have to tell ConstrainedOptimization, how your simulation can be started and how its results can be read.

The majority of errors belong to failure in one of these two cases. The first requirement is solved by two configuration files, commonly called *subset_target.lua* and *subset_sim.lua*. These files tell the plugin which files and entires of your experimental dataset belong to which files and entires of the output of your simulation.
The second requirement necessitates little more explanation. There are many different ways to link your simulation to the solvers defined in ConstrainedOptimization. You can read more about these modes (called Evaluation classes) in the doxygen manual supplied with the plugin.
In its default configuration, (ComputationMode::File) ConstrainedOptimization assumes that your simulation is written in another language and outputs its results to file. This means that internally the shell command

```
$ugshell -ex evaluate.lua
```

is called. The expectation is, that evaluate.lua evaluates your model and writes its output to a .txt file. This output is then parsed by ConstrainedOptimization and used in corresponding computations. Once this setup is understood, a fluent workflow should follow naturally.

## subset_target.lua

Usually you want to calibrate parameters with respect to experimental data. You therefore have to tell the plugin, where your experimental data is located. It is usually the case, that this experimental data is part of a big .txt or .csv file. Because often only a few columns of these big files are relevant for the optimization, ConstrainedOptimization offers the ability to select the columns of interest. Only these selected columns will be parsed and used by the optimizers.

The name *subset_target.lua* is arbitrary and can be changed by the user. This name was chosen as a default name, because you are usually interested in a subset of the real life (target) data in the .txt or .csv files. Please note that any filepaths are relative to the location of subset_target.lua itself.

### Scalar input variable

For onedimensional functions with multidimensional output $f:\mathbb{R}\rightarrow\mathbb{R}^n$, the standard Evaluations *BiogasEvaluation* and *EpidemicsEvaluation* have a straightforward form as seen in Figure 1. The scalar input variable case is also often encountered in PDE models, because although the model itself might accept multidimensional input variables, the loss function of the optimization formulation might still only depend on time.

```
outputFiles = {
   FileOne={
     filename="expData/houseprices.txt",
     selected_columns=[0,1],
   },
 }
```

*Figure 1 Typical setup of a subset.target for a scalar problem. This selects two columns from the experimental dataset located at the given path.*

Usualy, the first selected column in the first selected file (in the image above it is column 0) always indicates the time axis in the dataset. Multiple files can also be selected as seen in Figure 2.

```
outputFiles = {
   FileOne={
     filename="expData/houseprices.txt",
     selected_columns=[0,1],
   },
   FileTwo={
     filename="expData/housedimensions.txt",
     selected_columns=[4,6,9],
   },
 }
```

Figure 2 Typical setup of a subset_target.lua for a scalar problem with multiple experimental data files

With default settings, the dataset is assumed to be tab or comma separated, although more exotic separations might also work. Additionally, comments are indicated by a hashtag “#“. Scientific and numeric notation of the output is possible.

```
#This is a comment
#t        f(x)
8.360522401153480665e-01 1.078983405519934768e+00
8.707405635874532202e-01 1.042569455957488289e+00
1.341935589811247276e+00 7.497236848362418549e-01
```

Figure 3 Sample experimental dataset with one time and one data column. The output can but does not have to be in scientific notation.

## Multidimensional input variable

For multidimensional functions with multidimensional output $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , the structure of *subset_target.lua* is usually problem specific[1]. The default EpidemicsPDEEvaluation class can be used for many problems with multidimensional problems. It assumes that the first three columns of the first file are *time*, *x* and *y*, with x,y being spatial coordinates (Figure 4).

1Please note that even for PDE models this case is much more rarely encountered than the scalar input variable case, as many optimization problems optimize with respect to location independent aggregate values.

```
# Specifiy the selected columns from the experimental data that are used for the optimization.
# The first 3 columns refer to time, posX,posY
# Please add addtional files in the order shown.

filename="expdata/expData0.txt"
selected_columns=[0,1,2,3,4,5]
filename="expdata/expData1.txt"
```

Figure 4 Sample experimental dataset with one time, two positon and many data columns.

You again specify the path where the files can be found. In comparision to the other subset_target.lua for onedimensional input variables, you now only specify the selected columns once it is assumes, that the selected columns are the same across all the files. Please note that each individual file now represents the data with respect to a specific timepoint.

```
#time  posx   posy   Density Susceptibles  Density Exposed Density Deceased
0      0      0      0.00453999   1.38879e-10   0
0      0      1      0.00453999   1.38879e-10   0
0      1      0      0.00453999   1.38879e-10   0
0      1      1      0.00453999   1.38879e-10   0
0      0.5    1      0.673795     3.72665e-05   0
0      0.5    0      0.673795     3.72665e-05   0
0      0      0.5    0.673795     3.72665e-05   0
0      1      0.5    0.673795     3.72665e-05   0
0      0.5    0.5    100    10    0
0      0.5    0.2    16.5299      0.11109       0
0      0.5    0.8    16.5299      0.11109       0
0      0      0.2    0.111378     4.13994e-07   0
0      0      0.8    0.111378     4.13994e-07   0
0      1      0.8    0.111378     4.13994e-07   0
0      1      0.2    0.111378     4.13994e-07   0
0      0.7    0.5    44.9329      1.35335       0
0      0.3    0.5    44.9329      1.35335       0
0      0.5    0.4    81.8731      6.06531       0
0      0.5    0.6    81.8731      6.06531       0
0      0.5    0.1    4.07622      0.00335463    0
0      0.5    0.9    4.07622      0.00335463    0
```

Figure 5 Sample experimental dataset *expData0.txt*, which represets data at the first timepoint. If you have data for three timepoints, you will have three such files.

The input files are again to be assumed in .txt or .csv format, with scientific or numeric number notation output.

## subset_sim.lua

Now that the experimental data has been made known to the optimizers in the previous step, the user must subsequently tell the optimizer which files and columns of the simulation output are needed for e.g. parameter calibration. The default name subset_sim.lua for these configuration files is again arbitrary and can be changed by the user. It was selected because you usually are interested in a subset of the simulation output. Please note that in contrast to the previous step, any filepaths are now relative to a specific subfolder that is created for each evaluation of the simulation. This means the following: If you run your model from the directory C:Yourname/modelevaluation/ and it saves its output into a file simoutput.txt, then the filename entry in subset_sim.lua will be

filename=“simoutput.txt“

and not

filename=“C:Yourname/modelevaluation/ simoutput.txt“

## Scalar input variable

For onedimensional functions with multidimensional output $f:\mathbb{R}\rightarrow\mathbb{R}^n$ , the standard Evaluations *BiogasEvaluation* and *EpidemicsEvaluation* have a form as seen in Figure 6. The scalar input variable case is even in PDE models often encountered, because although the model itself might accept multidimensional input variables, the loss function of the optimization formulation might still only depend on time.

```
# Specifiy the dimensions that are being optimized from the model output.

filename = "output.txt"
selected_columns = [0,2,5],
```

Figure 6 Typical setup of a subset_sim.lua configuration file. Usually the first selected column represents the time axis.

The first selected column of the first file is again usually the variable (most often time) used to annotate the different datasets. Multiple files can also be chosen again.

```
inputFiles ={

    --output of the Epidemics simulations
    FileOne= {
    filename = "output_a.csv"
    selected_columns = [0,1,2,3,4,5,6,7],
    },
    FileTwo= {
    filename = "output_v.csv"
    selected_columns = [1,2,3,4,5,6,7],
    },
}
```

Figure 7 Typical setup of a subset_sim.lua configuration file with multiple selected files. Note that in the second file the time column does not have to be specified again.

Just as last time the simulation output can be in numeric or scientific number notation. Comments are again indicated by a hashtag “#“.

```
#t      f1      f2
0       1       -3
0.1     1.001   -2.79
```

Figure 8 Typical output of the simulation for scalar input variables.

Files and columns in subset_sim.lua cannot be selected arbitrarily. They must always be in relation to the experimental data in subset_target.lua. If subset_target.lua selects five columns, then subset_sim.lua must also select five columns (although the column indices can differ). However, it is not required that the experimental and simulation data have the same number of entries. Usually the simulation data is more dense than the experimental data, as arbitrary stepsizes can be chosen to evaluate the model. It is only required, that experimental data and simulation data have the same start time and the simulation data has an end time greater or equal to the end time of the experimental data set. All values in between are adjusted by linear interpolation.

## Multidimensional input variable

For multidimensional functions with multidimensional output $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ , the structure of *subset_sim.lua* is usually problem specific[2]. The default EpidemicsPDEEvaluation class can be used for many problems with multidimensional problems on planar grid. It is important to note that on any nonrectangular grid EpidemicsPDEEvaluation is not going to work. In these cases, a costum Evaluation class can be added. See the doxygen manual provided with the plugin for further details. In contrast to subset_target.lua, the subset_sim.lua of EpidemicsPDEEvaluation does not have a selected columns entry. Instead, relevant dimensions for the optimization can be selected through the selected_dimensions entry. The selected dimensions to not contain entries for *time* or locations *x,y*. For this to work, EpidemicsPDEEvaluation requries a very specific simulation output structure, as seen in Figure 9.

```
#[TIME]:0.000000
#Seen are the grid output values for t=0.000000. If you want to see which gridpoints correspond to which coordinate entries, look for gridmapping text file (if you generated it)
#The following dimensions are plotted:
#Susceptibles
#Exposed
#Infected
#Recovered
#Deaths
#The values are plotted in the above order. Each dimension has 11 rows
95.1229424500714      95.9829129947799      96.6571504637507      97.1416464466605      97.4335089608749      97.5309912028333      97.4335089608749      97.1416464466605      96.6571504637507      95.9829129947799      95.1229424500714
95.9829129947799      96.8506582079198      97.5309912028333      98.0198673306755      98.314368463491       98.4127320055285      98.314368463491       98.0198673306755      97.5309912028333      96.8506582079198      95.9829129947799
96.6571504637507      97.5309912028333      98.2161032358301      98.7084135020288      99.0049833749168      99.1040378772884      99.0049833749168      98.7084135020288      98.2161032358301      97.5309912028333      96.6571504637507
97.1416464466605      98.0198673306755      98.7084135020288      99.2031914837061      99.5012479192682      99.6007989343992      99.5012479192682      99.2031914837061      98.7084135020288      98.0198673306755      97.1416464466605
97.4335089608749      98.314368463491       99.0049833749168      99.5012479192682      99.8001998667333      99.9000499833375      99.8001998667333      99.5012479192682      99.0049833749168      98.314368463491       97.4335089608749
97.5309912028333      98.4127320055285      99.1040378772884      99.6007989343992      99.9000499833375      100                   99.9000499833375      99.6007989343992      99.1040378772884      98.4127320055285      97.5309912028333
97.4335089608749      98.314368463491       99.0049833749168      99.5012479192682      99.8001998667333      99.9000499833375      99.8001998667333      99.5012479192682      99.0049833749168      98.314368463491       97.4335089608749
97.1416464466605      98.0198673306755      98.7084135020288      99.2031914837061      99.5012479192682      99.6007989343992      99.5012479192682      99.2031914837061      98.7084135020288      98.0198673306755      97.1416464466605
96.6571504637507      97.5309912028333      98.2161032358301      98.7084135020288      99.0049833749168      99.1040378772884      99.0049833749168      98.7084135020288      98.2161032358301      97.5309912028333      96.6571504637507
95.9829129947799      96.8506582079198      97.5309912028333      98.0198673306755      98.314368463491       98.4127320055285      98.314368463491       98.0198673306755      97.5309912028333      96.8506582079198      95.9829129947799
95.1229424500714      95.9829129947799      96.6571504637507      97.1416464466605      97.4335089608749      97.5309912028333      97.4335089608749      97.1416464466605      96.6571504637507      95.9829129947799      95.1229424500714
3.85749969592784e-21  3.12576437866998e-17  3.42781686308403e-14  5.08733129475386e-12  1.02181780561266e-10  2.7775887729928e-10   1.02181780561266e-10  5.08733129475386e-12  3.42781686308403e-14  3.12576437866998e-17  3.85749969592784e-21
3.12576437866998e-17  2.53283310981882e-13  2.7775887729928e-10   4.1223072448771e-08   8.2798754375703e-07   2.25070349438517e-06  8.2798754375703e-07   4.12230724487712e-08  2.77758877299278e-10  2.53283310981882e-13  3.12576437866998e-17
3.42781686308403e-14  2.77758877299278e-10  3.04599594894251e-07  4.52065881396209e-05  0.000907998595249695  0.00246819608173359   0.000907998595249695  4.52065881396211e-05  3.04599594894251e-07  2.77758877299278e-10  3.42781686308403e-14
5.08733129475386e-12  4.12230724487712e-08  4.52065881396212e-05  0.00670925255805024   0.13475893998171      0.366312777774684     0.13475893998171      0.00670925255805026   4.52065881396211e-05  4.12230724487712e-08  5.08733129475386e-12
1.02181780561266e-10  8.27987543757033e-07  0.000907998595249697  0.134758939981709     2.70670566473226      7.35758882342885      2.70670566473226      0.13475893998171      0.000907998595249695  8.27987543757033e-07  1.02181780561266e-10
2.7775887729928e-10   2.25070349438517e-06  0.00246819608173359   0.366312777774683     7.35758882342885      20                    7.35758882342885      0.366312777774684     0.00246819608173359   2.25070349438517e-06  2.7775887729928e-10
1.02181780561266e-10  8.27987543757033e-07  0.000907998595249697  0.134758939981709     2.70670566473226      7.35758882342885      2.70670566473226      0.13475893998171      0.000907998595249695  8.27987543757033e-07  1.02181780561266e-10
5.08733129475386e-12  4.12230724487712e-08  4.52065881396212e-05  0.00670925255805024   0.13475893998171      0.366312777774684     0.13475893998171      0.00670925255805026   4.52065881396211e-05  4.12230724487712e-08  5.08733129475386e-12
3.42781686308403e-14  2.77758877299278e-10  3.04599594894251e-07  4.52065881396209e-05  0.000907998595249695  0.00246819608173359   0.000907998595249695  4.52065881396211e-05  3.04599594894251e-07  2.77758877299278e-10  3.42781686308403e-14
3.12576437866998e-17  2.53283310981882e-13  2.7775887729928e-10   4.1223072448771e-08   8.2798754375703e-07   2.25070349438517e-06  8.2798754375703e-07   4.12230724487712e-08  2.77758877299278e-10  2.53283310981882e-13  3.12576437866998e-17
3.85749969592784e-21  3.12576437866998e-17  3.42781686308403e-14  5.08733129475386e-12  1.02181780561266e-10  2.7775887729928e-10   1.02181780561266e-10  5.08733129475386e-12  3.42781686308403e-14  3.12576437866998e-17  3.85749969592784e-21
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0

0                     0                     0                     0                     0                     0                     0                     0                     0                     0                     0
```

Figure 9 Required simulation output structure *output0.txt* for a rectangular PDE problem with five dimensions. Font size has been decreased afterwards for the data entries. The first row indicates

2Please note that this case is even for PDE models much more rarely encountered than the scalar input variable case.

Then, after a few comments, the output of the five dimensions is given sequentially in a way that resembles the planar grid.

If the simulation output is structured like in Figure 9, then the subset_sim.lua might look like in Figure 10.

```
# Specifiy the dimensions that are being optimized from the Model output.

filenameprefix = "output"
selected_dimensions = [0,1,2]
```

Figure 10 Typical subset_sim.lua dataset with one time, two positon and many data columns.

It is assumed that simulation files are stored sequentially inside the same folder with filenameprefix and then the number as suffix, like output0.txt, output1.txt, output2.txt and so on. The grid and time discretization can differ from the experimental data. This is commonly the case, as the simulation mesh can be arbitrary fine while the mesh of the experimental data is fixed. Interpolation is then used to correspond both meshes.

## evaluate.lua

The default Evaluation classes are defined with the ConfigOutput::File enum set. This means that the optimizers expect to evaluate the model function via the

```
$ugshell -ex evaluate.lua
```

terminal command and parse the results that the model has writen to file. The evaluate.lua therefore runs the model and writes its output to file. It does not matter if evaluate.lua calls other lua files or does other things to achieve this objective. The following three requirements have to be satisfied by any valid evaluate.lua:

1) It loads a file called parameters.lua which is repeatedly generated durin the optimization
2) It evaluates the mathematical model based on the parameter values previously parsed.
3) It writes its output to file in a format that adheres to the requirements of subset_sim.lua

The following example shows how this is done for a logistic regression model:

```
--[[
This is the evaluate.lua file. It is called by the computation subroutine if ConfigOutput in the
Evaluation
class is set to File. All evaluations by the problem formulation happen here.
--]]
--Load Parameters
local pars = "parameters.lua"

local file = assert(loadfile(pars))
file()



--Retrieve parameter values
v_theta1=parameters.v_theta1:get_value_as_double()
v_theta2=parameters.v_theta2:get_value_as_double()
v_theta3=parameters.v_theta3:get_value_as_double()
----------------------------------------------------------------
--Start of model calculation

local time_end=20
local delta=0.5

function logistic(theta1, theta2,theta3,x)
        return theta1/(1+math.exp(-theta2*x))+theta3
end

--Write output to file. Output file name is same as in subset_sim.lua
file=io.open("produced_deGFP.txt","w")
result="#time val val"

for i=0,time_end/delta do
        result=result.."\n"
        result=result..i*delta.."\t"..logistic(v_theta1,v_theta2,v_theta3,i*delta)
end

file:write(result)
file:close()

print("logistic1 calculation ended")
```

Figure 11 Example evaluate.lua for a logistic regression model with three parameters to be optimized. It satisfies all three requirements: First, the file parameters.txt is loaded. Then the loaded parameters are assigned to the model. And lastly, after the model is evaluated the output is written to file.

The expression

```
parameters.v_theta3:get_value_as_double()
```

returns the value of a parameter as a double. This is necessary, as parameters are saved as the internal datatype EVar64 internally (Figure 12).

```
parameters={v_theta1=EVar64(EFloat64(4.1e-06), EFloat64(0),EFloat64(0.1)),
v_theta2=EVar64(EFloat64(4), EFloat64(4),EFloat64(10)),
v_theta3=EVar64(EFloat64(0.1), EFloat64(0),EFloat64(1)),
}
```

Figure 12 The parameters.lua file generated by the optimizers during one of the optimization iterations. The first EFloat64 indicates the parameter value and the subsequent two EFloat64 its minimum and maximum bounds in the triple.

## Running the optimizer

First, the experimental data itself and structure of the model output was made known to the optimizer via subset_target.lua and subset_sim.lua files. Then, the optimizer was taught how to call the model with different parameter values via evaluate.lua
The only thing missing is starting the optimizer itself. Currently, there are two ways to achieve this:

1) Call the optimizer from any Lua code via ugshell
2) Call the optimizer from any C++ code that includes the plugin

The first way is only possible, if UG4 is installed. The second way only requires a working C++ compiler to compile (although there will be runtime errors if the environment variable *ugshell* is not defined).

### Call the optimizer from Lua

The following example shows all there is to call the solver. First, the parameters that one wants to optimize are defined. These must be identical to the ones parsed in evaluate.lua. Then the solver is configured and run. Lastly, the found parameters are output. Note: The supplied path assumes to contain evaluate.lua, subset_sim.lua and subset_target.lua.

```
ug_load_script("ug_util.lua")
----------------------------------------------------------------
-- define Home-Directories
----------------------------------------------------------------
ug4_home        = ug_get_root_path().."/"
app_home        = ug4_home.."apps/parameteroptimization/"
common_scripts  = app_home.."scripts/"
geom_home       = app_home.."geometry/"

----------------------------------------------------------------


--Defining the parameters
-- Set inital value, lower bound, upper bound for each parameter
v_alpha=EVar64(EFloat64(0.0924629297966215),EFloat64(0.00000001),EFloat64(10))
v_kappa=EVar64(EFloat64(0.1),EFloat64(0),EFloat64(1))
v_theta=EVar64(EFloat64(0.02),EFloat64(0),EFloat64(1))
v_qq=EVar64(EFloat64(14),EFloat64(5),EFloat64(16))
v_pp=EVar64(EFloat64(7),EFloat64(4),EFloat64(15))

--Add the parameters into the Parameter Manager
--Note: Only the added parameters will be optimized
manager=EVar64Manager()
manager:add("alpha",v_alpha)
manager:add("kappa",v_kappa)
manager:add("theta",v_theta)
manager:add("qq",v_qq)
manager:add("pp",v_pp)

--Defining storage place for the the estimated paramteers
estimated_parameters=EVar64Manager()

--Running Newton Gauss
stepsize_alpha=1.0
RunNewtonGauss_BiogasEval("/yourpath/",manager,estimated_parameters,stepsize_alpha)

--[[Generate a lua table with the parameters.
Note: The paramters are converted to double which means they lose
their accumulated error bounds.
--]]
tab={}
for k=0,estimated_parameters:len()-1 do
			name=estimated_parameters:get_name(k)
			value=estimated_parameters:get_param(k):get_value_as_double()
			tab[name]=value
end
print("Parameter values converted into lua table:")
print(tab)
print("Execution completed")
```

Figure 13 Sample main.lua for the Newton-Gauss solver.

If this code is inside a *main.lua*, the solver is run by

```
$ugshell -ex main.lua
```

from inside the folder where *main.lua* is located.

## Call the optimizer from C++

The following example shows all there is to call the solver. First, the parameters that one wants to optimize are defined. These must be idential to the one parsed in evaluate.lua. Then the solver is configured and run. Lastly, the found parameters are output. Note: The supplied path assumes to contain evaluate.lua, subset_sim.lua and subset_target.lua.

```
#include<iostream>
#include "yourpathtotheplugin/ConstrainedOptimization/core/parameter_estimation.h"
#include ".youtpathtotheplugin/ConstrainedOptimization/core/parameters.h"
#include <string>

int main(){
        co::EVar64Manager initial_vars;
        co::EVar64 alpha(co::EFloat64(1.47e-7),co::EFloat64(1e-8),co::EFloat64(2));
        co::EVar64 tau(co::EFloat64(0.089),co::EFloat64(1e-15),co::EFloat64(5));
        co::EVar64 theta(co::EFloat64(0.00155),co::EFloat64(1e-15),co::EFloat64(5));

        initial_vars.add("alpha",alpha); //scale factor for susceptible derivative
        initial_vars.add("tau",tau); //scale factor for recovered derivative
        initial_vars.add("theta",theta); //scale factor for death derivative

        co::NewtonOptions options;
        options.set_stepsize_alpha(1);
        std::string dir
="/yourpathtothemodel/ConstrainedOpticfglrsmization/examples/SIR_model/";

co::BiogasEvaluation<co::EFloat64,co::ConfigComputation::Local,co::ConfigOutput::File>
evaluator(dir, "subset_target.lua","subset_sim.lua");
        co::EVarManager<co::EFloat64> estimated_vars;
        co::NewtonOptimizer<decltype(evaluator)> solver(options,evaluator);

        solver.run(initial_vars,estimated_vars);
}
```

Figure 14 Starting Newton-Gauss solver from C++.

# Working example: Fitting disease data to an epidemics model

The following example fits Covid-19 disease data to an odinary and partial differential equations model used for epidemiological purposes. First, the real world dataset is presented. Then the epidemiological model trying to explain the real life dataset is described. Then, the data output of the epidemiological model is rearranged into a format that ConstrainedOptimization can understand.

Afterwards, all the necessary files for an optimization of parameters through ConstrainedOptimization are created and the optimization algorithms are run. Lastly, results are discussed. For the entire paper where this exact scheme was used please refer to [Covid Paper].

Note: The model described in the [Covid Paper] requires the UG4 software suite to be configured with ConvectionDiffusion and LIMEX plugins.

## Dataset

The real world data set can be found in the CollectedData directory in the *parameteroptimization* folder.

For the PDE model the real world dataset comprises of one file which contains Covid-19 data for 7 cities spanning a total of 40 days.

The ODE models from the Epidemics plugin use a dataset spanning a time period of 30 days and are aggregated values for a larger region.

Source for both the datasets:

https://github.com/jgehrcke/covid-19-germany-gae

## Epidemiological Model

Both the ODE and the PDE models are based on compartamental mathematical models.  These models divide the population in, as the name suggests, compartments and describes the dynamics of the population flow from one compartment to another.

5 Compartments:

- Susceptibles
- Exposed
- Infected
- Recovered
- Deceased

The flow of the population between the compartments is governed by parameters.

Parameters:

- **alpha** = Rate of infection. This parameter is responsible for the flow from Suceptibles to Exposed.
- **kappa** = Rate of transmission between Exposed and Infected.
- **qq** = Incubation time. This determines how long an exposed person stays in the Exposed group.
- **pp** = Duration of sickness.
- **theta=** Mortalitiy rate.

The Models which can be found in the Epidemics plugin are sovled using Runga-Kutta-4 or a linear implicit solver. The PDE formulation is discretized using Finite Difference.

The model from the [Covid Paper] was spatially discretized using Finite Volumes and temporally discretized with LIMEX.

These parameters are essential for the understanding and investigation of the dynamics of the spread of the disease.

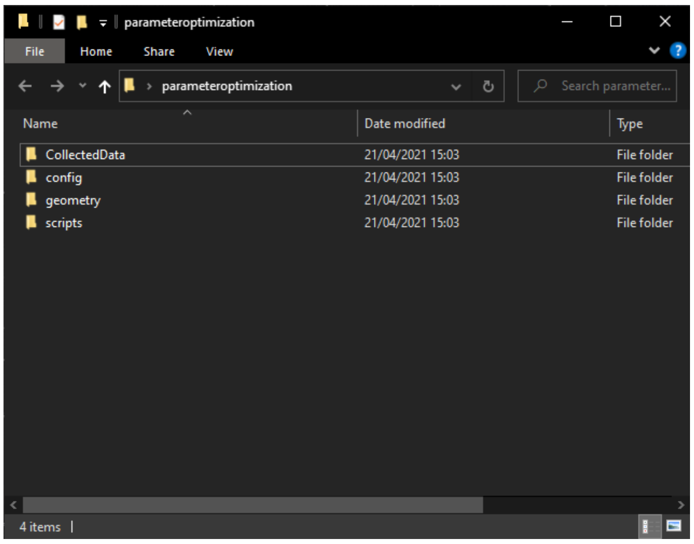
## Parameter Calibration with ConstrainedOptimization

Parameter calibration is an integral part of working with models. The parameters of the epidemiological models need to be fitted to the real world data. This allows researchers to test their models‘ correctness. Once decent values have been found, the epidemiological models can be used to forecast the spread of the disease.

ConstrainedOptimization (CO) reads the real data set and using different solvers tries to minimize the objection function (sum of squared errors, in this case). Hence the output of the model needs to be in the same format as the real world data.

The optimization process of using CO is as follows:

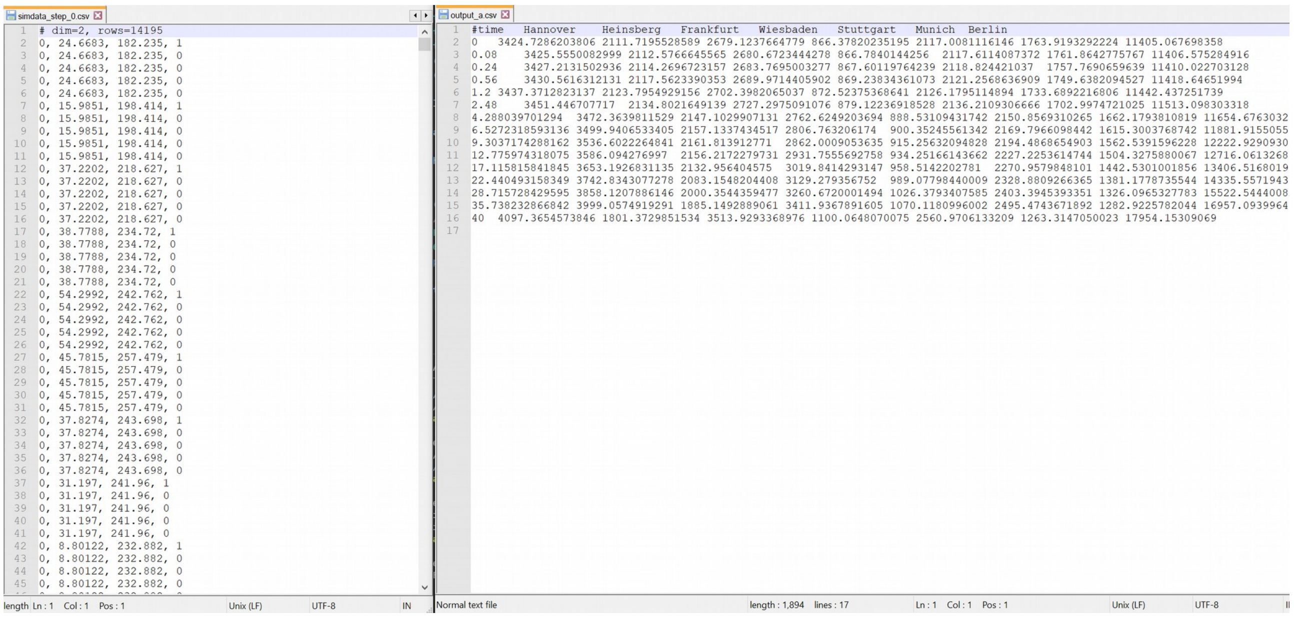
1.  Add the real dataset to the model folder which is going to be optimized. The first column of the file must be time and the first row of the file must start with ‚#‘ (column headers)



| Name | Date modified | Type |
|---|---|---|
| CollectedData | 21/04/2021 15:03 | File folder |
| config | 21/04/2021 15:03 | File folder |
| geometry | 21/04/2021 15:03 | File folder |
| scripts | 21/04/2021 15:03 | File folder |

*Figure 15: CollectedData contains the real world dataset. The other folders contain the required files for the epidemics model*

2. Depending on the model at hand, post processing of the model output is required. The [Covid Paper] model required heavy post processing, whereas the models in the Epidemics plugin do not.



```
# dim=2, rows=14195
0, 24.6683, 182.235, 1
0, 24.6683, 182.235, 0
0, 24.6683, 182.235, 0
0, 24.6683, 182.235, 0
0, 24.6683, 182.235, 0
0, 15.9851, 198.414, 1
0, 15.9851, 198.414, 0
0, 15.9851, 198.414, 0
0, 15.9851, 198.414, 0
0, 15.9851, 198.414, 0
0, 37.2202, 218.627, 1
0, 37.2202, 218.627, 0
0, 37.2202, 218.627, 0
0, 37.2202, 218.627, 0
0, 37.2202, 218.627, 0
0, 38.7788, 234.72, 1
0, 38.7788, 234.72, 0
0, 38.7788, 234.72, 0
0, 38.7788, 234.72, 0
0, 38.7788, 234.72, 0
0, 54.2992, 242.762, 1
0, 54.2992, 242.762, 0
0, 54.2992, 242.762, 0
0, 54.2992, 242.762, 0
0, 54.2992, 242.762, 0
0, 45.7815, 257.479, 1
0, 45.7815, 257.479, 0
0, 45.7815, 257.479, 0
0, 45.7815, 257.479, 0
0, 45.7815, 257.479, 0
0, 37.8274, 243.698, 1
0, 37.8274, 243.698, 0
0, 37.8274, 243.698, 0
0, 37.8274, 243.698, 0
0, 37.8274, 243.698, 0
0, 31.197, 241.96, 1
0, 31.197, 241.96, 0
0, 31.197, 241.96, 0
0, 31.197, 241.96, 0
0, 31.197, 241.96, 0
0, 8.80122, 232.882, 1
0, 8.80122, 232.882, 0
0, 8.80122, 232.882, 0
0, 8.80122, 232.882, 0
```

```
#time   Hannover    Heinsberg   Frankfurt   Wiesbaden   Stuttgart   Munich  Berlin
0   3424.7286203806 2111.7195528589 2679.1237664779 866.37820235195 2117.0081116146 1763.9193292224 11405.067698358
0.08    3425.5550082999 2112.5766645565 2680.6723444278 866.7840144256  2117.6114087372 1761.8642775767 11406.575284916
0.24    3427.2131502936 2114.2696723157 2683.7695003277 867.60119764239 2118.824421037  1757.7690659639 11410.022703128
0.56    3430.5616312131 2117.5623390353 2689.9714405902 869.23834361073 2121.2568636909 1749.6382094527 11418.64651994
1.2 3437.3712823137 2123.7954929156 2702.3982065037 872.52375368641 2126.1795114894 1733.6892216806 11442.437251739
2.48    3451.446707717  2134.8021649139 2727.2975091076 879.12236918528 2136.2109306666 1702.9974721025 11513.098303318
4.288039701294  3472.3639811529 2147.1029907131 2762.6249203694 888.53109431742 2150.8569310265 1662.1793810819 11654.6763032
6.5272318593136 3499.9406533405 2157.1337434517 2806.763206174  900.35245561342 2169.7966098442 1615.3003768742 11881.9155055
9.3037174288162 3536.6022264841 2161.813912771  2862.0009053635 915.25632094828 2194.4868654903 1562.5391596228 12222.9290930
12.775974318075 3586.094276997  2156.2172279731 2931.7555692758 934.25166143662 2227.2253614744 1504.3275880067 12716.0613268
17.115815841845 3653.1926831135 2132.956404575  3019.8414293147 958.5142202781  2270.9579848101 1442.5301001856 13406.5168019
22.440493158349 3742.8343077278 2083.1548204408 3129.279356752  989.07798440009 2328.8809266365 1381.1778735544 14335.5571943
28.715728429595 3858.1207886146 2000.3544359477 3260.6720001494 1026.3793407585 2403.3945393351 1326.0965327783 15522.5444008
35.738232866842 3999.0574919291 1885.1492889061 3411.9367891605 1070.1180996002 2495.4743671892 1282.9225782044 16957.0939964
40  4097.3654573846 1801.3729851534 3513.9293368976 1100.0648070075 2560.9706133209 1263.3147050023 17954.15309069
```

*Figure 16: unformatted data vs formatted data*

3. Then subset_sim.lua and subset_targe.lua are added. It must be noted that the column 0 in both files refers to the time column.

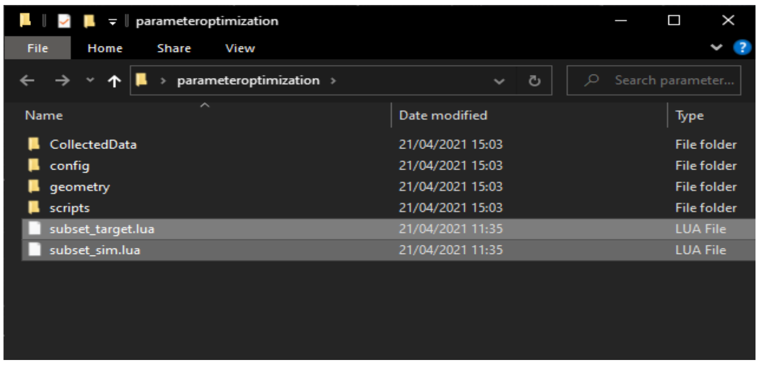
<u>subset_sim.lua</u>

This file describes the output files of the model and the columns of interest.

<u>subset_target.lua</u>

This file specifices the location and the columns of interest from the real world data.

The columns are read sequentially. So the order of the columns of the subset_target and subset_sim must correspond to each other. For dealing with multiple files for the real world data or output of the models, examples can be found in the Example folder. The idea remains the same, each colunm is read sequentially, one file after the next.



| Name | Date modified | Type |
|---|---|---|
| CollectedData | 21/04/2021 15:03 | File folder |
| config | 21/04/2021 15:03 | File folder |
| geometry | 21/04/2021 15:03 | File folder |
| scripts | 21/04/2021 15:03 | File folder |
| subset_target.lua | 21/04/2021 11:35 | LUA File |
| subset_sim.lua | 21/04/2021 11:35 | LUA File |

*Figure 17: subset_sim and subset_target go in the working directory*

4. Then the evaluate.lua is added. This file contains all the logic for the epidemics model with only a minor change. The parameters are loaded in via the output of ConstrainedOptimizations‘ iterations and the epidemics model simulation can be executed with new values for the parameters.

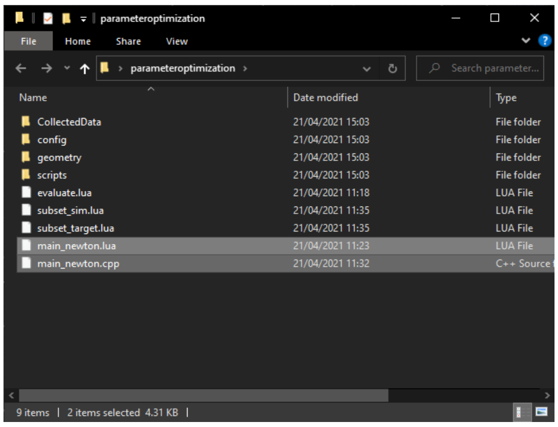
Evaluate.lua can be created in two ways. Either the evaluate.lua contains all the logic of the model, or once the parameters have been loaded, it calls another script that executes the model. Both examples can be found in the Examples folder.

5. The last step that remains is creating a main file that exectues and sets a few options for the solver chosen for the optimization procedure.

C++ files compilation Command:  g++ -O3 -pthread your/path/filename.cpp

Lua execution command: ugshell -ex your/path/filename.lua

<u>Note:</u> main.lua OR main.cpp are required but not both.

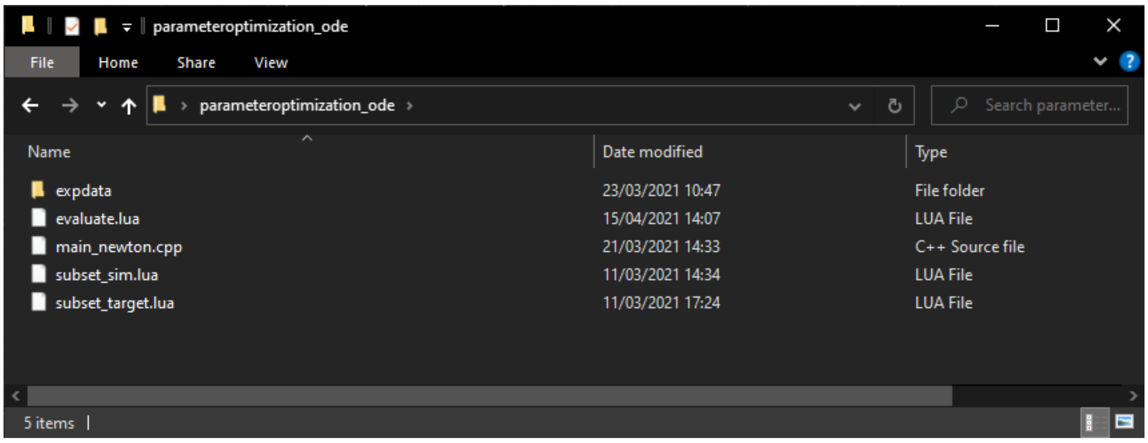


| Name | Date modified | Type |
|---|---|---|
| CollectedData | 21/04/2021 15:03 | File folder |
| config | 21/04/2021 15:03 | File folder |
| geometry | 21/04/2021 15:03 | File folder |
| scripts | 21/04/2021 15:03 | File folder |
| evaluate.lua | 21/04/2021 11:18 | LUA File |
| subset_sim.lua | 21/04/2021 11:35 | LUA File |
| subset_target.lua | 21/04/2021 11:35 | LUA File |
| main_newton.lua | 21/04/2021 11:23 | LUA File |
| main_newton.cpp | 21/04/2021 11:32 | C++ Source |

9 items | 2 items selected 4.31 KB |

*Figure 18: evaluate.lua and the main files also go in the working directory*

*Figure 19: The ODE model is defined in Epidemics plugin. In evalute.lua, by calling the right functions the model can be run. This figure, like Figure 3 shows a complete setup for an optimization problem.*

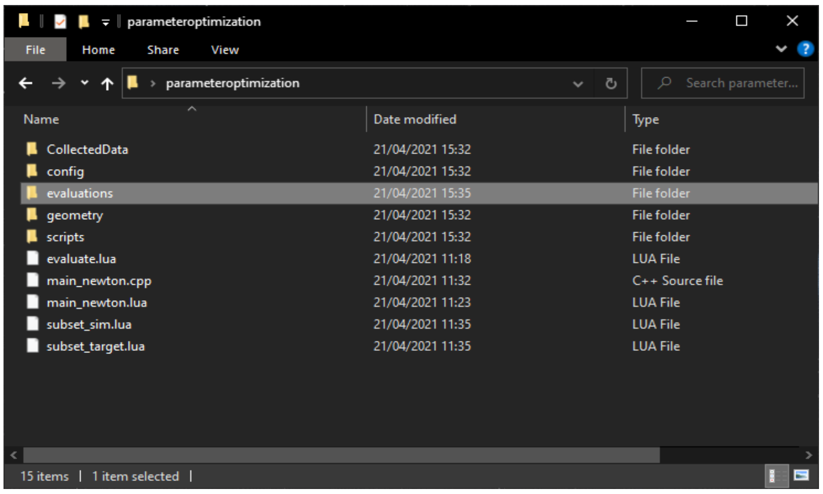


| Name | Date modified | Type |
|---|---|---|
| expdata | 23/03/2021 10:47 | File folder |
| evaluate.lua | 15/04/2021 14:07 | LUA File |
| main_newton.cpp | 21/03/2021 14:33 | C++ Source file |
| subset_sim.lua | 11/03/2021 14:34 | LUA File |
| subset_target.lua | 11/03/2021 17:24 | LUA File |

5 items |

# Folder Structure of the Output

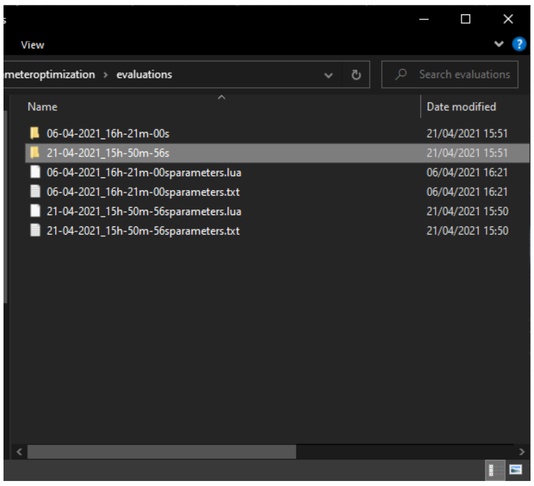
Once the main files are executed, the output of the Optimization procedure is written to file in an automatically generated folder called **‘evaluations’**.



parameteroptimization

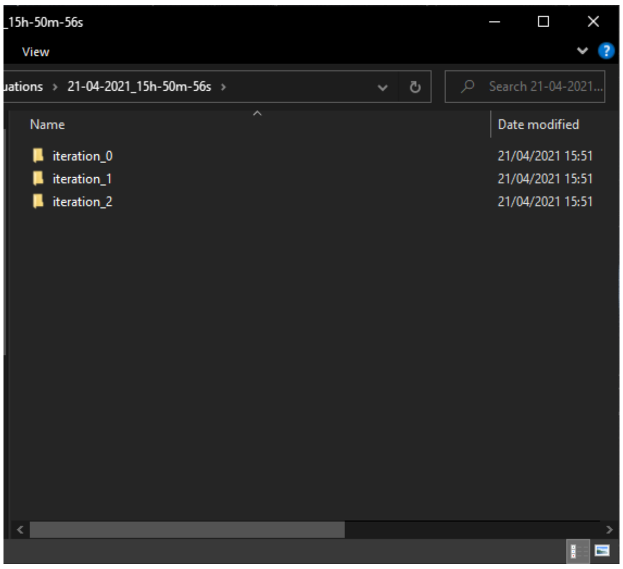
| Name | Date modified | Type |
|---|---|---|
| CollectedData | 21/04/2021 15:32 | File folder |
| config | 21/04/2021 15:32 | File folder |
| evaluations | 21/04/2021 15:35 | File folder |
| geometry | 21/04/2021 15:32 | File folder |
| scripts | 21/04/2021 15:32 | File folder |
| evaluate.lua | 21/04/2021 11:18 | LUA File |
| main_newton.cpp | 21/04/2021 11:32 | C++ Source file |
| main_newton.lua | 21/04/2021 11:23 | LUA File |
| subset_sim.lua | 21/04/2021 11:35 | LUA File |
| subset_target.lua | 21/04/2021 11:35 | LUA File |

15 items | 1 item selected |



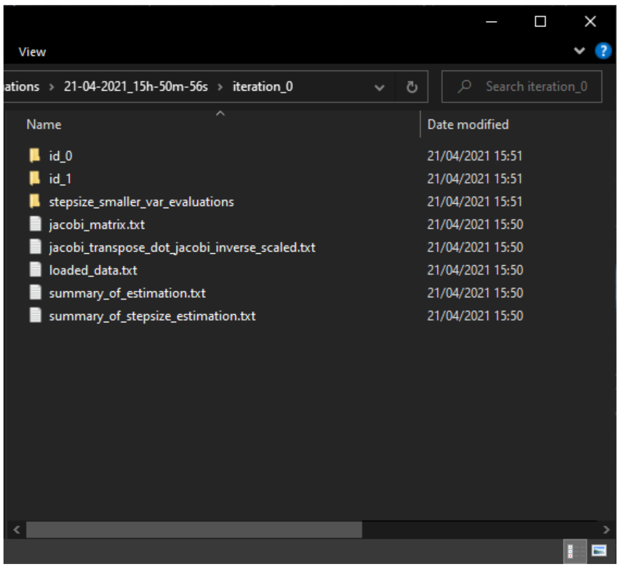
meteroptimization › evaluations

| Name | Date modified |
|---|---|
| 06-04-2021_16h-21m-00s | 21/04/2021 15:51 |
| 21-04-2021_15h-50m-56s | 21/04/2021 15:51 |
| 06-04-2021_16h-21m-00sparameters.lua | 06/04/2021 16:21 |
| 06-04-2021_16h-21m-00sparameters.txt | 06/04/2021 16:21 |
| 21-04-2021_15h-50m-56sparameters.lua | 21/04/2021 15:50 |
| 21-04-2021_15h-50m-56sparameters.txt | 21/04/2021 15:50 |



uations › 21-04-2021_15h-50m-56s ›

| Name | Date modified |
|---|---|
| iteration_0 | 21/04/2021 15:51 |
| iteration_1 | 21/04/2021 15:51 |
| iteration_2 | 21/04/2021 15:51 |



ations › 21-04-2021_15h-50m-56s › iteration_0

| Name | Date modified |
|---|---|
| id_0 | 21/04/2021 15:51 |
| id_1 | 21/04/2021 15:51 |
| stepsize_smaller_var_evaluations | 21/04/2021 15:51 |
| jacobi_matrix.txt | 21/04/2021 15:50 |
| jacobi_transpose_dot_jacobi_inverse_scaled.txt | 21/04/2021 15:50 |
| loaded_data.txt | 21/04/2021 15:50 |
| summary_of_estimation.txt | 21/04/2021 15:50 |
| summary_of_stepsize_estimation.txt | 21/04/2021 15:50 |

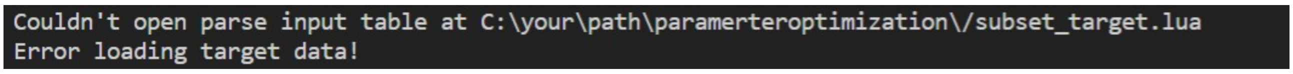
The images above depict the general layout of the evaluations folder.

- Each subfolder, with the date and timestamp as the name, in the evaluation directory contains the output for different simulations.
- The simulation output of each iteration is written to file.
- Each iteration contains *id_** folders. The number of *id_** folders generated depends on the number of active threads used in the current iteration.
- Inside the *id_** folders, there are folders called *eval**. These show the workload per thread.
- The last *id_** folder has the output of the model simulation with the unchanged parameter values and the others contain the derivative values.

- Based on the output of the *id_** folders a *loaded_data.txt* is generated. This is the model data of the last *eval_** of the last *id_** that is loaded into the optimization procedure. It can be used to verify that the correct values have been loaded.
- The *jacobi_matrix.txt* and the *jacobi_transpose_dot_jacobi_inverse.txt*, as the name suggests, are the Jacobian and the pseudoinverse of the Jacobian for the Gauss-Newton method.
- *Stepsize_smaller_var_evaluations* contains the output generated while the optimization procedure tries different step sizes. The results of the step size finding routine is then written in the *summary_of_stepsize_estimation.txt.*
- The results of the iteration are summarized in the *summary_of_estimation.txt.*

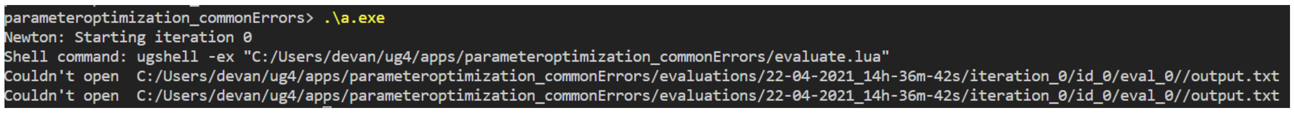
## Common Errors

- Path error. This error occurs when the directory specified in the main file is wrong or the subset_target.lua is not in the directory.

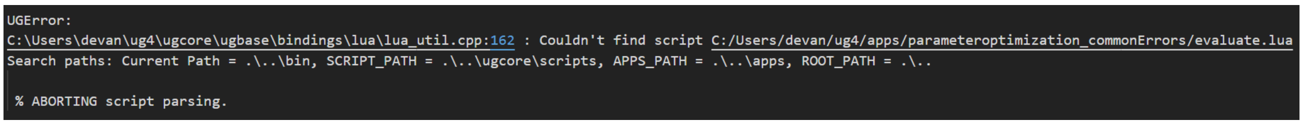


```
Couldn't open parse input table at C:\your\path\paramerteroptimization\/subset_target.lua
Error loading target data!
```

- If the evaluate.lua is not found in the working directory, error looks like the following image. As the error states that the output files were not found, the console_output.log will contain descriptive error, as shown in the following image.

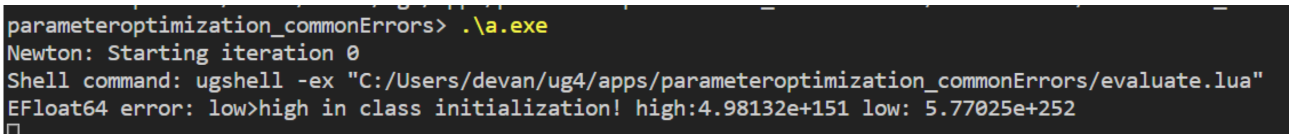


```
parameteroptimization_commonErrors> .\a.exe
Newton: Starting iteration 0
Shell command: ugshell -ex "C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluate.lua"
Couldn't open  C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluations/22-04-2021_14h-36m-42s/iteration_0/id_0/eval_0//output.txt
Couldn't open  C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluations/22-04-2021_14h-36m-42s/iteration_0/id_0/eval_0//output.txt
```



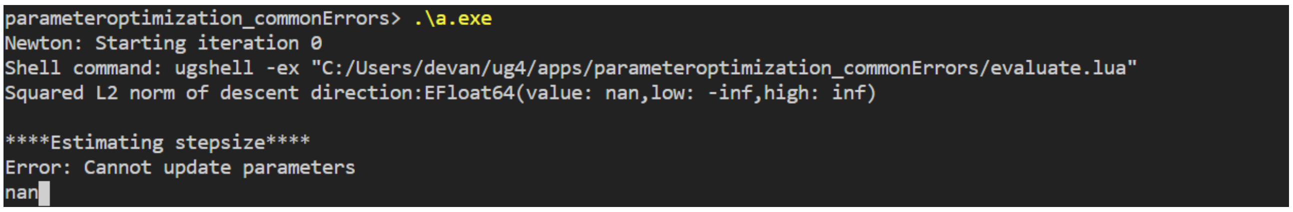
```
UGError:
C:\Users\devan\ug4\ugcore\ugbase\bindings\lua\lua_util.cpp:162 : Couldn't find script C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluate.lua
Search paths: Current Path = .\..\bin, SCRIPT_PATH = .\..\ugcore\scripts, APPS_PATH = .\..\apps, ROOT_PATH = .\..

 % ABORTING script parsing.
```

- If the numbers of columns in the subset files do not match, then the console output will look like this:



```
parameteroptimization_commonErrors> .\a.exe
Newton: Starting iteration 0
Shell command: ugshell -ex "C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluate.lua"
EFloat64 error: low>high in class initialization! high:4.98132e+151 low: 5.77025e+252
```

- Depending on the initial values, the bounds on the parameters and the finite differences stepsize, the following error could be generated. The finite differences stepsize accounts for the incremental changes in the parameter values, and if it is too big, the values of the parameters cross the bounds and are invalid.



```
parameteroptimization_commonErrors> .\a.exe
Newton: Starting iteration 0
Shell command: ugshell -ex "C:/Users/devan/ug4/apps/parameteroptimization_commonErrors/evaluate.lua"
Squared L2 norm of descent direction:EFloat64(value: nan,low: -inf,high: inf)

****Estimating stepsize****
Error: Cannot update parameters
nan
```

The above error can also be produced when a values in the *loaded_data, jacobi, jacobi_inverse* are nans. Not a number errors generally indicate that the model simulation outputs are not correct. It is advisable to check the values of the respective output files and compare them to the values of the *loaded_data.txt* file.