GHEtool

Release 2.0.5

Tobias Blanke

Oct 12, 2022
CONTENTS:

1 GHEtool: An open-source tool for borefield sizing in Python 1
  1.1 What is GHEtool? ................................................. 1
  1.2 Requirements .................................................. 1
  1.3 Quick start ...................................................... 2
  1.4 Functionalities .................................................. 3
  1.5 Precalculated data ............................................... 4
  1.6 License .......................................................... 5
  1.7 Contributing to GHEtool ........................................ 5
  1.8 Main contributors ............................................... 5
  1.9 Citation .......................................................... 5
  1.10 References ...................................................... 5
GHETOOL: AN OPEN-SOURCE TOOL FOR BOREFIELD SIZING IN PYTHON

1.1 What is GHETOOL?

GHETOOL is a Python package that contains all the functionalities needed to deal with borefield design. It is developed for both researchers and practitioners. The core of this package is the automated sizing of borefield under different conditions. The sizing of a borefield is typically slow due to the high complexity of the mathematical background. Because this tool has a lot of precalculated data (cf. infra), GHETOOL can size a borefield in the order of tenths of milliseconds. This sizing typically takes the order of minutes. Therefore, this tool is suited for being implemented in workflows where iterations are required.

1.1.1 Graphical user interface

GHETOOL also comes with a graphical user interface (GUI). This GUI is prebuilt as an exe-file (only for Windows platforms currently) because this provides access to all the functionalities without coding. A setup to install the GUI at the user-defined place is also implemented and available here. This graphical interface is made by Tobias Blanke from FH Aachen.

1.2 Requirements

This code is tested with Python 3.8 and requires the following libraries (the versions mentioned are the ones with which the code is tested)

- Numpy (>=1.20.2)
- Scipy (>=1.6.2)
- Matplotlib (>=3.4.1)
- Pygfunction (>=2.1.0)
- Openpyxl (>=3.0.7)
- Pandas (>=1.2.4)

For the GUI

- PyQt5 (>=5.10)

For the tests
• Pytest (>=7.1.2)

When working with Python 3.9 and higher, installing a newer version of pygfunction (>=2.1.0) can lead to problems
due to the fact that its dependency CoolProp is not compatible with Python 3.9 and higher (see also https://github.com/
the newer version of pygfunction and with Python 3.9 or higher, one can install a development version of CoolProp
using

```
pip install --extra-index-url https://test.pypi.org/simple/ CoolProp==6.4.2.dev0
```

### 1.3 Quick start

#### 1.3.1 Installation

One can install GHEtool by running Pip and running the command

```
pip install GHEtool
```

or one can install a newer development version using

```
pip install --extra-index-url https://test.pypi.org/simple/ GHEtool
```

Developers can clone this repository.

It is a good practise to use virtual environments (venv) when working on a (new) Python project so different Python
and package versions don’t conflict with eachother. For GHEtool, Python 3.8 is recommended. General information
about Python virtual environments can be found here and in this article.

#### 1.3.2 Check installation

To check whether everything is installed correctly, run the following command

```
pytest --pyargs GHEtool
```

This runs some predefined cases to see whether all the internal dependencies work correctly. 9 test should pass suc-
cessfully.

#### 1.3.3 Get started with GHEtool

To get started with GHEtool, one needs to create a Borefield object. This is done in the following steps.

```
from GHEtool import Borefield, GroundData
```

After importing the necessary classes, one sets all the relevant ground data.

```
data = GroundData(110, # depth of the field (m)
6,  # distance between the boreholes (m)
3,  # ground thermal conductivity (W/mK)
10,  # initial/undisturbed ground temperature (deg C)
0.2, # borehole equivalent resistance (mK/W)
10,  # number of boreholes in width direction of the field (/)
```

(continues on next page)
Furthermore, one needs to set the peak and monthly baseload for both heating and cooling.

```python
peak_cooling = [0., 0, 34., 69., 133., 187., 213., 240., 160., 37., 0., 0.] # Peak cooling in kW
peak_heating = [160., 142, 102., 55., 0., 0., 0., 0., 40.4, 85., 119., 136.] # Peak heating in kW
monthly_load_heating = [46500.0, 44400.0, 37500.0, 29700.0, 19200.0, 0.0, 0.0, 0.0, 18300.0, 26100.0, 35100.0, 43200.0] # in kWh
monthly_load_cooling = [4000.0, 8000.0, 8000.0, 8000.0, 12000.0, 16000.0, 32000.0, 32000.0, 0, 16000.0, 12000.0, 8000.0, 4000.0] # in kWh
```

Next, one creates the borefield object and sets the temperature constraints and the ground data.

```python
# create the borefield object
borefield = Borefield(simulation_period=20,
                      peak_heating=peak_heating,
                      peak_cooling=peak_cooling,
                      baseload_heating=monthly_load_heating,
                      baseload_cooling=monthly_load_cooling)

borefield.set_ground_parameters(data)

# set temperature boundaries
borefield.set_max_ground_temperature(16) # maximum temperature
borefield.set_min_ground_temperature(0) # minimum temperature
```

Once a Borefield object is created, one can make use of all the functionalities of GHEtool. One can for example size the borefield using:

```python
depth = borefield.size(100)
print("The borehole depth is: ", depth, "m")
```

Or one can plot the temperature profile by using

```python
borefield.print_temperature_profile(legend=True)
```

A full list of functionalities is given below.

## 1.4 Functionalities

GHEtool offers functionalities of value to all different disciplines working with borefields. The features are available both in the code environment and in the GUI. These functions are listed in the table below, alongside with a link to an example document where one can find how these functionalities can be used.
### Functionality

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Example document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sizing the borefield (i.e. calculating the required depth) for a given injection and extraction load for the borefield (three sizing methods are available).</td>
<td>main_functionalities.py</td>
</tr>
<tr>
<td>Calculating the temperature evolution of the ground for a given building load and borefield configuration</td>
<td>main_functionalities.py</td>
</tr>
<tr>
<td>Using dynamically calculated borehole thermal resistance (this is directly based on the code of pygfunction)</td>
<td>sizing_with_Rb_calculation.py</td>
</tr>
<tr>
<td>Optimising the load profile for a given heating and cooling load</td>
<td>optimise_load_profile.py</td>
</tr>
<tr>
<td>Finding the optimal rectangular borefield configuration for a given heating and cooling load</td>
<td>size_borefield_by_length_and_width.py</td>
</tr>
<tr>
<td>Importing heating and cooling loads from .csv and .xlsx files</td>
<td>import_data.py</td>
</tr>
<tr>
<td>Using your custom borefield configuration</td>
<td>custom_borefield_configuration.py</td>
</tr>
</tbody>
</table>

### Comparisons

<table>
<thead>
<tr>
<th>Comparisons</th>
<th>Example document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison of different sizing methods (L2, L3) for different random profiles</td>
<td>sizing_method_comparison.py</td>
</tr>
<tr>
<td>Comparison in calculation time and accuracy between using the precalculated gfunction data or not</td>
<td>speed_comparison.py</td>
</tr>
<tr>
<td>Comparison of different sizing methods (L2, L3 and L4) for the same hourly profile</td>
<td>sizing_method_comparison_L2_L3_L4.py</td>
</tr>
<tr>
<td>Comparison in calculation time and accuracy between the simplified L2 sizing methodology and the more accurate L3 method.</td>
<td>sizing_method_comparison.py</td>
</tr>
<tr>
<td>Comparison of Rb* calculation between GHEtool and EED.</td>
<td>validation_effective_borehole_thermal_resistance.py</td>
</tr>
</tbody>
</table>

### 1.5 Precalculated data

This tool comes with precalculated g-functions for all borefields of type nxm (for $0 < n, m < 21$) for which the boreholes are connected in parallel. For these borefield configurations, the g-functions are calculated for different depth-thermal diffusivity-spacing combinations. The ranges are:

- **Depth**: 25 - 350m in increments of 25m
- **Thermal diffusivity of the soil** (defined as thermal conductivity / volumetric heat capacity): 0.036 - 0.144m$^2$/day in increments of 0.018m$^2$/day (This is equal to a range of thermal conductivity from 1-4W/mK with a constant volumetric heat capacity of 2.4MJ/m$^3$K)
- **Spacings (equal)**: 3 - 9m in increments of 1m

Here a burial depth (D) of 4.0m is assumed even as a borehole radius of 7.5cm for all the precalculated data.

It is possible to calculate your own dataset to your specific project based on the pygfunction tool and use this one in the code.
1.6 License

GHEtool is licensed under the terms of the 3-clause BSD-license. See GHEtool license.

1.7 Contributing to GHEtool

You can report bugs and propose enhancements on the issue tracker. If you want to add new features and contribute to the code, please contact Wouter Peere (wouter.peere@kuleuven.be).

1.8 Main contributors

Wouter Peere, KU Leuven & boydens engineering (part of Sweco), wouter.peere@kuleuven.be
Tobias Blanke, Solar-Institute Jülich, FH Aachen, blanke@sij.fh-aachen.de

1.9 Citation

Please cite GHEtool using the JOSS paper.

1.10 References

1.10.1 Development of GHEtool


1.10.2 Applications/Mentions of GHEtool

Getting started

In the following, instructions for installing and using the tssm package on Windows are given. The installation instructions for installing and using tssm on Linux/macOS systems are however quite similar and can be, hopefully easily, derived from the instructions below.

GHEtool installation

You can also directly install tssm package from PyPi with the following command:

```
pip install GHEtool
```

Alternatively, download or clone a local copy of the repository to your computer:

```
git clone https://github.com/wouterpeere/GHEtool
```

And afterwards install (in the folder where the setup.py is located) as editable using:

```
pip install -e .
```

or install directly via python as

```
python setup.py install
```

Overview over scaling approaches

The different approaches available are:

- The linear approach scaling the original time series to their average using either a given scaling factor or a determined one calculated by the given simultaneity factor. Further explanations can be found here: Linear approach

Linear approach

The linear approach is scaling the original profile (\(P_{original}^t\)) of the current time step (\(t\)) to the mean value for the selected period (\(P_{mean}^f(t)\)). It is linearly interpolated using the scaling factor (\(S_f(t)\)). Therefore, the scaling factor and average value have to be assigned to the current time step. This is illustrated by \(f(t)\). For Example if a monthly period is considered the average value array has 12 entries. So one of these 12 entries has to be assigned to the current time step. All time steps in january are referred to the first entry, all time steps in february to the second entry, and so on.

\[
P_{new}^t = P_{original}^t \cdot S_f(t) + P_{mean}^f(t) \cdot \left(1 - S_f(t)\right)
\]

The scaling factor (\(S^k\)) can either be provided by the user or calculated based on the simultaneity factor (\(SF\)). Therefore the maximum of the original values (\(P_{original}\)) for all time steps in the current period (\(k\), \(A\)) and the average value have to be determined.

\[
S^k = \frac{SF \cdot \max(P_{original}^i \forall i \in A) - P_{original}^i \forall i \in A}{\max(P_{original}^i \forall i \in A) - P_{original}^i \forall i \in A}
\]

As periods can be selected a daily (1), a weekly (2), a monthly (3) or a yearly (4) one. An example code is shown below:
Graphical Using Interface (GUI)

How to add new options in the GUI?

New options can be added in the \GHEtool\gui\gui_structure.py script. The added options will be automatically integrated in the Datastorage and the GUI. The following sections will explain the gui structure and the different options which can be added.

The implemented options can then be used in the CalcProblem class in the \GHEtool\gui\gui_calculation_thread.py script. The value of the option get be get by the get_value() function.

GUI structure

The GUI is based on pages which consists of categories which consists of options. An example for a page is the borehole resistance page. Where the fluid data category can be found. This category has a double spin box option to set the mass flow rate. The order in which the options are created is also the tab order.

Page

To create a page the Page class has to be imported from \GHEtool\gui\gui_classes.py. Then a default widget parent has to be set. This can be just copied from the previous pages or set to default_parent. As second option the name of the page can be set. In the example below Example page. The third option is the button name for the page. In the example below Name of the button. The \n can be used to create a new line. The last option is the icon. In this case :/icons/icons/example_icon.svg. An explanation how to add an icon can be found here: How to add an icon?. Furthermore, the next and previous page can be set by using set_previous_page() or set_next_page().

```python
from GHEtool.gui.gui_classes import Page
page_example = Page(
    name='Example page',
    button_name='Name of the button',
    icon=':/icons/icons/example_icon.svg',
)
page_example.set_previous_page(page_previous)
page_example.set_next_page(page_next)
```

Example Page sources/_static/Example_Page.png
**Aim**

To create an aim the Aim class has to be imported from `GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous aims or set to `default_parent`. As second option the name of the aim can be set. In the example below `Example aim`. The third option is the icon. In this case `:/icons/icons/example_icon.svg`. An explanation how to add an icon can be found here: *How to add an icon?*. The last option is the page where the aim should be located. In this case `page_aim`. An option, which should be show and disappear if the aim is selected or not can be added using `add_link_2_show()`. This can be an option, hint, category or function button.

```python
from GHEtool.gui.gui_classes import Aim
aim_example = Aim(
    label='Example aim',
    icon=':/icons/icons/example_icon.svg',
    page=page_aim,
)
aim_example.add_link_2_show(option=option_example)
```

**Category**

To create a category the Category class has to be imported from `GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous categories or set to `default_parent`. As second option the name of the category can be set. In the example below `Example category`. The last option is the page where the category should be located. In this case `page_example`.

```python
from GHEtool.gui.gui_classes import Category
category_example = Category(
    label='Example category',
    page=page_example,
)
```

8 Chapter 1. GHEtool: An open-source tool for borefield sizing in Python
Float box

To create a float box the FloatBox class has to be imported from .\GHEtool\gui\gui_classes.py. Then a default widget parent has to be set. This can be just copied from the previous options or set to default_parent. As second option the name of the float box can be set. In the example below Float label text. The next option is a default value. In this case 0.5. The next option is the category which should contain the option. In this case category_example. The next option is the decimal position (0=1, 4=1.2345). In this case 2. The next option is a minimal value. In this case 0. The next option is a maximal value. In this case 1. The next option is a step value in which the value is increased if the arrows of the box are used. In this case 0.1. The function add_link_2_show() can be used to couple the float value to other options, hints, function buttons or categories. So in the example option_linked will be shown if the float value is below 0.1 or above 0.9.

```
from GHEtool.gui.gui_classes import FloatBox
option_float = FloatBox(
    label='Float label text',
    default_value=0.5,
    category=category_example,
    decimal_number=2,
    minimal_value=0,
    maximal_value=1,
    step=0.1,
)
option_float.add_link_2_show(option=option_linked, below=0.1, above=0.9)
```

![Example float box](sources/_static/Example_Float_Box.png)

Integer box

To create a integer box the IntegerBox class has to be imported from .\GHEtool\gui\gui_classes.py. Then a default widget parent has to be set. This can be just copied from the previous options or set to default_parent. As second option the name of the integer box can be set. In the example below Int label text. The next option is a default value. In this case 2. The next option is the category which should contain the option. In this case category_example. The next option is a minimal value. In this case 0. The next option is a maximal value. In this case 12. The next option is a step value in which the value is increased if the arrows of the box are used. In this case 2. The function add_link_2_show() can be used to couple the integer value to other options, hints, function buttons or categories. So in the example option_linked will be shown if the integer value is below 1 or above 10.

```
from GHEtool.gui.gui_classes import IntBox
option_int = IntBox(
    label='Int label text',
    default_value=2,
    category=category_example,
    minimal_value=0,
    maximal_value=12,
    step=2,
)
option_int.add_link_2_show(option=option_linked, below=1, above=10)
```
**Button box**

To create a button box the ButtonBox class has to be imported from `\GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous options or set to `default_parent`. As second option the name of the button box can be set. In the example below `Button box label text`. The next option is a default index. In this case `0`. The next option are the entries. In this case `Option 1`, `Option 2`. The next option is the category which should contain the option. In this case `category_example`. The function `add_link_2_show()` can be used to couple the selected index to other options, hints, function buttons or categories. So in the example `option_linked` will be shown if the first (0) option is selected.

```python
from GHEtool.gui.gui_classes import ButtonBox

option_buttons = ButtonBox(
    label='Button box label text',
    default_index=0,
    entries=['Option 1', 'Option 2'],
    category=category_example,
)

option_buttons.add_link_2_show(option=option_linked, on_index=0)
```

**List box**

To create a list box the ListBox class has to be imported from `\GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous options or set to `default_parent`. As second option the name of the list box can be set. In the example below `List box label text`. The next option is a default index. In this case `0`. The next option are the entries. In this case `Option 1`, `Option 2`. The next option is the category which should contain the option. In this case `category_example`. The function `add_link_2_show()` can be used to couple the selected index to other options, hints, function buttons or categories. So in the example `option_linked` will be shown if the first (0) option is selected.

```python
from GHEtool.gui.gui_classes import ListBox

option_list = ListBox(
    label='List box label text',
    default_index=0,
    entries=['Option 1', 'Option 2'],
    category=category_example,
)

option_list.add_link_2_show(option=option_linked, on_index=0)
```
Filename

To create a filename box the FileNameBox class has to be imported from `GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous file name boxes or set to `default_parent`. As second option the name of the file name box can be set. In the example below `File name box label text`. The next option is a default filename. In this case `example_file.XX`. The next option is the dialog text which will be shown if the button to select a file is selected. In this case `Choose *.XX file`. The next option is the error message if no file is found/selected. In this case `no file found`. This error message will be displayed in the status bar so the status bar object has to be provided. In this example `status_bar`. The last option is the category which should contain the option. In this case `category_example`.

```python
from GHEtool.gui.gui_classes import FileNameBox
option_file = FileNameBox(
    label='File name box label text',
    default_value='example_file.XX',
    dialog_text='Choose *.XX file',
    error_text='no file found',
    status_bar=status_bar,
    category=category_example,
)
```

Function button

To create a function button the FunctionButton class has to be imported from `GHEtool\gui\gui_classes.py`. Then a default widget parent has to be set. This can be just copied from the previous file function buttons or set to `default_parent`. As second option the name of the button text can be set. In the example below `Press Here to activate function`. The next option is the icon. In this case `:/icons/icons/example_icon.svg`. An explanation how to add an icon can be found here: How to add an icon?. The last option is the category which should contain the option. In this case `category_example`. The button can be linked to a function using `change_event()`. In this case every time the button is clicked the `function_to_be_called()` is activated.

```python
from GHEtool.gui.gui_classes import FunctionButton
function_example = FunctionButton(
    button_text='Press Here to activate function',
    icon=':/icons/icons/example_icon.svg',
    category=category_example,
)
function_example.change_event(function_to_be_called())
```
Hint

To create a hint the Hint class has to be imported from .\GHEtool\gui\gui_classes.py. Then a default widget parent has to be set. This can be just copied from the previous hints or set to default_parent. As second option the hint can be set. In the example below This is a hint to something important.. The next option is the category which should contain the option. In this case category_example. The last option is a boolean to set if the hint is a warning or not. In this case it is (True). So the hint will be display in a yellow and not white text color.

```python
from GHEtool.gui.gui_classes import Hint

hint_example = Hint(
    hint='This is a hint to something important.',
    category=category_example,
    warning=True,
)

function_example.change_event(function_to_be_called())
```

How to add or correct translations?

Translations for the GUI can be added in a new column in the .\GHEtool\gui\Translations.csv file. Correction can be made there as well. Important is that the separator in the CSV-file is not comma but semicolon ;. Also, an icon and shortcut can be linked there. An explanation how to add an icon can be found here: How to add an icon?. The name of the variable to be translated is shown in column 1 and afterwards the different translations. The variable’s name is the name in the .\GHEtool\gui\gui_structure.py or .\GHEtool\gui\gui_window.py script. Options with multiple inputs are created by separated the different inputs with comma ,. So for an ListBox this can seem like this:

```text
option_list;Option of list,First Option,Second Option;Optionen der Liste,Erste Option, Zweite Option
```

Afterwards the .\GHEtool\gui\translation_csv_to_py.py script needs to be run. This will add the changes to the Translations class in ./GHEtool/gui/translation_class.py.

Know the translation or correction is available in the GUI.
How to add an icon?

An Icon can be added to the gui by adding `example_icon.svg` icon to `icons.qrc` file and locating in the icon in the icons folder:

```
<file>icons/example_icon.svg</file>
```

Afterwards the `icons_rc.py` has to be recompiled using the following command line:

```
pyside6-rcc ./GHEtool/gui/icons.qrc -o ./GHEtool/gui/icons_rc.py
```

Know the icon can be used in the GUI.

How to create the *.exe file?

The exe can be created using PyInstaller.

The following line will create a windowed version of the executable:

```
python -m PyInstaller --noconfirm --onefile --windowed --splash "./GHEtool/gui/icons/Icon.ico" --name "GHEtool" --icon "./GHEtool/gui/icons/Icon.ico" ./GHEtool/gui/start_gui.py
```

The following line will create a version which also displays a windows console with error messages of the executable.

```
python -m PyInstaller --noconfirm --onefile --console --splash "./GHEtool/gui/icons/Icon.ico" --name "GHEtool_with_command_line" --icon "./GHEtool/gui/icons/Icon.ico" ./GHEtool/gui/start_gui.py
```

GHEtool's Change Log

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog.

unreleased

Added

- Documentation with ReadTheDocs
- GUI Documentation
- Changelog
- New features in the GUI
Changed

- GUI workflow to be simpler

2.0.5 - 2022-08-31

Added

- Hourly sizing method (L4) is implemented
- Hourly plotting method
- Volumetric heat capacity is included in the ground data

Changed

- Implemented numpy arrays everywhere
- Implemented convolution instead of matrix multiplication
- New implementation for L3 sizing

Fixed

- No more problems with iteration (L2) and sub 1m depth fields
- Fixed bug in main_functionalities example

Varia

- New validation document for the effective thermal borehole resistance, comparison with EED

2.0.4 - 2022-08-17

Fixed

- Final JOSS paper update

2.0.3 - 2022-08-12

Added

- Variable ground temperature
- Sizing with dynamic Rb*
Fixed

• General bug fixes

Changed

• Sizing setup with more streamlined sizing options

2.0.2 - 2022-06-12

Added

• Included a function (and example) on sizing a borefield by length and width

2.0.1 - 2022-06-12

Added

• Included a pytest document to check if package is correctly installed

2.0.0 - 2022-04-01

Added

• GUI
  • Borehole thermal resistance (based on the pygfunction package)

Changed

• More documentation and examples

1.0.1 - 2021-12-11

Changed

• longer simulation period up to 100 years
Fixed

- fixed bug in interpolation

Legal Notice

BSD 3-Clause License

Copyright (c) 2022, Wouter MC Peere All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.