

---

# **MOUSE Documentation**

***Release 0.3***

**M. Yetisir**

**Dec 31, 2016**



## CONTENTS:

<b>1</b>	<b>User Manual</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Installation Guide . . . . .	1
1.2.1	Installing python interpreter and dependencies . . . . .	1
1.2.2	Installing MOUSE . . . . .	1
1.3	Basic Usage . . . . .	1
1.4	Basic Settings . . . . .	1
<b>2</b>	<b>MOUSE Reference Manual</b>	<b>3</b>
2.1	Overview . . . . .	3
2.2	MOUSE Usage . . . . .	3
2.3	MOUSE Documentation . . . . .	4
2.4	Modules Documentation . . . . .	5
2.4.1	Modules.Module_ABAQUS module . . . . .	5
2.4.2	Modules.Module_HODS module . . . . .	6
2.4.3	Modules.Module_OSTRICH module . . . . .	8
2.4.4	Modules.Module_UDEC module . . . . .	10
2.4.5	Modules.Base module . . . . .	13
<b>3</b>	<b>HODS Reference Manual</b>	<b>19</b>
3.1	Modules.HODS package . . . . .	19
3.1.1	Submodules . . . . .	19
3.1.2	Modules.HODS.HODS module . . . . .	19
3.1.3	Module contents . . . . .	20
<b>4</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>



## **USER MANUAL**

### **1.1 Overview**

### **1.2 Installation Guide**

#### **1.2.1 Installing python interpreter and dependencies**

#### **1.2.2 Installing MOUSE**

### **1.3 Basic Usage**

### **1.4 Basic Settings**



## MOUSE REFERENCE MANUAL

### 2.1 Overview

### 2.2 MOUSE Usage

MOUSE: An Up-Scaling Utility for DEM Simulations

```
usage: MOUSE [-h] {UDEEC,HODS,OSTRICH,ABAQUS} . . .
```

#### Sub-commands:

**UDEEC** Undocumented

```
usage: MOUSE UDEEC [-h] -n NAME
```

#### Options:

**-n, --name** Name of the file containing the model data without the extension

**HODS** Undocumented

```
usage: MOUSE HODS [-h] -n NAME [-x REVX] [-y REVY] [-r REVRADIUS]
```

#### Options:

**-n, --name** Name of the file containing the model data without the extension

**-x, --revX** x coordinate of REV centre

**-y, --revY** y coordinate of REV centre

**-r, --revRadius** Radius of REV centre

**OSTRICH** Undocumented

```
usage: MOUSE OSTRICH [-h] -n NAME [-id IDENTITY] [-p PARALLEL] [-c CORES]
                        [-o OPTIMIZER]
```

#### Options:

**-n, --name** Name of the file containing the model data without the extension

**-id, --identity** identification Number

**-p=True, --parallel=True** use parallel processing

**-c=4, --cores=4** number of logical cores to use for parallel processing

**-o=ParticleSwarm, --optimizer=ParticleSwarm** optimization algorithm

**ABAQUS** Undocumented

```
usage: MOUSE ABAQUS [-h] -n NAME
```

**Options:**

**-n, --name** Name of the file containing the model data without the extension

## 2.3 MOUSE Documentation

**class** `MOUSE.SplashScreen` (*boxWidth=55, textWidth=70, padWidth=15*)

Bases: `object`

Creates the splash screen and interface for MOUSE

This class allows for the generation of an introduction screen for MOUSE. Here, a collection of printing methods are created in order to provide an environment for creating a consistent splash screen and interface.

**boxWidth**

*int* – Character width of text box for splash screen

**textWidth**

*int* – Character width of text area for splash screen

**padWidth**

*int* – Character width of text area for padding on splash screen

**printBoxLine** ()

Prints a horizontal line for the box in the centre of the console

**Returns** Prints a centred horizontal dashed line of length `self.boxWidth` on the console

**Return type** `None`

**printCentre** (*text*)

Prints text in the centre of the console

**Parameters** **text** (*str*) – text to be printed in the centre of the console

**Returns** Prints `str` to the centre of the console

**Return type** `None`

**printFullLine** ()

Prints a horizontal line across the text width of the console

**Returns** Prints a horizontal dashed line of length `self.textWidth` on the console

**Return type** `None`

**printInBox** (*text*)

Prints text in the centre of the box

**Parameters** **text** (*str*) – text to be printed in the centre of the box

**Returns** Prints a horizontal dashed line of length `self.textWidth` on the console

**Return type** `None`

**printModule** (*module, status*)

Prints module and installation status in the splash box



**Parameters**

- **module** (*str*) – name of the module
- **status** (*str*) – module status [installed, available, unavailable]

**Returns** Prints the module name with the status in the splash box

**Return type** None

**printSplash()**

Clears the console and prints the splash screen to the console

**Returns** Splash screen printed on console

**Return type** None

---

**Todo**

Import Modules and stuses from module files rather than hard coding them into this method

---

**MOUSE.createParser()**

Creates an argparse parser object for MOUSE and imports argparse subparsers for each MOUSE Module

---

**Todo**

Scan subparsers from module files and import in order to remove hard-coded dependance

---

**Note:** Currentlty subparser imports are hard-coded in

---

**Returns** the main argument parser for MOUSE populated with all required subparsers form modules.

**Return type** argparse.ArgumentParser

## 2.4 Modules Documentation

### 2.4.1 Modules.Module\_ABAQUS module

**class** Modules.Module\_ABAQUS.**Module\_ABAQUS** (*baseName*)

Bases: *Modules.Base.ContinuumModuleBaseClass*

**formatOutput()**

Formats ABAQUS data into consistent nested lists and writes them to binary file

**Returns** writes serialized binary data to file

**Return type** None

**parseInput()**

Parses input file

**Returns** returns data in a structured array

**Return type** struct

**run()**  
runs the HODS homogenization Module which creates input files for OSTRICH MOUSE Module

**Returns** MOUSE homogenization data files

**Return type** None

**setParameters** (*revCentreX=None, revCentreY=None, revRadius=None*)

Sets module parameters

---

#### Todo

assess revCentreX, revCentreY and revRadius from data rather than from input file

---

**Parameters** **parameters** (*dict*) – new parameters to be set

**Returns** Sets module parameters

**Return type** None

`Modules.Module_ABAQUS.importModelData(modelName)`

Imports the input model parameters and assigns them to a global modelData variable

**Parameters** **modelName** (*str*) – Name of file containing the model data.

**Returns** Assigns model parameters from file to global modelData

**Return type** None

`Modules.Module_ABAQUS.parserHandler(args)`

Function called after argparse subparser is executed

**Parameters** **args** (*argparse.Arguments*) – argparse parsed command line arguments.

**Returns** initializes ABAQUS Module and runs it

**Return type** None

`Modules.Module_ABAQUS.populateArgumentParser(parser)`

Adds arguments to the argument parser

**Parameters** **parser** (*argparse.ArgumentParser*) – empty argparse subparser

**Returns** same argparse subparser, now populated with arguments

**Return type** *argparse.ArgumentParser*

## 2.4.2 Modules.Module\_HODS module

**class** `Modules.Module_HODS.Module_HODS(baseName)`

Bases: *Modules.Base.HomogenizationModuleBaseClass*

Creates the HODS model interface for MOUSE

This class allows for the generation of the usage of HODS through the MOUSE framework. Because HODS was also written in python, a direct link can be established between the programs rather than relying on I/O protocols.

**stressHistory**

*nested list of float* – List of homogenized stress tensor history

**strainHistory***nested list of float* – List of homogenized strain tensor history**timeHistory***list of float* – List of simulation time steps**revCentreX***float* – x position of centre of REV**revCentreY***float* – y position of centre of REV**revRadius***float* – radius of REV**formatOutput ()**

Formats Homogenization data into consistent nested lists and writes them to binary file

**Returns** writes serialized binary data to file**Return type** None**parseInput ()**

Parses input file

**Returns** returns data in a structured array**Return type** struct**run ()**

runs the HODS homogenization Module which creates input files for OSTRICH MOUSE Module

**Returns** MOUSE homogenization data files**Return type** None**setParameters (args)**

Sets module parameters

---

**Todo**assess revCentreX, revCentreY and revRadius from data rather than from input file

---

**Parameters** **parameters** (*dict*) – new parameters to be set**Returns** Sets module parameters**Return type** None**Modules.Module\_HODS.importModelData (modelName)**

Imports the input model parameters and assigns them to a global modelData variable

**Parameters** **modelName** (*str*) – Name of file containing the model data.**Returns** Assigns model parameters from file to global modelData**Return type** None**Modules.Module\_HODS.parserHandler (args)**

Function called after argparse subparser is executed

**Parameters** **args** (*argparse.Arguments*) – argparse parsed command line arguments.**Returns** initializes HODS Module and runs it

**Return type** None

`Modules.Module_HODS.populateArgumentParser (parser)`

Adds arguments to the argument parser

**Parameters** `parser` (*argparse.ArgumentParser*) – empty argparse subparser

**Returns** same argparse subparser, now populated with arguments

**Return type** *argparse.ArgumentParser*

## 2.4.3 Modules.Module\_OSTRICH module

`class Modules.Module_OSTRICH.Module_OSTRICH (baseName)`

Bases: *Modules.Base.ParameterEstimationModuleBaseClass*

Creates the OSTRICH interface for MOUSE

This class allows for the generation of the usage of UDEC through the MOUSE framework. As of current, there is still no capacity for full automation due to UDEC API limitations. As such, a batch UDEC script is generated which can then be called in UDEC with one command.

**identity**

*int* – for repeat trials, a different identity can be assigned to each game

**MPI**

*bool* – OSTRICH uses parallel processing if True

**cores**

*int* – number of logical cores to be used for optimization

**optimizer**

*str* – optimization algorithm to use. List can be found in OSTRICH Documentation

**formatOutput ()**

Formats parameter estimation data for MOUSE

**Returns** Copies OSTRICH output files to output directory

**Return type** None

**getBoundaryDisplacements ()**

**getBoundaryStresses ()**

**getModelConstants ()**

**getModelParameters ()**

Returns abaqus model input parameters

---

**Todo**

move towards a more object oriented method of handling data

---

**Returns** dictionary of abaqus input parameters

**Return type** dict

**getOstrichParameters (frontBias=1)**

Returns OSTRICH parameters

---

**Todo**

move towards a more object oriented method of handling data

---

**Returns** dictionary of udec parameters

**Return type** dict

**parseInput ()**

Parses input file

**Returns** creates Ostrich input files

**Return type** None

**run ()**

runs the OSTRICH Module which creates input files for OSTRICH, then runs OSTRICH

**Returns** OSTRICH data files

**Return type** None

**setParameters (args)**

Sets module parameters

**Parameters** **parameters** (*dict*) – new parameters to be set

**Returns** Sets module parameters

**Return type** None

`Modules.Module_OSTRICH.fillTemplate (template, parameters, file)`

fills a template file with variable parameters

**Parameters**

- **template** (*str*) – file path to template file
- **parameters** (*dict*) – dictionary of parameters and corresponding values
- **file** (*str*) – destination file path for filled template

**Returns** saves filled template to file

**Return type** None

`Modules.Module_OSTRICH.getVelocityString (velTable)`

Generates a table of relative velocities (from -1 to 1) for the simulation in a linear string format for ABAQUS

**Parameters** **velTable** (*list of float*) – times at which the velocity changes from negative to positive

**Returns** table of relative velocities (-1 to 1) in ABAQUS format

**Return type** str

`Modules.Module_OSTRICH.importMaterialData (materialName)`

Imports the material parameters and assigns them to a global material variable

**Parameters** **material** (*str*) – Name of file containing the model data.

**Returns** Assigns model parameters from file to global material

**Return type** None

`Modules.Module_OSTRICH.importModelData (modelName)`

Imports the input model parameters and assigns them to a global `modelData` variable

**Parameters** `modelName` (*str*) – Name of file containing the model data.

**Returns** Assigns model parameters from file to global `modelData`

**Return type** None

`Modules.Module_OSTRICH.parserHandler (args)`

Function called after `argparse` subparser is executed

**Parameters** `args` (*argparse.Arguments*) – `argparse` parsed command line arguments.

**Returns** initializes UDEC Module and runs it

**Return type** None

`Modules.Module_OSTRICH.populateArgumentParser (parser)`

Adds arguments to the argument parser

**Parameters** `parser` (*argparse.ArgumentParser*) – empty `argparse` subparser

**Returns** same `argparse` subparser, now populated with arguments

**Return type** `argparse.ArgumentParser`

## 2.4.4 Modules.Module\_UDEC module

`class Modules.Module_UDEC.Module_UDEC (baseName)`

Bases: `Modules.Base.DemModuleBaseClass`

Creates the UDEC model interface for MOUSE

This class allows for the generation of the usage of UDEC through the MOUSE framework. As of current, there is still no capacity for full automation due to UDEC API limitations. As such, a batch UDEC script is generated which can then be called in UDEC with one command.

**fileName**

*str* – name of simulation data file

**UDecParameters**

*dict* – Dictionary of UDEC parameters as keys and the associated value as dictionary values

**createInputFiles ()**

Creates Input files for UDEC and a batch file to run them all

---

**Todo**

move towards a more object oriented method of handling data

---

**Returns** creates UDEC input files and corresponding batch file

**Return type** None

**formatOutput ()**

Formats DEM data into consistent nested hash tables and writes them to binary file

**Returns** writes serialized binary data to file

**Return type** None

**getUDECPParameters ()**

Returns UDEC parameters

---

**Todo**

move towards a more object oriented method of handling data

---

**Returns** dictionary of udec parameters

**Return type** dict

**getVelocityString (velTable)**

Generates a table of relative velocities (from -1 to 1) for the simulation in a linear string format for UDEC

**Parameters** **velTable** (*list of float*) – times at which the velocity changes from negative to positive

**Returns** table of relative velocities (-1 to 1) in UDEC format

**Return type** str

**inputFileName ()**

Overloads the default input settings to import python data

**Returns** full path of input python data

**Return type** str

**outputFileName ()**

Overloads the default input settings to export

**Returns** writes serialized binary data to file

**Return type** None

**parseInput ()**

Parses input file

**Returns** returns data in a structured array

**Return type** struct

**run ()**

runs the UDEC Module which creates input files for UDEC, then opens UDEC to allow the user to run UDEC, then compiles the UDEC output to MOUSE compatible output.

**Returns** MOUSE DEM data files

**Return type** None

**setParameters ()**

Sets module parameters

**Parameters** **parameters** (*dict*) – new parameters to be set

**Returns** Sets module parameters

**Return type** None

`Modules.Module_UDEC.compileFiles (simulations, files, rawPath, compiledPath)`

compiles the raw UDEC data files for each timestep into one file

**Parameters**

- **simulations** (*list of str*) – List of simulations names
- **files** (*list of str*) – List of files to be compiled
- **rawPath** (*str*) – directory in which the raw UDEC data is contained
- **compiledPath** (*str*) – directory in which the compiled UDEC data is contained

**Returns** saves compiled data to file.

**Return type** None

`Modules.Module_UDEC.fileList (path)`

Gets list of all files in a given directory

**Parameters** **path** (*str*) – directory to get file list from

**Returns** List of file names in directory

**Return type** list of str

`Modules.Module_UDEC.importModelData (modelName)`

Imports the input model parameters and assigns them to a global modelData variable

**Parameters** **modelName** (*str*) – Name of file containing the model data.

**Returns** Assigns model parameters from file to global modelData

**Return type** None

`Modules.Module_UDEC.parseDataFile (fileName)`

Parses raw DEM data from tab delimited text tile into nested python dictionary

The raw DEM Data is considered to be comprised of six distinct types: block data, contact data, corner data, domain data, grid point data, and zone data. Here, each block, contact, corner, domain, grid point, and zone is assigned a unique 7-digit numeric identifier (assuming here that the number of components in the system does not exceed 10 million) by which the associated data can be accessed. The same identifier may be repeated for different data types. Each DEM data hash table has three levels of nesting. The first level keys are the simulation times, which returns the second level of hash tables. The second level keys are the component identifiers, which returns a third level hash table. In this third level, the component attributes can be accessed using the attribute name as the key.

**Parameters** **fileName** (*str*) – name of data file to be parsed

**Returns** Tripple nested dictionary of DEM data

**Return type** nested DEM dict

`Modules.Module_UDEC.parserHandler (args)`

Function called after argparse subparser is executed

**Parameters** **args** (*argparse.Arguments*) – argparse parsed command line arguments.

**Returns** initializes UDEC Module and runs it

**Return type** None

`Modules.Module_UDEC.populateArgumentParser (parser)`

Adds arguments to the argument parser

**Parameters** **parser** (*argparse.ArgumentParser*) – empty argparse subparser

**Returns** same argparse supparser, now populated with arguments

**Return type** *argparse.ArgumentParser*



`Modules.Module_UDEC.simulationFiles` (*files, rawPath, compiledPath*)

Isolates files that contain UDEC simulation Data

**Parameters**

- **files** (*list of str*) – List of files to be searched
- **rawPath** (*str*) – directory in which the raw UDEC data is contained
- **compiledPath** (*str*) – directory in which the compiled UDEC data is contained

**Returns** List of simulation files without file extension

**Return type** list of str

## 2.4.5 Modules.Base module

`class Modules.Base.ContinuumModuleBaseClass` (*program, parameters, baseName*)

Bases: `Modules.Base.ModuleBaseClass`

Creates a base class for the continuum model modules containing common methods and attributes

A base continuum model module class is implemented here, inheriting from the module base class to provide a framework containing required methods and attributes for the continuum model modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementing of certain methods.

**type**

*str* – Type of module

**inputFileName** ()

Returns full path of input binary data

**Returns** full path of input binary data

**Return type** str

**outputFileName** ()

Returns full path of output binary data

**Returns** full path of output binary data

**Return type** str

`class Modules.Base.DemModuleBaseClass` (*program, parameters, baseName*)

Bases: `Modules.Base.ModuleBaseClass`

Creates a base class for the DEM modules containing common methods and attributes

A base dem module class is implemented here, inheriting from the module base class to provide a framework containing required methods and attributes for the DEM modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementing of certain methods.

**type**

*str* – Type of module

**inputFileName** ()

Returns full path of input binary data

**Returns** full path of input binary data

**Return type** str

**outputFileName ()**

Returns full path of output binary data

**Returns** full path of output binary data

**Return type** str

**class** `Modules.Base.HomogenizationModuleBaseClass` (*program, parameters, baseName*)

Bases: `Modules.Base.ModuleBaseClass`

Creates a base class for the homogenization modules containing common methods and attributes

A base homogenization module class is implemented here, inheriting from the module base class to provide a framework containing required methods and attributes for the homogenization modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementing of certain methods.

**type**

*str* – Type of module

**inputFileName ()**

Gets full path of input binary data

**Returns** full path of input binary data

**Return type** str

**outputFileName ()**

Returns full path of output binary data

**Returns** full path of output binary data

**Return type** str

**class** `Modules.Base.ModuleBaseClass` (*program, baseName, parameters={}, suppressText=False, suppressErrors=True*)

Bases: object

Creates a base class containing common module methods and attributes

A base module class is implemented here to provide a framework containing required methods and attributes for the MOUSE modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementing of certain methods.

**program**

*str* – String containing name of module software executable file.

**parameters**

*dict* – Dictionary of command line parameters as keys and corresponding arguments as entries

**suppressText**

*bool* – Suppresses text output from modules if True

**suppressErrors**

*bool* – Suppress error output from modules if True

**baseName**

*str* – Name of model input file

**binaryDirectory**

*str* – Directory in which MOUSE binary data is located

**textDirectory**

*str* – Directory in which MOUSE text data is located

**inputDirectory**

*str* – Directory in which MOUSE input data is located

**outputDirectory**

*str* – Directory in which MOUSE output data is located

**clearScreen()**

Clears all text from the console.

Returns: None: Clears all text from the console

**commandLineArguments()**

converts the parameters dictionary to a string which can be passed to the command line when running the specified program.

**Returns** string to be passed to command line

**Return type** str

**loadData()**

Loads module data from binary using the pickle serialization module

**Parameters** **data** (*any*) – Module data to be serialized and stored in file

**Returns** serialized data in binary file in specified binaryDirectory

**Return type** None

**printDone()**

Prints 'Done' to console.

Note: It is recommended that this method be used in conjunction with printStatus()

**Parameters** **status** (*str*) – status to be printed to console

**Returns** 'Done' printed to the console

**Return type** None

**printErrors(*error*)**

Prints error to console if not suppressed

---

**Note:** All errors caught should be routed through this function. Using this function allows for easy suppression and piping of output.

---

**Parameters** **error** (*str*) – error to be printed to console

**Returns** error printed to the console

**Return type** None

**printSection(*section*)**

Prints a section name to console.

Sections are displayed aligned to the left side of the console.

**Parameters** **section** (*str*) – section name to be printed to console

**Returns** section name printed to the console

**Return type** None

**printStatus** (*status*)

Prints a status to console.

Statuses are displayed proceeding a tab and are followed by ellipses with no new line character at the end of the print line.

Note: The no new line character at the end of the print line allows the printDone() method to print 'Done' at the end of the ellipses after some arbitrary code execution. It is recommended that these two methods always be used together

**Parameters** **status** (*str*) – status to be printed to console

**Returns** status printed to the console

**Return type** None

**printText** (*text*, *end*='\\n')

Prints text to console if not suppressed

---

**Note:** All text printed to the console should be routed through this function rather than using the built-in print() function. Using this function allows for easy suppression and piping of output.

---

---

**Todo**

If text suppression is on, route output to file.

---

**Parameters**

- **text** (*str*) – text to be printed to console
- **end** (*str*, *optional*) – character to be appended to end of print line

**Returns** text printed to the console

**Return type** None

**printTitle** (*title*)

Prints a title to console.

Titles are displayed with horizontal lines printed above and below the text and are alligned with the left side of the console.

**Parameters** **title** (*str*) – title to be printed to console

**Returns** title printed to the console

**Return type** None

**run** ()

runs specified program with specified parameters

**Returns** runs specified program with specified parameters

**Return type** None

**saveData** (*data*)

Saves module data as binary using the pickle serialization module

**Parameters** **data** (*any*) – Module data to be serialized and stored in file

**Returns** serialized data in binary file in specified binaryDirectory

**Return type** None

**updateParameters** (*parameters*)

Updates the parameter attribute so that the modul can be run with a different parameter set without being re-instantiated

**Parameters** *parameters* (*dict*) – dictionary of new parameters

**Returns** updates the parameter attribute

**Return type** None

**class** `Modules.Base.ParameterEstimationModuleBaseClass` (*program*, *parameters*, *base-Name*)

Bases: `Modules.Base.ModuleBaseClass`

Creates a base class for the parameter estimation modules containing common methods and attributes

A base parameter estimation module class is implemented here, inheriting from the module base class to provide a framework containing required methods and attributes for the parameter estimation modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementatation of certain methods.

**type**

*str* – Type of module

**inputFileName** ()

Returns full path of input binary data

**Returns** full path of input binary data

**Return type** *str*

**outputFileName** ()

Returns full path of output binary data

**Returns** full path of output binary data

**Return type** *str*



## HODS REFERENCE MANUAL

### 3.1 Modules.HODS package

#### 3.1.1 Submodules

#### 3.1.2 Modules.HODS.HODS module

```
class Modules.HODS.HODS.DataSet (fileName)
    Bases: object

    blocksWithContacts (blocks, contacts)
    blocksWithCorners (blocks, corners)
    contactsBetweenBlocks (blocks1, blocks2)
    contactsOnBlocks (blocks)
    cornerX (corners, time)
    cornerY (corners, time)
    cornersOnBlocks (blocks)
    cornersOnContacts (contacts)
    limits ()
    parseDataFile (fileName)
    zoneS11 (zones, time)
    zoneS12 (zones, time)
    zoneS22 (zones, time)
    zoneS33 (zones, time)
    zonesInBlocks (blocks)

class Modules.HODS.HODS.Homogenize (centre, radius, fileName)
    Bases: Modules.HODS.HODS.DataSet

    blocksInsideBoundary ()
    blocksOnBoundary ()
    blocksOutsideBoundary ()
    calculateHomogenizationParameters ()
```

```
contactsInsideBoundary ()
contactsOutsideBoundary ()
cornersInsideBoundary ()
cornersOutsideBoundary ()
duplicateCorners (corners, blocks)
orderBlocks (blocks, relevantContacts)
orderCorners (orderedBlocks, corners)
singleElementCorners ()
strain ()
stress ()
time ()
class Modules.HODS.HODS.common
  Bases: object
  angle (x1, y1, x2, y2)
  area (p)
  listIntersection (a, b)
  segments (p)
  triangleArea (gp)
```

### 3.1.3 Module contents



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### m

`Modules.Base`, [13](#)  
`Modules.Module_ABAQUS`, [5](#)  
`Modules.Module_HODS`, [6](#)  
`Modules.Module_OSTRICH`, [8](#)  
`Modules.Module_UDEC`, [10](#)  
`MOUSE`, [4](#)