CritOpS Documentation

Release 2.1.0

Andrew Johnson

Apr 19, 2017

CONTENTS:

CHAPTER

ONE

INTRO

This is the documentation for CritOpS, a Critical Optimization Search tool for use with NEWT[1]. CritOpS is designed to iteratively modify inputs for NEWT to obtain a desired eigenvalue. More documentation will be added before the final release of this code, including examples and validation testing.

1.1 Setup

```
git clone https://github.com/drewejohnson/CritOpS.git
cd CritOpS
python setup.py install
```

The code currently requires *python3* due to some formatting calls, and *pandas* for some better data output.

1.2 Usage

CritOpS can be run from the terminal while in the directory outside the critops folder with the command

```
$ python critops <mainfile> <paramfile>
```

Where <mainfile> is a valid NEWT input file with some variables in place of valid values and <paramfile> is the file that contains limits on iteration parameters, desired k-eff, and indicates the variable to be iterated upon. See `testing/iter_tester.inp` and `testing/param_tester.txt` for one example case.

```
python3 CritOpS.py inp_file param_file [-v] [-o OUTPUT]

positional arguments:
  inp_file              template SCALE input file
  param_file            file containing parameters for operation

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         reveal more of the mystery behind the operation
  -o OUTPUT, --output OUTPUT
                        write status to output file
```

The parameter file controls iteration procedure and *SCALE* execution. Parameters that can be updated with the parameter file include

1. *k_target*: Desired value of k-eff to be obtained from the *SCALE* runs
2. *eps_k*: Acceptable accuracy between *k_target* and each value of k-eff

3. *iter_lim*: Maximum number of times to run *SCALE*
4. *exe_str*: Absolute path to your *SCALE* executable.
5. *var_char*: Whatever character you want to use as a designator for the variables

Currently, *CritOpS* only supports one iteration variable, which is declared in the parameter file with:

```
iter_var <var> <start> <min> <max>
```

The input file should be a valid *NEWT* input file, with some minor modifications. There should exist certain values defined as variables preceeded by the *var_char*,

```
cuboid 20 5   0  0 -$del_z
```

Given some input and parameter files, the code will create and execute successive input files, parse the outputs for the update k-eff, and then update the iteration variables.

1.3 License

1.4 References

[1]: M. D. DeHart, and S. Bowman, “Reactor Physics Methods and Analysis Capabilities in SCALE,” Nuclear Technology, Technical Paper vol. 174, no.2, pp. 196-213, 2011.

CHAPTER

TWO

IMPORTING CRITOPS ITERATOR

2.1 External Usage

The CritOpS module can easily be imported into external processing scripts. Presented here is an example of executing the iteration routine as a standalone process.

```
from critops.iterator import itermain
from critops.utils import oprint
from critops.outputs import output_landing


critArgs = {
    'k-target': 9.98515E-01,
     'eps_k': 1E-8,
    'verbose': False,
    'output': None,
    'iter_lim': 15,
    'exe_str': 'C:\\Scale-6.2.1\\bin\\scalerte.exe',
    'k-id': 'Input buckling',
    'k-col': 9,
    'stalequit': False,
}

iter_var = {'buck': (1.2E-03, 1.00E-03, 5.00E-3)}

bFile = 'intHX5_buck.inp'

oprint('\nStarting buckling iteration\n', **critArgs)

iter_vec, k_vec, conv_type = itermain(bFile, iter_var,

output_landing(iter_vec, k_vec, conv_type, **critArgs)
```

2.2 Default Arguments

Parameter	Default	Note
`eps_k`	1E-4	Tightness on *k* convergence
`k_target`	1.0	Desired *k-eff*
`iter_lim`	50	Maximum number of iterations
`tiny`	1E-16	Numerical zero
`var_char`	`'$'`	Character to identify iteration variables
`k-id`	`'k-eff = '`	String to identify line containing *k-eff*
`k-col`	2	Location of *k-eff* in `line.split()`
`stalequit`	`True`	Terminate if *k-eff* hasn't changed
`exe_str`	C:\SCALE-6.2.1\bin\scalerte.exe	Absolute path to `SCALE` executable

CHAPTER

THREE

ITERATOR

CritOpS

Andrew Johnson

Objective: Main file for controlling the iteration scheme

Functions:

> iter_main: Landing function that drives the iteration
>
> makefile: Write the new output file using the value from iteration _iter
>
> update_itervar: Simple function to update the iteration variables.
>
> parse_scale_out_eig: Read through the SCALE output file specified by _ofile and return status and eigenvalue (if present)

`critops.iterator.`**`itermain`**(*file_name: str*, *iter_vars: dict*, *kwargs: dict*)

Main function for controlling the iteration

Parameters

- **`file_name`** – Name of template file
- **`iter_vars`** – Dictionary of iteration variables and their starting, minima, and maximum values
- **`kwargs`** – Additional keyword arguments

Returns k_vec: List of progression of eigenvalue through iteration procedure

Returns iter_vecs: Dictionary of iteration and their values through iteration procedure

Returns

> conv_type - reason for exiting iter_main
>
> 0: Accurately converged to target eigenvalue in specified iterations
>
> 1: iter_var exceeded specified maximum twice
>
> -1: iter_var exceeded specified minimum twice
>
> 2: Reached iteration limit without reaching target eigenvalue
>
> -2: Previous two k are close to similar

`critops.iterator.`**`update_itervar`**(*iter_vars: dict*, *iter_vec: dict*, *kvec: (<class 'list'>, <class 'tuple'>)*, *ktarg: float*, ***kwargs*)

Simple function to update the iteration variables. Currently set up for a positive feedback on the variables. I.e. increasing each iteration variable will increase k

Parameters

- **`iter_vars`** – Dictionary of iteration variables and their minima/maxima
- **`iter_vec`** – Dictionary of iteration variables and their values through the iteratio procedure
- **`kvec`** – Vector of eigenvalues
- **`ktarg`** – Target eigenvalue

Returns status status = 0 if the updated value is inside the intended range status = 1 if the desired updated value is greater than the specified maximum of the parameter and the max value is used as the updated value status = -1 if the desired updated value is less than the specified maximum of the parameter and the minimum value is used as the updated value

`critops.iterator.`**`parse_scale_out_eig`**(*_ofile: str*, ***kwargs*)

Read through the SCALE output file specified by _ofile and return status and eigenvalue (if present)

Parameters **`_ofile`** – SCALE .out file

Returns

Status, eigenvalue

status = True if output file exists and eigenvalue was extracted status = False if output file exists but no eigenvalue was found (possible error in input file syntax) exit operation if no output file found

CHAPTER

FOUR

READINPUTS

CritOps

Andrew Johnson

Objective: Read the inputs, update global variables, and check for proper variable usage

Functions:

> check_inputs: make sure values in global_parameters are good for running read_param: Read the parameter file and update values in globalparams readmain: Main driver for reading and processing the input files

`critops.readinputs.`**`readmain`**`(`*tmp_file*, *param_file*, *kwargs: dict*`)`

Main driver for reading and processing input files.

Parameters

- **`tmp_file`** – Template input file
- **`param_file`** – Parameter file
- **`kwargs`** – Additional arguments - verbose (True) - status updates - output (None) - print to screen - Plus additional iteration parameters

Returns List of valid template file lines and dictionary of interation variables Updates kwargs based on values in param_file

`critops.readinputs.`**`read_param`**`(`*_pfile*, ***kwargs*`)`

Read the parameter file and update values in kwargs

Parameters **`_pfile`** – Parameter file

Returns iter_vars: Dictionary of iteration variables and their starting, minima, and maximum values

Returns updated keyword arguments

`critops.readinputs.`**`check_inputs`**`(`*temp_lines: list*, *iter_vars: dict*, ***kwargs*`)`

Run over the inputs and make sure things are good for operation

CHAPTER FIVE

OUTPUTS

NRE6401 - Molten Salt Reactor

CritOpS

Objective: Functions for reading SCALE output files and writing output files

Functions:

parse_scale_output: Parse through the SCALE output file and return status

`critops.outputs.`**`output_landing`**(*iter_vecs: dict*, *k_vec: (<class 'list'>, <class 'tuple'>)*, *_outtype: int*, ***kwargs*)

Write the output file according to the type of output

Parameters

- **`iter_vecs`** – Dictionary with iteration variables and their values through the procedure
- **`k_vec`** – Vector of eigenvalues
- **`_outtype`** – Flag indicating the reason the program terminated - 0 Nothing went wrong - 1 Desired update value for iteration parameter twice exceeded the maximum value from the parameter file - -1 Desired update value for iteration parameter twice exceeded the minimum value from the parameter file - 2 Exceeded the total number of iterations allotted - -2 No excessive change in eigenvalue

Returns

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX