Introduction to optimisation studies with Dakota

Frank F.M. Milthaler[1]

[1]Department of Earth Science and Engineering, Imperial College London

May 6, 2016

Outline

DAKOTA — A brief overview

How does it work?

Examples

Conclusion

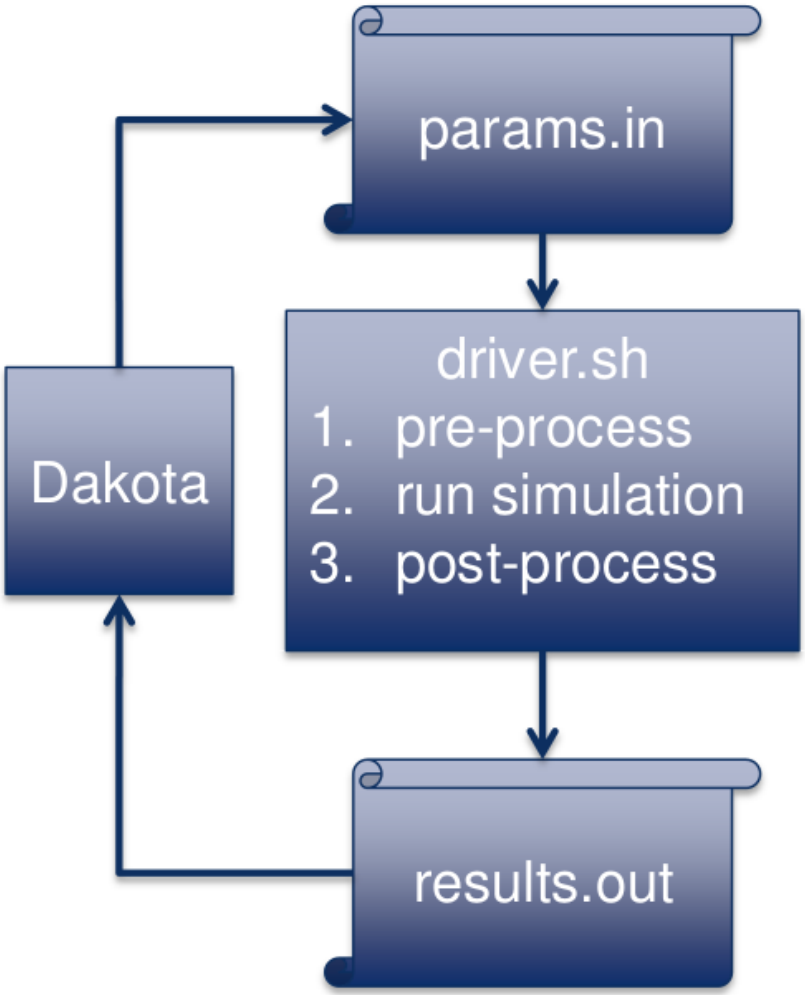
DAKOTA — What is it?

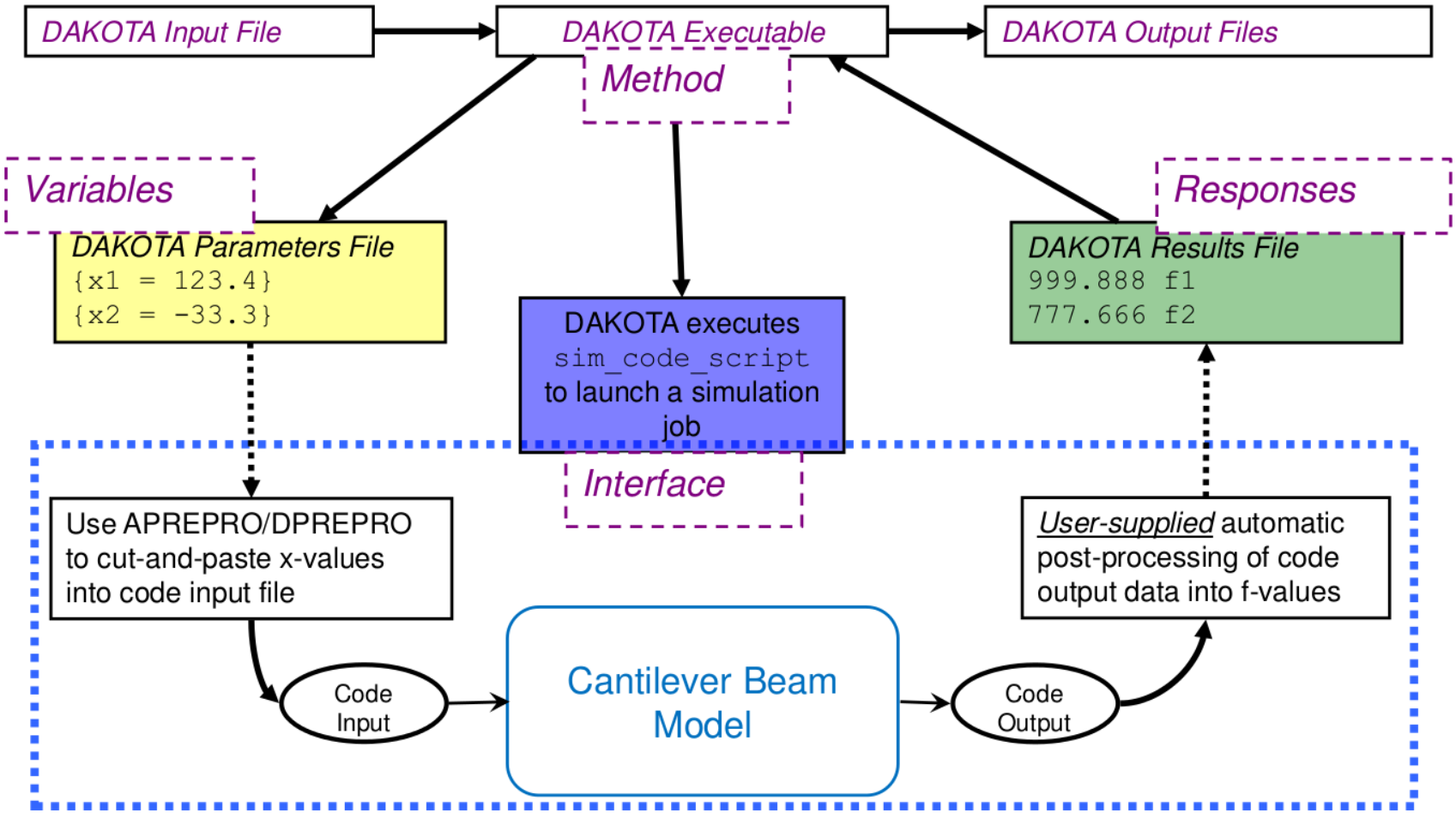
- *Design Analysis Kit for Optimization and Terascale Applications*
- Uncertainty quantification and nonlinear least squares
- Drives external tools, e.g. Fluidity or any other software
- Text file configuration, and scripts to start external tools

How does it work?

Usage:

```
dakota -i filename.in -o run.out > stdout.out
```


params.in
driver.sh
1. pre-process
2. run simulation
3. post-process
Dakota
results.out


DAKOTA Input File
DAKOTA Executable
DAKOTA Output Files
Method
Variables
Responses
DAKOTA Parameters File
{x1 = 123.4}
{x2 = -33.3}
DAKOTA executes
sim_code_script
to launch a simulation job
DAKOTA Results File
999.888 f1
777.666 f2
Interface
Use APREPRO/DPREPRO to cut-and-paste x-values into code input file
User-supplied automatic post-processing of code output data into f-values
Code Input
Cantilever Beam Model
Code Output

Input File

```
1  # Usage:
2  #   dakota -i filename.in -o run.out > stdout.out
3  environment
4    tabular_graphics_data
5    tabular_graphics_file = 'table_out.dat'
6  method
7    multidim_parameter_study
8    partitions = 6 6
9  model
10   single
11 variables,
12   continuous_design = 2
13     lower_bounds      1e4       1e4
14     upper_bounds      1e5       1e5
15     descriptors     'width' 'length'
```

Input File

```
16  # Usage:
17  #   dakota -i filename.in -o run.out > stdout.out
18  environment
19    tabular_graphics_data
20    tabular_graphics_file = 'table_out.dat'
21  method
22    multidim_parameter_study
23    partitions = 6 6
24  model
25    single
26  variables,
27    continuous_design = 2
28      lower_bounds      1e4       1e4
29      upper_bounds      1e5       1e5
30      descriptors     'width' 'length'
```

Input File

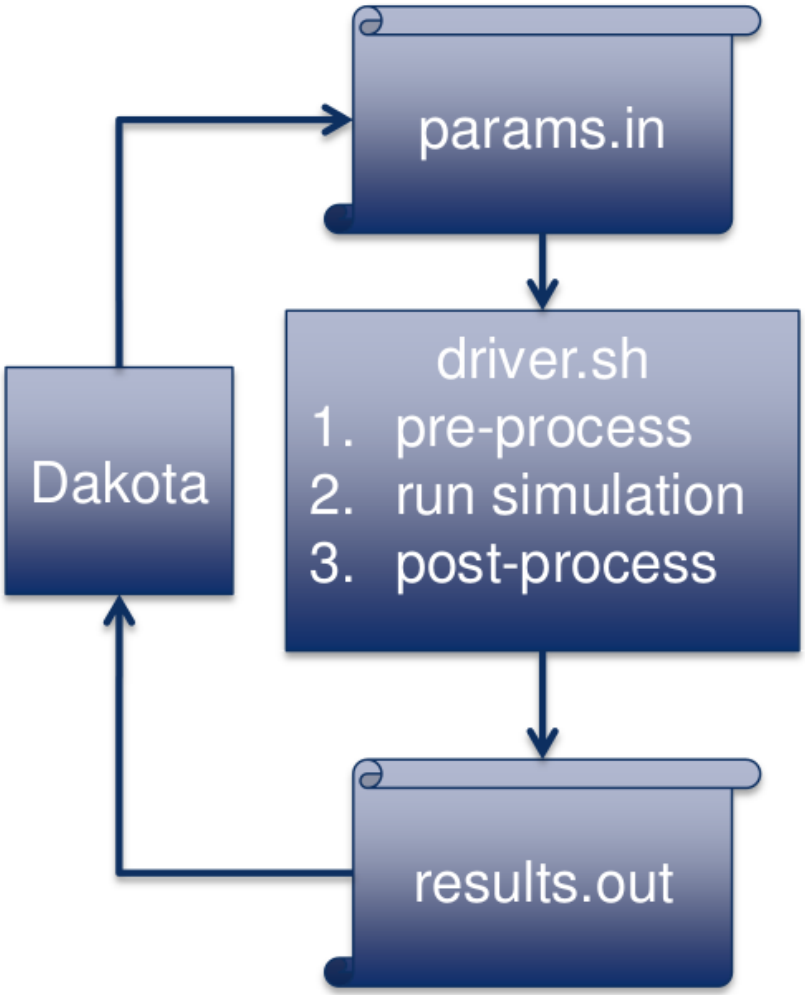
```
31  # Usage:
32  #   dakota -i filename.in -o run.out > stdout.out
33  environment
34    tabular_graphics_data
35    tabular_graphics_file = 'table_out.dat'
36  method
37    multidim_parameter_study
38    partitions = 6 6
39  model
40    single
41  variables,
42    continuous_design = 2
43      lower_bounds      1e4       1e4
44      upper_bounds      1e5       1e5
45      descriptors     'width' 'length'
```

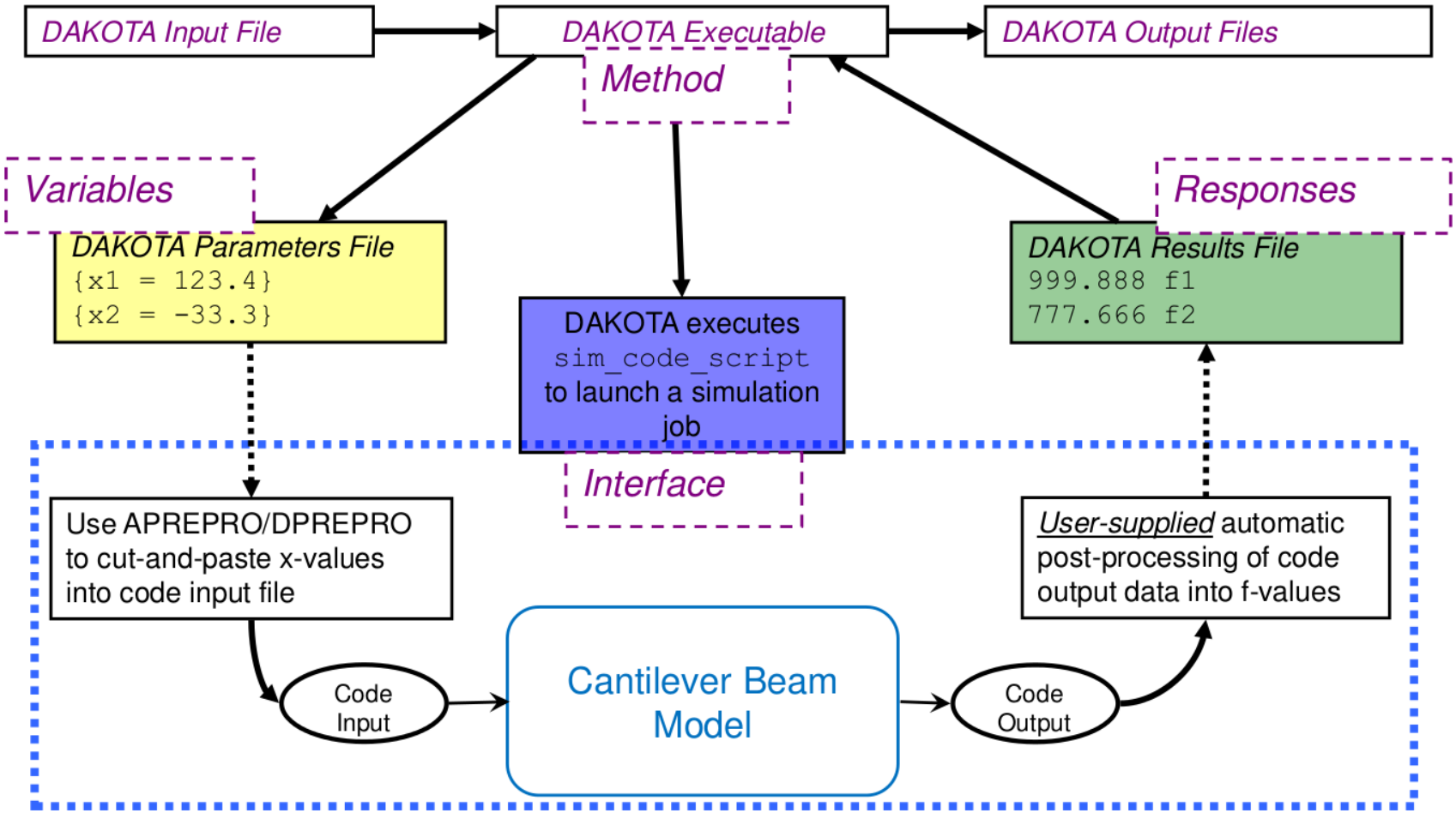
Input File

```
46  # Usage:
47  #   dakota -i filename.in -o run.out > stdout.out
48  environment
49    tabular_graphics_data
50    tabular_graphics_file = 'table_out.dat'
51  method
52    multidim_parameter_study
53    partitions = 6 6
54  model
55    single
56  variables,
57    continuous_design = 2
58      lower_bounds      1e4       1e4
59      upper_bounds      1e5       1e5
60      descriptors     'width' 'length'
```

Input File

```
61  interface,
62    fork
63      asynchronous
64      evaluation_concurrency = 6
65      analysis_driver = 'fluidity_tsunami.py'
66      parameters_file = 'params.in'
67      results_file    = 'results.out'
68      work_directory directory_tag
69      named 'workdir' file_save  directory_save
70      aprepro
71  responses
72    num_objective_functions = 1
73    no_gradients
74    no_hessians
75    sense 'max'
```


params.in
driver.sh
1. pre-process
2. run simulation
3. post-process
Dakota
results.out

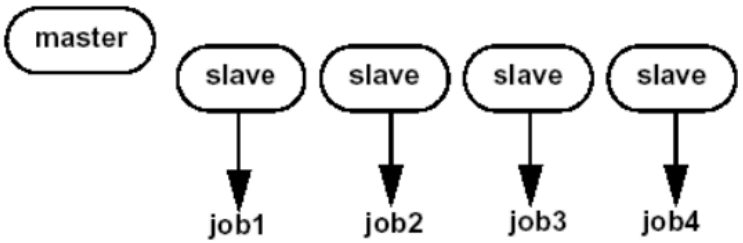

DAKOTA Input File
DAKOTA Executable
DAKOTA Output Files
Method
Variables
Responses
DAKOTA Parameters File
{x1 = 123.4}
{x2 = -33.3}
DAKOTA executes
sim_code_script
to launch a simulation job
DAKOTA Results File
999.888 f1
777.666 f2
Interface
Use APREPRO/DPREPRO to cut-and-paste x-values into code input file
User-supplied automatic post-processing of code output data into f-values
Code Input
Cantilever Beam Model
Code Output

Input File

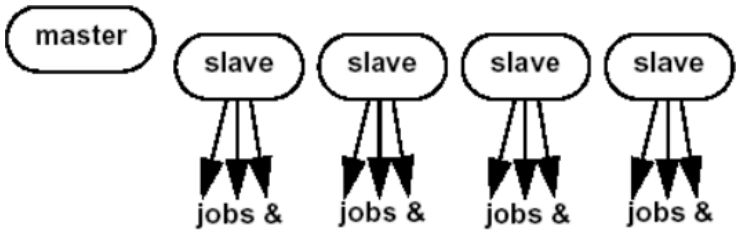
```
76  interface,
77    fork
78      asynchronous
79      evaluation_concurrency = 6
80      analysis_driver = 'fluidity_tsunami.py'
81      parameters_file = 'params.in'
82      results_file    = 'results.out'
83      work_directory directory_tag
84      named 'workdir' file_save  directory_save
85      aprepro
86  responses
87    num_objective_functions = 1
88    no_gradients
89    no_hessians
90    sense 'max'
```

Running in parallel

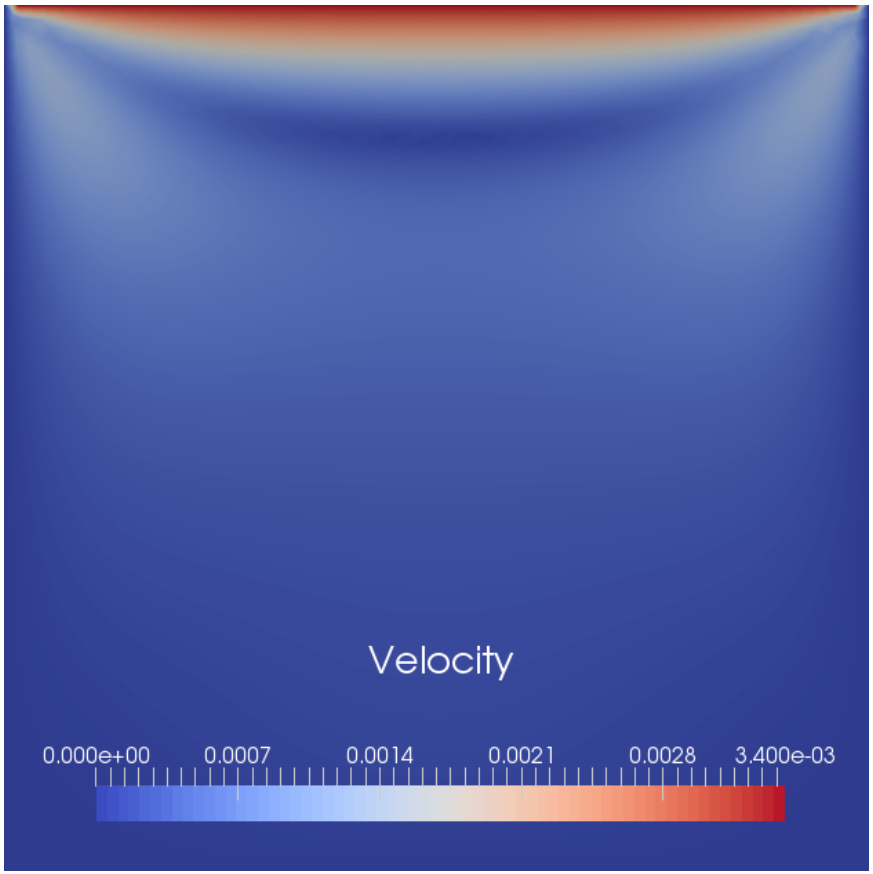
Running in parallel



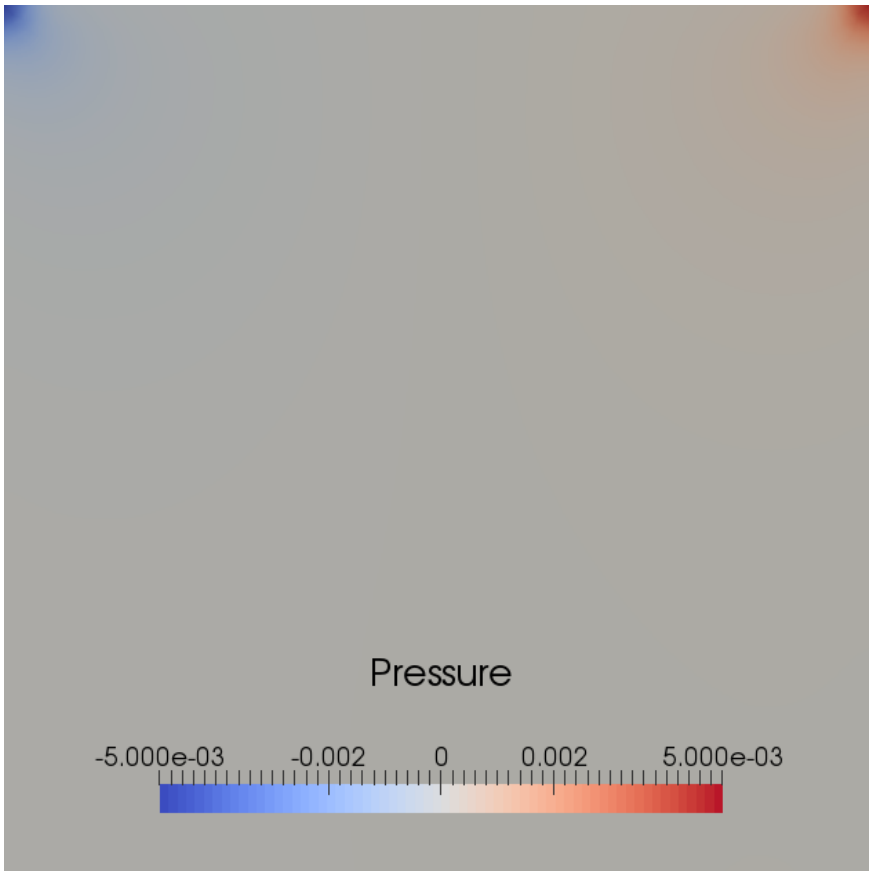
Running in parallel



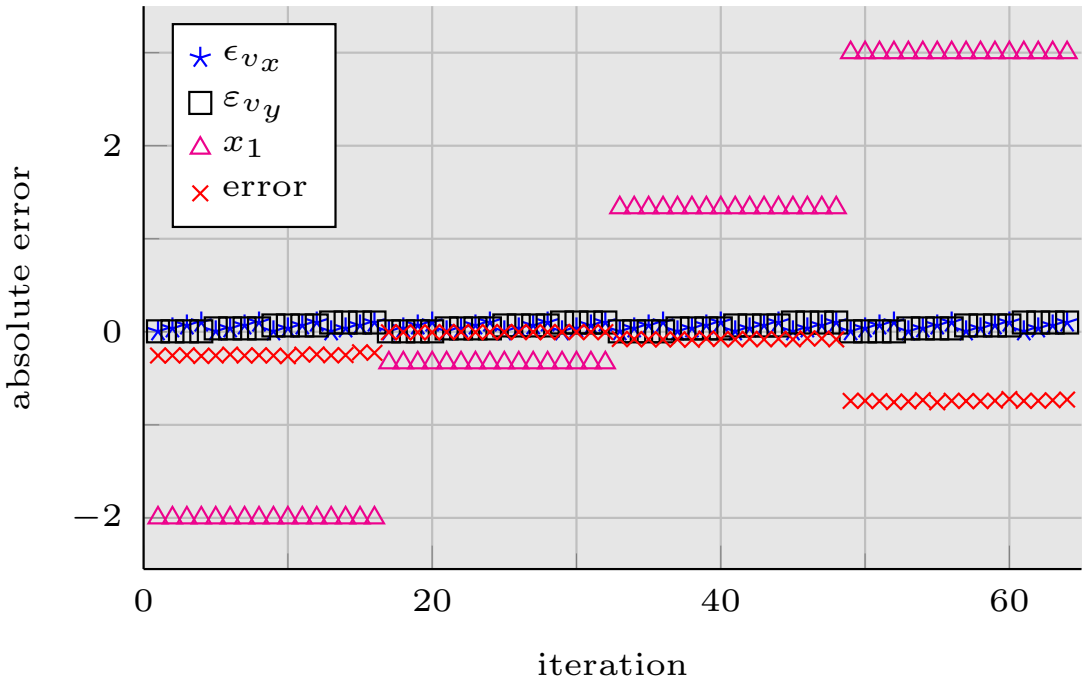
Example: Driven Cavity



Example: Driven Cavity

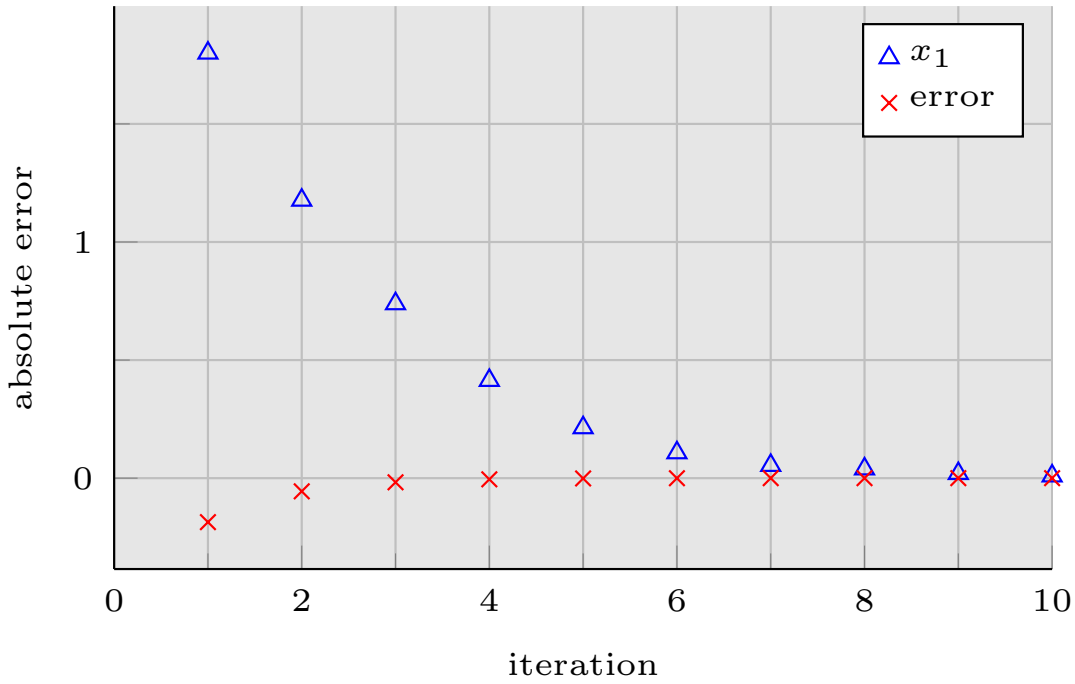


Example: Driven Cavity



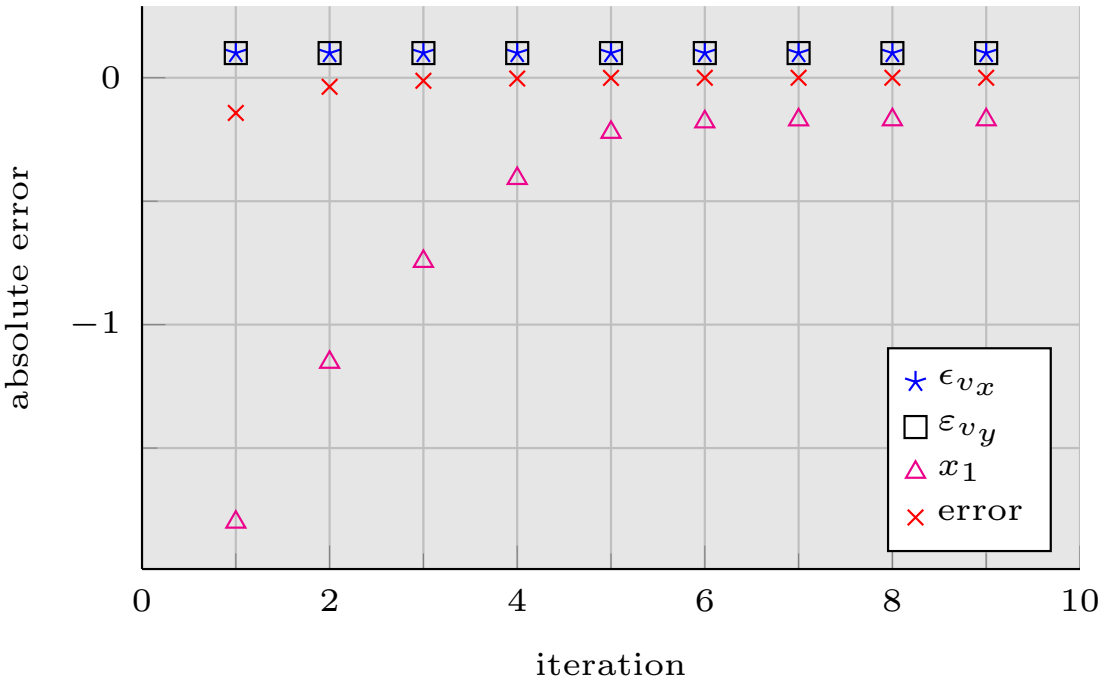
Multidimensional parameter study: Varying mesh adaptivity parameters and boundary condition.

Example: Driven Cavity



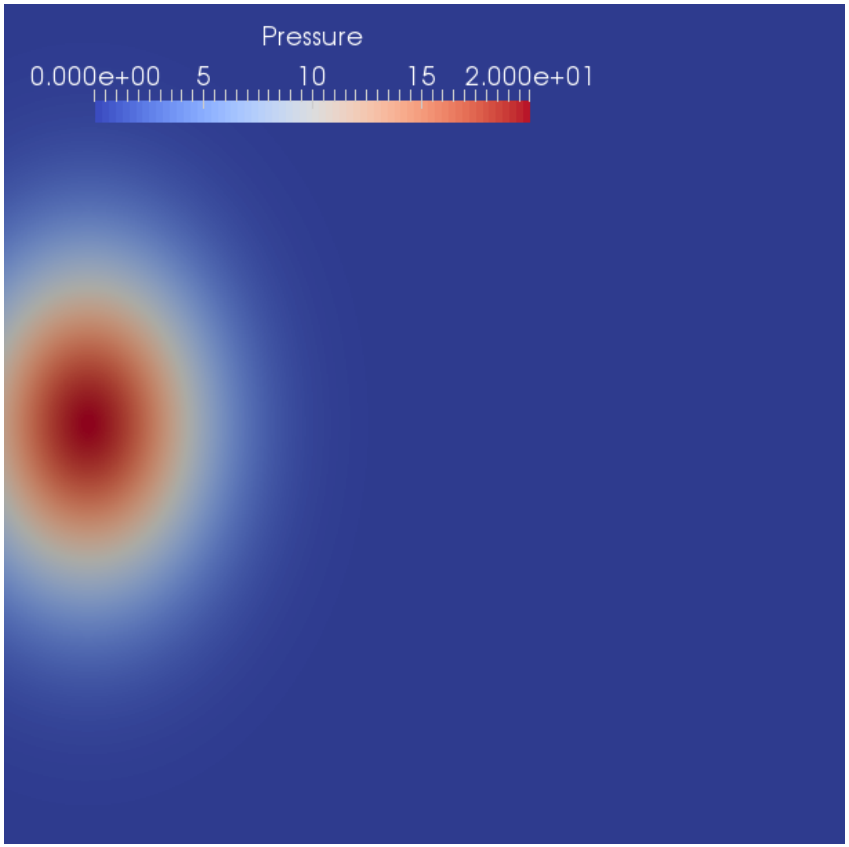
Nonlinear least squares: One input variable, top boundary condition.

Example: Driven Cavity

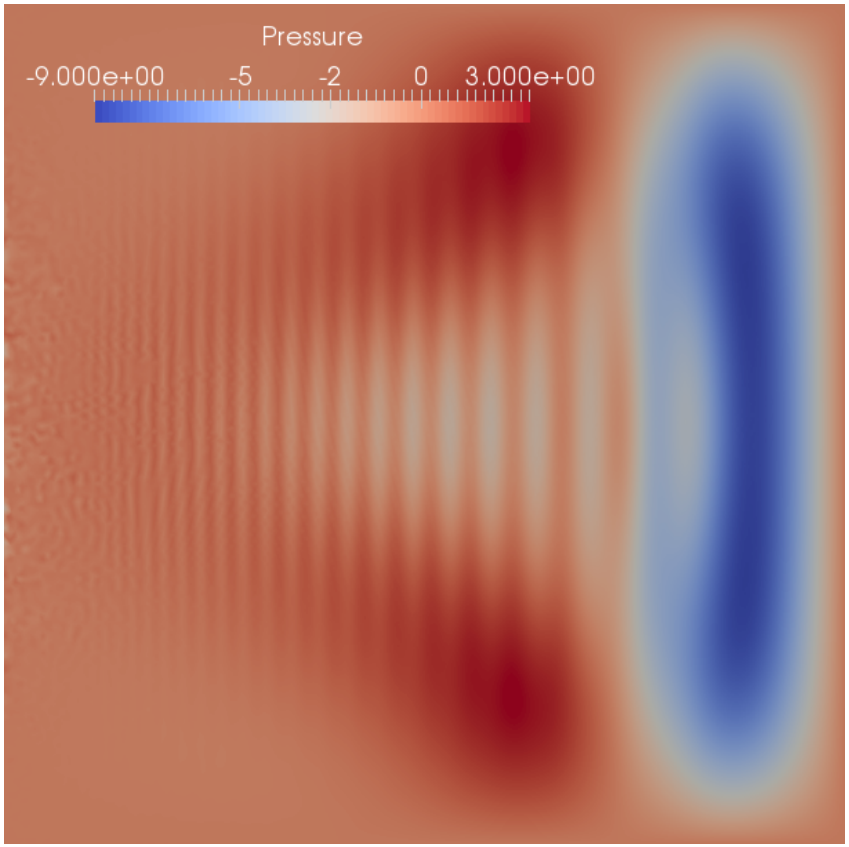


Nonlinear least squares: One input variable, top boundary condition.

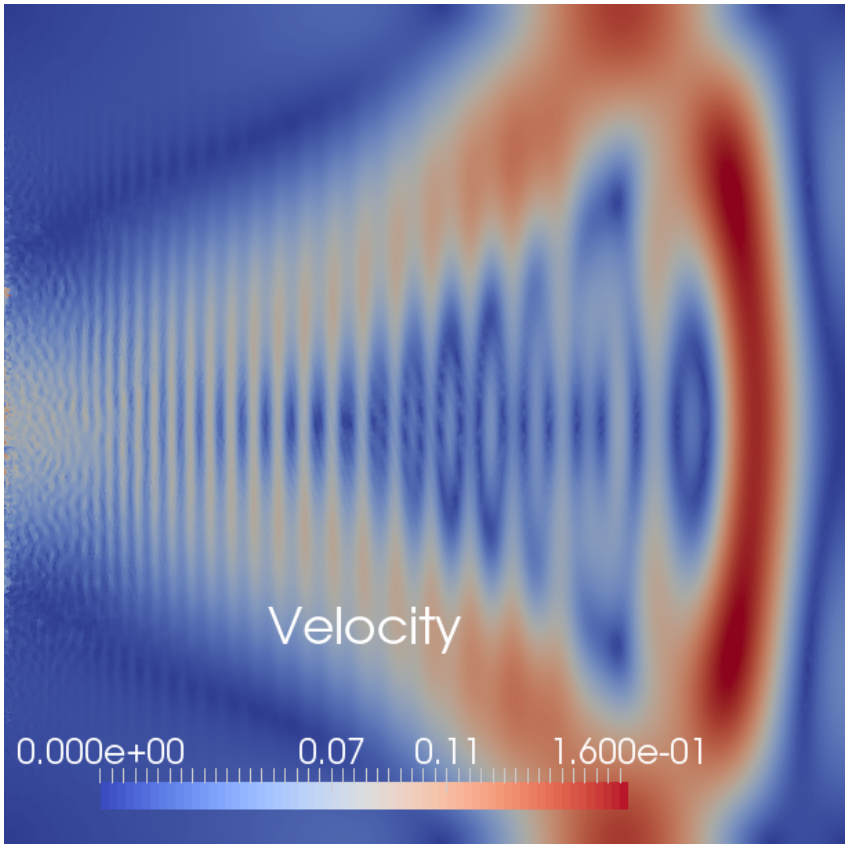
Example: Tsunami



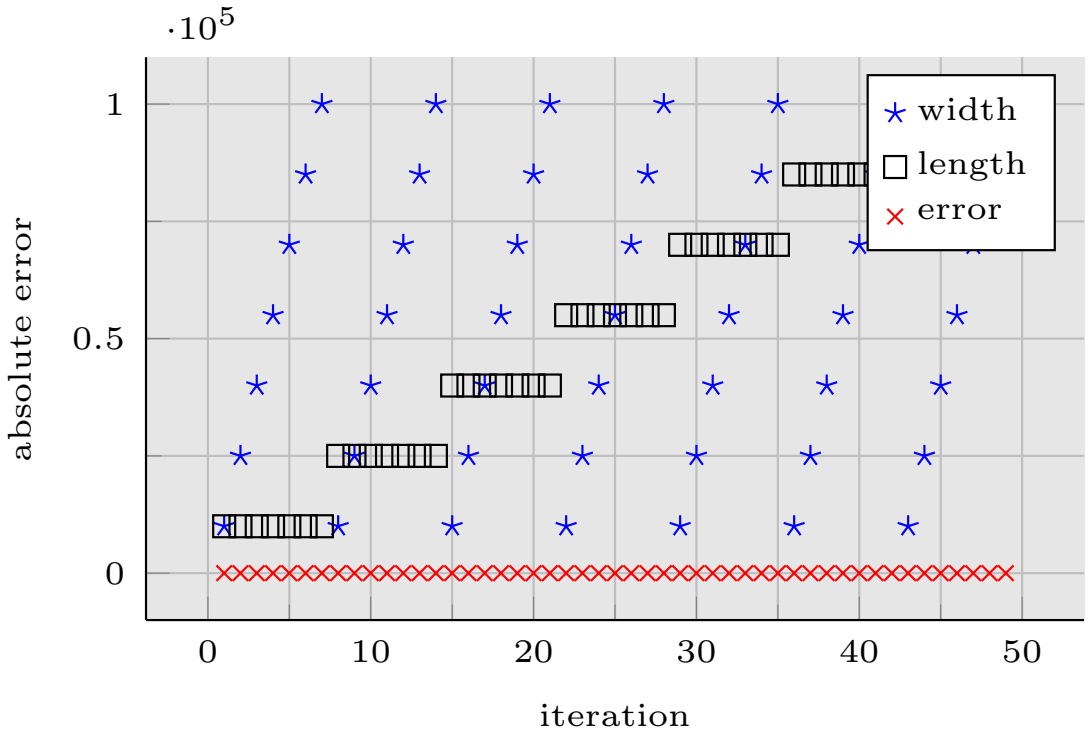
Example: Tsunami



Example: Tsunami

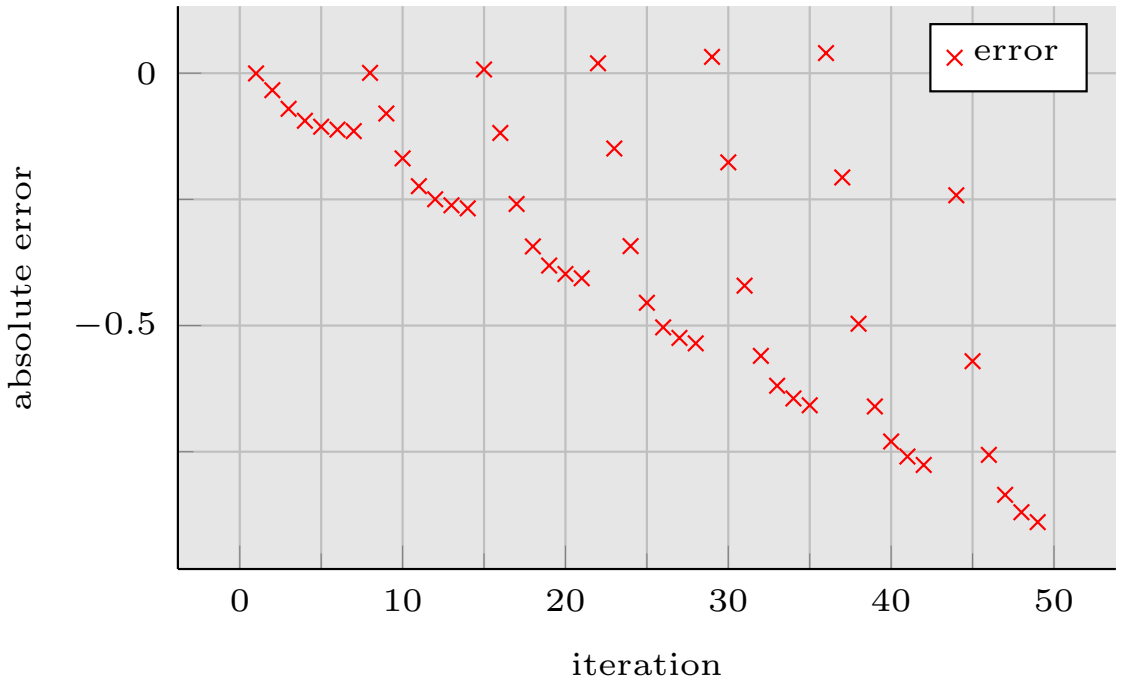


Example: Tsunami



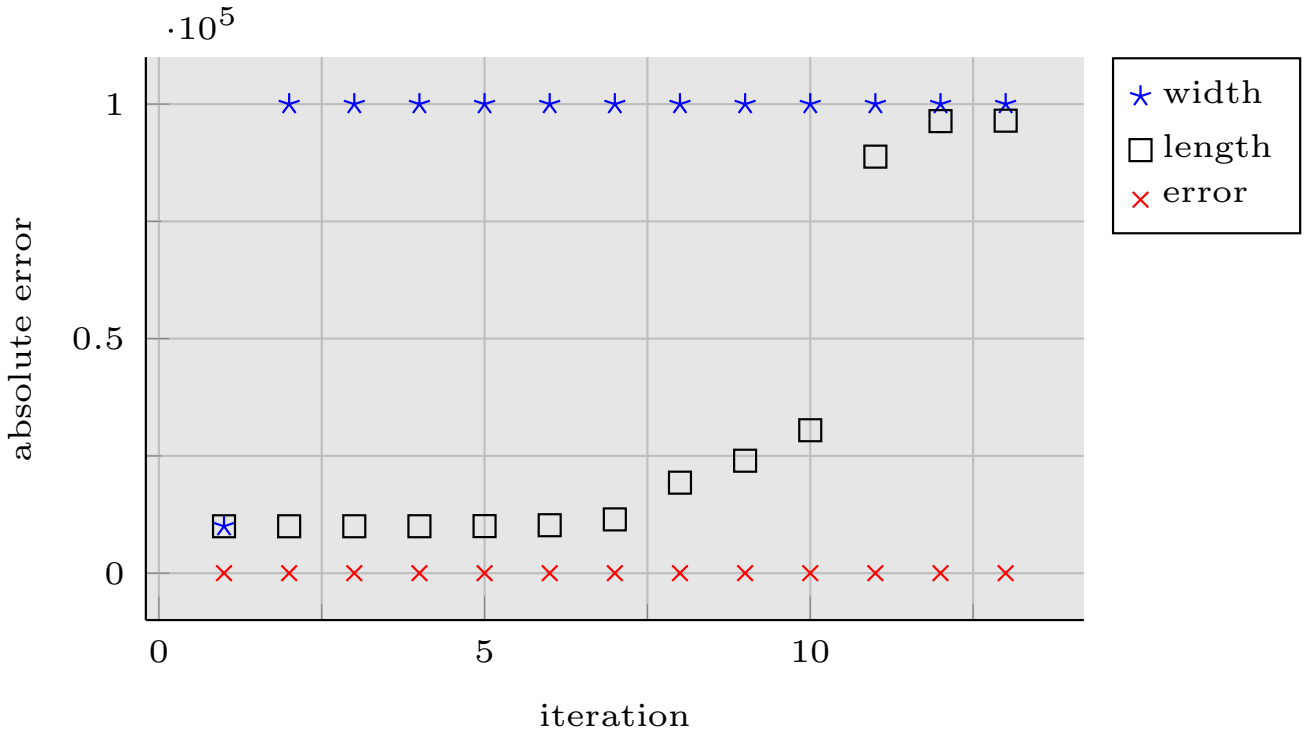
Multidimensional parameter study: Varying initial *width* and *length* of perturbation.

Example: Tsunami



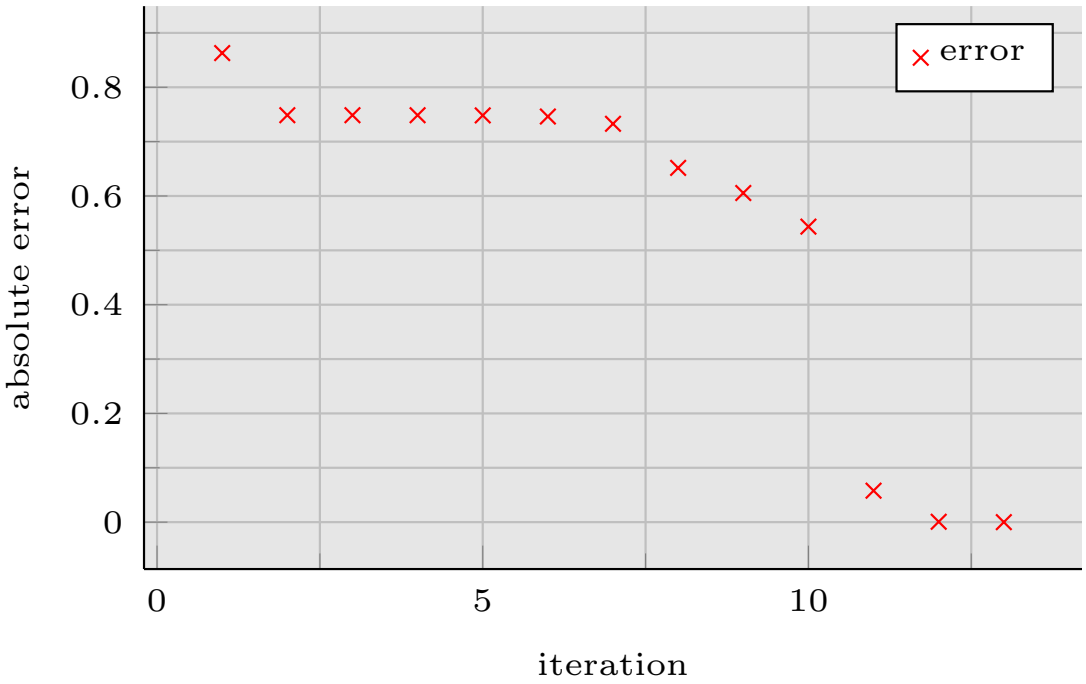
Multidimensional parameter study: Varying initial *width* and *length* of perturbation.

Example: Tsunami



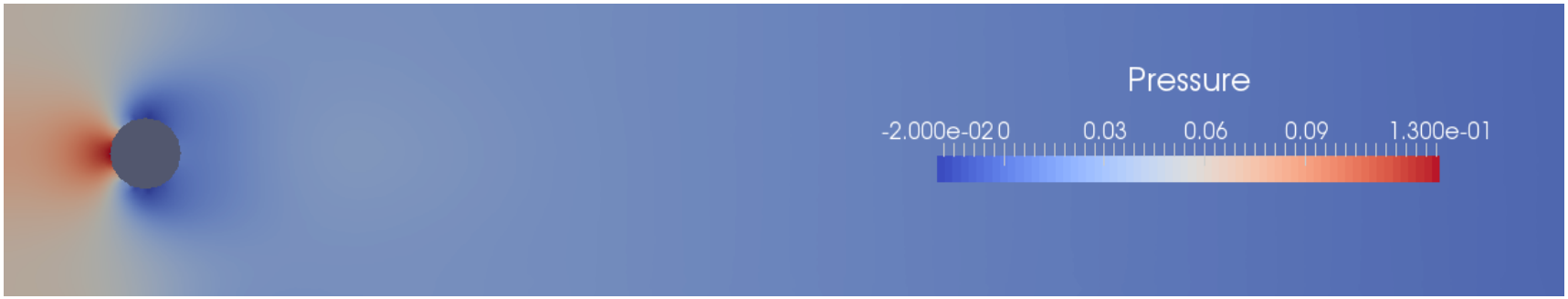
Nonlinear least squares: Varying initial *width* and *length* of perturbation.

Example: Tsunami

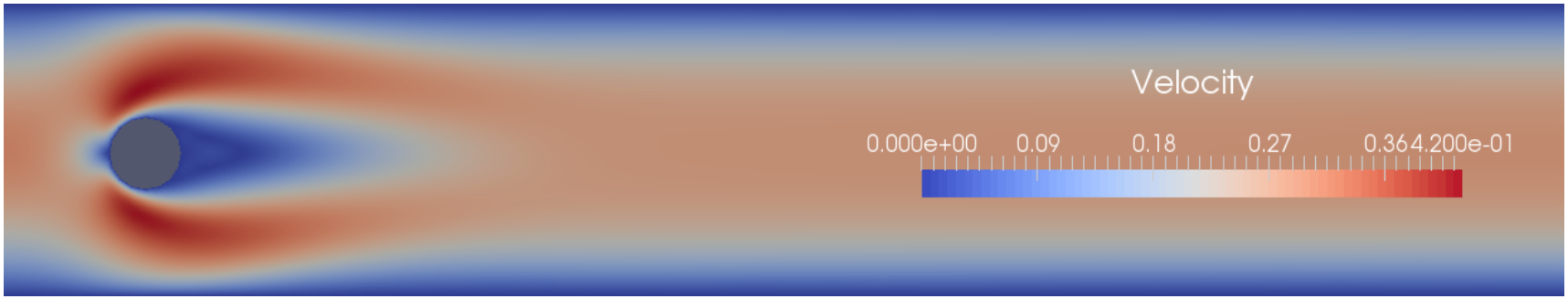


Nonlinear least squares: Varying initial *width* and *length* of perturbation.

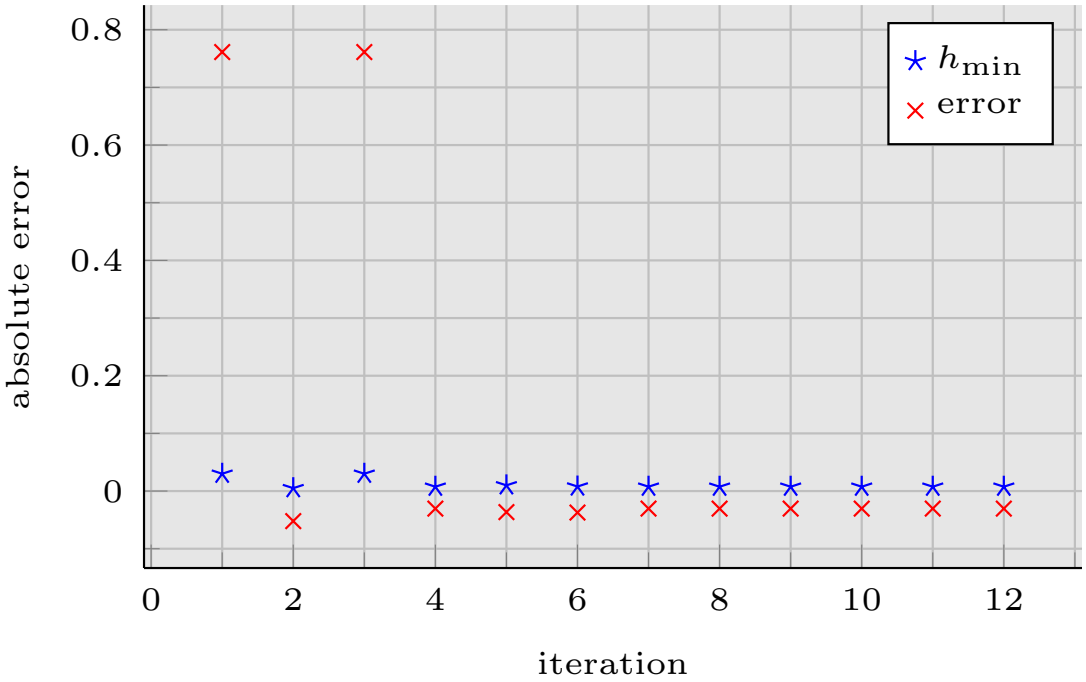
Example: Flow past cylinder



Example: Flow past cylinder

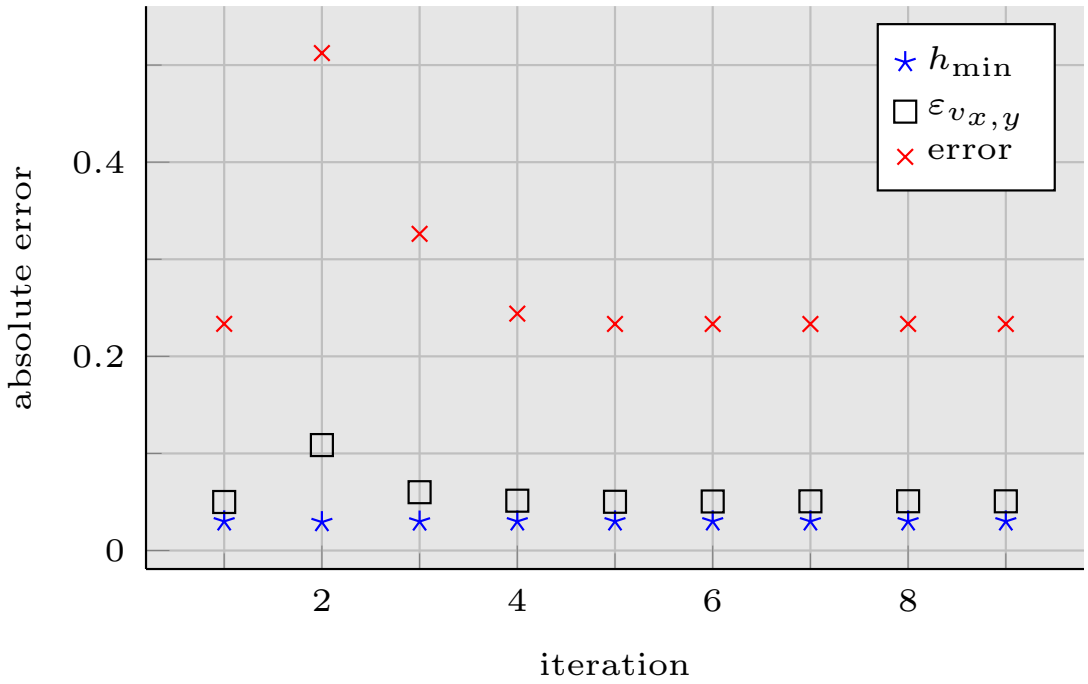


Example: Flow past cylinder



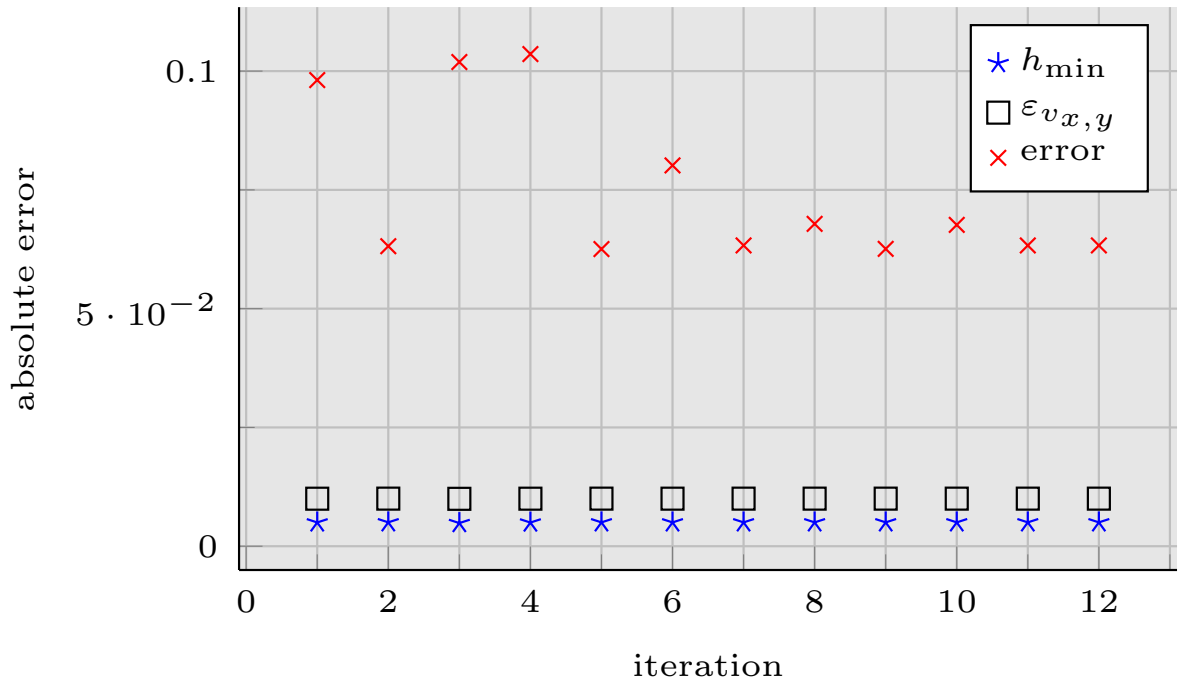
Nonlinear least squares: Varying initial element edge lengths of mesh.

Example: Flow past cylinder



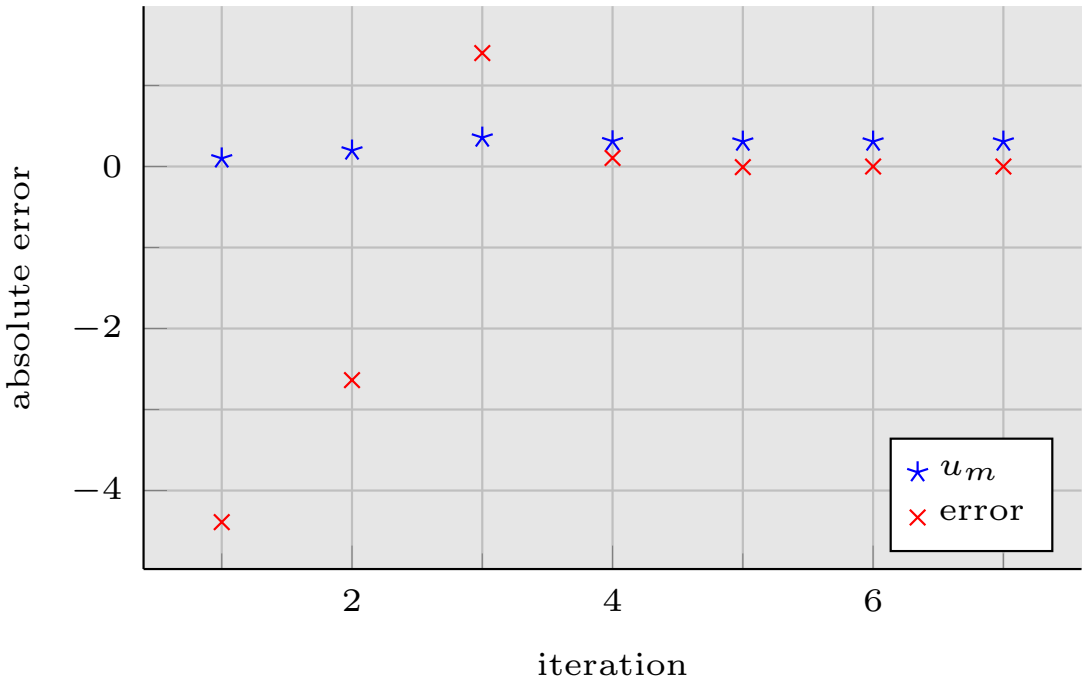
Nonlinear least squares: Varying mesh adaptivity parameters: $h_{\min}$ and $\varepsilon_{v_{x,y}}$, with initial values $h_{\min,0} = 0.03$, $\varepsilon_{v_{x,y},0} = 0.05$.

Example: Flow past cylinder



Nonlinear least squares: Varying mesh adaptivity parameters: $h_{\min}$ and $\varepsilon_{v_{x,y}}$, with initial values $h_{\min,0} = 0.005$, $\varepsilon_{v_{x,y},0} = 0.01$.

Example: Flow past cylinder



Nonlinear least squares: Varying inlet flow boundary condition,

$$U_m = ?,\ H = 0.41\,\mathrm{m},\ u_x = 4U_m z(H-z)/(H^2)$$

Conclusion

- Capable of performing gradient and non-gradient optimisations
- Easy to use with external packages, e.g. Fluidity
- Examples are can be found online
- Fluidity specific setups can be found at: http://github.com/fmilthaler/DAKOTA-Fluidity-examples

Thank you

Good bye

Good bye

and see you soon...

Good bye

and see you soon...

at the pub!