get it right®

Dakota OpenFOAM vector_parameter_study Tutorial

December 6, 2013

www.esi-group.com

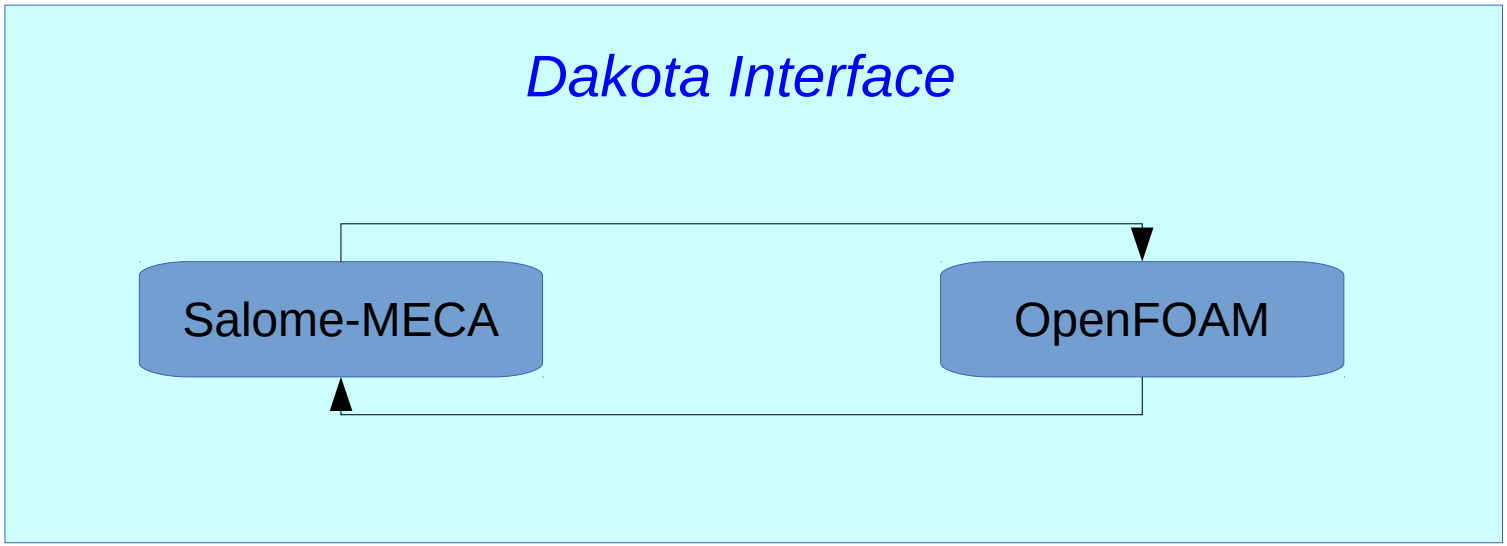
AGENDA

- Introduction
- Test case
- Dakota file structure
- Setting up files
- Dakota Output
- Compiling & Links

Dakota Project

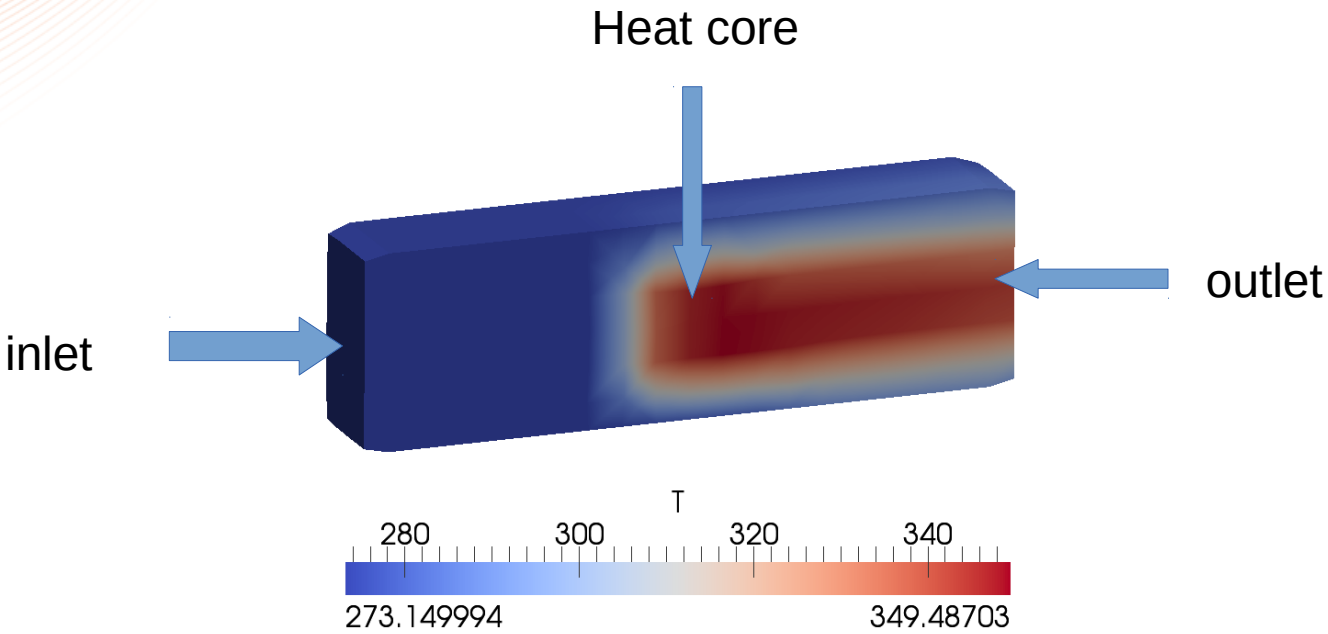
Design Analysis Kit for Optimization and Terascale Applications

- an open source toolkit that provides a flexible, extensible interface between analysis codes and iteration methods
- Useful tool for parametric and optimization studies.



- developed by Sandia National Laboratories

Test case

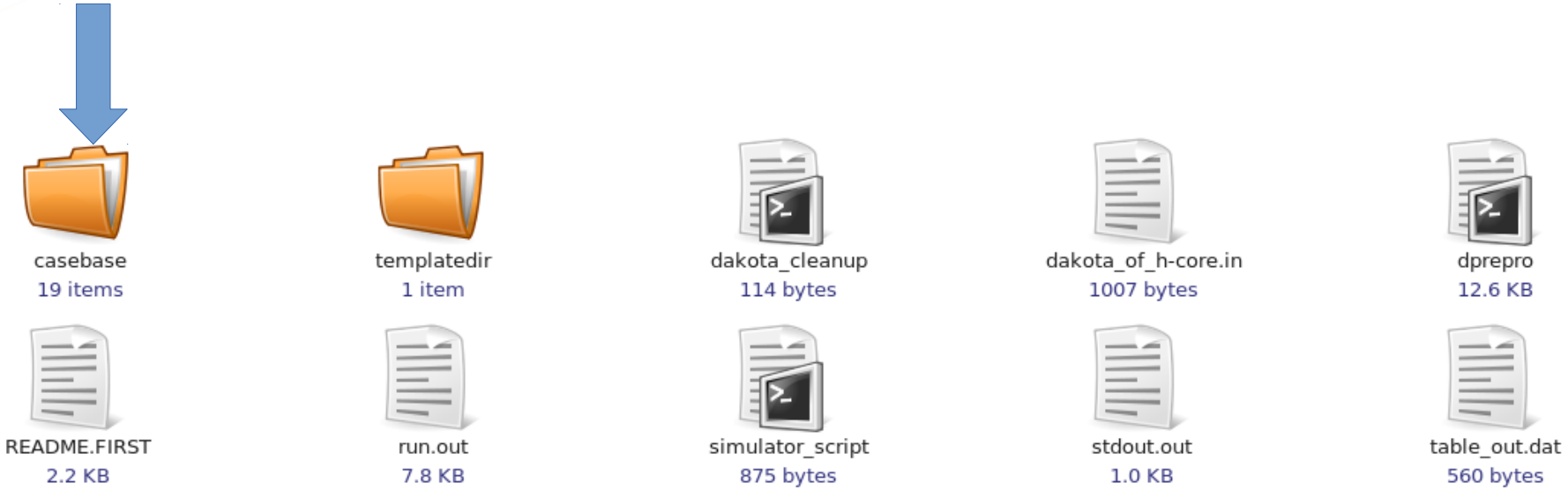


We will couple Dakota with openFoam to do a parametric study

- Objective function : Average Temperature at the outlet
- Design parameter : velocity at the inlet.

Dakota File Structure

This is the initial files structure of our Dakota case. We start by copying the openFOAM case (0, constant, system directories and all the files like effTable) inside the casebase directory



In this tutorial we will modify the input files directly since we just have one parameter, for more complicated cases you can use the Jaguar GUI.

Set up dakota_of_h-core.in

We ask Dakota to perform the analysis with an inlet Velocity range between 0.1 and 5m/s, and to execute 12 steps.
It will produce a Vinlet vector parameter of num_steps +1 elements (see following slide)

```
# Usage:
#   dakota -i xxx.in -o run.out > stdout.out

strategy
    graphics
    tabular_graphics_data
        tabular_graphics_file = 'table_out.dat'
    single_method

method
  vector_parameter_study
    final_point = 5

    num_steps = 12

model
  single

variables,
    continuous_design = 1
      initial_point    0.1
      descriptors       'x1'

interface,
    fork
#     asynchronous
      analysis_driver = 'simulator_script'
```

Method:
Vector parameter

Inlet velocity range

The design parameter 'x1'
in our case the inlet velocity

simulator_script is the script where we tell Dakota what to do in each directory case (ex. Launch rhoSimpleFoam) and which results to extract.

Set up of U.template

In the templateDir we save the following U.template file:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform ({x1} 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Dakota will set in {x1} the values of the design parameter as specified in the dakota_of_h-core.in file

Vector Parameter
Velocity inlet
(13 elements)

```
x1
0.1
0.5083333333
0.9166666667
1.325
1.733333333
2.141666667
2.55
2.958333333
3.366666667
3.775
4.183333333
4.591666667
5
```

Set up simulator_script

```
dprepro $1 U.template U.in

# --------
# ANALYSIS
# --------

pwd
cp -r ../casebase/* .
cp U.in 0/U

rhoSimpleFoam > log.rhoSimpleFoam

# ---------------
# POST-PROCESSING
# ---------------


#touch results.out
#cp ../casebase/results.out .

cat log.rhoSimpleFoam | grep 'areaAverage(outlet) for T' | cut -b32-39 | tail -1  >> tmp.txt
mv tmp.txt $2
```

$1 is params.in from DAKOTA

Using U.Template Dakota will set the new inlet velocity into the 0/U of the n step case (copied from the casebase)

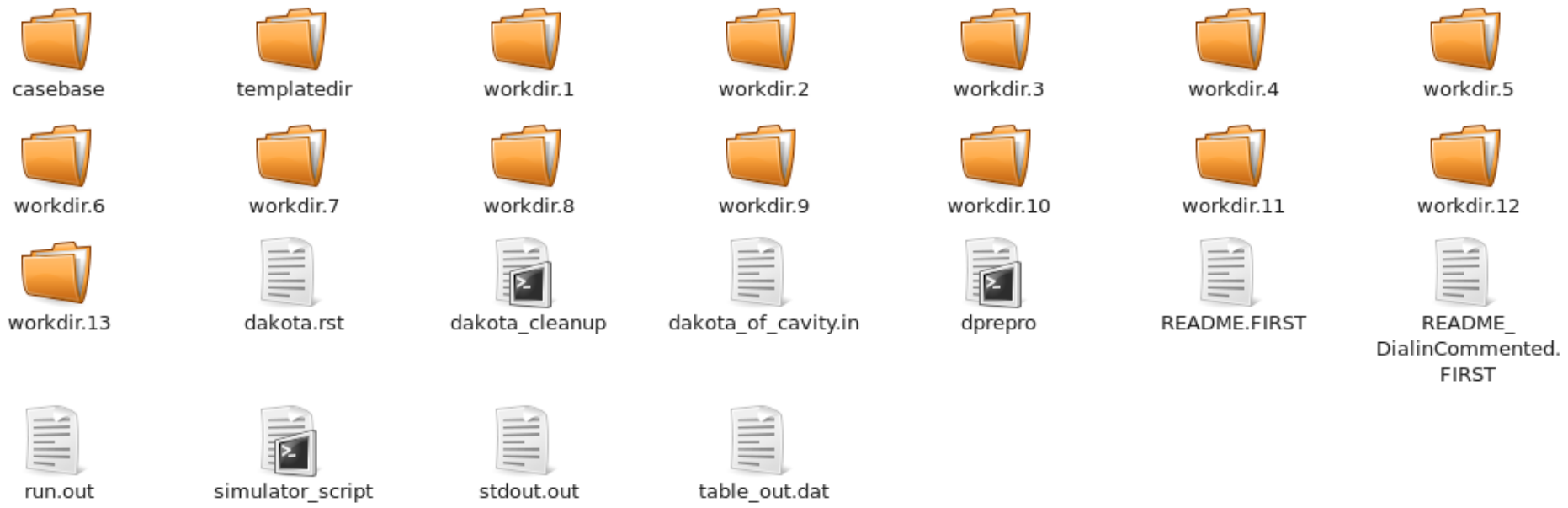
Execution of the analysis

We extract the average T value @outlet from the log file and we give the value to Dakota output $2

$2 is results.out returned to Dakota

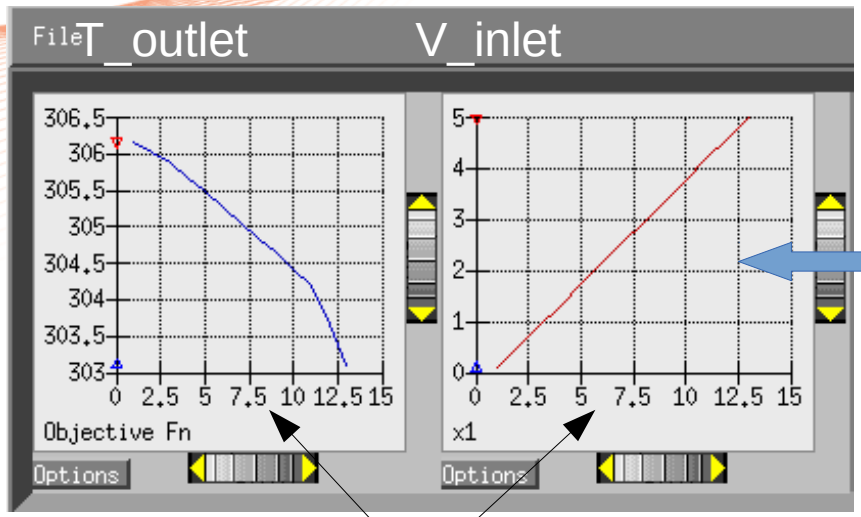
Dakota final File Structure

Since we have set step=12, Dakota has created 13 workdir:



It has: copy the openFOAM case from casebase dir, create workdir*, and launched the analysis for each of those directories with the specific velocity inlet within the range specified in the dakota_of_cavity.in file.

Dakota Output



Dakota has launched the case 13 times

velocity at inlet
from 0.1 to 5m/s

%eval_id

%eval_id	V_inlet x1	T_outlet obj_fn
1	0.1	306.18
2	0.5083333333	306.068
3	0.9166666667	305.892
4	1.325	305.695
5	1.733333333	305.489
6	2.141666667	305.28
7	2.55	305.069
8	2.958333333	304.855
9	3.366666667	304.641
10	3.775	304.425
11	4.183333333	304.209
12	4.591666667	303.659
13	5	303.106

Saved in table_out.dat

And we get as output the objective function (temperature average at the outlet).
The more the inlet velocity increases, the time in contact with the h-core reduces, so we will get a lower temperature at the outlet.

If you want to relaunch the analysis to clean up everything type in terminal:
./dakota_cleanup

Compiling & Links

Installation Susume

If after installing the DAKOTA binaries, you experience library errors while running it with OpenFOAM, you need to compile DAKOTA from source.
REASON: **OpenFOAM** uses a library called "**libsampling.so**" which has the **same name** of a **DAKOTA library** (see the DAKOTA_installation_dir/lib directory). To use Dakota and OpenFOAM together you need to compile Dakota using static libraries

SOLUTION: In the Dakota CmakeLists.txt set:
instead of
option(BUILD_SHARED_LIBS "Build shared libraries?" ON) #145 line
set libraries as static in the following way:

```
# option(BUILD_SHARED_LIBS "Build shared libraries?" ON)
# Build static libraries ONLY
set(BUILD_STATIC_LIBS ON CACHE BOOL "Set to ON to build static libraries" FORCE)
```

In this way instead of *.so -> you will get static libraries (*.a) and Dakota and OF can work together.

Compiling & Links

Useful links

- Dakota download:

http://dakota.sandia.gov/download.html

- Some simple tutorials to get started with DAKOTA+OF2.2

http://www.dicat.unige.it/guerrero/dakotaof.html

- my post installation instructions for CentOS 6.4

http://www.cfd-online.com/Forums/openfoam-programming-development/72558-dakota-openfoam.html#post463033

www.esi-group.com