December 5, 2019

LINKÖPING UNIVERSITY

HOPSAN

Tutorial - Optimizing Models With modeFRONTIER

Introduction

This tutorial will demonstrate how to run Hopsan simulations from modeFRONTIER[1], a multi-objective optimization environment. This makes it possible to use powerful optimization algorithms and data analysis tools on a Hopsan model. It is also possible to connect other programs to modeFRONTIER simultaneously and run multi-disciplinary optimizations. This is, however, not covered by this tutorial.

Requirements

In this tutorial modeFRONTIER version 4.5.3 and Hopsan version 0.6.8 on a Windows operating system are used. It should, however, work on Linux systems with some minor modifications. Some basic knowledge of Hopsan and design optimization is also required. It is recommended to do the "Getting Started", "Advanced Usage" and "Optimization" tutorials before this one.

Optimize using final values only

An optimization basically consists of three parts: A simulation model, one or more objective function(s), and an optimization algorithm. In this case Hopsan will be responsible for the simulation, while modeFRONTIER will take care of the algorithm. The evaluation of the objective function(s) can, however, be placed in either one of the programs. We will demonstrate both methods.

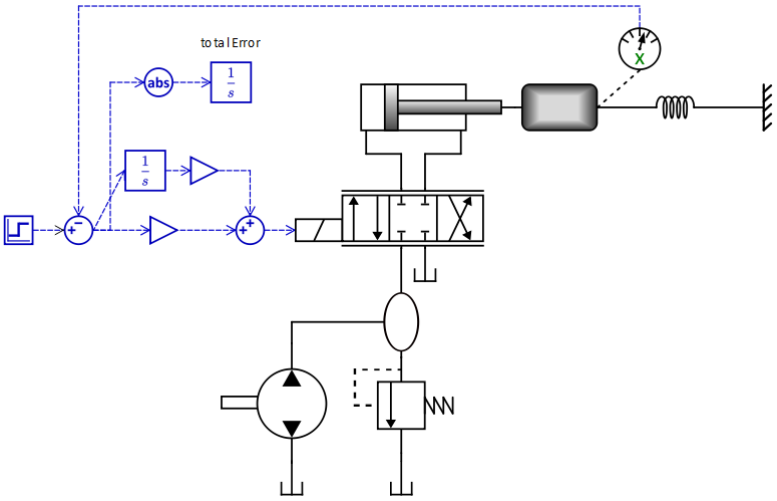
Calculating objective function values requires access to the resulting data variables from the simulation. Letting Hopsan do the calculation can therefore be easier, since the amount of data that must be send between the programs can be reduced. We will first show how to run an optimization by only sending the final simulation values from Hopsan.

1. **Preparations**
   For this example we will use the position servo example model. The objective is to optimize the parameters of the PI-controller by analyzing a step response. First we must modify the model so that it also calculates the objective function value.

   - Open the position servo example
   - Add an absolute value operaetor and an integrator
   - Connect them after the error signal and name the integrator "totalError"
   - Create a new empty folder
   - Save the model as "PositionServo.hmf" in the folder you created

   It should now look similar to the picture below:

[1] modeFRONTIER is the property of ESTECO SpA. The Hopsan developers are not affiliated with modeFRONTIER or ESTECO SpA.



Hopsan and modeFRONTIER will communicate by using comma-separated files (.csv). Before we create the modeFRONTIER project we will need to extract two files from Hopsan to use as templates.

- Open a command prompt in Windows
- Go to the folder you created
- Execute HopsanCLI with the following command (change the Hopsan installation path to the one you have):

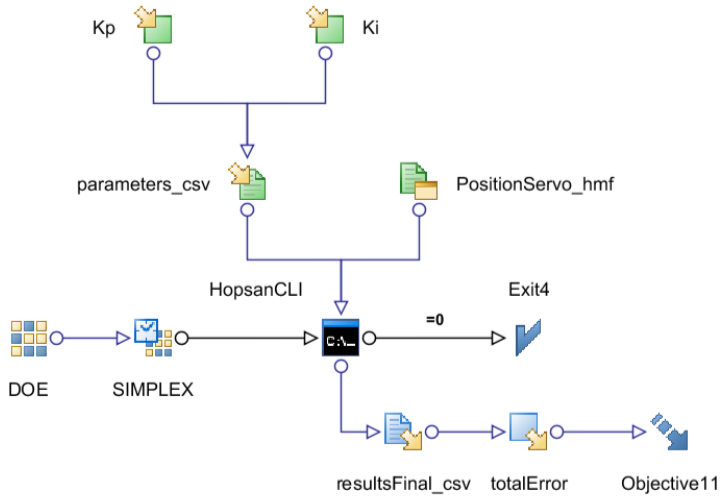
```
c:\"Program Files (x86)"\hopsan\bin\hopsancli --hmf PositionServo.hmf
    --simulate hmf --resultsFinalCSV resultsFinal.csv
    --parameterExport parameters.csv
```

This will open the model, simulate it and generate .csv files for parameters and final values of the data variables.

2. Build the modeFRONTIER project

Now it is time to build the project in modeFRONTIER.

- Open modeFRONTIER
- Create a new project and save it in the folder you created
- Add the following process nodes:
  - Scheduler
  - DOS Batch Script
  - Logic End
- Add the following nodes for input data:
  - 2x Input Variable
  - Input File
  - Support File
- Add the following nodes for output data:
  - Output File
  - Output Variable
  - Design Objective
- Connect the nodes as shown in the picture below



3. **Specify input variables**
   We will use two input variables, one for the proportional gain (*Kp*) and one for the integrating gain (*Ki*) in the PI controller.

   - Double-click on one of the input variable nodes

     Input Variable

   - Change *"Name"* to *"Kp"*
   - Set *"Lower Bound"* to 0.0 and *"Upper Bound"* to 0.01
   - Double-click on the second input variable node
   - Change name to *"Ki"*, lower bound to 0.0 and upper bound to 0.005

4. **Configure the input variable file**
   Input variables shall be written to the "`parameters.csv`" file by modeFRONTIER, which is in turn loaded by Hopsan.

   - Double-click on the input file node

     Input File

   - Set *"Input File Node Name"* to *"parameters_csv"*
   - Click on *"Edit Input File"*
   - Browse to the your folder and select "`parameters.csv`"
   - Select *"Kp"* in the list at the bottom
   - Find the row that begins with *"PositionServo$GainP#k#Value"*
   - Select the numerical value at the end of the row
   - Right-click and choose *"Insert Variable"*
   - Select *"Ki"* in the list at the bottom
   - Find the row that begins with *"PositionServo$GainI#k#Value"*
   - Select the numerical value at the end of the row
   - Right-click and choose *"Insert Variable"*
   - Click *"Ok"* twice to close the dialogs

5. **Configure the Hopsan model file**
   We must also specify the model file used by Hopsan.
   - Double-click on the support file node

      ***Support File***
   - Set *"Support File Node Name"* to *"PositionServo_hmf"*
   - Click on *"Add File"* and select "`PositionServo.hmf`"
   - Click *"Ok"*

6. **Specify the output variable**
   The output variable consist of the integrated error signal from the Hopsan model.
   - Double-click on the output variable node

      ***Output Variable***
   - Set *"Name"* to *"totalError"*
   - Click *"Ok"*

7. **Configure the output file**
   The output value is read by modeFRONTIER from "`resultsFinal.csv`", which contains the final values exported from Hopsan.
   - Double-click on the output file node

      ***Output File***
   - Set *"Output File Node Name"* to *"resultsFinal_csv"*
   - Click on *"Open Output File"*
   - Select "`resultsFinal.csv`"
   - Find the row that starts with *"PositionServo$totalError#out#Value"*
   - Select *"totalError"* at the bottom
   - Select the text *"PositionServo$totalError#out#Value"*
   - Right-click and choose *"Select Relative"*
   - Click *"Ok"* twice

8. **Define the objective function**
   The objective function in modeFRONTIER will be to simply minimize the value obtained from Hopsan.
   - Double-click on the design objective node

      ***Design Objective***
   - Change from *"Maximize"* to *"Minimize"*
   - Click *"Ok"*

9. **Write a script for controlling HopsanCLI**
   Hopsan will be controlled through the command line interface (`hopsancli.exe`) from a batch script in modeFRONTIER.

   - Double-click on the script node

     
     DOS Batch Script

   - Set *"Name"* to *"hopsancli"*
   - Click on *"Edit DOS Batch Script"*
   - Enter the following command:

     ```
     C:\"Program Files (x86)"\Hopsan\bin\hopsancli --hmf PositionServo.hmf
         --simulate hmf --resultsFinalCSV resultsFinal.csv
         --parameterImport parameters.csv
     ```

     This will tell HopsanCLI to load the specified model, load parameters from `parameters.csv`, simulate the model and then write the final values of all data variables to `resultsFinal.csv`.
   - Click *"Ok"* twice

10. **Choose initial distribution and algorithm**
    Finally, we must configure the optimization in modeFRONTIER. For this example we will use a random initial distribution and the Simplex algorithm. It is of course possible to use other distributions and algorithms as well.

    - Double-click on the DOE node

      
      DOE

    - Select *"Random"* under *"Space Fillers"* to the left
    - Click on *"Add DOE Sequence"*
    - Click *"Ok"*
    - Double-click on the scheduler node

      
      Scheduler

    - Select *"Simplex"* under *"Heuristic Optimizers"*
    - Click *"Ok"*

11. **Run an optimization**
    Now everything is prepared for starting the optimization!

    - Make sure the model is saved
    - Go to the *"Run Analysis"* mode
    - Start the optimization by clicking on the green arrow to the top right of the workspace

      
      Run/Stop

Optimizing using full variable export

Putting the objective function in Hopsan is not always possible (or desirable). Another method is to export the data to modeFRONTIER instead, so that the calculation can be made there.

1. **Save a copy of the model**
   Save the model from the previous section in the same folder but with a different name.

2. **Generate output variable file**
   First we need to generate a new .csv file to use as a template for the data.

   - Open the command prompt in Windows and go to your folder
   - Start HopsanCLI with the following command:

   ```
   c:\Program Files (x86)"\hopsan\bin\hopsancli --hmf PositionServo.hmf
       --simulate hmf --resultsFinalCSV resultsFinal.csv
   ```

   This will generate a file called "`resultsFinal.csv`", which we will use below.

3. **Modify the HopsanCLI script**
   Now we must modify the script where modeFRONTIER calls HopsanCLI, so that the full result file is updated after each simulation.

   - Double-click on the script node

     DOS Batch Script

   - Click on *"Edit DOS Batch Script"*
   - Modify the command to the following:

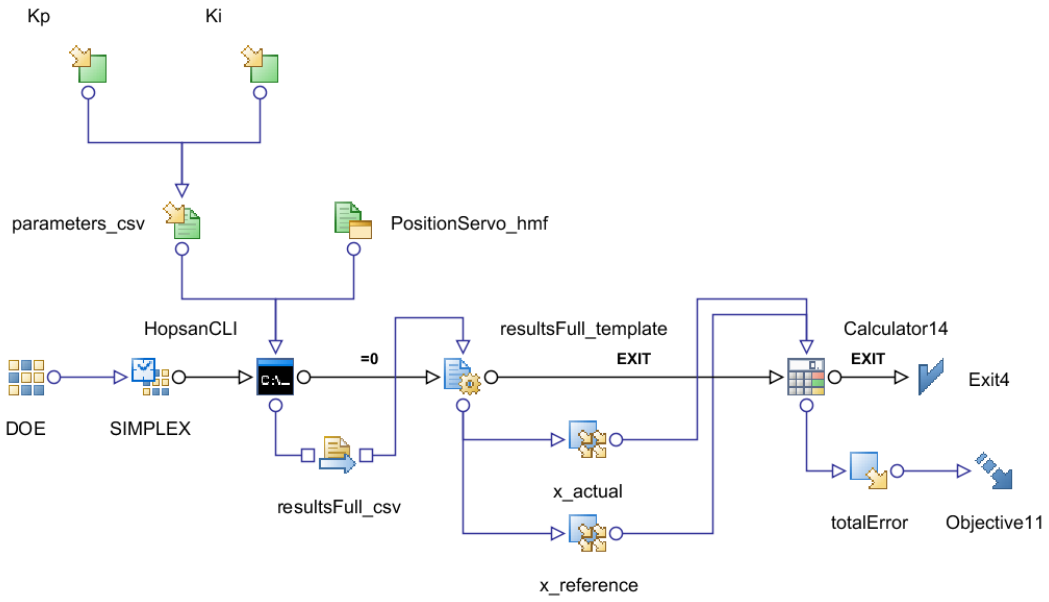
   ```
   C:\"Program Files (x86)"\Hopsan\bin\hopsancli --hmf PositionServo.hmf
       --simulate hmf --resultsFullCSV resultsFull.csv
       --parameterImport parameters.csv
   ```

   - Click *"Ok"* twice

4. **Modify the project**
   The project in modeFRONTIER must be modified, so that the objective function is calculated.

   - Remove the output file node (called *"resultsFinal_csv"*)
   - Add an Output Template node and a Calculator node between the script and the exit nodes
   - Add a Transfer File node between the Script node and the Output Template node
   - Add two Output Vector nodes between the Output Template node and the Calculator node
   - Connect the calculator node with the existing *"totalError"* node

   The model should now look like this:



5. **Specify output variables**
   We will use two output variables from Hopsan, the reference piston position and the actual position.
   - Double-click on one of the output vector nodes

      ***Vector Output Variable***
   - Set *"Name"* to *"x_actual"*
   - Set *"Size"* to 2048
   - Click *"Ok"*
   - Double-click on the second output vector node
   - Set *"Name"* to *"x_reference"*
   - Set *"Size"* to 2048
   - Click *"Ok"*

6. **Configure the output varible file**
   The output variables must be obtained from the generated .csv file.
   - Double-click on the transfer file node

      ***Transfer File***
   - Set *"Transfer File Node Name"* to *"resultsFull_csv"*
   - Click on *"Add File"*
   - Browse to your folder and select "`resultsFull.csv`"
   - Click *"Ok"* twice

7. **Configure the data mining**
   Now we must specify the data mining, to map each variable to a block in the .csv file.
   - Double-click on the output template node

      ***Output Template***
   - Set *"Output Template Node Name"* to *"resultsFull_template"*
   - Click on *"Edit Output Template"*
   - Browse to your folder and select "`resultsFull.csv`"
   - Click on *"x_actual"* at the bottom
   - Find the row that starts with *"PositionServo$Position_Sensor#out#Value"*
   - Select the text, right-click and choose *"Add Rule For"* →*"x_actual"*
   - Select the text again, right-click and choose *"Set Anchor"* →*"x_actual"*
   - Uncheck column 1, 2, 3 and 4 to the left
   - Click on *"x_reference"* at the bottom
   - Find the row that starts with *"PositionServo$Step#out#Value"*
   - Select the text, right-click and choose *"Add Rule For"* →*"x_reference"*
   - Select the text again, right-click and choose *"Set Anchor"* →*"x_reference"*
   - Uncheck column 1, 2, 3 and 4 to the left
   - Click *"Ok"* twice

8. **Define the objective function calculation**
   Finally, we must also specify the equation for the objective function.

   - Double-click on the calculator node

      ***Calculator***

   - Set *"Name"* to *"objective_function"*
   - Click on *"Edit Calculator Expression"*
   - Enter the following expression:

     ```
     totalError = sum(abs(subtract(x_actual, x_reference)))
     ```

   - Click *"Ok"* twice to close the dialog

9. **Save the model and run an optimization**
   If everything was successful, the optimization should work and give reasonable results. This method should, however, be less time-efficient, due to the large amount of data being exchanged between the programs.