UNIVERSITY OF ICELAND

Wind integration in Iceland

PRELIMINARY RESULTS

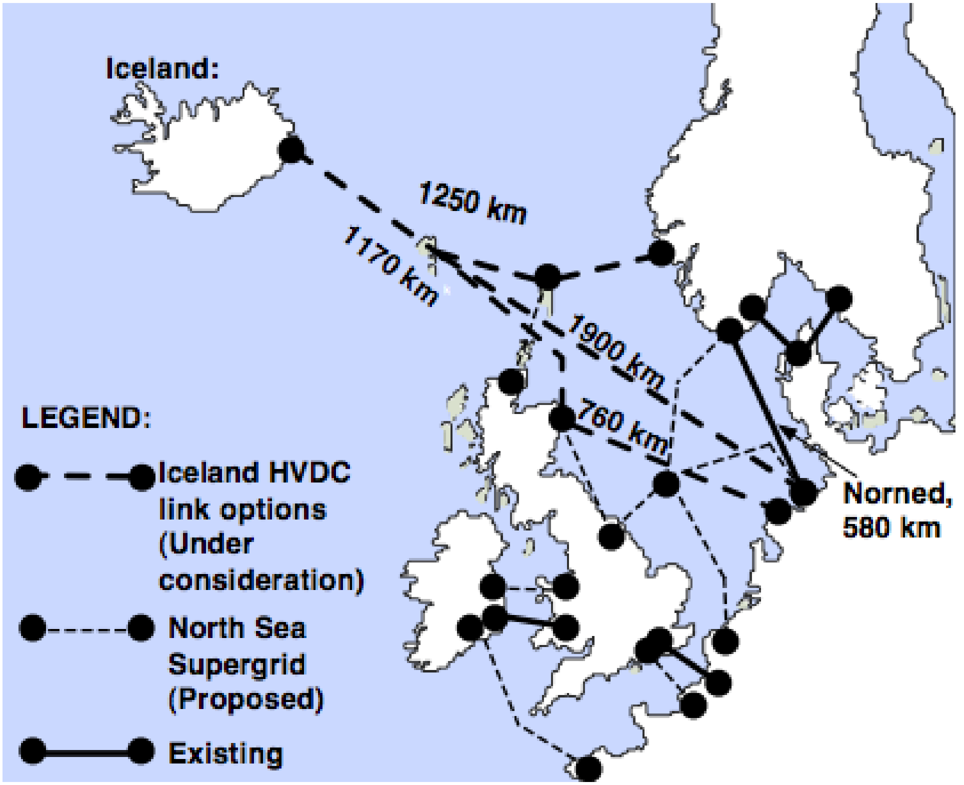
GUNNAR GEIR PÉTURSSON PH.D. STUDENT

Why

Wind power is an untapped resource in Iceland, despite the fit of this resource within the highest wind power class in Western Europe (Nawry *et al*, 2013).

If a submarine cable between Europe and Iceland should become reality, the relatively high energy prices will support more wind development.

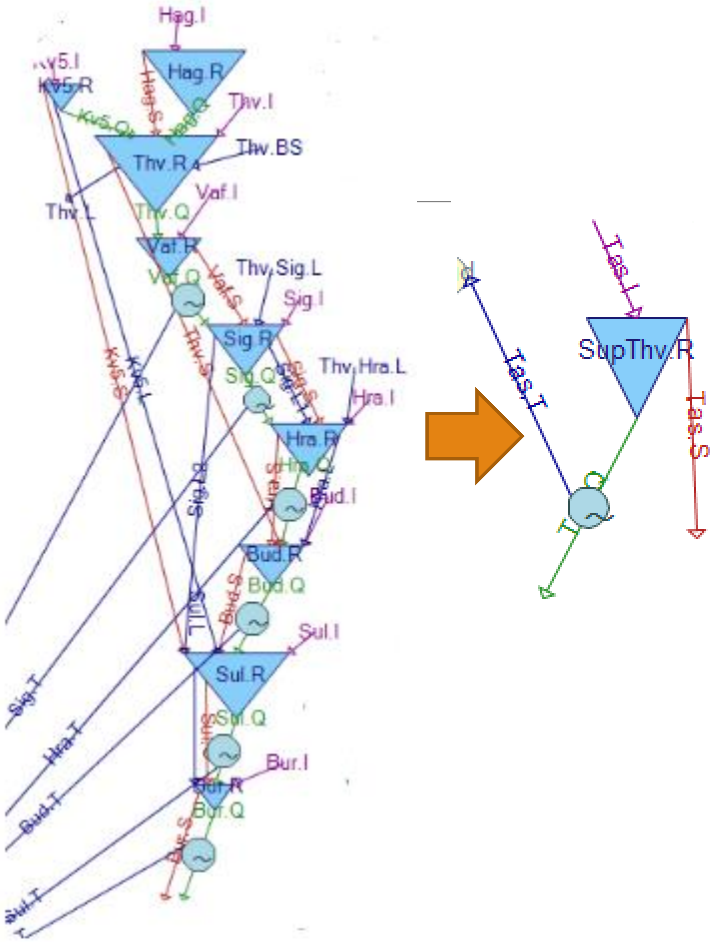
In order to measure the performance of a wind farm within a power system a tool that simulates and runs the system in an optimal way is a necessity.



Why (continued)

Wind power can fluctuate drastically within the hour while the only medium/long term tool in Iceland uses time steps of 1 week since that was enough during its development.

A tool that does not aggregate cascades of reservoirs into one is also desirable for future studies on the Icelandic power system (this is routinely done for computational purposes).

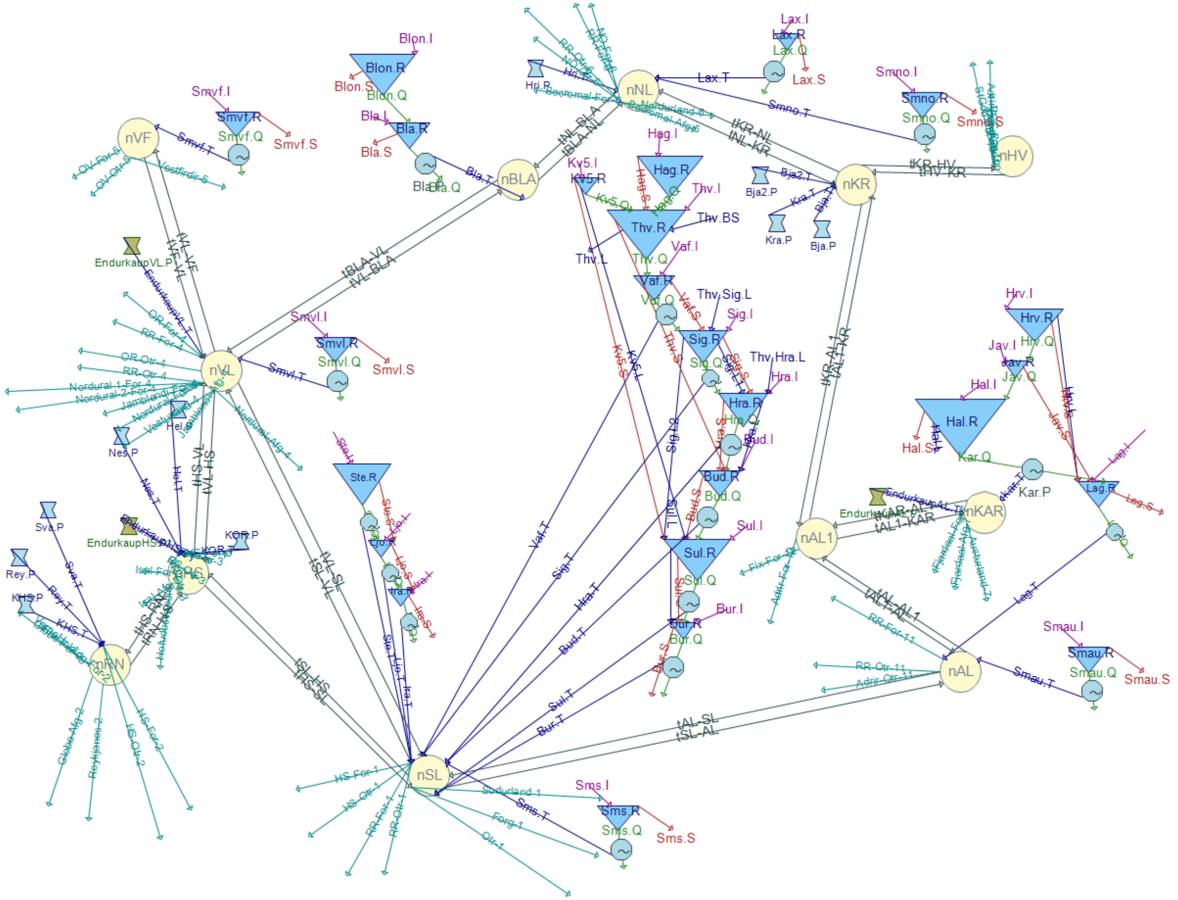


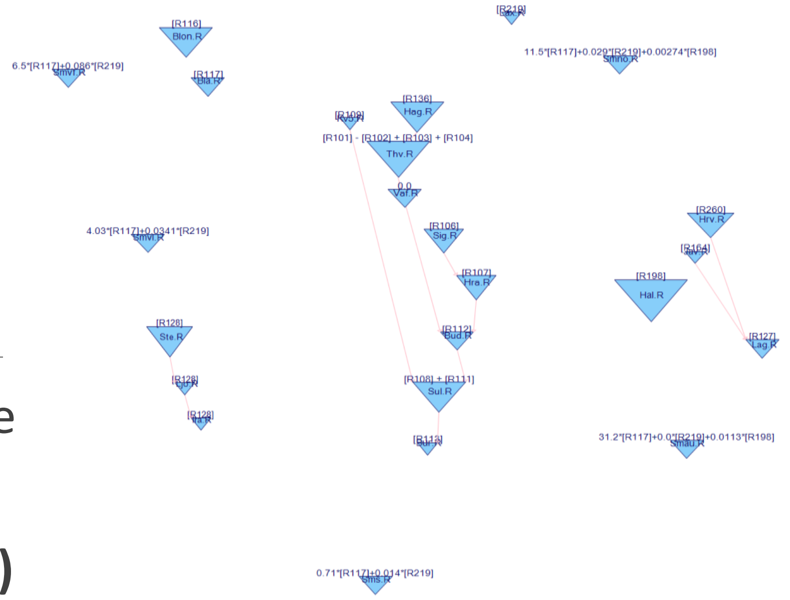
Method

Two tools have been developed during the last year and a half, a so called LP method (simplistic) and recently an SDDP algorithm.

These methods find the optimal operational decisions, given the historical inflows, wind scenarios and other inputs.

Neither of these methods aggregate reservoirs and the time resolution can be changed in both. However, computation time is increased...





Method (continued)

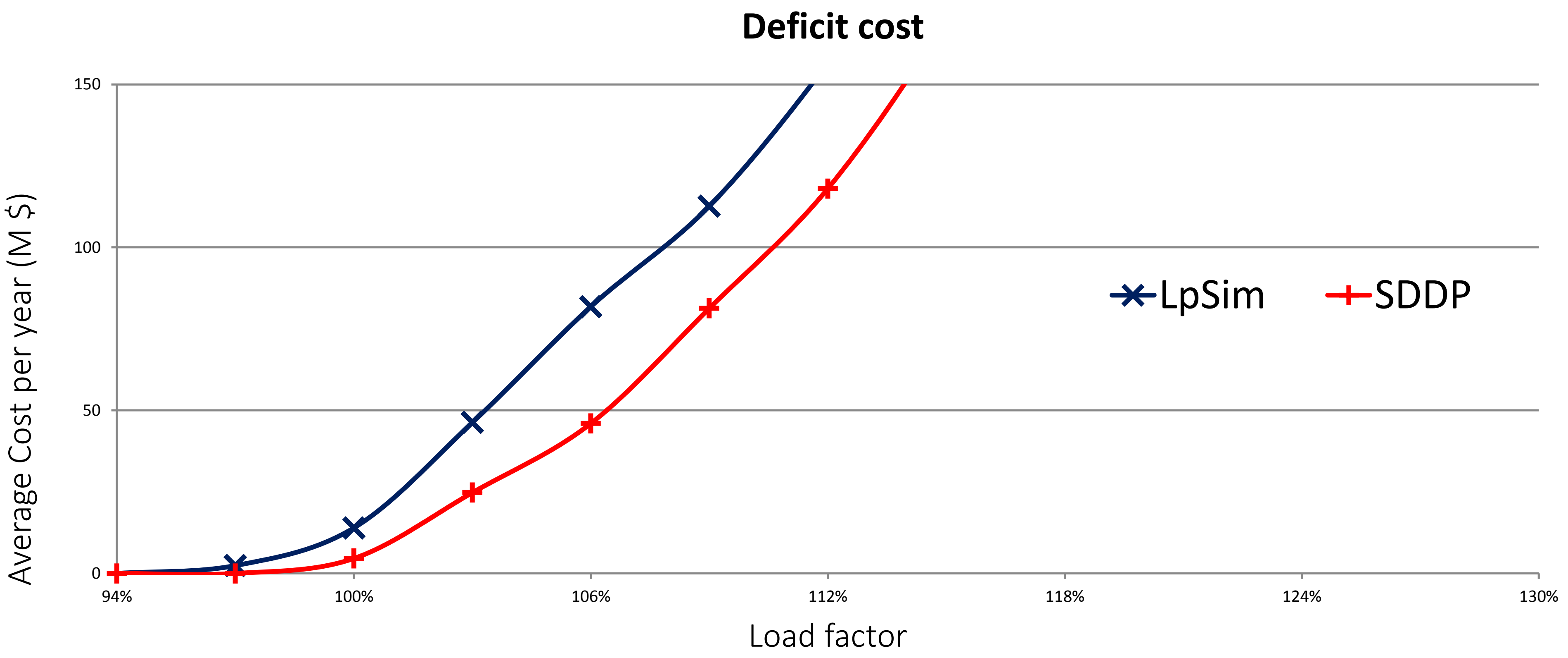
Without hydro power, a current operation decision does not affect the future to the same degree as when hydro power is present:

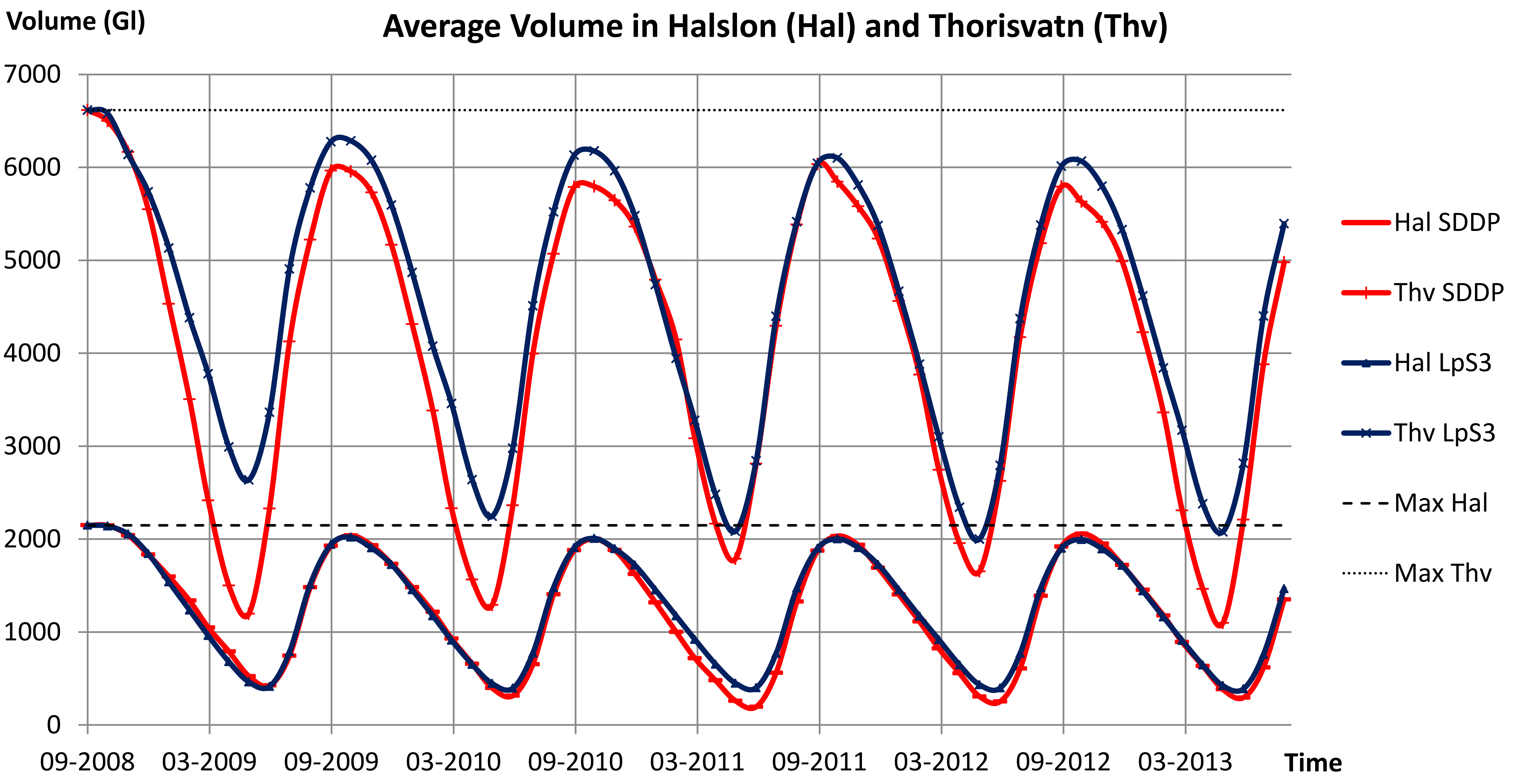
$$\textbf{Demand}_t = \textbf{Generated power}_t - \textbf{Losses}_t + \textbf{Curtailment}_t \quad \textbf{(for all buses)}$$

Iceland has a hydro dominated power system. Reservoirs states add unlimited possibilities to an unknown future in which one time depends on the next:

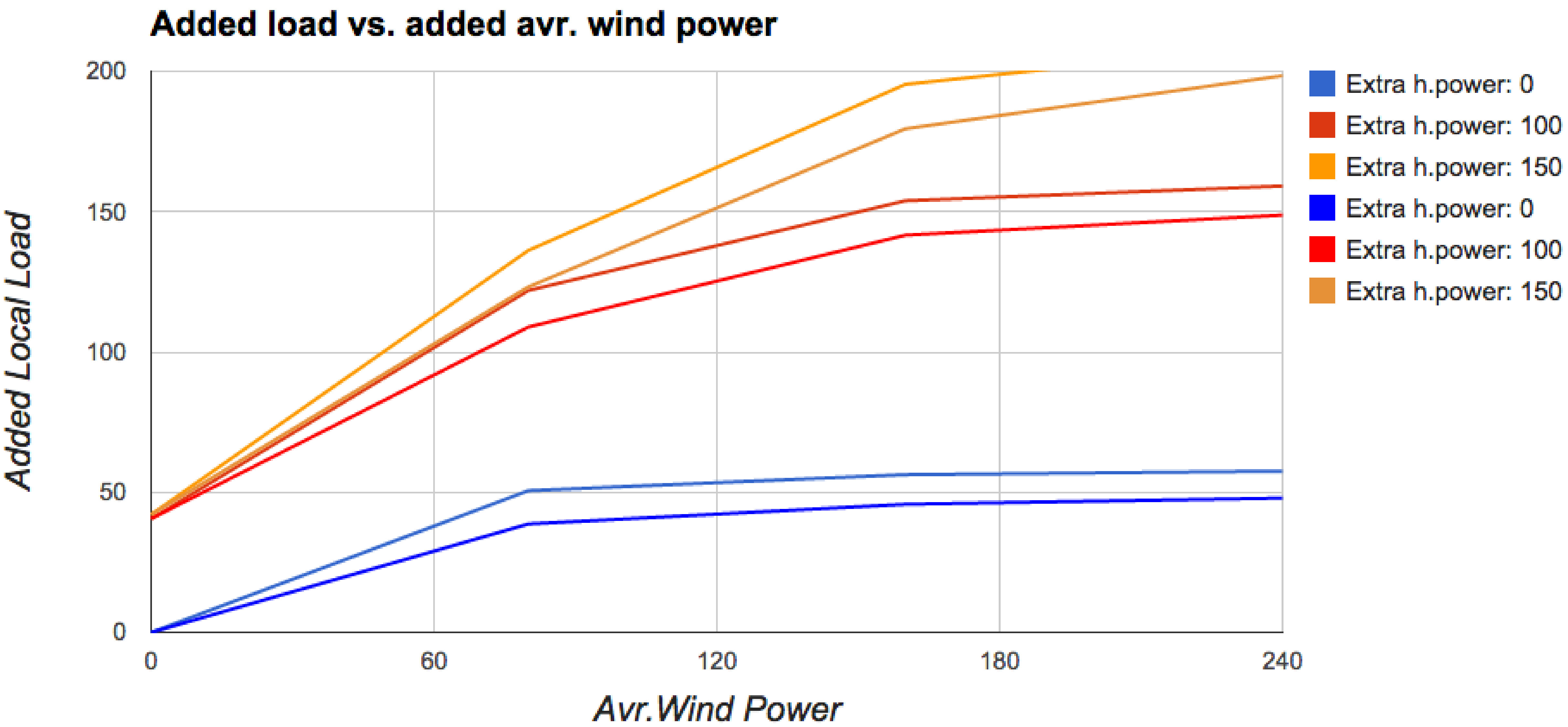
$$\textbf{Volume}_{t+1} = \textbf{Volume}_t + \textbf{inflow}_t - \textbf{use}_t - \textbf{spill}_t \quad \textbf{(for all reservoirs)}$$

Results, SDDP vs. LpSim





Average Volume in Halslon (Hal) and Thorisvatn (Thv)
Volume (Gl)
7000
6000
5000
4000
3000
2000
1000
0
09-2008
03-2009
09-2009
03-2010
09-2010
03-2011
09-2011
03-2012
09-2012
03-2013
Time
Hal SDDP
Thv SDDP
Hal LpS3
Thv LpS3
Max Hal
Max Thv


Added load vs. added avr. wind power
Added Local Load
200
150
100
50
0
0
60
120
180
240
Avr.Wind Power
Extra h.power: 0
Extra h.power: 100
Extra h.power: 150
Extra h.power: 0
Extra h.power: 100
Extra h.power: 150

Ahead

My funding runs out in the end of December. From that day on I will be working part time for Landsvirkjun, using same or similar models.

This week I’m sending in an extended abstract to the IEEE Transactions on Sustainable Energy.

This winter I will visit Sintef/NTNU, hopefully for some weeks, to start some research collaboration. A few experts in the SDDP methodology work there.

Hopefully Landsvirkjun will be interested in results from the new SDDP program.

Validation

In Trondheim, it would be nice to do a comparison with their running SDDP program. Otherwise, Landsvirkjun’s program could be compared against, although with limited features.

Thank you !

The need to get my hands dirty

Answer = Have you ever heard of a computational engineer?

if Answer == yes:

Have you ever heard of a computational engineer that never wrote code that computes?

Initially I wrote an all-knowing, god-like code (LP). Unrealistic since the future was known with 100% certainty.

Since October, a trained operator-like code has been in development that addresses the mentioned good qualities (previous slide). The methodology is called SDDP.

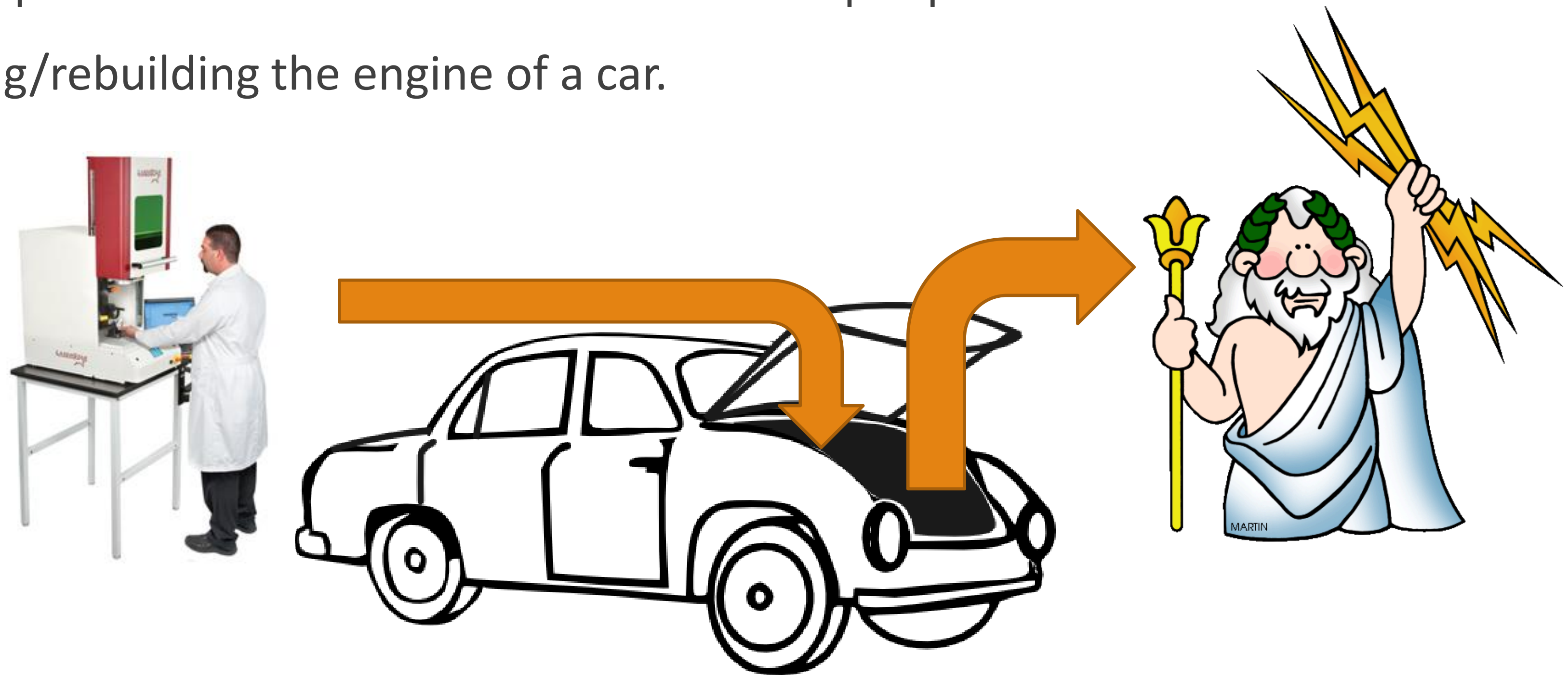
```
import sqlite3, math #csv,  codecs, shutil
import numpy as np
from datetime import date, timedelta
from multiprocessing import cpu_count
import sys
import os
#import time as timer
NUM_PROCS = cpu_count()


numyears = 1
numruns = 2 # MAX 55 - numyears if no carou
SysCase = "All"  # possibilities: All, thre
Dynamic = 1 # 1 if dynamic programming else
if Dynamic:
    psegments = 12*numruns
    poolsize = min(NUM_PROCS,numruns)
    hsamplnr = numruns  # How many hydro ye
    shortlistnumber = numruns #max(7,int(nu
#DualTolerance = 100
############ TIME ############
dH = [1,22,1]   # Load Hourly time steps, a
sumdH = sum(dH)
dT = int(sumdH/24)      # Reservoir number
yearlength = 364
```

SDDP: Not written from scratch

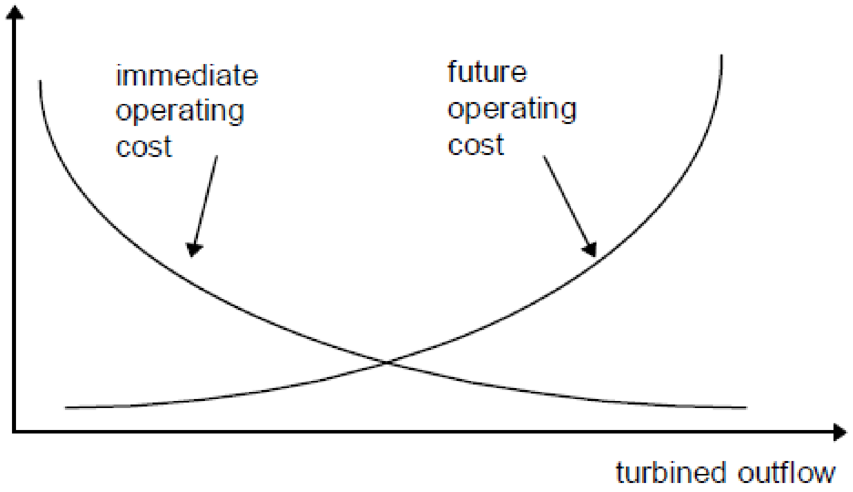
Uses the same optimization model and the same data input processor.

It is like changing/rebuilding the engine of a car.



Actually, like adding a second engine since the old engine is still alive!

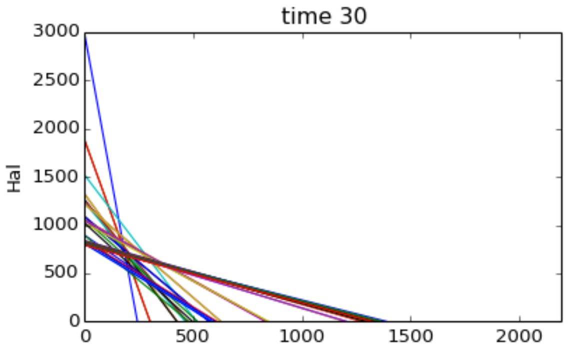
SDDP iterates between two phases until a desired accuracy is reached



Phase I, *simulation*:
The power system is run in weekly optimization steps, following a recorded inflow sequence.

Tangents to the fcf are added

Recorded volume states



Phase II, future cost estimation:
Given the volume states from Phase I, starting at time step T and going backwards, the shadow price of increased volume is calculated at every week and inflow, dc/dv in order to piecewise linearly approximate the future cost as a function of volume.

Features of both methods:

Time resolution can be easilly modified in steps of hours.

dH = [1,22,1]   dT = 1

Load modifier, average or by load duration curve

Wind modifier   (mean and st.dev conserved)

Spill binary variables

Minimum energy, minimum end volume and minimum load deposit.

Curtailment duration and uniform curtailment.

```
######### model tweaks #########
Highresprint = 1     # If 1, the
Spillbinary = 0 # if 1, then bin
SpillCost = Dynamic*0.001    # Co
q_cont = 0  # continuity of turb
prodlimit = 0 if Dynamic else 0*
#### Minimum volume or energy
v_end = -1 # Minimum energy in r
d_end = 0.95*v_end  # Used if v_
minEnergy = 0 if Dynamic else 0*
############## LOAD ############
Loadfactor = 1.1*0.993795    # No
Loadyear = 2016
loadcut = 2 # If 0 then coarse h
RepeatedLoad = 1  # If 1, first
DayStart = date(Loadyear,10,1)
LoadDBname = "HourLoad.db"  # di
load_sort_nodes = ['Hs']    # In
perm_node = 'Hs' # Used to set t
# Volume set
vin = 0.98     # Initial volume
volume_bite_tail = 0 # If the in
# Transmission
TransMult = 1   # Transmission l
TransLoss = [0,0,0] #[0.002,0.00
# Solver
warmstart = True
solvereport = False # solver giv
# Curtailment / deficit
```

LP *and* SDDP

The code is now at around 3000 lines of code.

Computation time for T time steps and n inflows:

LP methodology is $n*O(T)$

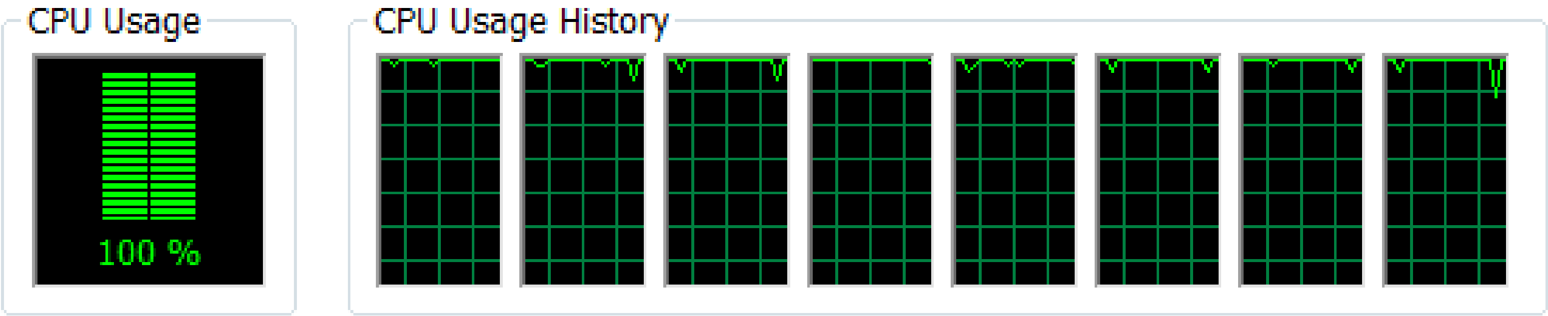
While SDDP is $k*T*O(n^2)$

-where k is the number of iterations

Phase I of SDDP gives an upper bound to the operation cost while phase II gives a lower bound.

SDDP upper		LP		SDDP lower
26 zsum.mean 1421.5	>	891.7	>	665.6

Parallelization



Given the load, inflows and wind during a period T the old LP „engine“ found the optimal operation of the power system by solving a single problem that contained all this information.

In contrast, the SDDP engine typically knows only the week ahead values of the scenario and solves many small week ahead LP problems.

Commercial solvers do a fairly good job in distributing a large problem to multiple cores of a CPU. Small problems, like those of SDDP, do not have the necessary size to harness many cores of a CPU.

Initially the SDDP algorithm ran at around 25% CPU usage. After parallelization it often goes up to 100%. The number of tasks is equal or close to the number of inflows and since inflows are in the tens, the problem has good parallelable options.

Preliminary results