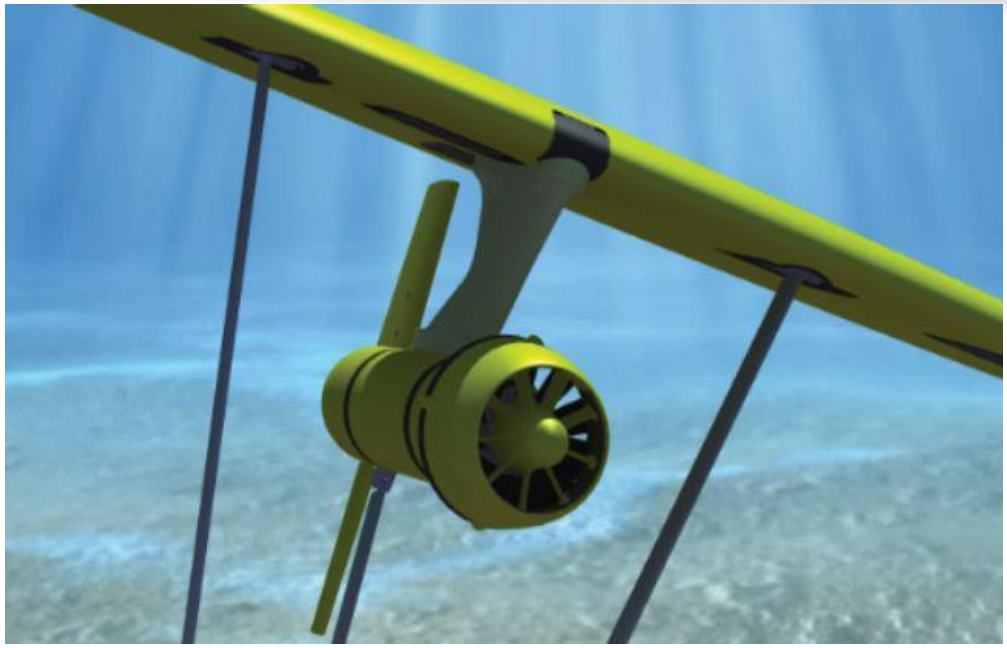
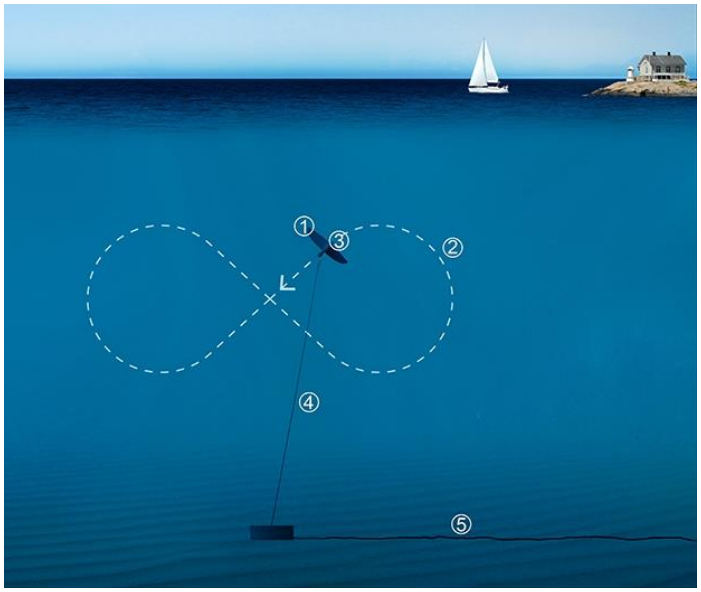
A parametric turbine using OpenFoam+cfMesh+scripts for the geometry

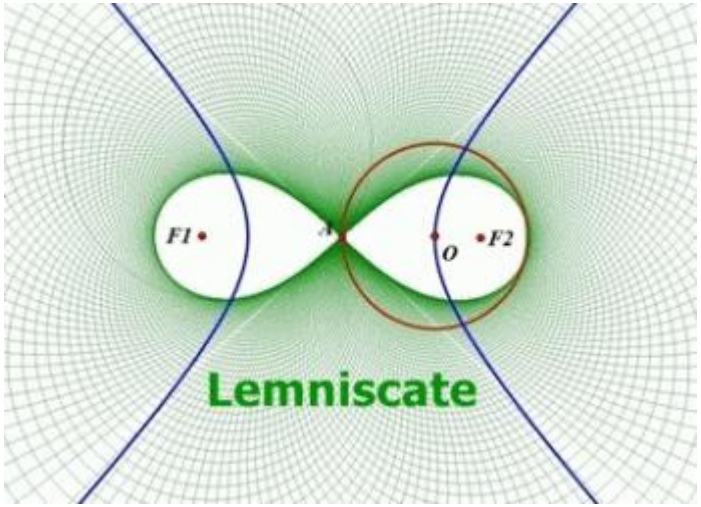
Minesto AB
Björn Bergqvist
Turbine Analyst

Minesto Tidal Energy





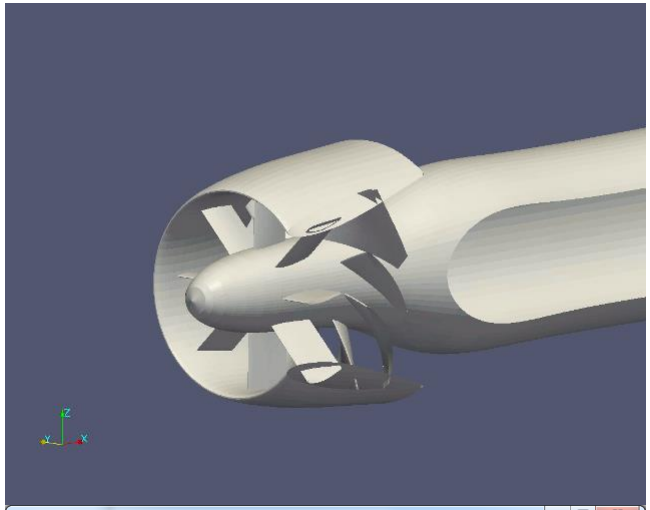
Minesto develops a new concept for tidal power plants called Deep Green. The power plant is applicable in areas where no other known technology can operate cost effectively due to its unique ability to operate in low velocities. Minesto expands the total marine energy potential and offers a step change in cost for tidal energy.



CFD simulation of ducted turbine

- A coarse model of the turbine has been made
  - Not for exact absolute values of cp and ct
  - Investigating trends
- Work procedure
  - Parametric geometry
    - Stl triangle mesh created with small script/program (Ruby script language used)
  - Automatic
    - Mesh (cfMesh)
    - Simulation (simpleFoam with modified solver+swak4Foam runtime function)
    - Evalutation (shell script)
- Future work
  - Connect to suitable optimization algorithm

Scripted Geometry



Turbine geometry created from script
Stl-file is created

← animation

turbineblade.rb (C:\Local\parame...ne\viewdir\ducted_geom001) - GVIM4

Arkiv Redigera Verktyg Syntax Buffertar Fönster Hjälp

```
require 'spliner'
#load 'equationnaca.rb'
#def naca65_008
#def naca65_010
#def naca63A008

x,y,dydx= naca65_010
x.pop
#x.pop
y.pop
#y.pop
pitch1=ARGV[0].to_f
camber1=ARGV[0].to_f

tab_len=100

rtab=[41,78.5,119] #First value has to match rtab1:s first value
ttab=[0.9,0.9,0.7]

pitchtab=[-43.5*pitch1,-55.5*pitch1,-74.7*pitch1]
cordatab=[87,88,89]
cambertab=[0.06*camber1,0.056*camber1,0.048*camber1]
rotx=[2,2,2]
roty=[0,0,0]
extrarottab=[-1,2.5,7.5]

xtab=[0,60]
rtab1=[41,51]
#rtab2=[103.2,118.8]
rtab2=[114.8,114.8]

xtop=Array.new
ytop=Array.new
ztop=Array.new
xbot=Array.new
ybot=Array.new
zbot=Array.new
```

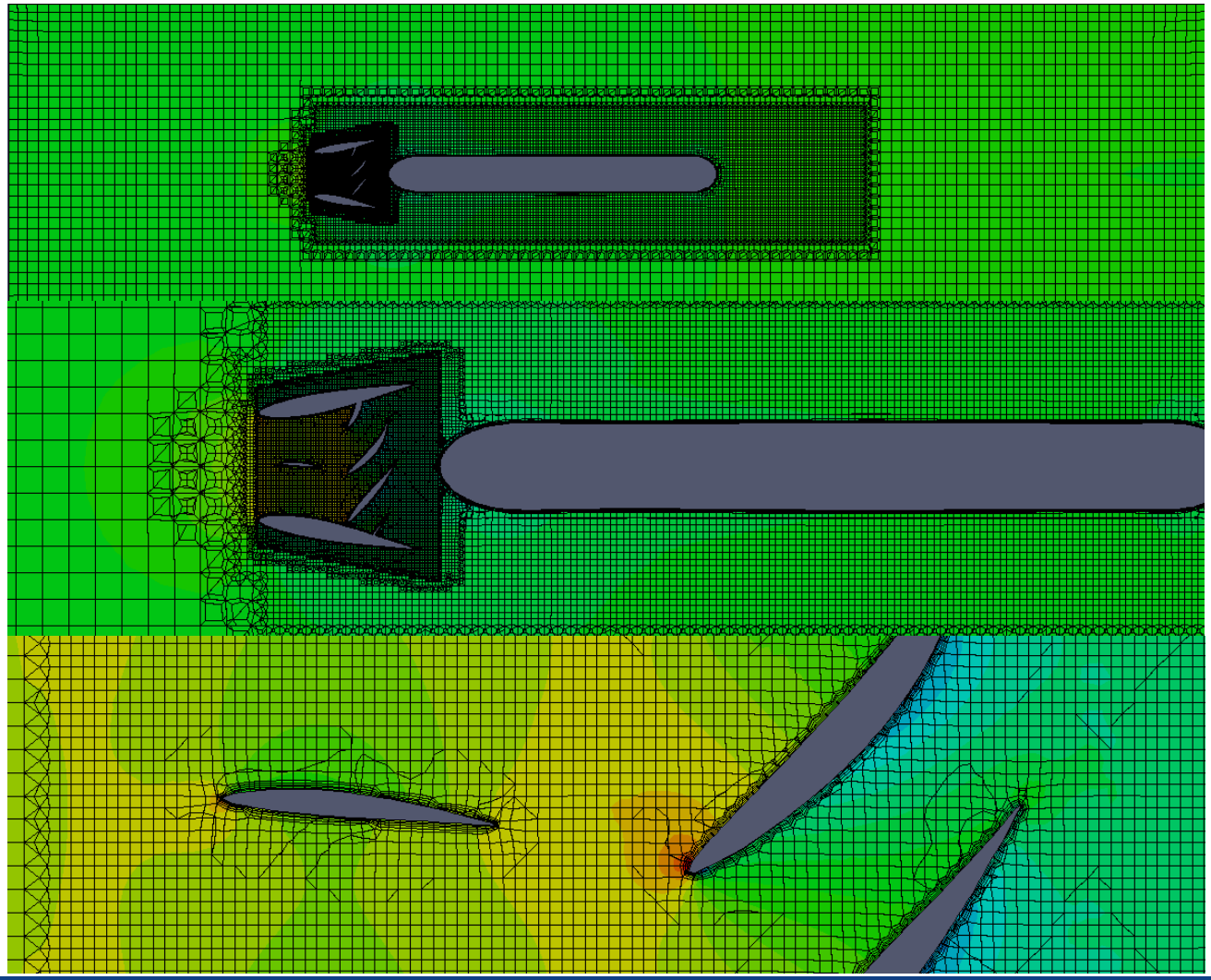
39,1 4%

This approach was easier than trying to do it in OpenSCAD. Creating STL-triangles are not that hard after all…

```
def stlfacet(v0,v1,v2)
   mindist=1e-5
   vec0=[v1[0]-v0[0],v1[1]-v0[1],v1[2]-v0[2]]
   vec1=[v2[0]-v0[0],v2[1]-v0[1],v2[2]-v0[2]]
   crossprod=crossNormal(vec0,vec1)
   returnstring="  facet normal #{crossprod[0]} #{crossprod[1]} #{crossprod[2]}\n"
   returnstring+="    outer loop\n"
   returnstring+="      vertex #{v0[0]} #{v0[1]} #{v0[2]}\n"
   returnstring+="      vertex #{v1[0]} #{v1[1]} #{v1[2]}\n"
   returnstring+="      vertex #{v2[0]} #{v2[1]} #{v2[2]}\n"
   returnstring+="    endloop\n"
   returnstring+="  endfacet\n"
   if checkpoints(v0,v1,v2,mindist)
      return returnstring
   else
      return ""
   end
end
def stlquad(v0,v1,v2,v3)
   returnstring=stlfacet(v0,v1,v2)
   returnstring+=stlfacet(v2,v3,v0)
   return returnstring
end
```

Volume mesh

- cfMesh, only refinement volumes



MRF region selected with toposet

```
actions
(
    {
        name    TURBINE;
        type    cellSet;
        action  new;
        source surfaceToCell;
        sourceInfo
        {
          file             "geom/sliding_zon
          outsidePoints    ((-0.3 0 0));
          includeCut       false;
          includeInside    true;
surf
          includeOutside   false;
surf
          nearDistance     -1;
surf

          curvature        0.9;
nearDistance
        }
    }
);
```

Solution strategy

- Iteration 0-2000 → converge max speed
- Iteration 2000-2050 → ramp down speed
- Iteration 2050-2200→ converge next speed
- Iteration 2200-2250→ ramp down speed
- …
- …

- This is only possible if MRF is reinitialized while ramping speed→ add mesh.update() in the simple.loop() in simpleFoam.C

fvOptions

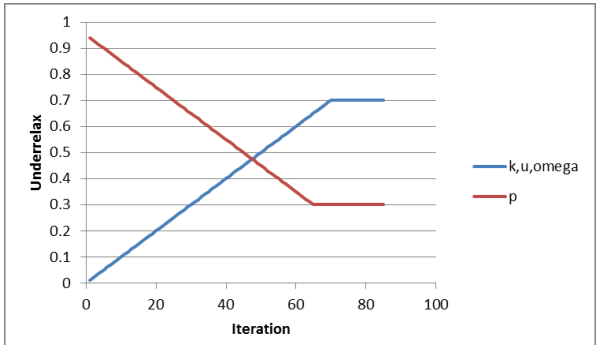
```
MRF1
{
    type            MRFSource;
    active          true;
    selectionMode   cellZone;
    cellZone        innerCylinderSmall;
    nonRotatingPatches (duct);

    MRFSourceCoeffs
    {
        origin      (0 0 0);
        axis        (1 0 0);
        omega       table
        (
            (0    -10)
            (100  -260)
            (2000 -260)
            (2050 -230)
            (2200 -230)
            (2250 -200)
            (2400 -200)
            (2450 -170)
            (2600 -170)
            (2650 -140)
            (2800 -140)
            (2850 -110)
            (3000 -110)
            (3050 -80)
            (3200 -80)
        );
    }
}
```

Some underrelax functions using swak4foam

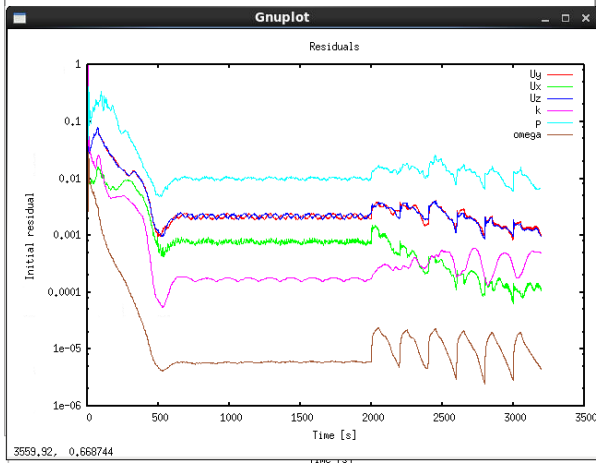
```
//functions (
    #include "readFields"
    #include "Q"
    //#include "surfaces"
    #include "forces"
//)
//functions (
    adaptRelaxation
    {
        type pythonIntegration;
        startCode
            #{
from os import path
from PyFoam.RunDictionary.ParsedParameterFile import ParsedParameterFile
from math import log,exp,pow
control=ParsedParameterFile(path.join(caseDir,"system","controlDict"))

end=float(control["endTime"])

del control

#scaleTill=int(end*0.5)
scaleTill=10000;
scaleFactor=10.

factor=exp(log(scaleFactor)/scaleTill)

print "Scale by actor:",factor
print

fvSol=ParsedParameterFile(path.join(caseDir,"system","fvSolution"),backup=True)
            #};
        executeCode
            #{
if runTime<=scaleTill:
    for v in fvSol["relaxationFactors"]:
        if v == 'p':
          if fvSol["relaxationFactors"][v] > 0.3:
            fvSol["relaxationFactors"][v] -= 0.01
            print v," underrelax now: ",fvSol["relaxationFactors"][v]
            print
        else:
          if fvSol["relaxationFactors"][v] < 0.7:
            fvSol["relaxationFactors"][v] += 0.01
            print v," underrelax now: ",fvSol["relaxationFactors"][v]
            print

    fvSol.writeFile()

print
            #};
        endCode
            #{
print "Restoring fvSolution to its old glory"
fvSol.restore()

print
            #};
        parallelMasterOnly true;  // In parallel runs only the master needs to execute this
    }
```

Initial values

```
relaxationFactors
{
p               0.95;
U               0.001;
k               0.001;
epsilon         0.001;
}
```



Ramping of underrelax



Residuals from a run

Input output

Input:

DOE_run1.sh (S:\bjorn\methods\MRF\parametric_turbine\rundir) - GVIM5

Arkiv Redigera Verktyg Syntax Buffertar Fönster Hjälp

```
#!/bin/sh
sbatch -c 60 ./runturbineOmega run1 135 -0.0075 195 10 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0075 195 11 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0075 195 9 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0075 195 12 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0075 195 8.5 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0095 195 10 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0095 195 11 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0095 195 9 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0095 195 12 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.0095 195 8.5 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.003 195 10 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.003 195 11 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.003 195 9 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.003 195 12 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.003 195 8.5 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.012 195 10 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.012 195 11 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.012 195 9 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.012 195 12 1 1 1 1
sbatch -c 60 ./runturbineOmega run1 135 -0.012 195 8.5 1 1 1 1
~
~
~
<tric_turbine\rundir\DOE_run1.sh" [unix] 21L, 1260C          1,1          Alla
```

Simple to run many designs.

Output:

optimization_20150601.txt (C:\Local) - GVIM

Arkiv Redigera Verktyg Syntax Buffertar Fönster Hjälp

```
xpoint,ductcamber,ductcorda,ductangle,statorpitch,statorcamber,turbinepitch,turbinecamber,cpctmax,cp,ct,omega_at_max,rpm_at_max,tpr_at_max
140,-0.008,205,8,0.7,0.9,0.95,1.1,0.55448024480633,0.213084527752928,0.384295977627399,110.0,1050.42262440651,1.43
140,-0.009,205,8,0.7,0.9,0.95,1,0.55464316821963,0.214182652409887,0.386162968701841,110.0,1050.42262440651,1.43
140,-0.008,195,8,0.7,0.9,0.95,1,0.57019639014668,0.213132731170639,0.373788285760904,110.0,1050.42262440651,1.43
130,-0.009,195,8,0.7,0.9,0.95,0.9,0.571369451938732,0.212246659286082,0.371470085714407,110.0,1050.42262440651,1.43
135,-0.009,195,8,0.7,1,0.95,0.9,0.554051631976956,0.208482300956602,0.376286773513688,110.0,1050.42262440651,1.43
135,-0.008,195,8,0.7,0.9,0.95,0.9,0.57[illegible]480272298071,0.212671274503402,0.37214106035226,110.0,1050.42262440651,1.43
135,-0.009,195,8,0.7,0.9,0.95,1.1,0.569978022037639,0.213713529936516,0.374950474708661,110.0,1050.42262440651,1.43
130,-0.008,195,8,0.7,0.9,0.95,1.1,0.570445784308994,0.210602281714999,0.369188952759307,110.0,1050.42262440651,1.43
130,-0.008,205,8,0.7,0.9,0.95,0.9,0.557457704649813,0.209564530460949,0.375929023337465,110.0,1050.42262440651,1.43
135,-0.008,195,8,0.7,0.9,0.95,1,0.571211815579l,0.211635276025566,0.37050227298083S,110.0,1050.42262440651,1.43
140,-0.009,205,8,0.7,0.9,0.95,0.9,0.554182247313412,0.213595451659968,0.385424565105513,110.0,1050.42262440651,1.43
                                                                                                                    7,39          Topp
```

Evaluation

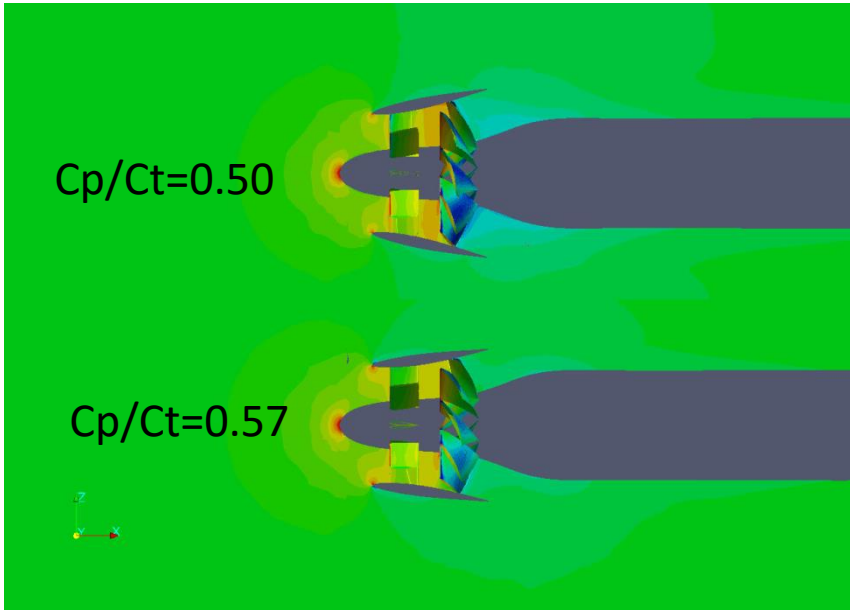
Basic result extraction (shell script)

```
#!/bin/sh
run=$1
#duct  nacelle  stator  totalforce  turbine
turbine="$run/postProcessing/turbine/0/forces.dat"
stator="$run/postProcessing/stator/0/forces.dat"
total="$run/postProcessing/totalforce/0/forces.dat"


#file=$body
#file=$1
startcollect=1990
startomega=260
echo "omega,M_turb_press,M_turb_visc,M_stat_press,M_stat_visc,F_tot_press,F_tot_visc"
while [ $startcollect -lt 3200 ]; do
   M1_PRESS=`cat $turbine |tr -d '()'|awk 'NR>'$startcollect'{sum+=$11;count+=1;print sum/count}' |head -n 10|tail -n 1`
   M1_VISC=`cat $turbine  |tr -d '()'|awk 'NR>'$startcollect'{sum+=$14;count+=1;print sum/count}' |head -n 10|tail -n 1`
   M2_PRESS=`cat $stator |tr -d '()'|awk 'NR>'$startcollect'{sum+=$11;count+=1;print sum/count}' |head -n 10|tail -n 1`
   M2_VISC=`cat $stator  |tr -d '()'|awk 'NR>'$startcollect'{sum+=$14;count+=1;print sum/count}' |head -n 10|tail -n 1`
   FX_PRESS=`cat $total |tr -d '()'|awk 'NR>'$startcollect'{sum+=$2;count+=1;print sum/count}' |head -n 10|tail -n 1`
   FX_VISC=`cat $total |tr -d '()'|awk 'NR>'$startcollect'{sum+=$5;count+=1;print sum/count}' |head -n 10|tail -n 1`
   echo "$startomega,$M1_PRESS,$M1_VISC,$M2_PRESS,$M2_VISC,$FX_PRESS,$FX_VISC"
   let startcollect=startcollect+200
   let startomega=startomega-30
done
~
~
```

Find Cp/Ct-max (Ruby script)

```
    f_tot_v = []
    File.open("output.txt") do |handle|
       handle.each_with_index do |line,idx|
          if idx == 0 then
             result = line.chomp.split("_")
             param << result[1]
             param << result[2]
             param << result[3]
             param << result[4]
             param << result[5]
             param << result[6]
             param << result[7]
             param << result[8]
          elsif idx ==  1 then
             #header
          else
             result = line.split(",")
             omega << result[0].to_f
             m_turb_p << result[1].to_f
             m_turb_v << result[2].to_f
             m_stat_p << result[3].to_f
             m_stat_v << result[4].to_f
             f_tot_p << result[5].to_f
             f_tot_v << result[6].to_f
          end

       end
    end

    res=[]
    (0..omega.size-1).each do |idx|
       res[idx] = cp(m_turb_p[idx]+m_turb_v[idx],omega[idx],incoming_energy)/ct(f_tot_p[idx]+f_tot
    end
    #puts res.join(" ")
    maxidx = res.find_index(res.max)
    cp_at_max=cp(m_turb_p[maxidx]+m_turb_v[maxidx],omega[maxidx],incoming_energy)
    ct_at_max=ct(f_tot_p[maxidx]+f_tot_v[maxidx],staticpressure)
    if res[maxidx] <= -2.0 || res[maxidx] >= 2.0 || cp_at_max > 1.0 || ct_at_max > 1.0 || cp_at_ma
       failed_sim.push("simulation #{file} failed")
    else
       print param.join(",")
       print ","
       print res[maxidx]
       print ","
       print cp_at_max
       print ","
       print ct_at_max
       print ","
       print omega[maxidx]
       print ","
       print omega[maxidx]*30.0/Math::PI
       print ","
       print radius*omega[maxidx]/inflow_vel
       print "\n"
    end
    rescue
       failed_sim.push("simulation #{file} failed")
    end

end
```

Automated CFD



Entire analysis automated via script.
Desired parameters are chosen and then CFD could be run from one script.
The output is processed with a script as well.

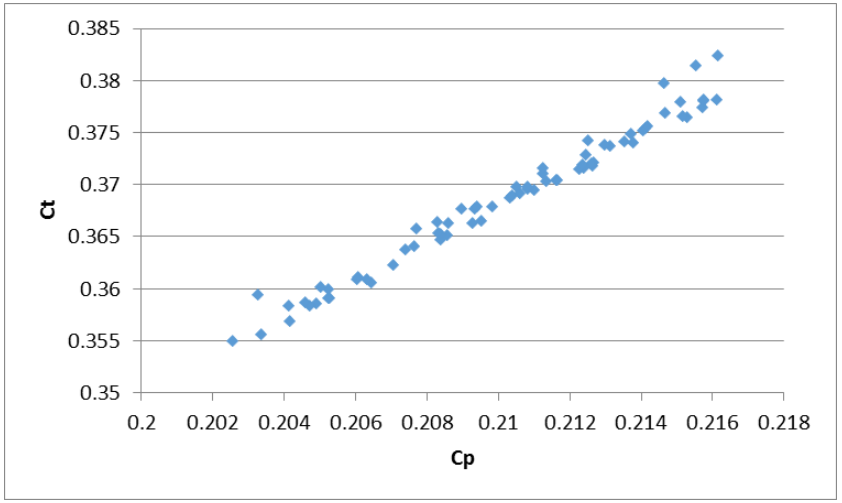
Best and worst design so far:
The turbine on top has the worst Cp/Ct-ratio and the other the best so far

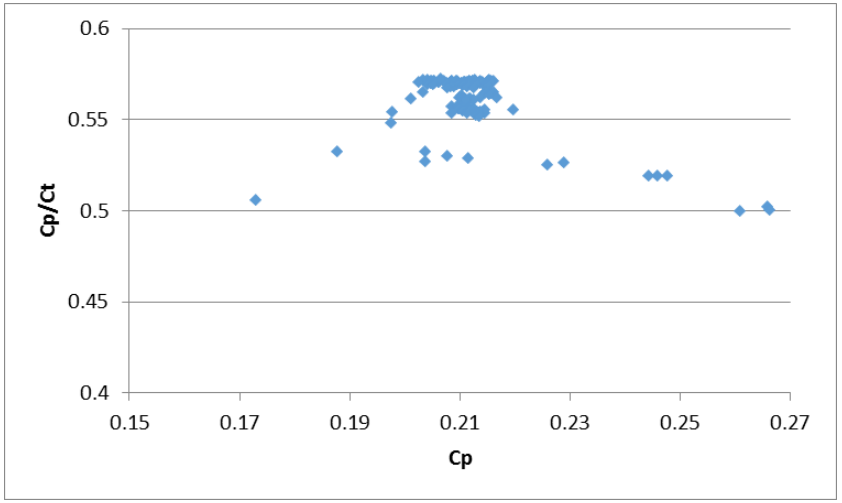
Output

From SSPA measurements

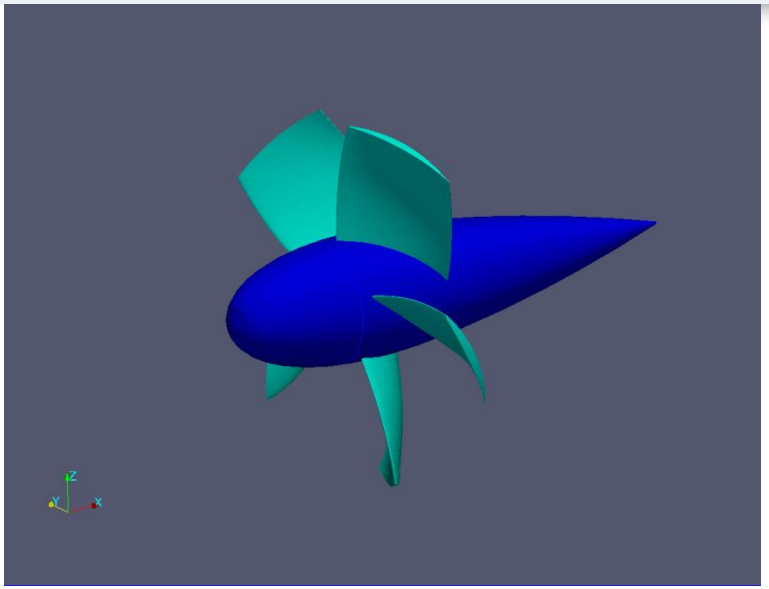
TPR	cp (Eq)	ct (Eq)	cp/ct (Eq)
1	0.130	0.446	0.291
1.5	0.214	0.483	0.443
1.7	0.230	0.482	0.478
2	0.235	0.471	0.499
2.5	0.192	0.439	0.438
3	0.086	0.393	0.218
3.5	-0.085	0.310	-0.273

From DOE

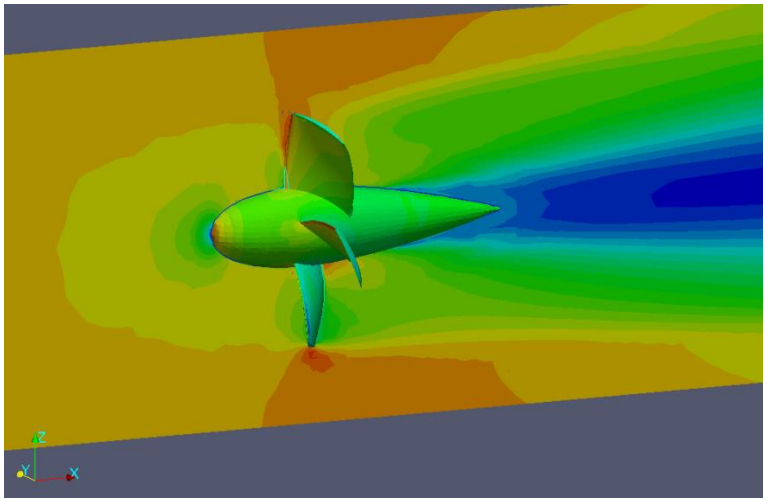




A case is prepared for you



Parametric Geometry (variable pitch)



Coarse mesh will give you results in minutes on a laptop

Will be available through http://www.discretizer.se
Follow me at Twitter @discretizer

Author to this method is Bjorn Bergqvist. Please refer to me or communicate if you use the turbine geometry model in your work.

Prerequisites:
OpenFOAM 2.?.? (used by 3_run_simulation.sh)
cfMesh (used by 2_create_volume_mesh.sh)
OpenSCAD (used by 2_create_volume_mesh.sh, tunnelfactor 1 is already generated)
Ruby 2 with spliner gem (for splined pitch, otherwise any Ruby will do)
  (install with: "gem install spliner")

Description of files and directories:

Allrun
  Run through script 1_ to 3_

1_create_geometry.sh
  Not necessary to run. Turbine geometry with pitchmod=0 is
  already generated in geom/turbine.stl.
  Usage: ./1_create_geometry <pitchmod>
  were <pitchmod> is modification of turbine blade pich
  if you are unsure: use pitchmod=0

2_create_volume_mesh.sh
  Usage: ./1_create_volume_mesh.sh <tunnelfactor>
  where <tunnelfactor> is the size of the tunnel where the turbine is simulated
  if you are unsure: use tunnelfactor=1

3_run_simulation.sh
  Usage: ./3_run_simulation.sh <tunnelfactor>
  will run on 4 cpus
  (change in decomposeParDict and 3_run_simulation.sh for other cpu count)

post_moment_table.sh
  Averages moments and forces from 990 to 1000 iterations.

Allclean
  Removes generated mesh and simulation.

turbine_geometry
  The Bjorn Bergqvist(C) turbine geometry.

cfMesh.template
  Directory with files used by cfMesh.

simpleFoam.template
  Directory for simulation (steady state, MRF)

I used this case to look at…

…our turbine supplier geometry, just had to modify MRF-region, mesh refinement cylinders, rotational speed etc