Cite as: Capitao Patrao, A.: Description and validation of the rotorDiskSource class for propeller performance estimation. In Proceedings of CFD with OpenSource Software, 2017, Edited by Nilsson. H.,
http://dx.doi.org/10.17196/OS_CFD#YEAR_2017

CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Håkan Nilsson

Description and validation of the rotorDiskSource class for propeller performance estimation

Developed for OpenFOAM-v1706

Author:
Alexandre Capitao Patrao
Chalmers University of Technology
capitao@chalmers.se
alexandre.capitao.patrao@gmail.com

Peer reviewed by:
Madhavan Vasudevan
Mohammad Hossein

Disclaimer: This is a student project work, done as part of a course where OpenFOAM and some other OpenSource software are introduced to the students. Any reader should be aware that it might not be free of errors. Still, it might be useful for someone who would like learn some details similar to the ones presented in the report and in the accompanying files. The material has gone through a review process. The role of the reviewer is to go through the tutorial and make sure that it works, that it is possible to follow, and to some extent correct the writing. The reviewer has no responsibility for the contents.

January 5, 2018

Learning outcomes

The reader will learn:

How to use it:

- How to setup a simplified simulation of the flow around a propeller using the `rotorDiskSource` class and the `simpleFoam` solver.
- How to define the propeller geometry in the correct way.
- How to setup a mesh in the correct way for best results.
- How to setup a case in the compressible flow solver `rhoSimpleFoam` which includes the `rotorDiskSource` functionality.

The theory of it:

- A brief description of the theory behind the actuator disk will be provided.
- The Blade element theory will be reviewed.

How it is implemented:

- Description of how the `rotorDiskSource` class is implemented in OpenFOAM.

How to modify it:

- It will be shown how to fix a critical bug in the `rotorDiskSource` class.
- It will be shown how to implement an improved tip correction factor and various propeller performance parameters.

Contents

Chapter 1

Introduction

1.1 Background

- Propeller performance prediction methods range in order of complexity and fidelity from analytical 1D momentum theory to the Blade Element-Momentum Theory (BEM), lifting line method, all the way up to 3D CFD.
- A method which in terms of complexity and computational effort lies between the lifting line and 3D CFD methods is the actuator disk method. This method simulates the time-averaged flow over a propeller by adding the blade forces as source terms in the momentum equations, and should therefore simulate the wake development and performance more accurately than the BEM and faster than 3D CFD.
- The capability to design/analyze propellers designs using BEM and 3D CFD already exists in-house. Obtaining an analysis method able to predict performance more accurately than the BEM and faster than CFD would allow the propeller designer to perform a check on the design on an intermediate level of complexity at a reasonable computational cost.
- The reduced computational cost when using an actuator disk versus full 3D CFD is of even greater importance when studying rotor/airframe interaction. Addiotionally, the actuator disk functionality could be of great use for optimizing open rotor nacelle geometries, since it greatly decreases the computational cost relative to simulations including the full blade geometries.
- The methodology being investigated in this report is part of the `rotorDiskSource` class, and is included in the OpenFOAM v1706 release. The original code was seemingly developed by Wahono [1] and has since then been incorporated into the standard OpenFOAM release.
- The accompanying files contain the final modified `rotorDiskSource` class and case files.

Chapter 2

Theory

2.1 Actuator Disk Theory

The fundamental philosophy of the actuator disk methodology is to replace the physical geometry of a rotor with only its effect on the flow. This is done by inserting the rotor forces as source terms S_i into the momentum equation of the governing equations (in this case for incompressible flow).

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial u_j u_i}{\partial x_j} - \frac{\partial R_{ij}}{\partial x_j} = -\frac{\partial p}{\partial x_i} + S_i \tag{2.2}$$

The term R_{ij} is the stress tensor and includes the turbulent Reynolds stresses in case the solver supports turbulent flow. The source terms S_i are only added to the governing equations for the cells in the mesh that are located inside a pre-defined cylindrical actuator disk volume. This approach is obviously computationally cheaper than simulating an entire rotor blade geometry, but the resulting rotor wake will only capture the time-averaged effects of the entire rotor on the flowfield. Additionally, this approach cannot account for flow separation in the blades or other 3D effects such as shocks (for compressible cases), tip vortices, or hub horseshoe vortices.

2.2 Blade Element Theory

In order to calculate which forces to use as input for the governing equations, the Blade Element Theory (BET) is used. According to the BET, one can divide a rotor into a discrete number of two-dimensional sections, on which one can calculate the blade sectional thrust dT and torque dQ given the local velocity W, flow angle ϕ, and blade section properties such as blade angle (β), chord (c), and tables of lift (c_l) and drag (c_d) coefficients. The flow angle is obtained from the axial and tangential velocity components from the rotor cylindrical coordinate system (r, θ, z) with its z-axis parallel with the rotor axial direction. The BET also ignores any velocity contribution in the radial direction.

$$\phi = atan(W_z/W_\theta) \tag{2.3}$$

$$\alpha = \beta - \phi \tag{2.4}$$

The blade lift and drag coefficients for each section airfoil are functions of the angle-of-attack α, and are interpolated from lookup tables.

$$c_l = f(\alpha) \tag{2.5}$$

$$c_d = f(\alpha) \tag{2.6}$$

The blade sectional forces f_z and f_θ in the axial and tangential direction of the rotor can then be calculated.

$$f_z = \frac{1}{2}\rho W^2 c\left(Fc_l cos\phi - c_d sin\phi\right) \tag{2.7}$$

$$f_\theta = \frac{1}{2}\rho W^2 c\left(Fc_l sin\phi + c_d cos\phi\right) \tag{2.8}$$

The tip factor F accounts for the decreased lift in sections close to the blade tip due to the presence of a tip vortex. The presence of a tip vortex decreases the level of attainable lift near the blade tip but does not significantly affect drag. In the present OpenFOAM implementation, this is a step function with the value 1 from the hub up to a certain reference radius, and is 0 from this radius until the tip. A more widely used alternative for propellers is to use the tip correction suggested by Prandtl and improved upon by Drela [2]:

$$F = \frac{2}{\pi}\, cos(exp(-f)) \tag{2.9}$$

$$f = \frac{B}{2}\left(1 - \frac{r}{R}\right)\left(\frac{1}{\lambda}\right) \tag{2.10}$$

$$\lambda = \frac{r}{R}\, tan\,\phi \tag{2.11}$$

This tip correction will later be implemented into the rotorDiskSource class. For calculating the overall propeller performance in the BET, one needs to integrate Eq. 2.7 and 2.8 in order to obtain the total amount of generated thrust T and torque Q. These are then used to calculate power P and efficiency η for an entire rotor blade using the rotational velocity ω and flight velocity V_0.

$$dT = f_z dr \rightarrow T = \int_{r_{hub}}^{r_{tip}} f_z dr \tag{2.12}$$

$$dQ = f_\theta r dr \rightarrow Q = \int_{r_{hub}}^{r_{tip}} f_\theta r\, dr \tag{2.13}$$

$$P = Q\omega \tag{2.14}$$

$$\eta = \frac{TV_0}{P} \tag{2.15}$$

The forces calculated above are per blade section, and they need to be translated into volume forces acting on mesh cells residing inside the pre-defined cylindrical actuator disk volume. This also means that the forces need to be transformed from the rotor cylindrical coordinate system to the cartesian coordinate system used by the solver. The `rotorDiskSource` class is designed to work on hexahedral cells, and the sides of these cells need to be parallel with the radial direction of the rotor cylindrical coordinate system, as shown in Figure 2.1. Following this, the volume force for each cell can be calculated by using a scaling factor that *spreads out* the thrust created by all blades (B is the number of blades) over the entire circumference:

$$F_{i,cell} = Bf_i dr\frac{rd\theta}{2\pi r} = \{rd\theta dr = dA\} = \frac{B}{2\pi}\frac{A_{cell}}{r_{cell}} f_i \tag{2.16}$$

The cell centres inside the pre-defined cylindrical actuator disk volume are used as the locations (r_{cell}) for evaluting the blade sectional forces.

$$F_{z,cell} = \frac{B}{2\pi}\frac{A_{cell}}{r_{cell}} f_z(r_{cell}) \tag{2.17}$$

$$F_{\theta,cell} = \frac{B}{2\pi}\frac{A_{cell}}{r_{cell}} f_\theta(r_{cell}) \tag{2.18}$$

The overall rotor thrust, torque, power, and efficiency are calculated by summing the contributions from each cell inside the actuator disk volume.

$$T = \sum_{k=1}^{nCells} F_{z,cell,k} \tag{2.19}$$

$$Q = \sum_{k=1}^{nCells} F_{\theta,cell,k} r_{cell,k} \tag{2.20}$$

$$P = Q\omega \qquad \eta = \frac{TV_0}{P} \tag{2.21}$$

The source terms S_i of the momentum equation (Eq. 2.2) are calculated by transforming the cell forces into the cartesian coordinate system and then dividing by the cell volume.

$$S_{i,cell} = \frac{F_{i,cell}}{V_{cell}} \tag{2.22}$$

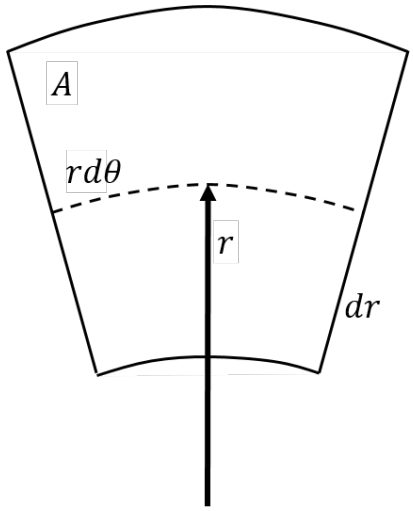


Figure 2.1: Hexahedral cell with its edges parallel with the radial and tangential directions of the rotor cylindrical coordinate system.

Chapter 3

The rotorDiskSource class

3.1 Implementation

The entire `rotorDiskSource` class source code can be found by typing the following commands:

```
OF1706+
cd $FOAM_SRC/fvOptions/sources/derived/rotorDiskSource
```

The directory contains the following folders and files:

```
rotorDiskSource
├── bladeModel
│   ├── bladeModel.C
│   └── bladeModel.H
├── profileModel
│   ├── lookup
│   │   ├── lookupProfile.C
│   │   └── lookupProfile.H
│   ├── profileModel.C
│   ├── profileModel.H
│   ├── profileModelList.C
│   ├── profileModelList.H
│   └── series
│       ├── seriesProfile.C
│       └── seriesProfile.H
├── rotorDiskSource.C
├── rotorDiskSource.H
├── rotorDiskSourceI.H
├── rotorDiskSourceTemplates.C
└── trimModel
    ├── fixed
    │   ├── fixedTrim.C
    │   └── fixedTrim.H
    ├── targetCoeff
    │   ├── targetCoeffTrim.C
    │   └── targetCoeffTrim.H
    └── trimModel
        ├── trimModel.C
        ├── trimModel.H
        └── trimModelNew.C
```

The `rotorDiskSource` class is activated at runtime if the solver finds a `fvOptions` file containing a `rotorDisk` type entry (the specific details of this file will be mentioned later on in this report).

A few selected files from the `rotorDiskSource` class will be briefly described below, but the main focus of this chapter is to show how the source terms representing the rotor forces are calculated, and this is done in the file `rotorDiskSourceTemplates.C` file.

- `rotorDiskSource.H` - Main class declaration file, which also shows that this class inherits from the abstract base class `cellSetOption`, which in turn inherits from the `fvOption` class. In essence, this means that the `rotorDiskSource` class adds source options to the momentum equation. Addtionally, this file declares important rotor properties such as number of blades, rotational velocity, and constructor functions.
- `rotorDiskSource.C` - This file contains the definition of functions for reading the `fvOptions` file entries, calculating the cell face areas, constructing the rotor coordinate system, and adding the source terms to the momentum equation.
- `rotorDiskSourceTemplates.C` - The bulk of the code in this file is directed into calculating the actual value of the source terms using the methodology described in the Theory section. The rest of the code in this file includes function which writes the source terms into the time directories when solving a particular flow case.
- `bladeModel/` - This folder contains two files that take in blade data such as radius, twist, and chord by reading the `fvOptions` file. These files also include member functions that interpolate values for twist and chord given a radial position of a cell inside the actuator disk volume.
- `profileModel/` - .`profileModel` class contains data on the lift and drag coefficients for each blade section. The airfoil lift and drag coefficients can be given in the `fvOptions` file either as lookup tables which are interpolated on, or as Fourier coefficients, which are then used to calculate the lift and drag coefficients. The lookup tables are the chosen approach for this report, and are explained in more detail in section 4.2.3.
- `trimModel/` - The `trimModel` class includes the possibility of simulating a fixed blade with constant blade angles, or vary the blade angle to reach a certain target thrust or torque. A fixed blade with constant blade angles is used throughout this report.

The most important file for this report is the `rotorDiskSourceTemplates.C` file which contains the code used for calculating the source terms for the momentum equation. A part of that file is shown below, and comments have been added (in capital letters) to it to explain the different steps involved in the process.

```
1  template<class RhoFieldType>
2  void Foam::fv::rotorDiskSource::calculate
3  (
4      const RhoFieldType& rho,
5      const vectorField& U,
6      const scalarField& thetag,
7      vectorField& force,
8      const bool divideVolume,
9      const bool output
10 ) const
11 {
12     const scalarField& V = mesh_.V();
13
14     // Logging info
15     scalar dragEff = 0.0;
16     scalar liftEff = 0.0;
17     scalar AOAmin = GREAT;
```

```
18      scalar AOAmax = -GREAT;
19
20      forAll(cells_, i)
21      {
22          if (area_[i] > ROOTVSMALL)
23          {
24              const label celli = cells_[i];
25
26              const scalar radius = x_[i].x();//THE CELL CENTER RADIUS IS
27                                              //OBTAINED HERE
28
29              // Transform velocity into local cylindrical reference frame
30              vector Uc = cylindrical_->invTransform(U[celli], i); //RADIUS,THETA,Z
31
32              // Transform velocity into local coning system
33              Uc = R_[i] & Uc;                //IN CASE THE ROTOR BLADES ARE SWEPT
34                                              //THEN THE COORDINATE SYSTEM BECOMES
35                                              //A CONE
36
37              // Set radial component of velocity to zero
38              Uc.x() = 0.0;                   //BET DISREGARDS THE RADIAL VELOCITY
39
40              // Set blade normal component of velocity
41              Uc.y() = radius*omega_ - Uc.y();//TAKE INTO CONSIDERATION ROTOR
42                                              //ROTATIONAL VELOCITY
43
44              // Determine blade data for this radius
45              // i2 = index of upper radius bound data point in blade list
46              scalar twist = 0.0;
47              scalar chord = 0.0;
48              label i1 = -1;
49              label i2 = -1;
50              scalar invDr = 0.0;
51              blade_.interpolate(radius, twist, chord, i1, i2, invDr);
52
53              // Flip geometric angle if blade is spinning in reverse (clockwise)
54              scalar alphaGeom = thetag[i] + twist;
55              if (omega_ < 0)
56              {
57                  alphaGeom = mathematical::pi - alphaGeom;
58              }
59
60              // Effective angle of attack    //CALCULATE ANGLE OF ATTACK OF SECTION
61              scalar alphaEff = alphaGeom - atan2(-Uc.z(), Uc.y());
62              if (alphaEff > mathematical::pi)
63              {
64                  alphaEff -= mathematical::twoPi;
65              }
66              if (alphaEff < -mathematical::pi)
67              {
68                  alphaEff += mathematical::twoPi;
69              }
70
71              AOAmin = min(AOAmin, alphaEff);
```

```
72              AOAmax = max(AOAmax, alphaEff);
73
74              // Determine profile data for this radius and angle of attack
75              const label profile1 = blade_.profileID()[i1];
76              const label profile2 = blade_.profileID()[i2];
77
78              scalar Cd1 = 0.0;
79              scalar Cl1 = 0.0;
80              profiles_[profile1].Cdl(alphaEff, Cd1, Cl1);
81
82              scalar Cd2 = 0.0;
83              scalar Cl2 = 0.0;
84              profiles_[profile2].Cdl(alphaEff, Cd2, Cl2);
85
86              scalar Cd = invDr*(Cd2 - Cd1) + Cd1; //INTERPOLATE ON CD
87              scalar Cl = invDr*(Cl2 - Cl1) + Cl1; //INTERPOLATE ON CL
88
89              // Apply tip effect for blade lift
90              scalar tipFactor = neg(radius/rMax_ - tipEffect_);
91
92              // Calculate forces perpendicular to blade
93              scalar pDyn = 0.5*rho[celli]*magSqr(Uc);
94
95              scalar f = pDyn*chord*nBlades_*area_[i]/radius/mathematical::twoPi;
96              vector localForce = vector(0.0, -f*Cd, tipFactor*f*Cl); //THIS PART OF THE CODE
97                                                                      //IS WRONG
98
99              // Accumulate forces
100             dragEff += rhoRef_*localForce.y();
101             liftEff += rhoRef_*localForce.z();
102
103             // Transform force from local coning system into rotor cylindrical
104             localForce = invR_[i] & localForce;
105
106             // Transform force into global Cartesian co-ordinate system
107             force[celli] = cylindrical_->transform(localForce, i);
108
109             if (divideVolume)
110             {
111                 force[celli] /= V[celli];
112             }
113         }
114     }
115
116     if (output)     //THIS OUTPUTS PERFORMANCE INTO THE TERMINAL
117     {
118         reduce(AOAmin, minOp<scalar>());
119         reduce(AOAmax, maxOp<scalar>());
120         reduce(dragEff, sumOp<scalar>());
121         reduce(liftEff, sumOp<scalar>());
122
123         Info<< type() << " output:" << nl
124             << "    min/max(AOA)   = " << radToDeg(AOAmin) << ", "
125             << radToDeg(AOAmax) << nl
```

```
126             << "    Effective drag = " << dragEff << nl
127             << "    Effective lift = " << liftEff << endl;
128     }
129 }
```

A much more in-depth explanation of the original `rotorDiskSource` implementation can be found in a report by Wahono [1]. The `rotorDiskSource` which is implemented in the standard OpenFOAM release is based on that work, but contains one major error where the local cell forces are calculated, leading to greatly under-predicted rotor torque and downstream swirl velocities. This error will be fixed in the next section.

3.2 Modifications

The three main modifications that are going to be made in the code involve:

1. Fixing an error in how the local forces are calculated.
2. Implementing a more general blade tip correction.
3. Implementing propeller performance parameters.

Start by copying the entire `fvOptions` folder and make sure that the compiled binary file ends up in the user directory by performing the following commands:

```
foam
cp -r --parents src/fvOptions $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/src/fvOptions
sed -i s/FOAM_LIBBIN/FOAM_USER_LIBBIN/g Make/files
cd sources/derived/rotorDiskSource/
```

Now we are in the main `rotorDiskSource` folder. The modifications to be done are shown below in the modified `rotorDiskSourceTemplates.C` file. Look for lines containing `NEW LINE` as comments on them.

```
130 /*---------------------------------------------------------------------------*\
131   =========                 |
132   \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
133    \\    /   O peration     |
134     \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
135      \\/     M anipulation  |
136 -------------------------------------------------------------------------------
137 License
138     This file is part of OpenFOAM.
139
140     OpenFOAM is free software: you can redistribute it and/or modify it
141     under the terms of the GNU General Public License as published by
142     the Free Software Foundation, either version 3 of the License, or
143     (at your option) any later version.
144
145     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
146     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
147     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
148     for more details.
149
150     You should have received a copy of the GNU General Public License
151     along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
```

```
152
153 \*---------------------------------------------------------------------------*/
154
155 #include "rotorDiskSource.H"
156 #include "volFields.H"
157 #include "unitConversion.H"
158
159 using namespace Foam::constant;
160
161 // * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * //
162
163 template<class RhoFieldType>
164 void Foam::fv::rotorDiskSource::calculate
165 (
166     const RhoFieldType& rho,
167     const vectorField& U,
168     const scalarField& thetag,
169     vectorField& force,
170     const bool divideVolume,
171     const bool output
172 ) const
173 {
174     const scalarField& V = mesh_.V();
175
176     // Logging info
177     scalar dragEff = 0.0;
178     scalar liftEff = 0.0;
179     scalar thrustEff = 0.0; //NEW LINE
180     scalar torqueEff = 0.0; //NEW LINE
181     scalar powerEff = 0.0; //NEW LINE
182     scalar AOAmin = GREAT;
183     scalar AOAmax = -GREAT;
184     scalar epsMin = GREAT; //NEW LINE
185     scalar epsMax = -GREAT; //NEW LINE
186
187     forAll(cells_, i)
188     {
189         if (area_[i] > ROOTVSMALL)
190         {
191             const label celli = cells_[i];
192
193             const scalar radius = x_[i].x();
194
195             // Transform velocity into local cylindrical reference frame
196             vector Uc = cylindrical_->invTransform(U[celli], i);
197
198             // Transform velocity into local coning system
199             Uc = R_[i] & Uc;
200
201             // Set radial component of velocity to zero
202             Uc.x() = 0.0;
203
204             // Set blade normal component of velocity
205             Uc.y() = radius*omega_ - Uc.y();
```

```
206
207             // Determine blade data for this radius
208             // i2 = index of upper radius bound data point in blade list
209             scalar twist = 0.0;
210             scalar chord = 0.0;
211             label i1 = -1;
212             label i2 = -1;
213             scalar invDr = 0.0;
214             blade_.interpolate(radius, twist, chord, i1, i2, invDr);
215
216             // Flip geometric angle if blade is spinning in reverse (clockwise)
217             scalar alphaGeom = thetag[i] + twist;
218             if (omega_ < 0)
219             {
220                 alphaGeom = mathematical::pi - alphaGeom;
221             }
222
223             // Effective angle of attack
224             scalar alphaEff = alphaGeom - atan2(-Uc.z(), Uc.y());
225             if (alphaEff > mathematical::pi)
226             {
227                 alphaEff -= mathematical::twoPi;
228             }
229             if (alphaEff < -mathematical::pi)
230             {
231                 alphaEff += mathematical::twoPi;
232             }
233
234             AOAmin = min(AOAmin, alphaEff);
235             AOAmax = max(AOAmax, alphaEff);
236
237             // Determine profile data for this radius and angle of attack
238             const label profile1 = blade_.profileID()[i1];
239             const label profile2 = blade_.profileID()[i2];
240
241             scalar Cd1 = 0.0;
242             scalar Cl1 = 0.0;
243             profiles_[profile1].Cdl(alphaEff, Cd1, Cl1);
244
245             scalar Cd2 = 0.0;
246             scalar Cl2 = 0.0;
247             profiles_[profile2].Cdl(alphaEff, Cd2, Cl2);
248
249             scalar Cd = invDr*(Cd2 - Cd1) + Cd1;
250             scalar Cl = invDr*(Cl2 - Cl1) + Cl1;
251
252             // NEW LINE - HERE WE COMMENT THE TWO LINES BELOW:
253             // Apply tip effect for blade lift
254             //scalar tipFactor = neg(radius/rMax_ - tipEffect_);
255
256             // Calculate forces perpendicular to blade
257             scalar pDyn = 0.5*rho[celli]*magSqr(Uc);
258
259             scalar f = pDyn*chord*nBlades_*area_[i]/radius/mathematical::twoPi;
```

```
260
261             /*======================NEW LINES=====================================*/
262             // Flow angle
263             scalar eps = atan2(-Uc.z(), Uc.y());
264             if (eps < -mathematical::pi)
265             {
266              eps = (2.0*mathematical::pi + eps);
267             }
268             if (eps > mathematical::pi)
269             {
270              eps = (eps - 2.0*mathematical::pi);
271             }
272             epsMin = min(epsMin, eps);
273             epsMax = max(epsMax, eps);
274
275             //Drela tip factor:
276             scalar lambdaVal = radius/(0.5*diameterRef_)*tan(eps);
277             scalar tipFactor_f = (nBlades_/2)*(1-radius/(0.5*diameterRef_))*(1/lambdaVal);
278             scalar tipFactor = 2/mathematical::pi*acos(exp(-tipFactor_f));
279
280             // Tangential and axial forces
281             scalar fTang = (f*Cd*cos(eps) + tipFactor*f*Cl*sin(eps));
282             scalar fAxial = (-f*Cd*sin(eps) + tipFactor*f*Cl*cos(eps));
283
284             vector localForce = vector(0.0,-fTang,fAxial);
285
286             /*====================================================================*/
287
288             // NEW LINE - HERE WE COMMENT THE LINE BELOW:
289             // vector localForce = vector(0.0, -f*Cd, tipFactor*f*Cl);
290
291             // Accumulate forces
292             dragEff += rhoRef_*localForce.y();
293             liftEff += rhoRef_*localForce.z();
294
295             thrustEff += rhoRef_*fAxial;  //NEW LINE
296             torqueEff += rhoRef_*fTang*radius;  //NEW LINE
297             powerEff += rhoRef_*fTang*radius*omega_;//NEW LINE
298
299             // Transform force from local coning system into rotor cylindrical
300             localForce = invR_[i] & localForce;
301
302             // Transform force into global Cartesian co-ordinate system
303             force[celli] = cylindrical_->transform(localForce, i);
304
305             if (divideVolume)
306             {
307                 force[celli] /= V[celli];
308             }
309         }
310     }
311
312 scalar etaProp = thrustEff*refVelEta_/powerEff; //NEW LINE
313
```

```
314     if (output)
315     {
316         reduce(AOAmin, minOp<scalar>());
317         reduce(AOAmax, maxOp<scalar>());
318         reduce(dragEff, sumOp<scalar>());
319         reduce(liftEff, sumOp<scalar>());
320 
321         Info<< type() << " output:" << nl
322             << "    min/max(AOA)   = " << radToDeg(AOAmin) << ", "
323             << radToDeg(AOAmax) << nl
324             << "    min/max(eps)   = " << radToDeg(epsMin) << ", "  //NEW LINE
325             << radToDeg(epsMax) << nl //NEW LINE
326             << "    Rotor thrust = " << thrustEff << nl //NEW LINE
327             << "    Rotor torque = " << torqueEff << nl //NEW LINE
328             << "    Rotor power = " << powerEff << nl //NEW LINE
329             << "    Rotor propeller efficiency = " << etaProp << nl //NEW LINE
330             << "    Effective drag = " << dragEff << nl
331             << "    Effective lift = " << liftEff << endl;
332     }
333 }
334 
335 
336 template<class Type>
337 void Foam::fv::rotorDiskSource::writeField
338 (
339     const word& name,
340     const List<Type>& values,
341     const bool writeNow
342 ) const
343 {
344     typedef GeometricField<Type, fvPatchField, volMesh> fieldType;
345 
346     if (mesh_.time().writeTime() || writeNow)
347     {
348         tmp<fieldType> tfield
349         (
350             new fieldType
351             (
352                 IOobject
353                 (
354                     name,
355                     mesh_.time().timeName(),
356                     mesh_,
357                     IOobject::NO_READ,
358                     IOobject::NO_WRITE
359                 ),
360                 mesh_,
361                 dimensioned<Type>("zero", dimless, Zero)
362             )
363         );
364 
365         Field<Type>& field = tfield.ref().primitiveFieldRef();
366 
367         if (cells_.size() != values.size())
```

```
368         {
369             FatalErrorInFunction
370                 << abort(FatalError);
371         }
372
373         forAll(cells_, i)
374         {
375             const label celli = cells_[i];
376             field[celli] = values[i];
377         }
378
379         tfield().write();
380     }
381 }
382
383
384 // ************************************************************************* //
```

The error in the code is located on line 289, which is now commented and replaced with the expressions on lines 281 to 284. The error is due to the fact that the original code used the blade drag as the tangential force and the blade lift as the axial force, which is only strictly true for rotors with a flow angle of zero, which is unrealistic for a generalized rotor. Instead, the lift and drag forces should be projected onto the rotor axial and tangential directions given the flow angle. This is what is done in lines 281 to 284.

The changes that have been made utilize some new variables (`diameterRef_` and `refVelEta_`) that have not yet been defined in the rest of the code. That is the next step, starting with `rotorDiskSource.H` just between the lines of code declaring the variables `rhoRef_` and `omega_`:

```
385         //- Reference density for incompressible case
386         scalar rhoRef_;
387
388         //- Reference diameter for calculating prandtl tip loss factor
389         scalar diameterRef_;  //NEW LINE
390
391         //- Reference velocity for calculating propeller efficiency
392         scalar refVelEta_;  //NEW LINE
393
394         //- Rotational speed [rad/s]
395         //  Positive anti-clockwise when looking along -ve lift direction
396         scalar omega_;
```

Next is the `rotorDiskSource.C` file where the constructors are defined:

```
397 Foam::fv::rotorDiskSource::rotorDiskSource
398 (
399     const word& name,
400     const word& modelType,
401     const dictionary& dict,
402     const fvMesh& mesh
403
404 )
405 :
406     cellSetOption(name, modelType, dict, mesh),
407     rhoRef_(1.0),
408     diameterRef_(1.0),  //NEW LINE
```

```
409     refVelEta_(0.0),//NEW LINE
410     omega_(0.0),
411     nBlades_(0),
412     inletFlow_(ifLocal),
413     inletVelocity_(Zero),
414     tipEffect_(1.0),
415     flap_(),
416     x_(cells_.size(), Zero),
417     R_(cells_.size(), I),
418     invR_(cells_.size(), I),
419     area_(cells_.size(), 0.0),
420     coordSys_(false),
421     cylindrical_(),
422     rMax_(0.0),
423     trim_(trimModel::New(*this, coeffs_)),
424     blade_(coeffs_.subDict("blade")),
425     profiles_(coeffs_.subDict("profiles"))
426 {
427     read(dict);
428 }
```

And one last change enabling the new variables to read from the `fvOptions` dictionary:

```
429 coeffs_.lookup("tipEffect") >> tipEffect_;
430
431 // Reference diameter for calculating prandtl tip loss factor
432 coeffs_.lookup("diameterRef") >> diameterRef_; //NEW LINE
433
434 // Reference velocity for calculating propeller efficiency
435 coeffs_.lookup("refVelEta") >> refVelEta_; //NEW LINE
436
437 const dictionary& flapCoeffs(coeffs_.subDict("flapCoeffs"));
438 flapCoeffs.lookup("beta0") >> flap_.beta0;
439 flapCoeffs.lookup("beta1c") >> flap_.beta1c;
440 flapCoeffs.lookup("beta2s") >> flap_.beta2s;
441 flap_.beta0 = degToRad(flap_.beta0);
442 flap_.beta1c = degToRad(flap_.beta1c);
443 flap_.beta2s = degToRad(flap_.beta2s);
```

All that remains now is to compile the code:

```
cd $WM_PROJECT_USER_DIR/src/fvOptions
wmake
```

3.3 Mesh requirements

The domain used in this report corresponds to the one showed in Fig. 3.1 and Fig. 3.2, with patches INLET, OUTLET and ROTORDISK. The boundary conditions for these patches are mentioned in section 4.2.1. The principal domain sizes are:

- ROTORDISK - The actual actuator disk volume, with a diameter D, hub to tip ratio of 0.3, and an axial length of 0.025D.
- FLUID - the overall domain volume, containing but not including ROTORDISK. It has a diameter of $10D$ and a length of $30D$.

The mesh used in the original tutorial case unfortunately caused the solver to diverge. After consulting the original report by Wahono [1] it was discovered that for best results the mesh used for the actuator disk volume (ROTORDISK) needs to be hexahedral and have cells with edges parallel to the radial and tangential directions, as is shown in Fig. 3.3. The number of cells in the axial direction should be one, and the mesh in the rest of the domain does not necessarily need to be of the hexahedral type, as in the ROTORDISK.

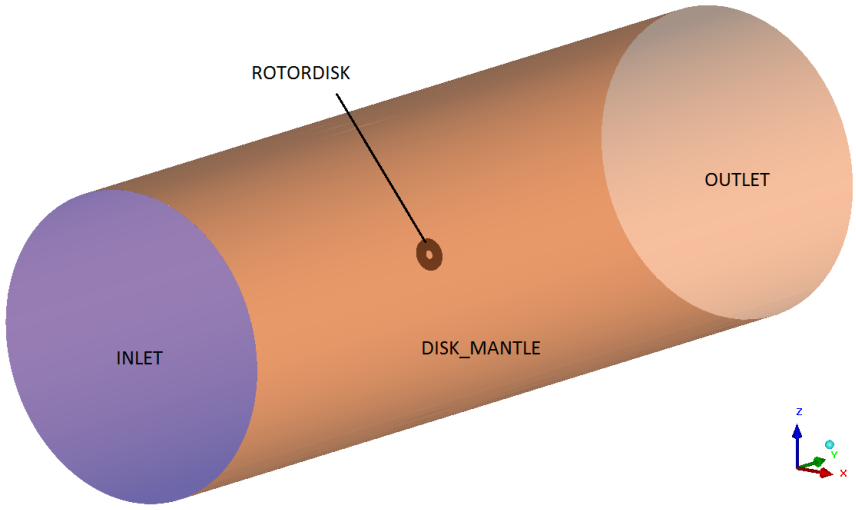


Figure 3.1: Simulation domain, surface patches and cylindrical actuator disk volume (ROTORDISK).

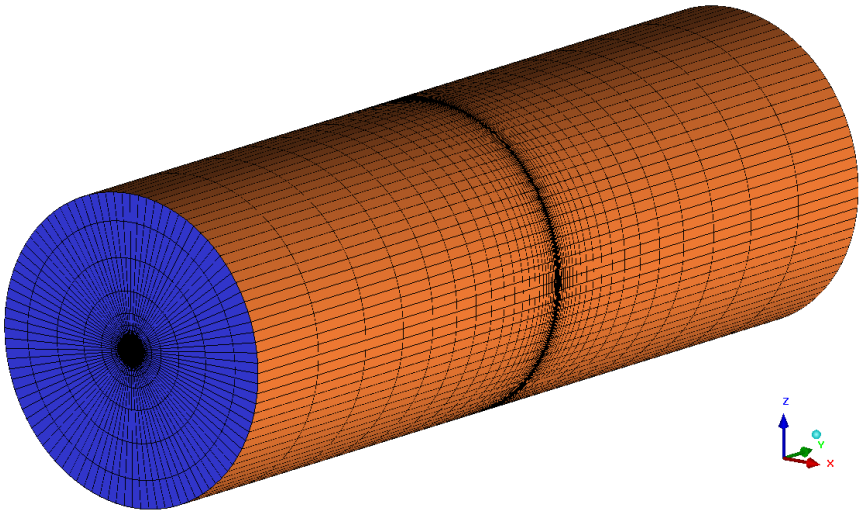


Figure 3.2: Computational mesh.

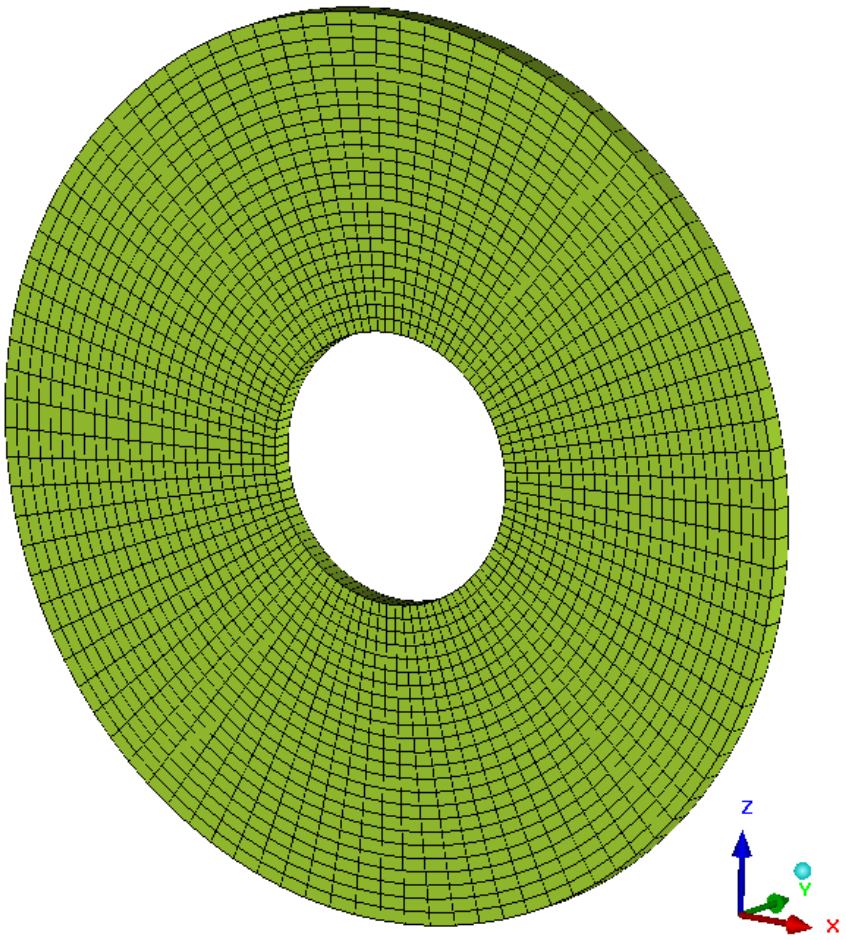


Figure 3.3: Mesh of cylindrical actuator disk volume (ROTORDISK).

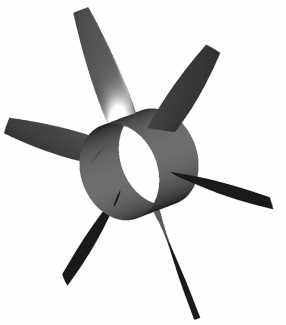
Chapter 4

Case setup

4.1 Propeller specification

The properties of the chosen reference propeller for analysis is shown in Table 4.1. The propeller was designed using an in-house propeller design code, but the properties needed to simulate in OpenFOAM will be provided in APPENDIX A. The 6-bladed reference propeller is unswept and designed to operate in the subsonic regime throughout the entire blade span. Beware that the specified thrust and efficiency values in Table 4.1 are not the values, but rather output from the propeller design program.

Number of blades B	6
Diameter D $[m]$	1.0
Hub-to-tip-ratio HTR	0.3
Altitude H $[m]$	0
Axial Mach number Ma	0.25
Advance ratio J	1.4208
Airfoil	NACA 16106
Design thrust T_{design} $[N]$	878.6
Design efficiency η_{design} $[N]$	82.76%



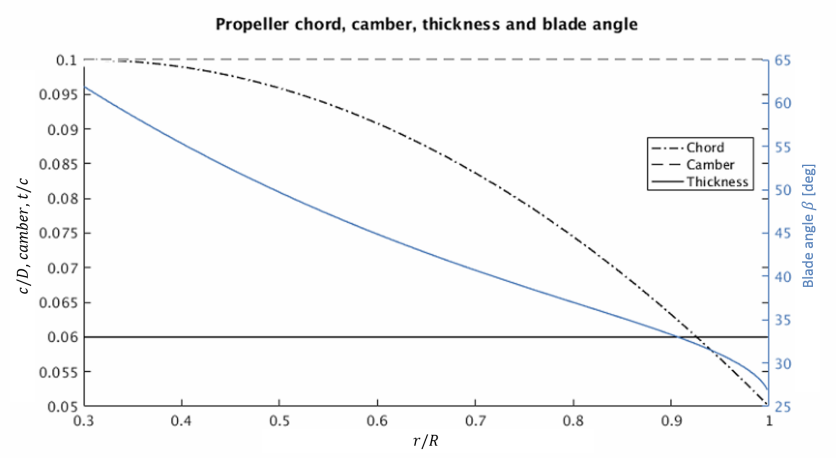


Figure 4.1: Reference propeller properties and geometry.

4.2 simpleFoam case setup

The case setup is started in the easiest way by copying the tutorial case from the OpenFOAM installation directory.

```
OF1706+
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/rotorDisk $FOAM_RUN/
cd $FOAM_RUN/rotorDisk
```

4.2.1 Initial and boundary conditions

Starting with the `0/k` file, one needs to change the names of the patches so that the file reads as below. Do the same type of changes on the patch names of the `0/nut` and `0/omega` files.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

kInlet          0.02;

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform $kInlet;

boundaryField
{
    INLET
    {
        type            fixedValue;
        value           uniform $kInlet;
    }

    OUTLET
    {
        type            inletOutlet;
        inletValue      uniform $kInlet;
        value           uniform $kInlet;
    }

    DISK_MANTLE
    {
        type            slip;
    }
```

```
485
486     #includeEtc "caseDicts/setConstraintTypes"
487 }
488
489 // ************************************************************************* //
```

The `0/p` file also needs its patches renamed, and the boundary condition for the `DISK_MANTLE` boundary condition to `zeroGradient`. The file should look like the one below.

```
490 /*--------------------------------*- C++ -*----------------------------------*\
491 | =========                 |                                                 |
492 | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
493 |  \\    /   O peration     | Version:  plus                                  |
494 |   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
495 |    \\/     M anipulation  |                                                 |
496 \*---------------------------------------------------------------------------*/
497 FoamFile
498 {
499     version     2.0;
500     format      ascii;
501     class       volScalarField;
502     object      p;
503 }
504 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
505
506 dimensions      [0 2 -2 0 0 0 0];
507
508 internalField   uniform 0;
509
510 boundaryField
511 {
512     INLET
513     {
514         type            zeroGradient;
515     }
516
517     OUTLET
518     {
519         type            fixedValue;
520         value           uniform 0;
521     }
522
523     DISK_MANTLE
524     {
525         type            zeroGradient;
526     }
527
528     #includeEtc "caseDicts/setConstraintTypes"
529 }
530
531 // ************************************************************************* //
```

Finally the `0/U` file should also have its patch names changed and the `OUTLET` patch should have a zeroGradient boundary condition. The inlet velocity should also be increased to 85.0725 m/s as is shown below.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

Uinlet          (0 85.0725 0);

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

internalField   uniform $Uinlet;

boundaryField
{
    INLET
    {
        type            fixedValue;
        value           uniform $Uinlet;
    }

    OUTLET
    {
        type            zeroGradient;
    }

    DISK_MANTLE
    {
        type            zeroGradient;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}

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

4.2.2 The constant folder

First we need to delete the existing mesh and extract the new one from the accompanying files. This assumes that you have moved the `prop_incomp_refCase.tar.gz` file to the current directory. The included mesh (`fluent1.msh`) needs to be converted from the ANSYS Fluent format to a mesh

format that can be used by OpenFOAM. This is done with the command `fluent3DMeshToFoam` as shown below.

```
cd $FOAM_RUN/rotorDisk
rm -r constant/triSurface/
tar -xvzf prop_incomp_refCase.tar
cp prop_incomp_refCase/fluent1.msh .
fluent3DMeshToFoam fluent1.msh
```

The `fluent3DMeshToFoam` command has been used since it successfully converts an ICEM CFD mesh with two blocking parts "FLUID" and "ROTORDISK" to a mesh with two correspondingly named cell zones that can be used for defining the overall domain and the cylindrical actuator disk volume. Check that the mesh has these two distinct cell zones, `FLUID` and `ROTORDISK`, by searching the text file using `grep`.

```
grep -n 'FLUID' constant/polyMesh/cellZones
grep -n 'ROTORDISK' constant/polyMesh/cellZones
```

The `constant/transportProperties` and `constant/turbulenceProperties` files can be left as they are.

4.2.3 The system folder

Since the tutorial case meshing routines are not used for this case, we can delete the `blockMeshDict`, `meshQualityDict`, `snappyHexMeshDict`, and `surfaceFeatureExtractDict` files.

```
rm system/blockMeshDict
rm system/meshQualityDict
rm system/snappyHexMeshDict
rm system/surfaceFeatureExtractDict
```

The most important file to change is the `system/fvOptions` file since it contains the dictionary entry needed for using the `rotorDiskSource` class. The entire file included in Appendix A due to its length. It is highly recommended to just take it from the case files of this report. The most important parts of that file are shown below, and includes entries for the number of propeller blades, diameter, rotational velocity, and how to set up the rotor coordinate system. The `system/fvOptions` file also includes entries regarding the blade geometry and its sectional properties (radius, chord, blade angle, c_l, and c_d) and are included in the full file in Appendix A. The blade geometry is described under `BLADE SPECIFICATION` below and is in this case composed of 20 blade sections (`sect0` to `sect19`), each with their own radial position, blade angle, and chord. As few as two sections can be used to describe the blade. The lift and drag coefficients for each section are function of angle-of-attack, and are tabulated under the `profiles` entry, one table per blade section.

```
576 /*--------------------------------*- C++ -*----------------------------------*\
577 | =========                 |                                                 |
578 | \      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
579 |  \    /   O peration     | Version:  plus                                  |
580 |   \  /    A nd           | Web:      www.OpenFOAM.com                      |
581 |    \/     M anipulation  |                                                 |
582 \*---------------------------------------------------------------------------*/
583 FoamFile
584 {
585     version     2.0;
586     format      ascii;
587     class       dictionary;
588     object      fvOptions;
589 }
```

```
590 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
591
592 disk
593 {
594     type            rotorDisk;  // Specifying that the rotorDiskSource class
595                                 // is to be used for sources in the domain
596
597     selectionMode   cellZone;
598     cellZone        ROTORDISK; // how to choose which cellZone to use as the
599                                 // cylindrical actuator disk volume
600
601     fields          (U);    // Names of fields on which to apply source
602     nBlades         6;      // Number of blades
603     tipEffect       1.00;   // Normalised radius above which lift = 0,
604                             // should be set to 1 if using a Prandtl tip corr.
605
606     diameterRef 1.0; // Propeller reference diameter (used to calculate
607                             // prandtl tip loss)
608     refVelEta 85.0725;// Reference velocity for calculating
609                             // propeller efficiency
610
611     inletFlowType   local;  // Inlet flow type specification
612     inletVelocity   (0 1 0);
613
614     geometryMode    specified;
615
616     origin          (0 0 0); //used for constructing the rotor coordinate system
617     axis            (0 -1 0);//used for constructing the rotor coordinate system
618
619     refDirection    (0 0 1);// Reference direction
620                             // Used as reference for psi angle (ie radial dir)
621
622     rpm             3592.49;//rotational velocity of the rotor
623
624     trimModel       fixedTrim;  // fixed || targetForce
625                                 // fixedTrim if the rotor blade angle is fixed
626
627     rhoRef          1.225; // reference density
628     rhoInf          1.225;
629
630     fixedTrimCoeffs // disregard if rotor blade angle is fixed
631     {
632          theta0         0;
633          theta1c        0;
634          theta1s        0;
635     }
636
637     flapCoeffs
638     {
639         beta0           0;  // Coning angle [deg] - set to 0 if rotor disk is
640                             // purely cylindrical
641
642         beta1c          0;  // Lateral flapping coeff (cos coeff)
643         beta2s          0;  // Longitudinal flapping coeff (sin coeff)
```

```
644     }
645
646 // BLADE SPECIFICATION
647
648     blade   // This entry lists a number of sections which together define a
649             // piece-wise linear propeller.
650     {
651         data
652         (
653             (sect0 (0.15 61.9206 0.1)) // radius [m], blade angle [deg], chord [m]
654             (sect1 (0.18953 56.6428 0.0993622))
655             (sect2 (0.226692 52.2266 0.0975993))
656             (sect3 (0.250152 49.6876 0.095906))
657             (sect4 (0.283367 46.3974 0.0927401))
658             (sect5 (0.304195 44.5014 0.0902955))
659             (sect6 (0.333464 42.0245 0.0862617))
660             (sect7 (0.360364 39.9141 0.0819375))
661             (sect8 (0.376982 38.6725 0.0789711))
662             (sect9 (0.399936 37.0064 0.0745029))
663             (sect10 (0.413923 36.0012 0.0715693))
664             (sect11 (0.43293 34.6101 0.0673269))
665             (sect12 (0.444285 33.741 0.0646515))
666             (sect13 (0.459345 32.504 0.060941))
667             (sect14 (0.472038 31.3201 0.0576701))
668             (sect15 (0.479183 30.5484 0.0557708))
669             (sect16 (0.487929 29.4165 0.0533895))
670             (sect17 (0.492443 28.6763 0.0521358))
671             (sect18 (0.497242 27.5931 0.050785))
672             (sect19 (0.499125 26.8997 0.0502497))
673         );
674     }
675
676     profiles
677     {
678         sect0 // This entry gives cl and cd as function of angle-of-attack
679         {
680             type lookup;
681             data
682             (
683                 (-180.0 0.0532754 -0.726446) //angle-of-attack [deg], cd, cl
684                 (-8.0 0.0532754 -0.726446)
685                 (-7.0 0.045269 -0.622595)
686                 (-6.0 0.0372626 -0.518745)
687                 (-5.0 0.0269264 -0.414894)
688                 (-4.0 0.0182272 -0.311043)
689                 (-3.0 0.00920077 -0.207193)
690                 (-2.0 0.00716674 -0.103342)
691                 (-1.0 0.00540988 0.000508721)
692                 (0.0 0.00525296 0.104359)
693                 (1.0 0.00540988 0.20821)
694                 (2.0 0.00716674 0.312061)
695                 (3.0 0.00920077 0.405666)
696                 (4.0 0.0182272 0.499271)
697                 (5.0 0.0269264 0.589483)
```

```
                (6.0 0.0372626 0.679694)
                (7.0 0.045269 0.748142)
                (8.0 0.0532754 0.816589)
                (180.0 0.0532754 0.816589)
            );
        }

        // HERE ARE SECTIONS 1-18, SEE APPENDIX

        sect19
        {
            type lookup;
            data
            (
                (-180.0 0.043025 -0.720867)
                (-8.0 0.043025 -0.720867)
                (-7.0 0.036952 -0.616603)
                (-6.0 0.0308791 -0.512338)
                (-5.0 0.0218817 -0.408073)
                (-4.0 0.0152767 -0.303809)
                (-3.0 0.00911313 -0.199544)
                (-2.0 0.00680099 -0.0952794)
                (-1.0 0.00551115 0.00898521)
                (0.0 0.00527861 0.11325)
                (1.0 0.00551115 0.217514)
                (2.0 0.00680099 0.321779)
                (3.0 0.00911313 0.421833)
                (4.0 0.0152767 0.521887)
                (5.0 0.0218817 0.619438)
                (6.0 0.0308791 0.716989)
                (7.0 0.036952 0.78283)
                (8.0 0.043025 0.848671)
                (180.0 0.043025 0.848671)
            );
        }
    }
}

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

Finally, the `system/fvSolution` file should have its `relaxationFactors` changed to 0.5.

```
relaxationFactors
{
    equations
    {
        U                   0.5;
        "(k|omega|epsilon)" 0.5;
    }
}
```

4.2.4 Running the case

The case can simply be run with the command below, with the propeller performance values written out in the log file.

```
simpleFoam &> log &
tailf log
```

While solving the code should output rotor thrust, torque, power, and efficiency:

```
Time = 171

rotorDisk output:
    min/max(AOA)   = 1.6617577, 4.5364105
    min/max(eps)   = 25.235736, 59.281513
    Rotor thrust = 671.4494
    Rotor torque = 188.8235
    Rotor power = 71036.283
    Rotor propeller efficiency = 0.80412257
    Effective drag = -576.03331
    Effective lift = 671.4494
```

4.3 rhoSimpleFoam case setup

This section will show how to setup a case which includes the functionality contained within the `rotorDisk` class on a different solver (`rhoSimpleFoam`) than the one included in the original `rotorDisk` tutorial case. Results will not be shown for this solver.

First step is to copy the existing case for `simpleFoam` to a new directory and clean any existing results files.

```
run
cp -r rotorDisk rotorDiskComp
cd rotorDiskComp
rm -r [1-9]* log
```

4.3.1 Initial and boundary conditions

Two initial/boundary condition files need to be created, namely `0/alphat` and `0/T`. See below for `0/alphat`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    INLET
    {
        type            calculated;
        value           uniform 0;
    }

    OUTLET
    {
        type            calculated;
        value           uniform 0;
    }

    DISK_MANTLE
    {
        type            calculated;
```

```
        value           uniform 0;
    }
}

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

The temperature boundary and initial file `0/T`:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 288.15;

boundaryField
{
    OUTLET
    {
        type            zeroGradient;
    }

    INLET
    {
        type            fixedValue;
        value           uniform 288.15;
    }

    DISK_MANTLE
    {
        type            zeroGradient;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}

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

The pressure value in the `0/p` file needs to be changed to ambient pressure, as is shown below. The dimensions are also required to be changed to what is seen below.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    INLET
    {
        type            zeroGradient;
    }

    OUTLET
    {
        type            fixedValue;
        value           uniform 101325;
    }

    DISK_MANTLE
    {
        type            zeroGradient;
    }

    #includeEtc "caseDicts/setConstraintTypes"
}

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

4.3.2 The constant folder

After setting the initial and boundary conditions, the `constant/thermophysicalProperties` file needs to be created:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       sutherland;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight   28.96;
    }
    thermodynamics
    {
        Cp          1004.5;
        Hf          0;
    }
    transport
    {
        As          1.4792e-06;
Ts 116;
Pr  0.7;
    }
}


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

4.3.3 The system folder

First the chosen solver needs to be changed to `rhoSimpleFoam`:

```
sed -i s/simpleFoam/rhoSimpleFoam/g system/controlDict
```

Solver, smoother, and tolerances needs to be set for the energy equation using the command below:

```
sed -i s/"(U|k|omega|epsilon)"/"(U|k|e|omega|epsilon)"/g system/fvSolution
```

The `fvSchemes` file needs to be modified to look like below:

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
grad(U) faceLimited Gauss linear 1;
limitedGrad(h) cellLimited Gauss linear 1;
}

divSchemes
{
    default         none;
div(phi,U) Gauss upwind grad(U);
div(phi,e) Gauss upwind;
div(phi,k) Gauss upwind;
    div(phi,omega)  Gauss upwind;
    div(phi,epsilon) Gauss upwind;
div(phi,K) Gauss upwind;
div(U,p) Gauss linear;
div((nuEff*dev(T(grad(U))))) Gauss linear;
div(((rho*nuEff)*dev2(T(grad(U))))      Gauss linear;
div(phi,Ekp) Gauss upwind;

}
```

```
963
964
965
966 laplacianSchemes
967 {
968     default         Gauss linear corrected;
969 }
970
971 interpolationSchemes
972 {
973     default         linear;
974 }
975
976 snGradSchemes
977 {
978     default         limited 0.333;
979 }
980
981 wallDist
982 {
983     method meshWave;
984 }
985
986 // ************************************************************************* //
```

4.3.4 Running the case

The case can be run using the command below.

```
rhoSimpleFoam &> log &
```

Chapter 5

Results

The reference propeller design has been analyzed using `simpleFoam` and the `rotorDiskSource` methodology. For the sake of comparison, the full blade geometry has been simulated in ANSYS CFX using the methodology from a previous paper by the author [3]. The results for these simulations and the design target performance are shown in Table 5.1. The efficiency values are relatively similar, but the thrust and power values in OpenFOAM and CFX differ by approximately 15%, so there is still some work to be done in terms of the choice of boundary conditions, turbulence models, and other settings. Also included in the table are the results for running the reference propeller design in the original OpenFOAM code (before the bug fix), which show that both thrust, power, and propeller efficiency are wildly over-predicted.

Table 5.1: Performance values for the reference propeller design from design program, OpenFOAM simulation, and CFX full blade simulations.

	$T\,[N]$	$P\,[kW]$	η
Design target:	878.6	90.31	82.76%
CFX:	783.8	81.20	82.15%
OpenFOAM:	671.4	71.03	80.41%
OpenFOAM (with bug):	1083	6.233	1478%

The axial and swirl velocities on a plane one D downstream of the rotor are shown in Fig. 5.1 and 5.2. The axial velocity values are similar in magnitude and the unloading of the tip (using the new tip correction) is visible in the OpenFOAM case and similar to the flow in the CFX case. Similar behaviour is seen for the swirl velocities, although the peak velocities are underpredicted by OpenFOAM, which is corroborated by difference in power between the OpenFOAM and CFX cases.

The reference propeller design has been simulated in OpenFOAM and CFX for different advance ratios $J = V_0/nD$ (i.e. different rotational velocities) and the thrust and efficiency values are plotted in Fig. 5.3. The thrust values are well-matched throughout the span of analyzed advance ratios, but the efficiency starts to deteriorate for lower advance ratios. This is probably due to some sections of the propeller stalling - a 3D flow phenomenon which cannot be accounted for by using the actuator disk and BET methodology.

The original `rotorDiskSource` implementation under-predicted the rotor torque and downstream swirl velocities. This error was fixed in section 3.2, and the effect on swirl velocities can be seen in Fig. 5.4, where the original code featured swirl velocities a magnitude lower than the fixed code.

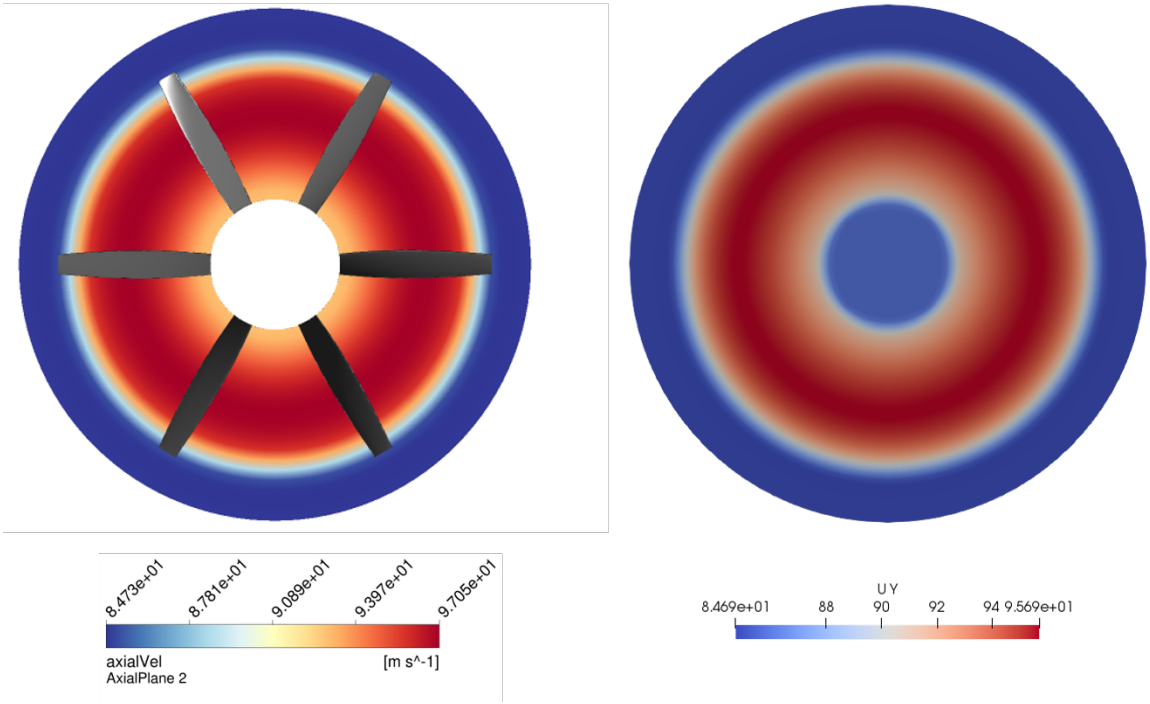


Figure 5.1: Axial velocity contours for the reference propeller simulated with ANSYS CFX (left) and OpenFOAM (right). Plane located one D downstream of propeller.

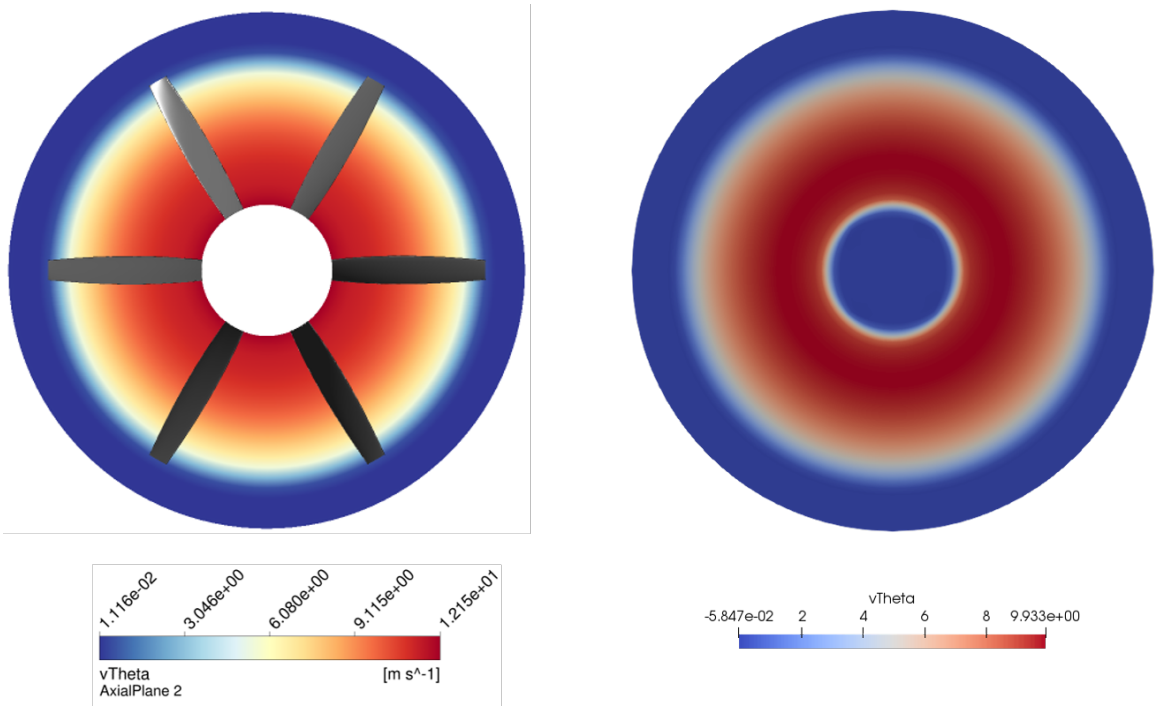


Figure 5.2: Swirl velocity contours for the reference propeller simulated with ANSYS CFX (left) and OpenFOAM (right). Plane located one D downstream of propeller.

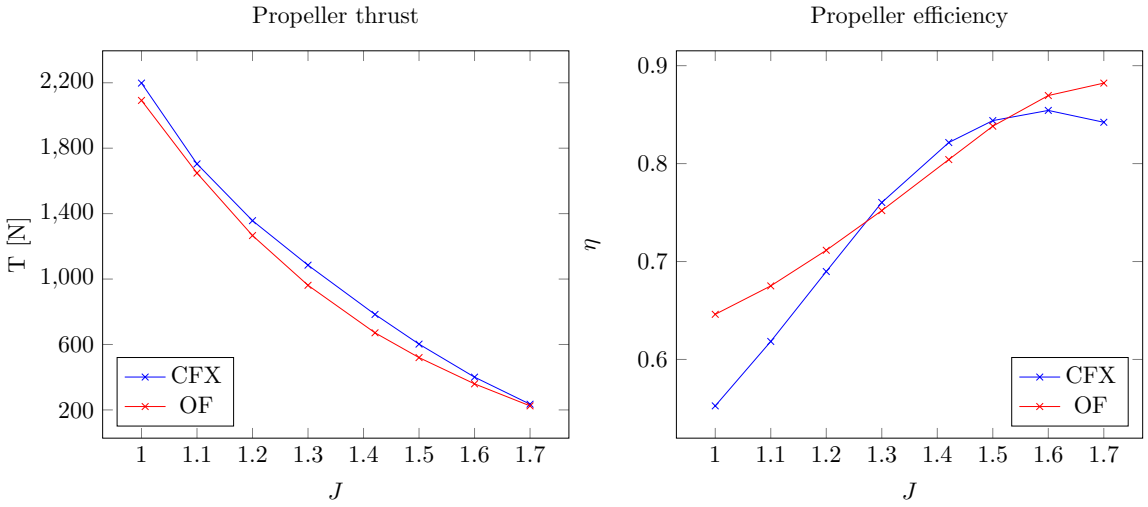


Figure 5.3: Performance values from OpenFoam (OF) and ANSYS CFX simulations for different advance ratios $J = V_0/nD$.

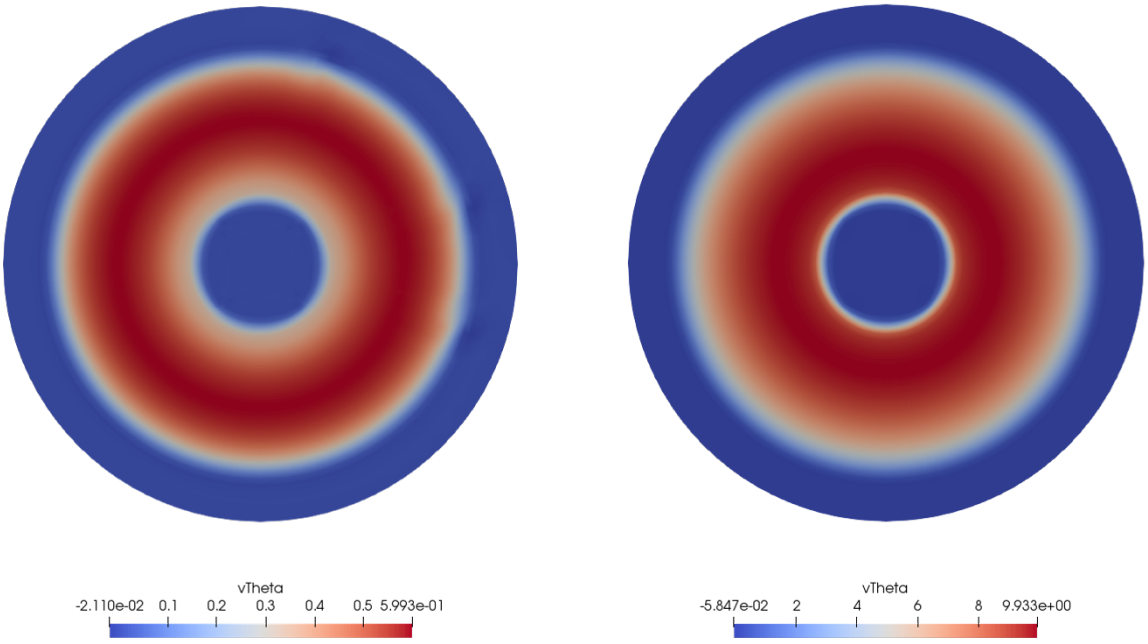


Figure 5.4: Swirl velocity contours for the reference propeller simulated with the original (left) and modified (right) rotorDiskSource class.

Chapter 6

Conclusions

- The OpenFOAM built-in `rotorDiskSource` class provides a very fast way of simulating the time-averaged performance of a propeller (3.5 minutes per case in this report).
- The `rotorDiskSource` class has a critical bug which results in greatly under-predicted rotor torque and downstream swirl velocities, but this bug can be fixed relatively easy.
- A more general tip correction factor than the existing one has been implemented, and results in a decrease in axial and swirl velocities at the rotor tip which are similar to what is found in CFD simulations of the entire blade geometry.
- The values of thrust and torque are under-predicted for the OpenFOAM simulations relative to the CFX full blade simulations, and future work should investigate the effect of different boundary conditions and turbulence models on the results.
- The variation in thrust and efficiency with respect to advance ratio is relatively similar for the OpenFOAM and CFX simulations, but the OpenFOAM values start to differ when the propeller starts stalling at lower advance ratios.

Study questions

How to use it:

1. What is the rotorDiskSource class useful for?
2. What kind cells are best suited for meshing the actuator disk volume?
3. What OpenFOAM utility can be used to convert fluent meshes so that they can be used by OpenFOAM?
4. Why does the propeller efficiency differ significantly between the OpenFOAM and CFX simulations for lower advance ratios?

The theory of it:

5. What is the main benefit in using an actuator disk over a full 3D blade simulation?
6. Does the Blade Element Theory account for 3D flow effects? Name some of these 3D flow phenomena.
7. Which flow velocities does the BET take into consideration when calculating a blade section angle-of-attack?

How it is implemented:

8. How did the main error in the rotorDiskSource class manifest itself in the flow?

How to modify it:

9. In which file is the main calculation for calculating the rotor forces located? Provide a full path using OpenFOAM environment variables.
10. How was the main error in the rotorDiskSource class fixed?

References

1. WAHONO, Stefano, "Development of Virtual Blade Model for Modelling Helicopter Rotor Downwash in OpenFOAM". 2013.

2. DRELA, Mark, "QPROP formulation", Massachusetts Inst. of Technology Aeronautics and Astronautics, Cambridge, MA, 2006.

3. Capitao Patrao, A., Grönstedt, T., Avellán, R., Lundbladh, A., and Montero Villar, G., "An Optimization Platform for High Speed Propellers", Swedish Aerospace Technology Congress 2016, 2016.

Appendix A

In this appendix the `system/fvOptions` file is included in its entirety.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \    /   O peration     | Version:  plus                                  |
|   \  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

disk
{
    type            rotorDisk;  // Specifying that the rotorDiskSource class is to be
// used for sources in the domain

    selectionMode   cellZone;
    cellZone        ROTORDISK; // how to choose which cellZone to use as the
// cylindrical actuator disk volume

    fields          (U);    // Names of fields on which to apply source
    nBlades         6;      // Number of blades
    tipEffect       1.00;   // Normalised radius above which lift = 0,
// should be set to 1 if using a Prandtl tip corr.

    diameterRef 1.0; // Propeller reference diameter (used to calculate
// prandtl tip loss)
    refVelEta 85.0725;// Reference velocity for calculating
// propeller efficiency

    inletFlowType   local;  // Inlet flow type specification
    inletVelocity   (0 1 0);

    geometryMode    specified;

    origin          (0 0 0); //used for constructing the rotor coordinate system
```

```
1028     axis            (0 -1 0);//used for constructing the rotor coordinate system
1029
1030     refDirection    (0 0 1);// Reference direction
1031                             // Used as reference for psi angle (ie radial dir)
1032
1033 rpm             3592.49;//rotational velocity of the rotor
1034
1035     trimModel       fixedTrim;  // fixed || targetForce - fixedTrim if the rotor
1036 // blade angle is fixed
1037
1038     rhoRef          1.225; // reference density
1039     rhoInf          1.225;
1040
1041     fixedTrimCoeffs // disregard if rotor blade angle is fixed
1042     {
1043          theta0         0;
1044          theta1c        0;
1045          theta1s        0;
1046     }
1047
1048     flapCoeffs
1049     {
1050         beta0           0;  // Coning angle [deg] - set to 0 if rotor disk is
1051 // purely cylindrical
1052
1053         beta1c          0;  // Lateral flapping coeff (cos coeff)
1054         beta2s          0;  // Longitudinal flapping coeff (sin coeff)
1055     }
1056
1057 // BLADE SPECIFICATION
1058
1059     blade  // This entry lists a number of sections which together define a
1060 // piece-wise linear propeller.
1061     {
1062         data
1063         (
1064             (sect0 (0.15 61.9206 0.1)) // radius [m], blade angle [deg], chord [m]
1065             (sect1 (0.18953 56.6428 0.0993622))
1066             (sect2 (0.226692 52.2266 0.0975993))
1067             (sect3 (0.250152 49.6876 0.095906))
1068             (sect4 (0.283367 46.3974 0.0927401))
1069             (sect5 (0.304195 44.5014 0.0902955))
1070             (sect6 (0.333464 42.0245 0.0862617))
1071             (sect7 (0.360364 39.9141 0.0819375))
1072             (sect8 (0.376982 38.6725 0.0789711))
1073             (sect9 (0.399936 37.0064 0.0745029))
1074             (sect10 (0.413923 36.0012 0.0715693))
1075             (sect11 (0.43293 34.6101 0.0673269))
1076             (sect12 (0.444285 33.741 0.0646515))
1077             (sect13 (0.459345 32.504 0.060941))
1078             (sect14 (0.472038 31.3201 0.0576701))
1079             (sect15 (0.479183 30.5484 0.0557708))
1080             (sect16 (0.487929 29.4165 0.0533895))
1081             (sect17 (0.492443 28.6763 0.0521358))
```

```
            (sect18 (0.497242 27.5931 0.050785))
            (sect19 (0.499125 26.8997 0.0502497))
        );
    }

    profiles
    {
        sect0 // This entry gives cl and cd as function of angle-of-attack
        {
            type lookup;
            data
            (
                (-180.0 0.0532754 -0.726446) //angle-of-attack [deg], cd, cl
                (-8.0 0.0532754 -0.726446)
                (-7.0 0.045269 -0.622595)
                (-6.0 0.0372626 -0.518745)
                (-5.0 0.0269264 -0.414894)
                (-4.0 0.0182272 -0.311043)
                (-3.0 0.00920077 -0.207193)
                (-2.0 0.00716674 -0.103342)
                (-1.0 0.00540988 0.000508721)
                (0.0 0.00525296 0.104359)
                (1.0 0.00540988 0.20821)
                (2.0 0.00716674 0.312061)
                (3.0 0.00920077 0.405666)
                (4.0 0.0182272 0.499271)
                (5.0 0.0269264 0.589483)
                (6.0 0.0372626 0.679694)
                (7.0 0.045269 0.748142)
                (8.0 0.0532754 0.816589)
                (180.0 0.0532754 0.816589)
            );
        }
        sect1
        {
            type lookup;
            data
            (
                (-180.0 0.052773 -0.726282)
                (-8.0 0.052773 -0.726282)
                (-7.0 0.0448664 -0.622431)
                (-6.0 0.0369598 -0.518581)
                (-5.0 0.0266581 -0.41473)
                (-4.0 0.0180578 -0.31088)
                (-3.0 0.00917065 -0.207029)
                (-2.0 0.00714089 -0.103178)
                (-1.0 0.00541159 0.000672425)
                (0.0 0.00525418 0.104523)
                (1.0 0.00541159 0.208374)
                (2.0 0.00714089 0.312224)
                (3.0 0.00917065 0.405996)
                (4.0 0.0180578 0.499767)
                (5.0 0.0266581 0.590514)
                (6.0 0.0369598 0.681261)
```

```
            (7.0 0.0448664 0.749592)
            (8.0 0.052773 0.817924)
            (180.0 0.052773 0.817924)
        );
    }
    sect2
    {
        type lookup;
        data
        (
            (-180.0 0.0522499 -0.726113)
            (-8.0 0.0522499 -0.726113)
            (-7.0 0.0444462 -0.622262)
            (-6.0 0.0366425 -0.518412)
            (-5.0 0.0263772 -0.414561)
            (-4.0 0.0178809 -0.31071)
            (-3.0 0.00913839 -0.20686)
            (-2.0 0.00711406 -0.103009)
            (-1.0 0.00541327 0.000841763)
            (0.0 0.00525545 0.104692)
            (1.0 0.00541327 0.208543)
            (2.0 0.00711406 0.312394)
            (3.0 0.00913839 0.406337)
            (4.0 0.0178809 0.50028)
            (5.0 0.0263772 0.591581)
            (6.0 0.0366425 0.682881)
            (7.0 0.0444462 0.751093)
            (8.0 0.0522499 0.819305)
            (180.0 0.0522499 0.819305)
        );
    }
    sect3
    {
        type lookup;
        data
        (
            (-180.0 0.0517095 -0.725939)
            (-8.0 0.0517095 -0.725939)
            (-7.0 0.0440111 -0.622088)
            (-6.0 0.0363127 -0.518238)
            (-5.0 0.0260852 -0.414387)
            (-4.0 0.0176975 -0.310536)
            (-3.0 0.00910411 -0.206686)
            (-2.0 0.00708644 -0.102835)
            (-1.0 0.00541489 0.00101553)
            (0.0 0.00525674 0.104866)
            (1.0 0.00541489 0.208717)
            (2.0 0.00708644 0.312567)
            (3.0 0.00910411 0.406687)
            (4.0 0.0176975 0.500807)
            (5.0 0.0260852 0.592675)
            (6.0 0.0363127 0.684544)
            (7.0 0.0440111 0.752633)
            (8.0 0.0517095 0.820721)
```

```
            (180.0 0.0517095 0.820721)
        );
    }
    sect4
    {
        type lookup;
        data
        (
            (-180.0 0.0511549 -0.725762)
            (-8.0 0.0511549 -0.725762)
            (-7.0 0.0435635 -0.621911)
            (-6.0 0.035972 -0.518061)
            (-5.0 0.0257936 -0.41421)
            (-4.0 0.0175287 -0.310359)
            (-3.0 0.00907786 -0.206509)
            (-2.0 0.00705818 -0.102658)
            (-1.0 0.00541646 0.00119264)
            (0.0 0.00525807 0.105043)
            (1.0 0.00541646 0.208894)
            (2.0 0.00705818 0.312745)
            (3.0 0.00907786 0.407156)
            (4.0 0.0175287 0.501568)
            (5.0 0.0257936 0.593903)
            (6.0 0.035972 0.686238)
            (7.0 0.0435635 0.754202)
            (8.0 0.0511549 0.822166)
            (180.0 0.0511549 0.822166)
        );
    }
    sect5
    {
        type lookup;
        data
        (
            (-180.0 0.050594 -0.725582)
            (-8.0 0.050594 -0.725582)
            (-7.0 0.0431109 -0.621732)
            (-6.0 0.0356278 -0.517881)
            (-5.0 0.025506 -0.41403)
            (-4.0 0.0173766 -0.31018)
            (-3.0 0.00906045 -0.206329)
            (-2.0 0.00702945 -0.102478)
            (-1.0 0.00541794 0.00137215)
            (0.0 0.00525941 0.105223)
            (1.0 0.00541794 0.209073)
            (2.0 0.00702945 0.312924)
            (3.0 0.00906045 0.407755)
            (4.0 0.0173766 0.502586)
            (5.0 0.025506 0.595288)
            (6.0 0.0356278 0.68799)
            (7.0 0.0431109 0.75581)
            (8.0 0.050594 0.823629)
            (180.0 0.050594 0.823629)
        );
```

```
        }
        sect6
        {
            type lookup;
            data
            (
                (-180.0 0.0500445 -0.725401)
                (-8.0 0.0500445 -0.725401)
                (-7.0 0.0426724 -0.621551)
                (-6.0 0.0353003 -0.5177)
                (-5.0 0.0252202 -0.413849)
                (-4.0 0.0172269 -0.309999)
                (-3.0 0.00904665 -0.206148)
                (-2.0 0.00700211 -0.102297)
                (-1.0 0.00541947 0.00155323)
                (0.0 0.00526076 0.105404)
                (1.0 0.00541947 0.209255)
                (2.0 0.00700211 0.313105)
                (3.0 0.00904665 0.408394)
                (4.0 0.0172269 0.503683)
                (5.0 0.0252202 0.596795)
                (6.0 0.0353003 0.689907)
                (7.0 0.0426724 0.757506)
                (8.0 0.0500445 0.825106)
                (180.0 0.0500445 0.825106)
            );
        }
        sect7
        {
            type lookup;
            data
            (
                (-180.0 0.0495146 -0.725737)
                (-8.0 0.0495146 -0.725737)
                (-7.0 0.0422559 -0.621822)
                (-6.0 0.0349971 -0.517906)
                (-5.0 0.0249359 -0.413991)
                (-4.0 0.0170753 -0.310076)
                (-3.0 0.00904132 -0.20616)
                (-2.0 0.00697975 -0.102245)
                (-1.0 0.00542214 0.00167043)
                (0.0 0.00526212 0.105586)
                (1.0 0.00542214 0.209501)
                (2.0 0.00697975 0.313416)
                (3.0 0.00904132 0.4091)
                (4.0 0.0170753 0.504784)
                (5.0 0.0249359 0.598405)
                (6.0 0.0349971 0.692026)
                (7.0 0.0422559 0.759315)
                (8.0 0.0495146 0.826604)
                (180.0 0.0495146 0.826604)
            );
        }
        sect8
```

```
1298        {
1299            type lookup;
1300            data
1301            (
1302                (-180.0 0.0489959 -0.727412)
1303                (-8.0 0.0489959 -0.727412)
1304                (-7.0 0.0418421 -0.623265)
1305                (-6.0 0.0346883 -0.519117)
1306                (-5.0 0.0246517 -0.41497)
1307                (-4.0 0.0169197 -0.310822)
1308                (-3.0 0.00904798 -0.206675)
1309                (-2.0 0.00696255 -0.102527)
1310                (-1.0 0.0054272 0.00162034)
1311                (0.0 0.00526348 0.105768)
1312                (1.0 0.0054272 0.209915)
1313                (2.0 0.00696255 0.314063)
1314                (3.0 0.00904798 0.409977)
1315                (4.0 0.0169197 0.505891)
1316                (5.0 0.0246517 0.600019)
1317                (6.0 0.0346883 0.694147)
1318                (7.0 0.0418421 0.761196)
1319                (8.0 0.0489959 0.828244)
1320                (180.0 0.0489959 0.828244)
1321            );
1322        }
1323        sect9
1324        {
1325            type lookup;
1326            data
1327            (
1328                (-180.0 0.0485209 -0.729127)
1329                (-8.0 0.0485209 -0.729127)
1330                (-7.0 0.041448 -0.624742)
1331                (-6.0 0.034375 -0.520358)
1332                (-5.0 0.0243894 -0.415973)
1333                (-4.0 0.0167629 -0.311589)
1334                (-3.0 0.00905368 -0.207204)
1335                (-2.0 0.00694523 -0.10282)
1336                (-1.0 0.00543213 0.00156492)
1337                (0.0 0.00526484 0.105949)
1338                (1.0 0.00543213 0.210334)
1339                (2.0 0.00694523 0.314718)
1340                (3.0 0.00905368 0.410883)
1341                (4.0 0.0167629 0.507048)
1342                (5.0 0.0243894 0.601655)
1343                (6.0 0.034375 0.696263)
1344                (7.0 0.041448 0.763274)
1345                (8.0 0.0485209 0.830286)
1346                (180.0 0.0485209 0.830286)
1347            );
1348        }
1349        sect10
1350        {
1351            type lookup;
```

```
            data
            (
                (-180.0 0.048066 -0.730699)
                (-8.0 0.048066 -0.730699)
                (-7.0 0.0410658 -0.626092)
                (-6.0 0.0340656 -0.521485)
                (-5.0 0.0241393 -0.416878)
                (-4.0 0.0166182 -0.312271)
                (-3.0 0.00906466 -0.207664)
                (-2.0 0.00692768 -0.103057)
                (-1.0 0.00543701 0.00154987)
                (0.0 0.0052662 0.106157)
                (1.0 0.00543701 0.210764)
                (2.0 0.00692768 0.315371)
                (3.0 0.00906466 0.411898)
                (4.0 0.0166182 0.508425)
                (5.0 0.0241393 0.603397)
                (6.0 0.0340656 0.698368)
                (7.0 0.0410658 0.765342)
                (8.0 0.048066 0.832316)
                (180.0 0.048066 0.832316)
            );
        }
        sect11
        {
            type lookup;
            data
            (
                (-180.0 0.0476121 -0.729281)
                (-8.0 0.0476121 -0.729281)
                (-7.0 0.0406835 -0.624751)
                (-6.0 0.033755 -0.520221)
                (-5.0 0.0238874 -0.415691)
                (-4.0 0.0164692 -0.311161)
                (-3.0 0.00907155 -0.206631)
                (-2.0 0.00690994 -0.102101)
                (-1.0 0.00544582 0.00242845)
                (0.0 0.00526765 0.106958)
                (1.0 0.00544582 0.211488)
                (2.0 0.00690994 0.316018)
                (3.0 0.00907155 0.412906)
                (4.0 0.0164692 0.509793)
                (5.0 0.0238874 0.605125)
                (6.0 0.033755 0.700457)
                (7.0 0.0406835 0.767394)
                (8.0 0.0476121 0.834332)
                (180.0 0.0476121 0.834332)
            );
        }
        sect12
        {
            type lookup;
            data
            (
```

```
            (-180.0 0.0471585 -0.72759)
            (-8.0 0.0471585 -0.72759)
            (-7.0 0.0403008 -0.623165)
            (-6.0 0.033443 -0.51874)
            (-5.0 0.0236344 -0.414315)
            (-4.0 0.0163166 -0.30989)
            (-3.0 0.00907484 -0.205465)
            (-2.0 0.00689231 -0.10104)
            (-1.0 0.00545459 0.00338444)
            (0.0 0.0052691 0.107809)
            (1.0 0.00545459 0.212234)
            (2.0 0.00689231 0.316659)
            (3.0 0.00907484 0.413903)
            (4.0 0.0163166 0.511147)
            (5.0 0.0236344 0.606836)
            (6.0 0.033443 0.702525)
            (7.0 0.0403008 0.769426)
            (8.0 0.0471585 0.836327)
            (180.0 0.0471585 0.836327)
        );
    }
    sect13
    {
        type lookup;
        data
        (
            (-180.0 0.0466766 -0.725904)
            (-8.0 0.0466766 -0.725904)
            (-7.0 0.0398971 -0.621584)
            (-6.0 0.0331177 -0.517265)
            (-5.0 0.02338 -0.412945)
            (-4.0 0.0161628 -0.308626)
            (-3.0 0.00907862 -0.204306)
            (-2.0 0.00687725 -0.0999862)
            (-1.0 0.00546324 0.00433343)
            (0.0 0.00527054 0.108653)
            (1.0 0.00546324 0.212973)
            (2.0 0.00687725 0.317292)
            (3.0 0.00907862 0.414888)
            (4.0 0.0161628 0.512484)
            (5.0 0.02338 0.608526)
            (6.0 0.0331177 0.704568)
            (7.0 0.0398971 0.771433)
            (8.0 0.0466766 0.838298)
            (180.0 0.0466766 0.838298)
        );
    }
    sect14
    {
        type lookup;
        data
        (
            (-180.0 0.0460879 -0.724242)
            (-8.0 0.0460879 -0.724242)
```

```
            (-7.0 0.0394174 -0.620026)
            (-6.0 0.032747 -0.51581)
            (-5.0 0.0231313 -0.411594)
            (-4.0 0.0160213 -0.307378)
            (-3.0 0.00908687 -0.203163)
            (-2.0 0.00686261 -0.0989467)
            (-1.0 0.00547151 0.00526919)
            (0.0 0.00527195 0.109485)
            (1.0 0.00547151 0.213701)
            (2.0 0.00686261 0.317917)
            (3.0 0.00908687 0.415965)
            (4.0 0.0160213 0.514014)
            (5.0 0.0231313 0.610299)
            (6.0 0.032747 0.706583)
            (7.0 0.0394174 0.773376)
            (8.0 0.0460879 0.840169)
            (180.0 0.0460879 0.840169)
        );
    }
    sect15
    {
        type lookup;
        data
        (
            (-180.0 0.0454611 -0.722606)
            (-8.0 0.0454611 -0.722606)
            (-7.0 0.0389106 -0.618492)
            (-6.0 0.0323601 -0.514378)
            (-5.0 0.0228842 -0.410265)
            (-4.0 0.0158828 -0.306151)
            (-3.0 0.00909425 -0.202037)
            (-2.0 0.00684796 -0.0979235)
            (-1.0 0.00547936 0.00619023)
            (0.0 0.00527334 0.110304)
            (1.0 0.00547936 0.214418)
            (2.0 0.00684796 0.318531)
            (3.0 0.00909425 0.417069)
            (4.0 0.0158828 0.515607)
            (5.0 0.0228842 0.612087)
            (6.0 0.0323601 0.708567)
            (7.0 0.0389106 0.77526)
            (8.0 0.0454611 0.841953)
            (180.0 0.0454611 0.841953)
        );
    }
    sect16
    {
        type lookup;
        data
        (
            (-180.0 0.0448389 -0.720998)
            (-8.0 0.0448389 -0.720998)
            (-7.0 0.0384051 -0.616985)
            (-6.0 0.0319712 -0.512972)
```

```
                (-5.0 0.0226305 -0.408958)
                (-4.0 0.015738 -0.304945)
                (-3.0 0.0090973 -0.200932)
                (-2.0 0.00683333 -0.0969182)
                (-1.0 0.0054868 0.0070952)
                (0.0 0.00527471 0.111109)
                (1.0 0.0054868 0.215122)
                (2.0 0.00683333 0.319135)
                (3.0 0.0090973 0.418157)
                (4.0 0.015738 0.517179)
                (5.0 0.0226305 0.613847)
                (6.0 0.0319712 0.710515)
                (7.0 0.0384051 0.7771)
                (8.0 0.0448389 0.843684)
                (180.0 0.0448389 0.843684)
            );
        }
        sect17
        {
            type lookup;
            data
            (
                (-180.0 0.0442256 -0.719638)
                (-8.0 0.0442256 -0.719638)
                (-7.0 0.037907 -0.615697)
                (-6.0 0.0315883 -0.511756)
                (-5.0 0.0223721 -0.407815)
                (-4.0 0.0155852 -0.303874)
                (-3.0 0.00909721 -0.199934)
                (-2.0 0.0068193 -0.0959926)
                (-1.0 0.00549405 0.0079484)
                (0.0 0.00527604 0.111889)
                (1.0 0.00549405 0.21583)
                (2.0 0.0068193 0.319771)
                (3.0 0.00909721 0.419249)
                (4.0 0.0155852 0.518726)
                (5.0 0.0223721 0.615623)
                (6.0 0.0315883 0.71252)
                (7.0 0.037907 0.778951)
                (8.0 0.0442256 0.845382)
                (180.0 0.0442256 0.845382)
            );
        }
        sect18
        {
            type lookup;
            data
            (
                (-180.0 0.0436214 -0.719046)
                (-8.0 0.0436214 -0.719046)
                (-7.0 0.0374273 -0.615087)
                (-6.0 0.0312333 -0.511128)
                (-5.0 0.0221243 -0.407169)
                (-4.0 0.0154286 -0.30321)
```

```
1568                (-3.0 0.00909833 -0.199252)
1569                (-2.0 0.00680721 -0.0952927)
1570                (-1.0 0.00550166 0.00866623)
1571                (0.0 0.00527735 0.112625)
1572                (1.0 0.00550166 0.216584)
1573                (2.0 0.00680721 0.320543)
1574                (3.0 0.00909833 0.420402)
1575                (4.0 0.0154286 0.520261)
1576                (5.0 0.0221243 0.617519)
1577                (6.0 0.0312333 0.714776)
1578                (7.0 0.0374273 0.780911)
1579                (8.0 0.0436214 0.847045)
1580                (180.0 0.0436214 0.847045)
1581            );
1582        }
1583        sect19
1584        {
1585            type lookup;
1586            data
1587            (
1588                (-180.0 0.043025 -0.720867)
1589                (-8.0 0.043025 -0.720867)
1590                (-7.0 0.036952 -0.616603)
1591                (-6.0 0.0308791 -0.512338)
1592                (-5.0 0.0218817 -0.408073)
1593                (-4.0 0.0152767 -0.303809)
1594                (-3.0 0.00911313 -0.199544)
1595                (-2.0 0.00680099 -0.0952794)
1596                (-1.0 0.00551115 0.00898521)
1597                (0.0 0.00527861 0.11325)
1598                (1.0 0.00551115 0.217514)
1599                (2.0 0.00680099 0.321779)
1600                (3.0 0.00911313 0.421833)
1601                (4.0 0.0152767 0.521887)
1602                (5.0 0.0218817 0.619438)
1603                (6.0 0.0308791 0.716989)
1604                (7.0 0.036952 0.78283)
1605                (8.0 0.043025 0.848671)
1606                (180.0 0.043025 0.848671)
1607            );
1608        }
1609    }
1610 }
1611
1612 // ************************************************************************* //
```