OpenFOAM® Journal

Volume 2 [16th OpenFOAM® Workshop - 2021], Pages 81–93
ISSN: 2753-8168

A COUPLED ACTUATOR LINE AND FINITE ELEMENT ANALYSIS TOOL

PÁL SCHMITT AND DESMOND ROBINSON

Queen's University, Marine Laboratory, 12-13 The Strand, Portaferry, BT22 1PF, United Kingdom
Email address: p.schmitt@qub.ac.uk

Queen's University, David Keir Building, Stranmillis Road, Belfast, BT9 5AG, United Kingdom
Email address: des.robinson@qub.ac.uk

DOI: 10.51560/ofj.v2.51
Version(s): OpenFOAM® 8
Repo: https://github.com/palschmitt/turbinesFoam

Abstract. Fluid-dynamic loading of many solid bodies can be simulated using geometry resolving computational fluid dynamics methods, where the body's shape is resolved in the mesh. In some cases though slender bodies, like ropes or cables, spars, turbine blades, foils and lattice structures would require prohibitively high cell counts, since the geometrical features to be resolve are much smaller than the overall domain. Such bodies are usually made up of generic cross sections like round, square or standardised technical profiles like the famous NACA digit series for which good parametrisations of reaction forces to incoming flow exist. Actuator line methods apply inflow dependent reaction forces to the fluid domain, thus allowing the computationally efficient simulation of slender bodies and have been used extensively, for example in wind and tidal turbine simulations. Beam elements representing slender bodies are standard building blocks in structural finite element models. Combining actuator line theory with a finite element beam model allows the efficient simulation of flexible structures under fluid loads, like turbine blades or nettings used in fish farms. This paper presents an implementation of such a coupled model in OpenFOAM based on the existing turbineFoam actuator line model. The underlying numerical method is detailed and first test cases are provided.

1. Introduction

With sufficient spatial and temporal resolution even the most complex flow effects like flow separation and highly turbulent flows can be resolved using computational fluid dynamics methods (CFD) [1].

However, for practical design purposes, computational resources are limited and simplifications are still needed. Computational domains in marine engineering typically extend over several wave lengths or the water depth, and thus often range over 100s of metres. Challenges arise especially if details of the structure of interest are orders of magnitude smaller than the domain, as is frequently the case for lattice structures and foils, for example for tidal turbines, mooring systems, fish farms and wave energy converters.

Correctly resolving a NACA0012 profile in all operating conditions at a Reynolds Number of 6 million requires roughly 0.5 million cells in only 2 dimensions [2]. For most industrially relevant three dimensional problems the same level of resolution results in a very high cell count.

Slender structures have been studied extensively and parametric descriptions or lookup tables exist for most features like lift or drag coefficients. Actuator Line Models (AL) use such parametric descriptions of profiles to apply the resulting force to the flow field, without representing the structure in the mesh [3, 4, 5]. AL simplify the meshing, since only the fluid domain needs to be created. The resulting lower mesh size can reduce the computational burden and thus allows to focus on resolving far field effects like wakes in wind and tidal turbines [6, 7, 8, 9, 10]. Applications to other marine structures are still limited, one interesting exception is the work on the underwater kite system by Fredriksson et al. [11].

An interesting feature of slender bodies is their structural response. Hydrofoils are sensitive to the angle of attack (AoA) and thus torsional deformation. Mooring lines, nets and lattice frameworks can deform significantly, especially in dense fluids. Wind turbines are sometimes designed to deform into their ideal shape under wind loading. Tidal as well as wind turbines have been built to shed excessive loads by bend-twist coupling, see Nicholls-Lee et al. [12].

* Corresponding author
Received: 17 September 2021, Accepted: 11 April 2022, Published: 2 May 2022

This paper describes a first version of `ALFEA`, implementing the coupling of an AL toolbox with Finite Element Analysis (FEA) to enable the simulation of fluid structure interaction of arbitrary slender structures. The focus in this paper lies on marine engineering applications. However, the methods should be applicable to a much wider range of physical problems. Similar implementations like Sale et al. [13], Neumann [14] are limited to turbine blades, do not seem to be maintained any more or are not available as open source code.

The toolbox is based on OpenFOAM® and extends the AL method originally implemented by Bachant et al. [5] with a custom written FEA model to take structural deformations into account.

The paper is structured as follows. The following section 2 provides an overview of the FEA module. Section 3 details the coupling with the AL method. A further section 4 provides details on test cases. Conclusions are drawn and an outlook on future applications in section 5.

2. Frame Analysis Implementation

A wide range of open source FEA methods are available and could be used for coupling. However, most FEA methods offer many more features than could possibly be used in this application and are not written in `C++`. To ease maintenance of the code base and integration with OpenFOAM® , it was decided to write a stand alone `C++` library with only the required functions.

This section provides a basic overview of the code design and implementation. The method used closely follows the formulation in Przemieniecki [15]. Beam elements are based on 2 nodes describing the end positions of each element and can evaluate bending along and torsion around their length axis and extension/compression due to normal forces according to linear theory. The current implementation employs a steady state formulation.

A first implementation in GNU octave [16] was used for initial development and validation. In a second step the code was reimplemented in `C++` making use of the Armadillo library for matrix manipulation [17]. OpenFOAM® [18] offers its own syntax for matrix operations but since the Armadillo library was specifically written to provide a similar interface to GNU octave, its use simplified the reimplementation of the code signficantly. The entire frame analysis class definition consists of less than 800 lines of code.

All frame analysis code is located in a new folder called `FEA`, located inside the original turbinesFOAM library inside the `src` folder.

The FEA mesh is defined by lists of node locations and connections. The constructor of the novel `FrameAnalysis` class calls the `beam1()` function, which iterates over all elements and by calling the `elemstiff()` function creates the respective element matrices K_e in local coordinates.

$$K_e = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\
0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & \frac{I_P G}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{I_P G}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\
-\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\
0 & 0 & -\frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\
0 & 0 & 0 & -\frac{I_P G}{L} & 0 & 0 & 0 & 0 & 0 & \frac{I_P G}{L} & 0 & 0 \\
0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\
0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L}
\end{bmatrix} \tag{2.1}$$

Here E is the Young modulus, A the cross section area, L the length, I_z and I_y the cross section inertias, $G = \frac{E}{2(1+P)}$ the shear modulus with P the Poisson coefficient.

`elemstiff()` also creates the transformation matrices required to transform from local (element) to global coordinates. The function call `assem()` then assembles the overall system matrix K in global coordinates. K is then reordered by the function `neworder()` according to the boundary conditions provided, yielding separate submatrices for free ($_\mathfrak{f}$) and restrained ($_\mathfrak{r}$) nodes, with u being the deformations and F the forces:

$$\begin{bmatrix} K_{\mathfrak{f}\mathfrak{f}} & K_{\mathfrak{r}\mathfrak{f}} \\ K_{\mathfrak{r}\mathfrak{f}} & K_{\mathfrak{r}\mathfrak{r}} \end{bmatrix} \begin{bmatrix} u_\mathfrak{f} \\ u_\mathfrak{r} \end{bmatrix} = \begin{bmatrix} F_\mathfrak{f} \\ F_\mathfrak{r} \end{bmatrix} \tag{2.2}$$

Algorithm 1 provides a pseudo code to help understand the code structure.

The armadillo library's `solve()` function is then used to obtain the deformations at the free nodes by solving

$$u_\mathfrak{f} = K_{\mathfrak{f}\mathfrak{f}}^{-1}(F_\mathfrak{f} - K_{\mathfrak{f}\mathfrak{r}} u_\mathfrak{r}) \tag{2.3}$$

and the reaction forces are then obtained by evaluating

$$F_\mathfrak{r} = K_{\mathfrak{r}\mathfrak{f}} u_\mathfrak{f} + K_{\mathfrak{r}\mathfrak{r}} u_\mathfrak{r} \tag{2.4}$$

Algorithm 1 FrameAnalysis Workflow

Import nodes, elements..
for all elements **do**
  `elemstiff()` create element stiffness and transformation matrix K_e and $RotMat$
  `assem()` Assemble system stiffness matrix K
end for
`neworder()` find free and restrained nodes
`reorder()` Partition K according to free and restrained nodes, such that $\begin{bmatrix} K_{\mathfrak{f}\mathfrak{f}} & K_{\mathfrak{r}\mathfrak{f}} \\ K_{\mathfrak{r}\mathfrak{f}} & K_{\mathfrak{r}\mathfrak{r}} \end{bmatrix} \begin{bmatrix} u_f \\ u_\mathfrak{r} \end{bmatrix} = \begin{bmatrix} F_\mathfrak{f} \\ F_\mathfrak{r} \end{bmatrix}$
`loading()` Partition system force F into $F_\mathfrak{f}$ and $F_\mathfrak{r}$
`predisp()` Partition system deformation u into $u_\mathfrak{r}$ and u_f
solve $u_f = K_{\mathfrak{f}\mathfrak{f}}^{-1}(F_\mathfrak{f} - K_{\mathfrak{f}\mathfrak{r}}u_\mathfrak{r})$
solve $F_\mathfrak{r} = K_{\mathfrak{r}\mathfrak{f}}u_\mathfrak{f} + K_{\mathfrak{r}\mathfrak{r}}u_\mathfrak{r}$
`nodaldisp()` Rearrange deformation according to input

The armadillo library employs its own matrix type definition which differs from the lists used in the AL model. Helper functions to convert lists of floats or integers into armadillo matrices were implemented and are called `List2Mat()` and `List2intMat()`. The function `Mat2List()` allows to reconvert data from armadillo matrices to lists.

3. Coupling with AL model

Details and first validation cases of the AL method are provided by Bachant et al. [5]. This section focusses on the coupling methodology.

As shown in the schematic in Fig 1, forces are evaluated at the midpoint of each AL element, while in the FEA model forces and restrains must be defined at the nodes or endpoints of each element. The FEA mesh is thus generated with twice as many elements as the AL definition. Forces evaluated by the AL for a given node position and orientation are passed to the FEA model. These fluid forces are then applied to every second node in the FEA mesh. The use of equal numbers of FEA and AL elements would also be possible, but require the evaluation of equivalent loads and thus a significant increase in the codebase and runtime. While it is unclear whether the additional operations would outweigh the benefit of a smaller FEA system matrix, it should be noted that in typical application scenarios the FEA matrix is expected to be orders of magnitude smaller than the matrices involved in the fluid solver. The FEA simulation are thus not believed to be a bottleneck for the overall simulation and efficiency considerations less relevant.

Resulting deformations from the FEA run are then applied to the AL method by updating the node positions and re-evaluating the AL elements' centre pitch and fluid force properties. Restraints can only be defined for the first or last element of a section and are applied at the first or last node respectively.

All required FEA inputs are defined through the AL model definition in `system/fvOptions` as shown in listing 1 which was extended to allow for the definition of material and geometrical properties like the Young modulus, section area/inertias and restraints.

```
1
2 \\point spanDir chordLength chordRefDir chordMount pitch
3 E nu sects A        Iz       Iy          J        alpha  restraints
4 (
5 ((0 0 0) (0 0 1) (0.10) (1 0 0) (0.25) (10.0)
6 (11E9  0.2) (0.003 7.E-06 7.E-06 0.02  0) (1 1 1  1 1 1))
7 ((0 0 1) (0 0 1) (0.1) (1 0 0) (0.25) (10.0)
8 (11E9  0.2) (0.003 7.E-06 7.E-06 0.02  0) (0 0 0 0 0 0))
9 );
```

Listing 1. Example FEA parameter defintion in the AL settings

Restraints are defined as a list of six values, set to either 1 for fixed or 0 for a free degree of deformation. The first three values refer to translation in x, y and z direction and the last three to the respective rotations.

The original AL model interpolates section values linearly along a geometric section for any wanted number of elements, this same interpolation is also applied to FEA properties except the restraints, which are only applied to the first or last node of each section.

The AL evaluates source terms for the momentum equation. The force acting on the flow is thus applied as a discrete source term every time the velocity equation is solved. For each AL element, the

last available flow field is used to evaluate parameters like the AoA and velocity magnitude. Each actuator line element also saves the applied forces and moments. The differential between new and current time step is applied in the FEA model, such that a constant load in time will not lead to further deformation after finding the equilibrium state. The FEA model is solved at every new time step and the positions and rotations returned to the AL model. The AL then updates element positions and profile pitch is adjusted by the negative cross product of the elements' span direction and FEA rotation vector. A high level pseudo code of the algorithm is provided in Algorithm 2.

Algorithm 2 Coupling between AL and FEA model

```
for Timesteps do
    Solve fluid field with AL source terms
    Obtain change in fluid forces since last time step
    Create FEA mesh from AL positions
    Evaluate deformation of structure
    Update AL positions and orientations
end for
```

4. Test Cases

AL methods have been tested extensively for a wide range of applications [5, 3, 19, 20, 11], the test cases presented here thus focus on the FEA part.

The fluid domain is identical for all cases and based on the original `static` actuatorLine tutorial case from the turbinesFoam library. In all cases the pimpleFoam solver was used with OpenFOAM® -8 in transient mode with 2 outer correctors. The `kEpsilon` turbulence model was employed and cases were run for at least 2 seconds. Resulting deformations are always presented for the last timestep unless explicitly defined differently. A steady-state simulation might save some computational effort and would also have sufficed for demonstrating the steady state FEA implementation. However, to ease testing for users already familiar with the AL methods and because we plan to extend the method to transient cases, we chose to keep the fluid solver settings of the test cases and setup unchanged.

The domain is a cube with a bounding box with extreme positions (-1.52, -1.83, -1.22) and (2.16, 1.83, 1.22). The basic mesh is refined with 32, 32 and 24 cells in x, y and z direction respectively. `snappyHexMesh` is used to create a zone with refinementLevel 2 within a cylinder with 0.5 m radius between points (0 0 -1.5) to (0 0 1.5). An example of the resulting mesh is shown in Fig 2. Boundary conditions used in the fluid simulation are presented in Table 1, numerical values used are given in listing 2.

4.1. Test Case 1: Bending.

A first test was devised to evaluate deformation due to bending. Case files can be found under `turbinesFoam/tutorials/actuatorLine/bendingtestFEA`. A blade profile of $L = 2\,m$ length is fixed in rotation and deflection in all directions at the position (0,0,-1) and points in z direction as shown in Fig 3. A flow of $v = 1\,m/s$ in x direction acts on the profile at an AoA of 10° for which the coefficient of lift C_L is 0.8.

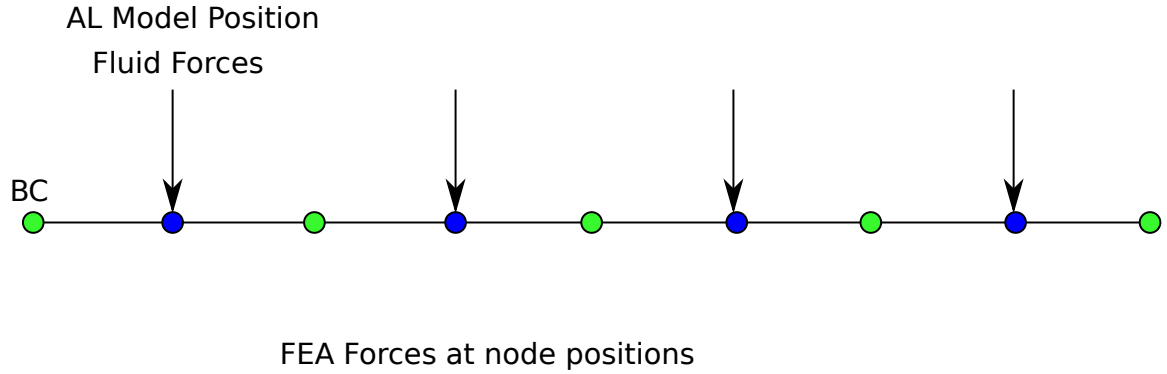

Figure 1. Schematic representation of the data structures used in ALFEA. Boundary conditions (BC) are applied to first and/or last nodes of section. Fluid forces (black arrows) are applied from AL element positions to every second FEA node.

```
flowVelocity          (1 0 0)
pressure              0;
turbulentKE           2e-4;
turbulentEpsilon      3e-5;
```

LISTING 2. Numerical values used in boundary and initial conditions

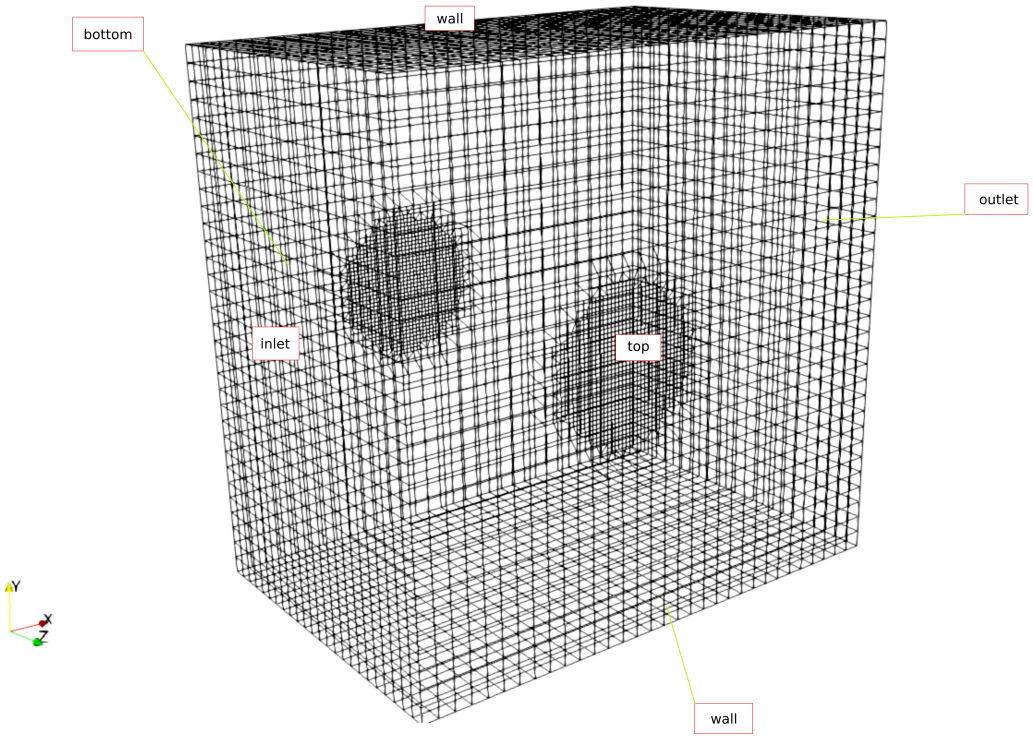

FIGURE 2. Layout of the computational domain with patch names, fluid mesh resolution and main dimensions.

TABLE 1. Boundary Conditions used in fluid solver. (WF stands for WallFunction.)

Patch	p	U	k	nut	epsilon
inlet	zeroGradient	fixedValue	uniformFixedValue	calculated	fixedValue
outlet	fixedValue	inletOutlet	inletOutlet	calculated	inletOutlet
top	zeroGradient	fixedValue	kqRWF	nutkWF	epsilonWF
bottom	zeroGradient	fixedValue	kqRWF	nutkWF	epsilonWF
wall	zeroGradient	fixedValue	kqRWF	nutkWF	epsilonWF

Further input data is provided in table 2. The given parameters result in a continuous load of

$$q = \frac{v^2}{2} C_L c \rho = 40\, N/m \tag{4.1}$$

and the analytical solution for deflection δ under constant load is [21]

$$\delta(x) = \frac{qL^4}{24EI_z}\left(\frac{x^4}{L^4} - 4\frac{x}{L} + 3\right); \tag{4.2}$$

End effect models, used to correct for the loss in lift close to wing or blade tips were not used. Although unphysical this setting allows better comparison with a simple analytical load model of constant load along the beam.

Fig 4 shows element positions from the ALFEA model in direct comparison to results from the analytical model, Eq 4.2. Agreement is good but minor differences for the biggest deflection at the free end of the wing can be observed. It was found that fluid forces acting on the wing do not exactly agree with

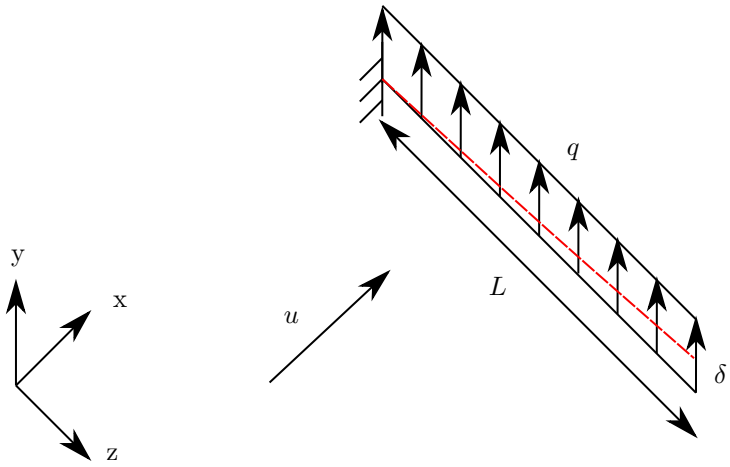

FIGURE 3. Schematic representation of the bending test case. The red line indicates the deformed shape of the beam, caused by the continuous load $q = \frac{v^2}{2} C_L c \rho$

the analytical solution. It should be noted though, that the analytical solution does not take the three dimensional effects around the blade ends into account. Applying the overall forces obtained from the ALFEA model as a distributed load $q' = F/L$ in the analytical equation 4.2 is shown as Eq_{Sim} in Fig 4 and yields excellent agreement.

4.2. **Test Case 2: Torsion.** A second test case aimed at evaluating torsional deformation can be found under `tutorials/actuatorLine/torsiontestFEA/`. As shown in Fig 5, a wing is now supported by a

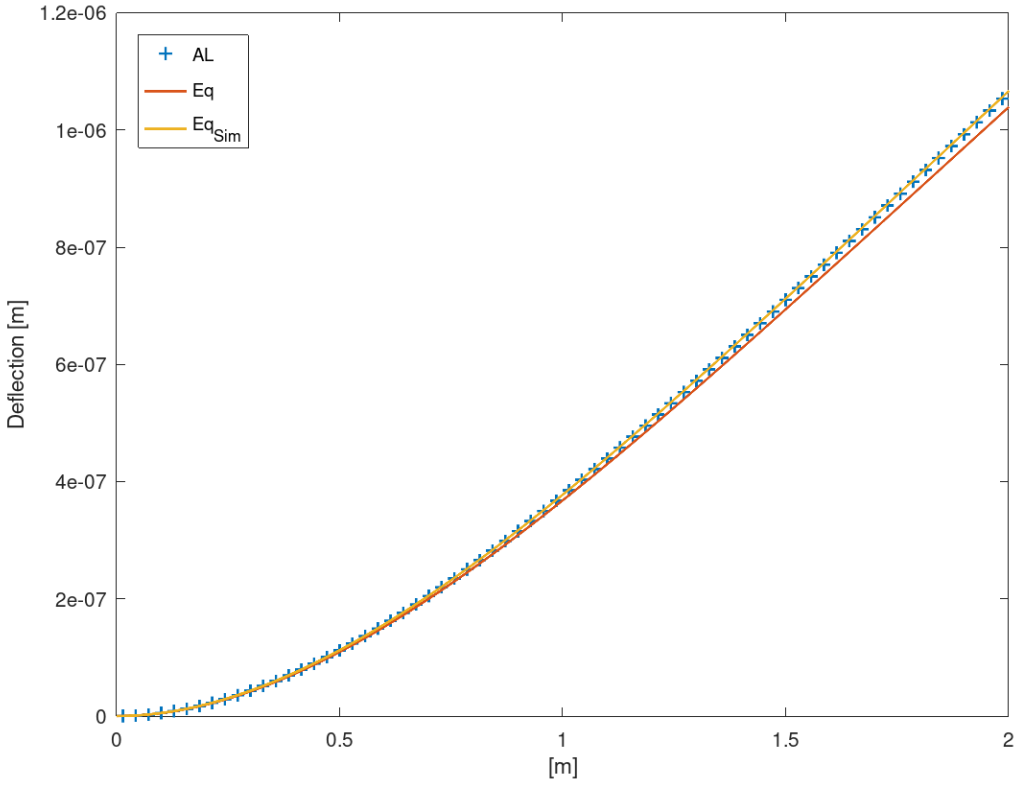

FIGURE 4. Results for the deflection along the blade section according to the ALFEA model (AL), analytical solution (Eq) and analytical solution using loads from ALFEA model (Eq_{Sim}).

TABLE 2. Input Data for Bending Test Case

Symbol	Meaning	Value	Unit
L	Length	2	m/s
C_L	Lift Coefficient	0.8	—
c	Chordlength	0.1	m
v	Velocity	1	m/s
ν	Viscosity	1E-6	m^2/s
ρ	Density	1	kg/m^3
Re	Reynolds Number	1E5	—
E	Young modulus	11E9	N/m^2
P	Poisson Coefficient	0.2	—
A	Cross Section Area	0.003	m^2
I_x, I_z	Inertia	7.E-6	m^4
I_p	Polar Inertia	0.05E-6	m^4
q	Continuous Load	40	N/m

beam of length L_S in x direction, such that the lift force on the wing will create a torsional moment [21]

$$M_t = qL^2/2 \tag{4.3}$$

and a deformation angle of

$$\phi = \frac{M_t L_S 2(1+P)}{I_{S,p}E} \tag{4.4}$$

Variable names and values used are given in Table 3

Table 4 lists resulting deformation angles from Eq 4.4 and the ALFEA simulation (AL). Eq 4.4 yields to 5.5% less deformation than the simulation. The cause for this deviation is again the difference between analytically derived force and ALFEA results. Results obtained using Eq 4.4 with the loads from the ALFEA model (Eq_{Sim}) yield excellent agreement with the ALFEA simulation.

4.3. **Test Case 3: Changing angle of attack by a deforming substructure.** A last test scenario demonstrates the change of AoA by the deformation of the supporting structure. Two cases are simulated. Case 1 simulates a deforming support structure with the inertia of the supporting beam set to $7.E{-}11\ m^4$. Case 2 simulates a virtually rigid support structure with the inertia of the supporting beams set to $7.E{-}6\ m^4$. The two simulations can be found under `tutorials/actuatorLine/changeAoA` and `fixedAoA` respectively. Key input parameters of the cases are listed in Table 5.

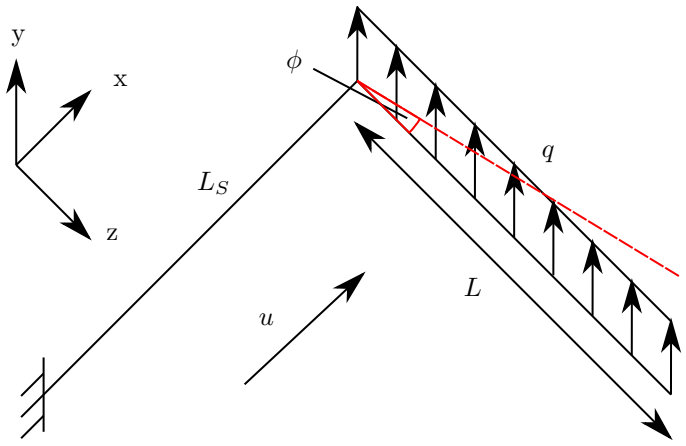

FIGURE 5. Schematic representation of the torsion test case. The red line indicates the rotated position of the beam, the red arc the deformation angle ϕ.

TABLE 3. Additional/Amended Input Data for Torsion Case

Symbol	Meaning	Value	Unit
L	Length	1	m/s
I_x, I_z	Inertia	7.E-6	m^4
L_S	Length Support Beam	0.9	m/s
$I_{S,p}$	Polar Inertia Support Beam	0.05E-6	m^4
c_S	Chord Length Support Beam	0.01	m

TABLE 4. Absolute torsional deformation and percentage error compared to ALFEA results

Case	Value [°]	% Error
ALFEA	0.00477	0.0
Eq 4.4	0.00450	-5.6
Eq_{Sim}	0.00478	0.25

A profile of length L is supported by two supporting beams of lengths L_S as shown in Fig 6.

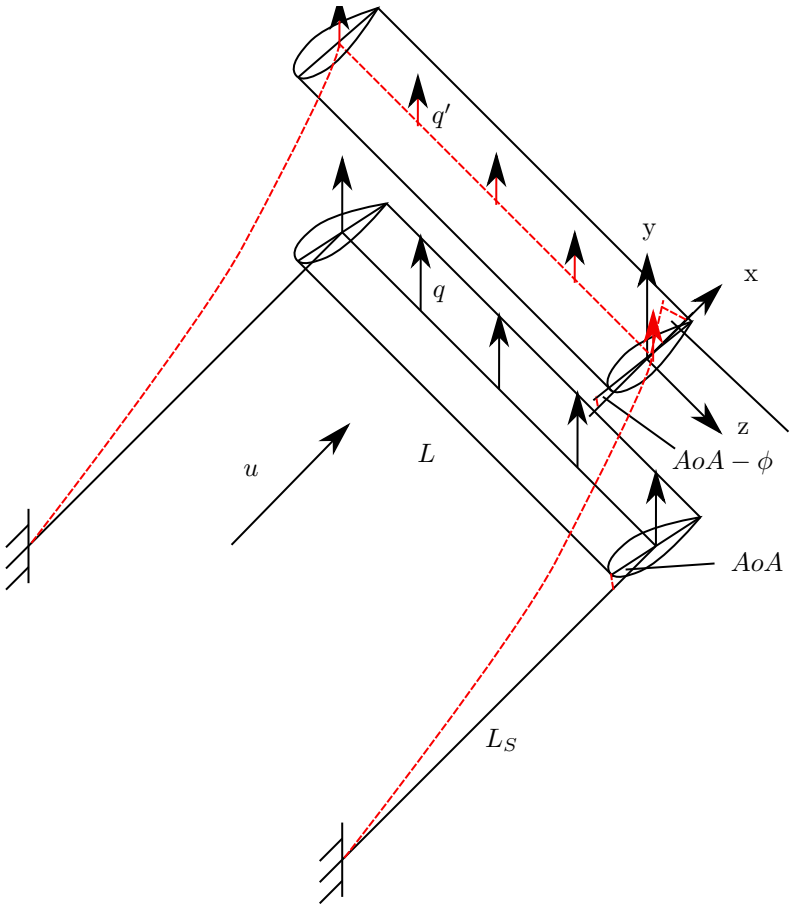

FIGURE 6. Schematic representation of the changing AoA test case. The red line indicates the deformed support beams. As the beam ends are lifted in y direction and rotate by the angle ϕ around the z axis, the AoA of the foil reduces by the same amount, decreasing the lift force q to q'.

TABLE 5. Additional/Amended Input Data for AoA Case

Symbol	Meaning	Value Case 1	Value Case 2	Unit
L	Length	1	1	m/s
I_x, I_z	Inertia Wing	7.E-6	7.E-6	m^4
L_S	Length	1.5	1.5	m/s
$I_{s,x}, I_{s,z}$	Inertia of Support	7.E-11	7.E-6	m^4

The upward force

$$F = \frac{v^2 C_l c L}{2} \tag{4.5}$$

causes the deformation $\delta(x)$ along the length of the supporting beams

$$2\delta(x) = \frac{F L_S^3}{6 E I_z}\left(2 - \frac{3x}{L} + \frac{x^3}{L^3}\right) \tag{4.6}$$

and a deflection angle at the profile position [21]

$$\tan 2\theta = \frac{F L_S^2}{2 E I_z} \tag{4.7}$$

The factor 2 in front of the angle θ and $\delta(x)$ accounts for the number of supporting beams.

For a rigid structure the AoA remains constant and the C_L only varies during the transient startup while the flow field develops. The soft support structure causes a reduction of AoA due to the deflection angle and thus limits the loading.

Fig 7 shows the deflection of the beam under loading equivalent to the undeformed structure, with an AoA of 10° (Eq) according to Eq 4.6. Deformation by a load corresponding to a reduction in C_L by this maximum deformation is shown as Eq_{Iter} and yields much lower deflections. ALFEA simulations (AL) yield to the mean deformation of those extreme assumptions. Eq_{Sim}, obtained using the loading from the ALFEA model show excellent agreement with the simulation results.

Fig 8 shows AoA and C_L for Case 1 and Case 2 over time. The inital AoA is 10° and the corresponding C_L 0.8. The AoA for Case 2 remains constant over time and C_L settles after 2 seconds at a value of 0.85. Case 1 shows a drop in AoA to 8.3°. The C_L also settles at a lower value of 0.725.

5. Conclusions

This paper presents details on a first version of the ALFEA toolbox. Coupling of AL and FEA methods now enables the efficient simulation of flexible slender structures in OpenFOAM® . Details on the implementation are provided. Two test cases demonstrate correct application of the forces evaluated by the AL model to the newly implemented FEA toolbox for bending and torsional deformation. A third test case demonstrates interactions affecting the AoA and the change of fluid loading caused by the structure's deformation.

Acknowledgements

Pál Schmitt was supported by the Bryden Centre. a project supported by the European Union's INTERREG VA Programme, managed by the Special EU Programmes Body (SEUPB).

Disclaimer

The views and opinions expressed in this paper do not necessarily reflect those of the European Commission or the Special EU Programmes Body (SEUPB).

Author Contributions: Conceptualisation, P.S. and D.R.; methodology, P.S. and D.R.; software, P.S. and D.R. ; validation, P.S. and P.M.; formal analysis, P.S. and D.R.; investigation, P.S.; resources, P.S.; data curation, P.S.; writing—original draft preparation, P.S. and D.R.; writing—review and editing, P.S.; visualisation, P.S.; supervision, P.S.; project administration, P.S. All authors have read and agreed to the published version of the manuscript.

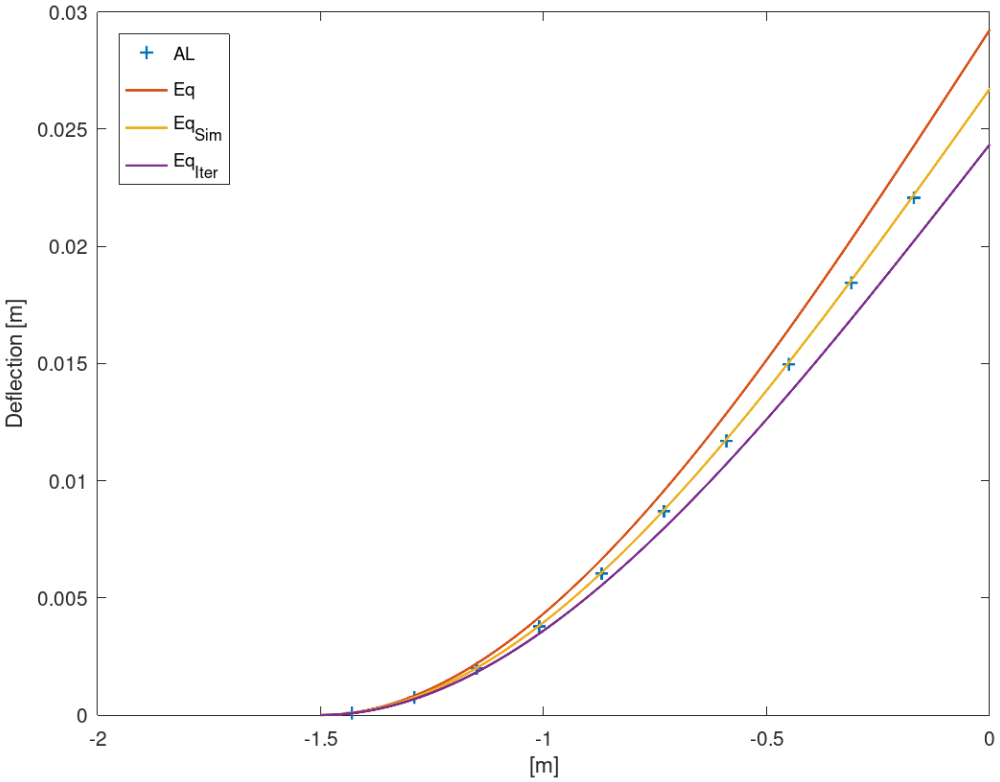

FIGURE 7. Resulting deflection along the supporting beam according to simulation (AL) and three analytical solutions. Eq is the result assuming no deformation and using a C_L corresponding to the AoA of the undeformed structure at 10°. Eq_{Sim} presents results with the load taken from the last simulation step and shows best agreement. Eq_{Iter} assumes a load corresponding to the AoA after deformation by the load of the undeformed beam.

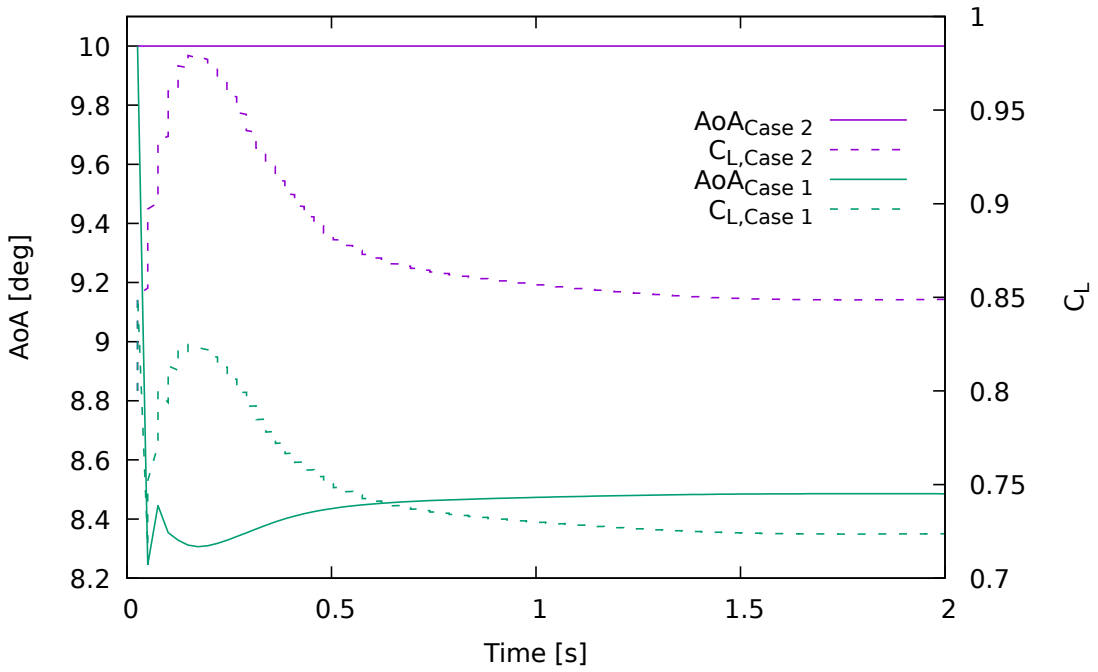

FIGURE 8. Angle of attack (AoA) and C_L for Case 1 and 2 over time. For Case 1 the AoA drops to below 8.3° before recovering to 8.45°, resulting in a C_L of 0.74, while for Case 2 the AoA remains constant at 10° with the C_L settling at 0.85.

References

[1] P. Schmitt and B. Elsaesser, "On the use of OpenFOAM to model oscillating wave surge converters," *Ocean Engineering*, 2015.

[2] C. Rumsey, "The langley research center turbulence modeling resource. 2dn00: 2d naca 0012 airfoil validation case." NASA, Tech. Rep., 2018. [Online]. Available: https://turbmodels.larc.nasa.gov/naca0012_val.html

[3] J. Nathan, A. Meyer Forsting, N. Troldborg, and C. Masson, "Comparison of OpenFOAM and EllipSys3D actuator line methods with (NEW) MEXICO results: Paper," in *Wake Conference 2017*, ser. Journal of Physics: Conference Series, vol. 854. United Kingdom: IOP Publishing, 2017, wake Conference 2017 ; Conference date: 30-05-2017 Through 01-06-2017. [Online]. Available: http://standupforwind.se/wake-conference-2017/

[4] J. Schluntz and R. Willden, "An actuator line method with novel blade flow field coupling based on potential flow equivalence," *Wind Energy*, vol. 18, no. 8, pp. 1469–1485, 2015, cited By 11. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84937163310&doi=10.1002%2fwe.1770&partnerID=40&md5=fe134ad2df4302eddfcd84d7480a2c91

[5] P. Bachant, A. Goude, and M. Wosnik, "Actuator line modeling of vertical-axis turbines," 2018. [Online]. Available: arXiv:1605.01449

[6] M. Baba-Ahmadi and P. Dong, "Numerical simulations of wake characteristics of a horizontal axis tidal stream turbine using actuator line model," *Renewable Energy*, vol. 113, pp. 669–678, 2017, cited By 17. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85020905481&doi=10.1016%2fj.renene.2017.06.035&partnerID=40&md5=162ff1a876edfea8119827dfbfcaf2d0

[7] A. Creech, A. Borthwick, and D. Ingram, "Effects of support structures in anLES actuator line model of a tidal turbine with contra-rotating rotors," *Energies*, vol. 10, no. 5, 2017, cited By 11. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85044269268&doi=10.3390%2fen10050726&partnerID=40&md5=6bb9852464afc780be076882bd389a0a

[8] C. Liu and C. Hu, "CFD simulation of tidal current farm by using al model," *Journal of Hydrodynamics*, vol. 31, no. 1, pp. 34–40, 2019, cited By 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85061774714&doi=10.1007%2fs42241-019-0010-8&partnerID=40&md5=c84efa9d1a900da350a2bf12d9feea77

[9] M. Ravensbergen, A. Bayram Mohamed, and A. Korobenko, "The actuator line method for wind turbine modelling applied in a variational multiscale framework," *Computers & Fluids*, vol. 201, p. 104465, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045793020300384

[10] A. Wimshurst and R. Willden, "Validation of an actuator line method for tidal turbine rotors." International Society of Offshore and Polar Engineers, 2016.

[11] S. T. Fredriksson, G. Broström, B. Bergqvist, J. Lennblad, and H. Nilsson, "Modelling deep green tidal power plant using large eddy simulations and the actuator line method," *Renewable Energy*, vol. 179, pp. 1140–1155, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148121010636

[12] R. Nicholls-Lee, S. Turnock, and S. Boyd, "Application of bend-twist coupled blades for horizontal axis tidal turbines," *Renewable Energy*, vol. 50, pp. 541–550, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148112003941

[13] D. Sale, M. Churchfield, and P. Bachant, "dcsale/sowfa: v1," Jan. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3632051

[14] S. Neumann, "Fluid-structure interaction of flexible lifting bodies with multi-body dynamics of order-reduced models and the actuator-line method," doctoralThesis, Technische Universität Hamburg, 2016. [Online]. Available: https://hdl.handle.net/11420/1288

[15] J. S. Przemieniecki, *Theory of matrix structural analysis*. McGraw-Hill, 1968.

[16] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave version 6.1.0 manual: a high-level interactive language for numerical computations*, 2020. [Online]. Available: https://www.gnu.org/software/octave/doc/v6.1.0/

[17] C. Sanderson and R. Curtin, "Armadillo: a template-based C++ library for linear algebra," *The Journal of Open Source Software*, vol. 1, no. 2, p. 26, Jun. 2016.

[18] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *COMPUTERS IN PHYSICS*, vol. 12-6, pp. 620–631, 1998.

[19] K. Nilsson, W. Z. Shen, J. N. Sørensen, S.-P. Breton, and S. Ivanell, "Validation of the actuator line method using near wake measurements of the mexico rotor," *Wind Energy*, vol. 18, no. 9, pp.

1683–1683, 2015.

[20] X.-f. Lin, J.-s. Zhang, Y.-q. Zhang, J. Zhang, and S. Liu, "Comparison of actuator line method and full rotor geometry simulations of the wake field of a tidal stream turbine," *Water*, vol. 11, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2073-4441/11/3/560

[21] K. . R. Gieck, Ed., *Technische Formelsammlung*, 75th ed. Gieck Verlag, 1995.