46415
Structural Analysis and Design Optimization of Wind Turbine Blades

Mini-Project 2: Optimum Blade

(2 page report is due on the 13th of June 2018 at 13.00)

The 46415 teaching team *

June 7, 2018
DTU Wind Energy

1. Introduction

Mini-Project 2 builds on Mini-Project 1 introducing the following changes and new concepts:

- A 3D beam finite element model is now used.
- The thin-walled ellipse is added to the cross section as a simple representation of the airfoil.
- The wall thickness e of the ellipse is added as design variable.
- A edgewise bending loadcase is added.
- Eigenfrequency constraints are added.

2. Optimization Problem

The considered blade design optimization now reads

*Contact: Mathias Stolpe. E-mail: matst@dtu.dk

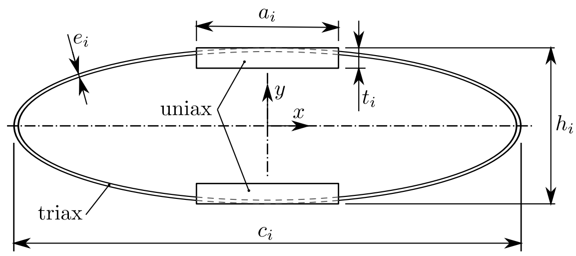

Figure 1: Cross section of section i.

$$
\begin{array}{lll}
\underset{\mathbf{a},\mathbf{t},\mathbf{e}\,\in\,\mathbb{R}^N}{\text{minimize}} & m(\mathbf{a},\mathbf{t},\mathbf{e}) & \\
\text{subject to} & \delta(\mathbf{a},\mathbf{t},\mathbf{e}) \leq \delta^{\max} & \\
 & \varepsilon_i(a_i,t_i,e_i) \leq \varepsilon^{\max} & i = 1,\ldots,N \\
 & \eta_i(a_i,t_i) \leq \eta^{\max} & i = 1,\ldots,N \\
 & f_j^{\min} \leq f_j(\mathbf{a},\mathbf{t},\mathbf{e}) \leq f_j^{\max} & j = 1,\ldots,k \\
 & a_i^{\min} \leq a_i \leq a_i^{\max} & i = 1,\ldots,N \\
 & t_i^{\min} \leq t_i \leq t_i^{\max} & i = 1,\ldots,N \\
 & e_i^{\min} \leq e_i \leq e_i^{\max} & i = 1,\ldots,N
\end{array}
$$

where $m(\mathbf{a},\mathbf{t},\mathbf{e})$ is the mass of the blade and $\mathbf{a} = [a_1,\ldots,a_N]^T$, $\mathbf{t} = [t_1,\ldots,t_N]^T$ and $\mathbf{e} = [e_1,\ldots,e_N]^T$ are the design variables. a_i is the cap width at section i, t_i is the cap thickness at section i and e_i is the thickness of the ellipse at section i, as shown in Figure 1. The tip displacement $\delta(\mathbf{a},\mathbf{t},\mathbf{e})$, the bending strain $\varepsilon_i(a_i,t_i,e_i)$ in each section and the the buckling coefficient $\eta_i(a_i,t_i)$ in each section must be smaller or equal to given values. The eigenfrequencies of the blade are constrained by an upper and lower limit. N is the number of sections and k is the number of eigenfrequency constraints.

3. Analysis

The theory underlying the functions used for the analysis of the structural response of the beam are presented in here. The subsections mimic the same structure adopted in the Matlab implementation, i.e., analysis of cross section properties in `compute_csprops`, assembly of the cross section constitutive matrices in `compute_constitutive`, assemble of the beam finite element model in `compute_beam`, and solve the beam finite element equations in `compute_beam_solution`.

3.1. Assumptions and Parameters

Is is assumed that the caps of the box girder and the ellipse representing the airfoil are made from unidirectional plies and triaxial plies, respectively. The effective material properties of the unidirectional and triaxial plies are given in Table 1. It may again be assumed that the plies are available in any thickness, so the cap thickness t_i and the thickness of the ellipse e_i can be treated as a continuous variables.

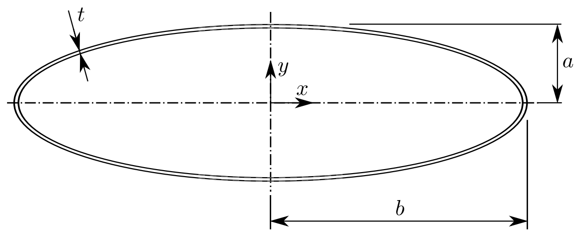

Figure 2: Thinwalled ellipse for definition of cross section properties.

The height $h(z)$, the chord length $c(z)$, and the bending moments $M_x(z)$ and $M_y(z)$ are described by given spline curves, which can be loaded into Matlab using the command `load('splines')`. The splines are also defined in Appendix A. The first derivative of the spline curves describing bending moments yields the shear force, the second derivative yields the distributed load (force per length).

The box girder is partitioned into $N = 20$ sections of equal length Δz. Table 1 lists a number of model parameters.

3.2. Cross Section Stiffness and Mass Properties

The cross section stiffness and mass properties are discussed in this section. See the function `compute_csprops` for a Matlab implementation of the theory presented here.

3.2.1. Area

The total area is the sum of the area of the caps and ellipse. The evaluation of the area of the caps has been presented in Mini-project 1. The area of the thin-walled ellipse with constant wall thickness t shown if Figure 2 is [1, Table A.1/26]:

$$A_{ellipse} = \pi t(a+b)\left[1 + K_1\left(\frac{a-b}{a+b}\right)^2\right] \quad \text{where} \tag{1}$$

$$K_1 = 0.2464 + 0.002222\left(\frac{a}{b} + \frac{b}{a}\right) \quad . \tag{2}$$

Note that the definition of a and t in Figure 2 differs from the global definition in Figure 1. The total area is given by

$$A = A_{caps} + A_{ellipse} \tag{3}$$

3.2.2. Mass per unit length

The total mass per unit length m_s is the sum of the contributions from the caps and ellipse and is thus given as

$$m_s = m_{s,caps} + m_{s,ellipse} = \rho_{uniax} A_{caps} + \rho_{triax} A_{ellipse} \tag{4}$$

	Material Data: Uniax	
E_1	42 GPa	Young's modulus in 1-direction
E_2	10 GPa	Young's modulus in 2-direction
G_{12}	4 GPa	In-plane shear modulus
ν_{12}	0.28	Poisson ratio 12
ρ	$1950\,\mathrm{kg/m^3}$	Mass density
	Material Data: Triax	
E_1	22 GPa	Young's modulus in 1-direction
E_2	15 GPa	Young's modulus in 2-direction
G_{12}	9.5 GPa	In-plane shear modulus
ν_{12}	0.45	Poisson ratio 12
ρ	$1845\,\mathrm{kg/m^3}$	Mass density
	Geometry of the blade	
L	89.166 m	Blade length measured from $r = 0$ to $r = r^{\max}$
h	$h(z)$	Height of the profile: defined by spline function `height_spline` or Appendix A
c	$c(z)$	Chord length: defined by spline function `chord_spline` or Appendix A
N	20	Number of sections
	Load	
M_x	$M_x(z)$	Flapwise bending moment: defined by spline function `moment_x_spline` or Appendix A
M_y	$M_y(z)$	Edgewise bending moment: defined by spline function `moment_y_spline` or Appendix A
	Design variable bounds	
$a^{\min}$	0.1 m	Lower bound for cap width
$a^{\max}$	$a^{\max}(z)$	Upper bound for cap width: Piecewise linear function through points $(0, 1.0)$, $(40, 1.0)$, $(89.166, 0.3)$
$t^{\min}$	0.01 m	Lower bound for cap thickness
$t^{\max}$	0.1 m	Upper bound for cap thickness
$e^{\min}$	0.001 m	Lower bound for ellipse thickness
$e^{\max}$	0.02 m	Upper bound for ellipse thickness
	Constraints	
$\delta^{\max}$	16 m	Maximum allowed tip displacement
$\varepsilon^{\max}$	0.0045	Maximum allowed strain
$\eta^{\max}$	0.5	Maximum allowed buckling coefficient
$f_1^{\min}$	0.4	Minimum allowed first eigenfrequency
$f_1^{\max}$	0.6	Maximum allowed first eigenfrequency
	Accuracy of the optimization solver	
ϵ_d	1×10^{-6}	Optimality tolerance
ϵ_r	1×10^{-6}	Constraint violation tolerance

Table 1: Model parameters.

3.2.3. Moment of inertia

The second moment of area I_x of the thin-walled ellipse with constant wall thickness t shown if Figure 2 is [1, Table A.1/26]:

$$I_{x,ellipse} = \frac{\pi}{4} t a^2 (a+3b) \left[1 + K_2 \left(\frac{a-b}{a+b}\right)^2\right] + \frac{\pi}{16} t^3 (3a+b) \left[1 + K_3 \left(\frac{a-b}{a+b}\right)^2\right] \quad (5)$$

$$K_2 = 0.1349 + 0.1279 \frac{a}{b} - 0.01284 \left(\frac{a}{b}\right)^2 \quad (6)$$

$$K_3 = 0.1349 + 0.1279 \frac{b}{a} - 0.01284 \left(\frac{b}{a}\right)^2 \quad (7)$$

For $I_{y,ellipse}$ interchange a and b in the expressions for $I_{x,ellipse}$, K_2 and K_3.

The second *mass* moment of inertia is obtained as

$$I_{xx} = \rho I_x \text{ and } I_{yy} = \rho I_y. \quad (8)$$

Consequently, the total second mass moment of inertia is given as

$$I_{xx} = I_{xx,caps} + I_{xx,ellipse} = \rho_{uniax} I_{x,caps} + \rho_{triax} I_{x,ellipse} \quad (9)$$

$$I_{yy} = I_{yy,caps} + I_{yy,ellipse} = \rho_{uniax} I_{y,caps} + \rho_{triax} I_{y,ellipse} \quad (10)$$

3.2.4. Longitudinal Stiffness

The longitudinal stiffness of the beam is henceforth referred to as EA. It is obtained as the sum of the longitudinal stiffness of the caps and ellipse and is given as

$$EA = E_{1,uniax} A_{caps} + E_{1,triax} A_{ellipse} \quad (11)$$

where $A_{ellipse}$ is defined in (1).

3.2.5. Shear Stiffness

As a simplification both shear correction factors of the ellipse are assumed to be constant and identical:

$$k_{ellipse} = k_{x,ellipse} = k_{y,ellipse} = 0.53\ . \quad (12)$$

The shear stiffness of the ellipse is:

$$k_x G A_{ellipse} = k_y G A_{ellipse} = k_{ellipse}\, G_{12,triax}\, A_{ellipse} \quad (13)$$

As the shear webs of the box girder are not included in the model, it is not possible to compute a realistic shear stiffness for the box girder. Therefore, a very high value is assigned to the shear stiffness of the caps in x- and y-direction, in fact turning the Timoshenko beam model into a Euler-Bernoulli beam model.

Note that the shear stiffness of a cross section *cannot* be computed as the sum of the shear stiffnesses of parts of the cross section. Formally, the respective cross section stiffness properties of the caps and the ellipse are added in the Matlab code to compute the "total" cross section stiffness properites. In the case of the shear stiffness this does not matter, because of the assumptions regarding the shear stiffness of the caps described above.

3.2.6. Bending Stiffness

The bending stiffness of the beam is the sum of the bending stiffness of the caps and the bending stiffness of the ellipse.

The bending stiffness around the x and y axis of the cross section is henceforth referred to as EI_x and EI_y, respectively. It is obtained as the sum of the bending stiffness of the caps and ellipse and is given as

$$EI_x = E_{1,uniax} I_{x,caps} + E_{1,triax} I_{x,ellipse} \tag{14}$$

$$EI_y = E_{1,uniax} I_{y,caps} + E_{1,triax} I_{y,ellipse} \tag{15}$$

3.2.7. Torsional Stiffness

The torsional stiffness constant K of the thin-walled ellipse with constant wall thickness t shown if Figure 2 is [1, Table 10.1/13]:

$$K_{ellipse} = \frac{4\pi^2 t \left[\left(a - \frac{t}{2}\right)^2 \left(b - \frac{t}{2}\right)^2\right]}{U} \tag{16}$$

$$U = \pi(a + b - t)\left[1 + 0.258\frac{(a-b)^2}{(a+b-t)^2}\right] \tag{17}$$

As a simplification the contribution of the caps to the torsional stiffness of the beam is neglected and thus the torsional stiffness GK is given by

$$GK = G_{12,triax} K_{ellipse} \tag{18}$$

Note that the torsional stiffness of a cross section (like the shear stiffness) *cannot* be computed as the sum of the torsional stiffnesses of parts of the cross section. Formally, the respective cross section stiffness properties of the caps and the ellipse are added in the Matlab code to compute the "total" cross section stiffness properites. In the case of the torsional stiffness this does not matter, because the torsional stiffness of the caps is set to zero in the Matlab code.

3.3. Cross Section Stiffness and Mass Matrix

It is assumed that the load application point, the elastic center, the mass center, and the position of the beam finite element node coincide at each cross section. Moreover, all relevant cross section properties are determined with respect to this position.

For a linear elastic beam there exists a linear relation between the cross section generalized forces $\mathbf{T}$ and moments $\mathbf{M}$ in $\boldsymbol{\theta} = \left[\mathbf{T}^T\mathbf{M}^T\right]^T$, and the resulting strains $\boldsymbol{\tau}$ and curvatures $\boldsymbol{\kappa}$ in $\boldsymbol{\psi} = \left[\boldsymbol{\tau}^T\boldsymbol{\kappa}^T\right]^T$. This relation is given in its stiffness form as $\mathbf{K}_s\boldsymbol{\psi} = \boldsymbol{\theta}$, where $\mathbf{K}_s$ is the 6×6 cross section stiffness matrix. In the most general case, considering material anisotropy and inhomogeneity, all the 21 stiffness parameters in $\mathbf{K}_s$ may be required to describe the deformation of the cross section. In the current project, the entries of $\mathbf{K}_s$ are determined as

$$\mathbf{K}_s = \begin{bmatrix} k_xGA & 0 & 0 & 0 & 0 & 0 \\ 0 & k_yGA & 0 & 0 & 0 & 0 \\ 0 & 0 & EA & 0 & 0 & 0 \\ 0 & 0 & 0 & EI_x & 0 & 0 \\ 0 & 0 & 0 & 0 & EI_y & 0 \\ 0 & 0 & 0 & 0 & 0 & GK \end{bmatrix} \tag{19}$$

The 6×6 cross section mass matrix $\mathbf{M}_s$ relates the linear and angular velocities in $\boldsymbol{\phi}$ to the generalized inertial linear and angular momentum in $\boldsymbol{\gamma}$ through $\boldsymbol{\phi} = \mathbf{M}_s\boldsymbol{\gamma}$. The coefficients of $\mathbf{M}_s$ for the general case are

$$\mathbf{M}_s = \begin{bmatrix} m_s & 0 & 0 & 0 & 0 & -m_sy_m \\ 0 & m_s & 0 & 0 & 0 & m_sx_m \\ 0 & 0 & m_s & m_sy_m & -m_sx_m & 0 \\ 0 & 0 & m_sy_m & I_{xx} & -I_{xy} & 0 \\ 0 & 0 & -m_sx_m & -I_{xy} & I_{yy} & 0 \\ -m_sy_m & m_sx_m & 0 & 0 & 0 & I_{xx}+I_{yy} \end{bmatrix} \tag{20}$$

where m_s is the mass per unit length, I_{xx} and I_{yy} are the *mass* moment of inertia with respect to x and y, respectively, and I_{xy} is the product of inertia. The off-diagonal terms are due to the offset between the position of the cross section reference center and the mass center $\mathbf{m}_c = (x_m, y_m)$. In this project the $\mathbf{m}_c = (0, 0$ and thus the cross section mass matrix is given as

$$\mathbf{M}_s = \begin{bmatrix} m_s & 0 & 0 & 0 & 0 & 0 \\ 0 & m_s & 0 & 0 & 0 & 0 \\ 0 & 0 & m_s & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{xx}+I_{yy} \end{bmatrix} \tag{21}$$

The theory presented in this section is implemented in `compute_constitutive` in Matlab.

3.4. Beam Finite Element Analysis

3.4.1. Beam Finite Element Stiffness and Mass Matrix

The beam finite element stiffness and mass matrix for element b are given by

$$\mathbf{K}_b = \int_0^{L_b} \mathbf{B}_b^T\mathbf{K}_s\mathbf{B}_b \,\mathrm{d}z \quad \text{and} \quad \mathbf{M}_b = \int_0^{L_b} \mathbf{N}_b^T\mathbf{M}_s\mathbf{N}_b \,\mathrm{d}z \tag{22}$$

where L_b is the length of element b. The beam finite element stiffness matrix $\mathbf{K}_b$ for element b is given in function of $\mathbf{B}_b = \mathcal{B}(\mathbf{N}_b)$ where $\mathcal{B}$ is the strain-displacement relation which is a function of $\mathbf{N}_b$, the finite element shape function matrix. The cross section stiffness and mass matrices $\mathbf{K}_s$ and $\mathbf{M}_s$, respectively, have been defined in the previous section. The global beam stiffness and mass matrix $\mathbf{K}$ and $\mathbf{M}$ are defined as

$$\mathbf{K} = \sum_{b=1}^{n_b} \mathbf{K}_b \quad \text{and} \quad \mathbf{M} = \sum_{b=1}^{n_b} \mathbf{M}_b \tag{23}$$

where n_b is the number of elements in the beam finite element assemblage, and $\mathbf{K}_b$ and $\mathbf{M}_b$ are the beam finite element stiffness and mass matrix for element b, respectively. The summation refers to the typical finite element assembly. The cross section stiffness and mass matrix, $\mathbf{K}_s$ and $\mathbf{M}_s$, are defined in (19) and (21), respectively.

The theory presented in this section is implemented in `compute_beam` in Matlab.

3.4.2. Displacement Solution

The deformation $\mathbf{u}$ resulting from the loads $\mathbf{f}$ is the solution to the following linear system of equations

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{24}$$

where $\mathbf{K}$ is the beam finite element stiffness matrix defined in (23).

Implementation notes: In Matlab the solution to a system like the one above is obtained by writing `u=K\f`. The theory presented in this section is implemented in `compute_beam_solution`.

3.4.3. Cross Section Forces and Moments

The cross section forces $\mathbf{T}$ and moments $\mathbf{M}$ are determined based on the displacement solution obtained from (24). For the four node beam finite element from FRANS, the forces and moments at each element of the beam finite element assembly $\mathbf{f}_e = [\mathbf{T}_{e,1}\ \mathbf{M}_{e,1}\ ...\ \mathbf{T}_{e,4}\ \mathbf{M}_{e,4}]$ are determined as

$$\mathbf{f}_e = \mathbf{K}_e\mathbf{u}_e \tag{25}$$

where $\mathbf{u}_e$ are the entries of the displacement vector $\mathbf{u}$ which are associated with the element e, and $\mathbf{T}_{e,k}$ and $\mathbf{M}_{e,k}$ are the cross section forces and moments, respectively, at node k of element e. The vector of cross section strains and curvatures are obtained using the cross section constitutive relation in its compliance form, i.e.,

$$\boldsymbol{\Psi} = \mathbf{K}_s^{-1}\mathbf{f}_e \tag{26}$$

The theory presented in this section is implemented in `compute_beam_solution`.

3.4.4. Bending Strains and Buckling Analysis

The bending strains and buckling coefficient are analyzed using the same routines as in Mini-project 1. The only difference is that the curvatures in $\boldsymbol{\kappa}$ are determined from the beam finite element assembly as detailed in Section 3.4.3. Also, for buckling calculations, the thickness of the caps and ellipse are considered.

Implementation notes: Note that the cross section forces and moments are analyzed at the node closest to the root of the blade as this is the node at which the bending moment is higher. As a result the strains measured using the beam finite element are slightly higher than those measured in Mini-project 1 in which the cross section forces and moments were measured at the center of the element.

3.4.5. Eigenfrequency solution

The finite element form of the beam structural eigenvalue problem is

$$\left(\mathbf{K} - \omega_f^2 \mathbf{M}\right) \mathbf{v}_f = 0, \ \forall f = 1, ..., n_d \tag{27}$$

where n_d is the number of degrees of freedom associated with the finite element stiffness and mass matrices, $\mathbf{K}$ and $\mathbf{M}$, respectively. The problem above yields the eigenfrequencies $\boldsymbol{\omega} = \{\omega_1, ..., \omega_{n_d}\}$ associated with the eigenvectors $\mathbf{v} = \{\mathbf{v}_1, ..., \mathbf{v}_{n_d}\}$.

Implementation notes: In the current implementation the eigenfrequencies in $\boldsymbol{\omega}$ are given in ascending order of magnitude, i.e., $\omega_1 \leq \omega_2 \leq ... \leq \omega_{n_d}$. The eigenvectors are ordered accordingly and are already mass normalized. The theory presented in this section is implemented in `compute_beam_solution`.

4. Sensitivity analysis

4.1. Cross Section Stiffness and Mass Matrix

The gradients of the cross section stiffness matrix are given by

$$\frac{\partial \mathbf{K}_s(\mathbf{x})}{\partial x_i} = \begin{bmatrix} \frac{\partial k_x GA(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial k_y GA(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial EA(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial EI_x(\mathbf{x})}{\partial x_i} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial EI_y(\mathbf{x})}{\partial x_i} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial GK(\mathbf{x})}{\partial x_i} \end{bmatrix} \tag{28}$$

which resolves to the calculation of the gradients of each of the entries. The gradient of the cross section mass matrix is given by

$$\frac{\partial \mathbf{M}_s(\mathbf{x})}{\partial x_i} = \begin{bmatrix} \frac{\partial m_s(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial m_s(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial m_s(\mathbf{x})}{\partial x_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial I_{xx}(\mathbf{x})}{\partial x_i} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial I_{yy}(\mathbf{x})}{\partial x_i} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial I_x(\mathbf{xx})}{\partial x_i} + \frac{\partial I_{yy}(\mathbf{x})}{\partial x_i} \end{bmatrix} \tag{29}$$

Implementation Notes: The gradients of each of the entries has been derived in the Maple file accompanying the code.

4.2. Finite Element Stiffness and Mass Matrix

The sensitivities of the global beam finite element stiffness matrix $\mathbf{K}$ are obtained through differentiation of $\mathbf{K}$ in (23) to yield

$$\frac{\partial \mathbf{K}(\mathbf{x})}{\partial x_i} = \sum_{b=1}^{n_b} \int_0^{L_b} \mathbf{B}_b^T \frac{\partial \mathbf{K}_s(\mathbf{x})}{\partial x_i} \mathbf{B}_b \, \mathrm{d}z \tag{30}$$

The gradient of the cross section stiffness matrix $\mathbf{K}_s$ is described in Section 4.1. The gradients of the global beam finite element mass matrix $\mathbf{M}(\mathbf{x})$ are obtained through differentiation of $\mathbf{M}$ in (23) and defined as

$$\frac{\partial \mathbf{M}(\mathbf{x})}{\partial x_i} = \sum_{b=1}^{n_b} \int_0^{L_b} \mathbf{N}_b^T \frac{\partial \mathbf{M}_s(\mathbf{x})}{\partial x_i} \mathbf{N}_b \, \mathrm{d}z \tag{31}$$

Implementation Notes: Note that the same routines used for computing the element stiffness matrices $\mathbf{K}$ and $\mathbf{M}$ can be used to build the gradients by simply providing $\frac{\partial \mathbf{K}_s(\mathbf{x})}{\partial x_i}$ and $\frac{\partial \mathbf{M}_s(\mathbf{x})}{\partial x_i}$ instead of $\mathbf{K}_s$ and $\mathbf{M}_s$.

4.2.1. Displacement

The solution to the linear system of equations in (24) yields the displacements $\mathbf{u}$. The gradients of the displacement with respect to the design variables for the case of design independent loads is given by

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial \mathbf{x}} = -\frac{\partial \mathbf{K}}{\partial \mathbf{x}}\mathbf{u} \tag{32}$$

which is obtained after applying the chain rule to (24).

Implementation Notes: Similarly to the displacement solution in (24), the gradients $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ are obtained as `dudx = - K \ (dKdx*u)`. Note that this corresponding to solving the system for as many right hand sides as number of design variables. The procedure can be implemented efficiently by considering elementwise computations of the right hand side.

4.2.2. Eigenfrequencies

The solution to the structural eigenvalue problem in (27) yields the eigenfrequencies and eigenvectors $\boldsymbol{\omega} = \{\omega_1, ..., \omega_{n_d}\}$ and $\mathbf{v} = \{\mathbf{v}_1, ..., \mathbf{v}_{n_d}\}$, respectively. It is assumed that the eigenvectors are mass-normalized such that

$$\mathbf{v}_p^T \mathbf{M}(\mathbf{x}) \mathbf{v}_q = \delta_{pq}, \ \forall p, q = 1, ..., n_d.$$

where n_d is the number of degrees of freedom, and δ_{pq} is the Kronecker delta such that $\delta_{pq} = 1$ if $p = q$ and $\delta_{pq} = 0$ otherwise. The gradient of a single eigenfrequency ω_p with respect to the design variable x_i is given by

$$\frac{\partial \omega_p^2(\mathbf{x})}{\partial x_i} = \mathbf{v}_p^T \left(\frac{\partial \mathbf{K}(\mathbf{x})}{\partial x_i} - \omega_p^2(\mathbf{x}) \frac{\partial \mathbf{M}(\mathbf{x})}{\partial x_i} \right) \mathbf{v}_p \qquad (33)$$

Implementation Notes: Note that in order to solve the eigenvalue problem above the boundary conditions on $\mathbf{K}$ and $\mathbf{M}$ are enforced by removing the rows and columns corresponding to the d.o.f. where the beam is clamped. This is different from the approach employed for enforcing the boundary conditions when solving for the displacements.

5. Implementation

The focus of this section is on the description of the data necessary for running the optimization procedure, and the output from the structural analysis functions. A series of auxiliary symbols and parameters are described in Table 2.

Name	Size	Type	Description
N	(1×1)	scalar	Number of beam finite elements.
n_{dof}	(1×1)	scalar	Number of degrees of freedom (d.o.f.) in the finite element assembly.
n_p	(1×1)	scalar	Number of load cases.
n_{bc}	(1×1)	scalar	Number of d.o.f. which are constrained at the boundary.
n_f	(1×1)	scalar	Number of frequencies constraints considered throughout the optimization.
n_m	(1×1)	scalar	Number of distinct materials considered.

Table 2: Symbols used for describing the fields in Tables 3 and 4

5.1. Input

The input stored in the structure **problem**, required to run the optimization, is described in Table 3.

Name	Size	Type	Description
problem.	(1×1)	structure	Structure with problem definition, input, etc.
.info.	(1×1)	structure	Structure with user provided input and other parameters determined based on input and required for the optimization.
.beam_length	(1×1)	scalar	Length of the beam
.num_sec	(1×1)	scalar	Number of sections
.delta_z	(1×1)	scalar	Distance between sections or length of each beam finite element.
.z_eval	$(N \times 1)$	array	Position of points where cross section height and chord length are evaluated.
.z_nodes	$((N+1) \times 1)$	array	Position of the nodes of the beam finite elements.
.height_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section height.
.chord_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section chord.
.moment_x_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section bending moment M_x.
.moment_y_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section bending moment M_x.
.shearforce_y_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section shear forces T_y.
.distributed_load_y_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of loads in the y direction.
.shearforce_x_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of cross section shear forces T_x.
.distributed_load_x_spline.	(1×1)	structure	Parameters associated with the spline defining the lengwise distribution of loads in the x direction.
.form	(1×1)	string	String defining the form of the spline.
.breaks	$(1 \times n_c)$	array	Longitudinal position of the control points.
.coefs	$(n_c \times 4)$	array	Coefficients of the control points.
.pieces	(1×1)	scalar	Number of pieces in each spline.
.order	(1×1)	scalar	Order of spline polynomial.
.saved_solution.	(1×1)	structure	Solution saved in the last iteration of the optimization process.
.initial_solution.	(1×1)	structure	Solution obtained based on the initial design whose input is provided by the user.
.final_solution.	(1×1)	structure	Solution obtained based on the final design resulting from the optimization process.
.deflection	$(n_{dof} \times 1)$	array	Displacement solution at each node of the beam finite element model for all d.o.f..
.tip_disp	(1×1)	scalar	Displacement at the tip of the blade in the flapwise or y direction.
.strain	$(N \times 2)$	array	Maximum strains at each of the sections of the blade.

.eta	$(N \times 1)$	array	Buckling coefficient at each section of the blade.
.freq	$(n_f \times 1)$	array	Magnitude of the eigenfrequencies.
.m	(1×1)	scalar	Total mass of the blade.
.iteration	(1×1)	scalar	Number of current iteration.
.firstorderopt	(1×1)	scalar	Magnitude of th first order optimality criteria at current iteration.
.materialdata.	$(1 \times n_m)$	structure	Material properties for the n_m considered material.
.E1	(1×1)	scalar	Young's modulus in the 1-direction.
.E2	(1×1)	scalar	Young's modulus in the 2-direction.
.nu12	(1×1)	scalar	Poisson's ration 12.
.G12	(1×1)	scalar	In -plane shear modulus.
.rho	(1×1)	scalar	Mass density.
.name	(1×1)	string	Name or label of material.
.constraints.	(1×1)	structure	Magnitude of constraints.
.max_tip_disp	(1×1)	scalar	Maximum value of tip displacement.
.max_buckling_factor	(1×1)	scalar	Maximum buckling factor.
.max_strain	(1×2)	array	Maximum allowed strain at caps and ellipse.
.max_freq	(1×2)	array	Upper bound on the eigenfrequencies.
.min_freq	(1×2)	scalar	Lower bound on the eigenfrequencies.
.options.	(1×1)	structure	Options associated with plotting and optimization algorithm.
.plotting.	(1×1)	structure	Options associated with plotting.
.activate	(1×1)	axes	Flag to activate the plotting throughout the optimization.
.plotsplines	(1×1)	axes	Flag to plot the input splines.
.axes.	(1×1)	structure	Axes structure containing information on the properties of the plots.
.optimizer.	(1×1)	structure	Options associated with the optimization algorithm.
optimality_tol	(1×1)	scalar	Setting optimality tolerance.
constr_viol_tol	(1×1)	scalar	Setting constraint violation tolerance.
check_derivatives	(1×1)	scalar	Turning on/off derivative check using finite differences (useful for debugging sensititivities).
.varbounds.	(1×1)	structure	Bounds on design variables.
.lb. (a, t, e)	$(N \times 1)$	array	Setting lower bound values.
.ub. (a, t, e)	$(N \times 1)$	array	Setting upper bound values.
.initial_design. (a, t, e)	$(N \times 1)$	array	Defining initial values for design variables **a**, **t**, and **e**.
.load.	(1×1)	structure	Setting the magnitude of the loads acting on the beam finite element model.
.M_x	$(N \times 1)$	array	Bending moment M_x around the x-axis.
.M_y	$(N \times 1)$	array	Bending moment M_y around the y-axis.
.nodal_forces_x	$((N+1) \times 1)$	array	Nodal forces acting on the beam finite element model in the x-direction.
.nodal_forces_y	$((N+1) \times 1)$	array	Nodal forces acting on the beam finite element model in the y-direction.
.geometry. (h, c)	$(N \times 1)$	array	Setting the geometrical parameters for section height h, and chord c.
.scaling.	(1×1)	structure	Scaling parameters.
.char_a	(1×1)	scalar	Characteristic magnitude of cap width a.
.char_t	(1×1)	scalar	Characteristic magnitude of cap thickness t.
.char_e	(1×1)	scalar	Characteristic magnitude of ellipse thickness e.
.char_m	(1×1)	scalar	Characteristic magnitude of total blade mass m.
.char_disp	(1×1)	scalar	Characteristic magnitude of blade tip displacement.
.char_strain	(1×1)	scalar	Characteristic magnitude of strains.

.char_freq	(1×1)	scalar	Characteristic magnitude of eigenfrequencies.
.frans.	(1×1)	structure	FRANS specific structure with parameters required to assemble the beam finite element model.
.ne_1d	(1×1)	scalar	Number of elements in the beam finite element assembly.
.nn_1d	(1×1)	scalar	Number of nodes in the beam finite element assembly.
.el_1d	$(N \times 5)$	scalar	Element connectivity list.
.nl_1d	$((3 \times N + 1) \times 5)$	array	List of nodal positions.
.bc_1d	$(n_b \times 3)$	array	Boundary conditions of the beam finite element model.
.f_1d	$(n_p \times 5)$	array	Load vector of the beam finite element model.
.nnpe_1d	(1×1)	scalar	Number of nodes per element in the beam finite element assembly.
.mdim_1d	(1×1)	scalar	Number of degrees of freedom per element in the beam finite element assembly.
.nb_1d	(1×1)	scalar	Number of constrained degrees of freedom.
.np_1d	(1×1)	scalar	Number of loads applied in the beam finite element model.
.nlc_1d	(1×1)	scalar	Number of load cases.
.pr_1d	$(12 \times N)$	array	Nodal positions order for beam finite element assemblage.
.edof_1d	$(24 \times N)$	array	Degrees of freedom of each element for mapping from element to global.
.GQ	$(12 \times 2 \times 11)$	array	Weights for Gauss quadrature.

Table 3: Description of all the fields included in the structure **problem** containing the parameters necessary to solve the optimization problem in Matlab.

5.2. Analysis Functions

The analysis of the structural response of the blade is based on the code FRANS. This is a beam finite element code developed at DTU Wind Energy for the analysis of straight beams whose section properties are described by a 6x6 cross section stiffness and mass matrix. FRANS is a linear elastic analysis tool based on four node beam elements with cubic Lagrangian shape functions.

The analysis of the total mass of the blade in `compute_objective` which is used as the objective function is relatively simple. Most of the work is carried out for the calculation of the constraints in the function `compute_constraints`. Here there are four functions which compose the analysis part, namely:

- `[ csprops ] = compute_csprops( problem, design )` - Function for evaluation of the cross section stiffness mass and stiffness properties and its gradients.
- `[ constitutive ] = compute_constitutive( problem, csprops )` - Function to assemble the cross section constitutive stiffness and mass matrix and its gradients.
- `[ beam ] = compute_beam( problem, constitutive )` - Function to assemble the beam finite element stiffness and mass matrix and calculate the gradients of the element stiffness and mass matrices.

- `[ solution ] = compute_beam_solution( problem, beam, constitutive )` - Function to calculate the solution to the beam finite element analysis displacement and eigenvalue problems.

The content of each of the structures is described in detail in the next section. These results are the building blocks for the calculation of the gradients of the constraint functions.

5.2.1. Output of Analysis Functions

The output from the analysis of the structural response of the beam is summarized in Table 4.

Name	**Size**	**Type**	**Description**
csprops.	(1×1)	structure	Cross section stiffness and mass properties.
caps.	(1×1)	structure	Cross section stiffness and mass properties of the caps.
ellipse.	(1×1)	structure	Cross section stiffness and mass properties of the ellipse.
total.	(1×1)	structure	Cross section stiffness and mass properties of the caps and ellipse added together.
.A	$(N \times 1)$	array	Cross section area.
.Ix	$(N \times 1)$	array	Second area moment of inertia around x.
.Iy	$(N \times 1)$	array	Second area moment of inertia around y.
.K	$(N \times 1)$	array	Torsional constant.
.EA	$(N \times 1)$	array	Axial stiffness.
.kGA	$(N \times 1)$	array	Shear stiffness.
.EIx	$(N \times 1)$	array	Bending stiffness around x.
.EIy	$(N \times 1)$	array	Bending stiffness around y.
.GK	$(N \times 1)$	array	Torsional stiffness.
.ms	$(N \times 1)$	array	Cross section mass per unit length.
.me	$(N \times 1)$	array	Mass of beam finite element.
.dt. (A, Ix, ...)	(1×1)	structure	Gradients of cross section properties with respect to $\mathbf{t}$ (same fields (A, Ix, Iy, ...) as csprops.total).
.da. (A, Ix, ...)	(1×1)	structure	Gradients of cross section properties with respect to $\mathbf{a}$ (same fields (A, Ix, Iy, ...) as csprops.total).
.de. (A, Ix, ...)	(1×1)	structure	Gradients of cross section properties with respect to $\mathbf{e}$ (same fields (A, Ix, Iy, ...) as csprops.total).
constitutive.	(1×1)	structure	Cross section constitutive matrices
.Ks	$(6 \times 6 \times N)$	array	Cross section constitutive stiffness matrix.
.Ms	$(6 \times 6 \times N)$	array	Cross section constitutive mass matrix.
.da. (Ks, Ms)	(1×1)	structure	Gradients of cross section constitutive matrices with respect to a.
.dt. (Ks, Ms)	(1×1)	structure	Gradients of cross section constitutive matrices with respect to t.
.de. (Ks, Ms)	(1×1)	structure	Gradients of cross section constitutive matrices with respect to e.
.d*.Ks	$(6 \times 6 \times N)$	array	Gradient of cross section constitutive stiffness matrix.
.d*.Ms	$(6 \times 6 \times N)$	array	Gradient of cross section constitutive mass matrix.
beam.	(1×1)	structure	Beam finite element assembly.
.K	$(n_{dof} \times n_{dof})$	array	Global beam finite element stiffness matrix with boundary conditions enforced by setting diagonal to unity and remaining terms of rows an columns to zero.
.M	$(n_{dof} \times n_{dof})$	array	Global beam finite element mass matrix with boundary conditions enforced by setting diagonal to unity and remaining terms of rows an columns to zero.
.p	$(n_{dof} \times n_p)$	array	Global beam finite element load vector.

.eigK	$((n_{dof}-n_{bc})\times(n_{dof}-n_{bc}))$	array	Global beam finite element stiffness matrix with boundary conditions enforced by removing the d.o.f. where boundary conditions are applied.
.eigM	$((n_{dof}-n_{bc})\times(n_{dof}-n_{bc}))$	array	Global beam finite element mass matrix with boundary conditions enforced by removing the d.o.f. where boundary conditions are applied.
.da. (Ke, Me)	(1×1)	structure	Gradients of beam finite element stiffness matrices with respect to a.
.dt. (Ke, Me)	(1×1)	structure	Gradients of beam finite element stiffness matrices with respect to t.
.de. (Ke, Me)	(1×1)	structure	Gradients of beam finite element stiffness matrices with respect to e.
.d*.Ke	$(24\times 24\times N)$	array	Gradients of beam finite element stiffness matrices with respect to e.
.d*.Me	$(24\times 24\times N)$	array	Gradients of beam finite element stiffness matrices with respect to e.
solution.	(1×1)	structure	Solutions to beam finite element analysis problem.
.u	$(n_{dof}\times 1)$	array	Displacements at each node of the beam finite element model.
.f	$(24\times N)$	array	Cross section forces and moments at each node of the beam finite element model.
.eigfreq	(10×1)	array	First 10 eigenfrequencies for the beam finite element assembly sorted in ascending order.
.eigvec	$(n_{dof}\times 10)$	array	First 10 eigenvectors for the beam finite element assembly, mass normalized, and in the same order as the eigenfrequencies.

Table 4: Description of all the fields included in the structures output by the analysis functions in Matlab. The information contained in these structures are the building blocks for the implementation of the gradients.

6. Mini-project tasks

In order to complete the mini-project please perform (at least) the following tasks. A summary of the most important results must be included in the report.

- Download the Matlab program for solving the optimization problem from DTU Inside.
- Study the description of the project and the Matlab code (basis for the final project).
- A small part of the program is missing and needs to be completed: The computation of the gradients of the eigenfrequencies in `analysis/compute_frequency.m`
- Check your implementation of the gradients of the eigenfrequencies using the finite difference checks in fmincon. If necessary implement your own finite difference checker of the user supplied gradients.
- Perform a sensitivity analysis on the lower and upper bounds on the frequency constraints by solving a number of different problems. What happens to the optimal design? What happens to the optimal mass?

- Perform the sensitivity analysis on the design driving constraints using the Lagrange multipliers. Using the Lagrange multipliers, discuss what constraints and variable bounds would have the greatest impact on the final design. Discuss in terms of relative (*i.e.* varying constraints by $\pm 1\%$) and absolute variations (*i.e.* varying constraints by ± 0.001 units) what constraints are driving the design. Which constraint is the objective most sensitive to? Pick the most sensitive constraint and change the constraint bound by 10%, solve the optimization problem. Comment on whether the objective (and design) improved as you expected it. How did your lagrange multipliers change? How can this information be used in an engineering design process.

References

[1] Warren C. Young and Richard G. Budynas. *Roark's Formulas for Stress and Strain.* McGraw-Hill, 2002.

Appendix A Splines describing geometrical properties and loading

Splines are functions defined piecewise by polynomials (see Figure 3). At the intersections of two polynomial pieces, continuity conditions (e.g. n-times continuously differentiable) are usually assigned in order to achieve a smooth curve.

A spline $p(x)$ can be described in terms of its breaks $\xi_1, \xi_2, \ldots \xi_l$ and its polynomial coefficients c_{ji}:

$$p_j(x) = \sum_{i=1}^{k} (x - \xi_j)^{k-i} c_{ji} \qquad j = 1, 2, \ldots l \quad , \tag{34}$$

where l is the number of polynomial pieces and k is the number of coefficients in each polynomial ($k = 4$ for a cubic spline). The polynomial $p_j(x)$ describes the spline in the interval $\xi_j \leq x \leq \xi_{j+1}$.

In this project splines haven been used to describe bending moments ($M_x(z)$, $M_y(z)$) and geometrical properties ($h(z)$, $c(z)$) as a function of the radial coordinate z.

Tables 5 to 8 display the breaks and polynomial coefficients of the splines used in this project.

Putting together splines in Matlab

The example below demonstrates how a spline defined by its breaks and coefficients can be put together in Matlab using the `ppmak` command.

```
breaks = [2.8000 4.8000 18.8310 27.1510 37.4240 63.6150 89.1660];

coefs = [
0.000   4.800    0.0000E+00   0.0000E+00   0.0000E+00  5.3800E+00;
4.800   18.000   7.1197E-04  -2.0874E-02   0.0000E+00  5.3800E+00;
18.000  35.000  -1.3286E-04   7.3198E-03  -1.7892E-01  3.3804E+00;
35.000  80.000  -3.2478E-06   5.4401E-04  -4.5231E-02  1.8015E+00;
80.000  89.166  -3.9380E-04   8.3219E-04  -1.6000E-02  5.7179E-01];

rel_thick_spline = ppmak(breaks,coefs,1);
```

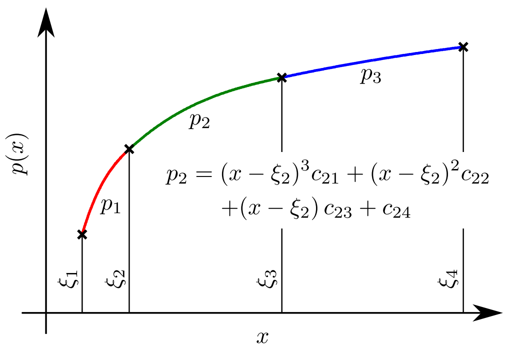

Figure 3: Cubic spline approximating data.

	ξ_j	ξ_{j+1}	c_{j1}	c_{j2}	c_{j3}	c_{j4}
j=1	0.000	4.800	0.0000E+00	0.0000E+00	0.0000E+00	5.3800E+00
j=2	4.800	18.000	7.1197E-04	-2.0874E-02	0.0000E+00	5.3800E+00
j=3	18.000	35.000	-1.3286E-04	7.3198E-03	-1.7892E-01	3.3804E+00
j=4	35.000	80.000	-3.2478E-06	5.4401E-04	-4.5231E-02	1.8015E+00
j=5	80.000	89.166	-3.9380E-04	8.3219E-04	-1.6000E-02	5.7179E-01

Table 5: Spline describing the profile height $h(z)$ in m as a function of the z-coordinate in m.

	ξ_j	ξ_{j+1}	c_{j1}	c_{j2}	c_{j3}	c_{j4}
j=1	0.000	8.196	0.0000E+00	0.0000E+00	0.0000E+00	5.3800E+00
j=2	8.196	19.955	-4.6130E-04	1.0423E-02	0.0000E+00	5.3800E+00
j=3	19.955	28.012	9.5406E-05	-5.8502E-03	5.3769E-02	6.0711E+00
j=4	28.012	38.222	7.9431E-05	-3.5440E-03	-2.1926E-02	6.1745E+00
j=5	38.222	55.027	2.4247E-05	-1.1111E-03	-6.9452E-02	5.6657E+00
j=6	55.027	70.058	8.5976E-06	1.1132E-04	-8.6253E-02	4.2999E+00
j=7	70.058	78.159	4.7121E-06	4.9899E-04	-7.7080E-02	3.0578E+00
j=8	78.159	85.000	-4.3908E-04	6.1351E-04	-6.8067E-02	2.4686E+00
j=9	85.000	86.252	-1.3287E-03	-8.3983E-03	-1.2133E-01	1.8911E+00
j=10	86.252	88.659	-1.4760E-02	-1.3389E-02	-1.4861E-01	1.7234E+00
j=11	88.659	88.986	-6.7982E+00	-1.1999E-01	-4.6970E-01	1.0821E+00
j=12	88.986	89.166	1.0436E+01	-6.7809E+00	-2.7235E+00	6.7906E-01

Table 6: Spline describing the chord length $c(z)$ in m as a function of the z-coordinate in m.

	ξ_j	ξ_{j+1}	c_{j1}	c_{j2}	c_{j3}	c_{j4}	c_{j5}	c_{j6}
j=1	0.000	70.000	-6.0100E-04	5.8905E-02	3.0765E+01	2.5734E-14	-6.8673E+05	3.9149E+07
j=2	70.000	80.000	-1.7972E-02	-1.5145E-01	1.7810E+01	6.1311E+03	-2.2581E+05	2.0344E+06
j=3	80.000	87.000	-2.5632E-01	-1.0501E+00	-6.2202E+00	6.3948E+03	-9.9352E+04	4.0393E+05
j=4	87.000	89.166	-3.6869E+01	-1.0021E+01	-1.6122E+02	5.0763E+03	-1.5257E+04	1.2846E+04

Table 7: Spline describing the bending moment $M_x(z)$ in Nm as a function of the z-coordinate in m.

	ξ_j	ξ_{j+1}	c_{j1}	c_{j2}	c_{j3}	c_{j4}	c_{j5}
j=1	0.000	89.166	-3.2904E-01	6.0494E+01	0.0000E+00	-5.0982E+05	2.3373E+07

Table 8: Spline describing the bending moment $M_y(z)$ in Nm as a function of the z-coordinate in m.