

# Modal Analysis with



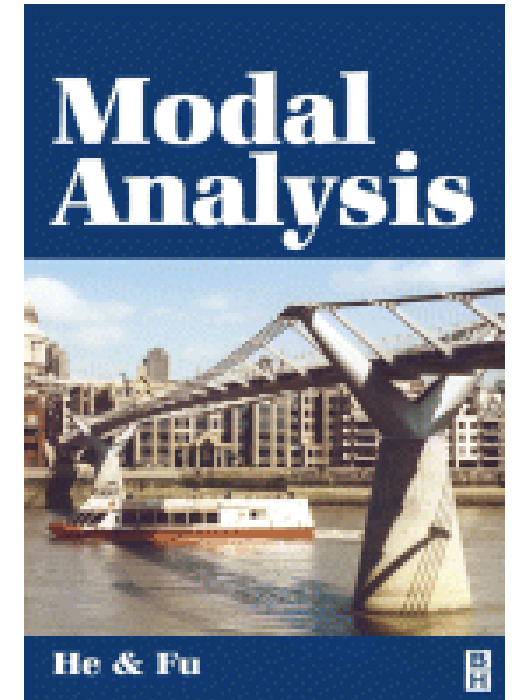
Saullo G. P. Castro

S.G.P.Castro@tudelft.nl



## What is modal analysis? (don't trust the Wikipedia!)

- Modal analysis is the process of determining the inherent dynamic characteristics of a system in forms of natural frequencies, damping factors and mode shapes, and using them to formulate a mathematical model for its dynamic behavior. The formulated mathematical model is referred to as the modal model of the system and the information for the characteristics are known as its modal data.
- The dynamics of a structure are physically decomposed by frequency and position.



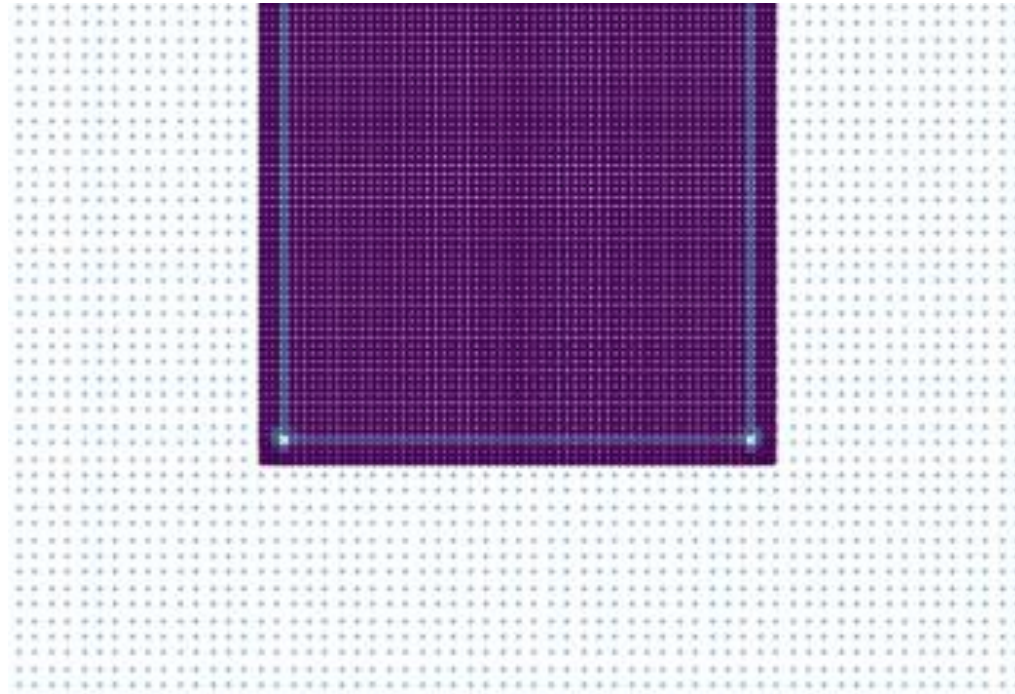
## Why Python?

- Most popular and fastest growing programming language
- Highly wanted in industry (employability)
- It is very nice to program in Python
- Highly portable (Linux, Windows, Android etc)
- Scripts can be slightly changed to perform with C-level speed
- Compared to Matlab...



# Formulation of a Structural Dynamics Problem

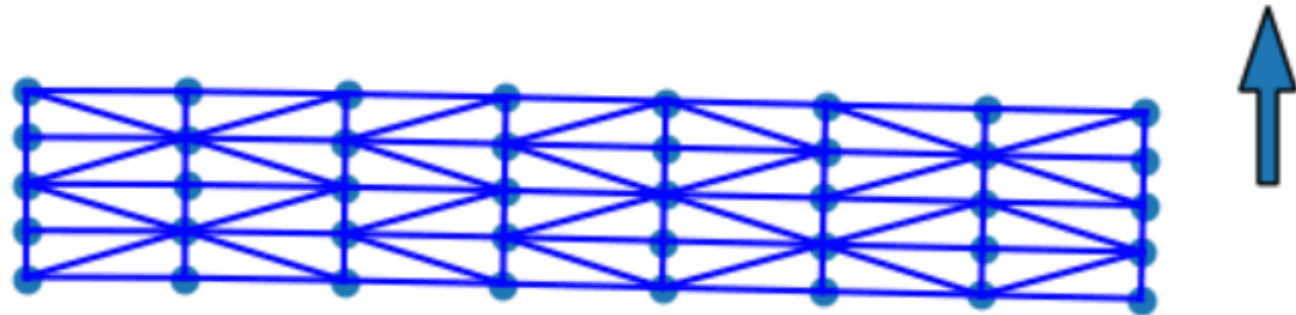
- Wave propagation
  - Interest in short-time responses
  - Propagation of disturbances along the structure



# Formulation of a Structural Dynamics Problem

- Vibrations

- Interest in long-time responses (compared to time required for waves to traverse the structure)
- Structural oscillations in a more global sense



## Downloading and citing the material from this course

Saullo G. P. Castro. "Modal Analysis with Python". COBEM 2019, Uberlândia, Brazil. [DOI:10.5281/zenodo.3514373](https://doi.org/10.5281/zenodo.3514373).



# Course Program

- Days 1: SDOF (single degree-of-freedom) systems
  - Solutions for free and forced vibration, undamped and damped
  - Harmonic and general loads
- Days 2, 3, 4: MDOF (multiple degree-of-freedom) systems
  - Generalized eigenvalue problem
  - Symmetric eigenvalue problem
  - Solution for free vibration
  - Frequency domain, experimental estimation of natural frequencies
  - Discretization of continuous systems using finite elements
  - Consistent vs. lumped mass matrix
- Day 5: Efficient implementations for large systems
  - Sparse Matrices
  - Better eigenvalue solvers
  - Compiling Python code

# Part 1 – SDOF (single degree-of-freedom) systems

Solutions for free/forced vibration of SDOF systems,  
undamped/damped, harmonic/general loading

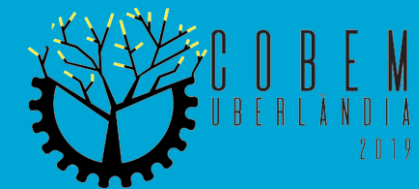


Modal Analysis  
with

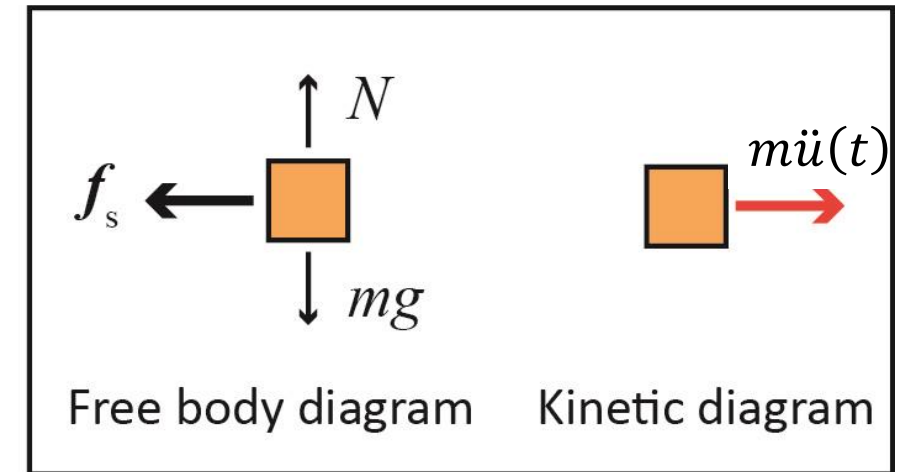
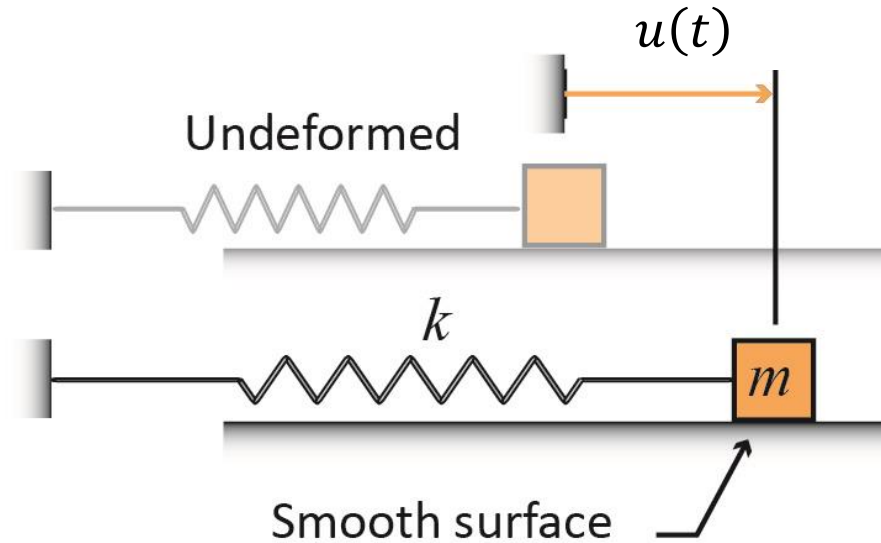


Saullo G. P. Castro

# Review on vibration of SDOF systems



# Free vibration of undamped SDOF



Equation of Motion:

$$k u(t) + m\ddot{u}(t) = 0$$

# Free vibration of undamped SDOF

$$k u(t) + m\ddot{u}(t) = 0$$

Exercise 1: Get general solution

Exercise 1 part b: Plot a particular solution

- $u(0) = 0.4$
- $\dot{u}(0) = 2$
- $k = 150$
- $m = 2$
- $0 \leq t \leq 1$

Using SymPy's plot

# Free vibration of undamped SDOF

$$k u(t) + m \ddot{u}(t) = 0$$

General solution

$$u(t) = C_1 \sin(\omega_n t) + C_2 \cos(\omega_n t)$$

$$\omega_n = \sqrt{\frac{k}{m}}$$

**Important Question: units of  $\omega_n$ ?**

# Free vibration of undamped SDOF

$$k u(t) + m\ddot{u}(t) = 0$$

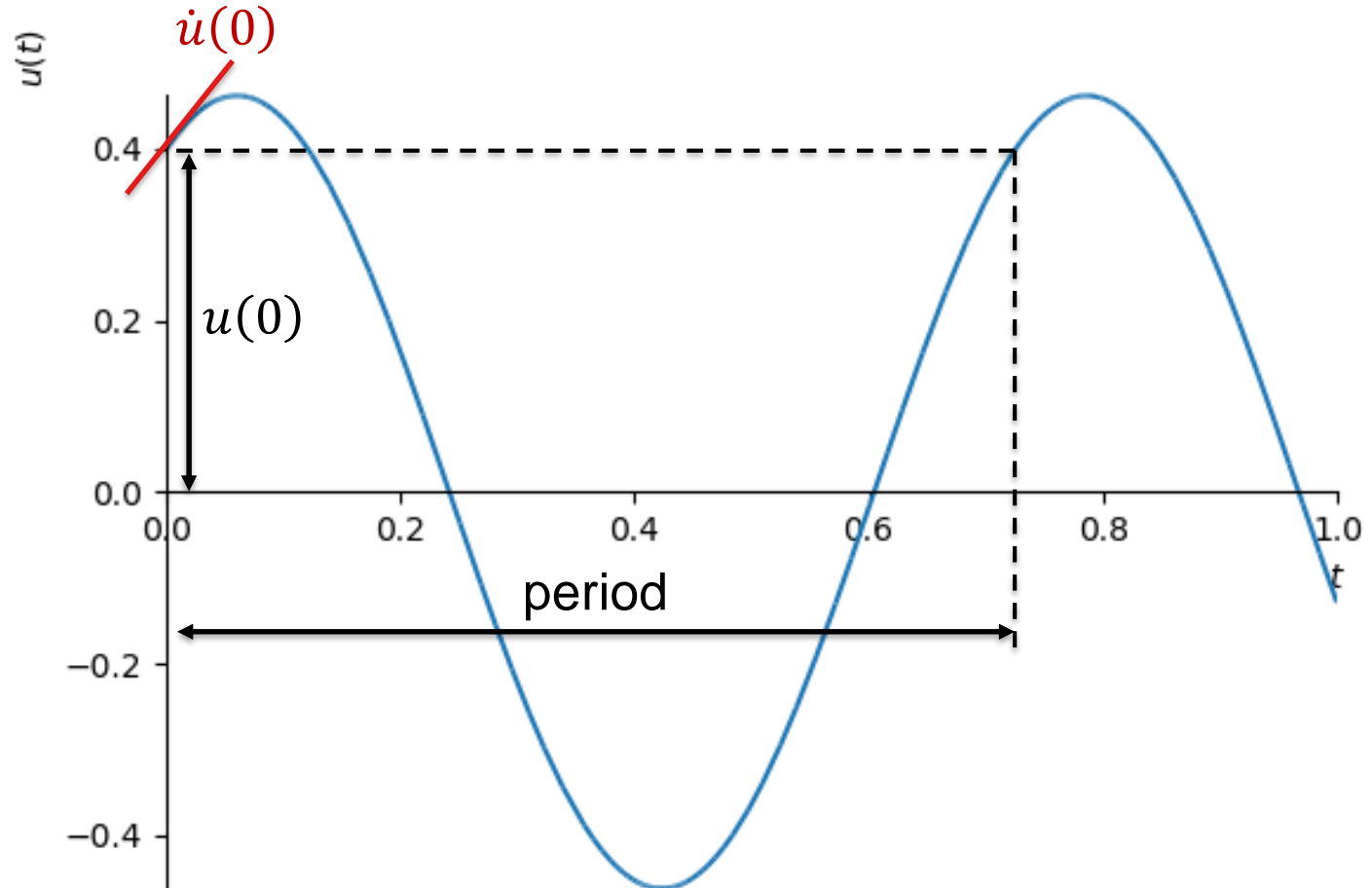
Another way to write using  $\omega_n = \sqrt{k/m}$

$$\omega_n^2 u(t) + \ddot{u}(t) = 0$$

**Mass normalized eq. of motion!**

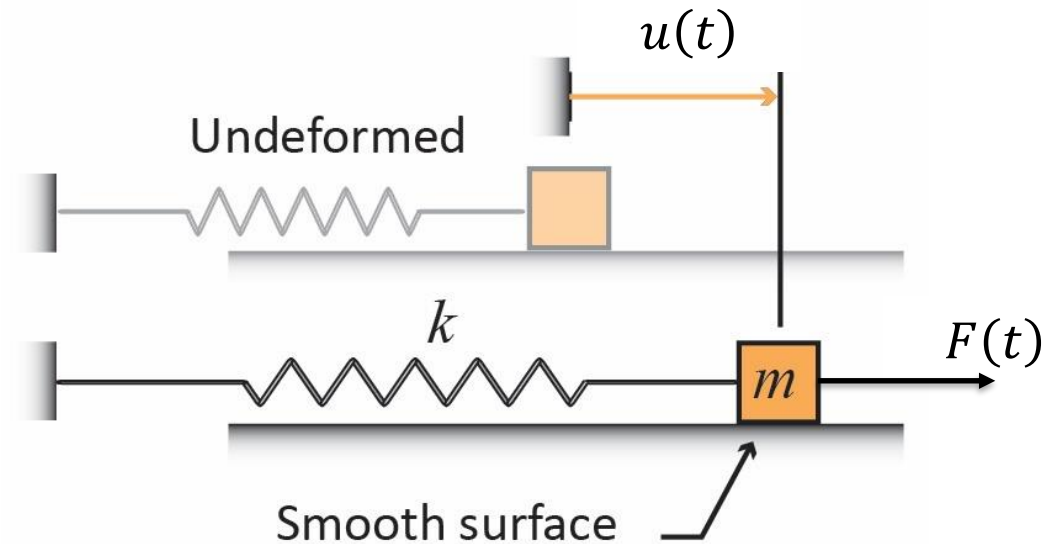
# Free vibration of undamped SDOF

Plot



$$\text{period} = \frac{2\pi}{\omega_n} \text{ (converting from s/rad to s)}$$

# Forced vibration of undamped SDOF



Equation of Motion:

$$k u(t) + m \ddot{u}(t) = F(t)$$

# Forced vibration of undamped SDOF

Equation of Motion:

$$k u(t) + m\ddot{u}(t) = F(t)$$

Harmonic force:

$$F(t) = F_0 \sin(\omega_f t)$$

Exercise 2: Get general solution



# Forced vibration of undamped SDOF

Equation of Motion:

$$k u(t) + m\ddot{u}(t) = F(t)$$

Harmonic force:

$$F(t) = F_0 \sin(\omega_f t)$$

General solution:

$$u(t) = \boxed{\frac{F_0}{m}} \frac{\sin(\omega_f t)}{\omega_n^2 - \omega_f^2} + C_1 \sin(\omega_n t) + C_2 \cos(\omega_n t)$$

# Forced vibration of undamped SDOF

Mass-normalized Equation of Motion:

$$\omega_n^2 u(t) + \ddot{u}(t) = f(t)$$

Mass-normalized harmonic force:

$$f(t) = f_0 \sin(\omega_f t)$$

$$f_0 = F_0/m$$

General solution, all mass-normalized:

$$u(t) = f_0 \frac{\sin(\omega_f t)}{\omega_n^2 - \omega_f^2} + c_1 \sin(\omega_n t) + c_2 \cos(\omega_n t)$$

# Forced vibration of undamped SDOF

General solution:

$$u(t) = \frac{F_0}{m} \frac{\sin(\omega_f t)}{\omega_n^2 - \omega_f^2} + C_1 \sin(\omega_n t) + C_2 \cos(\omega_n t)$$

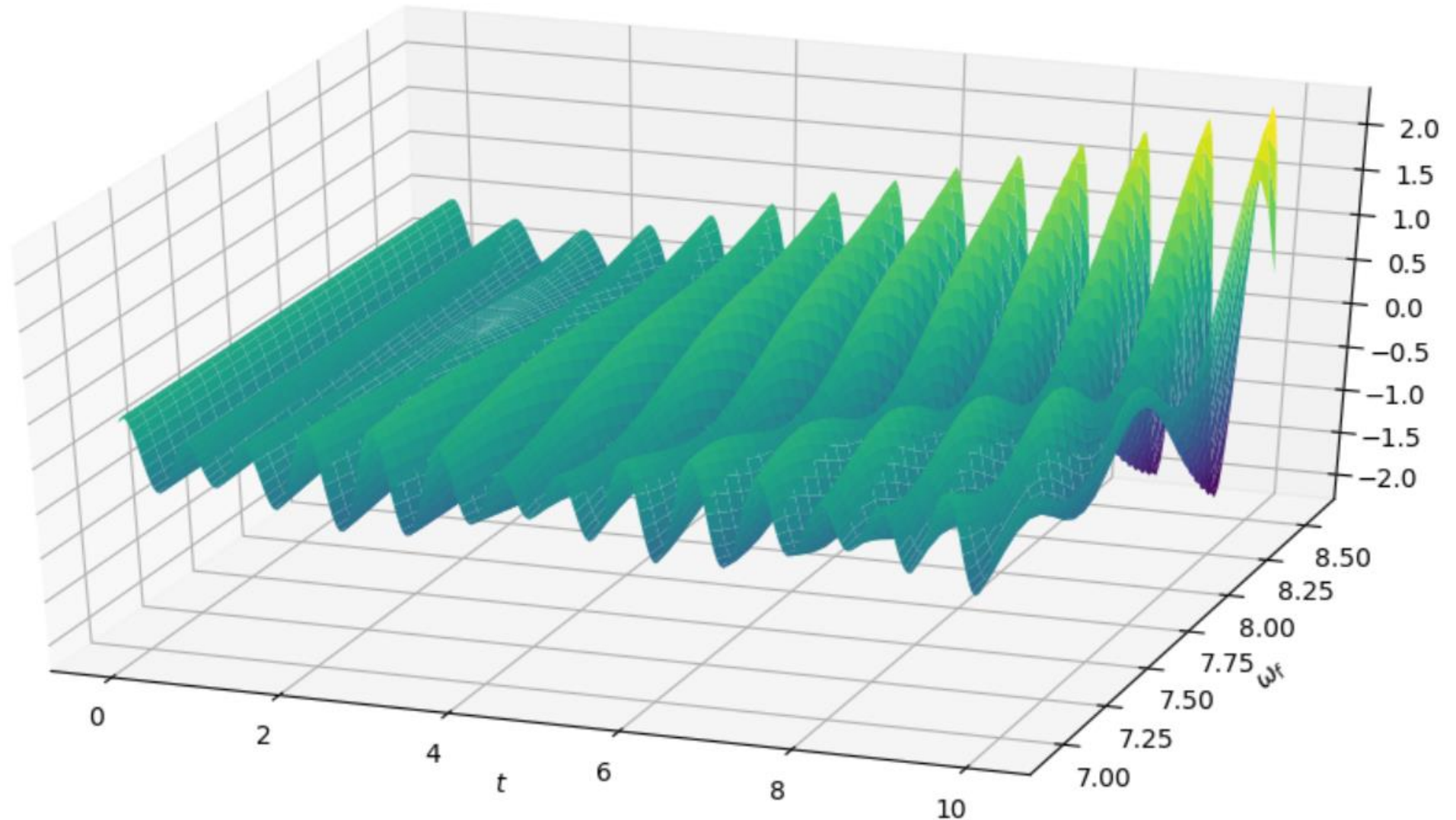
Exercise 2 part b: Plot a particular solution

- $u(0) = 0.4$
- $\dot{u}(0) = 2$
- $k = 150$
- $m = 2$
- $F_0 = 10$
- $0.8\omega_n \leq \omega_f \leq 0.99\omega_n$  (run 1 modal analysis for each)
- $0 < t < 10$

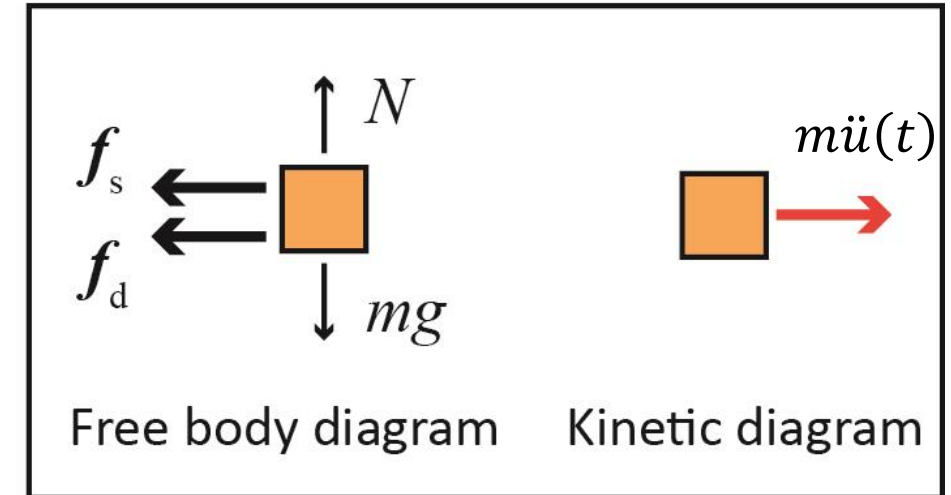
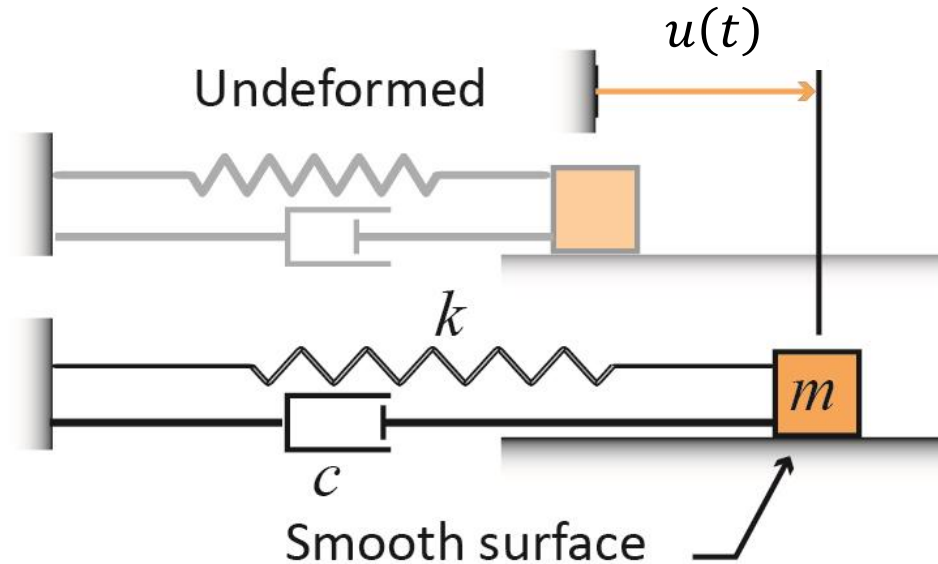
Using SymPy's plot3d

# Forced vibration of undamped SDOF

## Plot



# Free vibration of damped SDOF



Equation of Motion:

$$k u(t) + c\dot{u}(t) + m\ddot{u}(t) = 0$$

# Free vibration of damped SDOF

Equation of Motion:

$$k u(t) + c \dot{u}(t) + m \ddot{u}(t) = 0$$

Example 3: Solve using SymPy

# Free vibration of damped SDOF

Equation of Motion:

$$k u(t) + c\dot{u}(t) + m\ddot{u}(t) = 0$$

Solution

$$u(t) = C_1 e^{\frac{t(-c - \sqrt{c^2 - 4km})}{2m}} + C_2 e^{\frac{t(-c + \sqrt{c^2 - 4km})}{2m}}$$

What are the conditions to become oscillatory?

Tip:

$$\cos(\theta) = \frac{1}{2} (e^{+i\theta} + e^{-i\theta})$$

$$\sin(\theta) = \frac{1}{2i} (e^{+i\theta} - e^{-i\theta})$$

# Free vibration of damped SDOF

Equation of Motion:

$$k u(t) + c\dot{u}(t) + m\ddot{u}(t) = 0$$

Solution

$$u(t) = C_1 e^{\frac{t(-c - \sqrt{c^2 - 4km})}{2m}} + C_2 e^{\frac{t(-c + \sqrt{c^2 - 4km})}{2m}}$$

Critical damping:

$$c_{cr} = 2\sqrt{km}$$

In structures  
without dashpots  
 $\zeta$  is usually  $< 0.05$

Damping ratio:

$$\zeta = \frac{c}{c_{cr}}$$



# Free vibration of damped SDOF

Equation of Motion in terms of damping ratio:

$$k u(t) + 2\zeta\sqrt{km} \dot{u}(t) + m\ddot{u}(t) = 0$$

Mass-normalizing:

$$\omega_n^2 u(t) + 2\zeta\sqrt{k} \frac{\sqrt{m}}{m} \dot{u}(t) + \ddot{u}(t) = 0$$

$$\omega_n^2 u(t) + 2\zeta\omega_n \dot{u}(t) + \ddot{u}(t) = 0$$

Example 4: Solve using SymPy

# Free vibration of damped SDOF

$$\omega_n^2 u(t) + 2\zeta\omega_n\dot{u}(t) + \ddot{u}(t) = 0$$

Solution:

$$u(t) = \left( C_1 e^{-\omega_n t \sqrt{\zeta^2 - 1}} + C_2 e^{\omega_n t \sqrt{\zeta^2 - 1}} \right) e^{-\omega_n \zeta t}$$

What are the conditions to become oscillatory?

Tip:

$$\cos(\theta) = \frac{1}{2} (e^{+i\theta} + e^{-i\theta})$$

$$\sin(\theta) = \frac{1}{2i} (e^{+i\theta} - e^{-i\theta})$$

# Free vibration of damped SDOF

$$u(t) = \left( C_1 e^{-\omega_n t \sqrt{\zeta^2 - 1}} + C_2 e^{\omega_n t \sqrt{\zeta^2 - 1}} \right) e^{-\omega_n \zeta t}$$

What are the conditions to become oscillatory?

- Underdamped

$$0 < \zeta < 1$$
$$\sqrt{\zeta^2 - 1} \equiv i\sqrt{1 - \zeta^2}$$

$$u(t) = \left( C_1 e^{-i \omega_n \sqrt{1 - \zeta^2} t} + C_2 e^{i \omega_n \sqrt{1 - \zeta^2} t} \right) e^{-\omega_n \zeta t}$$

Damped natural frequency

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}$$

# Free vibration of damped SDOF

Underdamped solution with  $u(0) = u_0, \dot{u}(0) = v_0$  :

$$u(t) = A_0 e^{-\zeta \omega_n t} \sin(\omega_d t + \varphi)$$

$$A_0 = \sqrt{u_0^2 + \left( \zeta \frac{\omega_n}{\omega_d} u_0 + \frac{v_0}{\omega_d} \right)^2}$$

$$\varphi = \arctan \frac{\omega_d u_0}{\zeta \omega_n u_0 + v_0}$$

Tedious  
algebra alert!

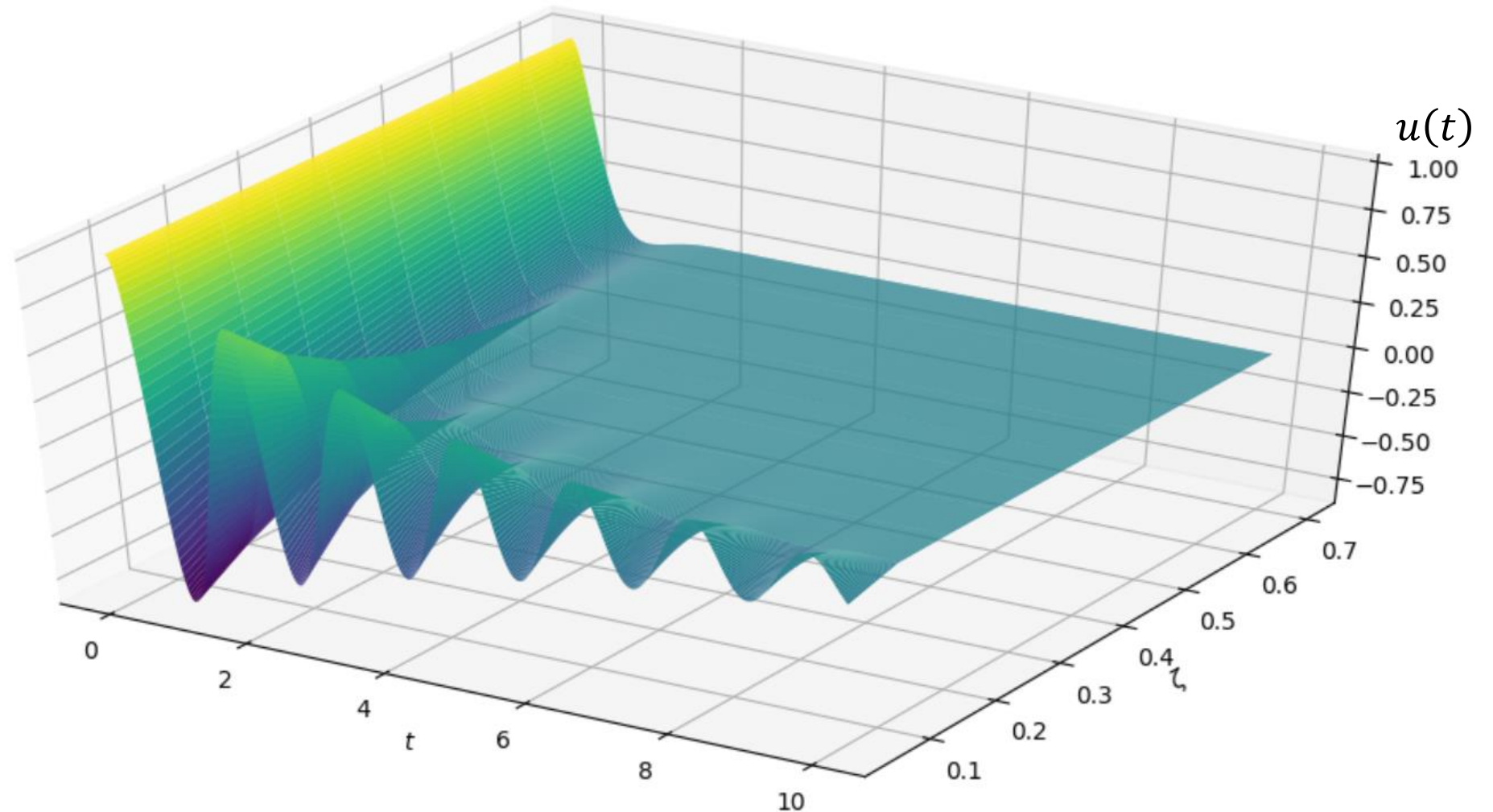
Damped natural frequency

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}$$

# Free vibration of damped SDOF

Plotting with  $\omega_n = 4$ ,  $u_0 = 1$ ,  $v_0 = 0$ ,  $0.05 \leq \zeta \leq 0.7$ ,  $0 \leq t \leq 10$ :

Script: `exercise04_underdamped_free_plot.py`



# Free vibration of damped SDOF

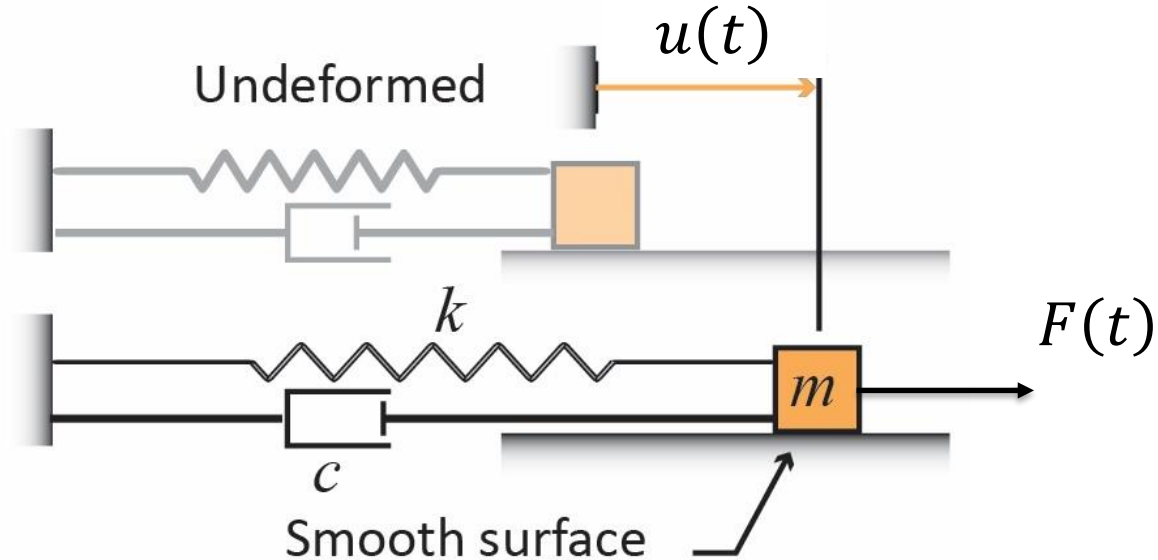
$$\omega_n^2 u(t) + 2\zeta\omega_n\dot{u}(t) + \ddot{u}(t) = 0$$

Homework for you:

- Critically damped ( $\zeta = 1$ )
- Overdamped ( $\zeta > 1$ )

NOTE: These will not be used in the rest of the course

# Forced vibration of damped SDOF



Equation of Motion:

$$k u(t) + c \dot{u}(t) + m \ddot{u}(t) = F(t)$$

# Forced vibration of damped SDOF

Equation of Motion:

$$k u(t) + c\dot{u}(t) + m\ddot{u}(t) = F(t)$$

Mass-normalized Equation of Motion:

$$\omega_n^2 u(t) + 2\zeta\omega_n\dot{u}(t) + \ddot{u}(t) = f(t)$$

Solution on next slide



# Forced vibration of damped SDOF

Assuming:

- harmonic load:  $f(t) = f_0 \cos(\omega_f t)$
- $u(0) = u_0, \dot{u}(0) = v_0$

Solution for underdamped case ( $0 < \zeta < 1$ ):

$$u(t) = u_h(t) + u_p(t)$$

$$u_h(t) = A_0 e^{-\zeta \omega_n t} \sin(\omega_d t + \varphi)$$

$$u_p(t) = |u_p| \cos(\omega_f t - \theta)$$

Tedious  
algebra alert!

$$|u_p| = \frac{f_0}{\sqrt{(\omega_n^2 - \omega_f^2)^2 + (2\zeta \omega_n \omega_f)^2}}$$

$$A_0 = \frac{u_0 - |u_p| \cos \theta}{\sin \varphi}$$

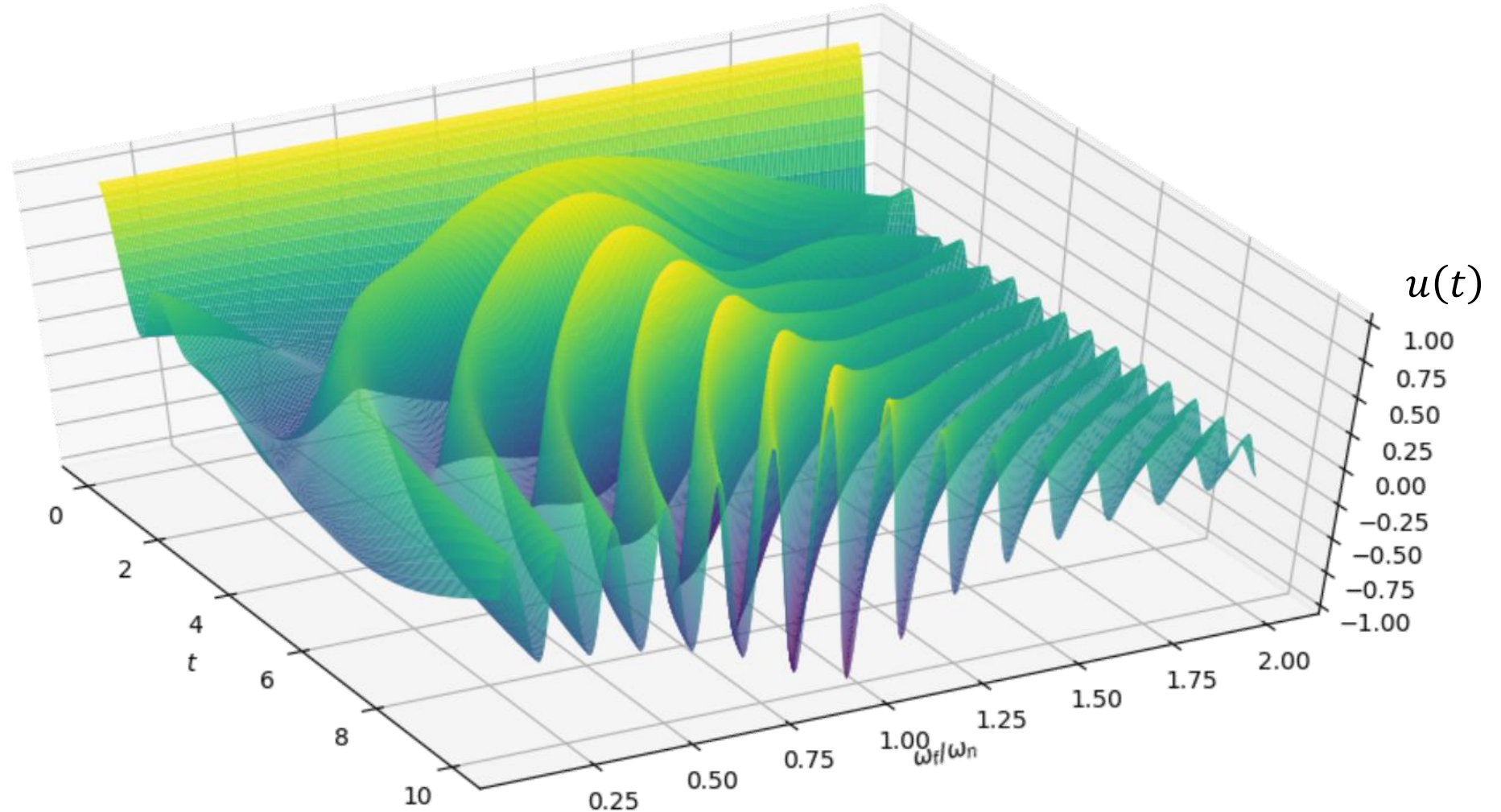
$$\varphi = \arctan \frac{\omega_d (u_0 - |u_p| \cos \theta)}{v_0 + (u_0 - |u_p| \cos \theta) \zeta \omega_n - \omega_f |u_p| \sin \theta}$$

$$\theta = \arctan \frac{2\zeta \omega_n \omega_f}{\omega_n^2 - \omega_f^2}$$

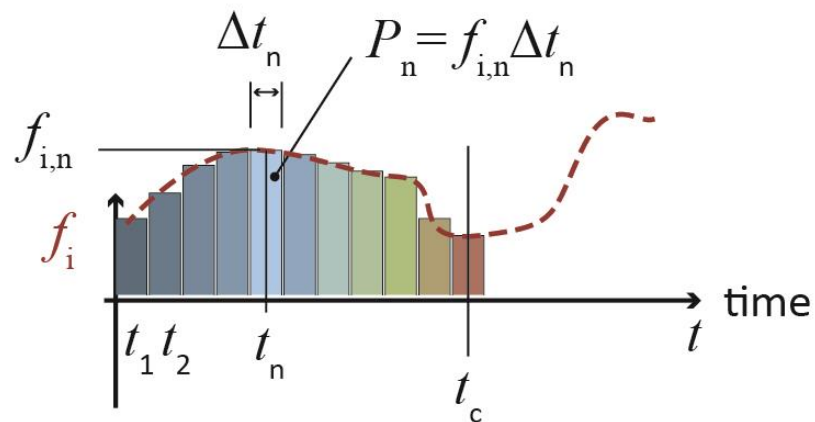
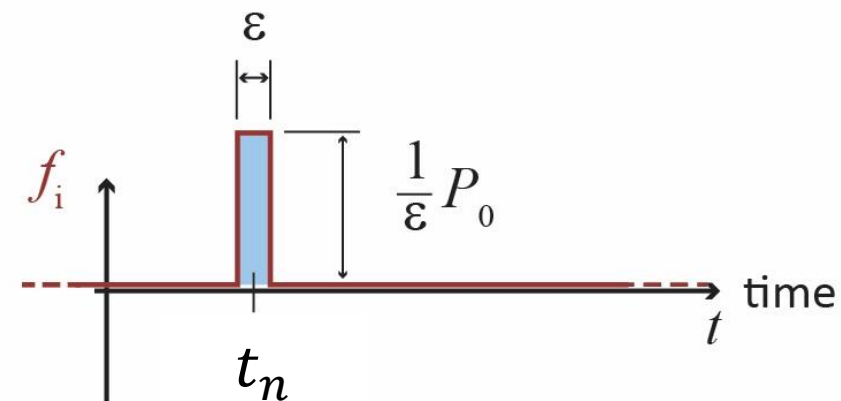
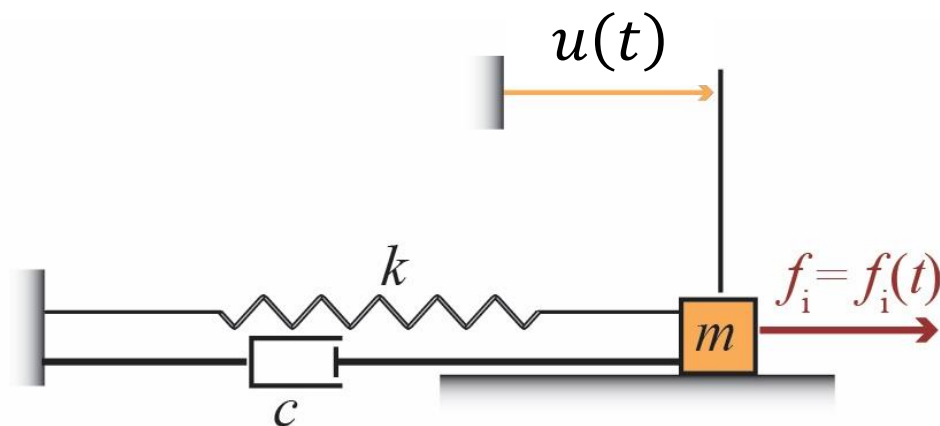
# Forced vibration of damped SDOF

Plotting (script `exercise05_harmonic_f_damped_plot.py`):

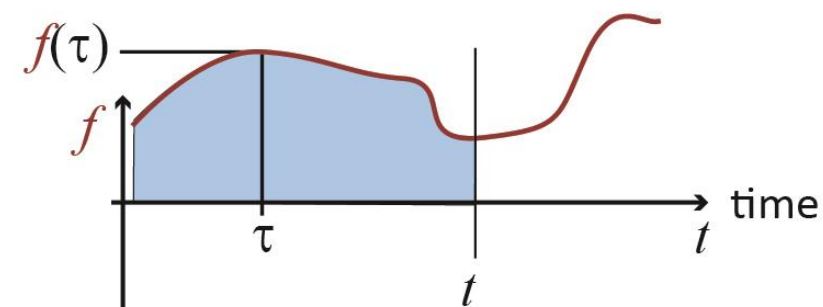
- $\omega_n = 5, u_0 = 1, v_0 = 0, f_0 = 10, \zeta = 0.2$



# Towards general load cases: Impulsive Load

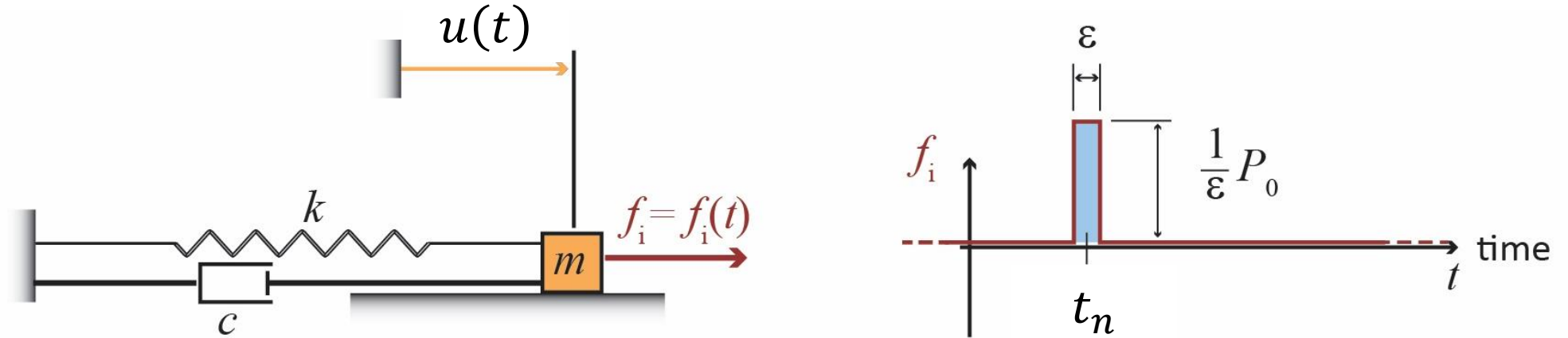


**Discrete Impulsive** loadings



**Continuous loading**

# SDOF, Damped Response for an Impulsive Load

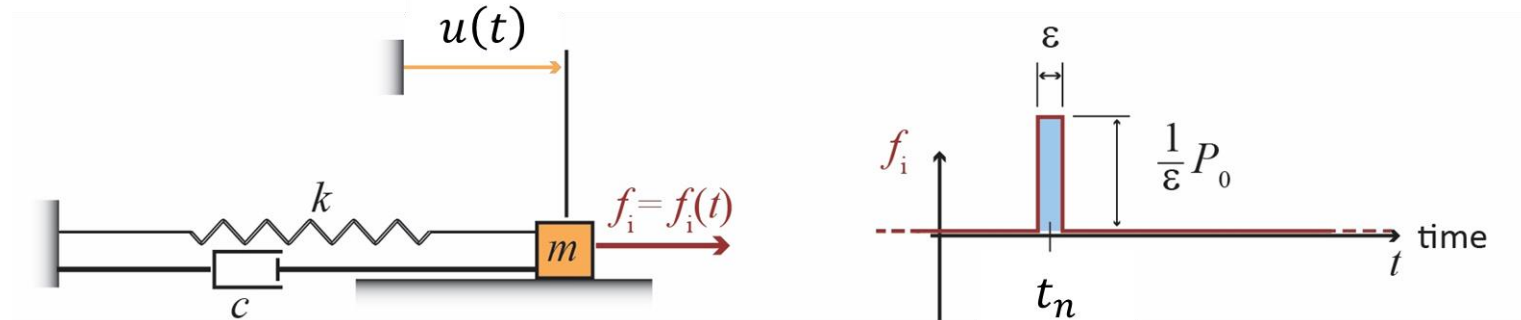


$$u(t) = \begin{cases} 0 & t < t_n \\ P_0 h(t - t_n) & t \geq t_n \end{cases}$$

$h(t - t_n) \equiv$  **unit impulse response function**

$$h(t - t_n) = \frac{1}{m\omega_d} e^{-\zeta\omega_n(t-t_n)} \sin \omega_d(t - t_n); t > t_n$$

# SDOF, Response for an Impulsive Load



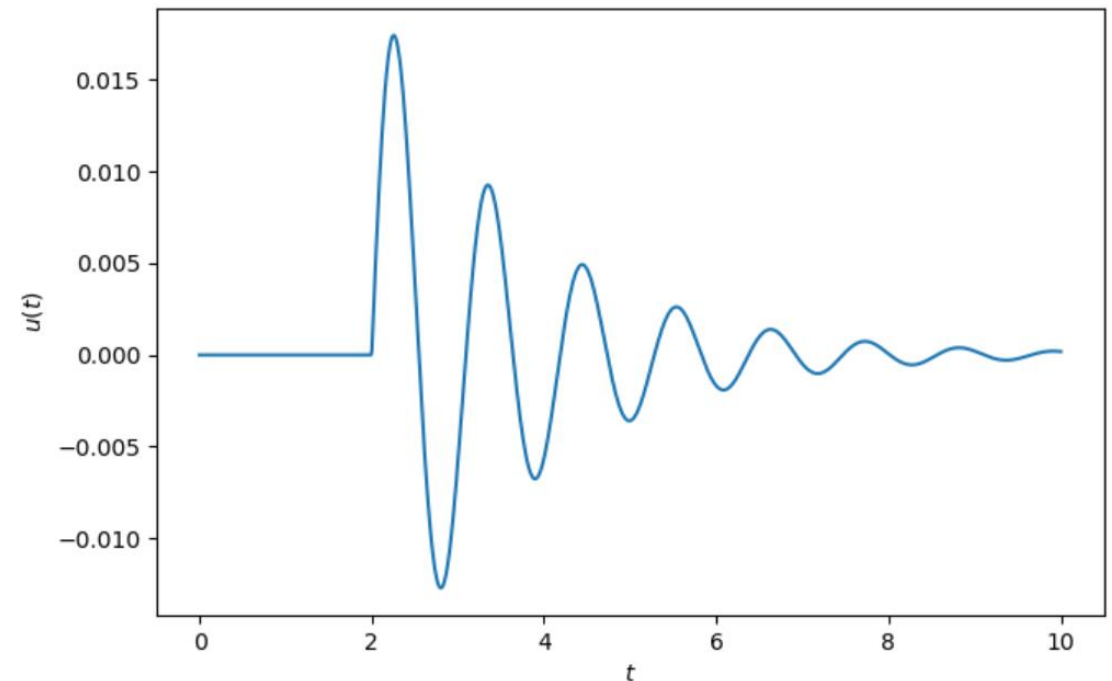
## Challenge for you:

- Find problem with scripts, probably related to the Piecewise function in SymPy

`exercise06_f_impulse_FIXME.py`  
`exercise06_f_impulse_plot_FIXME.py`

- I tried for a couple of hours without success

`exercise06_hard_coded_solution.py`



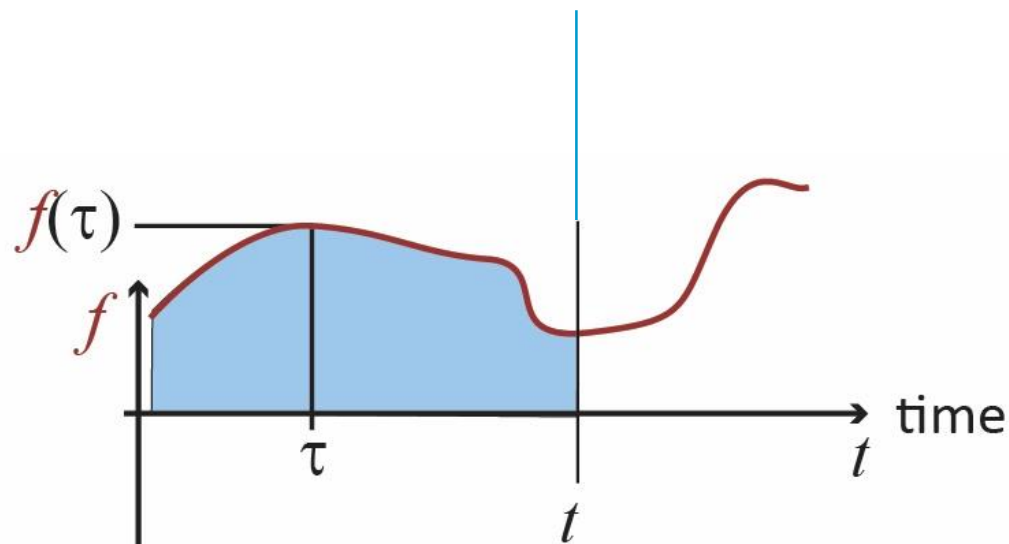
# SDOF, Response to General Loads

Duhamel integral:

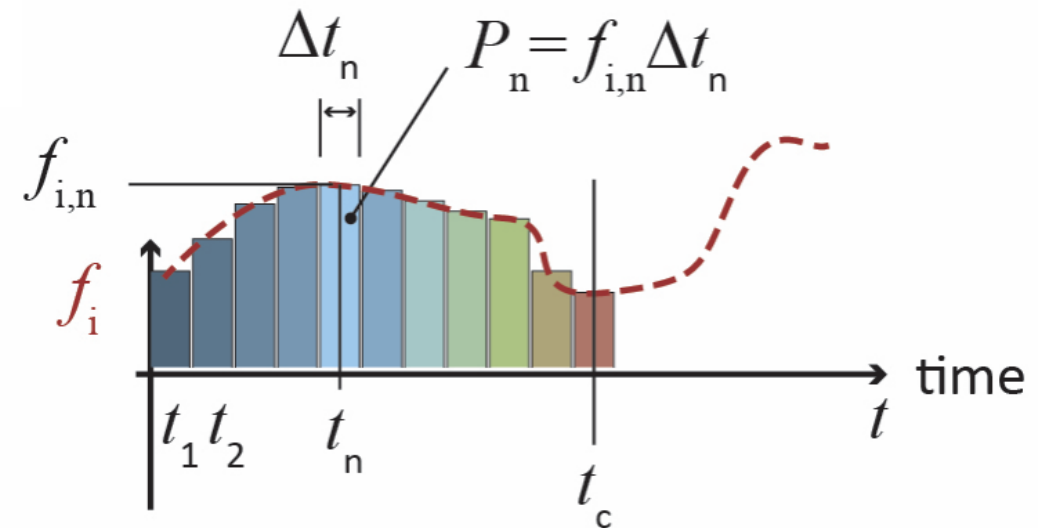
$$u(t) = \int_{-\infty}^t f(\tau) h(t - \tau) d\tau \longrightarrow$$

Approximation:

$$u(t_c) = \sum_{n=1}^c f_{i,n} h(t_c - t_n) \Delta t_n$$



**Continuous loading**



**Discrete Impulsive loadings**

# MDOF Systems

Generalized eigenvalue problem  
Orthonormal, from nodal to modal representation  
Symmetric eigenvalue problem  
Free/forced, undamped/damped vibration

# Free vibration of undamped MDOF

General formulation: for a system with  $N$  degrees of freedom:

$$M\ddot{u} + Ku = 0$$

DoF vector

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}$$

(Global) mass matrix

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1N} \\ m_{21} & m_{22} & \cdots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \cdots & m_{NN} \end{bmatrix}$$

Null vector

$$0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

(Global) stiffness matrix

$$K = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1N} \\ k_{21} & k_{22} & \cdots & k_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ k_{N1} & k_{N2} & \cdots & k_{NN} \end{bmatrix}$$

Symmetric  
and positive  
definite

$$K = K^T$$
$$k_{ij} = k_{ji}$$



# Free vibration of undamped MDOF

Special case: **lumped** mass matrix

In some cases it is possible to work with a **diagonal** matrix that may be an **approximation** to the (consistent) mass matrix

(Global) mass matrix

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1N} \\ m_{21} & m_{22} & \cdots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \cdots & m_{NN} \end{bmatrix}$$

(Lumped) mass matrix

$$M_l = \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_N \end{bmatrix}$$

# Direct solution to free vibrations of undamped MDOFs systems: The generalized eigenvalue approach

# Separation of Variables

$$M\ddot{\mathbf{u}} + K\mathbf{u} = \mathbf{0}$$

Assume a general solution of the form

$$\mathbf{u}(x, t) = \mathbf{U}(x)T(t) \quad \mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix}$$

What does this assumption imply?

Synchronous motion for all degrees of freedom  $U_i$

# Separation of Variables

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}$$

Assume a general solution of the form

$$\mathbf{u}(x, t) = \mathbf{U}(x)T(t) \quad \mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} \quad T(t) \text{ is a scalar function}$$

Calculating the derivatives:

First derivative

$$\dot{\mathbf{u}}(x, t) = \mathbf{U}(x)\dot{T}(t)$$

Second derivative

$$\ddot{\mathbf{u}}(x, t) = \mathbf{U}(x)\ddot{T}(t)$$

# Separation of Variables

$$\mathbf{M}\ddot{\mathbf{u}}(x, t) + \mathbf{K}\mathbf{u}(x, t) = \mathbf{0}$$

$$\left( \mathbf{M}\ddot{T}(t) + \mathbf{K}T(t) \right) \mathbf{U}(x) = \mathbf{0}$$

Gives  $N$  systems of equations, with  $I = 1, 2 \dots N$

$$-\frac{\ddot{T}(t)}{T(t)} = \frac{\sum_{J=1}^N k_{IJ} U_J(x)}{\sum_{J=1}^N m_{IJ} U_J(x)} = \text{constant} = \omega_n^{(I)^2}$$

$$\ddot{T}(t) + \omega_n^{(I)^2} T(t) = 0$$

$$\ddot{T}(t) = -\omega_n^{(I)^2} T(t)$$

# Generalized Eigenvalue Problem

$$\left(-\omega_n^{(I)2} \mathbf{M} + \mathbf{K}\right) \mathbf{U}^{(I)} T(t) = \mathbf{0}$$

$$\underbrace{\left(-\omega_n^{(I)2} \mathbf{M} + \mathbf{K}\right)}_{\text{Singular}} \mathbf{U}^{(I)} = \mathbf{0}$$

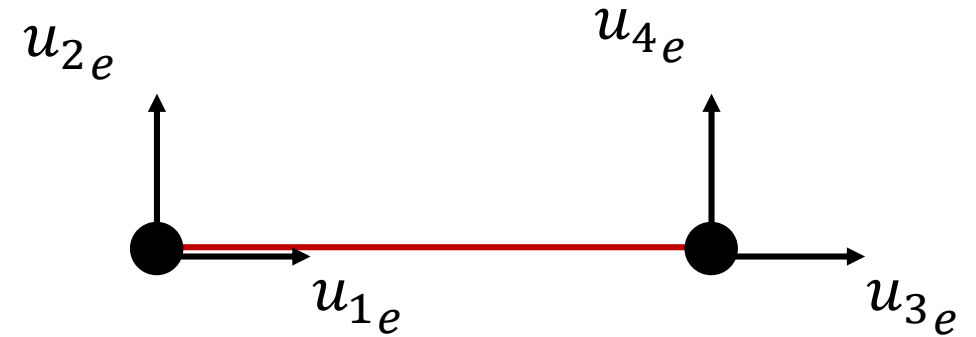
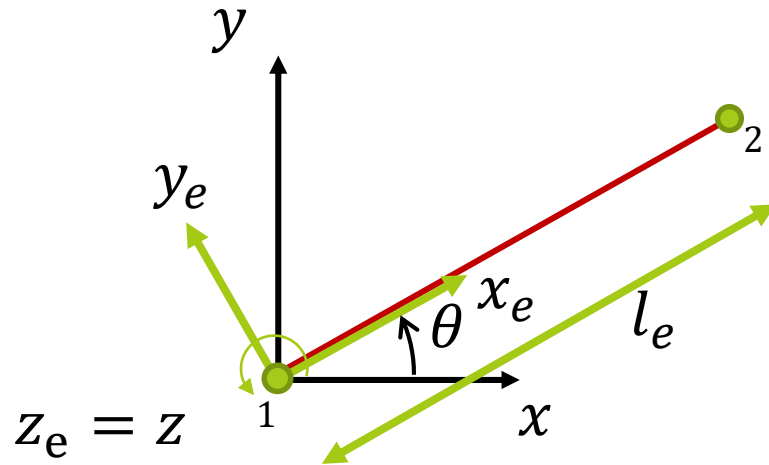
Singular

$$\det\left(-\omega_n^{(I)2} \mathbf{M} + \mathbf{K}\right) = 0$$

**NOTE:**  $\mathbf{U}^{(I)}$  represents a mode shape corresponding to  $\omega_n^{(I)}$  and it is not  $\mathbf{U}(x)$  in the general solution  $\mathbf{u}(x, t) = \mathbf{U}(x)T(t)$

# Structural Matrices for the 2D Truss Element

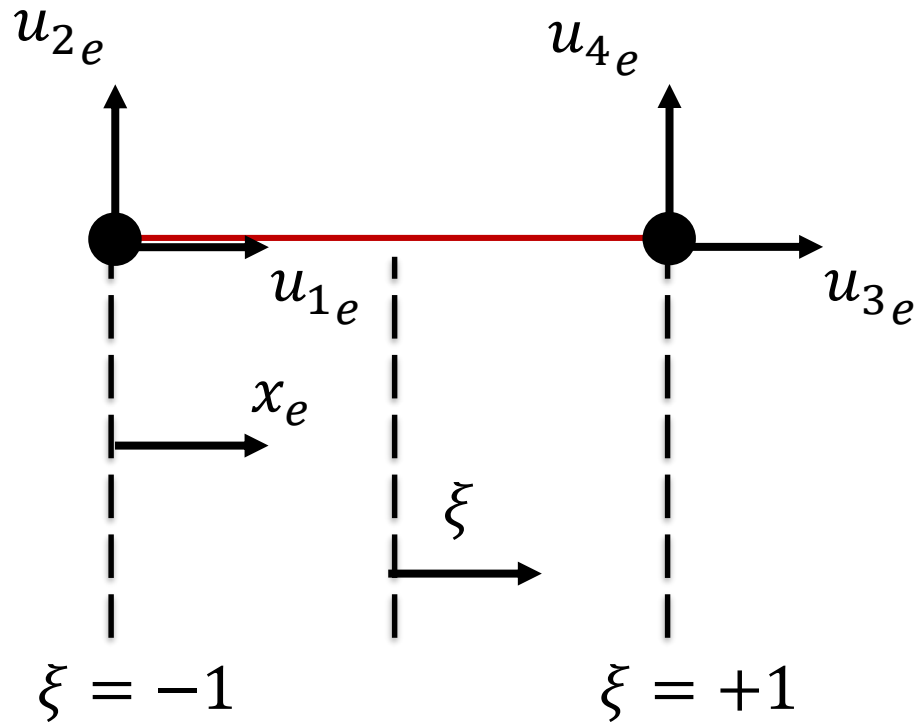
Element and global coordinates



$$\begin{Bmatrix} u_{1e} \\ u_{2e} \\ u_{3e} \\ u_{4e} \end{Bmatrix} = [R] \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

$$[R] = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & \cos \theta & \sin \theta \\ 0 & 0 & -\sin \theta & \cos \theta \end{bmatrix}$$

# Structural Matrices for the 2D Truss Element



Displacement:

$$u_{x_e} = \frac{(1 - \xi)}{2} u_{1e} + \frac{(1 + \xi)}{2} u_{3e}$$

Strain:

$$\varepsilon_{xx_e} = \frac{\partial u}{\partial x_e}$$

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial \xi} \frac{\partial \xi}{\partial x_e} = \frac{2}{\ell_e} \frac{\partial}{\partial \xi}$$

$$\varepsilon_{xx_e} = \frac{2}{\ell_e} \left( -\frac{1}{2} \right) u_{1e} + \frac{2}{\ell_e} \left( \frac{1}{2} \right) u_{3e}$$

$$\xi = \frac{2x_e}{\ell_e} - 1$$





# Structural Matrices for the 2D Truss Element

Displacement:



$$u_{x_e} = \frac{(1 - \xi)}{2} u_{1_e} + \frac{(1 + \xi)}{2} u_{3_e}$$

$$u_{y_e} = 0$$

  $N_1$         $N_2$

$$\begin{Bmatrix} u_{x_e} \\ u_{y_e} \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_{1_e} \\ u_{2_e} \\ u_{3_e} \\ u_{4_e} \end{Bmatrix}$$

Strain:

  $N_{1,x_e}$         $N_{2,x_e}$

$$\{\varepsilon\} = [B_L] \begin{Bmatrix} u_{1_e} \\ u_{2_e} \\ u_{3_e} \\ u_{4_e} \end{Bmatrix}$$

$$\varepsilon_{xx_e} = \frac{2}{\ell_e} \left( -\frac{1}{2} \right) u_{1_e} + \frac{2}{\ell_e} \left( \frac{1}{2} \right) u_{3_e}$$

$$[B_L] = \begin{bmatrix} N_{1,x_e} & 0 & N_{2,x_e} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Structural Matrices for the 2D Truss Element

Stiffness Matrix:

$$[K]_e = \int_{x_e} EA [B_L]^T [B_L] dx_e$$

$$[K]_e = \frac{\ell_e}{2} \int_{\xi} EA [B_L]^T [B_L] d\xi$$

$$[K]_G = [R]^T [K]_e [R]$$

`derive_finite_element_matrices\fem_derive_truss2d`

# Structural Matrices for the 2D Truss Element

Consistent Mass Matrix:

$$[M]_e = \int_{x_e} A\rho[N]^T[N] dx_e$$

$$[M]_e = \frac{\ell_e}{2} \int_{\xi=-1}^{\xi=+1} A\rho[N]^T[N] d\xi$$

$$[M]_e = \frac{A\ell_e\rho}{6} \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$

$$[M]_G = [R]^T[M]_e[R]$$

`derive_finite_element_matrices\fem_derive_truss2d`

# Structural Matrices for the 2D Truss Element

## Lumped Mass Matrix:

For a truss with area  $A(x)$  varying over the axial direction  $x$  (note that it also works with  $A$  constant):

$$m_A = \int_{x=0}^{\ell_e/2} \rho A(x) dx$$

$$m_B = \int_{x=\ell_e/2}^{\ell_e} \rho A(x) dx$$

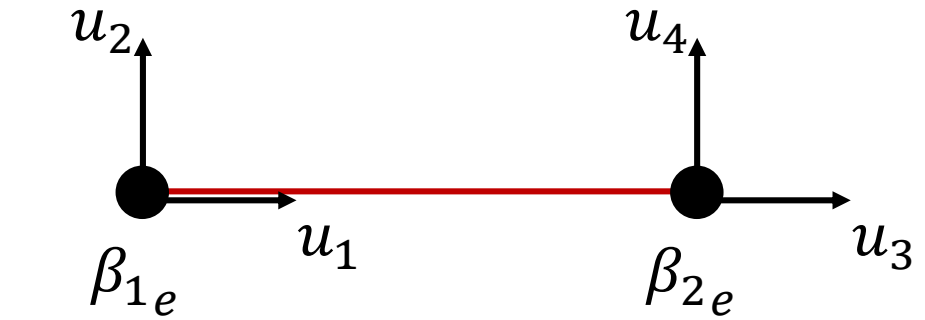
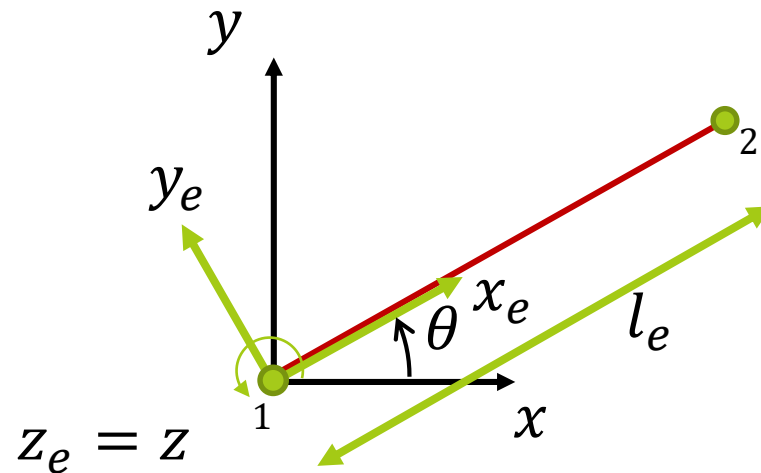
$$[M]_e = \begin{bmatrix} m_A & & & 0 \\ & m_A & & \\ & & m_B & \\ 0 & & & m_B \end{bmatrix}$$

$$[M]_G = [R]^T [M]_e [R]$$

`derive_finite_element_matrices\fem_derive_truss2d`

# Structural Matrices for the 2D Beam Element

Element and global coordinates



$$\begin{Bmatrix} u_{1e} \\ u_{2e} \\ \beta_{1e} \\ u_{3e} \\ u_{4e} \\ \beta_{2e} \end{Bmatrix} = [R] \begin{Bmatrix} u_1 \\ u_2 \\ \beta_1 \\ u_3 \\ u_4 \\ \beta_2 \end{Bmatrix}$$

$$[R] = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos \theta & \sin \theta & 0 \\ 0 & 0 & 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

# Structural Matrices for the 2D Beam Element

## Displacement

$$\{u_e\} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 \\ 0 & N_1 & N_{1\beta} & 0 & N_2 & N_{2\beta} \\ 0 & -\frac{2}{\ell_e} \frac{dN_1}{d\xi} & -\frac{2}{\ell_e} \frac{dN_{1\beta}}{d\xi} & 0 & -\frac{2}{\ell_e} \frac{dN_2}{d\xi} & -\frac{2}{\ell_e} \frac{dN_{2\beta}}{d\xi} \end{bmatrix} \begin{Bmatrix} u_{1e} \\ u_{2e} \\ \beta_{1e} \\ u_{3e} \\ u_{4e} \\ \beta_{2e} \end{Bmatrix}$$

$$\{u_e\} = \begin{bmatrix} N^u \\ N^v \\ N^\beta \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \beta_1 \\ u_3 \\ u_4 \\ \beta_2 \end{Bmatrix}$$

# Structural Matrices for the 2D Beam Element

Stiffness Matrix:

$$\{u_e\} = \begin{bmatrix} N^u \\ N^v \\ N^\beta \end{bmatrix} \begin{Bmatrix} u_{1e} \\ u_{2e} \\ \beta_{1e} \\ u_{3e} \\ u_{4e} \\ \beta_{2e} \end{Bmatrix}$$

$$[K]_e = \frac{2}{\ell_e} \int_{\xi} \left( E I_{zz} [N^\beta]^T [N^\beta] + EA [N^u]^T [N^u] \right) d\xi$$

# Structural Matrices for the 2D Beam Element

Consistent Mass Matrix:

$$[M]_e = \frac{\ell_e}{2} \int_{\xi} \rho \left( A [N^u]^T [N^u] + A [N^v]^T [N^v] + I_{zz} [N^{\beta}]^T [N^{\beta}] \right) d\xi$$



# Structural Matrices for the 2D Beam Element

## Lumped Mass Matrix:

For a beam with area  $A(x)$  and moment of inertia  $I_{zz}(x)$  varying over the axial direction  $x$  (note that it also works with  $A$  and  $I_{zz}$  constant):

$$m_A = \int_{x=0}^{\ell_e/2} \rho A(x) dx$$

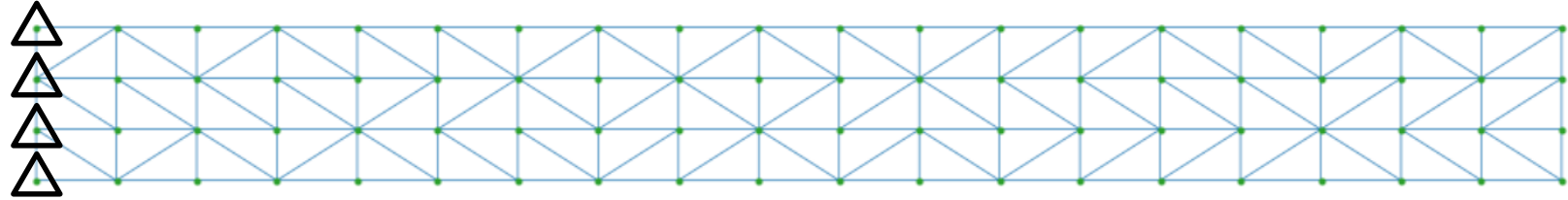
$$\begin{aligned} I_{zzA} &= \int_{x=0}^{\ell_e/2} I_{zz}(x) dx + \int_{x=0}^{\ell_e/2} x^2 dm \\ &= \int_{x=0}^{\ell_e/2} (I_{zz}(x) + x^2 \rho A(x)) dx \end{aligned}$$

$$m_B = \int_{x=\ell_e/2}^{\ell_e} \rho A(x) dx$$

$$I_{zzB} = \int_{x=\ell_e/2}^{\ell_e} (I_{zz}(x) + (\ell_e - x)^2 \rho A(x)) dx$$

$$[M]_e = \begin{bmatrix} m_A & & & & 0 \\ & m_A & & & \\ & & I_{zzA} & & \\ & & & m_B & \\ 0 & & & & m_B & I_{zzB} \end{bmatrix}$$

# Example of Generalized Eigenvalue Problem



- $\rho = 2.6e3$
- $E = 70e9$
- $A = 0.01^2$
- length= 10
- width=1
- Solve a generalized eigenvalue problem
- Find natural frequencies
- Plot first 5 mode shapes

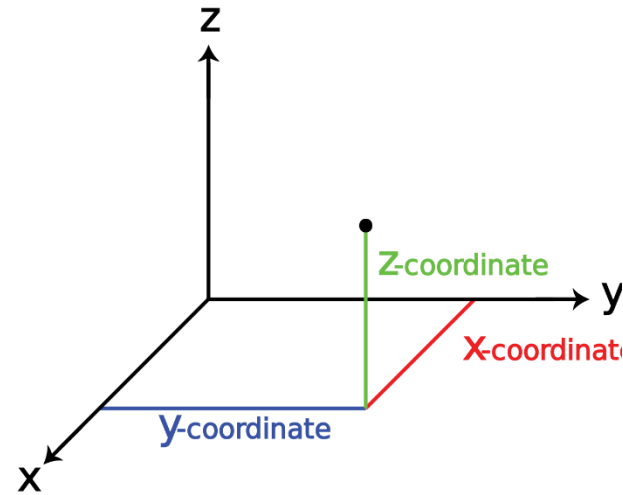
Script `exercise07_generalized_eigenvalue_problem.py`  
compare `lumped=True/False` (will be explained later)

# Next Steps...

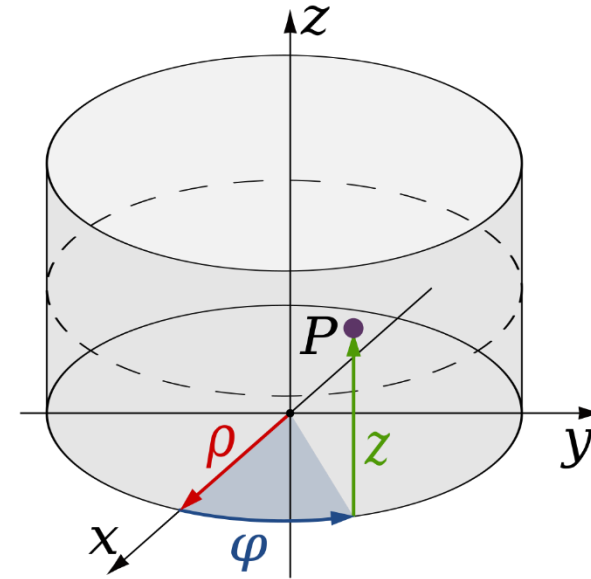
- Now we know how to find  $\omega_n^{(I)}$  and the corresponding mode shapes  $\mathbf{U}^{(I)}$  for a MDOF system
- Are these modes orthonormal?
- Why is this important?

**NOTE:**  
Orthonormal is  
Orthogonal and Normalized

# Orthonormal Basis in 3D Space



$$\mathbf{v} = v_x \mathbf{i} + v_y \mathbf{j} + v_z \mathbf{k}$$



$$\mathbf{v} = v_r \mathbf{u}_r + v_\phi \mathbf{u}_\phi + v_z \mathbf{u}_z$$

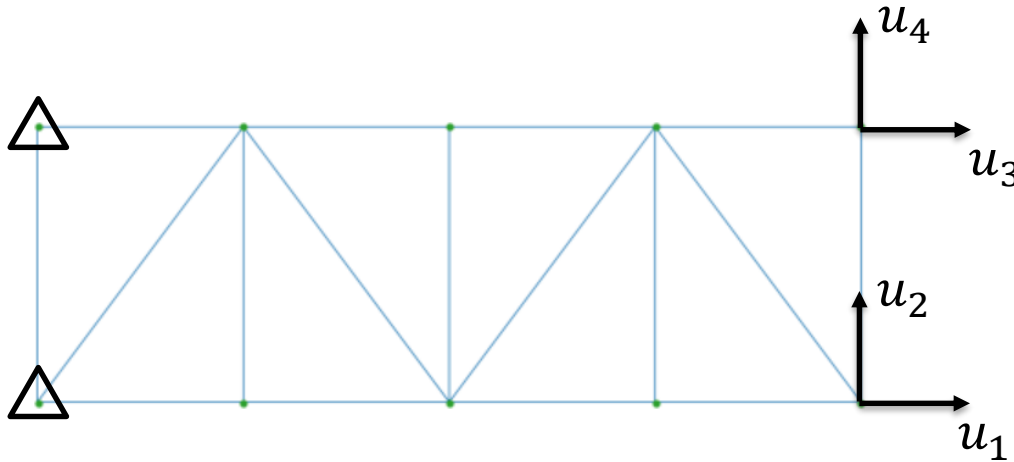
$$\mathbf{i} = \{1, 0, 0\}^T$$

$$\mathbf{j} = \{0, 1, 0\}^T$$

$$\mathbf{k} = \{0, 0, 1\}^T$$

Orthonormal Basis:  $\mathbf{i} \cdot \mathbf{j} = 0, \mathbf{i} \cdot \mathbf{k} = 0, \mathbf{j} \cdot \mathbf{k} = 0$

# Orthonormal Basis in n-dimensional Space



Consider:

$$\mathbf{u}^{(1)} = \{u_1^{(1)}, u_2^{(1)}, u_3^{(1)}, u_4^{(1)}, \dots, u_N^{(1)}\}^T$$

$$\mathbf{u}^{(2)} = \{u_1^{(2)}, u_2^{(2)}, u_3^{(2)}, u_4^{(2)}, \dots, u_N^{(2)}\}^T$$

$\vdots$

$$\mathbf{u}^{(N)} = \{u_1^{(N)}, u_2^{(N)}, u_3^{(N)}, u_4^{(N)}, \dots, u_N^{(N)}\}^T$$

Assuming they are  
orthonormal:

$$\mathbf{u}^{(I)} \cdot \mathbf{u}^{(J)} = 0 \\ \text{for } I \neq J$$

Any vector in this MDOF space can be  
represented as a linear combination of  
different orthonormal  $\mathbf{u}^{(I)}$ :

$$\mathbf{u} = \sum_I^N c_I \mathbf{u}^{(I)}$$

# Expansion Theorem

Orthogonal vectors can form a new basis in the n-dimensional space of the MDOF system.

Therefore, any vector in the n-dimensional space can be represented as a linear combination of the n linearly independent eigenvectors.

$$\mathbf{u} = \sum_I^N c_I \mathbf{u}^{(I)}$$

Note that **we are changing from nodal coordinates in our MDOF to modal coordinates**, using the new n-dimensional basis formed by the orthonormal eigenmodes.

# From nodal coordinates to modal coordinates

Nodal coordinates:

$$\mathbf{u} = \{u_1, u_2, u_3, u_4, \dots, u_N\}^T$$

$N$  unknowns  $u_1, u_2, \dots$

Modal coordinates:

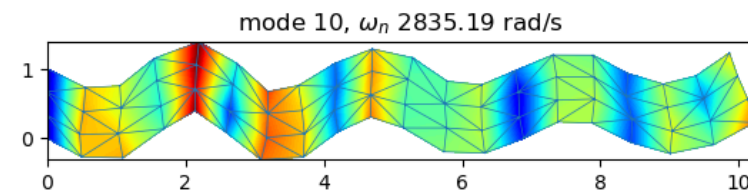
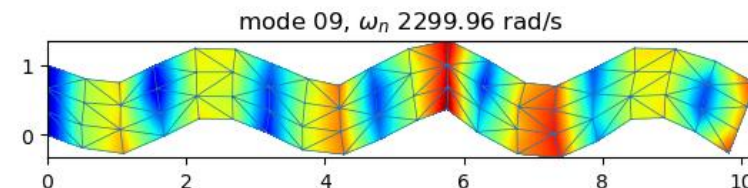
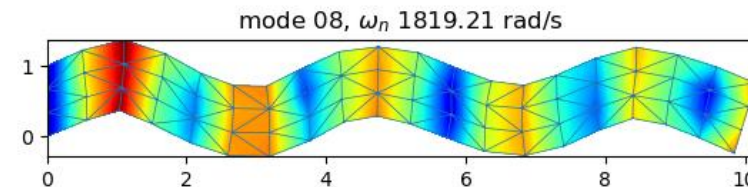
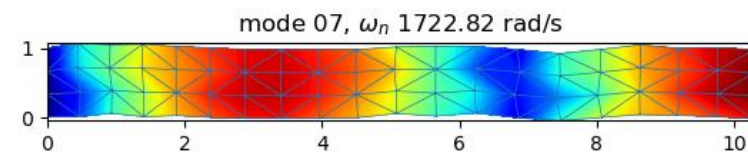
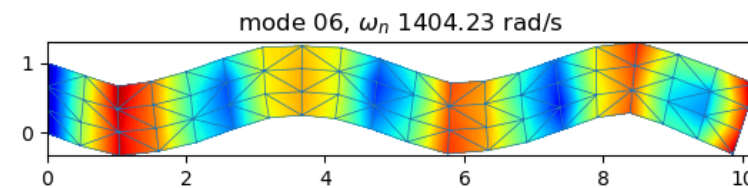
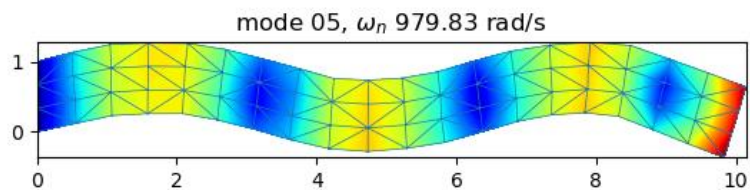
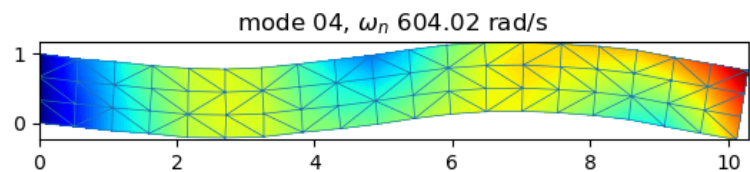
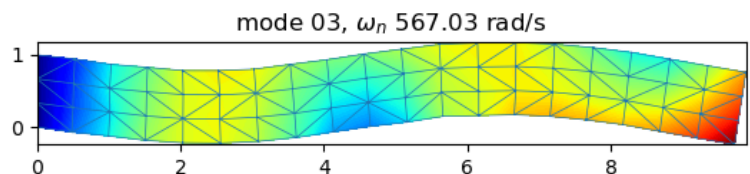
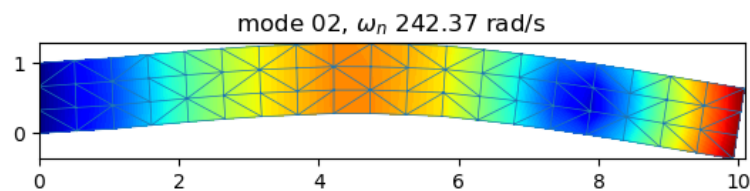
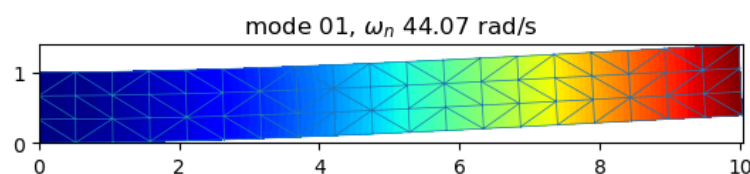
$$\mathbf{u} = \sum_I^N c_I \mathbf{u}^{(I)}$$

$N$  unknowns  $c_1, c_2, \dots$

so... What is the advantage?

# From nodal coordinates to modal coordinates

What is the advantage?





# Next Steps...

- Make our mode shapes become a valid orthonormal basis

$$\mathbf{u}^{(I)} \cdot \mathbf{u}^{(J)} = 0 \\ \text{for } I \neq J$$

# Orthonormalization of Mode Shapes

$$\left(-\omega_n^{(I)^2} \mathbf{M} + \mathbf{K}\right) \mathbf{U}^{(I)} = \mathbf{0}$$

Rewriting as:

$$\omega_n^{(I)^2} \mathbf{M} \mathbf{U}^{(I)} = \mathbf{K} \mathbf{U}^{(I)}$$

Taking another mode:

$$\omega_n^{(J)^2} \mathbf{M} \mathbf{U}^{(J)} = \mathbf{K} \mathbf{U}^{(J)}$$

# Orthonormalization of Eigenmodes

Left-multiplying by  $\mathbf{U}^{(J)T}$

$$\mathbf{U}^{(J)T} \omega_n^{(I)2} \mathbf{M} \mathbf{U}^{(I)} = \mathbf{U}^{(J)T} \mathbf{K} \mathbf{U}^{(I)}$$

Left-multiplying by  $\mathbf{U}^{(I)T}$

$$\mathbf{U}^{(I)T} \omega_n^{(J)2} \mathbf{M} \mathbf{U}^{(J)} = \mathbf{U}^{(I)T} \mathbf{K} \mathbf{U}^{(J)}$$

Note that:

$$\mathbf{U}^{(J)T} \mathbf{K} \mathbf{U}^{(I)} = \mathbf{U}^{(I)T} \mathbf{K} \mathbf{U}^{(J)}$$

$$\mathbf{U}^{(J)T} \mathbf{M} \mathbf{U}^{(I)} = \mathbf{U}^{(I)T} \mathbf{M} \mathbf{U}^{(J)}$$

Subtracting:

$$\left( \omega_n^{(I)2} - \omega_n^{(J)2} \right) \mathbf{U}^{(J)T} \mathbf{M} \mathbf{U}^{(I)} = 0$$

# Orthonormalization of Eigenmodes

Thus, for  $I \neq J$ :

$$\mathbf{U}^{(J)T} \mathbf{M} \mathbf{U}^{(I)} = 0$$

For  $I = J$ :

$$\mathbf{U}^{(I)T} \mathbf{M} \mathbf{U}^{(I)} = 1$$

Thus:

$$\mathbf{U}^{(J)T} \mathbf{M} \mathbf{U}^{(I)} = \delta_{IJ}$$

and:

$$\mathbf{U}^{(J)T} \mathbf{K} \mathbf{U}^{(I)} = \delta_{IJ} \omega_n^{(I)2}$$

Script `exercise08_orthonormality_checks.py`

Do we already have a orthonormal basis with  $\mathbf{U}^{(I)}$ ?

# Orthonormalization of Eigenmodes

We can conclude that Scipy's `eigh` already gives the mass-normalized eigenvectors:

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.eigh.html#scipy.linalg.eigh>

Do we already have a orthonormal basis with  $\mathbf{U}^{(I)}$ ?

# Quick overview ...

- We did separation of variables assuming that all DOFs are synchronous:

$$\mathbf{u}(x, t) = \mathbf{U}(x)T(t)$$

- Then replaced this into  $\mathbf{K}\mathbf{u} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{0}$  to get:

$$\ddot{T}(t) + \omega_n^{(I)^2} T(t) = 0$$

- Which has the general solution:

$$T^{(I)}(t) = C_1^{(I)} \cos \omega_n^{(I)} t + C_2^{(I)} \sin \omega_n^{(I)} t$$

- We are seeking an orthonormal basis that will allow us to do:

$$\mathbf{U} = \sum_I^N c_I \mathbf{U}^{(I)}$$

- Such that the final solution will have the format:

$$\mathbf{u}(x, t) = \sum_I^N \left( c_1^{(I)} \cos \omega_n^{(I)} t + c_2^{(I)} \sin \omega_n^{(I)} t \right) \mathbf{U}^{(I)}$$

# Symmetric Eigenvalue Problem

Expected format:

$$(\lambda^{(I)} \mathbf{I} + \mathbf{A}) \mathbf{V}^{(I)} = \mathbf{0}, \quad \text{with } A_{ij} = A_{ji}$$

Our generalized eigenvalue problem:

$$(-\omega_n^{(I)^2} \mathbf{M} + \mathbf{K}) \mathbf{U}^{(I)} = \mathbf{0}$$

What can we do?

# Symmetric Eigenvalue Problem

Expected format:

$$(\lambda^{(I)} \mathbf{I} + \mathbf{A}) \mathbf{V}^{(I)} = \mathbf{0}, \quad \text{with } A_{ij} = A_{ji}$$

Our generalized eigenvalue problem:

$$\left( -\omega_n^{(I)^2} \mathbf{M} + \mathbf{K} \right) \mathbf{U}^{(I)} = \mathbf{0}$$

Left-multiplying by  $\mathbf{M}^{-1}$ :

$$\left( -\omega_n^{(I)^2} \mathbf{M}^{-1} \mathbf{M} + \mathbf{M}^{-1} \mathbf{K} \right) \mathbf{U}^{(I)} = \mathbf{0}$$

$$\left( -\omega_n^{(I)^2} \mathbf{I} + \mathbf{M}^{-1} \mathbf{K} \right) \mathbf{U}^{(I)} = \mathbf{0}$$

What else can we do?



# Symmetric Eigenvalue Problem

Expected format:

$$(\lambda^{(I)} \mathbf{I} + \mathbf{A}) \mathbf{V}^{(I)} = \mathbf{0}, \quad \text{with } A_{ij} = A_{ji}$$

Our generalized eigenvalue problem:

$$\left( -\omega_n^{(I)^2} \mathbf{M} + \mathbf{K} \right) \mathbf{U}^{(I)} = \mathbf{0}$$

Represent  $\mathbf{M} = \mathbf{L}\mathbf{L}^*$  (**Cholesky decomposition**), which is  $\mathbf{M} = \mathbf{L}\mathbf{L}^T$  for symmetric matrices

$$\left( -\omega_n^{(I)^2} \mathbf{L}\mathbf{L}^T + \mathbf{K} \right) \mathbf{U}^{(I)} = \mathbf{0}$$

# Symmetric Eigenvalue Problem

$$\left(-\omega_n^{(I)^2} \mathbf{L}\mathbf{L}^T + \mathbf{K}\right) \mathbf{U}^{(I)} = \mathbf{0}$$

Assuming  $\mathbf{U} = \mathbf{L}^{-T}\mathbf{V}$  and left-multiplying by  $\mathbf{L}^{-1}$ :

$$\left(-\omega_n^{(I)^2} \mathbf{L}^{-1}\mathbf{L}\mathbf{L}^T\mathbf{L}^{-T} + \mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}\right) \mathbf{V}^{(I)} = \mathbf{0}$$

$$\left(-\omega_n^{(I)^2} \mathbf{I} + \tilde{\mathbf{K}}\right) \mathbf{V}^{(I)} = \mathbf{0}$$

with:

$$\tilde{\mathbf{K}} = \mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}$$

Mass-normalized stiffness matrix

# Exercise with Symmetric Eigenvalue Problem

Using script `exercise09_symmetric_eigenvalue_problem.py`

Let's solve:

$$\left(-\omega_n^{(I)^2} \mathbf{I} + \tilde{\mathbf{K}}\right) \mathbf{V}^{(I)} = \mathbf{0}$$

with:

$$\tilde{\mathbf{K}} = \mathbf{L}^{-1} \mathbf{K} \mathbf{L}^{-T}$$

And see the properties of  $\mathbf{V}^{(I)}$

NOTE:  $\mathbf{V}^{(I)}$  calculated with `np.linalg.eigh` are  
already normalized to  $|\mathbf{V}^{(I)}| = 1$

# Solving our free undamped vibration problem

From:

$$\mathbf{u}(x, t) = \sum_I^N \left( c_1^{(I)} \cos \omega_n^{(I)} t + c_2^{(I)} \sin \omega_n^{(I)} t \right) \mathbf{U}^{(I)}$$

Using  $\mathbf{U} = \mathbf{L}^{-T} \mathbf{V}$  :

$$\mathbf{u}(x, t) = \sum_I^N \left( c_1^{(I)} \cos \omega_n^{(I)} t + c_2^{(I)} \sin \omega_n^{(I)} t \right) \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

The problem now consists on finding constants  $c_1^{(I)}$  and  $c_2^{(I)}$  ...

# Solving our free undamped vibration problem

$$\mathbf{u}(x, t) = \sum_I^N \left( c_1^{(I)} \cos \omega_n^{(I)} t + c_2^{(I)} \sin \omega_n^{(I)} t \right) \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

$$\dot{\mathbf{u}}(x, t) = \sum_I^N \left( -\omega_n^{(I)} c_1^{(I)} \sin \omega_n^{(I)} t + \omega_n^{(I)} c_2^{(I)} \cos \omega_n^{(I)} t \right) \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

Initial conditions:  $\mathbf{u}(x, 0) = \mathbf{u}_0$ :

$$\mathbf{u}_0 = \sum_I c_1^{(I)} \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

Left-multiplying by  $\mathbf{L}^T$  and  $\mathbf{V}^{(J)}$

$$\mathbf{V}^{(J)} \mathbf{L}^T \mathbf{u}_0 = \sum_I c_1^{(I)} \mathbf{V}^{(J)} \mathbf{L}^T \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

$$\mathbf{V}^{(J)} \mathbf{L}^T \mathbf{u}_0 = \sum_I c_1^{(I)} \mathbf{V}^{(J)} \mathbf{I} \mathbf{V}^{(I)}$$

$$c_1^{(J)} = \mathbf{V}^{(J)} \mathbf{L}^T \mathbf{u}_0$$

Initial conditions:  $\dot{\mathbf{u}}(x, 0) = \mathbf{v}_0$ :

$$\mathbf{v}_0 = \sum_I c_2^{(I)} \omega_n^{(I)} \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

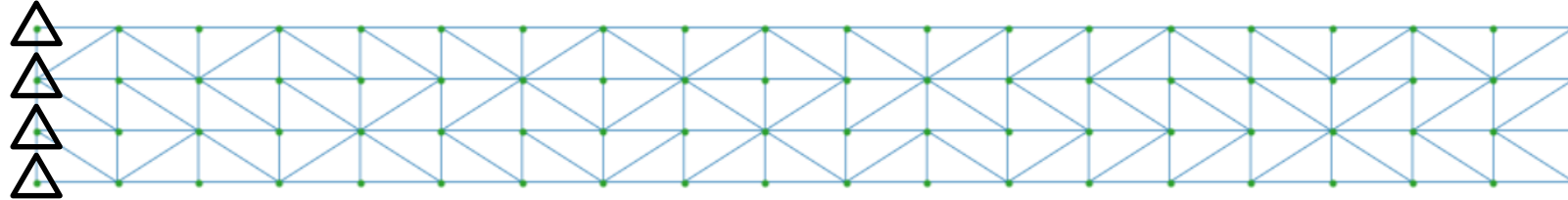
Left-multiplying by  $\mathbf{L}^T$  and  $\mathbf{V}^{(J)}$

$$\mathbf{V}^{(J)} \mathbf{L}^T \mathbf{v}_0 = \sum_I c_2^{(I)} \omega_n^{(I)} \mathbf{V}^{(J)} \mathbf{L}^T \mathbf{L}^{-T} \mathbf{V}^{(I)}$$

$$\mathbf{V}^{(J)} \mathbf{L}^T \mathbf{v}_0 = \sum_I c_2^{(I)} \omega_n^{(I)} \mathbf{V}^{(J)} \mathbf{I} \mathbf{V}^{(I)}$$

$$c_2^{(J)} = \frac{\mathbf{V}^{(J)} \mathbf{L}^T \mathbf{v}_0}{\omega_n^{(J)}}$$

# Example: Free vibration of a truss



## Using script

`exercise10_free_undamped_vibration_truss.py`

- Check the two initial conditions for  $u_0$
- Check how good is our approximation for different number of modes
- Does the number of modes for a good approximation changes with the initial condition?

# Transforming a MDOF system into uncoupled SDOF systems

Objective:

Use the relations for SDOF that we already know for forced vibration, damped/undamped

# MDOF system into uncoupled SDOF systems

Equation of motion for free undamped vibration:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}$$

From the previous definition:  $\mathbf{U}(x) = \mathbf{L}^{-T}\mathbf{V}(x)$ ; multiplying both sides by  $T(t)$  gives  $\mathbf{u}(x, t) = \mathbf{L}^{-T}\mathbf{v}(x, t)$ , thus:

$$\mathbf{I}\ddot{\mathbf{v}} + \tilde{\mathbf{K}}\mathbf{v} = \mathbf{0}$$

With  $\tilde{\mathbf{K}} = \mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}$ . This is a set of N coupled equations. Using our orthonormal eigenvectors of  $\tilde{\mathbf{K}}$ , previously named  $\mathbf{V}^{(I)}$ , let's define:

$$\mathbf{P}(x) = [\mathbf{V}^{(1)} \quad \mathbf{V}^{(2)} \quad \dots \quad \mathbf{V}^{(p)}]$$

Where  $p$  is the number of modes wanted in the approximation. Defining  $\mathbf{v}(x, t) = \mathbf{P}(x)\mathbf{r}(t)$  and left-multiplying by  $\mathbf{P}^T$  gives:

$$\mathbf{P}^T\mathbf{P}\ddot{\mathbf{r}} + \mathbf{P}^T\tilde{\mathbf{K}}\mathbf{P}\mathbf{r} = \mathbf{0}$$

Let's check the properties of this system

script `exercisel1_check_uncoupling.py`



# MDOF system into uncoupled SDOF systems

$$\mathbf{P}^T \mathbf{P} \ddot{\mathbf{r}} + \mathbf{P}^T \tilde{\mathbf{K}} \mathbf{P} \mathbf{r} = \mathbf{0}$$

Gives us an uncoupled system with  $p$  equations,  $p \leq N$ :

$$\mathbf{I} \ddot{\mathbf{r}}(t) + \mathbf{\Lambda} \mathbf{r}(t) = \mathbf{0}$$

$$\mathbf{\Lambda} = \mathbf{P}^T \tilde{\mathbf{K}} \mathbf{P}$$

Note that  $\mathbf{\Lambda}$  is simple (it does not have to be calculated):

$$\mathbf{\Lambda} = \begin{bmatrix} \left(\omega_n^{(1)}\right)^2 & & & 0 \\ & \left(\omega_n^{(2)}\right)^2 & & \\ & & \ddots & \\ 0 & & & \left(\omega_n^{(p)}\right)^2 \end{bmatrix}$$

# MDOF system into uncoupled SDOF systems

The uncoupled system has the form:

$$\ddot{r}_I + \left(\omega_n^{(I)}\right)^2 r_I = 0 \quad I = 1, 2, \dots, p$$

Noting that  $r_I = r_I(t)$ , the general solution is, for each DOF:

$$r_I(t) = C_1^{(I)} \cos \omega_n^{(I)} t + C_2^{(I)} \sin \omega_n^{(I)} t$$

Where  $r(0)$  and  $\dot{r}(0)$  are needed, which can be calculated from  $\mathbf{u}_0(x, t)$  and  $\mathbf{v}_0(x, t)$ . Using our definitions:

$$\mathbf{v} = \mathbf{L}^T \mathbf{u} \quad \text{and} \quad \mathbf{v} = \mathbf{P} \mathbf{r}$$

Left-multiplying the later by  $\mathbf{P}^T$  gives  $\mathbf{r} = \mathbf{P}^T \mathbf{v}$   
(remembering that  $\mathbf{P}^T \mathbf{P} = \mathbf{I}$ ). Thus:

$$\mathbf{r} = \mathbf{P}^T \mathbf{L}^T \mathbf{u}$$

Similarly:

$$\mathbf{u} = \mathbf{L}^{-T} \mathbf{P} \mathbf{r}$$

# Damped free vibration

The SDOF system has the form:

$$\ddot{r}_I + 2\zeta_I \omega_n^{(I)} \dot{r}_I + \left(\omega_n^{(I)}\right)^2 r_I = 0 \quad I = 1, 2, \dots, p$$

Note that the concept of damping ratio can be adopted for each mode (modal damping).

# Undamped forced vibration

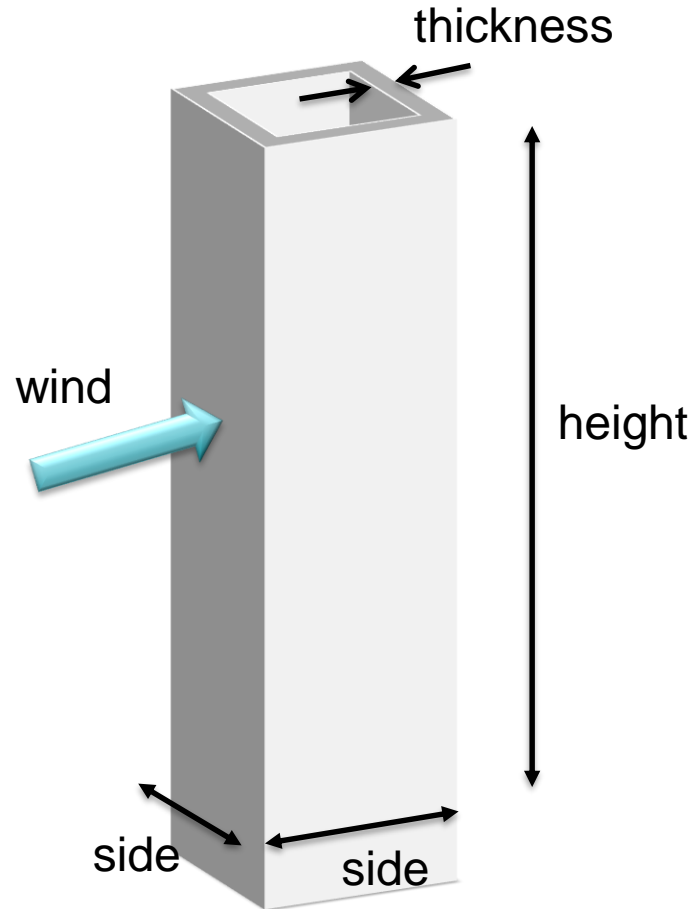
The SDOF system has the form:

$$\ddot{r}_I + \left(\omega_n^{(I)}\right)^2 r_I = f_I \quad I = 1, 2, \dots, p$$

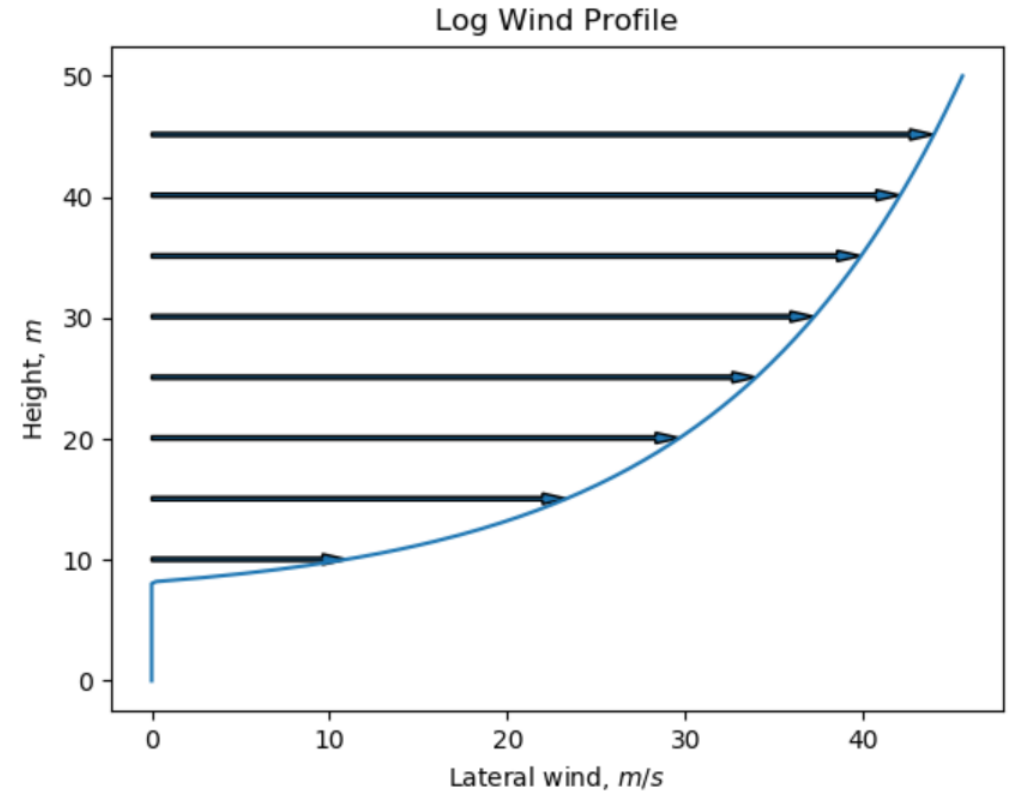
Here  $f_I$  is called modal force, which is the  $I^{th}$  row of  $f = P^T L^{-1} F$

$$\begin{aligned} M\ddot{u} + Ku &= F & \leftarrow u = L^{-T}v \\ ML^{-T}\ddot{v} + KL^{-T}v &= F \\ LL^T L^{-T}\ddot{v} + KL^{-T}v &= F \\ L^{-1}LL^T L^{-T}\ddot{v} + L^{-1}KL^{-T}v &= L^{-1}F \\ I\ddot{v} + \tilde{K}v &= L^{-1}F & \leftarrow v = Pr \\ P\ddot{r} + \tilde{K}Pr &= L^{-1}F \\ P^T P\ddot{r} + P^T \tilde{K}Pr &= P^T L^{-1}F \\ I\ddot{r} + \Lambda r &= f \end{aligned}$$

# Example: Undamped forced vibration



NOTE: properties can change along the height



$$wind(t) = wind + \frac{wind}{10} \sin(\omega_{wind} t)$$

$$\omega_{wind} = \frac{2\pi}{5} \text{ rad/s}$$

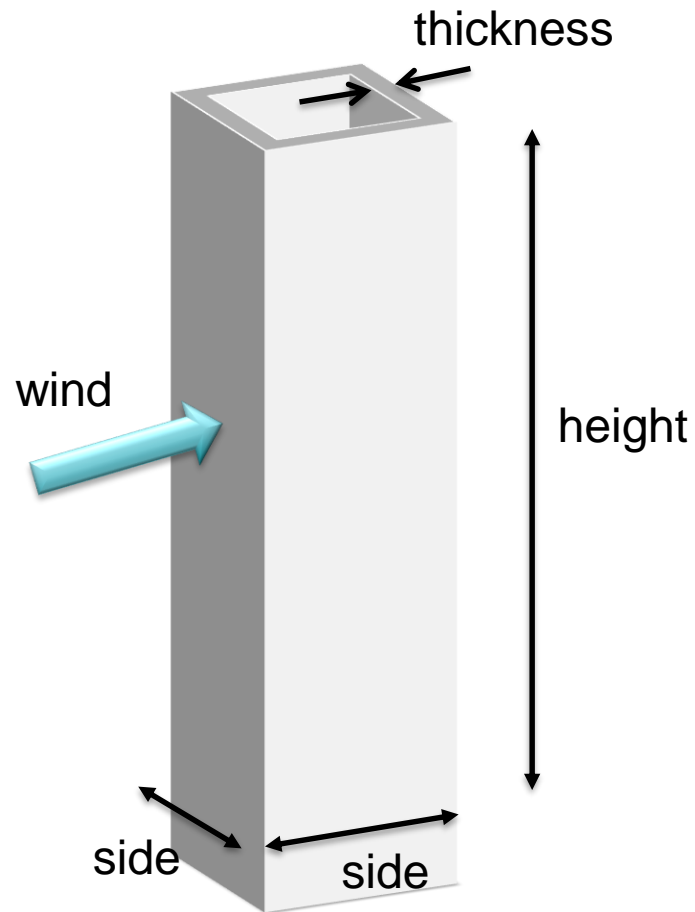
script `exercise12_building_unsteady_wind_undamped.py`

# Damped forced vibration

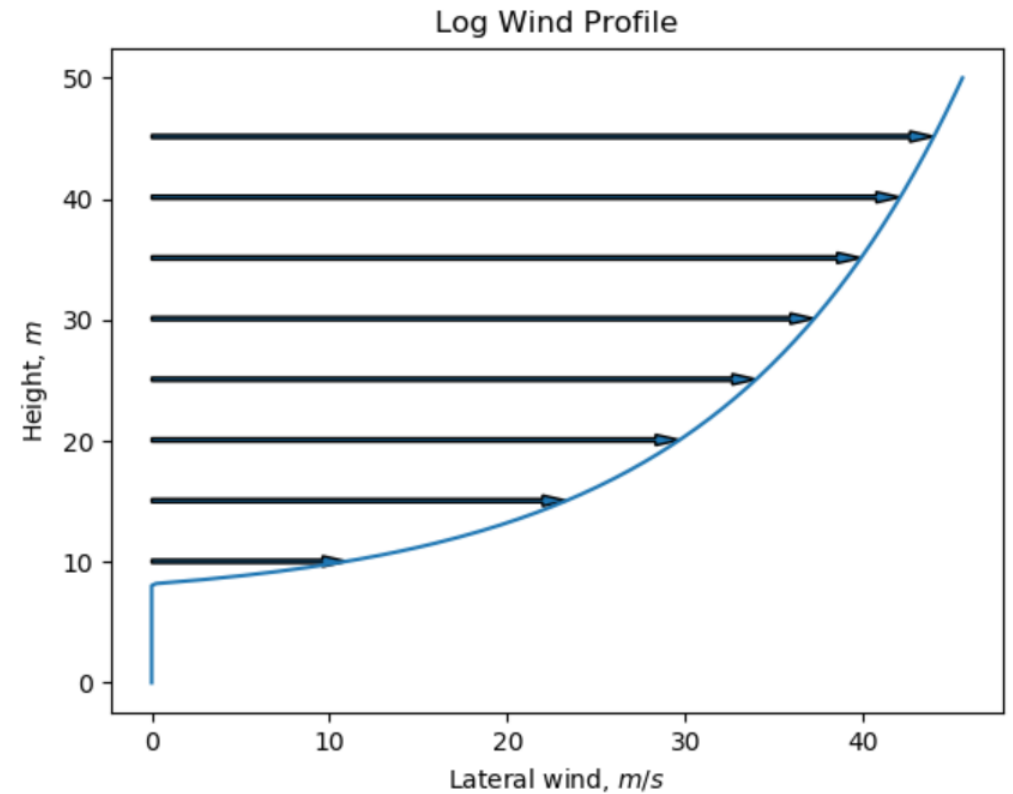
The SDOF system has the form:

$$\ddot{r}_I + 2\zeta_I \omega_n^{(I)} \dot{r}_I + \left(\omega_n^{(I)}\right)^2 r_I = f_I \quad I = 1, 2, \dots, p$$

# Example: Damped forced vibration



NOTE: properties can change along the height



$$wind(t) = wind + \frac{wind}{10} \sin(\omega_{wind} t)$$

$$\omega_{wind} = \frac{2\pi}{5} \text{ rad/s}$$

script `exercise13_building_unsteady_wind_damped.py`

# Example: Hard landing on rough terrain

This assignment is part of TU Delft's discipline Ae4ASM511 (Structural Analysis and Stability II)

The beam formulation herein applied is derived in details in:

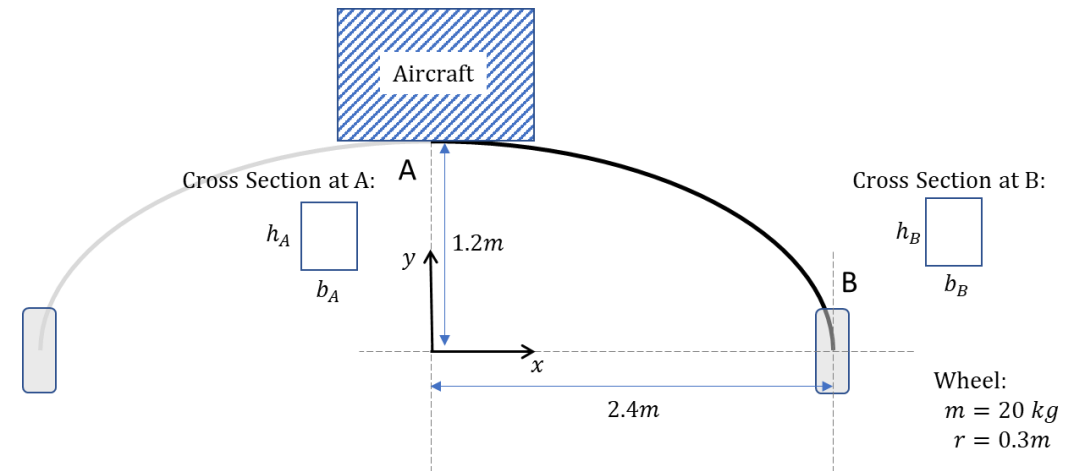
Saullo G P Castro. (2019, March). Stability and Analysis of Structures II (SAS II) 2019 Q3 Assignment 1 (Version 2019). Zenodo.

<http://doi.org/10.5281/zenodo.3238152>

This example is solving what is described in:

Saullo G P Castro. (2019, March). Stability and Analysis of Structures II (SAS II) 2019 Q3 Assignment 2 (Version 2019). Zenodo.

<http://doi.org/10.5281/zenodo.3238156>



Novelties of this model:

- Concentrated masses (aircraft and wheel)
- Concentrated mass moment of inertia (wheel)
- Displacement controlled (explained in assignments)
- Code vectorizing the evaluation for the dynamic response, doing the responses for all times at once
- Creating animation using Matplotlib

Scripts

`exercise14_landing_gear_undamped.py`

`exercise15_landing_gear_damped.py`



# Considerably more efficient implementations

Objective: Learn how to achieve much higher computational efficiency, important for large systems

# Bottlenecks of current approach

Cholesky decomposition  $M = LL^T$  and computation of  $L^{-1}$

We are using dense matrices to represent sparse systems

Algorithms `scipy.linalg.eigh` and `numpy.linalg.eigh` compute all  $N$  pairs of eigenvalues and eigenvectors, even if we only need  $p$  for the modal analysis

Pure Python

# Lumped Mass

Here  $\mathbf{M}$  is diagonal, such that  $\mathbf{L}$  will also be diagonal:

$$L_{ii} = \sqrt{M_{ii}}$$

Thus,  $\mathbf{L}^{-1}$  is also diagonal:

$$L_{ii}^{-1} = \frac{1}{L_{ii}}$$

# SciPy's Sparse Matrices

We can define all structural matrices  $M, K$  using  
`scipy.sparse.csr_matrix` (row-compact) or  
`scipy.sparse.csc_matrix` (column-compact)  
Both are easily created from `scipy.sparse.coo_matrix`

Build everything already using sparse matrices

# SciPy's Sparse Solvers

Eigenvalue solver `scipy.sparse.linalg.eigsh` works for both generalized and symmetric eigenvalue problems

Allows the computation of the first  $p$  eigenvalues that will be used in modal analysis, much more efficient than computing all  $N$ .

# Numba or Cython

These are options to compile the core part of the code.

Cython requires more experience, especially in C/C++

Numba offers JIT compilers, which really work in functions taking only integers, floats, NumPy arrays, and no classes, dictionaries and other Python objects

# Example for 2 million degrees-of-freedom

```
Script: exerciseXX_2million_dofs.py
```

```
Creating mesh
```

```
    Delaunay
```

```
    done (13.800249 s)
```

```
done (33.660385 s)
```

```
Number of degrees-of-freedom: 2000000
```

```
Computing K, M
```

```
done (17.646885 s)
```

```
Partitioning due to boundary conditions
```

```
done (1.233670 s)
```

```
Solving symmetric eigenvalue problem
```

```
[ 40.11186674 213.93027026 504.00858015 640.84402584]
```

```
done (196.419322 s)
```

Modal Analysis  
with



Saullo G. P. Castro

# Closing Remarks





Modal Analysis  
with



Saullo G. P. Castro

This course should be seen as an initial reference for you

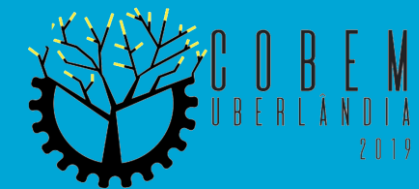


# Modal Analysis with



Saullo G. P. Castro

Implement other types of finite element



Modal Analysis  
with



Saullo G. P. Castro

Keep in touch in case you have any questions!

[S.G.P.Castro@tudelft.nl](mailto:S.G.P.Castro@tudelft.nl)

