Numerical methods and their application to periodic structures

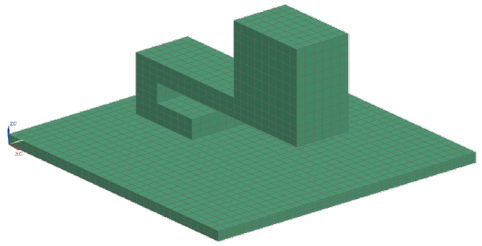
Authors: Elke Deckers, Vanessa Cool, Lucas Van Belle, Claus Claeys

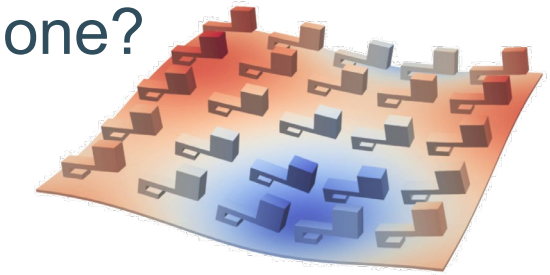
KU Leuven – Department of Mechanical Engineering
Flanders Make @ KU Leuven

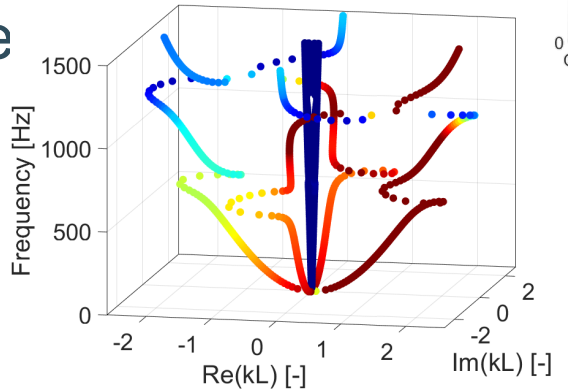
Training School on Acoustic & Elastic (Meta)-Materials, UPV, València, 13-17 November 2023

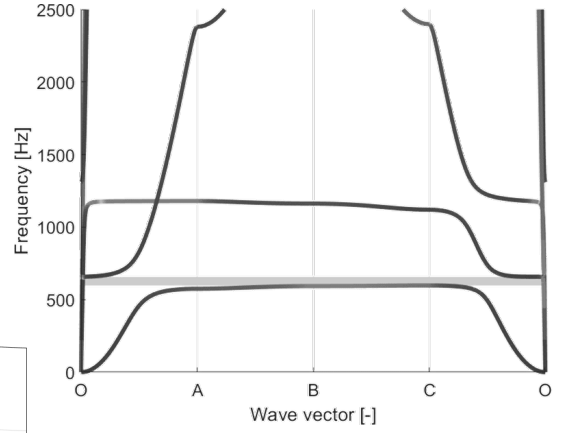
Outline

- Numerical methods in engineering and science
  - A focus to the Finite Element Method
- Infinite periodic structure modelling using FEM
  - With Matlab implementations
- What else can be done?



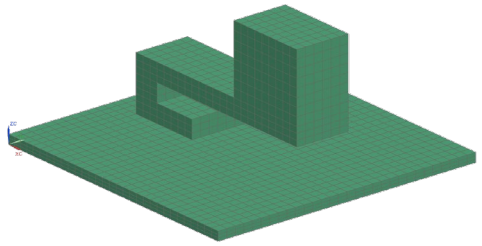


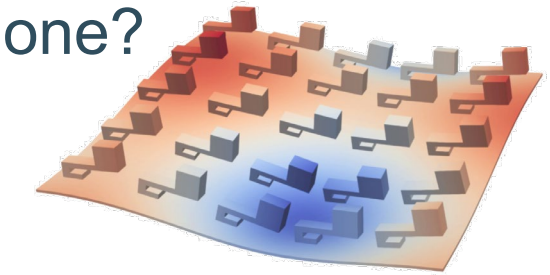


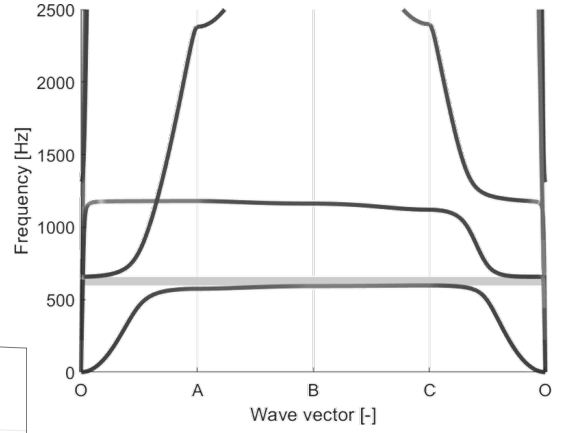


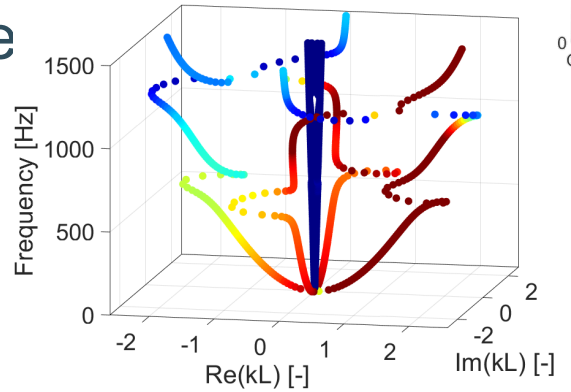
Outline

- Numerical methods in engineering and science
  - A focus to the Finite Element Method
- Infinite periodic structure modelling using FEM
  - With Matlab implementations
- What else can be done?





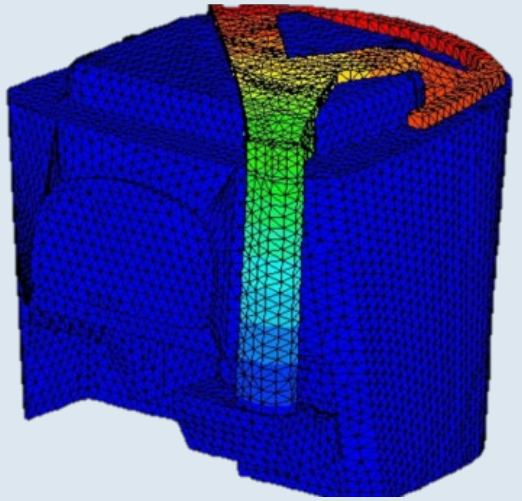




Additional material:

Educational paper:
http://arxiv.org/abs/2311.09843

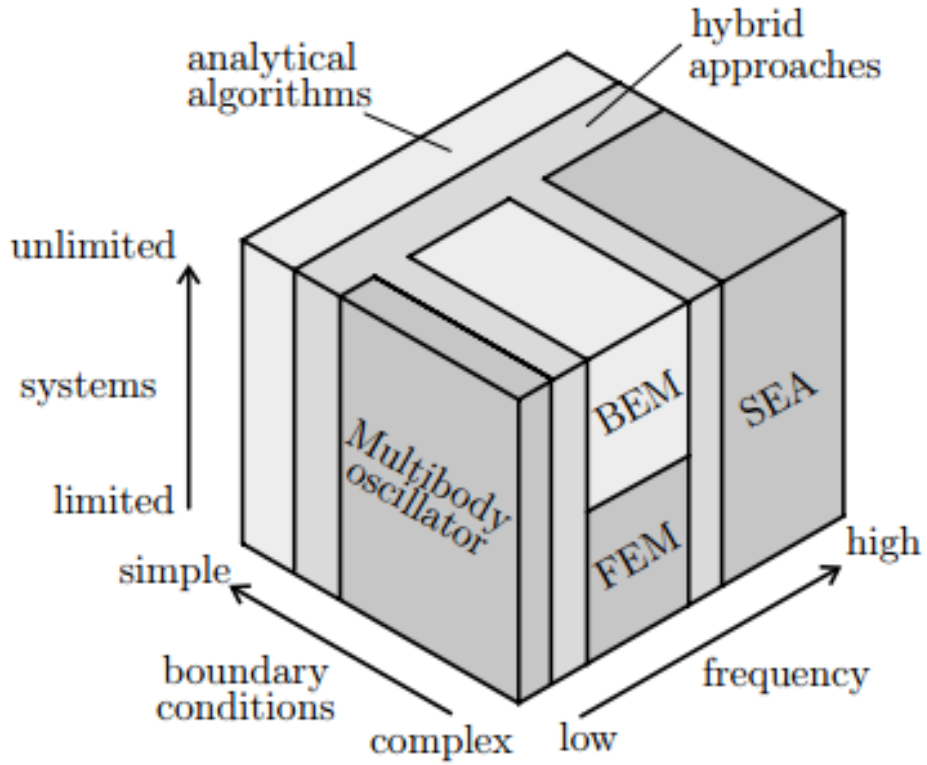
Matlab script: https://github.com/LMSD-KULeuven/2D_InverseUndamped_DispersionCurves .



Numerical methods in engineering and science

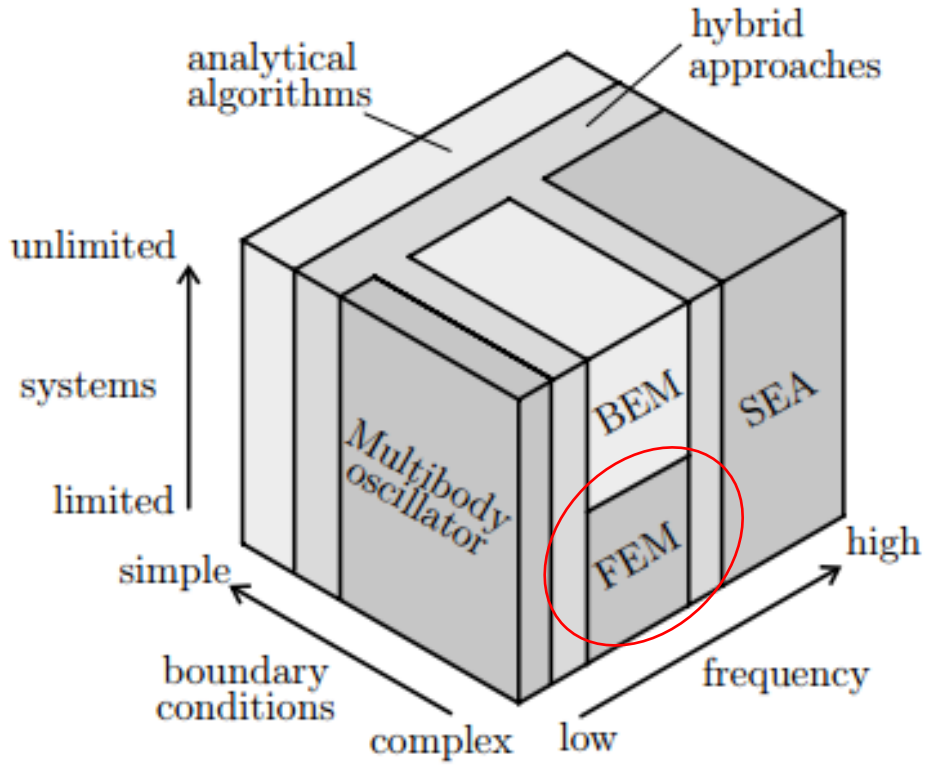
And focus on the Finite Element Method for acoustic problems

Numerical methods for frequency domain calculations



Frequency domain methods

Numerical methods for frequency domain calculations



Frequency domain methods

FEM

- Numerical discretization method
- Frequency & Time domain
- Bounded domains
- Low-Mid frequencies
- Complex Geometries

Main features

- Volume mesh

Numerical discretization methods

Partial differential equation (PDE)

$$\mathcal{L}q = F$$

Discretization of the computational domain =

- dividing medium into a number of small sub-regions: **ELEMENTS**
- defining a number of discrete points: **NODES** for which an approximate solution $\tilde{q}$ will be determined

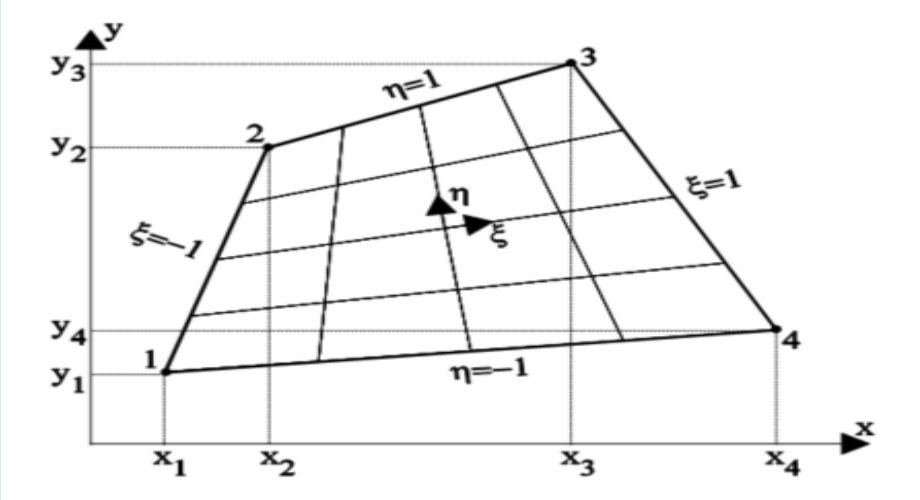
Replacing time/space derivatives using e.g.

- Difference equations (Finite Difference methods)
- Integral formulation (Finite Volume methods)
- Weighted residual formulation: integral formulation (Finite Element Methods)

ALGEBRAIC SYSTEM OF EQUATION (SOLVED USING MATRIX ALGEBRA) => solution is not exact anymore

$$[L]\hat{q} = \hat{F}$$

Finite Element Method



$$\int_{\Omega^e} \tilde{p} p d\Omega + j\rho_0\omega \int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\, d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega =$$

$$+ j\rho_0\omega \int_{\Omega^e} \tilde{p} q d\Omega - \int_{\Gamma_p^e} j\rho_0\omega \tilde{p} v_p \, d\Gamma$$

approximation

$$[N^e(x,y,z)]\{\hat{p}_e\} \qquad \vec{\nabla} p\,(x,y,z) = [B^e(x,y,$$

oproach

$$[N^e(x,y,z)]\{\tilde{p}_e\} \qquad \vec{\nabla}\tilde{p}(x,y,z) = [B^e(x,$$

linear acoustic problem

Problem setting

Assumptions
- small perturbations
- no viscosity
- Homogeneous (no flow, no temperature gradients)

Basic definitions
- speed of sound c
- fluid density $\boldsymbol{\rho_0}$
- source distribution q
- $k = \frac{\omega}{c} = \frac{2\pi}{\lambda}$, $c = \lambda . f$

Fourier transform

$$\boldsymbol{\nabla^2 p(r,t) - \frac{1}{c^2}\frac{\partial^2 p(r,t)}{\partial t^2} = -\rho_0 \frac{\partial q(r,t)}{\partial t}} \quad \Longleftrightarrow \quad \boldsymbol{\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)}$$

Time domain ***Frequency domain***

linear acoustic problem

Problem setting

Assumptions
- small perturbations
- no viscosity
- Homogeneous (no flow, no temperature gradients)

Basic definitions
- speed of sound c
- fluid density $\boldsymbol{\rho_0}$
- source distribution q
- $k = \frac{\omega}{c} = \frac{2\pi}{\lambda}$, $c = \lambda . f$

Fourier transform

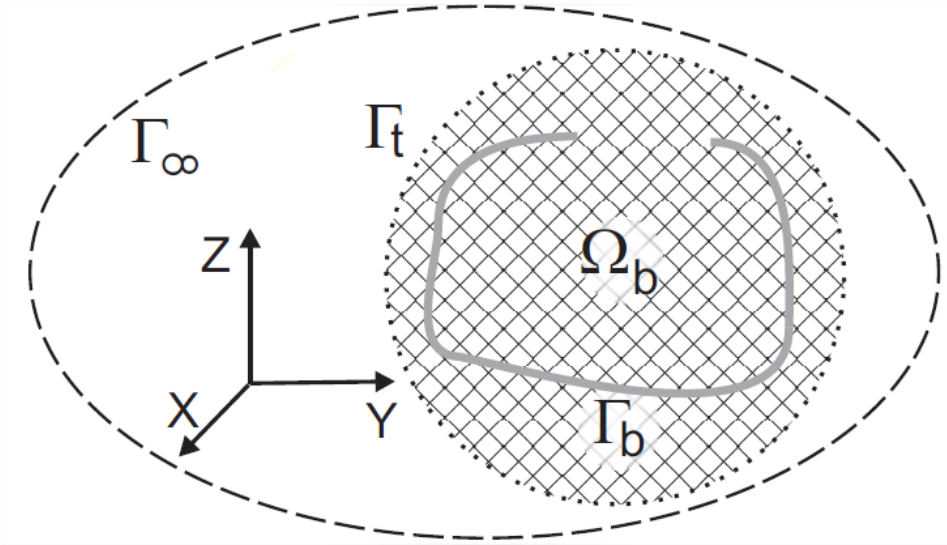
$$\boldsymbol{\nabla^2 p(r,t) - \frac{1}{c^2}\frac{\partial^2 p(r,t)}{\partial t^2} = -\rho_0 \frac{\partial q(r,t)}{\partial t}} \quad \longleftrightarrow \quad \boldsymbol{\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)}$$

Time domain ***Frequency domain***

$$k = \frac{\omega}{c} = \frac{2\pi}{\lambda} \qquad c = \lambda . f$$

linear acoustic problem

Problem setting



$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

Helmholtz equation

boundary conditions

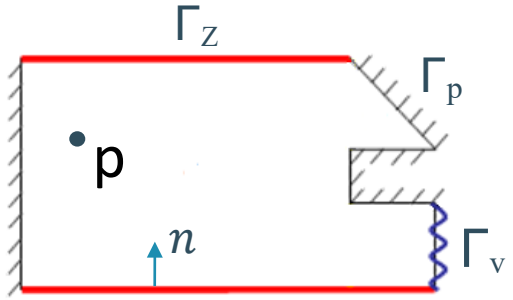
on problem surface Γ_b:
- prescribed pressure
- boundary (normal) velocity
- impedance

at Γ_∞:
- Sommerfeld

artificial truncation surface Γ_t:
- non-reflecting BC

linear acoustic problem

Problem setting – interior problem



$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

Helmholtz equation

boundary conditions

- Prescribed pressure:

$$p = \bar{p} \quad \text{on } \Gamma_p$$

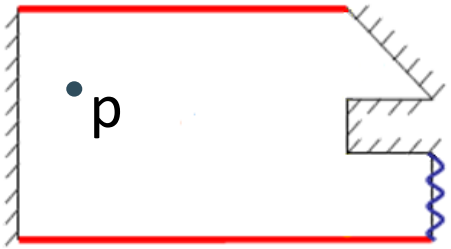
- Prescribed normal velocity:

$$\frac{j}{\rho_0 \omega} \frac{\partial p}{\partial n} = \bar{v} \text{ on } \Gamma_v$$

- Prescribed normal impedance:

$$\frac{j}{\rho_0 \omega} \frac{\partial p}{\partial n} = \frac{p}{Z_n} \text{ on } \Gamma_Z$$

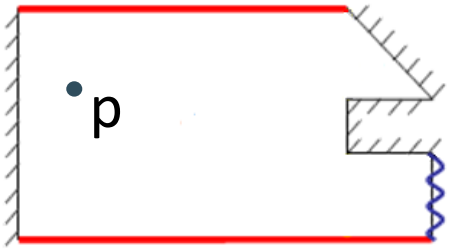
Finite element method



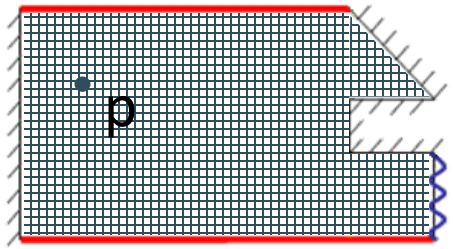
$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

+ boundary conditions

Finite element method



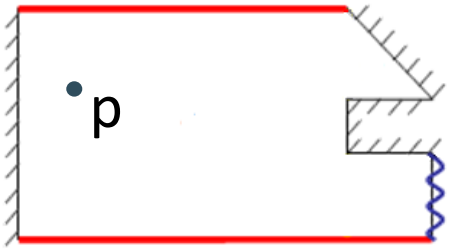
domain discretization



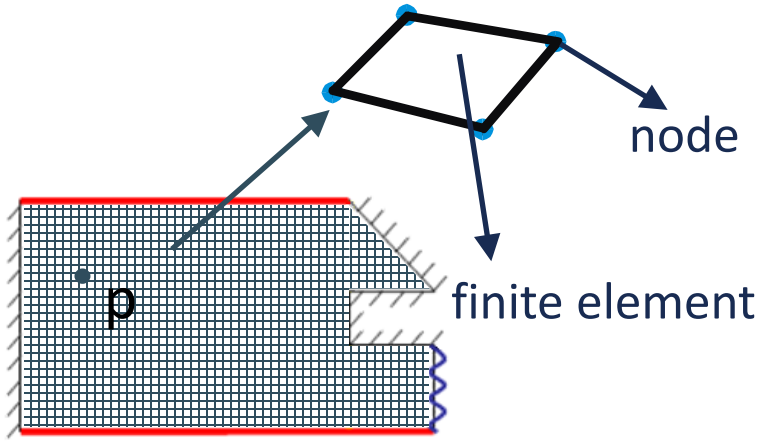
$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

+ boundary conditions

Finite element method



domain discretization



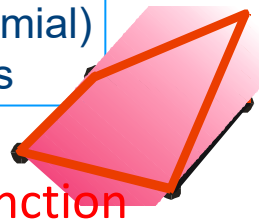
$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

+ boundary conditions

$$p(\boldsymbol{r}) \approx \sum_{i=1}^{n} N_i(\boldsymbol{r}).p_i$$

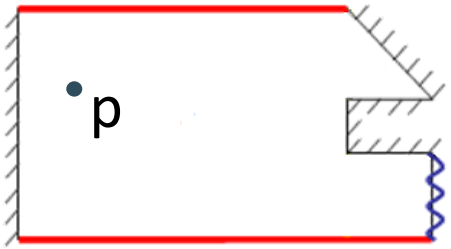
- local
- a priori defined
- simple (polynomial) approximations

N_i

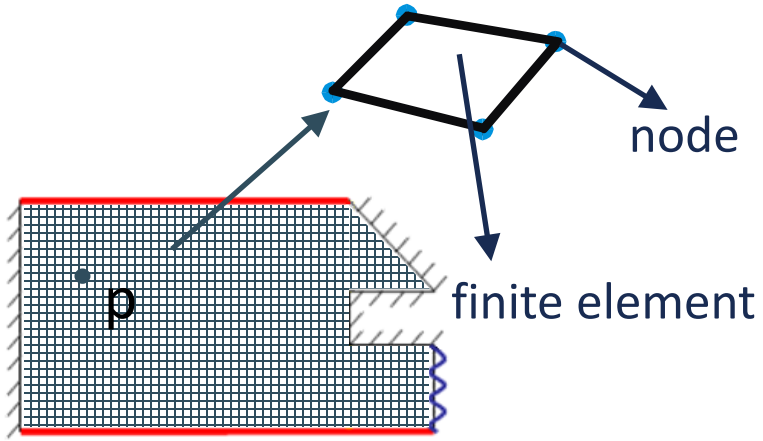


shape function

Finite element method



domain discretization



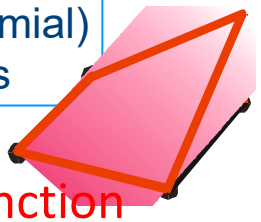
$$\nabla^2 p(r) + k^2 p(r) = -j\rho_0 \omega q(r)$$

+ boundary conditions

$$p(\boldsymbol{r}) \approx \sum_{i=1}^{n} N_i(\boldsymbol{r}).p_i$$

- local
- a priori defined
- simple (polynomial) approximations

N_i



shape function

$$\left([K_a] + j\omega[C_a] - \omega^2[M_a]\right).\{p_i\} = \{F_a\}$$

Finite Element Method

Derivation of weighted residual formulation

equivalent integral formulation

- *original acoustic problem = STRONG FORM*

pressure field in domain Ω

$$\nabla^2 p(r) + k^2\, p(r) = -\mathrm{j}\rho_0 \omega q(r)$$

+Boundary Conditions

Finite Element Method

Derivation of weighted residual formulation

equivalent integral formulation

- *original acoustic problem = STRONG FORM*

  pressure field in domain Ω

$$\nabla^2 p(r) + k^2 p(r) = -\mathrm{j}\rho_0 \omega q(r)$$

  +Boundary Conditions

- *weighted residual formulation*

$$\forall\, \tilde{p}\colon \int_\Omega \tilde{p}(\nabla^2 p + k^2 p + j\rho_0 \omega q \, d\Omega) = 0$$

Finite Element Method

Derivation of weighted residual formulation

equivalent integral formulation

- *original acoustic problem = STRONG FORM*

  pressure field in domain Ω

$$\nabla^2 p(r) + k^2 p(r) = -\mathrm{j}\rho_0 \omega q(r)$$

  +Boundary Conditions

- *weighted residual formulation*

$$\forall\, \tilde{p}\colon \int_\Omega \tilde{p}(\nabla^2 p + k^2 p + j\rho_0 \omega q\, d\Omega) = 0$$

- *Integration by parts* $\vec{\nabla} \cdot (\tilde{p}\vec{\nabla} p) = \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p + \tilde{p}(\nabla^2 p)$

Finite Element Method

Derivation of weighted residual formulation

equivalent integral formulation

- *original acoustic problem = STRONG FORM*

  pressure field in domain Ω

$$\nabla^2 p(r) + k^2 p(r) = -\mathrm{j}\rho_0 \omega q(r)$$

  +Boundary Conditions

- *weighted residual formulation*

$$\forall\, \tilde{p}\colon \int_\Omega \tilde{p}\left(\nabla^2 p + k^2 p + j\rho_0 \omega q \, d\Omega\right) = 0$$

- *Integration by parts* $\vec{\nabla} \cdot (\tilde{p}\vec{\nabla} p) = \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p + \tilde{p}(\nabla^2 p)$

Finite Element Method

Derivation of weighted residual formulation

equivalent integral formulation

- *original acoustic problem = STRONG FORM*

  pressure field in domain Ω

$$\nabla^2 p(r) + k^2 p(r) = -\mathrm{j}\rho_0 \omega q(r)$$

  +Boundary Conditions

- *weighted residual formulation*

$$\forall\, \tilde{p}\colon \int_\Omega \tilde{p}\left(\nabla^2 p + k^2 p + j\rho_0 \omega q\, d\Omega\right) = 0$$

- *Integration by parts* $\vec{\nabla} \cdot (\tilde{p}\vec{\nabla} p) = \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p + \tilde{p}(\nabla^2 p)$

$$\forall\, \tilde{p}\colon \int_\Omega \vec{\nabla} \cdot (\tilde{p}\vec{\nabla} p) d\Omega + \int_\Omega k^2 \tilde{p} p d\Omega + \int_\Omega \tilde{p} j\rho_0 \omega q\, d\Omega = + \int_\Omega \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega$$

Finite element method

Derivation of weighted residual formulation

$$\forall\, \tilde{p}\colon \int_{\Omega} \vec{\nabla} \cdot \left(\tilde{p}\vec{\nabla}p\right) d\Omega + \int_{\Omega} k^2 \tilde{p} p d\Omega + \int_{\Omega} \tilde{p} j \rho_0 \omega q \, d\Omega = + \int_{\Omega} \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega$$

Finite element method

Derivation of weighted residual formulation

$$\forall\, \tilde{p}\colon \int_{\Omega} \vec{\nabla} \cdot \left(\tilde{p}\vec{\nabla}p\right) d\Omega + \int_{\Omega} k^2 \tilde{p} p d\Omega + \int_{\Omega} \tilde{p} j \rho_0 \omega q \, d\Omega = + \int_{\Omega} \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega$$

- *Gauss divergence theorem + relation between pressure gradient and particle velocity*

Finite element method

Derivation of weighted residual formulation

$$\forall\, \tilde{p}: \int_\Omega \vec{\nabla} \cdot (\tilde{p}\vec{\nabla}p) d\Omega + \int_\Omega k^2 \tilde{p} p d\Omega + \int_\Omega \tilde{p} j \rho_0 \omega q \, d\Omega = + \int_\Omega \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega$$

- *Gauss divergence theorem + relation between pressure gradient and particle velocity*

$$\forall\, \tilde{p}: - \int_\Omega k^2 \tilde{p} p d\Omega + \int_\Omega \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega = + \int_\Omega \tilde{p} j \rho_0 \omega q \, d\Omega - \int_\Gamma j \rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma$$

Finite element method

Derivation of weighted residual formulation

$$\forall\, \tilde{p}: \int_\Omega \vec{\nabla} \cdot (\tilde{p}\vec{\nabla}p) d\Omega + \int_\Omega k^2 \tilde{p} p d\Omega + \int_\Omega \tilde{p} j\rho_0 \omega q \, d\Omega = + \int_\Omega \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega$$

- *Gauss divergence theorem + relation between pressure gradient and particle velocity*

$$\forall\, \tilde{p}: - \int_\Omega k^2 \tilde{p} p d\Omega + \int_\Omega \vec{\nabla}\tilde{p} \cdot \vec{\nabla} p d\Omega = + \int_\Omega \tilde{p} j\rho_0 \omega q \, d\Omega - \int_\Gamma j\rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma$$

Final integral $\int_\Gamma j\rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma$ *can be expressed as sum of:*

- $\int_{\Gamma_p} j\rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma = \int_{\Gamma_p} j\rho_0 \omega \tilde{p} v_p \, d\Gamma$
- $\int_{\Gamma_v} j\rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma = \int_{\Gamma_v} j\rho_0 \omega \tilde{p} \bar{v}_n \, d\Gamma$
- $\int_{\Gamma_z} j\rho_0 \omega \tilde{p} \vec{v}\vec{n} \, d\Gamma = \int_{\Gamma_z} \frac{j\rho_0 \omega \tilde{p} p}{\bar{z}} \, d\Gamma$

Finite element method

Derivation of weighted residual formulation

Final WEIGHTED RESIDUAL FORMULATION

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega} \tilde{p}p\,d\Omega + j\rho_0\omega\int_{\Gamma_z} \frac{\tilde{p}p}{\bar{Z}}\,d\Gamma + \int_{\Omega} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\,d\Omega =$$

$$+j\rho_0\omega\int_{\Omega} \tilde{p}q\,d\Omega - \int_{\Gamma_p} j\rho_0\omega\tilde{p}v_p\,d\Gamma + \int_{\Gamma_v} j\rho_0\omega\tilde{p}\bar{v}_n\,d\Gamma$$

Finite element method

Derivation of weighted residual formulation

Final WEIGHTED RESIDUAL FORMULATION

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega} \tilde{p}p\,d\Omega + j\rho_0\omega\int_{\Gamma_z} \frac{\tilde{p}p}{\bar{Z}}\,d\Gamma + \int_{\Omega} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\,d\Omega =$$

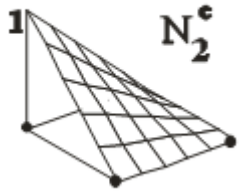
$$+j\rho_0\omega\int_{\Omega} \tilde{p}q\,d\Omega - \int_{\Gamma_p} j\rho_0\omega\tilde{p}v_p\,d\Gamma + \int_{\Gamma_v} j\rho_0\omega\tilde{p}\bar{v}_n\,d\Gamma$$

Boundary conditions naturally included!

Finite element method

Primary field variable approximation

$$p\,(x,y,z) = \sum_{i=1}^{n_e} N_i^e(x,y,z)\hat{p}_i$$



element shape functions N_i^e

- locally defined within an element
- n_e = # degrees of freedom = # nodes * # dofs/node
- dof evaluation at node locations: 1 shape function equals 1 / others equal 0
  $\rightarrow$ nodal dof ($\hat{p}_i$) = value of the degree of freedom at the node location
- predefined shapes between the nodes
  (usually polynomial based / no exact solutions of governing equations)

Finite element method

Field variable approximation

primary field variable approximation

$$p\,(x,y,z) \approx \sum_{i=1}^{n_e} N_i^e(x,y,z)\hat{p}_i = \underbrace{[N^e(x,y,z)]}_{(1 \times n_e)}\underbrace{\{\hat{p}_e\}}_{(n_e \times 1)}$$

secondary field variable approximation

$$\begin{Bmatrix} \frac{\partial p(x,y,z)}{\partial x} \\ \frac{\partial p(x,y,z)}{\partial y} \\ \frac{\partial p(x,y,z)}{\partial z} \end{Bmatrix} \approx \sum_{i=1}^{n_e} \begin{Bmatrix} \frac{\partial N_i^e(x,y,z)}{\partial x} \\ \frac{\partial N_i^e(x,y,z)}{\partial y} \\ \frac{\partial N_i^e(x,y,z)}{\partial z} \end{Bmatrix} \hat{p}_i = \underbrace{[B^e\,(x,y,z)]}_{(3 \times n_e)}\underbrace{\{\hat{p}_e\}}_{(n_e \times 1)}$$

Finite element method

Single element FE model

- ***construction of single element FE model***
  - weighted residual formulation (in elemental domain Ω^e)

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega^e} \tilde{p} p\, d\Omega + j\rho_0\omega \int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\, d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\, d\Omega =$$

$$+j\rho_0\omega\int_{\Omega^e}\tilde{p}q\, d\Omega - \int_{\Gamma_p^e} j\rho_0\omega\tilde{p}v_p\, d\Gamma + \int_{\Gamma_v^e} j\rho_0\omega\tilde{p}\bar{v}_n\, d\Gamma$$

Finite element method

Single element FE model

- ***construction of single element FE model***
  - weighted residual formulation (in elemental domain Ω^e)

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega^e} \tilde{p} p\, d\Omega + j\rho_0\omega \int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\, d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\, d\Omega =$$

$$+j\rho_0\omega\int_{\Omega^e}\tilde{p}q\, d\Omega - \int_{\Gamma_p^e} j\rho_0\omega\tilde{p}v_p\, d\Gamma + \int_{\Gamma_v^e} j\rho_0\omega\tilde{p}\bar{v}_n\, d\Gamma$$

  - shape function approximation

$$p\,(x,y,z) = [N^e(x,y,z)]\{\hat{p}_e\} \qquad \vec{\nabla}p\,(x,y,z) = [B^e(x,y,z)]\{\hat{p}_e\}$$

Finite element method

Single element FE model

- ***construction of single element FE model***
  - weighted residual formulation (in elemental domain Ω^e)

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega^e} \tilde{p} p\, d\Omega + j\rho_0\omega \int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\, d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\, d\Omega =$$

$$+j\rho_0\omega\int_{\Omega^e}\tilde{p}q\, d\Omega - \int_{\Gamma_p^e} j\rho_0\omega\tilde{p}v_p\, d\Gamma + \int_{\Gamma_v^e} j\rho_0\omega\tilde{p}\bar{v}_n\, d\Gamma$$

  - shape function approximation

$$p\,(x,y,z) = [N^e(x,y,z)]\{\hat{p}_e\} \qquad \vec{\nabla}p\,(x,y,z) = [B^e(x,y,z)]\{\hat{p}_e\}$$

  - GALERKIN approach

$$\tilde{p}\,(x,y,z) = [N^e(x,y,z)]\{\tilde{p}_e\} \qquad \vec{\nabla}\tilde{p}(x,y,z) = [B^e(x,y,z)]\{\tilde{p}_e\}$$

Finite element method

Single element FE model

- ***construction of single element FE model***

- weighted residual formulation (in elemental domain Ω^e)

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega^e} \tilde{p}p\,d\Omega + j\rho_0\omega\int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\,d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\,d\Omega =$$

$$+j\rho_0\omega\int_{\Omega^e}\tilde{p}q\,d\Omega - \int_{\Gamma_p^e} j\rho_0\omega\tilde{p}v_p\,d\Gamma + \int_{\Gamma_v^e} j\rho_0\omega\tilde{p}\bar{v}_n\,d\Gamma$$

- shape function approximation

$$p\,(x,y,z) = [N^e(x,y,z)]\{\hat{p}_e\} \qquad \vec{\nabla}p\,(x,y,z) = [B^e(x,y,z)]\{\hat{p}_e\}$$

- GALERKIN approach

$$\tilde{p}\,(x,y,z) = [N^e(x,y,z)]\{\tilde{p}_e\} \qquad \vec{\nabla}\tilde{p}(x,y,z) = [B^e(x,y,z)]\{\tilde{p}_e\}$$

Finite element method

Single element FE model

- ***construction of single element FE model***

- weighted residual formulation (in elemental domain Ω^e)

$$\forall\, \tilde{p}: -\frac{\omega}{c}\int_{\Omega^e} \tilde{p}p\,d\Omega + j\rho_0\omega\int_{\Gamma_p^e} \frac{\tilde{p}p}{\bar{Z}}\,d\Gamma + \int_{\Omega^e} \vec{\nabla}\tilde{p}\cdot\vec{\nabla}p\,d\Omega =$$

$$+j\rho_0\omega\int_{\Omega^e}\tilde{p}q\,d\Omega - \int_{\Gamma_p^e} j\rho_0\omega\tilde{p}v_p\,d\Gamma + \int_{\Gamma_v^e} j\rho_0\omega\tilde{p}\bar{v}_n\,d\Gamma$$

- shape function approximation

$$p\,(x,y,z) = [N^e(x,y,z)]\{\hat{p}_e\} \qquad \vec{\nabla}p\,(x,y,z) = [B^e(x,y,z)]\{\hat{p}_e\}$$

- GALERKIN approach

$$\tilde{p}\,(x,y,z) = [N^e(x,y,z)]\{\tilde{p}_e\} \qquad \vec{\nabla}\tilde{p}(x,y,z) = [B^e(x,y,z)]\{\tilde{p}_e\}$$

$$\{\tilde{\boldsymbol{p}}^e\}^T.\left([\boldsymbol{K}^e]+j\boldsymbol{\omega}[\boldsymbol{C}^e]-\boldsymbol{\omega}^2[\boldsymbol{M}^e]\right).\{\boldsymbol{p}^e\} = \{\tilde{\boldsymbol{p}}^e\}^T.\left(\{\boldsymbol{Q}^e\}-\{\boldsymbol{V}_i^e\}-\{\boldsymbol{P}^e\}\right)$$

Finite element method

Single element FE model

$$\{\tilde{p}^e\}^T.\left([K^e]+j\omega[C^e]-\omega^2[M^e]\right).\{p^e\}=\{\tilde{p}^e\}^T.\left(\{Q^e\}-\{V_i^e\}-\{P^e\}\right)$$

$$[K^e]=\int_{\Omega^e}[B^e]^T[B^e]\mathrm{d}\Omega$$

element stiffness matrix
(geometry, material properties)

$$[M^e]=\int_{\Omega^e}\frac{1}{c^2}[N^e]^T[N^e]\mathrm{d}\Omega$$

element mass matrix
(geometry, material properties)

$$[C^e]=\int_{\Gamma_Z^e}\frac{\rho_0}{\bar{Z}_n}[N^e]^T[N^e]\mathrm{d}\Omega$$

element damping matrix
(geometry, impedance BCs)

$$\{Q^e\}=\int_{\Omega^e}j\rho_0\omega[N^e]^Tq\mathrm{d}\Omega-\int_{\Gamma_v^e}j\rho_0\omega[N^e]^T\bar{v}_n\mathrm{d}\Omega$$

element external excitation vector
(geometry, external load, velocity BC)

$$\{V_i^e\}=\int_{\Gamma_i^e}j\rho_0\omega[N^e]^Tv_i\mathrm{d}\Omega$$

element internal excitation vector
v_i *: velocity input from adjacent elements*

$$\{P^e\}=\int_{\Gamma_p^e}j\rho_0\omega[N^e]^T\bar{v}_p\mathrm{d}\Omega$$

element external excitation vector
(geometry, pressure BCs)

$\bar{v}_p$? *: (unknown) velocity input on problem boundary with pressure BC*

Finite element method

Single element FE model

$$\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{[K^e]+j\omega[C^e]-\omega^2[M^e]}_{(n_e\,x\,n_e)}\Big).\underbrace{\{p^e\}}_{(n_e\,x\,1)}=\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{\{Q^e\}-\{V_i^e\}-\{P^e\}}_{(n_e\,x\,1)}\Big)$$

Finite element method

Single element FE model

$$\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{[K^e]+j\omega[C^e]-\omega^2[M^e]}_{(n_e\,x\,n_e)}\Big).\underbrace{\{p^e\}}_{(n_e\,x\,1)}=\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{\{Q^e\}-\{V_i^e\}-\{P^e\}}_{(n_e\,x\,1)}\Big)$$

should hold for any weighting function $\Rightarrow$

Finite element method

Single element FE model

$$\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{[K^e]+j\omega[C^e]-\omega^2[M^e]}_{(n_e\,x\,n_e)}\Big).\underbrace{\{p^e\}}_{(n_e\,x\,1)}=\underbrace{\{\tilde{p}^e\}^T}_{(1\,x\,n_e)}.\Big(\underbrace{\{Q^e\}-\{V_i^e\}-\{P^e\}}_{(n_e\,x\,1)}\Big)$$

should hold for any weighting function $\Rightarrow$

element FE model

$$\underbrace{\Big([K^e]+j\omega[C^e]-\omega^2[M^e]\Big)}_{(n_e\,x\,n_e)}.\underbrace{\{p^e\}}_{(n_e\,x\,1)}=\underbrace{\{Q^e\}-\{V_i^e\}-\{P^e\}}_{(n_e\,x\,1)}$$

n_e equations in the n_e unknown element degrees of freedom (dofs)

Finite element method

Assembly of global FE model

assembly of element FE models into a global FE model

accounting for:

- *'compatibility' conditions*
  … coinciding nodes have shared dofs
- *'equilibrium' conditions*
  … internal excitation vectors compensate each other (v_i: *equal but opposite in sign*)

Finite element method

Assembly of global FE model

assembly of element FE models into a global FE model

accounting for:

- *'compatibility' conditions*
  … coinciding nodes have shared dofs
- *'equilibrium' conditions*
  … internal excitation vectors compensate each other (v_i: *equal but opposite in sign*)

---

global FE model

$$\underbrace{\left([K]+j\omega[C]-\omega^2[M]\right)}_{(n\,x\,n)}\underbrace{\{p\}}_{(n\,x\,1)}=\underbrace{\{F\}}_{(n\,x\,1)}$$

n equations in the *n* degrees of freedom

Finite element method

Assembly of global FE model

global excitation vector *{F}*

$$\{\boldsymbol{F}\} = \{\boldsymbol{Q}\} - \{\boldsymbol{P}\}$$

'summation' of element vectors

$$\{\boldsymbol{Q}\} \longleftarrow \{Q^e\} = \int_{\Omega^e} j\rho_0\omega[N^e]^T q \mathrm{d}\Omega - \int_{\Gamma_v^e} j\rho_0\omega[N^e]^T \bar{v}_n \mathrm{d}\Omega$$

$$\{0\} \longleftarrow \{V_i^e\} = \int_{\Gamma_i^e} j\rho_0\omega[N^e]^T v_i \mathrm{d}\Omega$$

… internal excitation vectors compensate each other
(v_i: *equal but opposite in sign*)

$$\{\boldsymbol{P}\} \longleftarrow \{P^e\} = \int_{\Gamma_p^e} j\rho_0\omega[N^e]^T \bar{v}_p \mathrm{d}\Omega$$

unknown !

Finite element method

Partitioning of the FE system of equations

$$\begin{Bmatrix} \boldsymbol{F}_f \\ \boldsymbol{F}_c \end{Bmatrix} = \begin{bmatrix} \boldsymbol{Z}_{ff} & \boldsymbol{Z}_{fc} \\ \boldsymbol{Z}_{cf} & \boldsymbol{Z}_{cc} \end{bmatrix} \cdot \begin{Bmatrix} \boldsymbol{p}_f \\ \boldsymbol{p}_c \end{Bmatrix}$$

known / *unknown* (left)

unknown / *known* (pressure BC on Γ_p) (right)

---

contribute to F_f ⟵

$$\{Q^e\} = \int_{\Omega^e} j\rho_0\omega[N^e]^T q \mathrm{d}\Omega - \int_{\Gamma_v^e} j\rho_0\omega[N^e]^T \bar{v}_n \mathrm{d}\Omega$$

contribute only to F_c not to F_f ! ⟵

$$\{P_i^e\} = \int_{\Gamma_i^e} j\rho_0\omega[N^e]^T \overline{v}_p \mathrm{d}\Omega$$

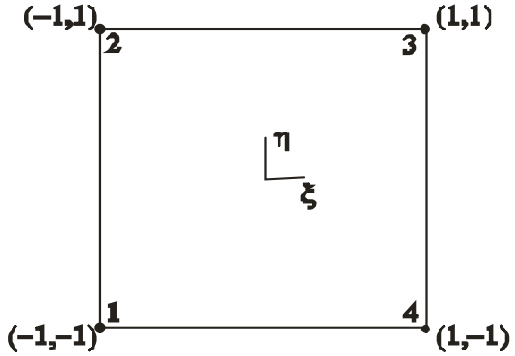
$$\mathrm{p_f} = Z_{ff}^{-1}(F_f - Z_{fc}p_c)$$

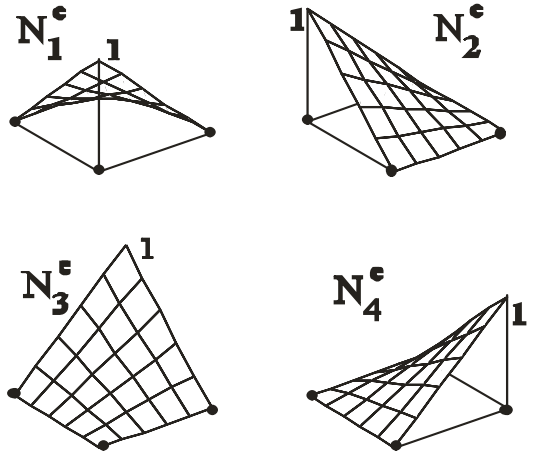
Finite element method

QUAD element

4-noded square element (order p=1)

local element coordinate system (ξ,η)





4 nodes
1 dof per node

$$\phi \approx N_1^e(\xi,\eta).\phi_1 + N_2^e(\xi,\eta).\phi_2 + N_3^e(\xi,\eta).\phi_3 + N_4^e(\xi,\eta).\phi_4$$

$$N_1^e(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta) \quad N_2^e(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta) \quad N_3^e(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta) \quad N_4^e(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_1^e(+1,+1) = 0$$
$$N_1^e(-1,+1) = 0$$
$$N_1^e(-1,-1) = 1$$
$$N_1^e(+1,-1) = 0$$

$$N_2^e(-1,-1) = 0$$
$$N_2^e(-1,+1) = 1$$
$$N_2^e(+1,+1) = 0$$
$$N_2^e(+1,-1) = 0$$

$$N_3^e(-1,-1) = 0$$
$$N_3^e(-1,+1) = 0$$
$$N_3^e(+1,+1) = 1$$
$$N_3^e(+1,-1) = 0$$

$$N_4^e(-1,-1) = 0$$
$$N_4^e(-1,+1) = 0$$
$$N_4^e(+1,+1) = 0$$
$$N_4^e(+1,-1) = 1$$

Finite element method

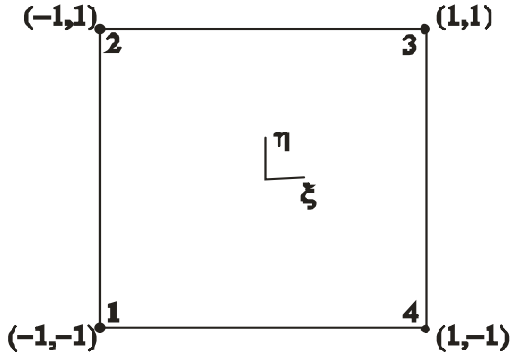
QUAD element

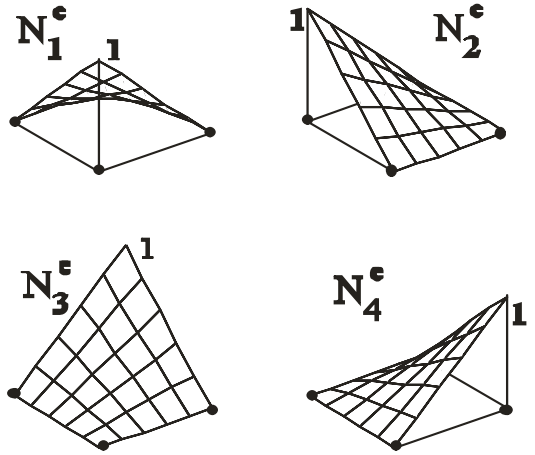
- *consistency : partition-of-unity*

$$\sum N_i^e = 1$$

4-noded square element (order p=1)

local element coordinate system (ξ,η)





4 nodes
1 dof per node

$$\phi \approx N_1^e(\xi,\eta).\phi_1 + N_2^e(\xi,\eta).\phi_2 + N_3^e(\xi,\eta).\phi_3 + N_4^e(\xi,\eta).\phi_4$$

$$N_1^e(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta) \qquad N_2^e(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta) \qquad N_3^e(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta) \qquad N_4^e(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_1^e(+1,+1) = 0$$
$$N_1^e(-1,+1) = 0$$
$$N_1^e(-1,-1) = 1$$
$$N_1^e(+1,-1) = 0$$

$$N_2^e(-1,-1) = 0$$
$$N_2^e(-1,+1) = 1$$
$$N_2^e(+1,+1) = 0$$
$$N_2^e(+1,-1) = 0$$

$$N_3^e(-1,-1) = 0$$
$$N_3^e(-1,+1) = 0$$
$$N_3^e(+1,+1) = 1$$
$$N_3^e(+1,-1) = 0$$

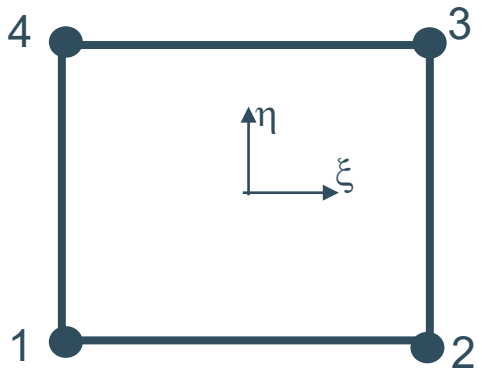
$$N_4^e(-1,-1) = 0$$
$$N_4^e(-1,+1) = 0$$
$$N_4^e(+1,+1) = 0$$
$$N_4^e(+1,-1) = 1$$

Finite element method

QUAD element – linear and quadratic

element definition

linear (p=1)



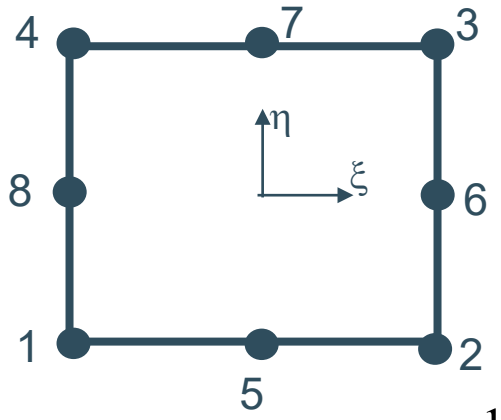
$$N_1 = \frac{1}{4}(1-\xi)(1-\eta)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta)$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

quadratic (p=2)



$$N_1 = \frac{1}{4}(1-\xi)(1-\eta)(-\xi-\eta-1)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta)(\xi-\eta-1)$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta)(\xi+\eta-1)$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta)(-\xi+\eta-1)$$

$$N_5 = \frac{1}{2}(1-\xi^2)(1-\eta)$$

$$N_6 = \frac{1}{2}(1+\xi)(1-\eta^2)$$

$$N_7 = \frac{1}{2}(1-\xi^2)(1+\eta)$$

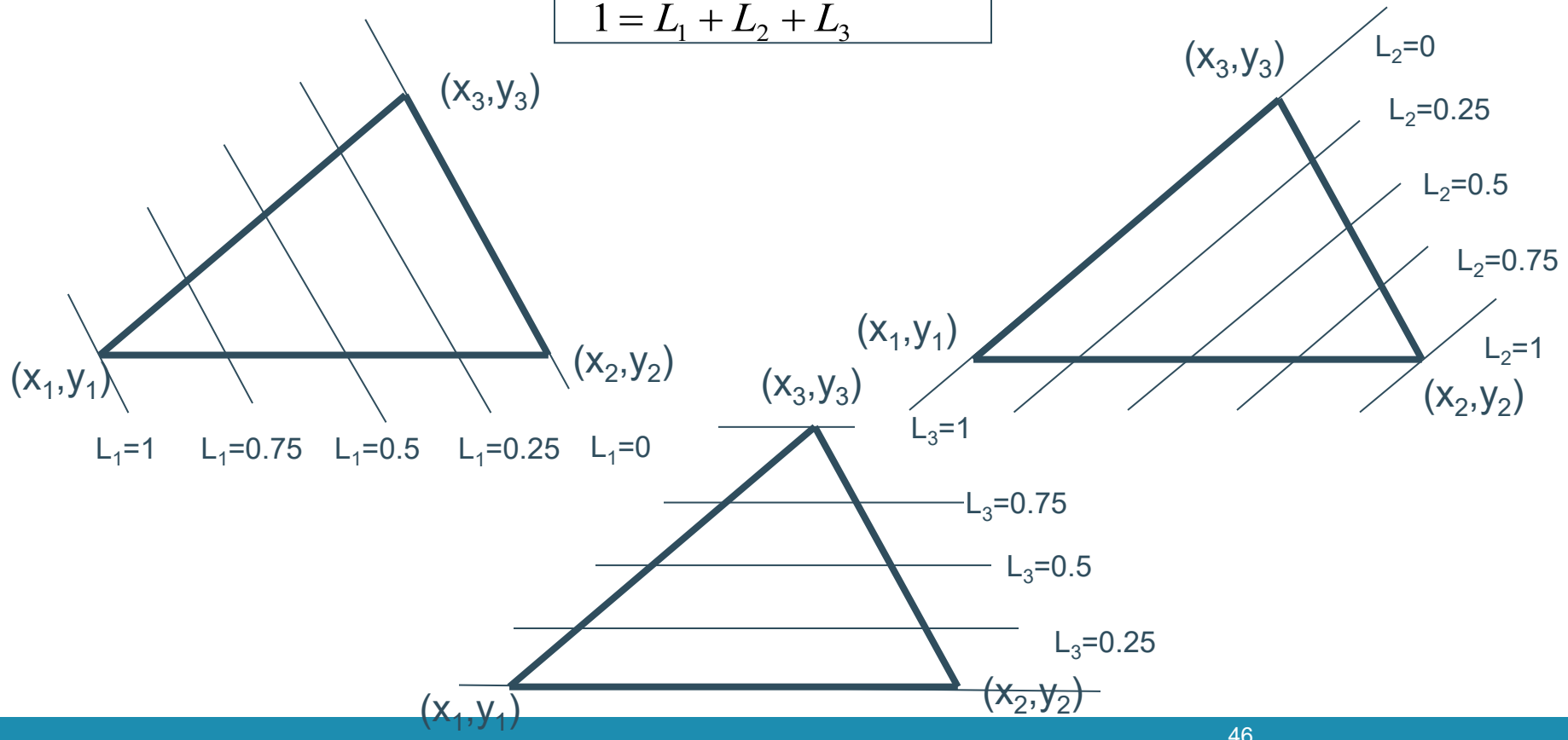
$$N_8 = \frac{1}{2}(1-\xi)(1-\eta^2)$$

Finite element method

TRIA element

surface co-ordinates

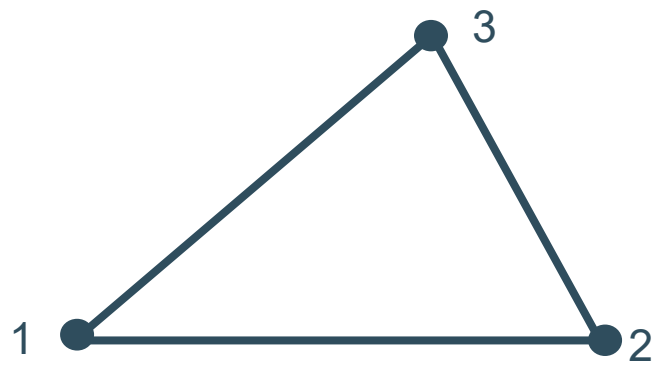
$$x = L_1.x_1 + L_2.x_2 + L_3.x_3$$
$$y = L_1.y_1 + L_2.y_2 + L_3.y_3$$
$$1 = L_1 + L_2 + L_3$$



Finite element method

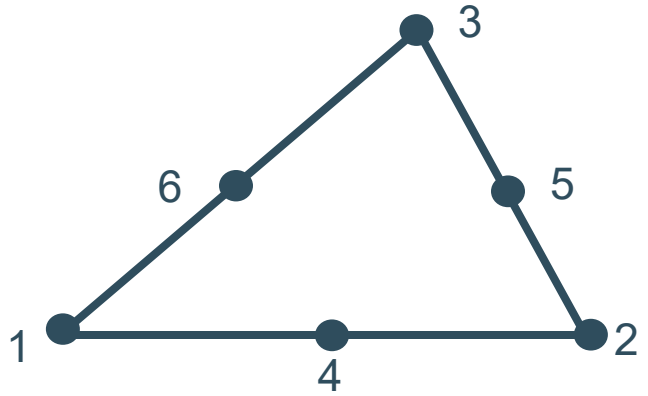
TRIA element – linear and quadratic

linear (p=1)



$$N_1 = L_1$$
$$N_2 = L_2$$
$$N_3 = L_3$$

quadratic (p=2)

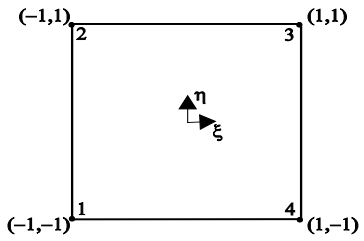


$$N_1 = L_1.(2L_1 - 1) \qquad N_4 = 4L_1L_2$$
$$N_2 = L_2.(2L_2 - 1) \qquad N_5 = 4L_2L_3$$
$$N_3 = L_3.(2L_3 - 1) \qquad N_6 = 4L_1L_3$$

Finite element method

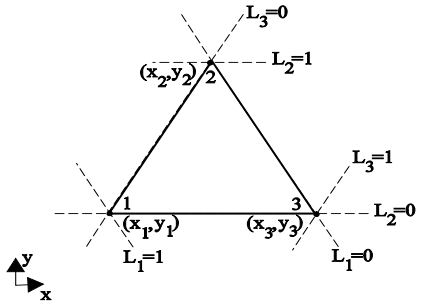
TRIA element – linear and quadratic

2D



(a)

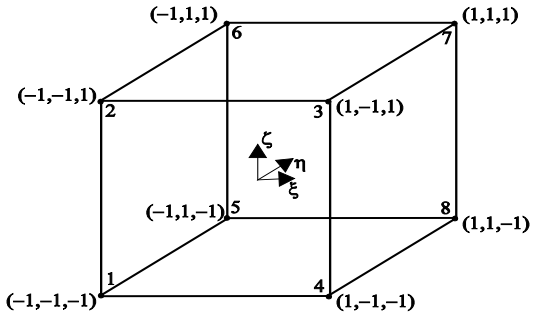
4-noded rectangular element



(b)

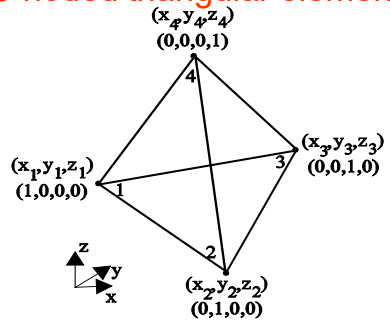
3-noded triangular element

3D



(c)

8-noded brick element

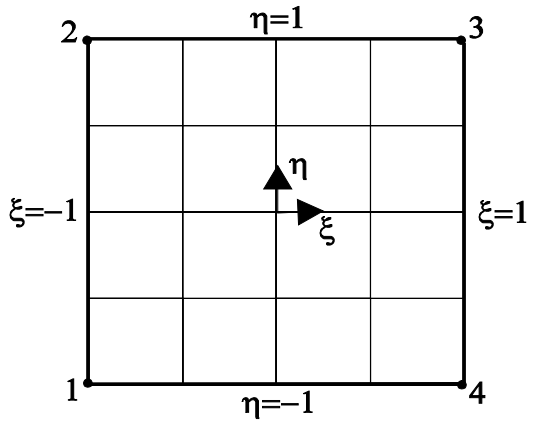


(d)

4-noded tetrahedral element

Finite element method

Isoparametric element



parent element

$$x=\sum_{i=1}^{4} N_i^e(\xi,\eta).x_i$$

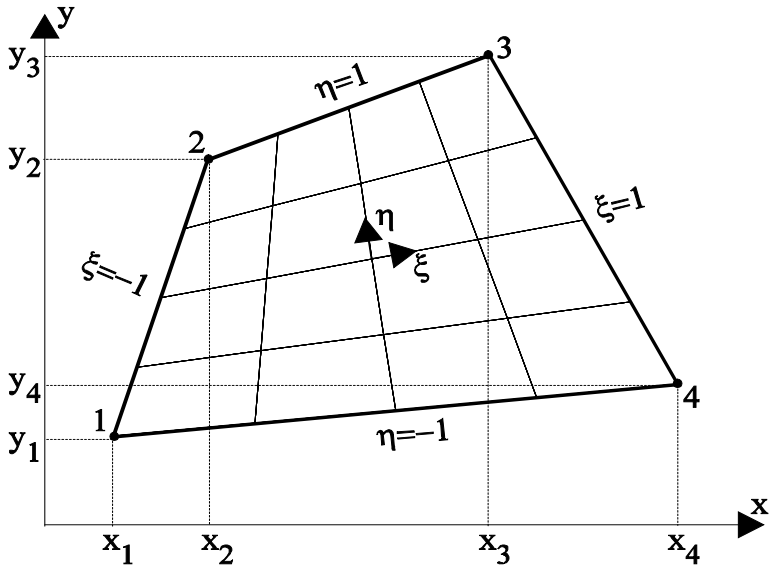
$$y=\sum_{i=1}^{4} N_i^e(\xi,\eta).y_i$$

co-ordinate transformation

$x=f_x(\xi,\eta,\zeta)$

$y=f_y(\xi,\eta,\zeta)$

$z=f_z(\xi,\eta,\zeta)$



mapped element

isoparametric elements: same shape functions for geometry and field variables

Finite element method

Isoparametric mapping

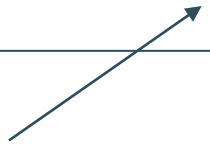
numerical integration

$$\boldsymbol{K}^{(e)} = \int_{\Omega_e} \boldsymbol{B}^{(e)T} \boldsymbol{D} \boldsymbol{B}^{(e)} .d\Omega$$

---

numerical integration of mapped elements

$$\iint_{\Omega^{(e)}} \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}).d\boldsymbol{x}.d\boldsymbol{y} = \int_{-1}^{+1}\int_{-1}^{+1} \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{\xi},\boldsymbol{\eta}),\boldsymbol{y}(\boldsymbol{\xi},\boldsymbol{\eta})).\det(\boldsymbol{J}).d\boldsymbol{\xi}.d\boldsymbol{\eta}$$



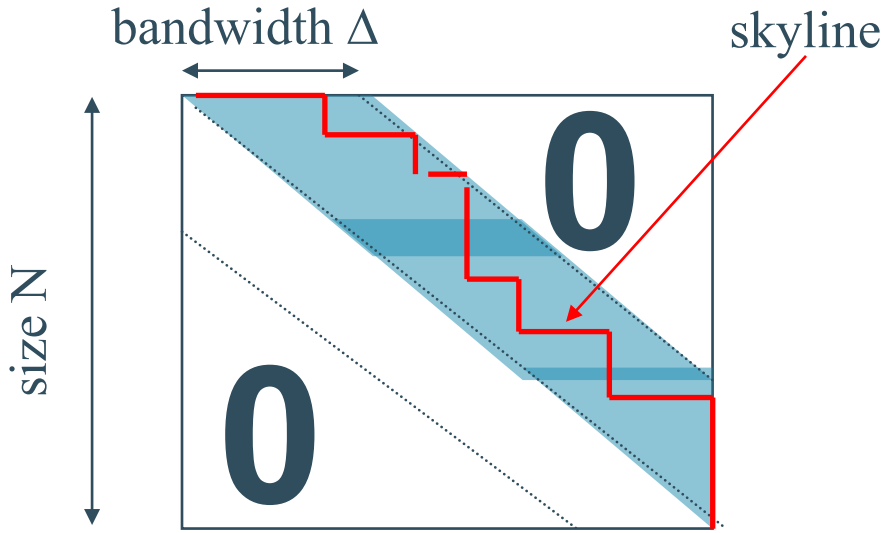
det(J) should not change sign ! ... restriction on mapping

det(J) can be position dependent (function of ξ and η)

Finite element method

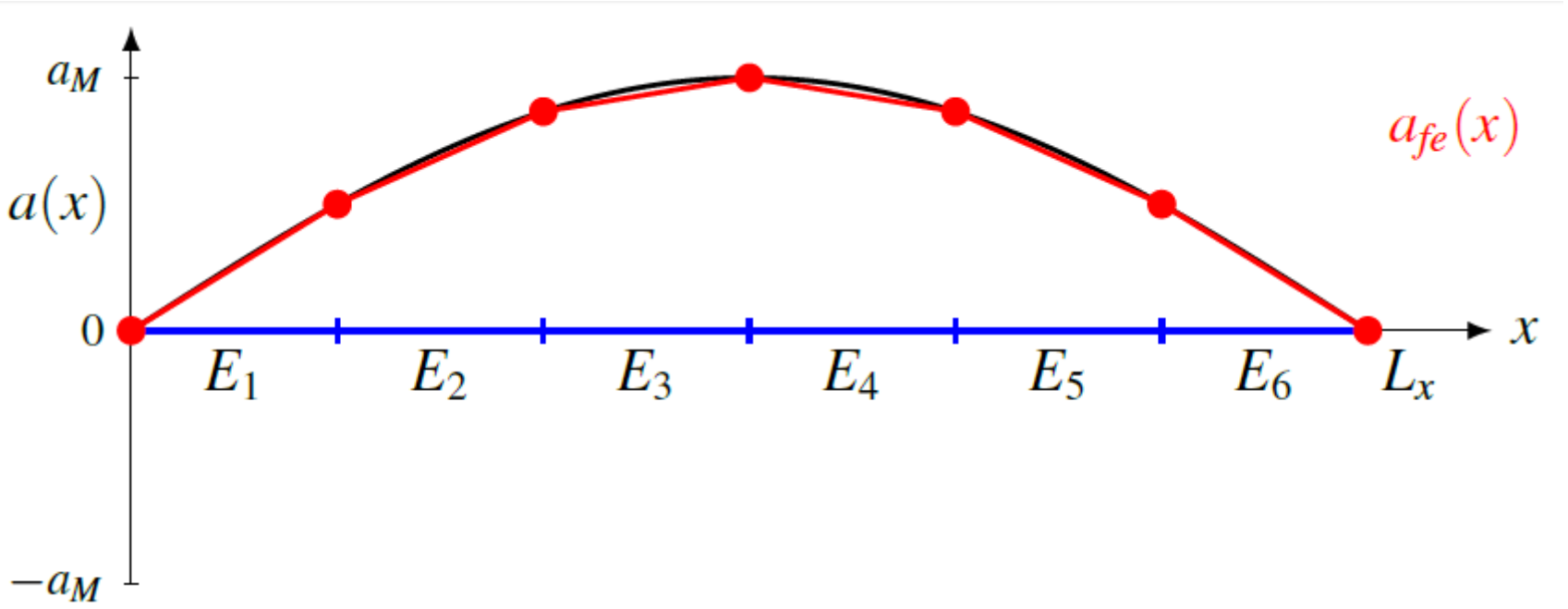
Properties of the resulting matrix

- symmetrical
- sparsely populated
- banded structure
- coefficients obtained from simple numerical integrations
- positive diagonal elements



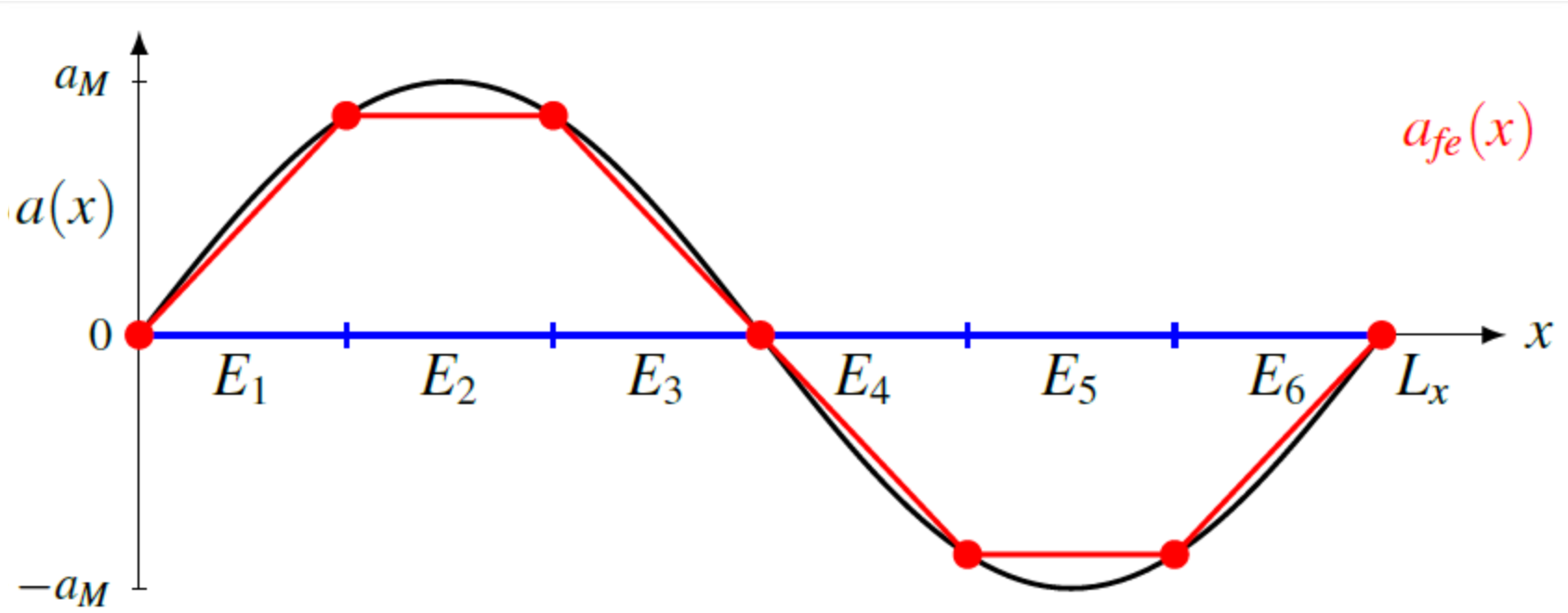
Finite element method

Interpolation error



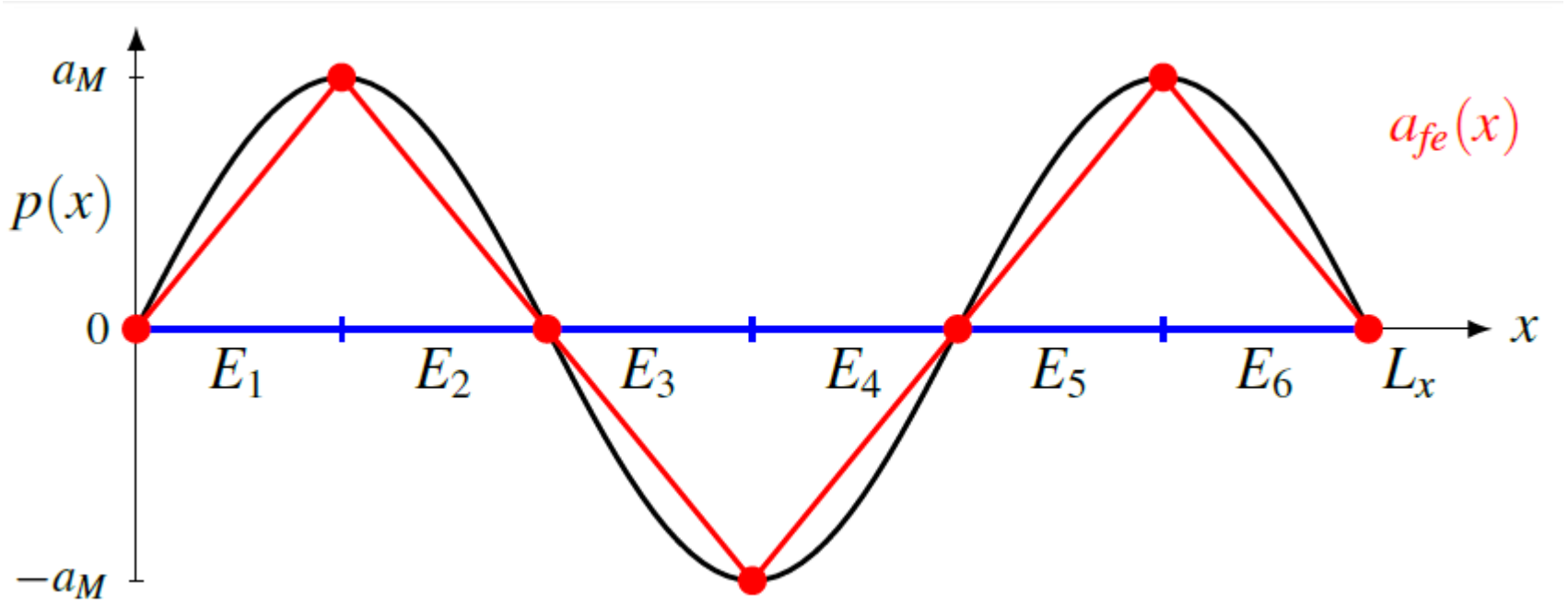
Finite element method

Interpolation error



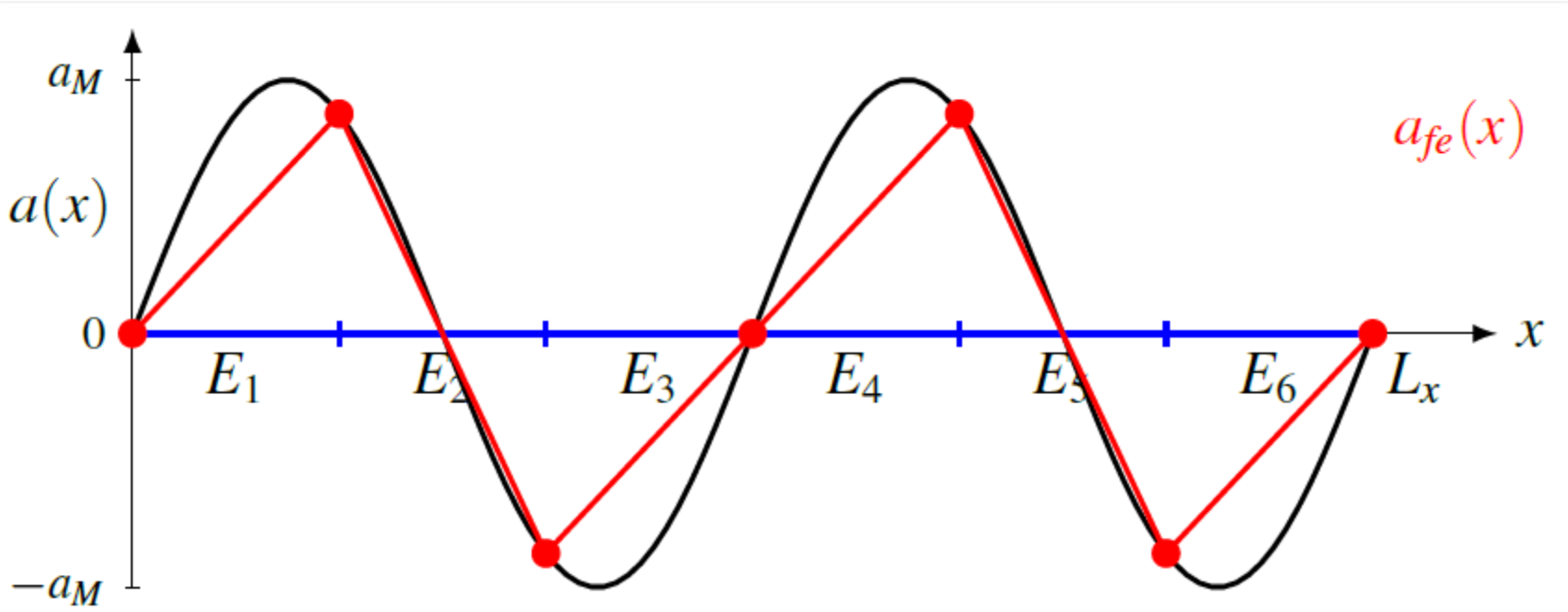
Finite element method

Interpolation error



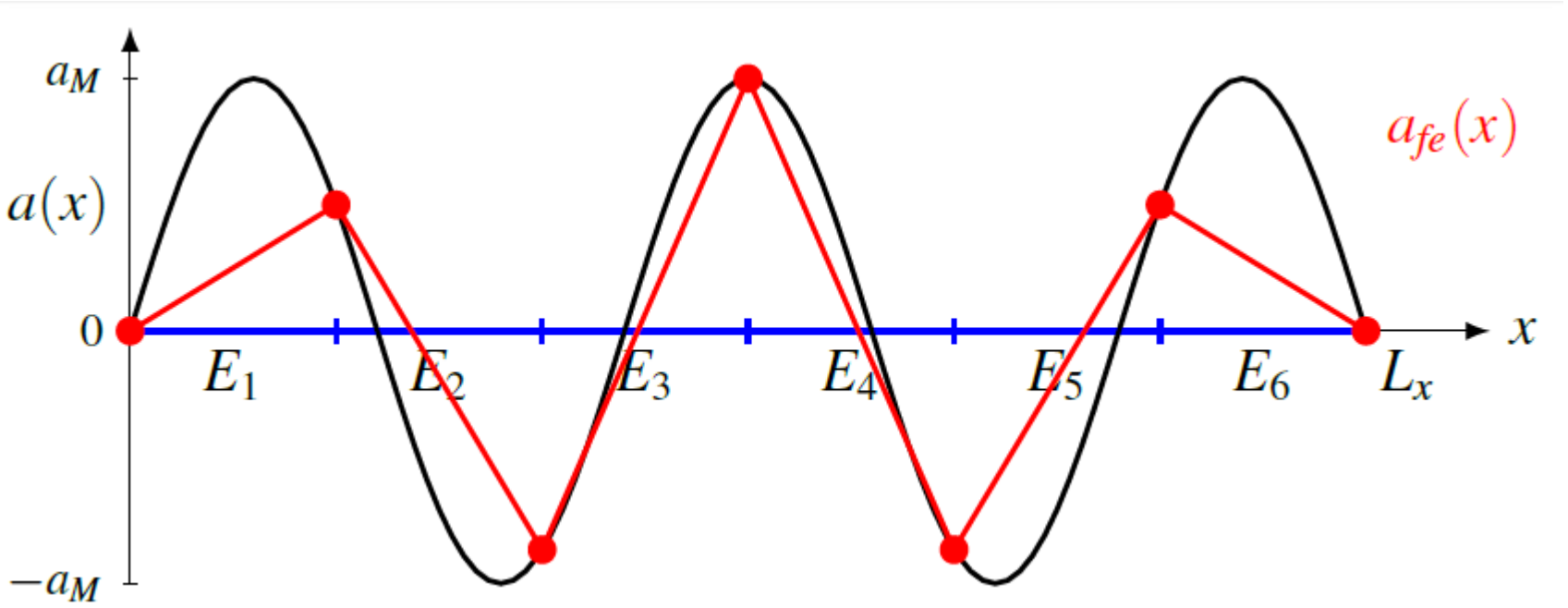
Finite element method

Interpolation error



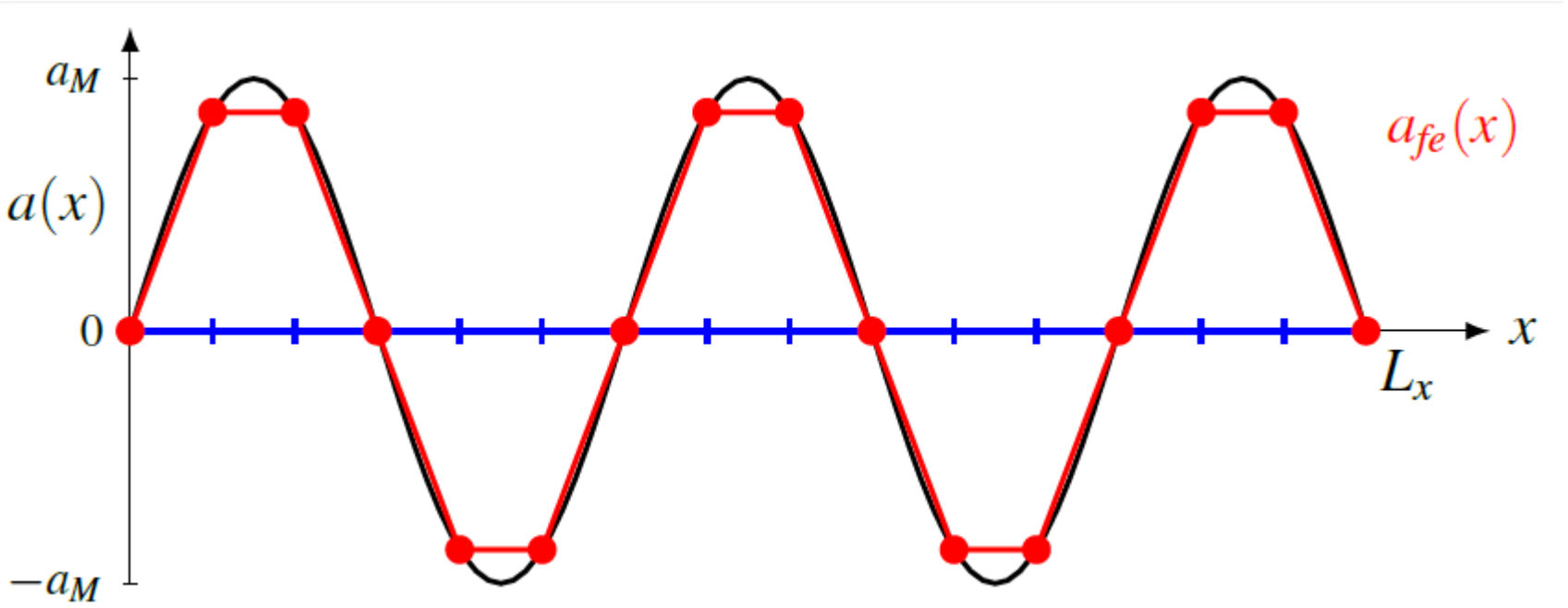
Finite element method

Interpolation error



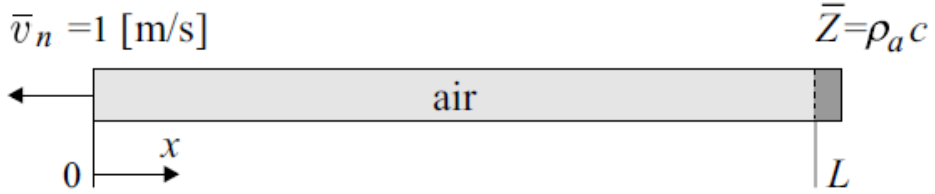
Finite element method

Interpolation error



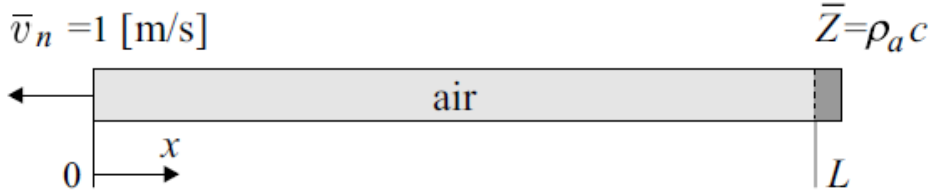
Finite element method

Interpolation and dispersion error



Finite element method

Interpolation and dispersion error



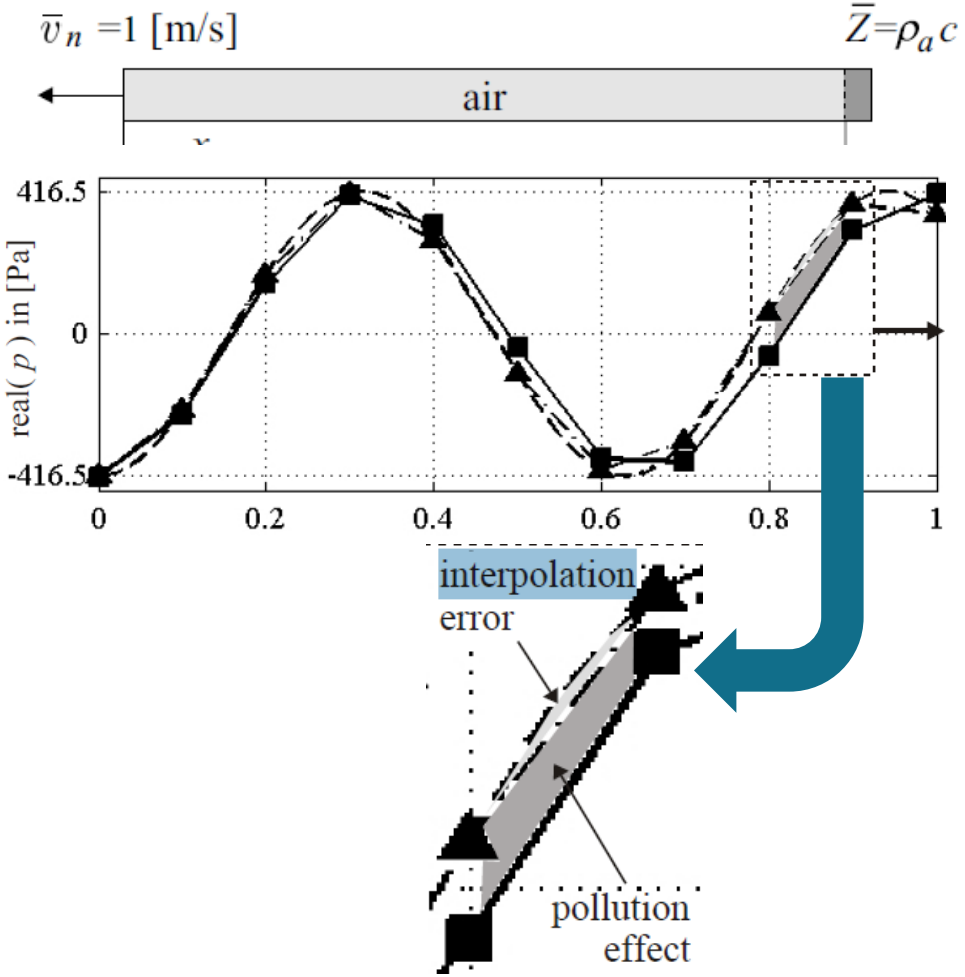
$$\phi \approx \sum_a N_a^e . \phi_a$$

N_a^e

- locally defined within element (size h)
- simple (polynomial) functions (order p)
- restricted spatial variation
- no exact solutions !

Finite element method

Interpolation and dispersion error



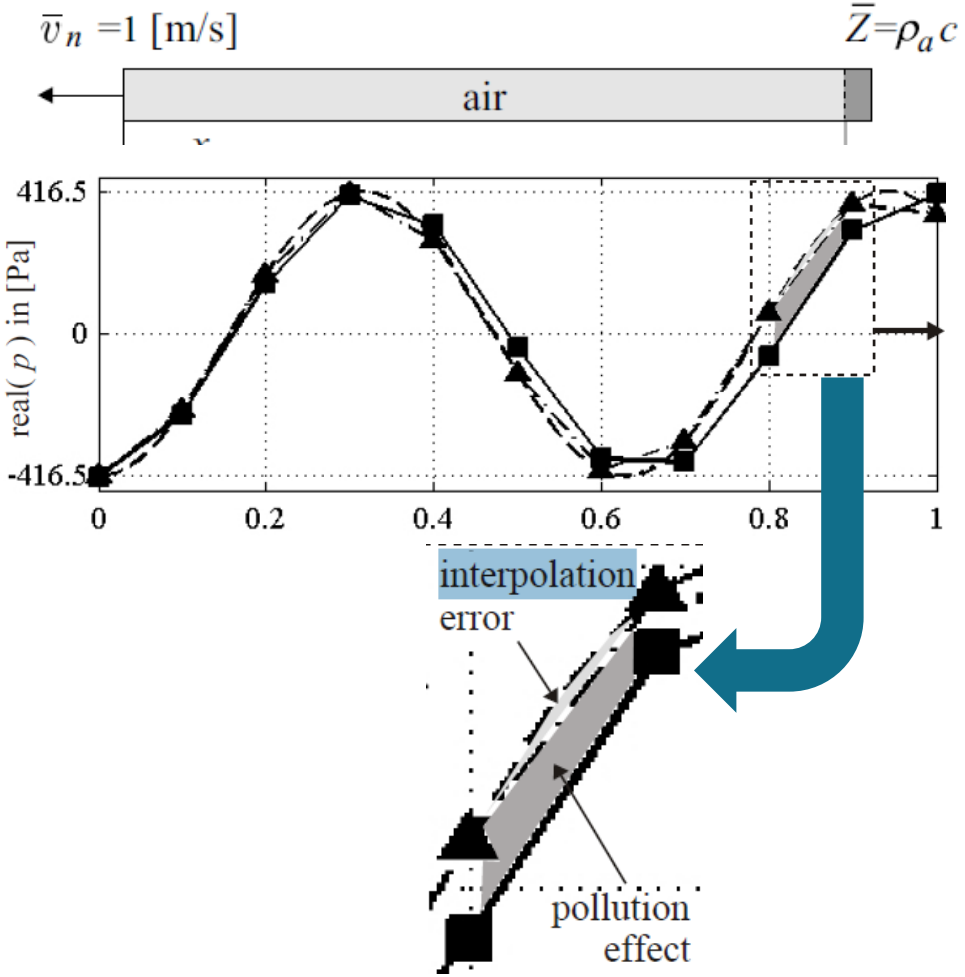
$$\phi \approx \sum_a N_a^e . \phi_a$$

N_a^e

- locally defined within element (size h)
- simple (polynomial) functions (order p)
- restricted spatial variation
- no exact solutions !

Finite element method

Interpolation and dispersion error



$$\phi \approx \sum_a N_a^e . \phi_a$$

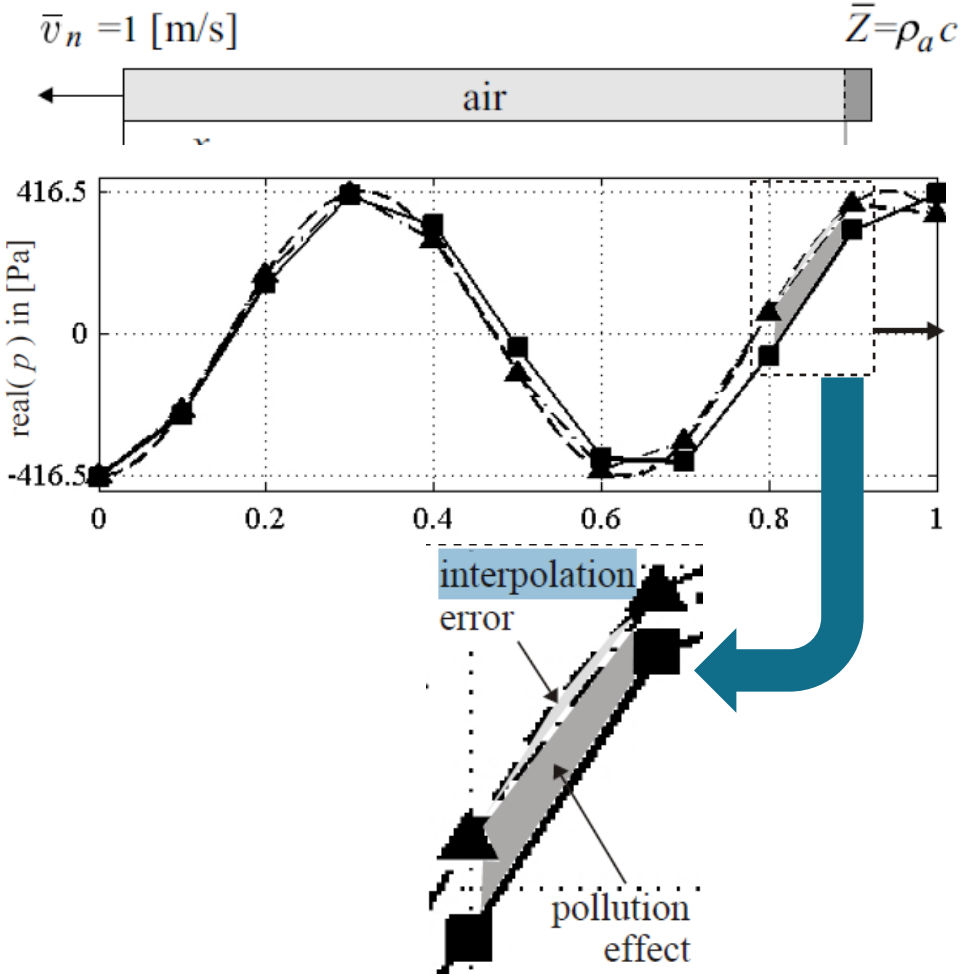
N_a^e

- locally defined within element (size h)
- simple (polynomial) functions (order p)
- restricted spatial variation
- no exact solutions !

$$\varepsilon \leq \underbrace{C_1 . \left(\frac{kh}{p}\right)^p}_{\text{interpolation error}} + \underbrace{C_2 . kL \left(\frac{kh}{p}\right)^{2p}}_{\text{pollution error (dispersion)}}$$

Finite element method

Interpolation and dispersion error



$$\phi \approx \sum_a N_a^e . \phi_a$$

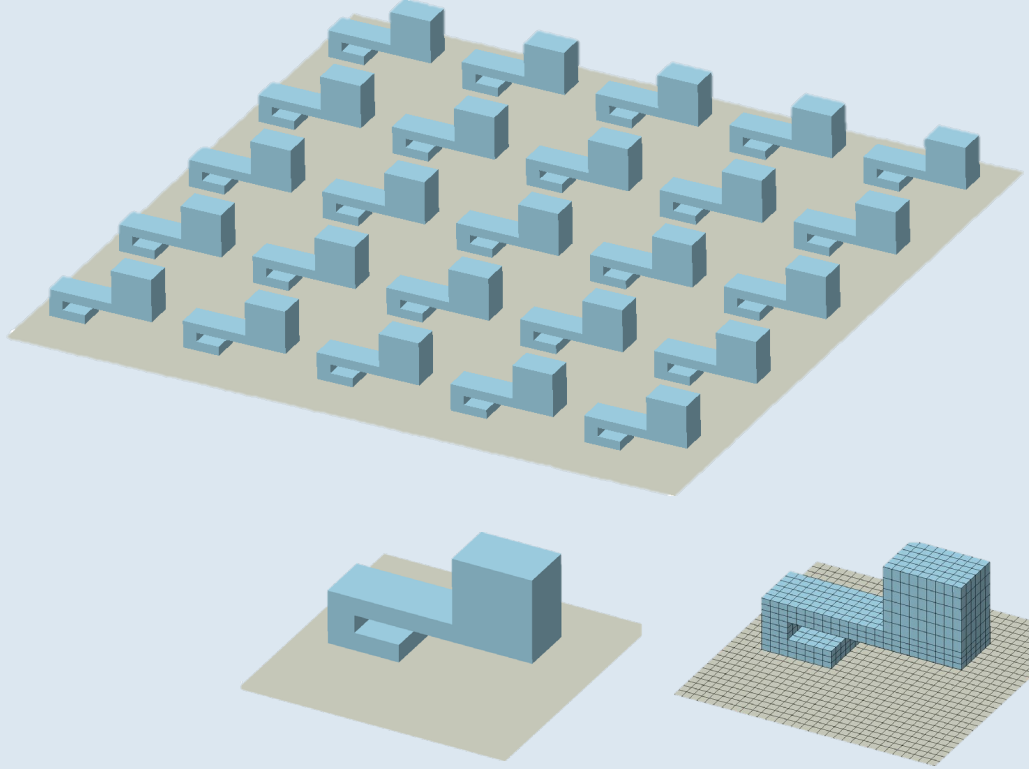
N_a^e

- locally defined within element (size h)
- simple (polynomial) functions (order p)
- restricted spatial variation
- no exact solutions !

$$\varepsilon \leq \underbrace{C_1 . \left(\frac{kh}{p}\right)^p}_{\text{interpolation error}} + \underbrace{C_2 . kL \left(\frac{kh}{p}\right)^{2p}}_{\text{pollution error (dispersion)}}$$

practical frequency limitation !!

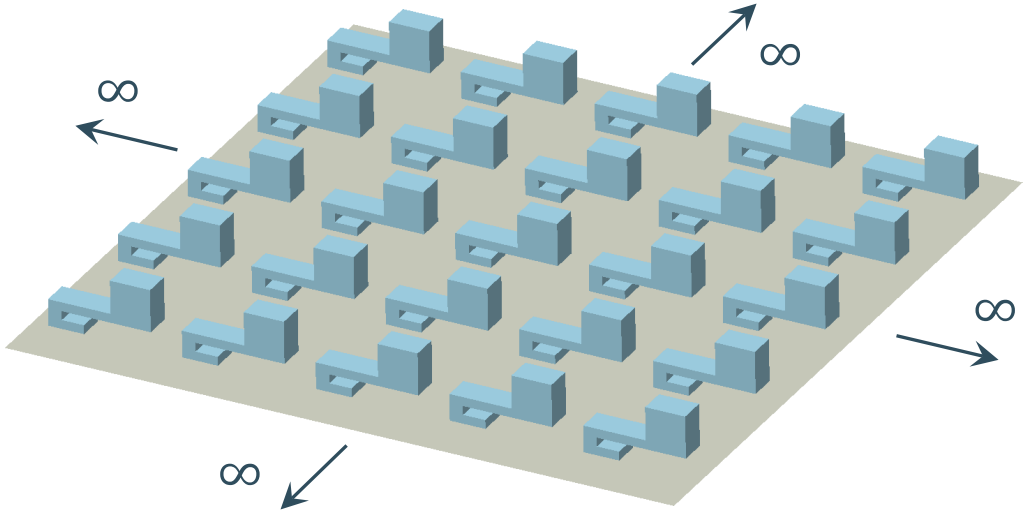
- keeping kh constant … insufficient !



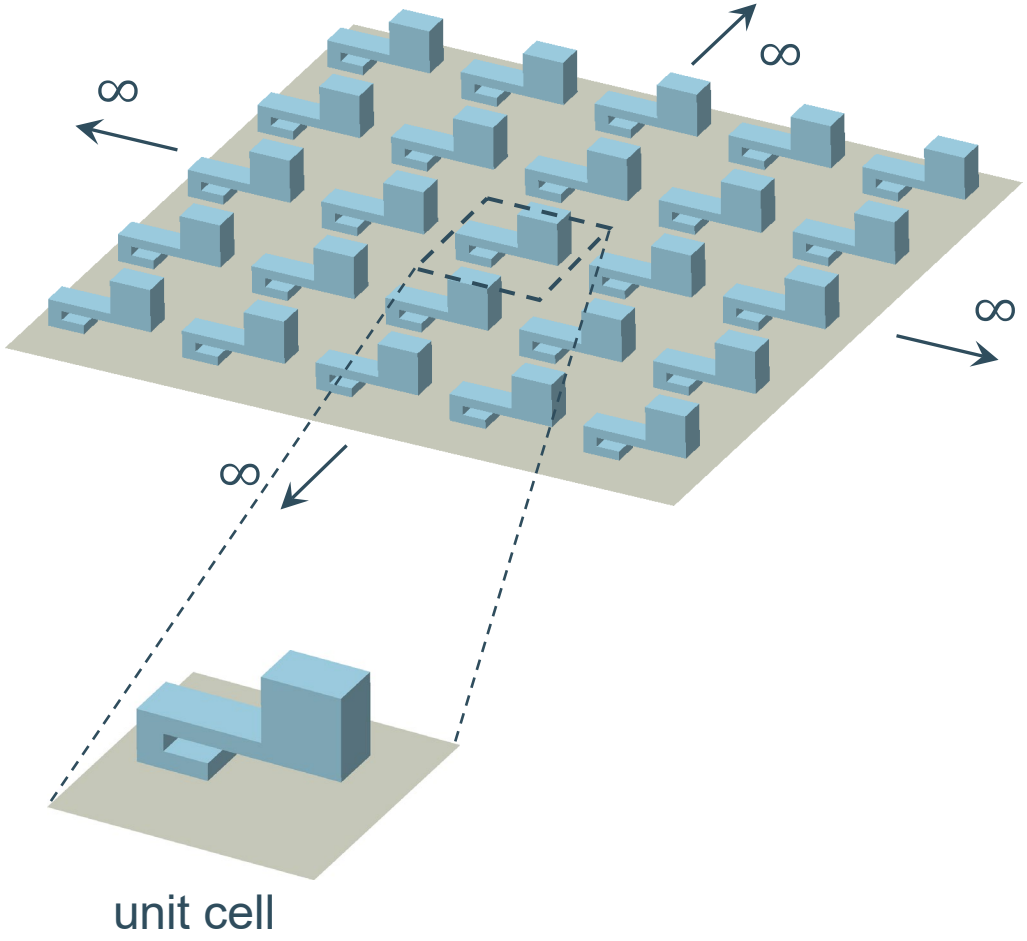
Infinite periodic structure modelling

How can we efficiently model and analyse metamaterials using FEM?

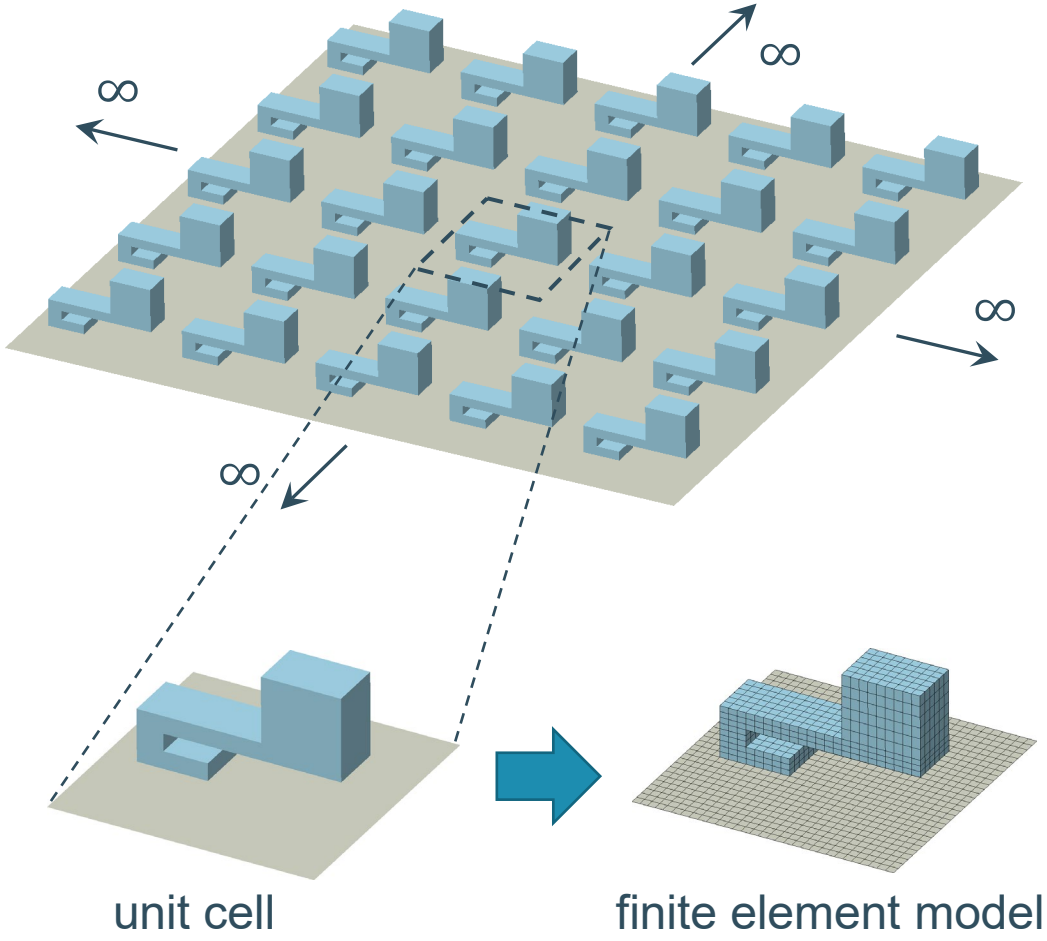
Infinite periodic structure modelling



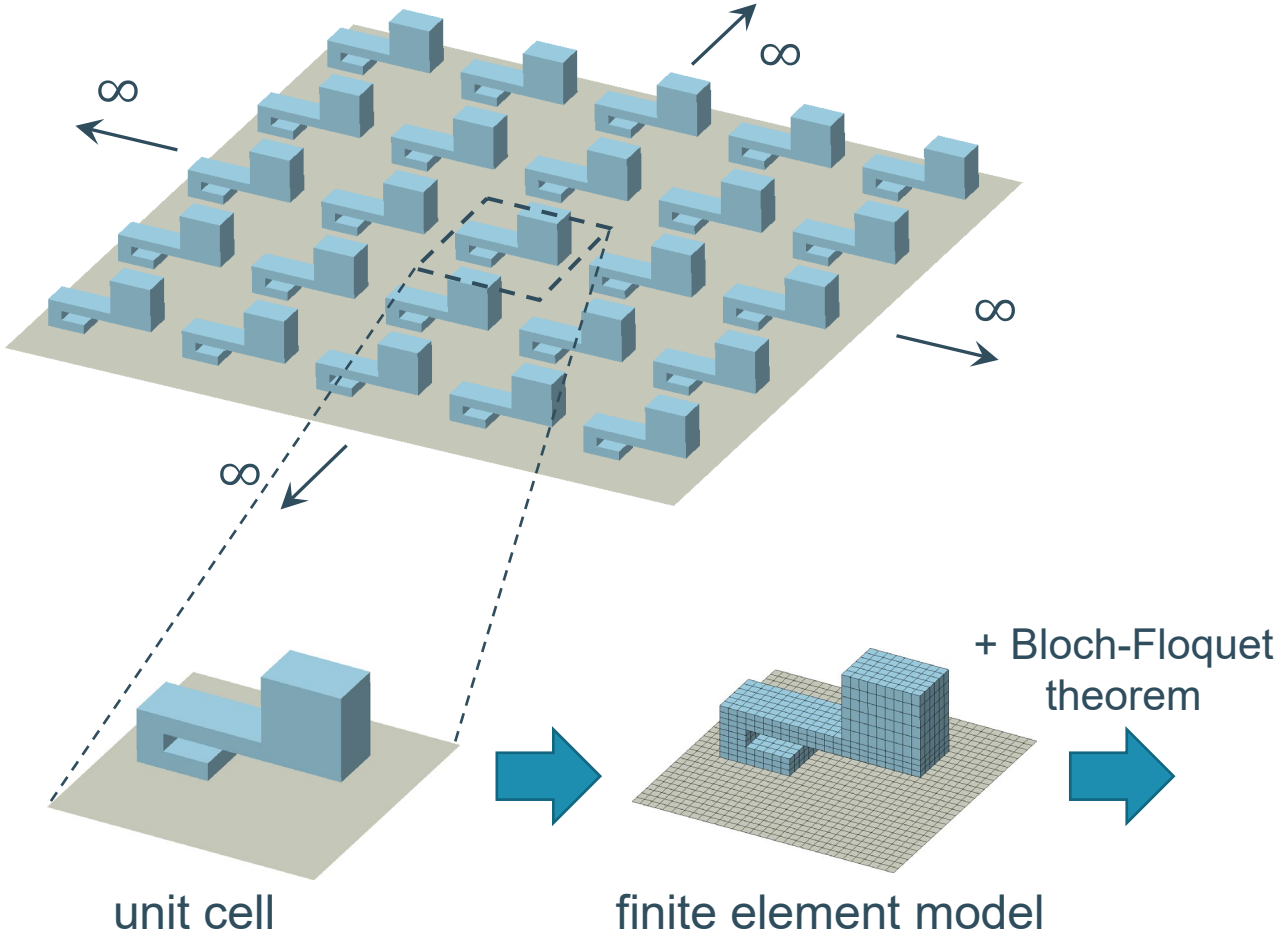
Infinite periodic structure modelling



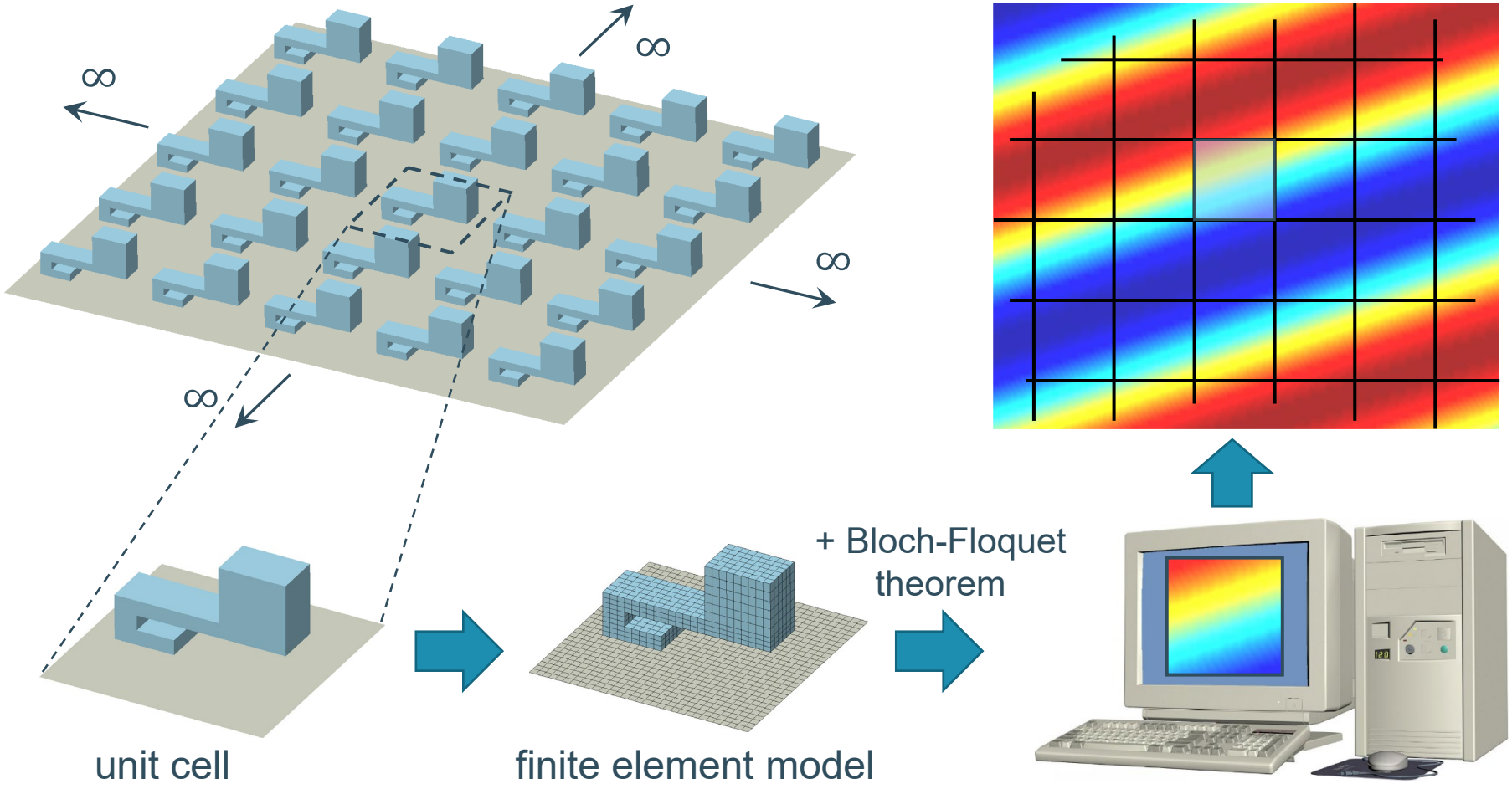
Infinite periodic structure modelling



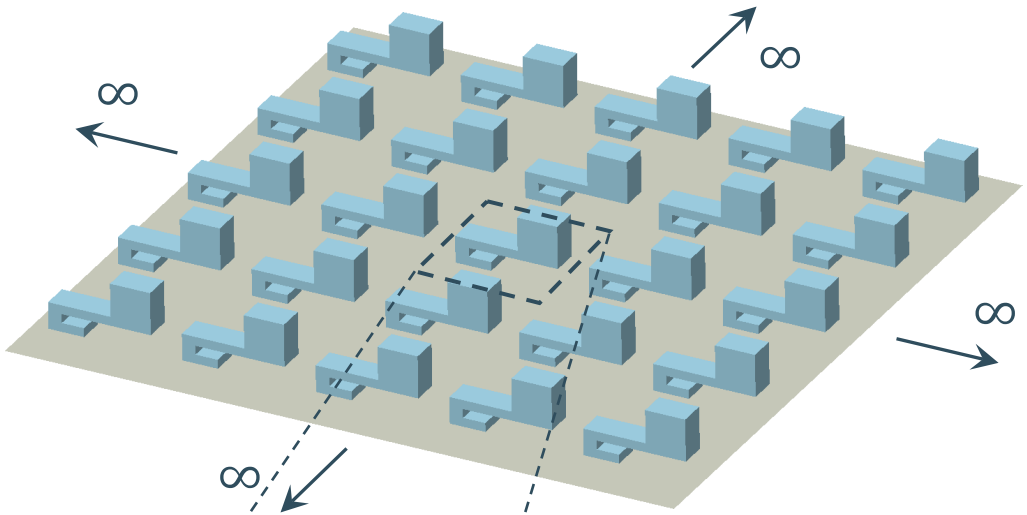
Infinite periodic structure modelling



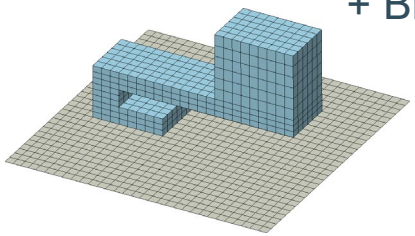
Infinite periodic structure modelling



Infinite periodic structure modelling

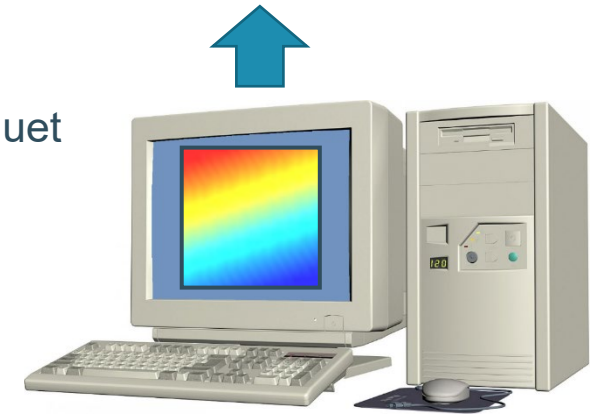


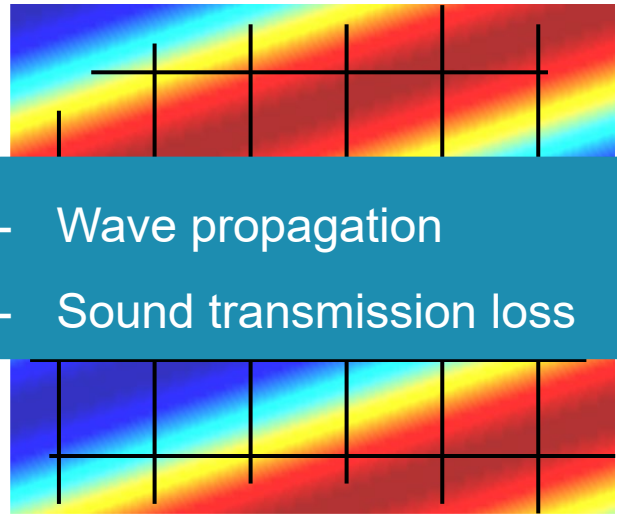
unit cell



finite element model

+ Bloch-Floquet theorem



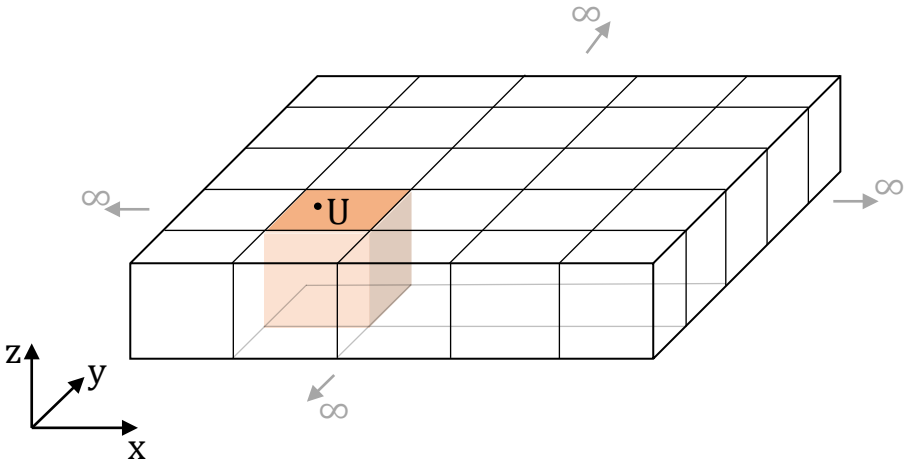


- Wave propagation
- Sound transmission loss

Unit cell modeling

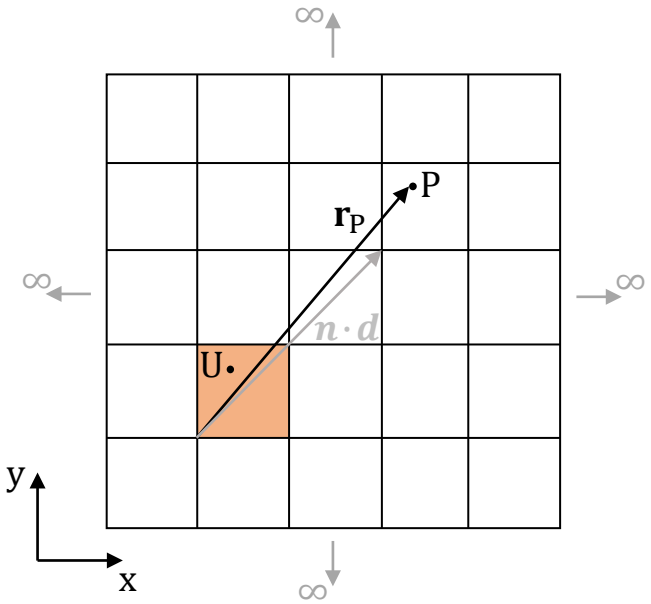
From unit cell model to dispersion eigenvalue problem

In a nutshell



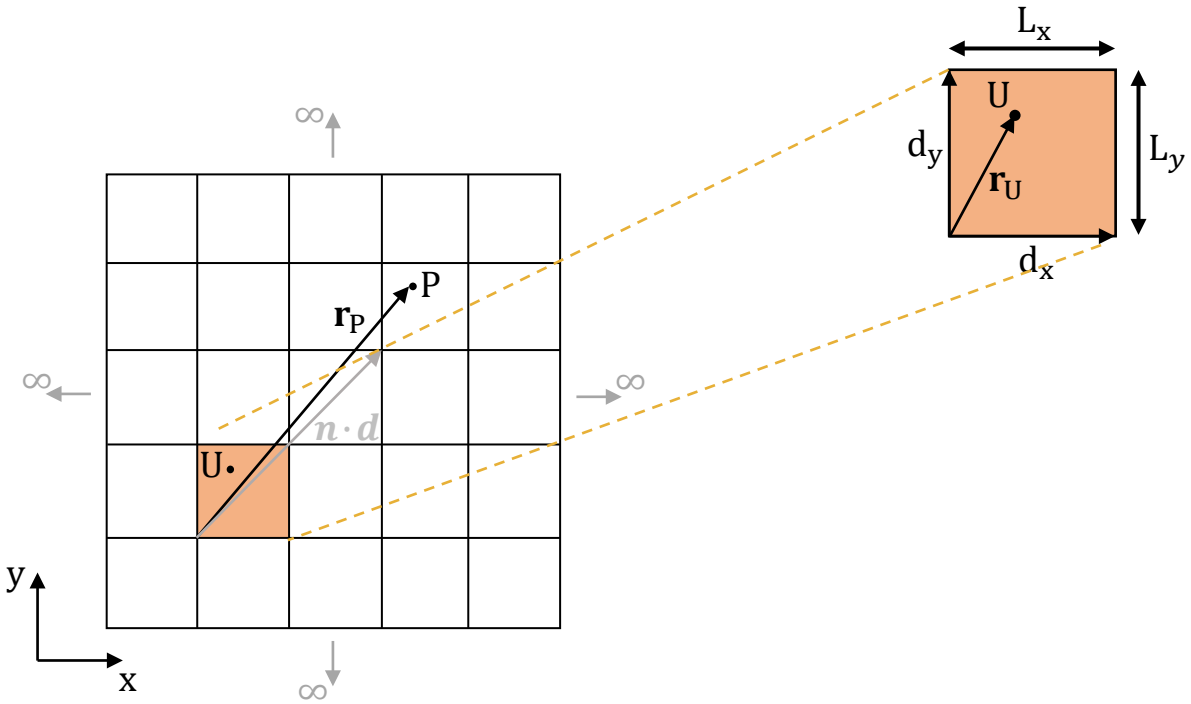
B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In a nutshell



B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

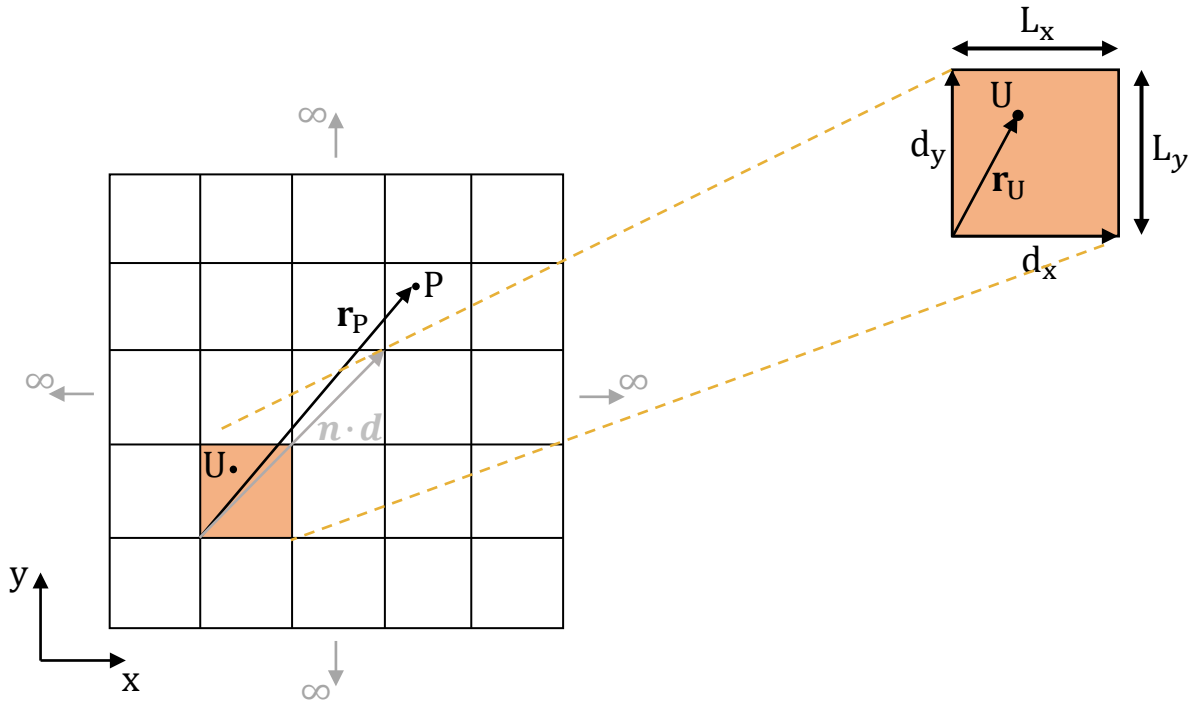
In a nutshell



B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In a nutshell

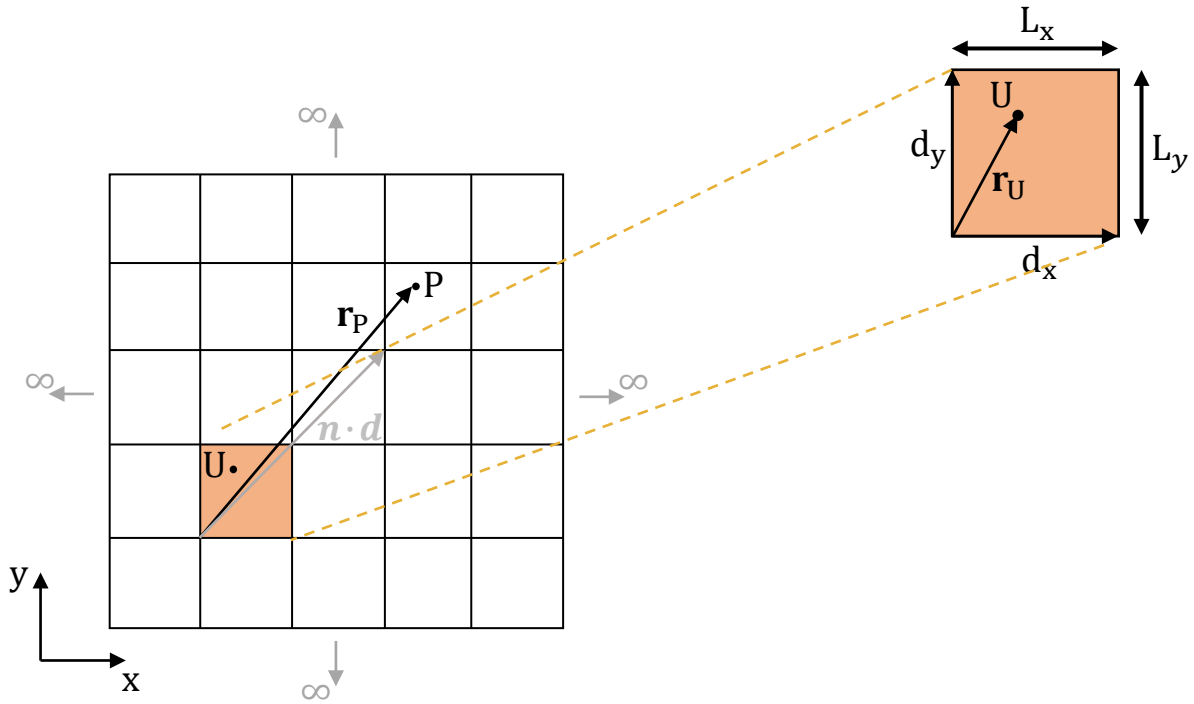
$$\boldsymbol{r}_P = \boldsymbol{r}_U + n_x \boldsymbol{d}_x + n_y \boldsymbol{d}_y$$



B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In a nutshell

$$\boldsymbol{r}_P = \boldsymbol{r}_U + n_x \boldsymbol{d}_x + n_y \boldsymbol{d}_y$$



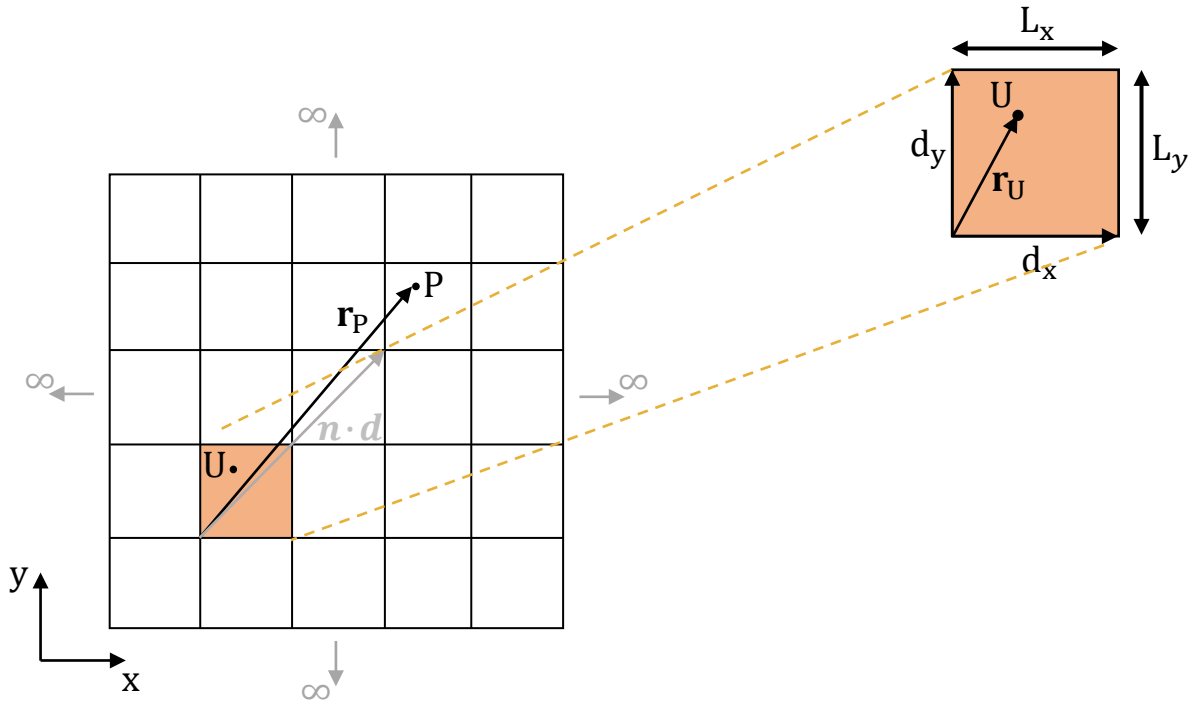
UC equations of motion (FEM)

$$(\mathbf{K} + j\omega\mathbf{C} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In a nutshell

$$\boldsymbol{r}_P = \boldsymbol{r}_U + n_x \boldsymbol{d}_x + n_y \boldsymbol{d}_y$$



UC equations of motion (FEM)

$$(\mathbf{K} + j\omega\mathbf{C} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

Bloch-Floquet theorem

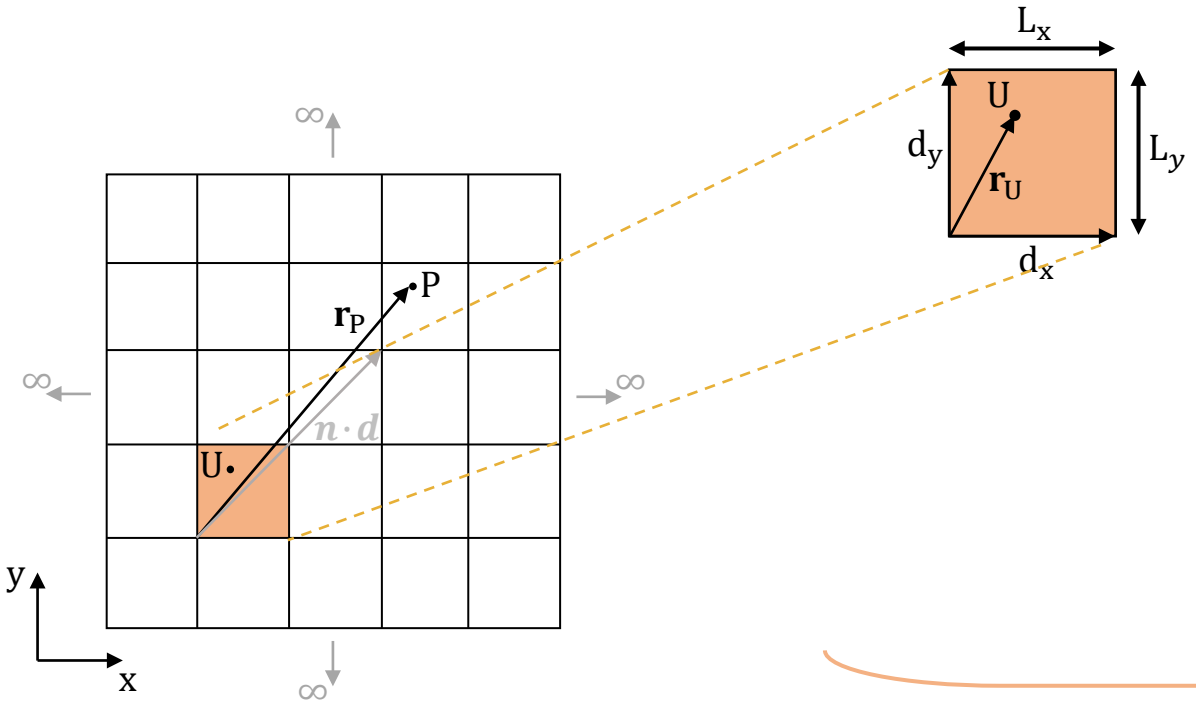
$$\mathbf{q}(\boldsymbol{r}_P) = \mathbf{q}(\boldsymbol{r}_U) e^{j(n_x\mu_x + n_y\mu_y)}$$

$$(\mu_x, \mu_y) = (\boldsymbol{k} \cdot \boldsymbol{d}_x, \boldsymbol{k} \cdot \boldsymbol{d}_y)$$

B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In a nutshell

$$\boldsymbol{r}_P = \boldsymbol{r}_U + n_x \boldsymbol{d}_x + n_y \boldsymbol{d}_y$$



UC equations of motion (FEM)

$$(\mathbf{K} + j\omega\mathbf{C} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

Bloch-Floquet theorem

$$\mathbf{q}(\boldsymbol{r}_P) = \mathbf{q}(\boldsymbol{r}_U) e^{j(n_x\mu_x + n_y\mu_y)}$$

$$(\mu_x, \mu_y) = (\boldsymbol{k} \cdot \boldsymbol{d}_x, \boldsymbol{k} \cdot \boldsymbol{d}_y)$$

$$\mathbf{A}(\omega, \mu_x, \mu_y)\tilde{\mathbf{q}} = \mathbf{0}$$

Dispersion eigenvalue problem → dispersion curves

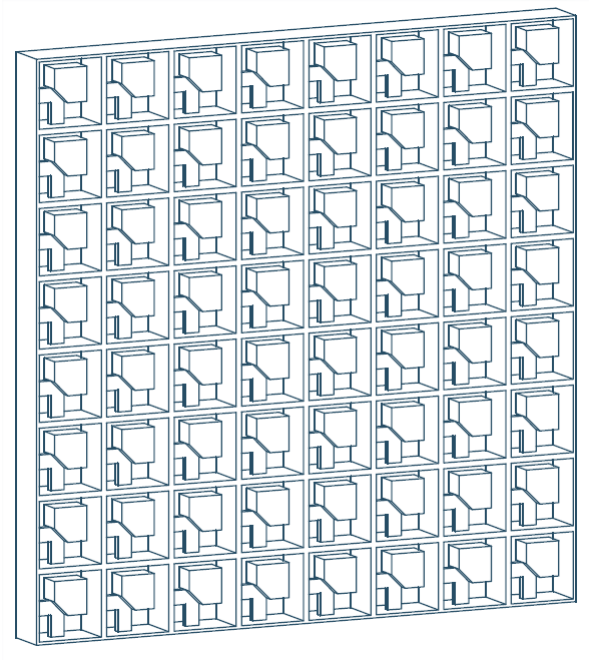
B. Mace and E. Manconi. Modelling wave propagation in two-dimensional structures using finite element analysis. *Journal of Sound and Vibration*, 318(4-5):884-902, 2008.

In more detail

Demo acoustic metamaterial: acoustic enclosure - YouTube

In more detail

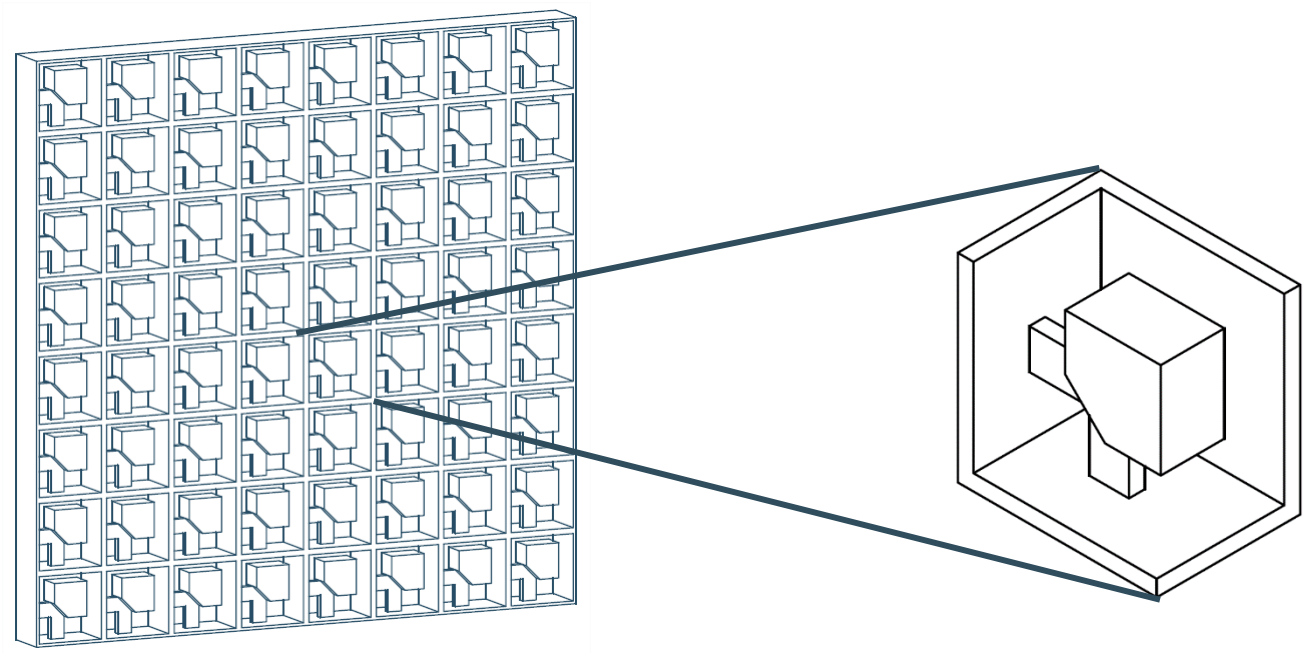
exploiting periodicity



Demo acoustic metamaterial: acoustic enclosure - YouTube

In more detail

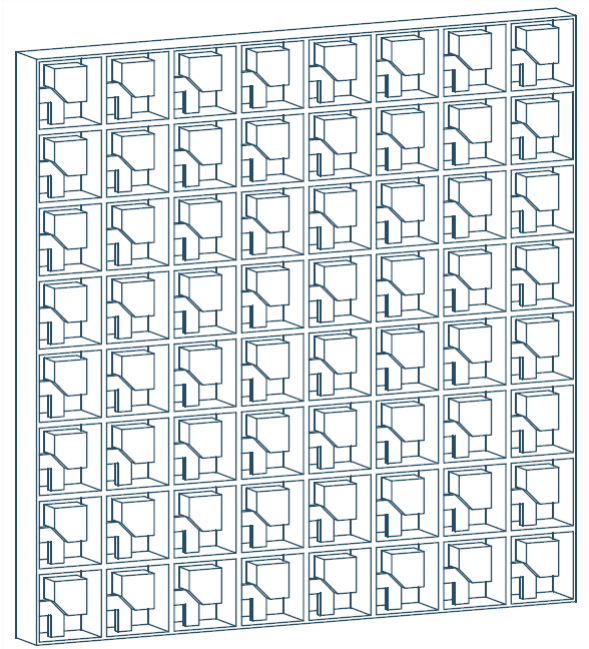
exploiting periodicity

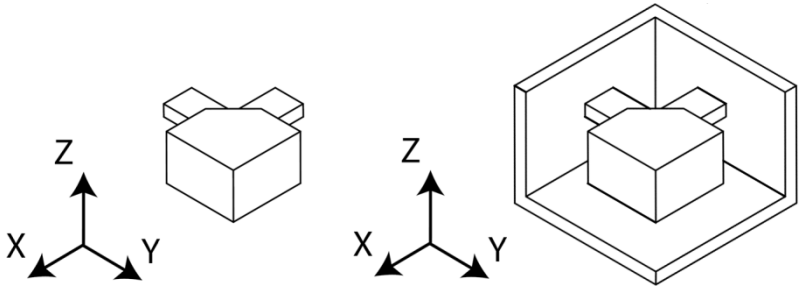


by considering a **unit cell (UC)**

Demo acoustic metamaterial: acoustic enclosure - YouTube

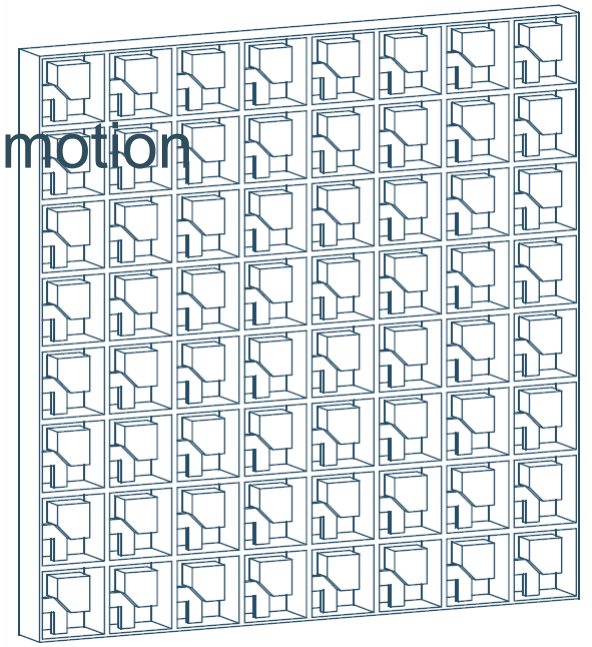
Unit cell model

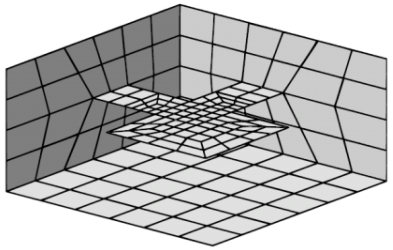


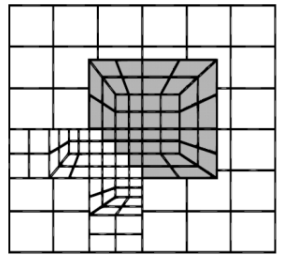


Unit cell model

- Finite Element method
- Equations of motion





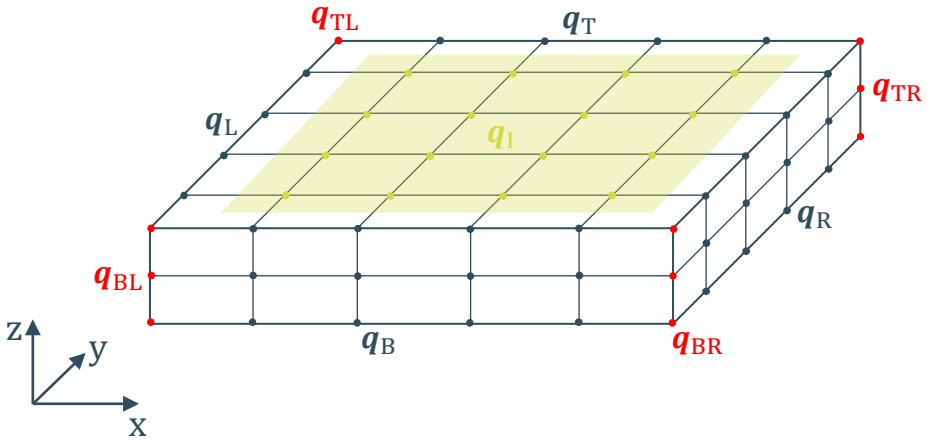


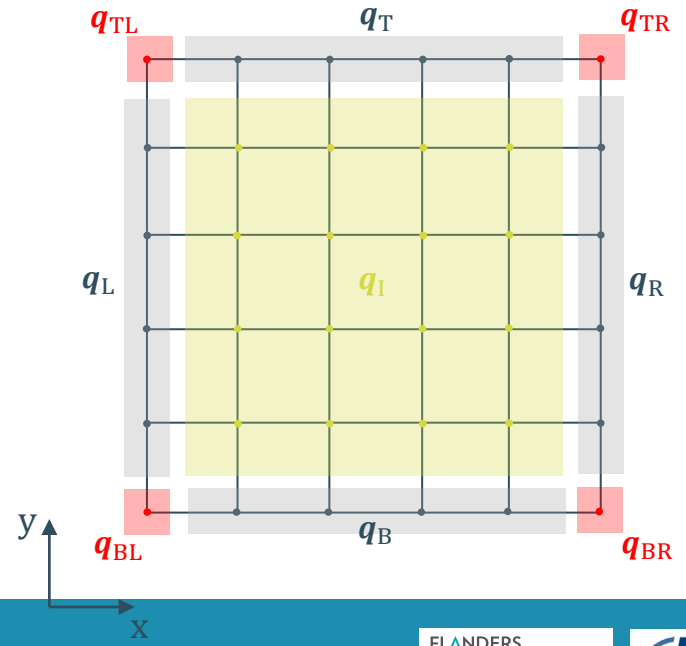
Unit cell model

- Finite Element method
- Equations of motion

Unit cell model

- Finite Element method
- Equations of motion



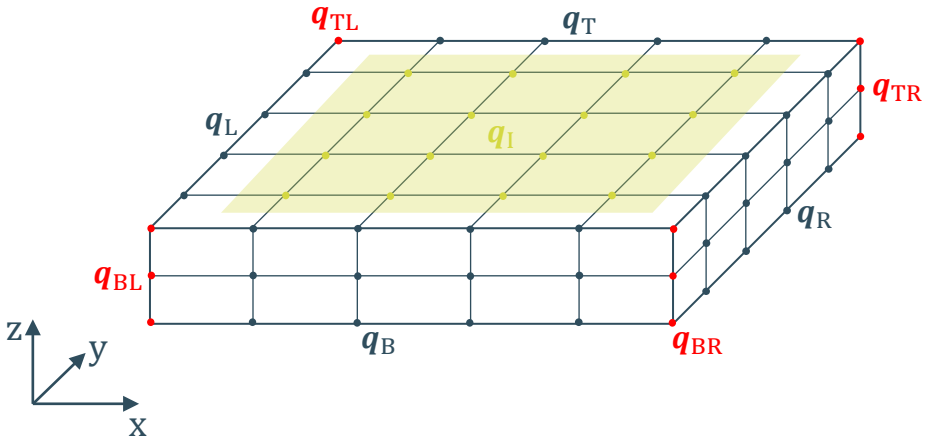


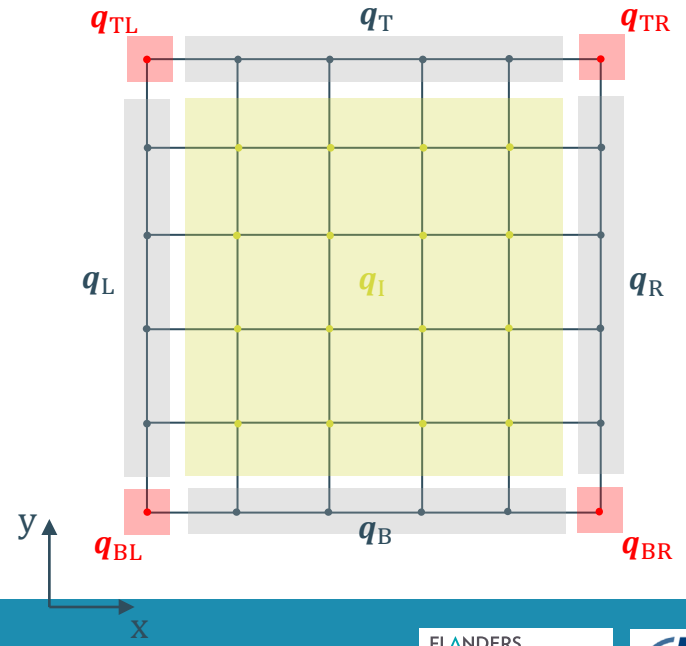
Unit cell model

- Finite Element method
- Equations of motion

$$(\mathbf{K} + j\omega\mathbf{C} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\mathbf{D}\mathbf{q} = \mathbf{f}$$



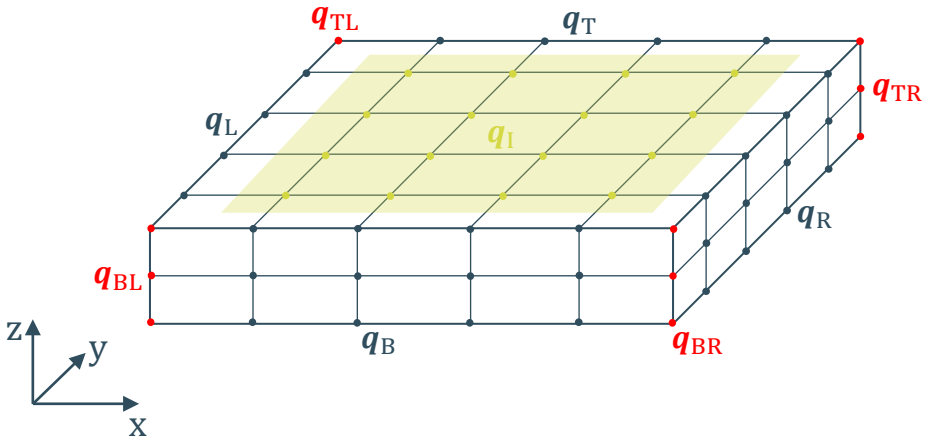


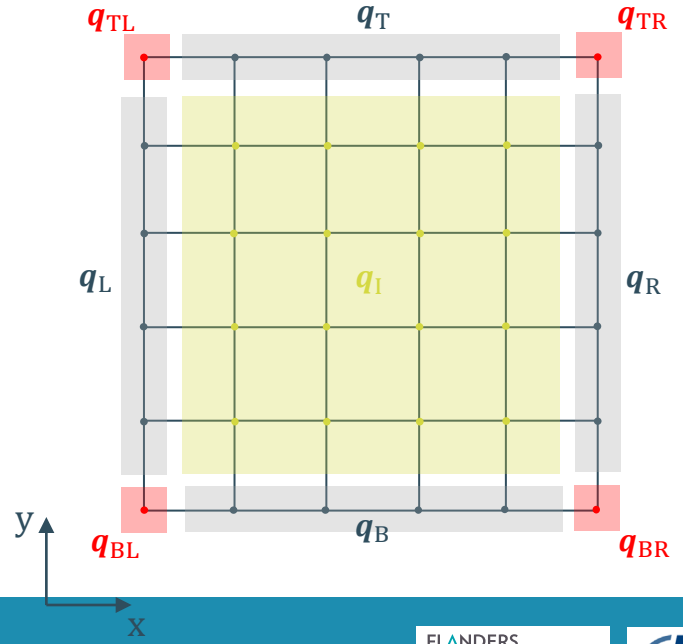
Unit cell model

- Finite Element method
- Equations of motion

$$(\mathbf{K} + j\omega\mathbf{C} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\mathbf{D}\mathbf{q} = \mathbf{f}$$



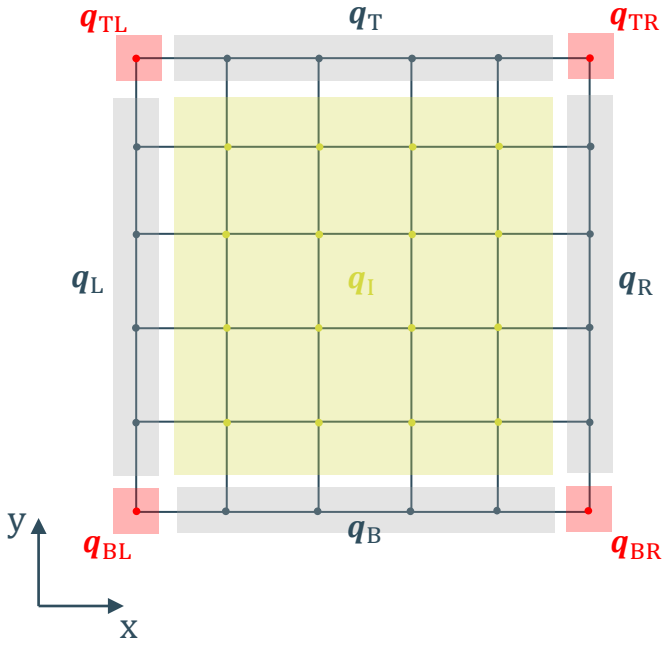


Note:

$$[K^e] = \int_{\Omega^e} [B^e]^T [E][B^e] \mathrm{d}\Omega$$

$$[M^e] = \int_{\Omega^e} \rho [N^e]^T [N^e] \mathrm{d}\Omega$$

Displacement continuity



Displacement continuity

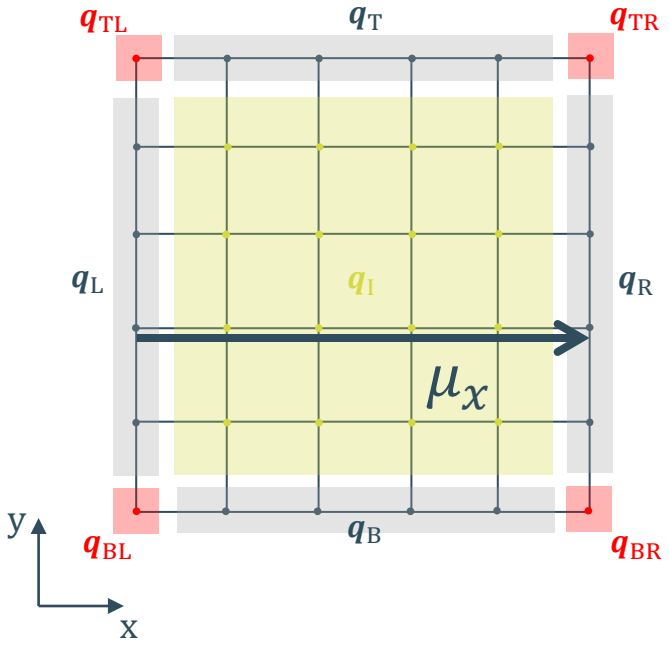
$$\mathbf{q}_R = e^{j\mu_x}\mathbf{q}_L = \lambda_x \mathbf{q}_L$$

$$\mathbf{q}_{BR} = e^{j\mu_x}\mathbf{q}_{BL} = \lambda_x \mathbf{q}_{BL}$$

$$\mathbf{q}_T = e^{j\mu_y}\mathbf{q}_B = \lambda_y \mathbf{q}_B$$

$$\mathbf{q}_{TL} = e^{j\mu_y}\mathbf{q}_{BL} = \lambda_y \mathbf{q}_{BL}$$

$$\mathbf{q}_{TR} = e^{j(\mu_x+\mu_y)}\mathbf{q}_{BL} = \lambda_x\lambda_y \mathbf{q}_{BL}$$



Displacement continuity

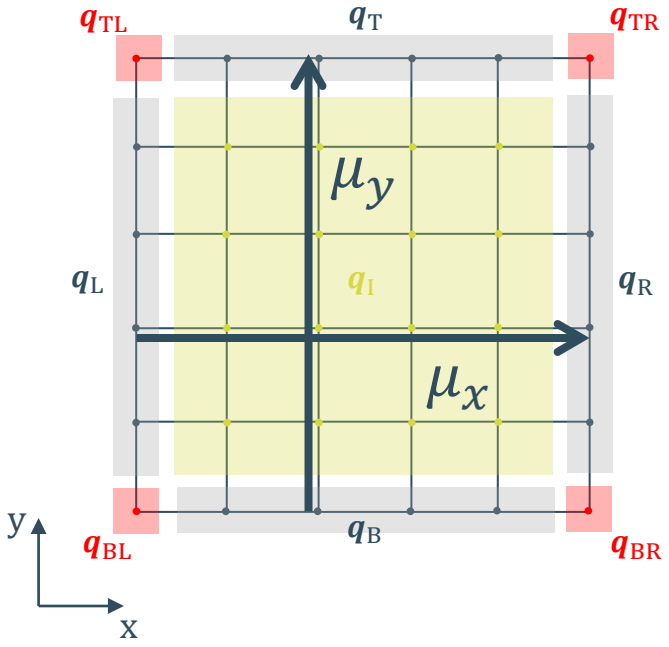
$$\mathbf{q}_R = e^{j\mu_x}\mathbf{q}_L = \lambda_x \mathbf{q}_L$$

$$\mathbf{q}_{BR} = e^{j\mu_x}\mathbf{q}_{BL} = \lambda_x \mathbf{q}_{BL}$$

$$\mathbf{q}_T = e^{j\mu_y}\mathbf{q}_B = \lambda_y \mathbf{q}_B$$

$$\mathbf{q}_{TL} = e^{j\mu_y}\mathbf{q}_{BL} = \lambda_y \mathbf{q}_{BL}$$

$$\mathbf{q}_{TR} = e^{j(\mu_x+\mu_y)}\mathbf{q}_{BL} = \lambda_x\lambda_y \mathbf{q}_{BL}$$



Displacement continuity

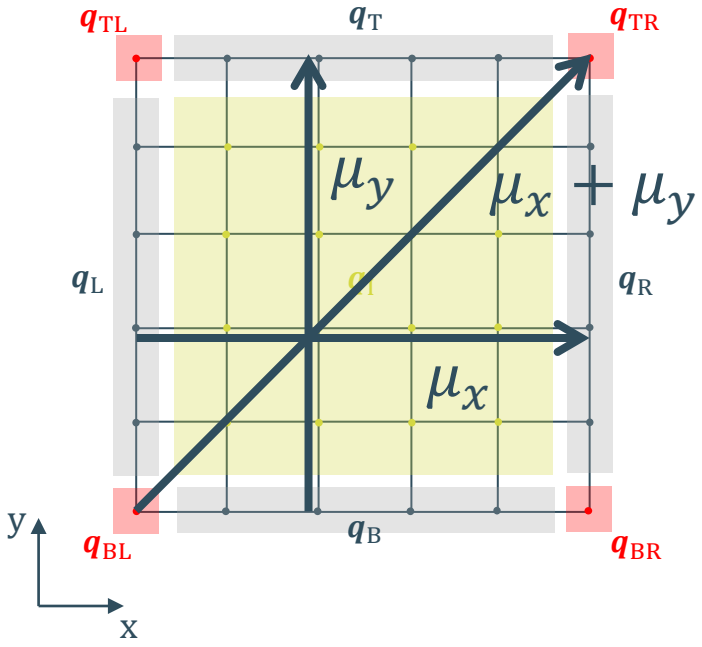
$$\mathbf{q}_R = e^{j\mu_x}\mathbf{q}_L = \lambda_x \mathbf{q}_L$$

$$\mathbf{q}_{BR} = e^{j\mu_x}\mathbf{q}_{BL} = \lambda_x \mathbf{q}_{BL}$$

$$\mathbf{q}_T = e^{j\mu_y}\mathbf{q}_B = \lambda_y \mathbf{q}_B$$

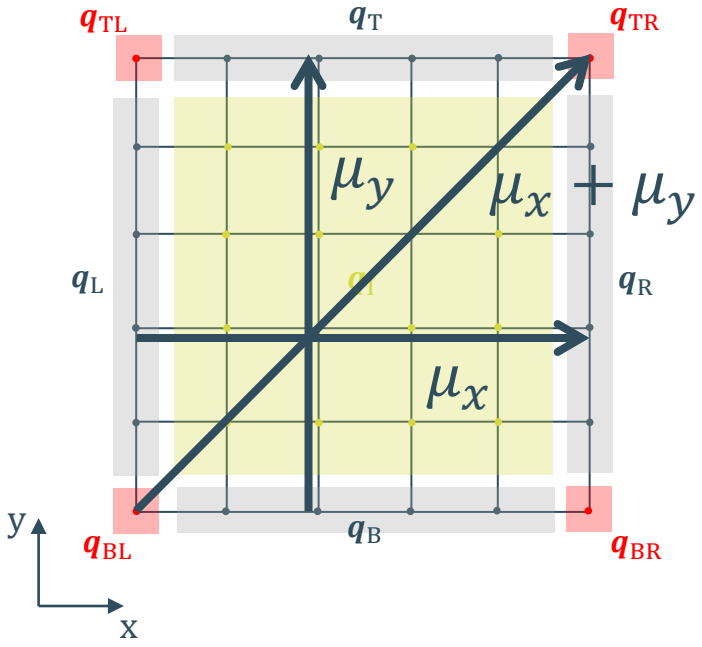
$$\mathbf{q}_{TL} = e^{j\mu_y}\mathbf{q}_{BL} = \lambda_y \mathbf{q}_{BL}$$

$$\mathbf{q}_{TR} = e^{j(\mu_x+\mu_y)}\mathbf{q}_{BL} = \lambda_x\lambda_y \mathbf{q}_{BL}$$



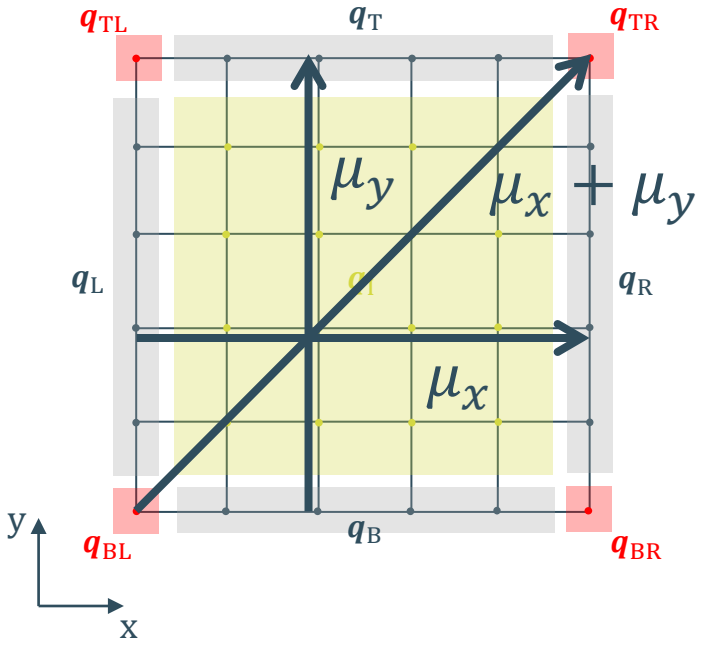
Displacement continuity

$$\left.\begin{aligned}
\mathbf{q}_R &= e^{j\mu_x}\mathbf{q}_L = \lambda_x\mathbf{q}_L \\
\mathbf{q}_{BR} &= e^{j\mu_x}\mathbf{q}_{BL} = \lambda_x\mathbf{q}_{BL} \\
\mathbf{q}_T &= e^{j\mu_y}\mathbf{q}_B = \lambda_y\mathbf{q}_B \\
\mathbf{q}_{TL} &= e^{j\mu_y}\mathbf{q}_{BL} = \lambda_y\mathbf{q}_{BL} \\
\mathbf{q}_{TR} &= e^{j(\mu_x+\mu_y)}\mathbf{q}_{BL} = \lambda_x\lambda_y\mathbf{q}_{BL}
\end{aligned}\right\} \quad \mathbf{q} = \boldsymbol{R}\begin{Bmatrix}\mathbf{q}_{BL}\\ \mathbf{q}_L\\ \mathbf{q}_B\\ \mathbf{q}_I\end{Bmatrix}$$

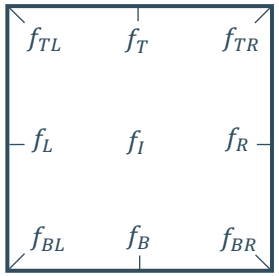


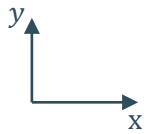
Displacement continuity

$$\mathbf{q} = \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & \lambda_x \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & \lambda_y \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \\ 0 & 0 & 0 & \lambda_x \mathbf{I} \\ 0 & 0 & 0 & \lambda_y \mathbf{I} \\ 0 & 0 & 0 & \lambda_x \lambda_y \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{q}_I \\ \mathbf{q}_L \\ \mathbf{q}_B \\ \mathbf{q}_{BL} \end{bmatrix} = \mathbf{R}\tilde{\mathbf{q}}$$

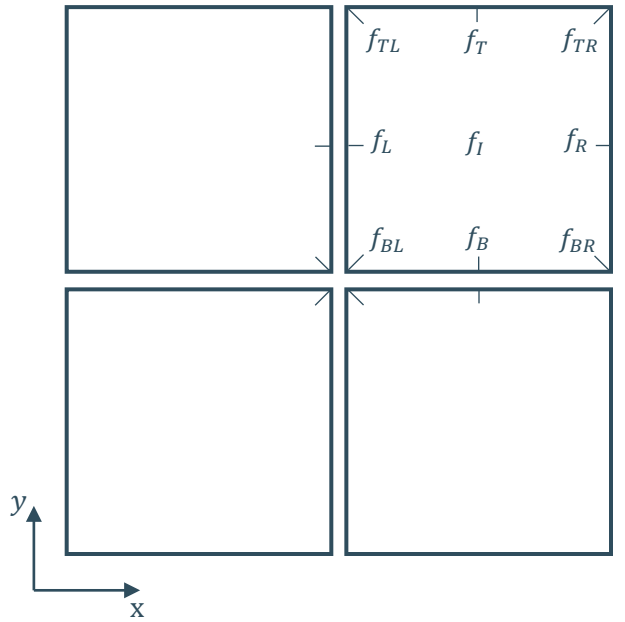


Force balance



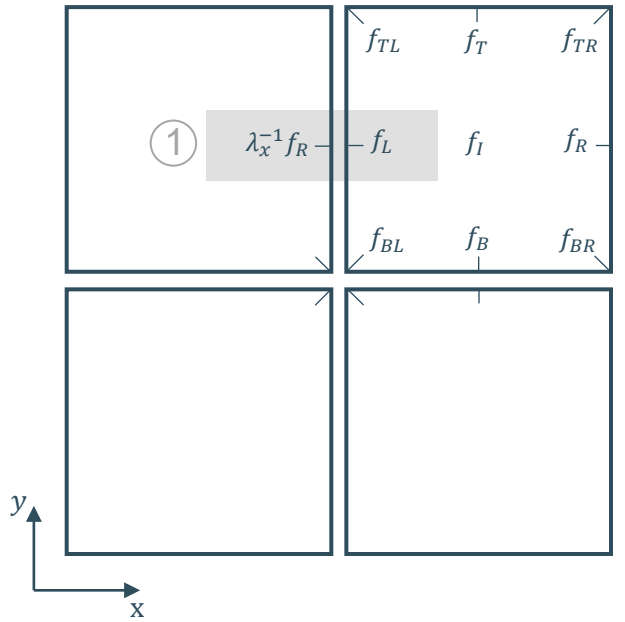


Force balance



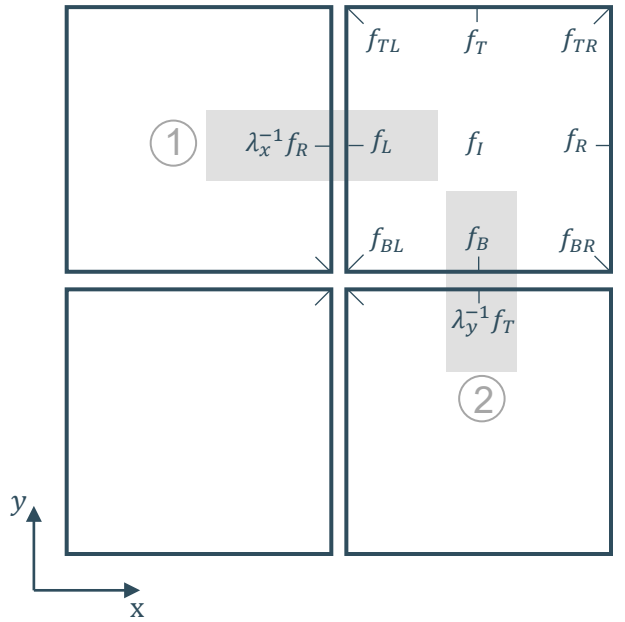
Force balance

$$\mathbf{0} = \lambda_x^{-1}\mathbf{f}_R + \mathbf{f}_L$$
$$\mathbf{0} = \lambda_y^{-1}\mathbf{f}_T + \mathbf{f}_B$$
$$\mathbf{0} = \lambda_x^{-1}\lambda_y^{-1}\mathbf{f}_{TR} + \lambda_y^{-1}\mathbf{f}_{TL} + \lambda_x^{-1}\mathbf{f}_{BR} + \mathbf{f}_{BL}$$



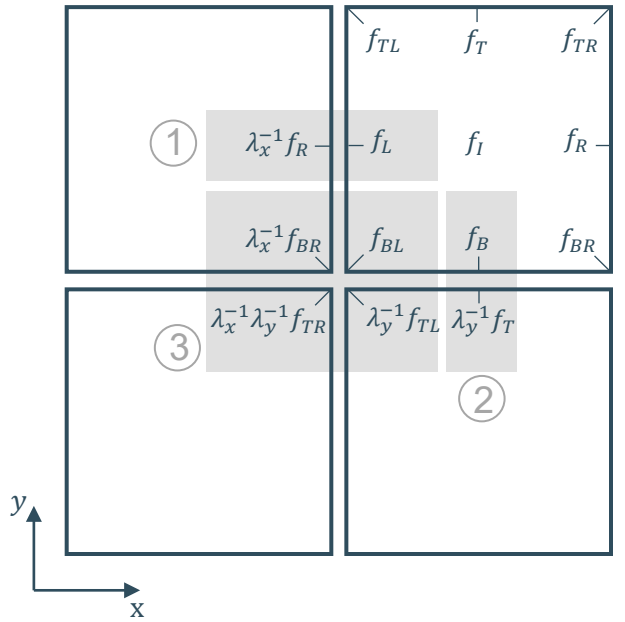
Force balance

$$\mathbf{0} = \lambda_x^{-1}\mathbf{f}_R + \mathbf{f}_L$$
$$\mathbf{0} = \lambda_y^{-1}\mathbf{f}_T + \mathbf{f}_B$$
$$\mathbf{0} = \lambda_x^{-1}\lambda_y^{-1}\mathbf{f}_{TR} + \lambda_y^{-1}\mathbf{f}_{TL} + \lambda_x^{-1}\mathbf{f}_{BR} + \mathbf{f}_{BL}$$



Force balance

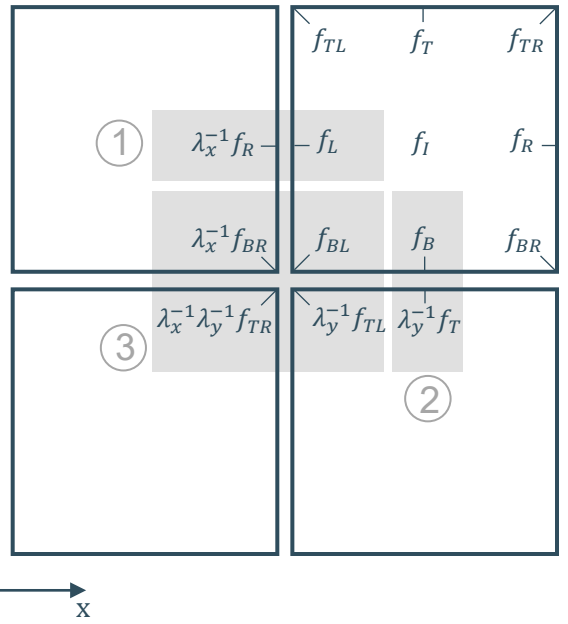
$$\mathbf{0} = \lambda_x^{-1}\mathbf{f}_R + \mathbf{f}_L$$
$$\mathbf{0} = \lambda_y^{-1}\mathbf{f}_T + \mathbf{f}_B$$
$$\mathbf{0} = \lambda_x^{-1}\lambda_y^{-1}\mathbf{f}_{TR} + \lambda_y^{-1}\mathbf{f}_{TL} + \lambda_x^{-1}\mathbf{f}_{BR} + \mathbf{f}_{BL}$$



Force balance

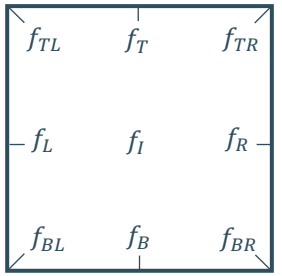
$$\mathbf{0} = \lambda_x^{-1}\mathbf{f}_R + \mathbf{f}_L$$
$$\mathbf{0} = \lambda_y^{-1}\mathbf{f}_T + \mathbf{f}_B$$
$$\mathbf{0} = \lambda_x^{-1}\lambda_y^{-1}\mathbf{f}_{TR} + \lambda_y^{-1}\mathbf{f}_{TL} + \lambda_x^{-1}\mathbf{f}_{BR} + \mathbf{f}_{BL}$$

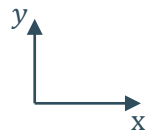
$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_y^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \lambda_y^{-1}\mathbf{I} & \lambda_x^{-1}\lambda_y^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_I \\ \mathbf{f}_L \\ \mathbf{f}_R \\ \mathbf{f}_B \\ \mathbf{f}_T \\ \mathbf{f}_{BL} \\ \mathbf{f}_{BR} \\ \mathbf{f}_{TL} \\ \mathbf{f}_{TR} \end{bmatrix} = \mathbf{R}'\mathbf{f} = \begin{bmatrix} \mathbf{f}_I \\ \mathbf{0} \end{bmatrix}.$$



Force balance

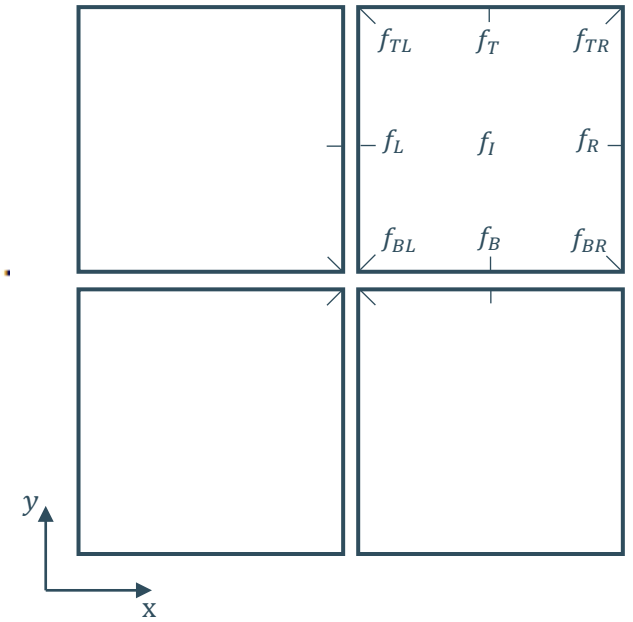
$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_y^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \lambda_y^{-1}\mathbf{I} & \lambda_x^{-1}\lambda_y^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_I \\ \mathbf{f}_L \\ \mathbf{f}_R \\ \mathbf{f}_B \\ \mathbf{f}_T \\ \mathbf{f}_{BL} \\ \mathbf{f}_{BR} \\ \mathbf{f}_{TL} \\ \mathbf{f}_{TR} \end{bmatrix} = \mathbf{R}'\mathbf{f} = \begin{bmatrix} \mathbf{f}_I \\ \mathbf{0} \end{bmatrix}.$$





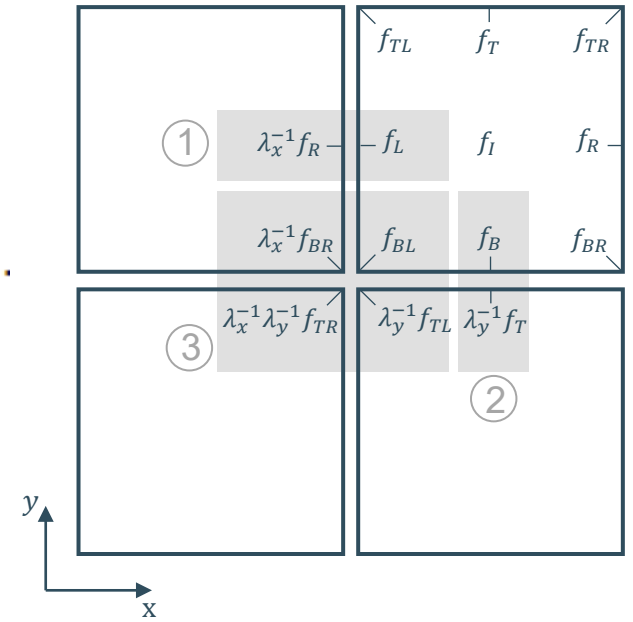
Force balance

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_y^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \lambda_y^{-1}\mathbf{I} & \lambda_x^{-1}\lambda_y^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_I \\ \mathbf{f}_L \\ \mathbf{f}_R \\ \mathbf{f}_B \\ \mathbf{f}_T \\ \mathbf{f}_{BL} \\ \mathbf{f}_{BR} \\ \mathbf{f}_{TL} \\ \mathbf{f}_{TR} \end{bmatrix} = \mathbf{R}'\mathbf{f} = \begin{bmatrix} \mathbf{f}_I \\ \mathbf{0} \end{bmatrix}.$$



Force balance

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_y^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \lambda_x^{-1}\mathbf{I} & \lambda_y^{-1}\mathbf{I} & \lambda_x^{-1}\lambda_y^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}_I \\ \mathbf{f}_L \\ \mathbf{f}_R \\ \mathbf{f}_B \\ \mathbf{f}_T \\ \mathbf{f}_{BL} \\ \mathbf{f}_{BR} \\ \mathbf{f}_{TL} \\ \mathbf{f}_{TR} \end{bmatrix} = \mathbf{R}'\mathbf{f} = \begin{bmatrix} \mathbf{f}_I \\ \mathbf{0} \end{bmatrix}.$$



Dispersion EVP

- UC equations of motion

$$\mathbf{Dq} = \mathbf{f}$$

$$\rightarrow \mathbf{R'DR}\begin{Bmatrix}\mathbf{q}_{\mathrm{BL}}\\ \mathbf{q}_{\mathrm{L}}\\ \mathbf{q}_{\mathrm{B}}\\ \mathbf{q}_{\mathrm{I}}\end{Bmatrix} = \mathbf{R'}(-\boldsymbol{\omega}^{\mathbf{2}}\mathbf{M} + \mathbf{K})\mathbf{R}\begin{Bmatrix}\mathbf{q}_{BL}\\ \mathbf{q}_{L}\\ \mathbf{q}_{B}\\ \mathbf{q}_{I}\end{Bmatrix} = \mathbf{R'f} = \mathbf{0}$$

$$\rightarrow \mathbf{A}\left(\omega, \mu_x, \mu_y\right) = \mathbf{0}$$

Dispersion EVP

- UC equations of motion + Bloch-Floquet BCs

$$\mathbf{Dq} = \mathbf{f}$$

$$\rightarrow \mathbf{R}'\mathbf{D}\mathbf{R}\begin{Bmatrix}\mathbf{q}_{\mathrm{BL}}\\ \mathbf{q}_{\mathrm{L}}\\ \mathbf{q}_{\mathrm{B}}\\ \mathbf{q}_{\mathrm{I}}\end{Bmatrix} = \mathbf{R}'(-\boldsymbol{\omega}^2\mathbf{M} + \mathbf{K})\mathbf{R}\begin{Bmatrix}\mathbf{q}_{BL}\\ \mathbf{q}_{L}\\ \mathbf{q}_{B}\\ \mathbf{q}_{I}\end{Bmatrix} = \mathbf{R}'\mathbf{f} = \mathbf{0}$$

$$\rightarrow \mathbf{A}\left(\omega, \mu_x, \mu_y\right) = \mathbf{0}$$

Dispersion EVP

- UC equations of motion + Bloch-Floquet BCs

$$\mathbf{Dq} = \mathbf{f}$$

$$\rightarrow \mathbf{R}'\mathbf{D}\mathbf{R}\begin{Bmatrix}\mathbf{q}_{\mathrm{BL}}\\ \mathbf{q}_{\mathrm{L}}\\ \mathbf{q}_{\mathrm{B}}\\ \mathbf{q}_{\mathrm{I}}\end{Bmatrix} = \mathbf{R}'(-\boldsymbol{\omega}^2\mathbf{M} + \mathbf{K})\mathbf{R}\begin{Bmatrix}\mathbf{q}_{BL}\\ \mathbf{q}_{L}\\ \mathbf{q}_{B}\\ \mathbf{q}_{I}\end{Bmatrix} = \mathbf{R}'\mathbf{f} = \mathbf{0}$$

$$\boxed{\rightarrow \mathbf{A}\left(\omega, \mu_x, \mu_y\right) = \mathbf{0}}$$

Solving the EVP

Solving the EVP

$$\mathbf{A}(\omega, \mu_x, \mu_y)\tilde{\mathbf{q}} = \mathbf{0}$$

- Inverse approach: $\omega(\mu)$

  Real μ, free wave propagation

- Direct approach: $\mu(\omega)$

  Real ω, harmonic wave propagation

- Mixed approach: $\mu_y(\omega, \mu_x)$, $\mu(\omega, \alpha)$, $\omega(\mu(\omega))$

Solving the EVP

$$\mathbf{A}(\omega, \mu_x, \mu_y)\tilde{\mathbf{q}} = \mathbf{0}$$

- Inverse approach: $\omega(\mu)$

  Real μ, free wave propagation

- Direct approach: $\mu(\omega)$

  Real ω, harmonic wave propagation

- Mixed approach: $\mu_y(\omega, \mu_x)$, $\mu(\omega, \alpha)$, $\omega(\mu(\omega))$

Solving the EVP

$$\mathbf{A}(\omega, \mu_x, \mu_y)\tilde{\mathbf{q}} = \mathbf{0}$$

- Inverse approach: $\omega(\mu)$

  Real μ, free wave propagation

- Direct approach: $\mu(\omega)$

  Real ω, harmonic wave propagation

- Mixed approach: $\mu_y(\omega, \mu_x)$, $\mu(\omega, \alpha)$, $\omega(\mu(\omega))$

Solving the EVP

$$\mathbf{A}(\omega, \mu_x, \mu_y)\tilde{\mathbf{q}} = \mathbf{0}$$

- Inverse approach: $\omega(\mu)$

  Real μ, free wave propagation

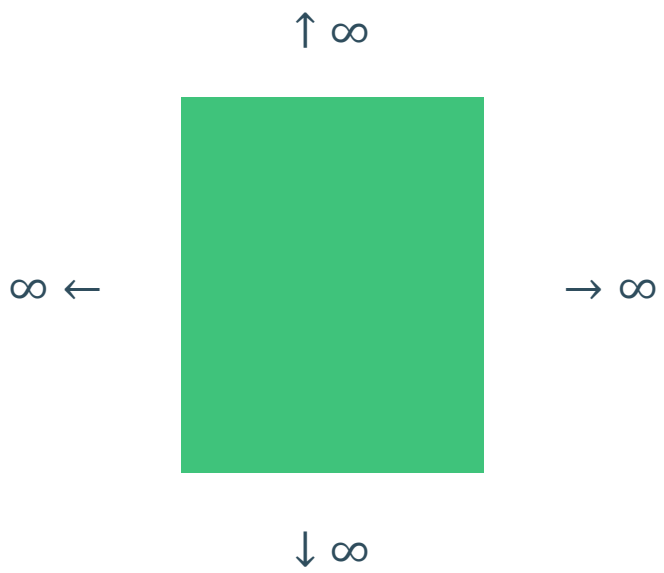
- Direct approach: $\mu(\omega)$

  Real ω, harmonic wave propagation

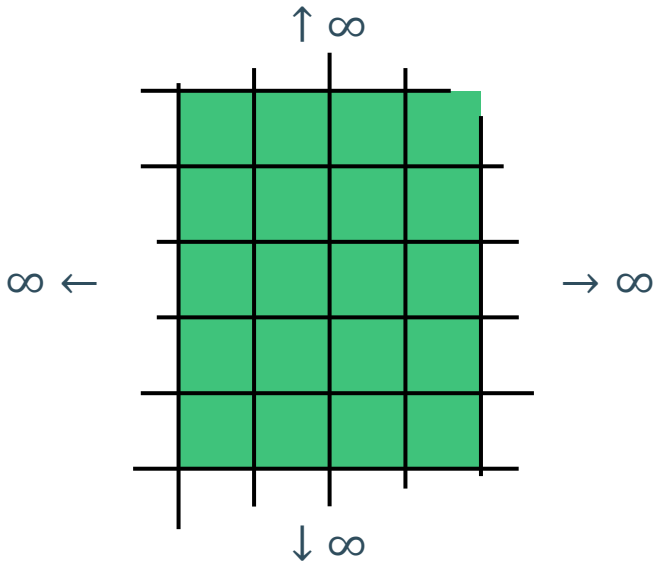
- Mixed approach: $\mu_y(\omega, \mu_x)$, $\mu(\omega, \alpha)$, $\omega(\mu(\omega))$

→ Wave propagation in the ∞ periodic structure

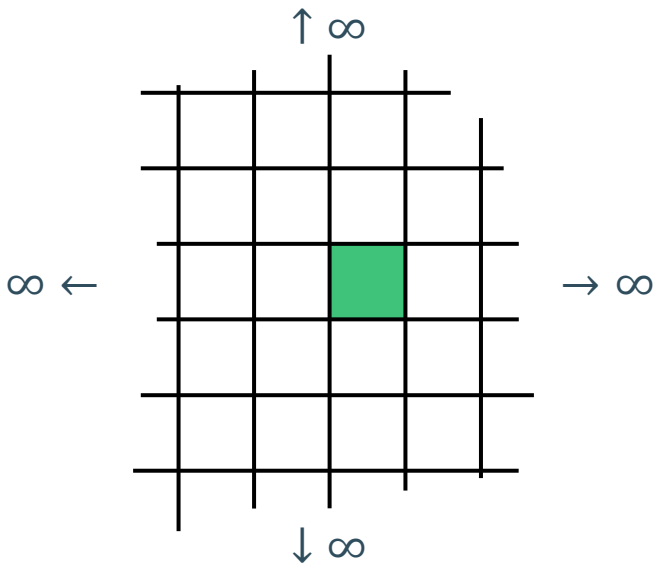
Example: isotropic steel plate



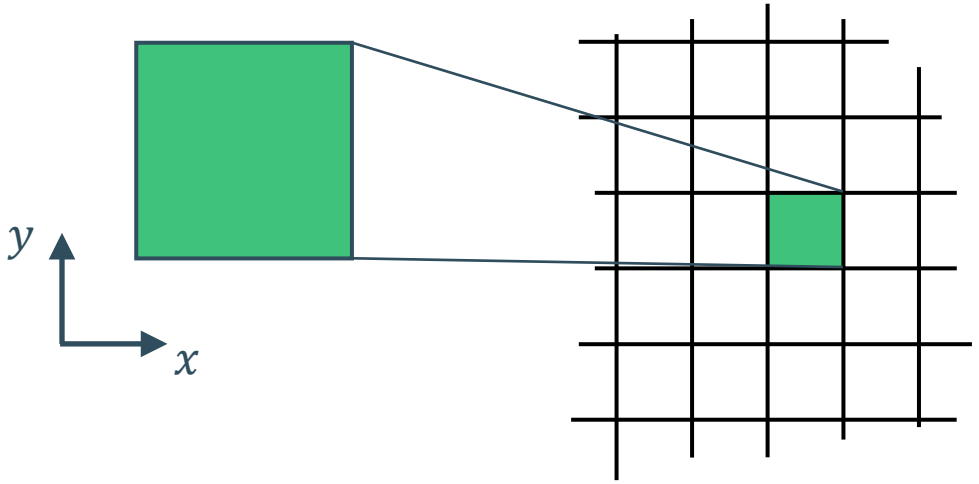
Example: isotropic steel plate



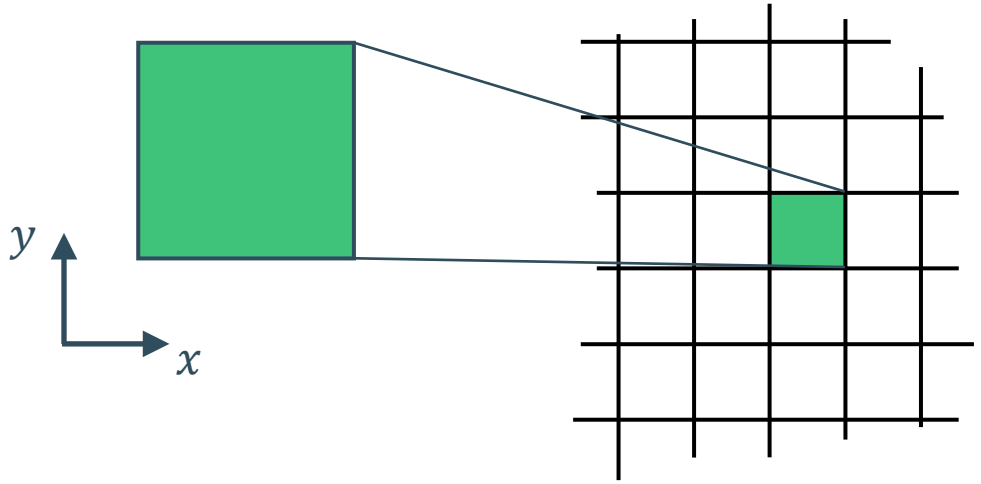
Example: isotropic steel plate



Example: isotropic steel plate



Example: isotropic steel plate



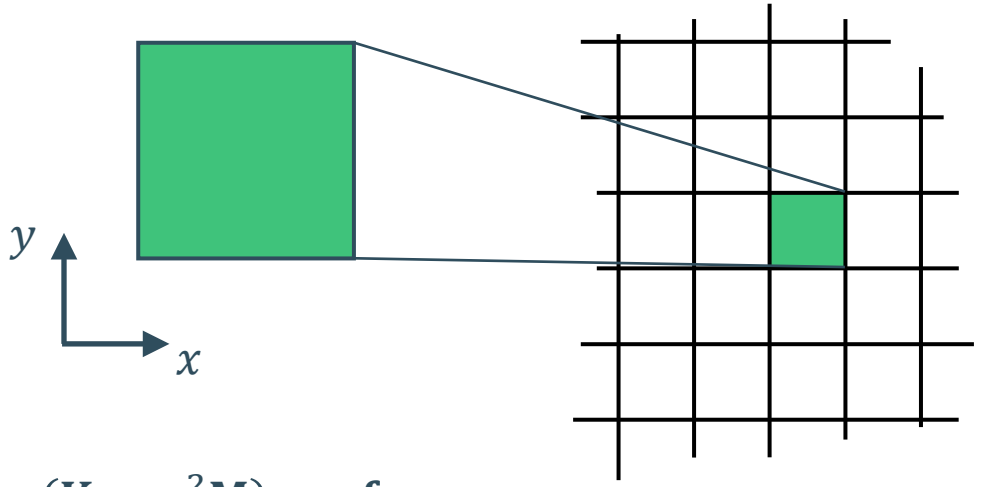
UC:

- $L_x = L_y = 0{,}05m$

Material properties

- $E = 210\text{GPa}$
- $\rho = 7800\,\frac{kg}{m^3}$
- $\nu = 0{,}3$

Example: isotropic steel plate



$$(\mathbf{K} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\downarrow$$

$$[\,\tilde{\mathbf{K}} - \omega^2\tilde{\mathbf{M}}]\tilde{\mathbf{q}} = \mathbf{0}$$

Solve EVP for real (μ_x, μ_y)

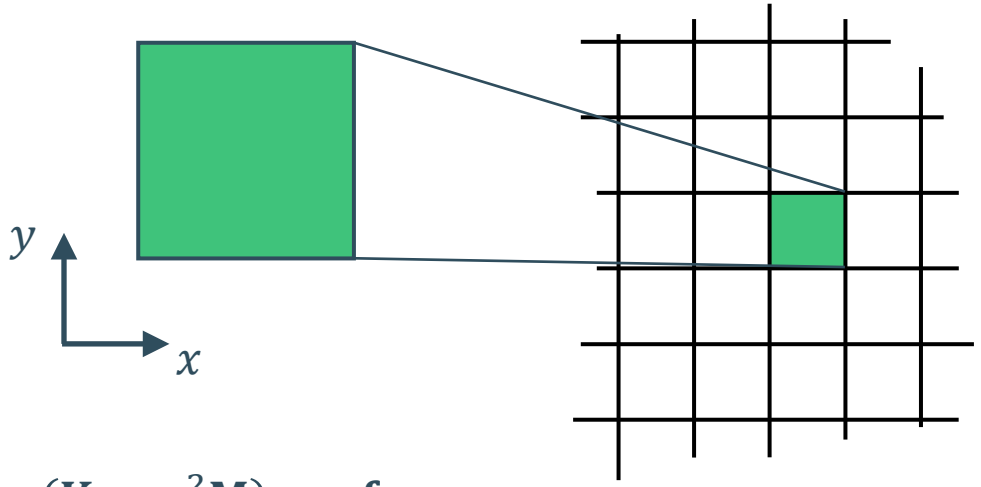
UC:

- $L_x = L_y = 0{,}05m$

Material properties

- $E = 210\text{GPa}$
- $\rho = 7800\,\frac{kg}{m^3}$
- $\nu = 0{,}3$

Example: isotropic steel plate



$$(\mathbf{K} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\downarrow$$

$$[\,\tilde{\mathbf{K}} - \omega^2\tilde{\mathbf{M}}]\tilde{\mathbf{q}} = \mathbf{0}$$

Solve EVP for real (μ_x, μ_y)

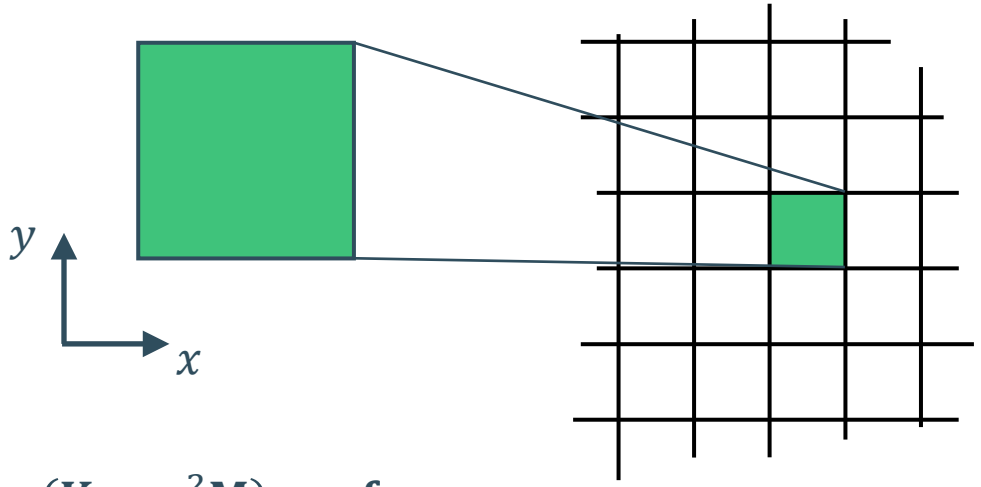
UC:

- $L_x = L_y = 0{,}05m$

Material properties

- $E = 210\text{GPa}$
- $\rho = 7800\,\frac{kg}{m^3}$
- $\nu = 0{,}3$

Example: isotropic steel plate



$$(\mathbf{K} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\downarrow$$

$$[\tilde{\mathbf{K}} - \omega^2\tilde{\mathbf{M}}]\tilde{\mathbf{q}} = \mathbf{0}$$

Solve EVP for real (μ_x, μ_y)

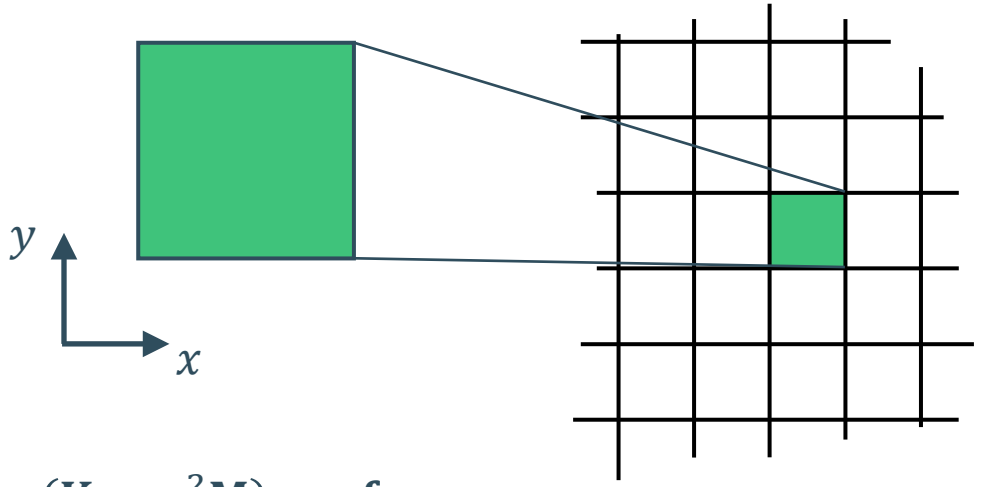
UC:

- $L_x = L_y = 0{,}05m$

Material properties

- $E = 210\text{GPa}$
- $\rho = 7800\,\frac{kg}{m^3}$
- $\nu = 0{,}3$

Example: isotropic steel plate



$$(\mathbf{K} - \omega^2\mathbf{M})\mathbf{q} = \mathbf{f}$$

$$\downarrow$$

$$[\,\tilde{\mathbf{K}} - \omega^2\tilde{\mathbf{M}}]\tilde{\mathbf{q}} = \mathbf{0}$$

Solve EVP for real (μ_x, μ_y)

UC:

- $L_x = L_y = 0{,}05m$

Material properties

- $E = 210\text{GPa}$
- $\rho = 7800\,\frac{kg}{m^3}$
- $\nu = 0{,}3$

Thin plate → Kirchhoff-Love theory
Analytical expression known!
Paraboloid in wave space:

$$k^4 = \frac{12\omega^2\rho(1-\nu\text{^}2)}{Eh^2}$$

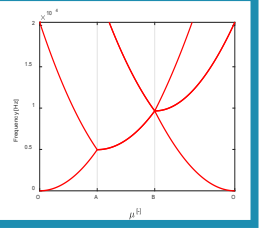
EVP to dispersion curves

Dispersion
eigenvalue problem

EVP to dispersion curves

Dispersion eigenvalue problem

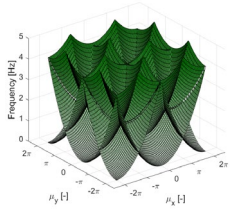
Dispersion curves



EVP to dispersion curves

Dispersion eigenvalue problem

Dispersion surfaces

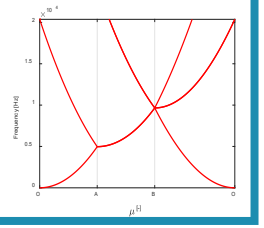


First Brillouin zone

Reduced Brillouin zone

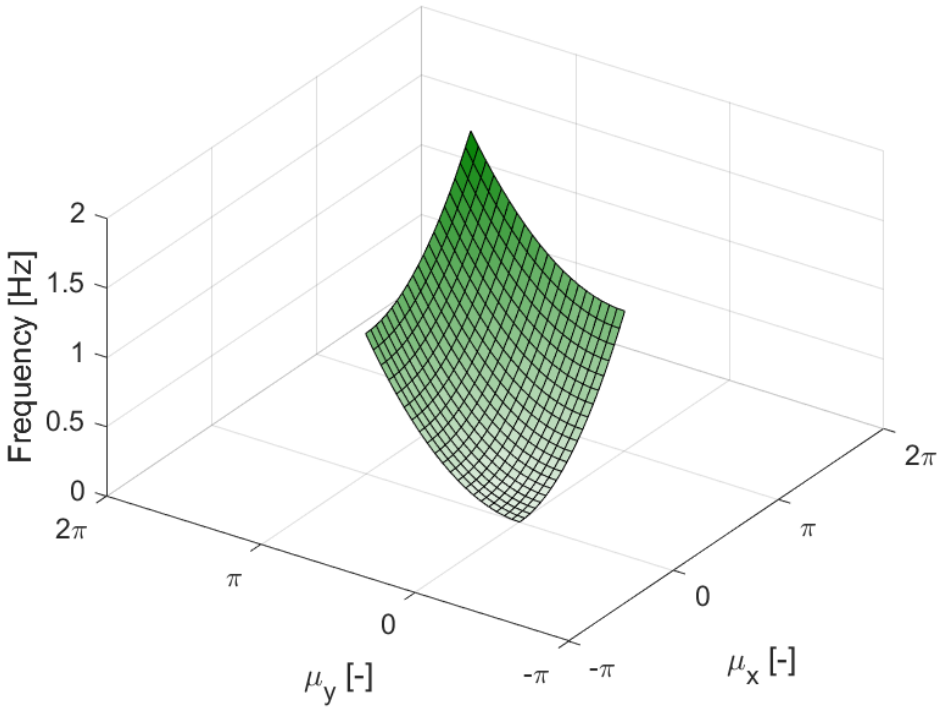
Irreducible Brillouin contour

Dispersion curves



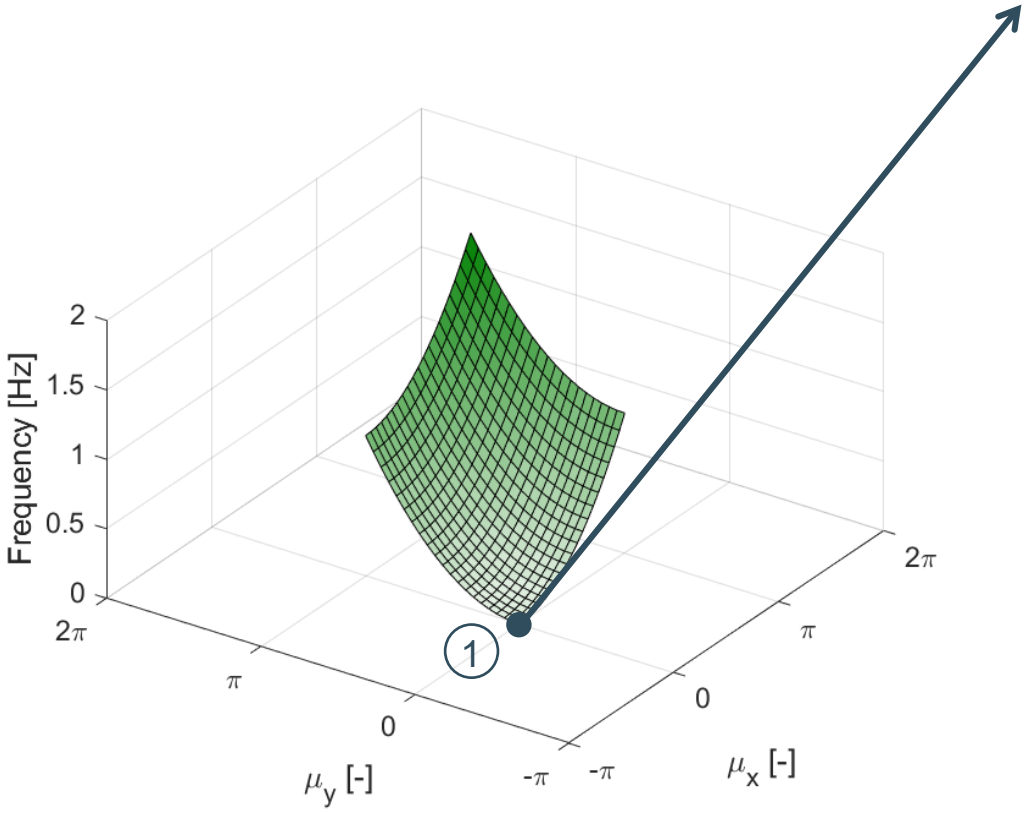
Dispersion surfaces

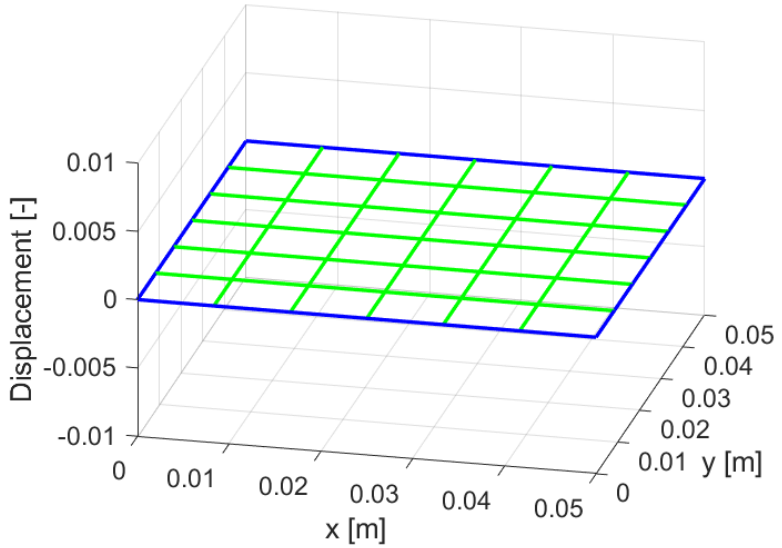
- First bending dispersion surface
- Free wave propagation



Dispersion surfaces

- First bending dispersion surface
- Free wave propagation

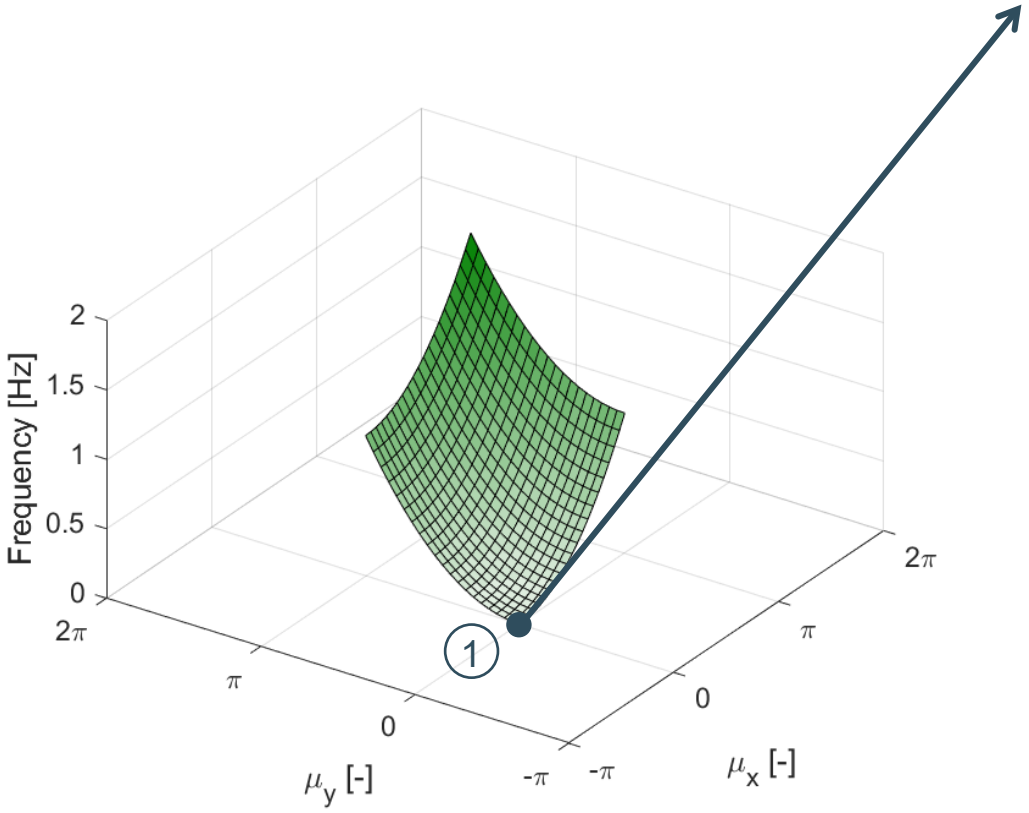


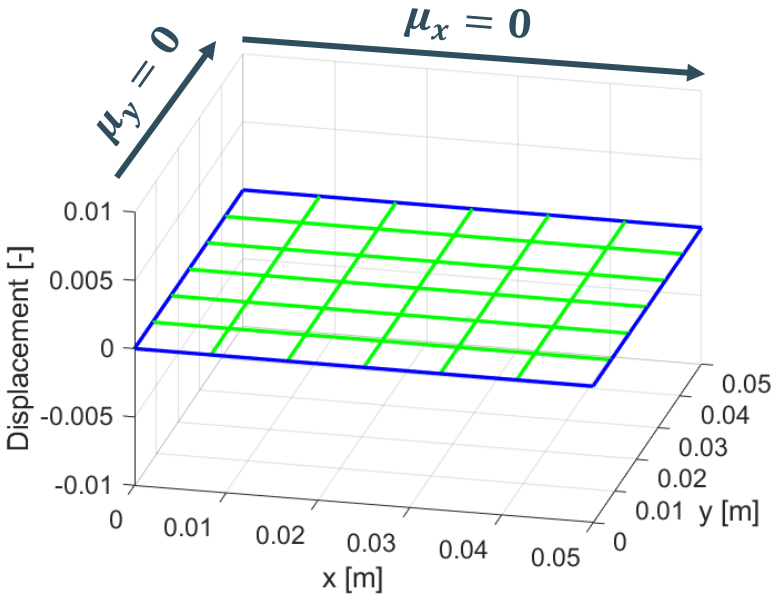


1 UC

Dispersion surfaces

- First bending dispersion surface
- Free wave propagation

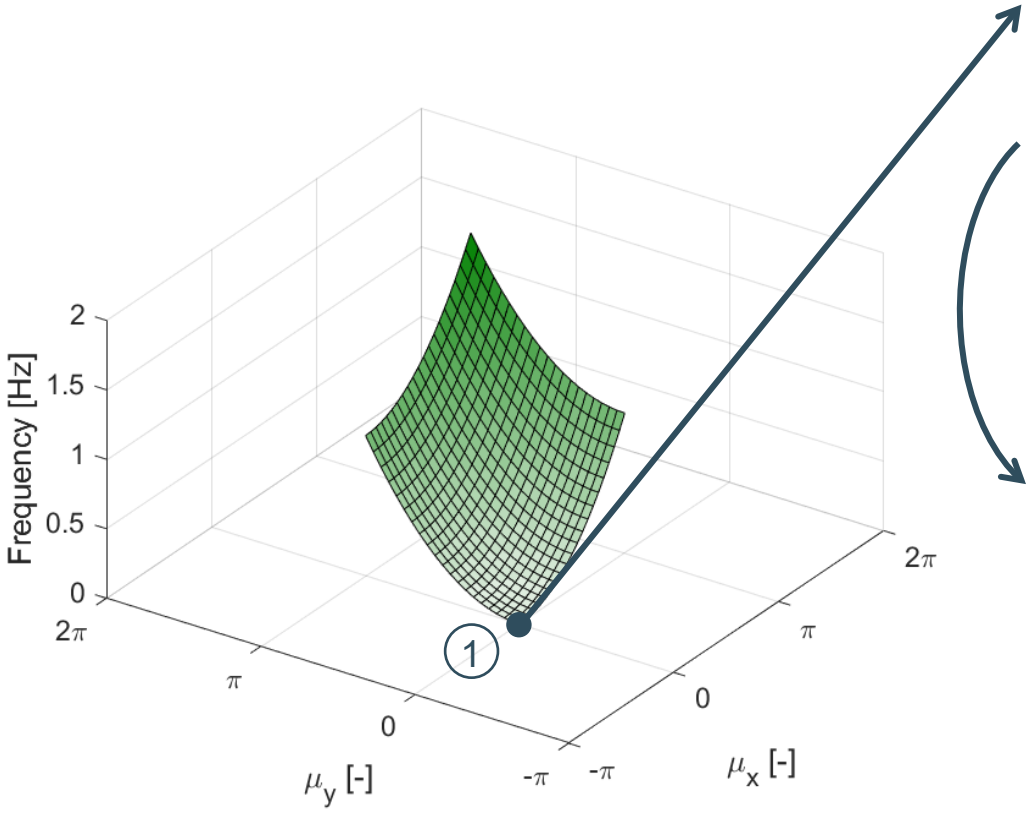


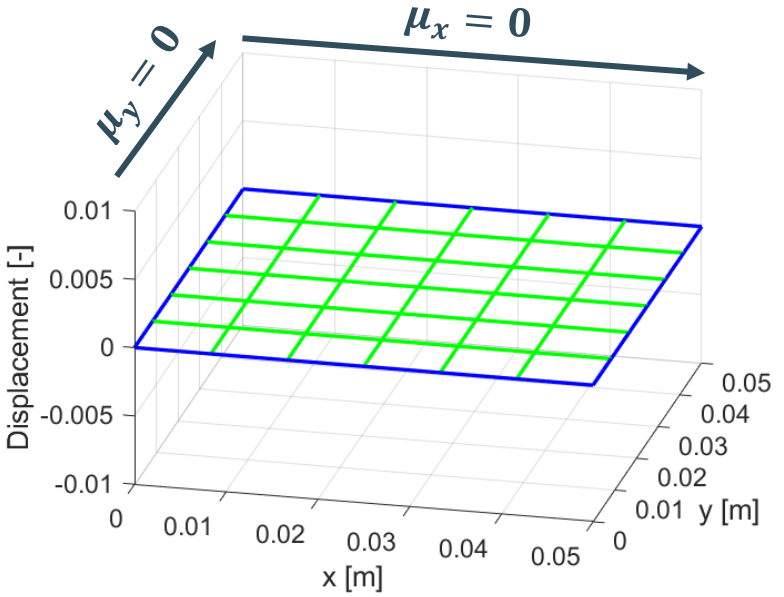


1 UC

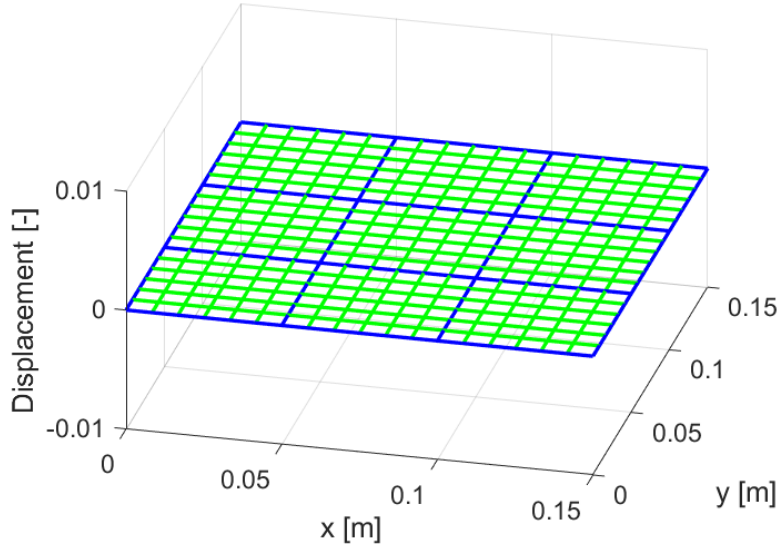
Dispersion surfaces

- First bending dispersion surface
- Free wave propagation





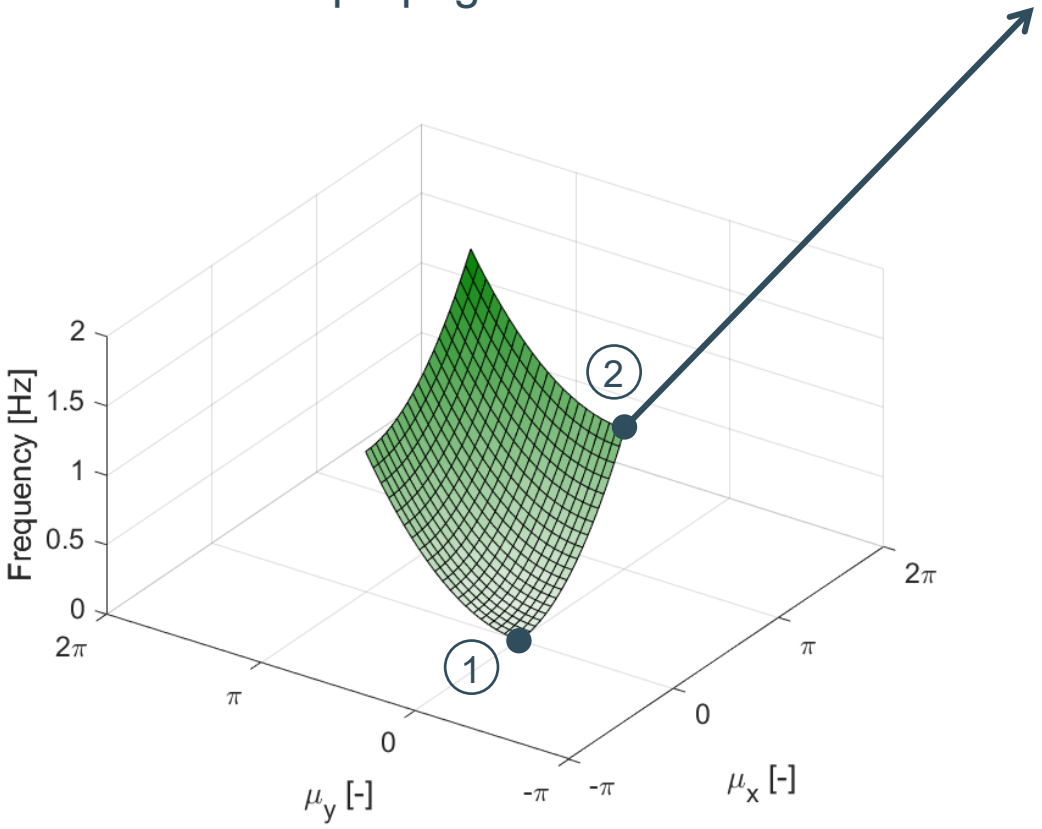
1 UC

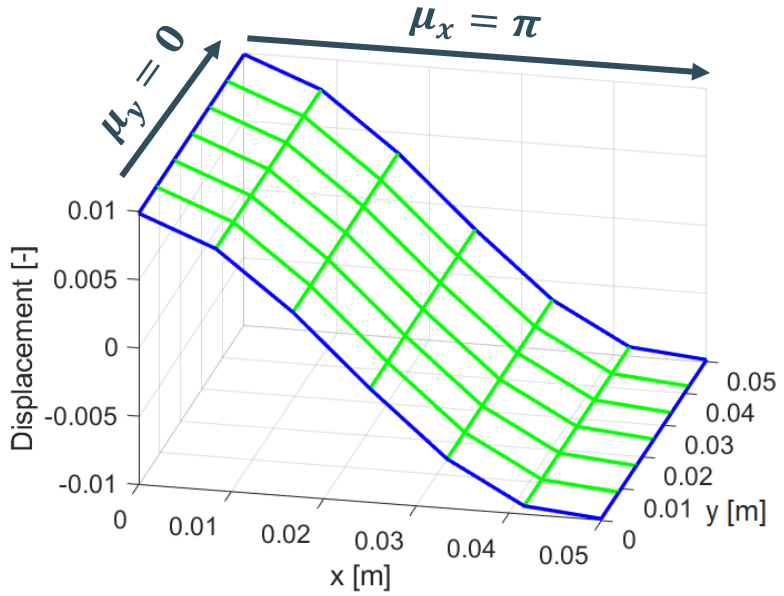


3x3 UCs

Dispersion surfaces

- First bending dispersion surface
- Free wave propagation

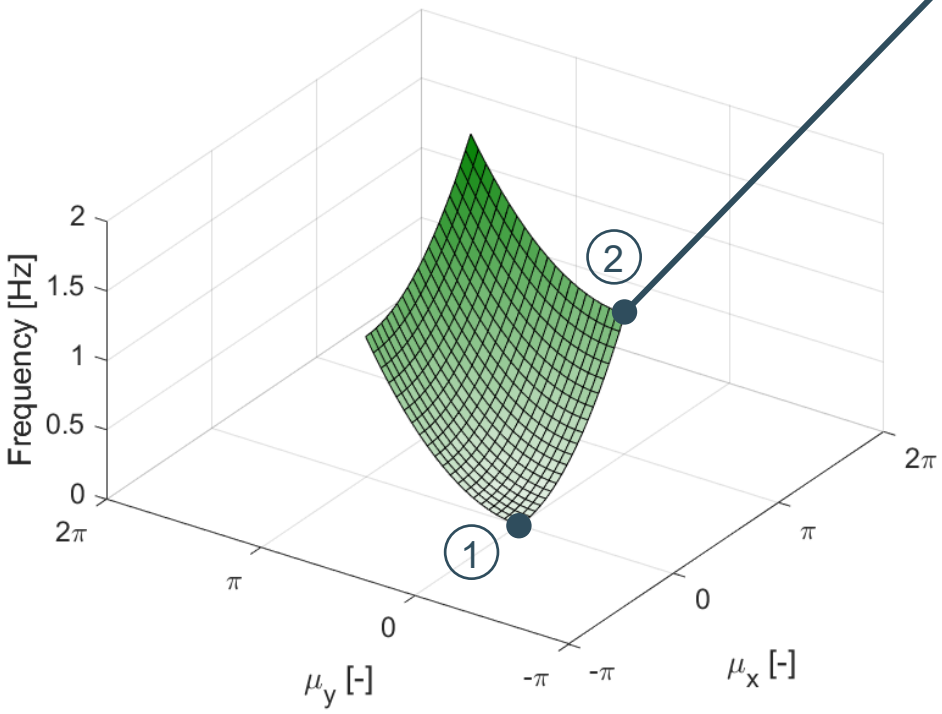


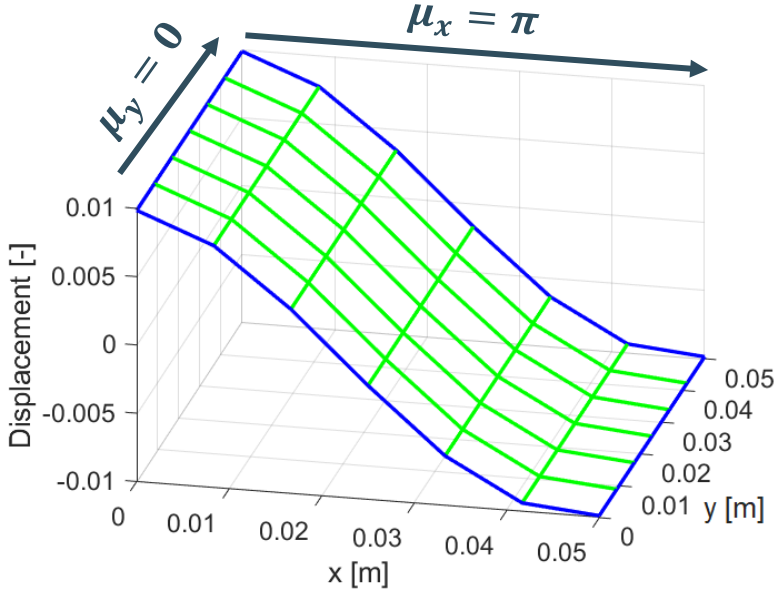


1 UC

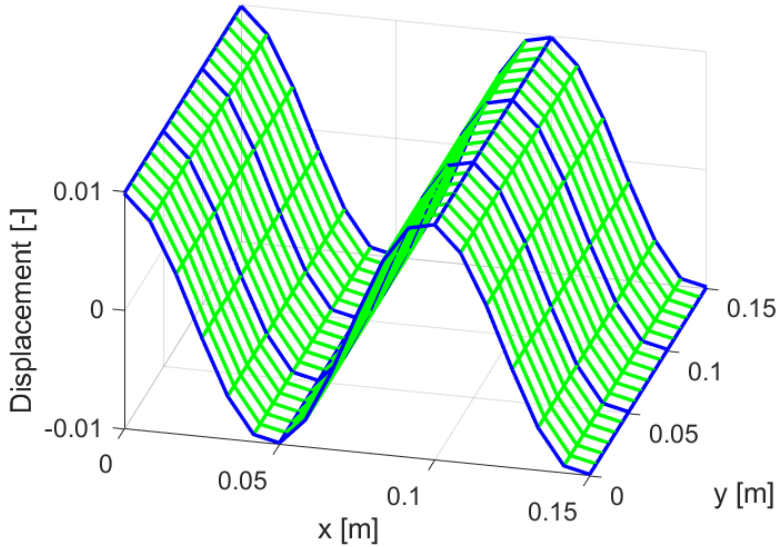
Dispersion surfaces

- First bending dispersion surface
- Free wave propagation





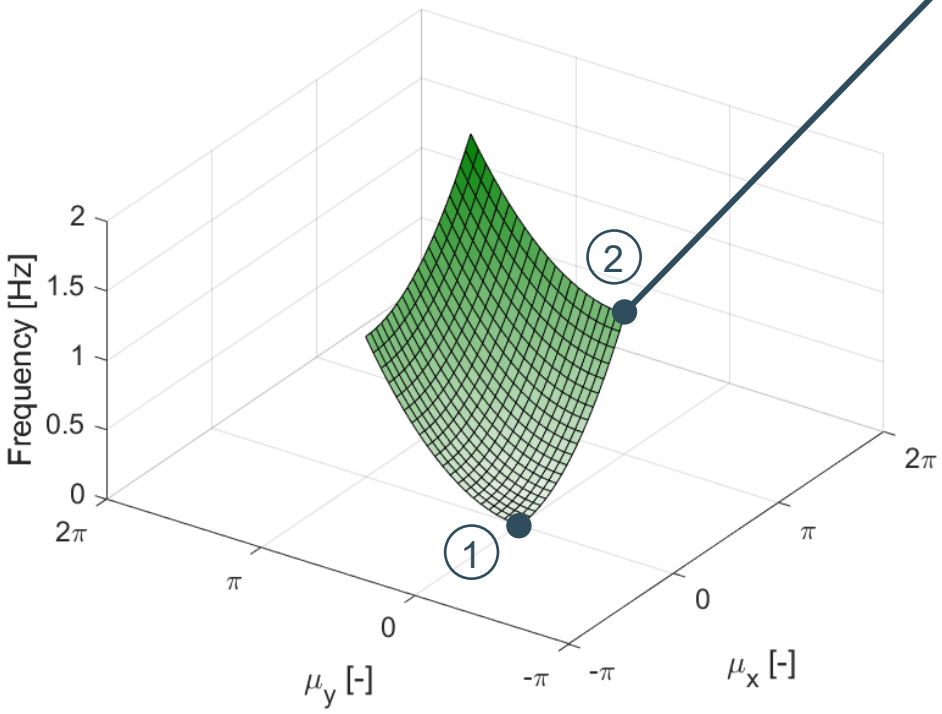
1 UC

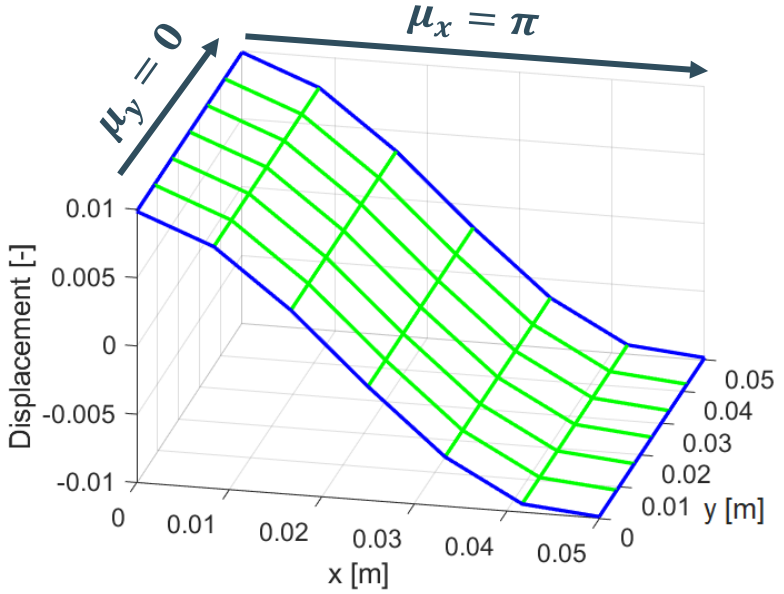


3x3 UCs

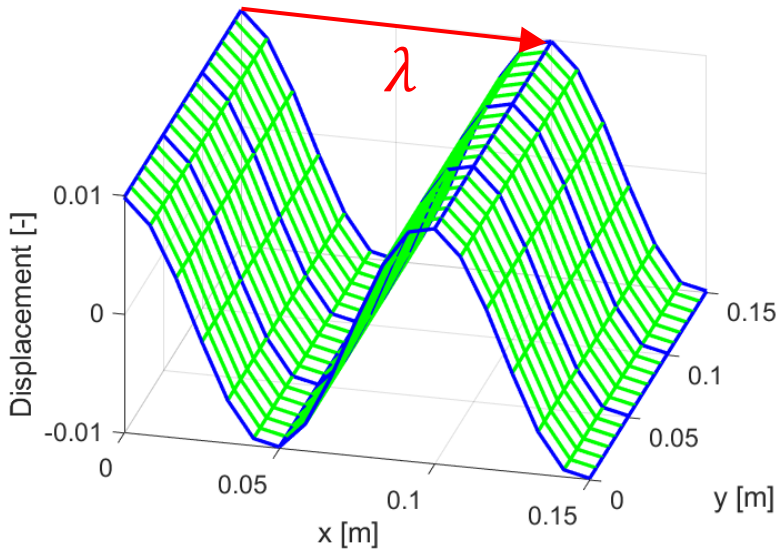
Dispersion surfaces

- First bending dispersion surface
- Free wave propagation





1 UC



3x3 UCs

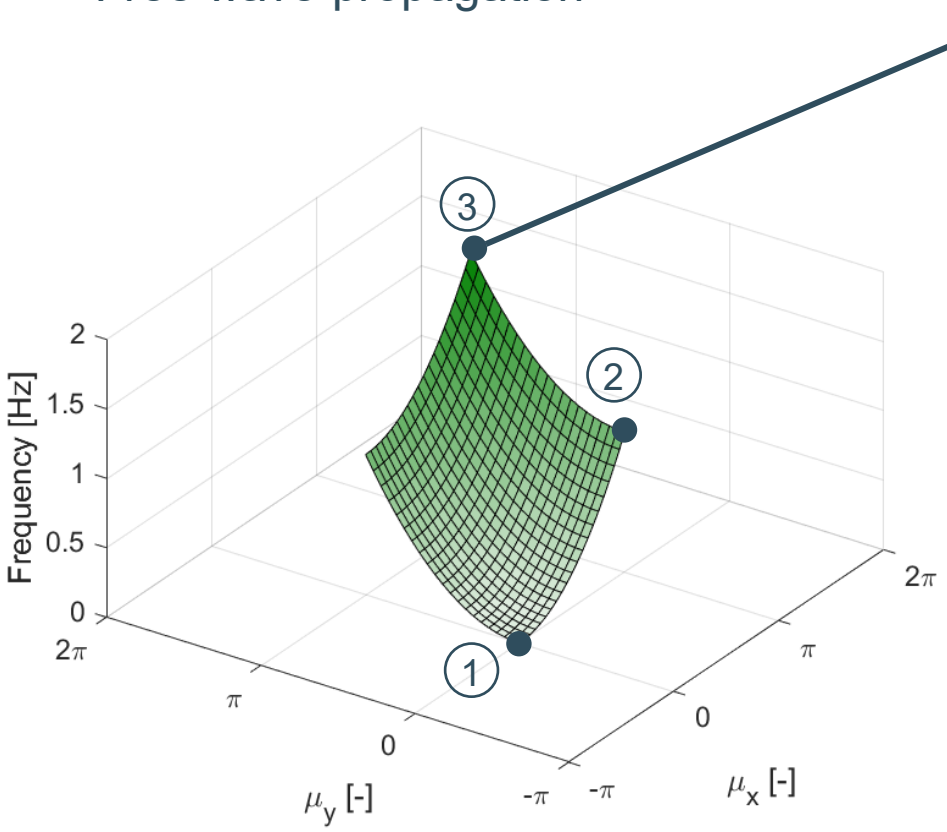
$$\mu_x = \pi \rightarrow \lambda = 2L_x$$

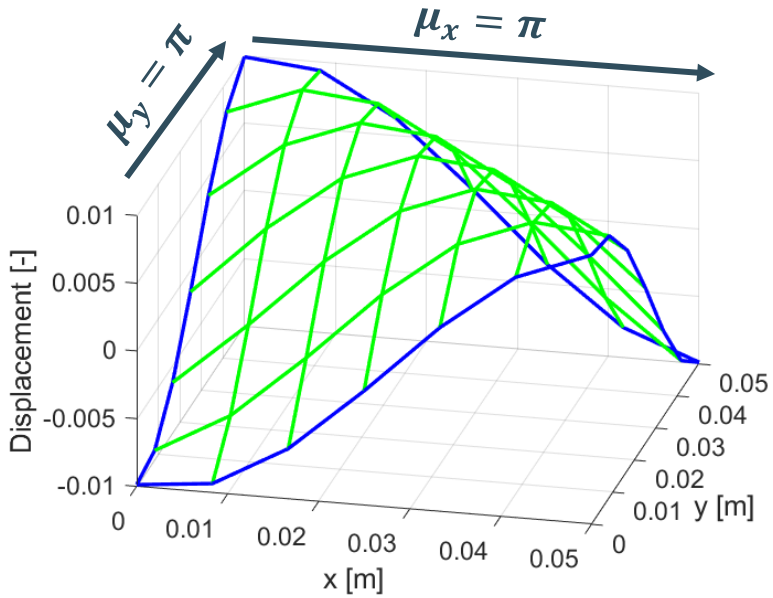
$$\omega(\pi/2, 0) = 2\pi f_{\lambda/2}$$

$$f_{\lambda/2} = \frac{\pi}{2L^2}\sqrt{\frac{t^2 E}{12(1-\nu^2)\rho}}$$

Dispersion surfaces

- First bending dispersion surface
- Free wave propagation

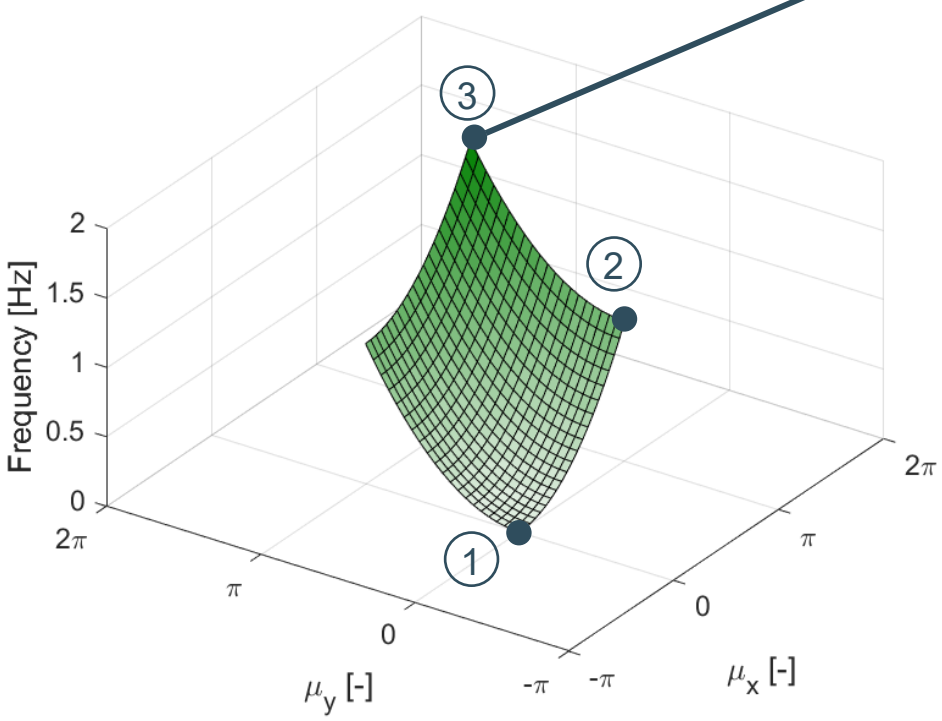


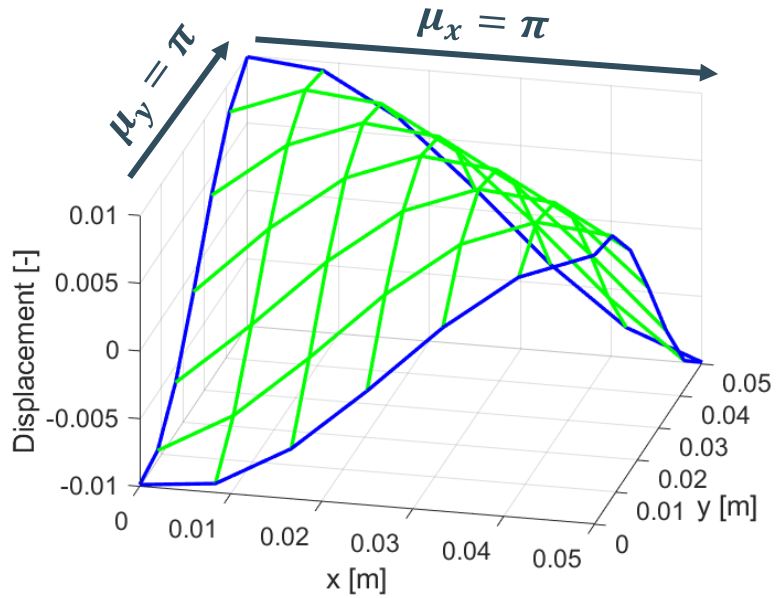


1 UC

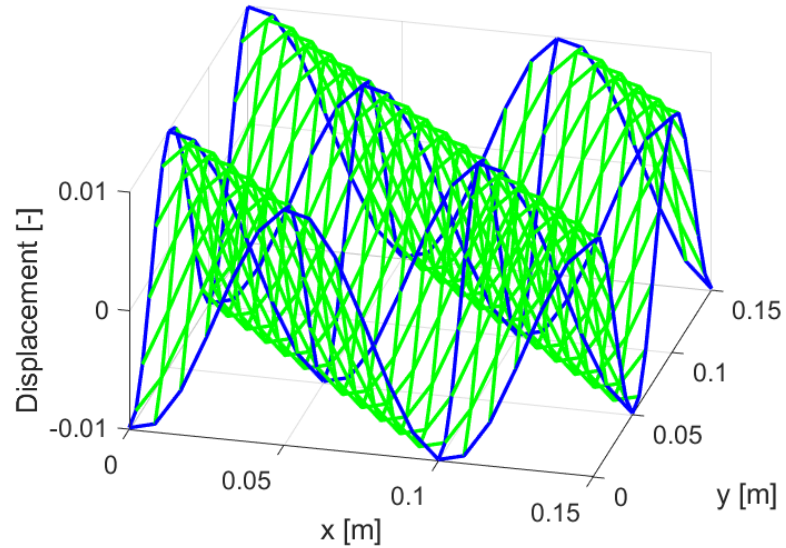
Dispersion surfaces

- First bending dispersion surface
- Free wave propagation





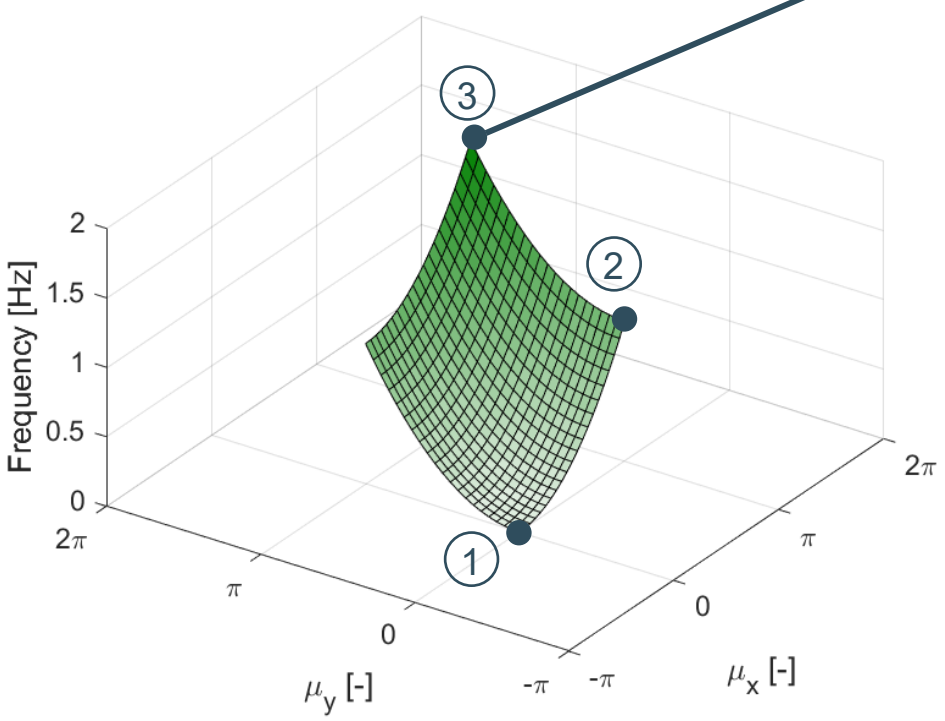
1 UC

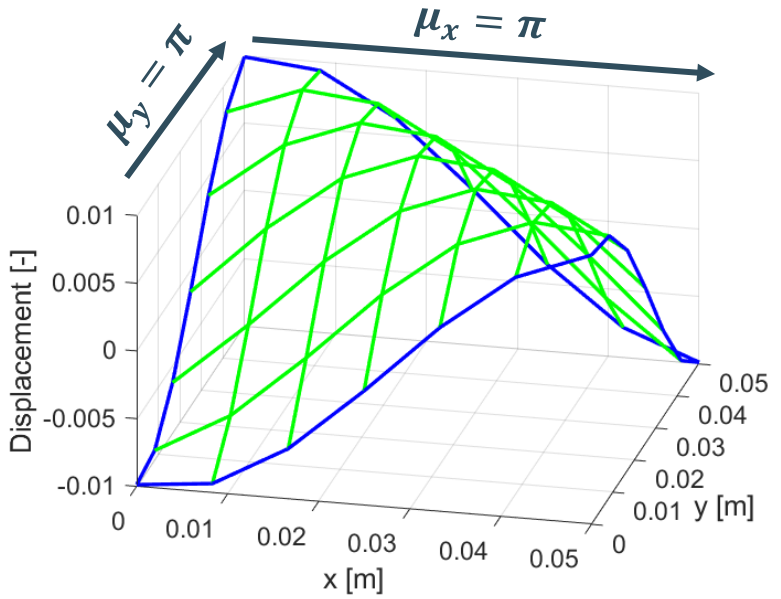


3x3 UCs

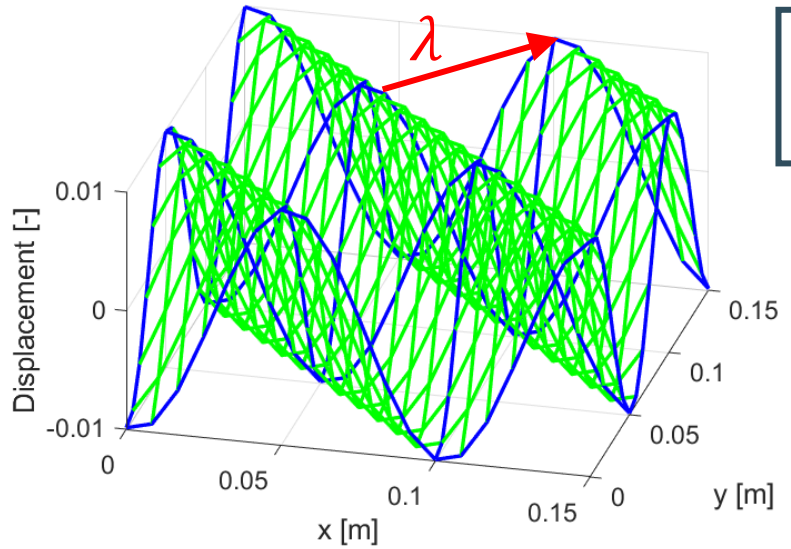
Dispersion surfaces

- First bending dispersion surface
- Free wave propagation





1 UC

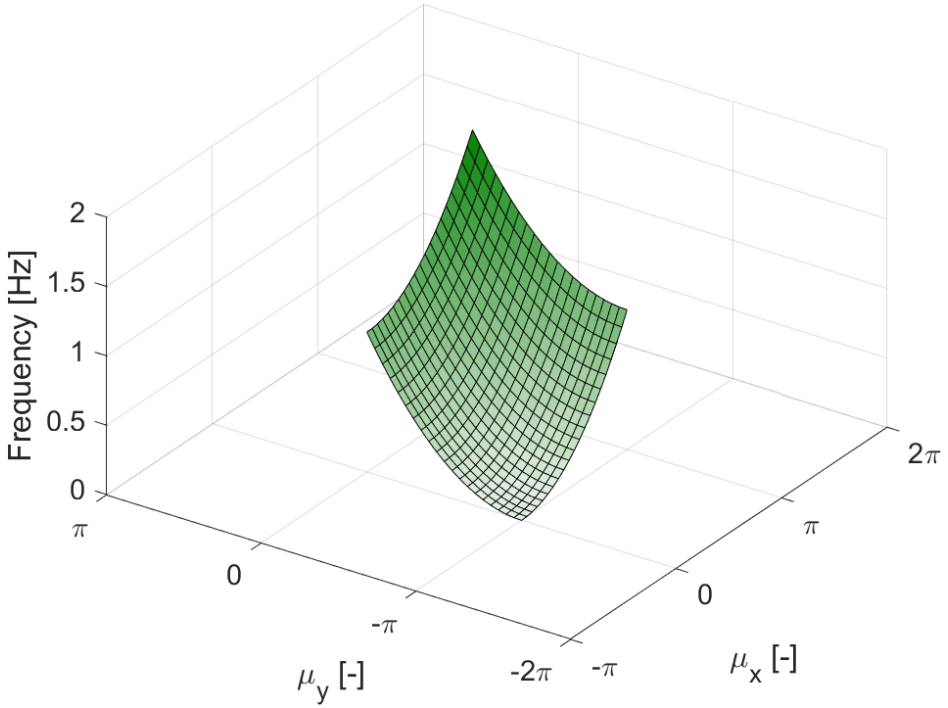


3x3 UCs

$$\mu_x + \mu_y = 2\pi \rightarrow \lambda = \sqrt{2}L_x$$
$$\omega(\pi,\pi) = 2\pi 2 f_{\lambda/2}$$

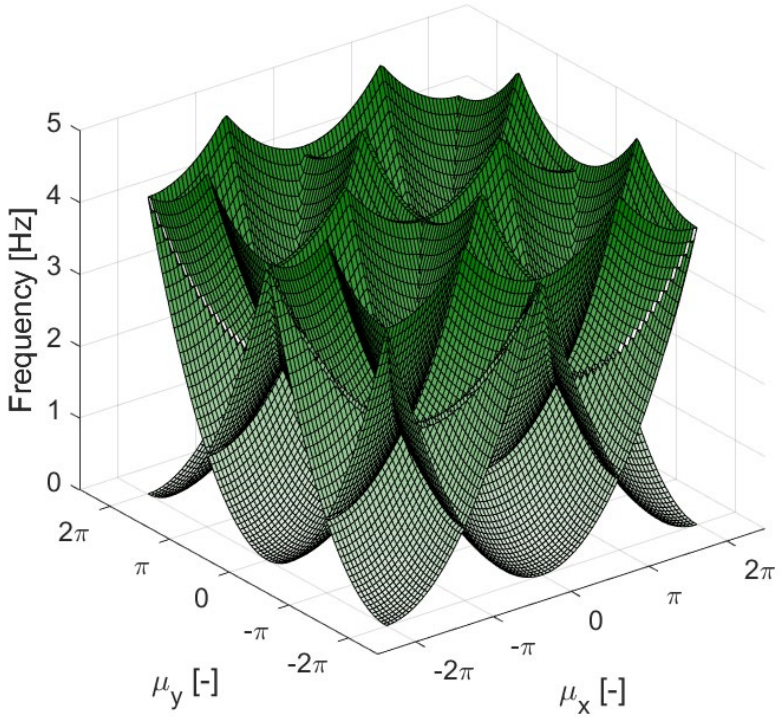
Periodicity $x, y \rightarrow$ periodicity μ_x, μ_y

First Brillouin zone



Periodicity $x, y \rightarrow$ periodicity μ_x, μ_y

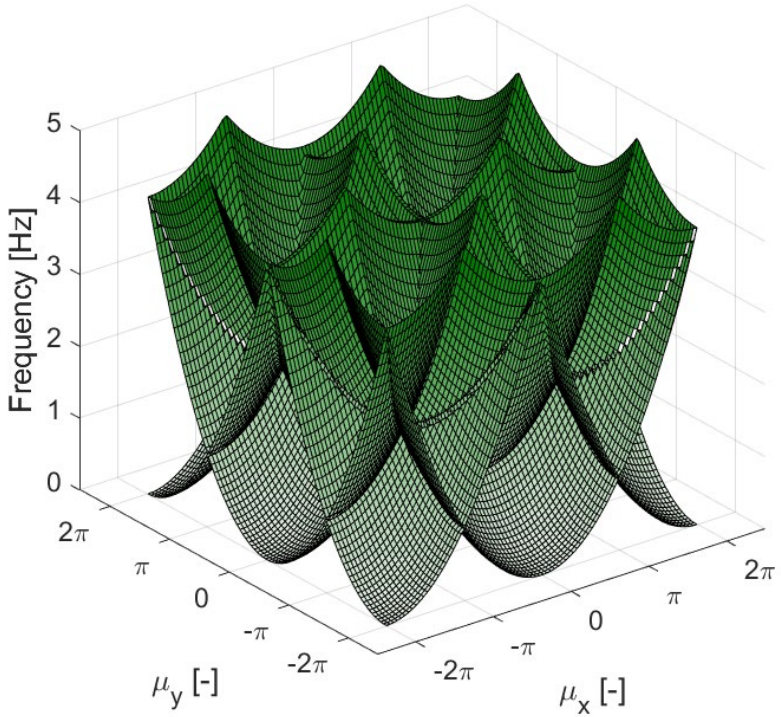
First Brillouin zone



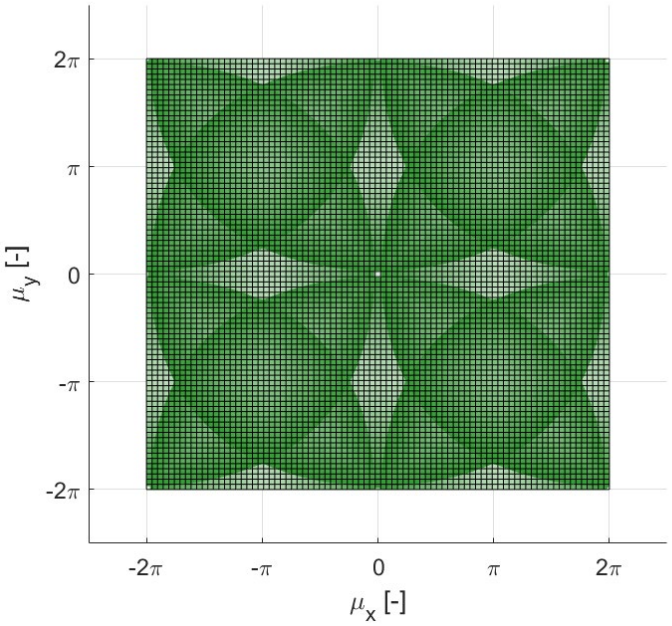
Dispersion surfaces

Periodicity $x, y \rightarrow$ periodicity μ_x, μ_y

First Brillouin zone



Dispersion surfaces



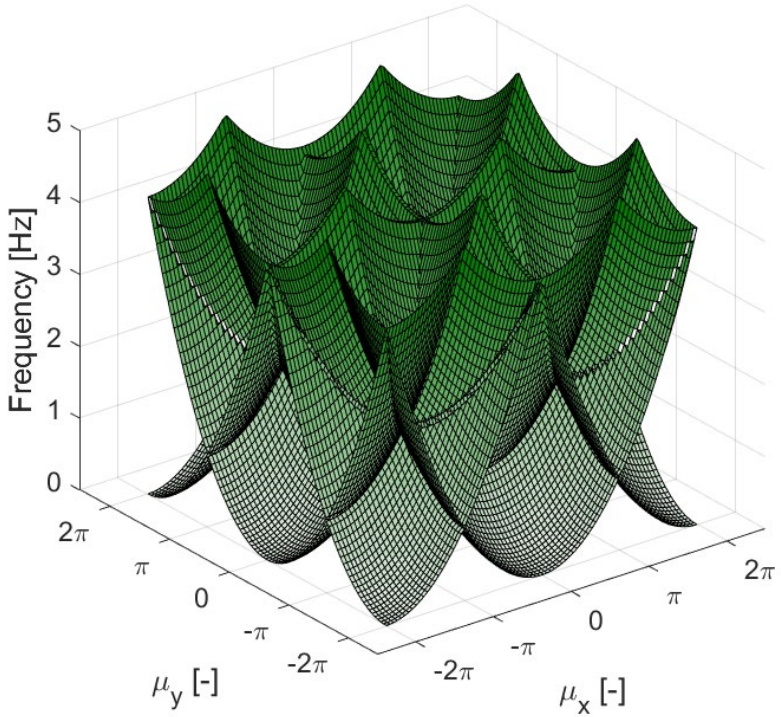
Top view

Periodicity $x, y \rightarrow$ periodicity μ_x, μ_y

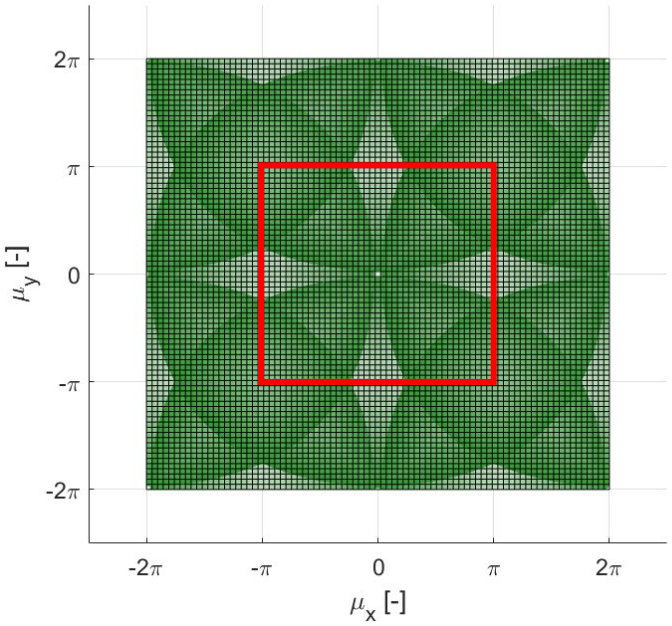
First Brillouin zone

Bloch's theorem $\rightarrow e^{i\mu}$

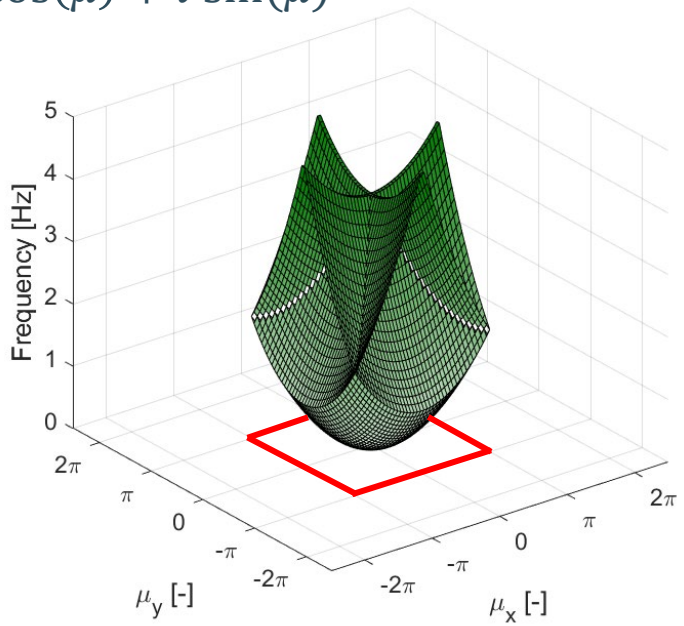
Euler's theorem $\rightarrow \cos(\mu) + i\,\sin(\mu)$



Dispersion surfaces

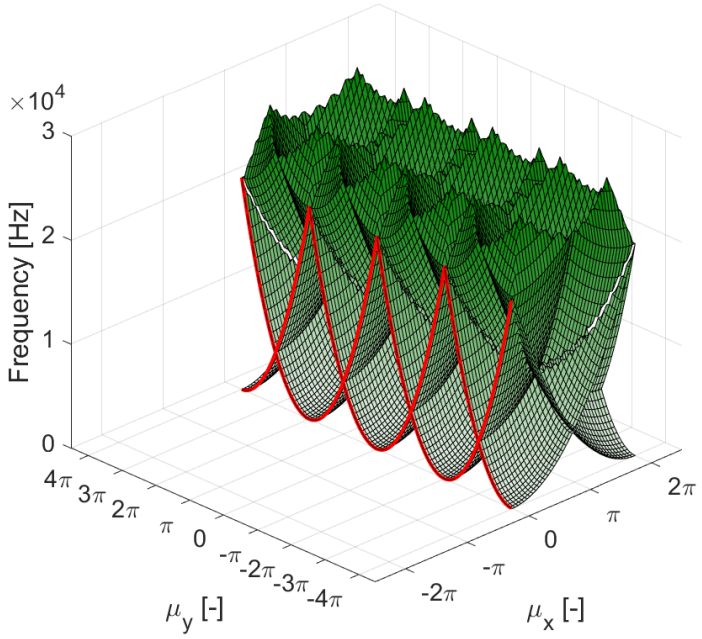


Top view

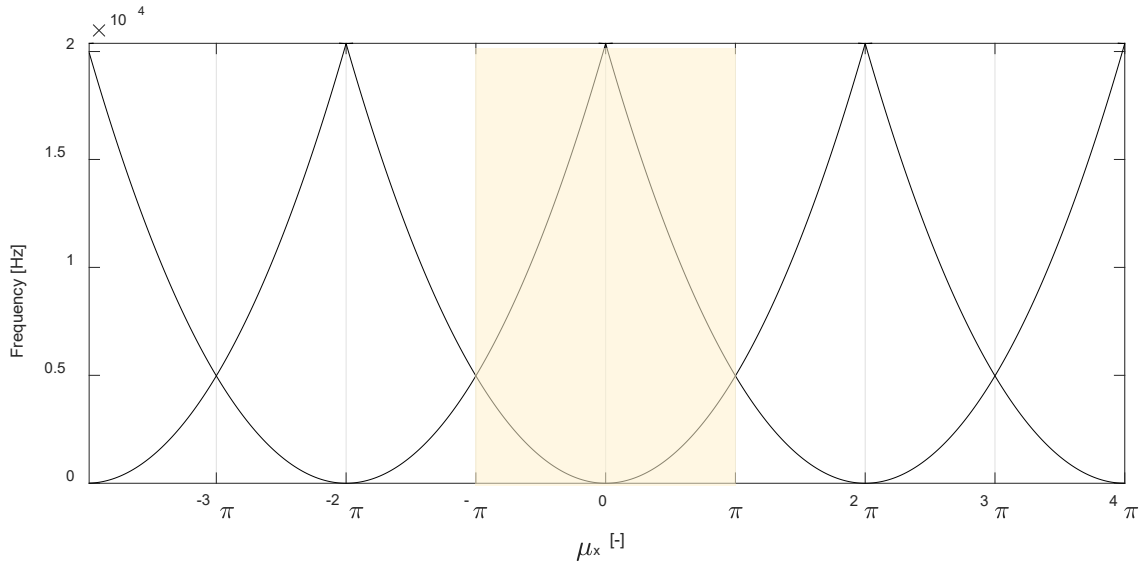


Dispersion surfaces
on first Brillouin zone

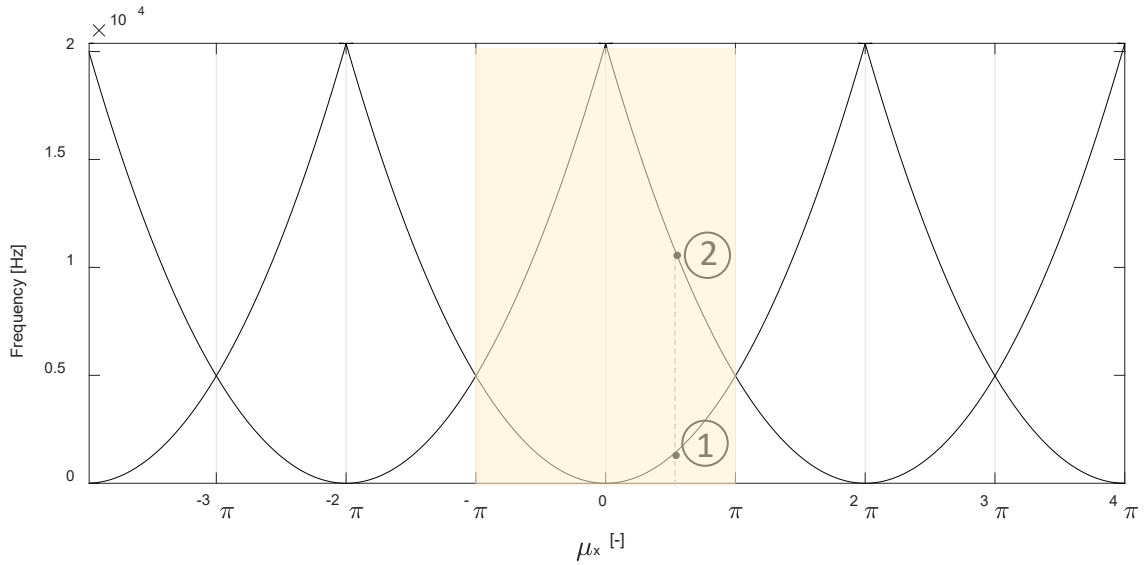
Periodicity wave domain



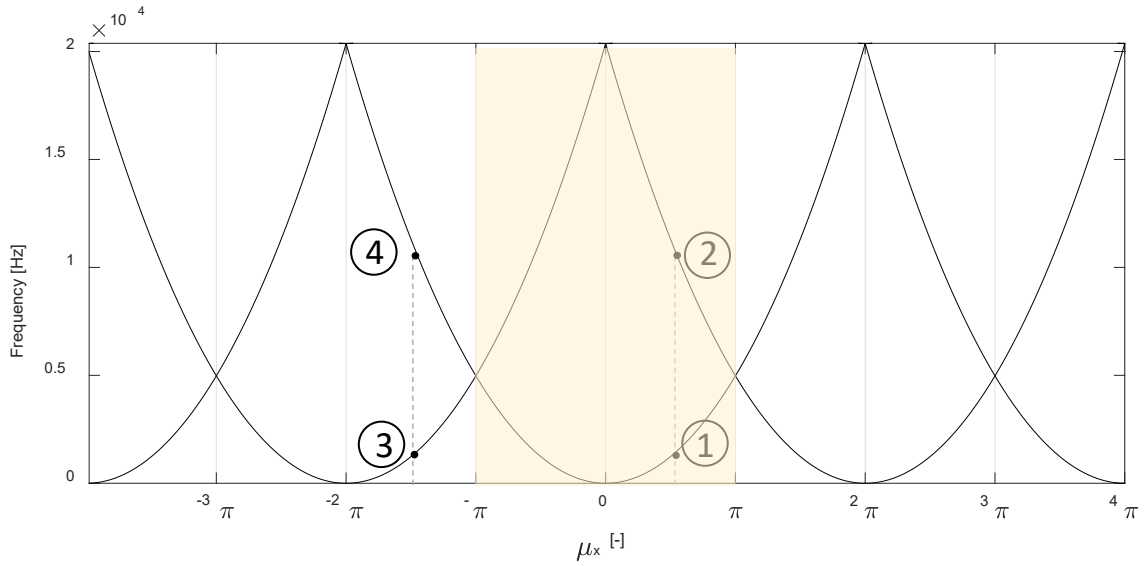
Periodicity wave domain



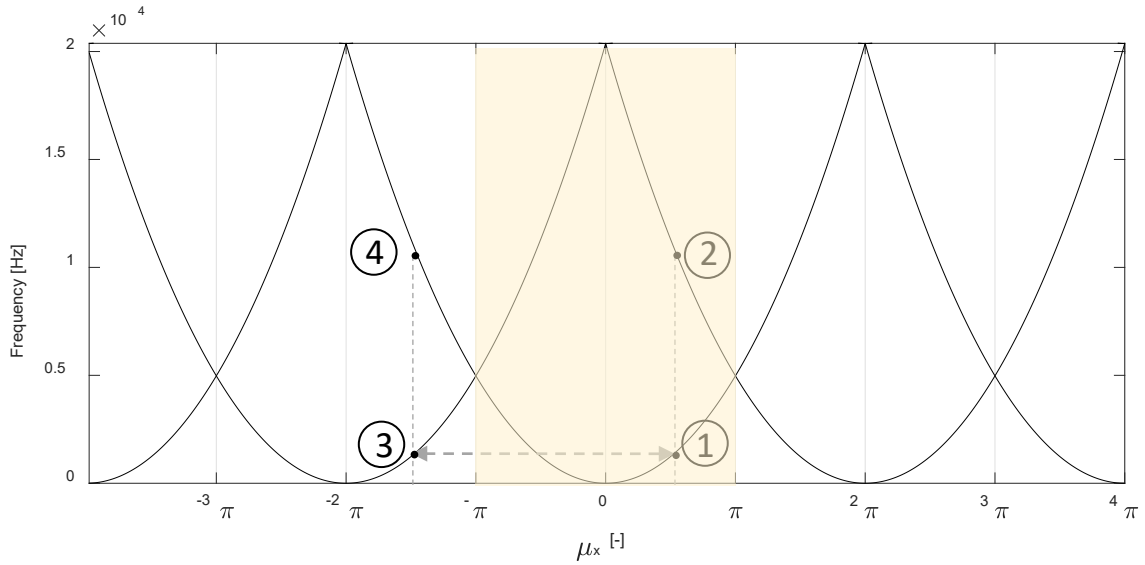
Periodicity wave domain



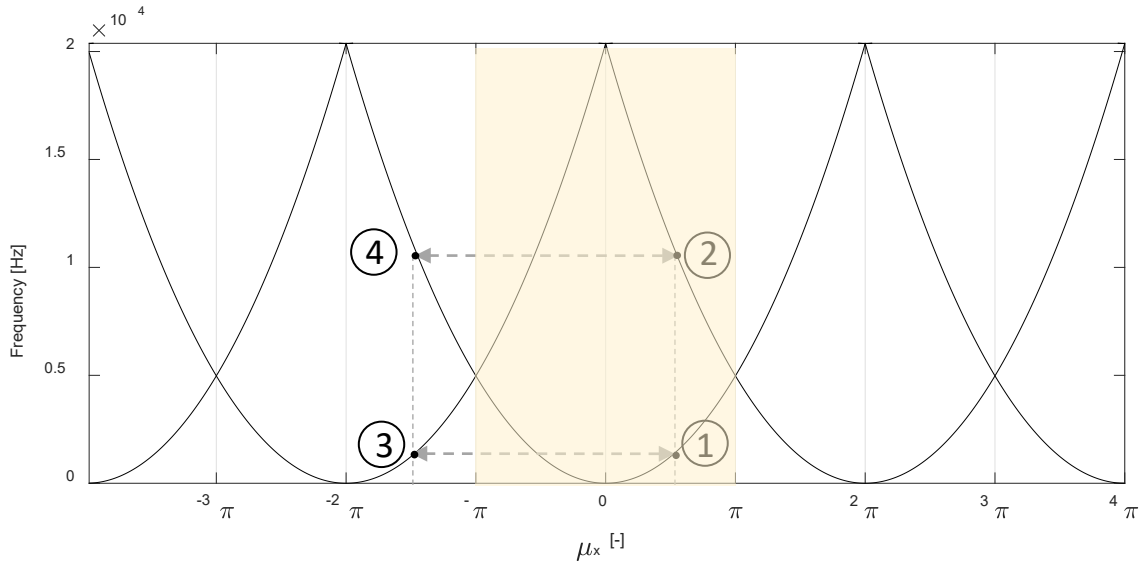
Periodicity wave domain



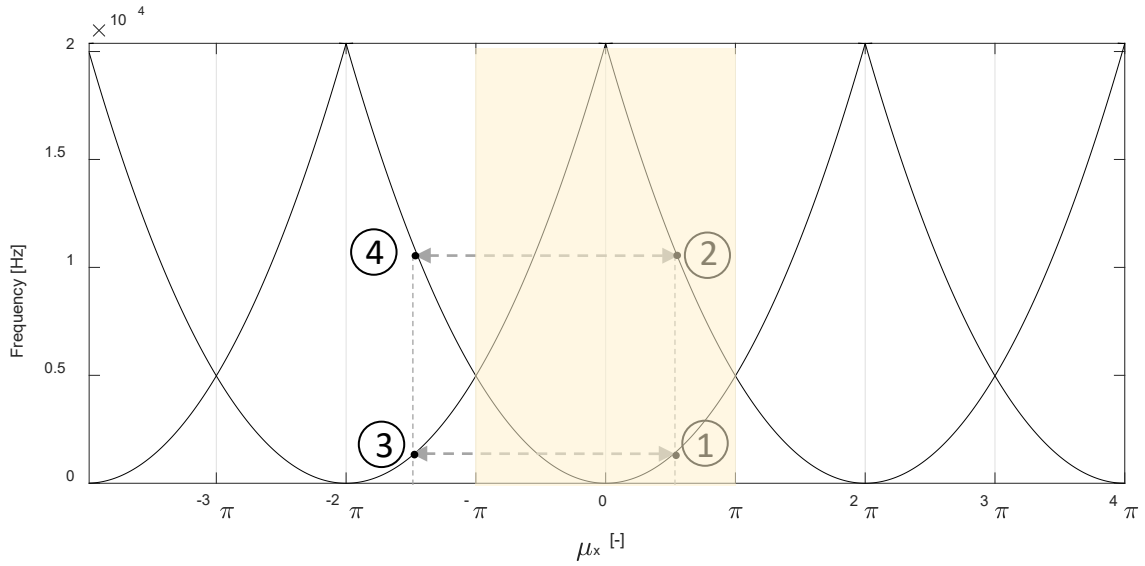
Periodicity wave domain

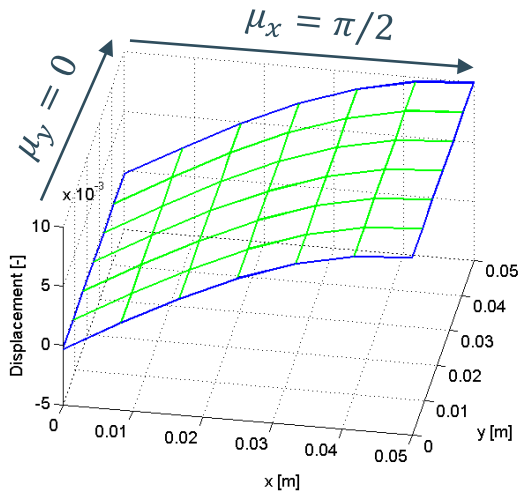


Periodicity wave domain

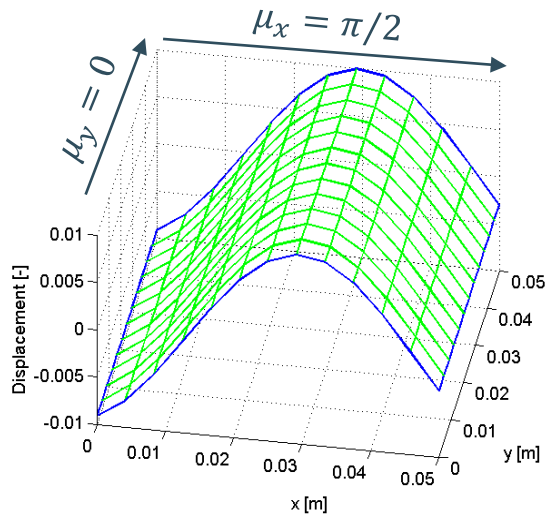


Periodicity wave domain

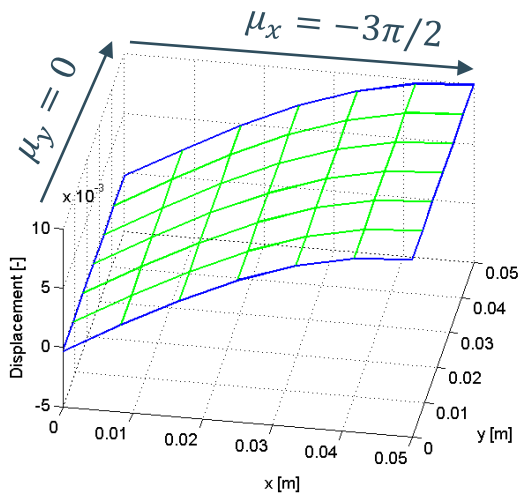




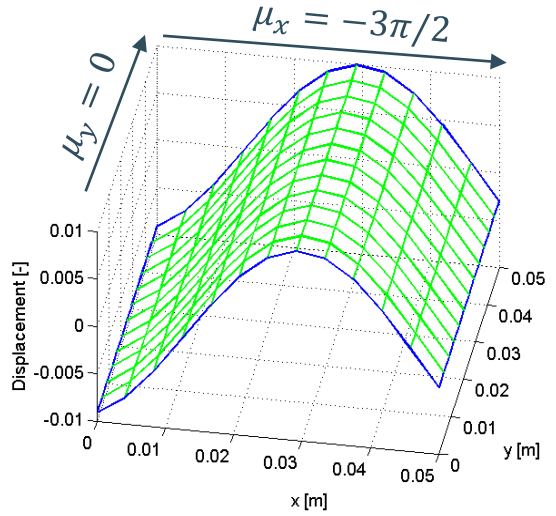
1



2

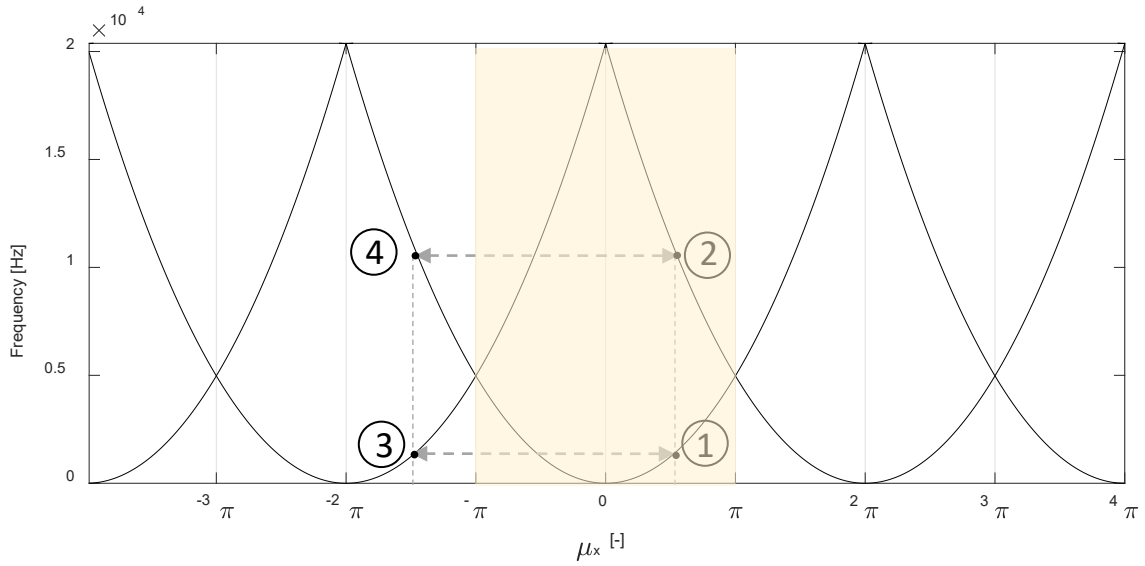


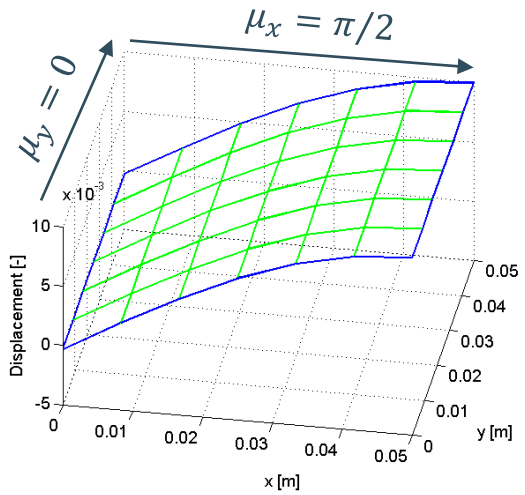
3



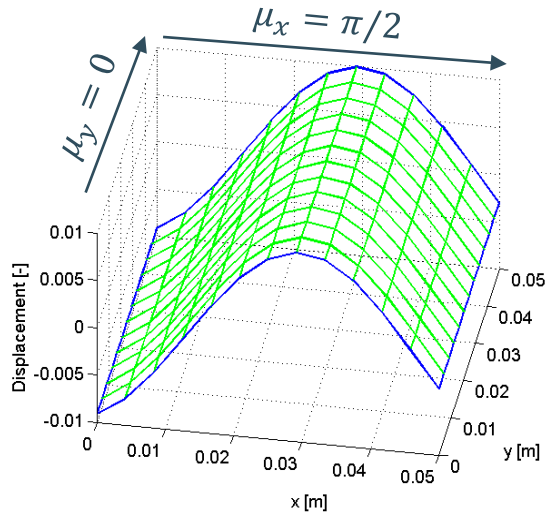
4

Periodicity wave domain

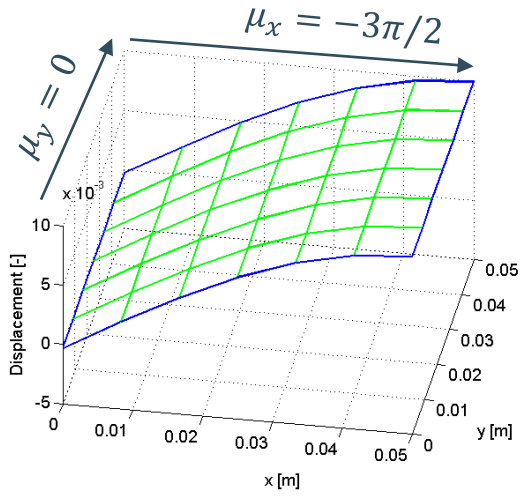




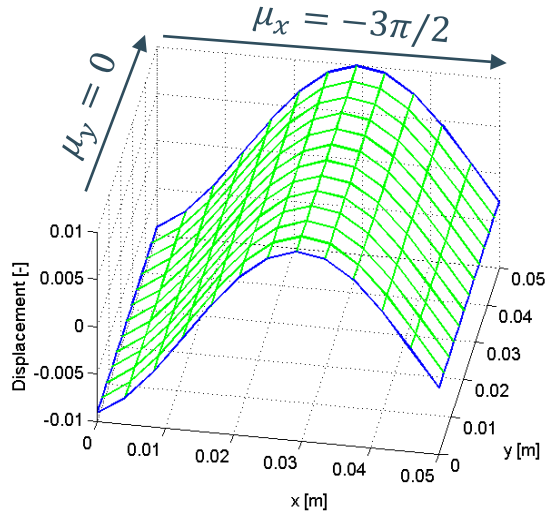
①



②



③



④

Periodic zones in wave domain

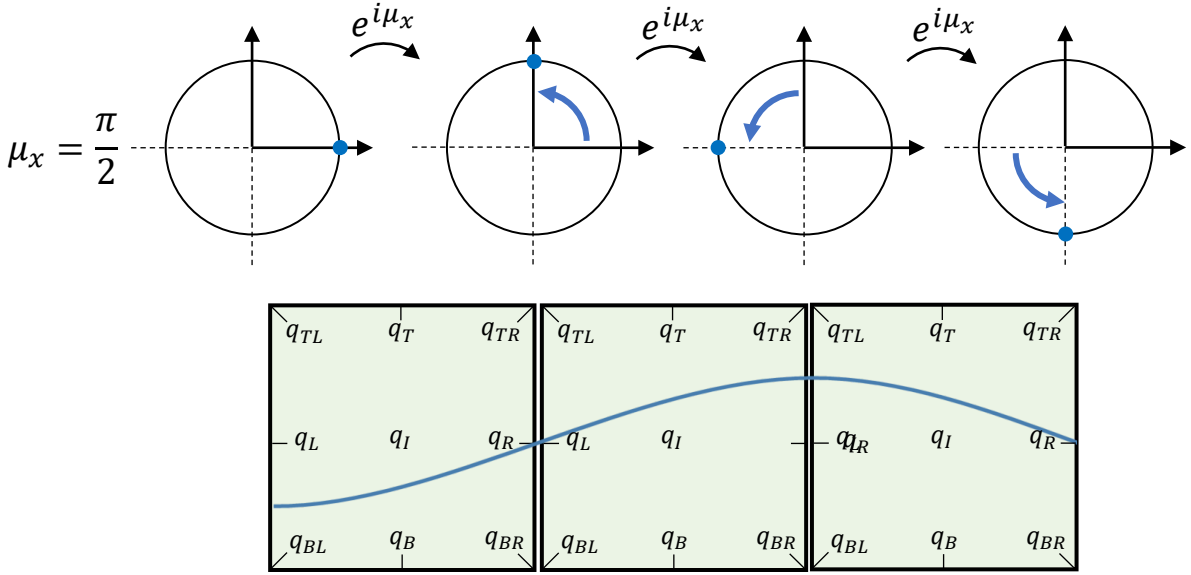
- Brillouin zones
- First Brillouin zone $\mu_x = -\pi \dots \pi, \pi_y = -\pi \dots \pi$ contains all info

Periodicity wave domain

Phase shift: $\mu_y = \pi/2$
→ left running wave

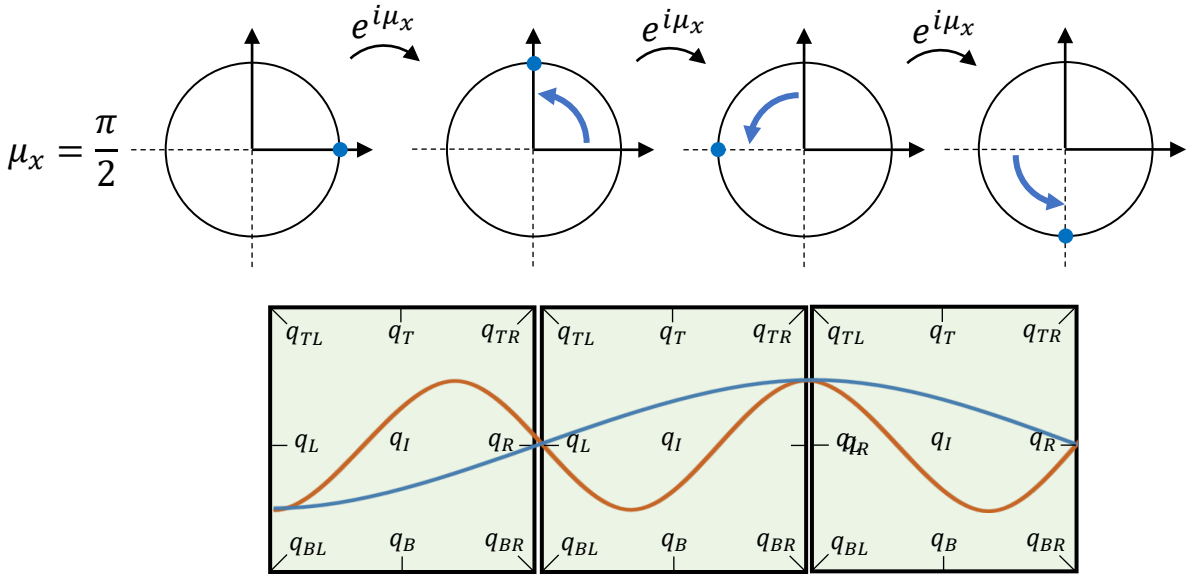
Periodicity wave domain

Phase shift: $\mu_y = \pi/2$
→ left running wave

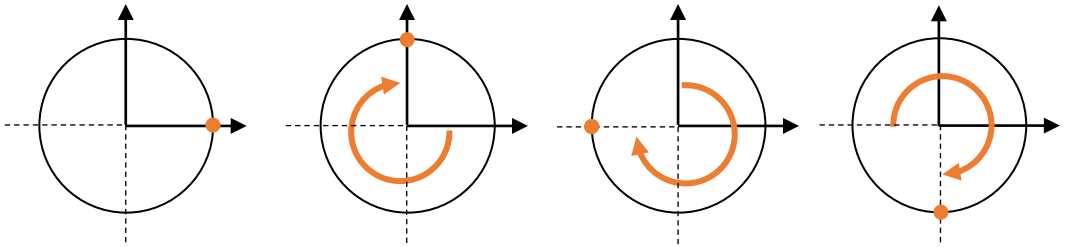


Periodicity wave domain

Phase shift: $\mu_y = \pi/2$
→ left running wave

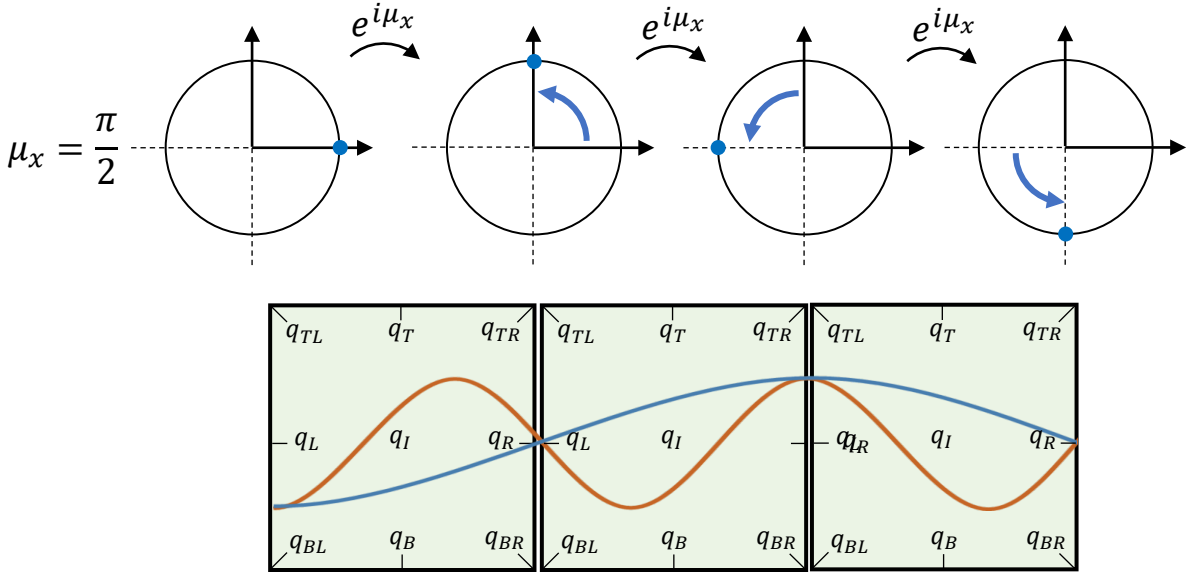


Phase shift: $\mu_y = -3\pi/2$
→ right running wave

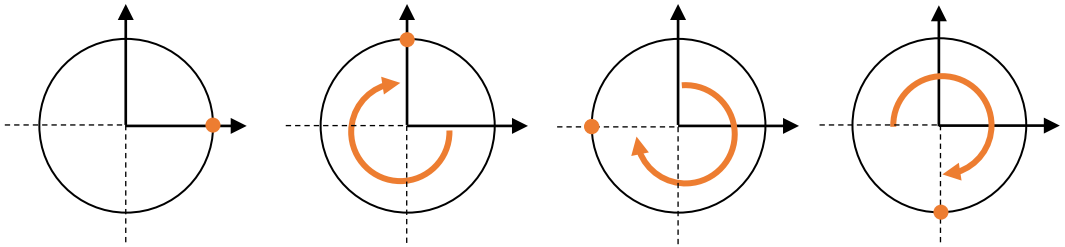


Periodicity wave domain

Phase shift: $\mu_y = \pi/2$
→ left running wave



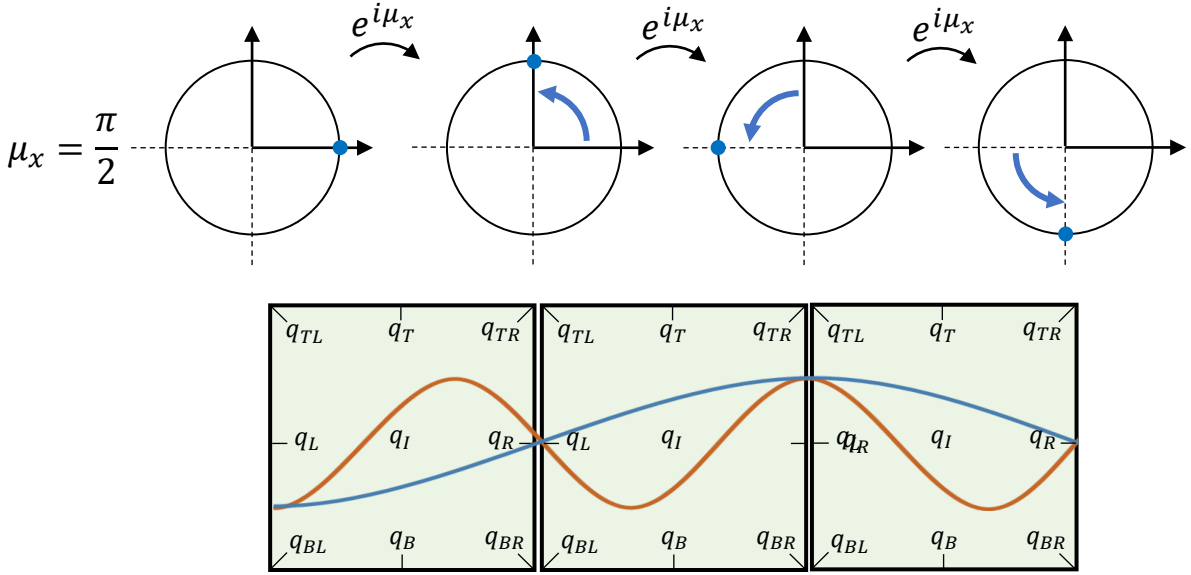
Phase shift: $\mu_y = -3\pi/2$
→ right running wave



$$\Longrightarrow \omega(\epsilon_x, \epsilon_y) = \omega(\epsilon_x \pm 2\pi n, \epsilon_y \pm 2\pi m) \text{ with } n, m \in \mathbb{N}$$

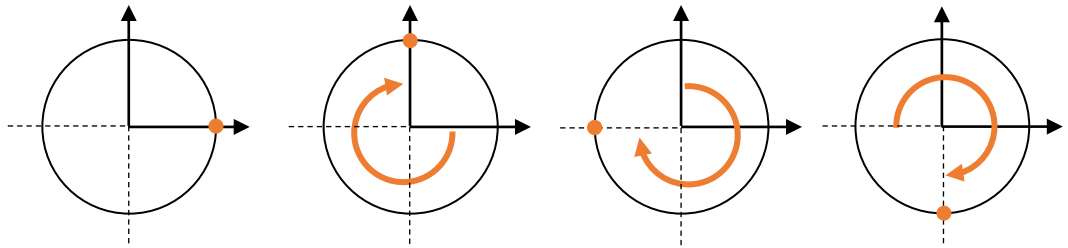
Periodicity wave domain

Phase shift: $\mu_y = \pi/2$
→ left running wave



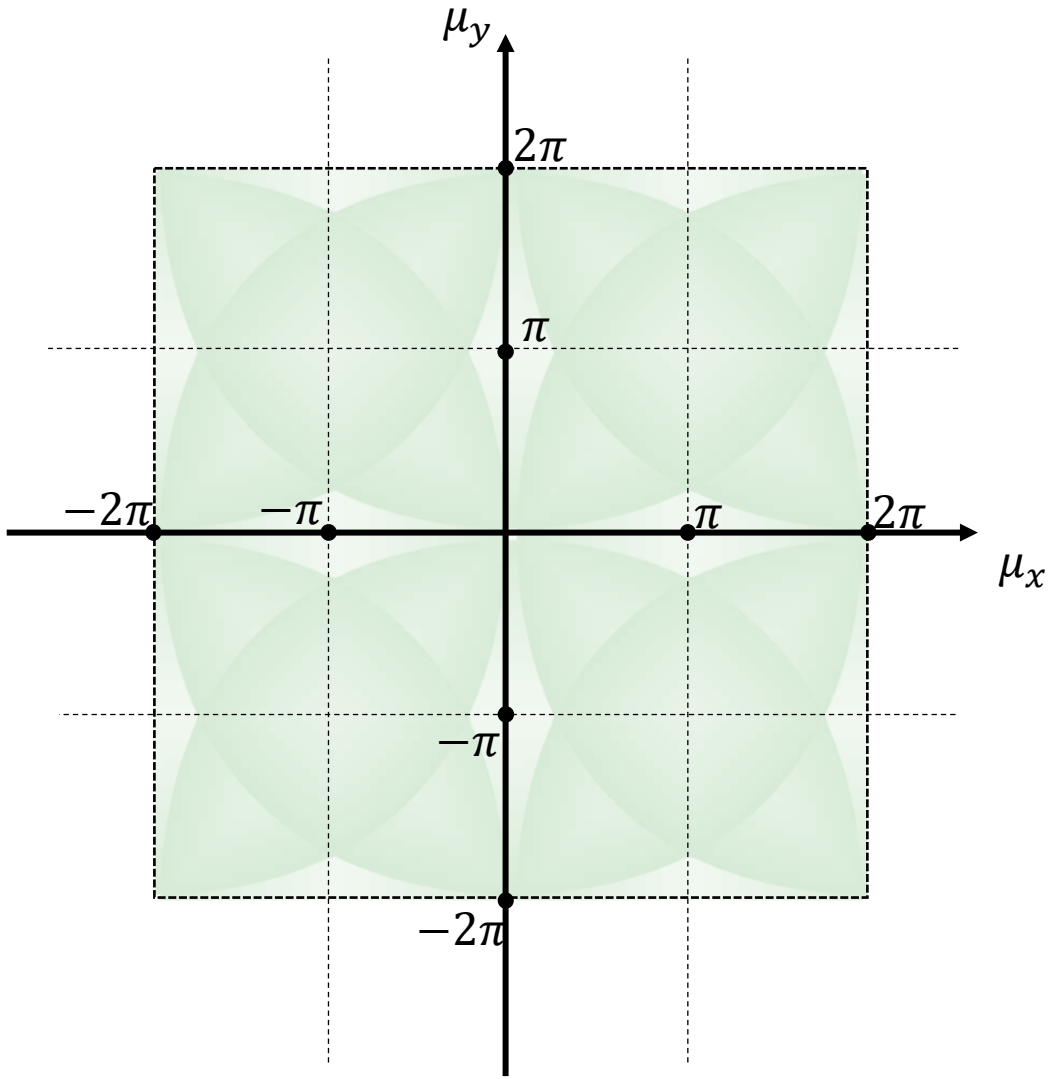
Both waves satisfy the BC: $\mathbf{q}_R = e^{j\pi/2}\mathbf{q}_L$

Phase shift: $\mu_y = -3\pi/2$
→ right running wave



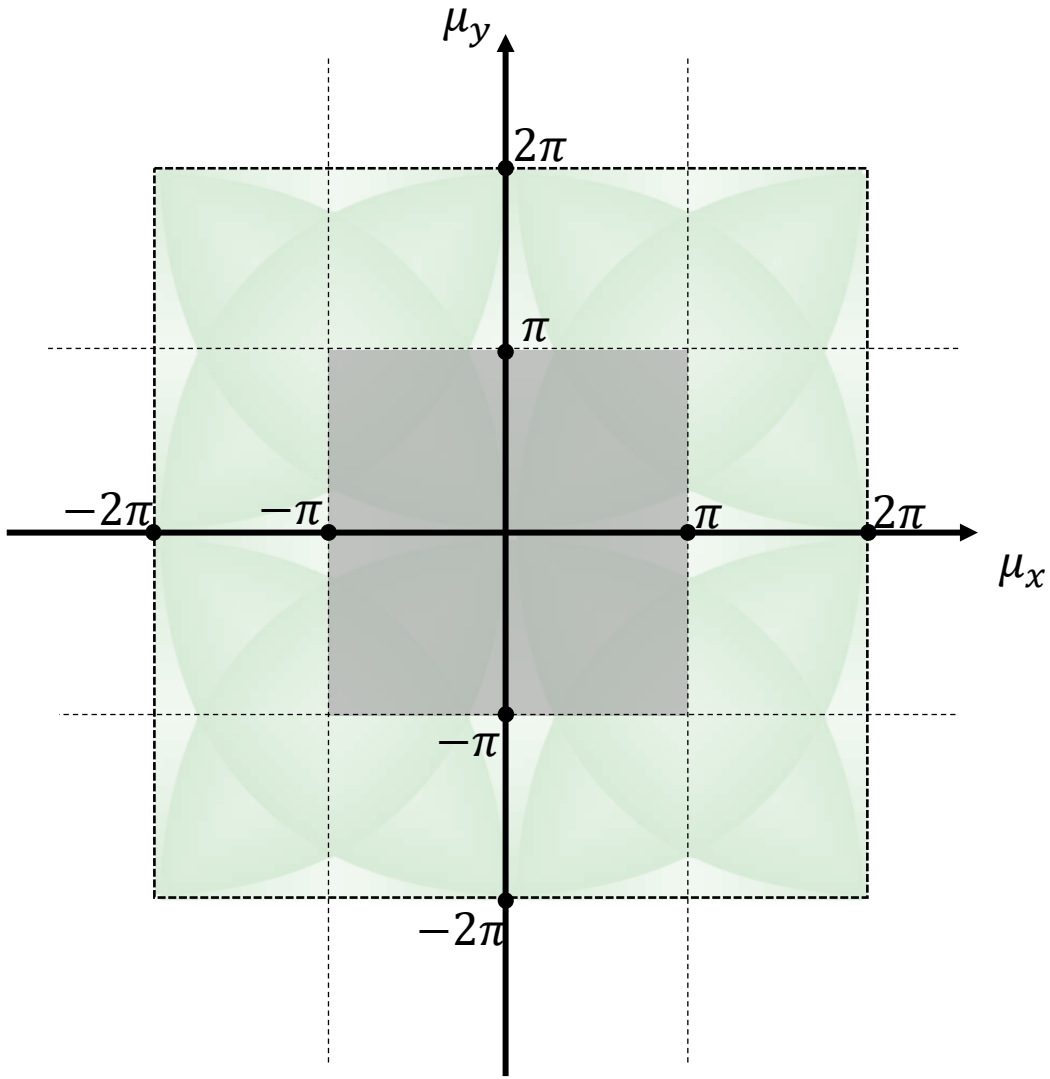
$$\Longrightarrow \omega(\epsilon_x, \epsilon_y) = \omega(\epsilon_x \pm 2\pi n, \epsilon_y \pm 2\pi m) \text{ with } n, m \in \mathbb{N}$$

Further reduction



Further reduction

Brillouin zone

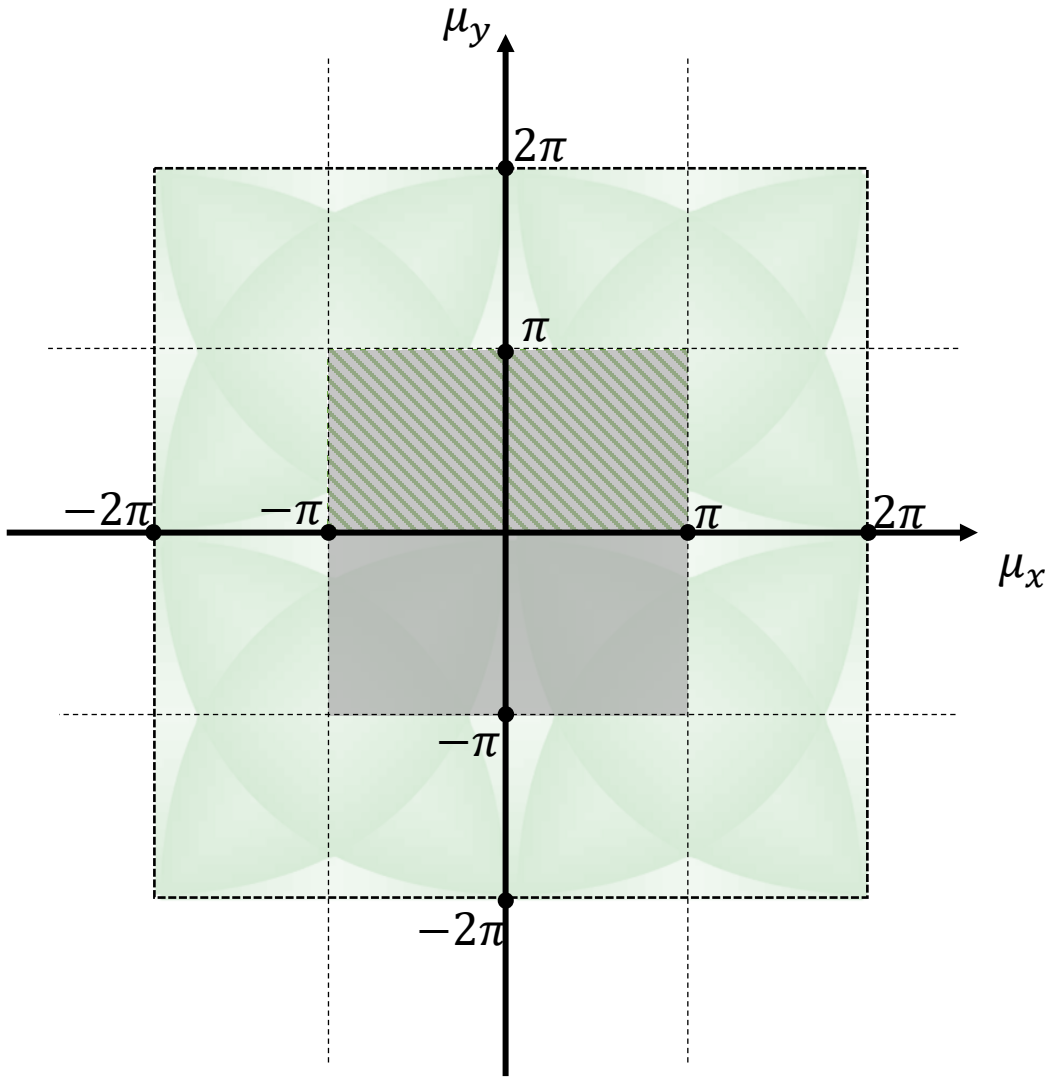


Further reduction

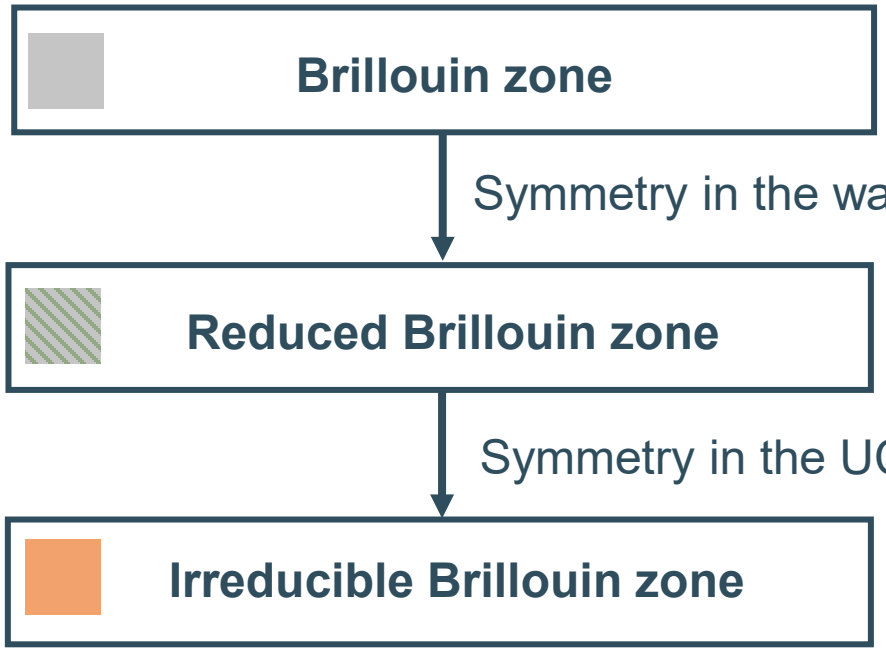
Brillouin zone

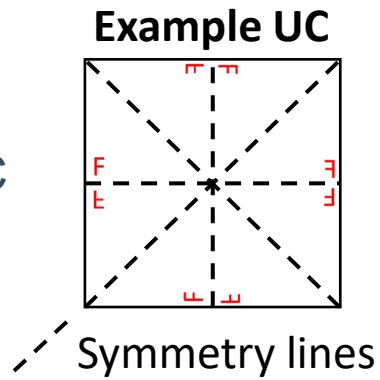
Symmetry in the wave propagation

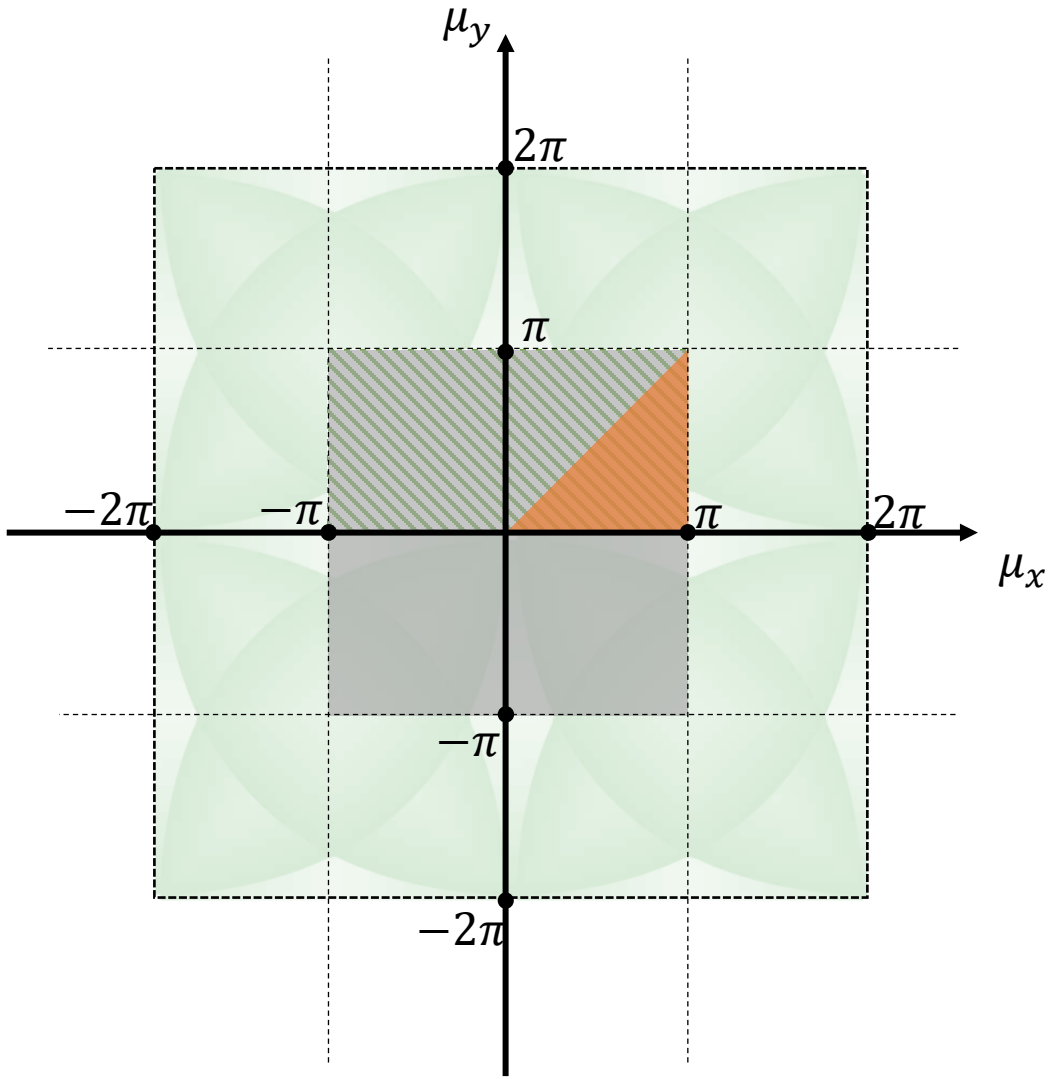
Reduced Brillouin zone



Further reduction







Further reduction

Brillouin zone

Symmetry in the wave propagation

Reduced Brillouin zone

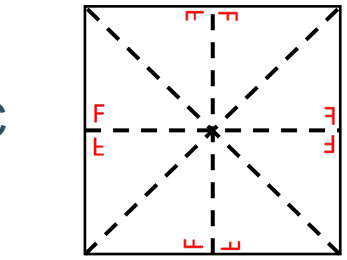
Symmetry in the UC

Irreducible Brillouin zone

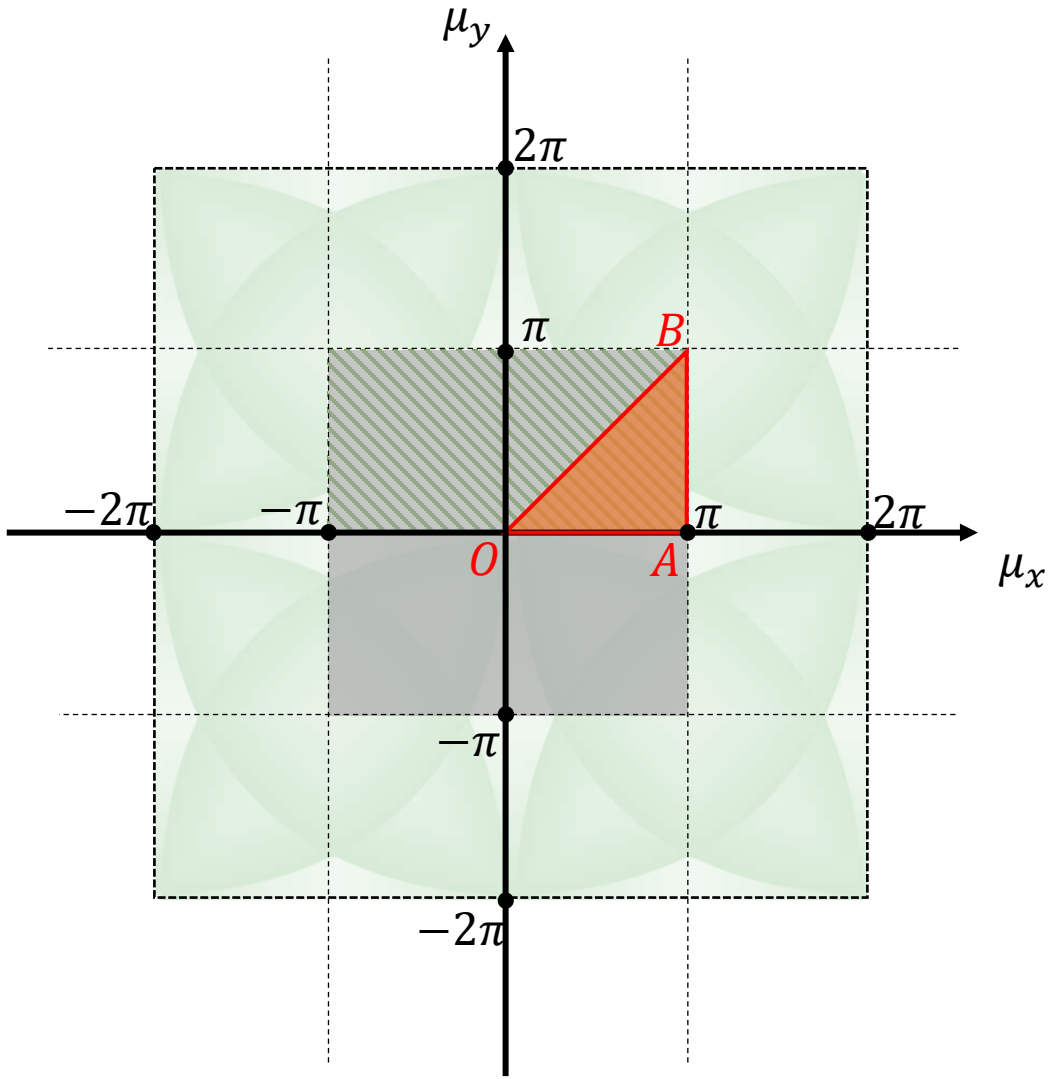
Assumption extrema

Irreducible Brillouin contour

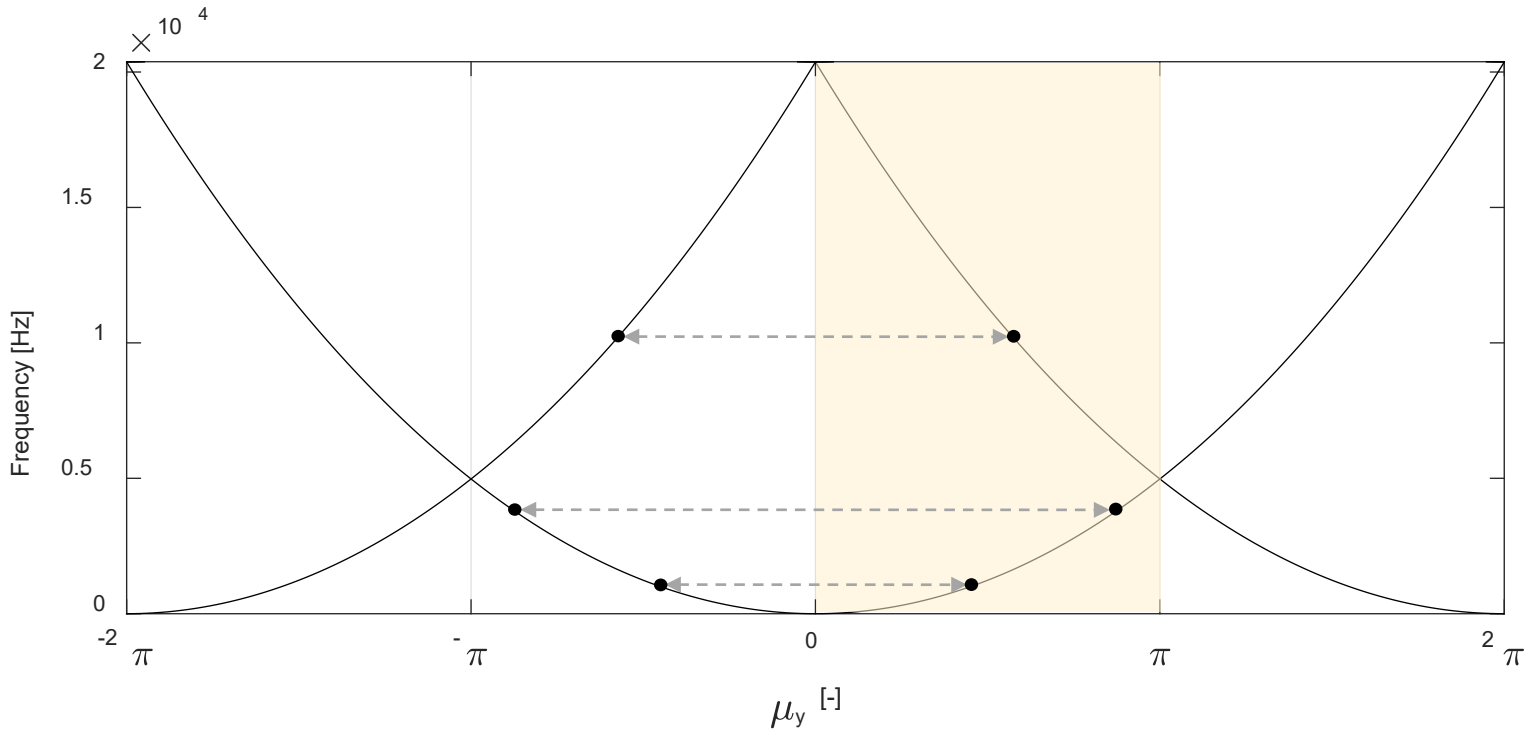
Example UC



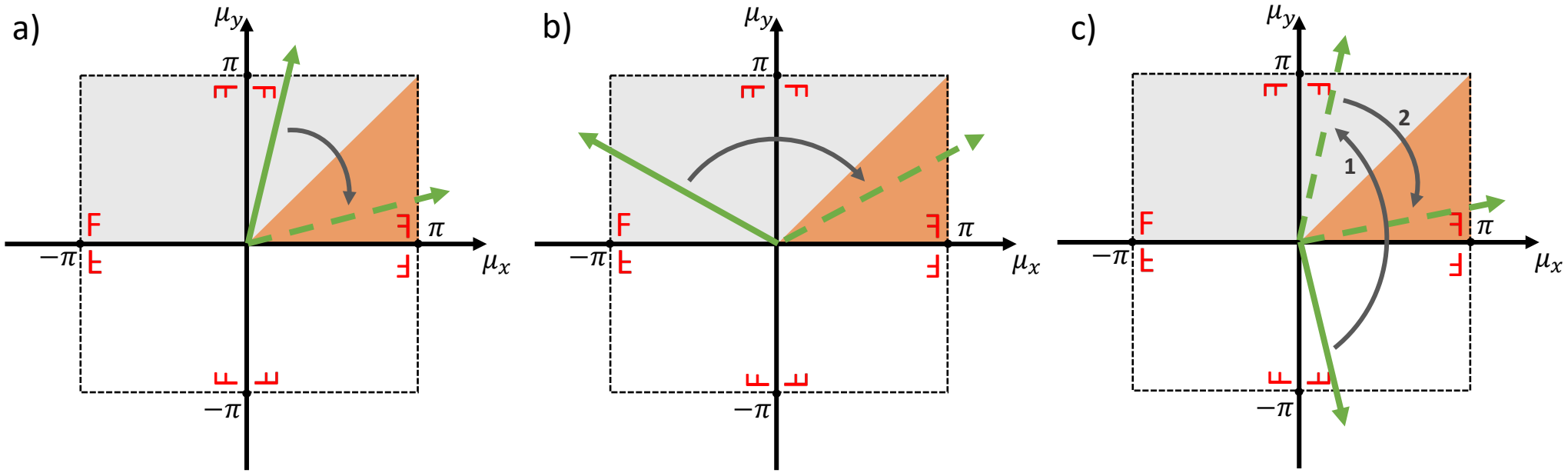
Symmetry lines



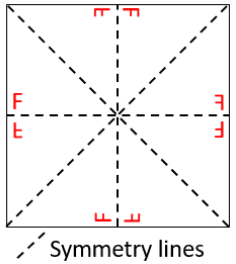
Symmetry in the wave propagation



Symmetry in the UC geometry



Example UC



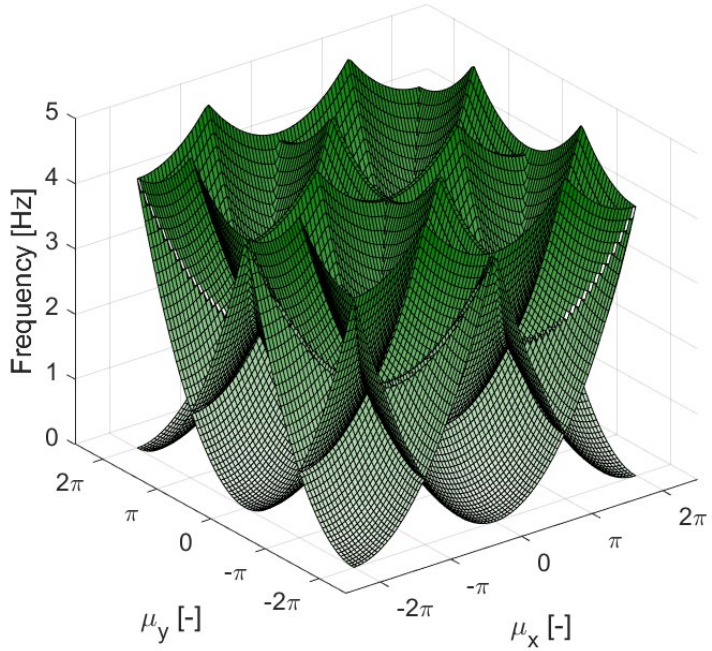
Irreducible Brillouin zone

Irreducible Brillouin zone after UC symmetry

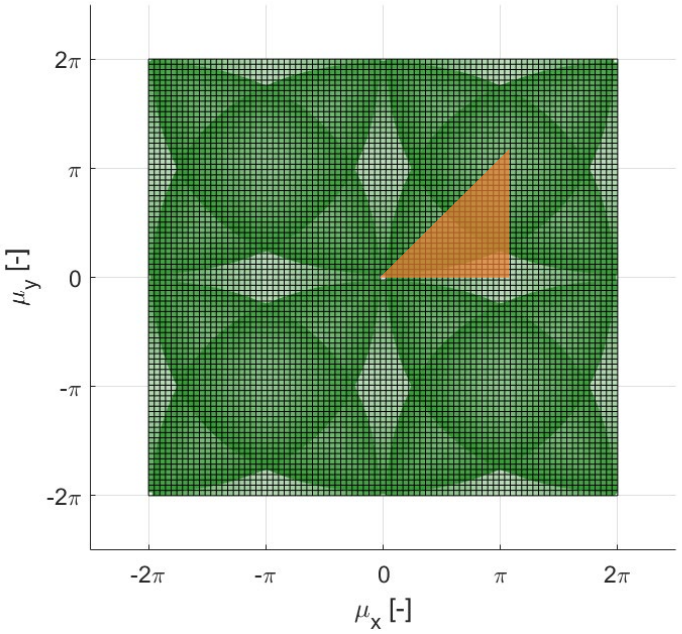
Wave of interest

Wave related by symmetry in the UC

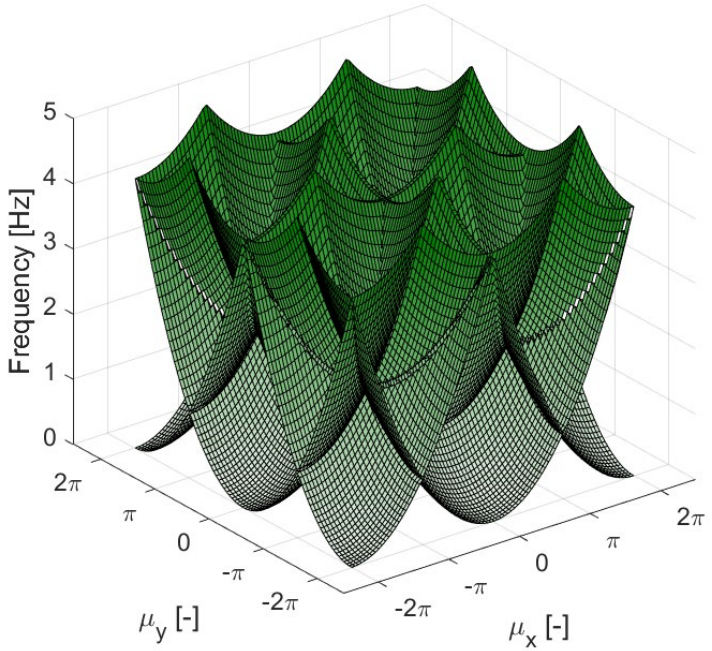
IBZ to IBC



IBZ to IBC



Irreducible Brillouin zone



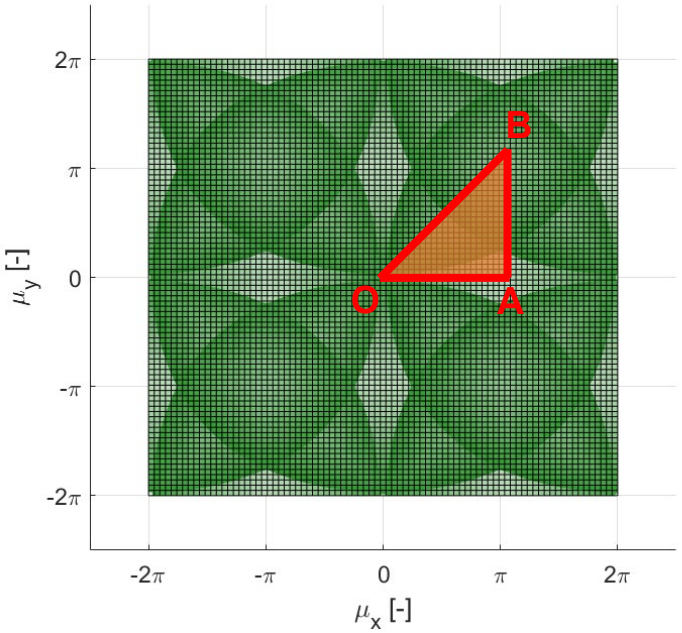
Contour of IBZ:

$$OABO \rightarrow (0,0), (\pi, 0), (\pi, \pi), (0,0)$$

- Relates half λ points across UC
- Points of high chance of interference
- Contains minima and maxima of dispersion surfaces (without proof)

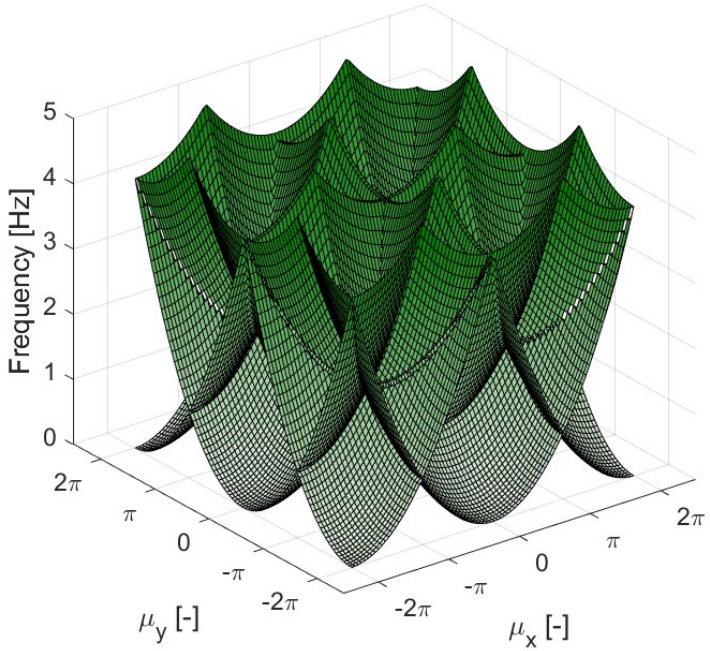
Irreducible Brillouin contour (IBC)

IBZ to IBC



Irreducible Brillouin zone

Irreducible Brillouin contour



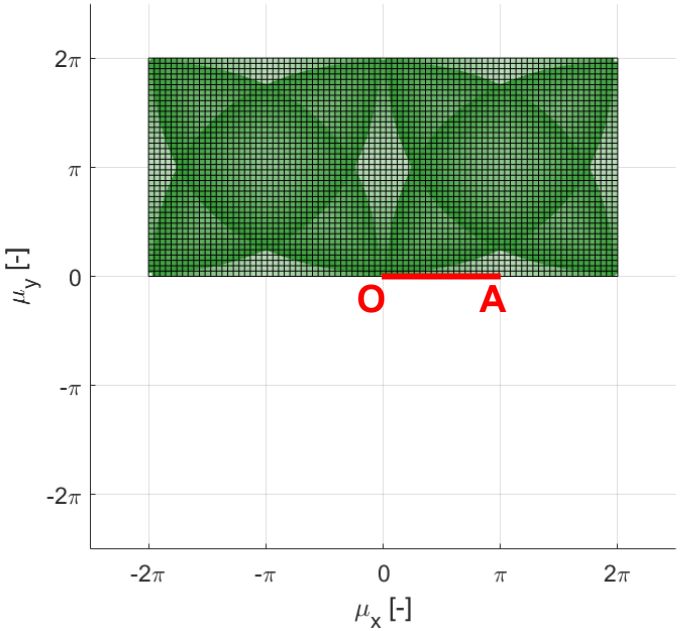
Contour of IBZ:

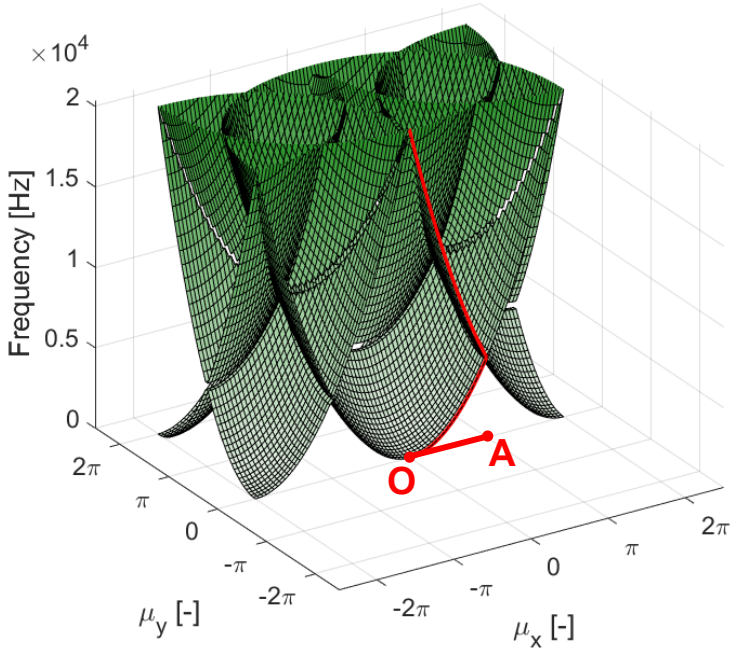
$$OABO \rightarrow (0,0), (\pi,0), (\pi,\pi), (0,0)$$

- Relates half λ points across UC
- Points of high chance of interference
- Contains minima and maxima of dispersion surfaces (without proof)

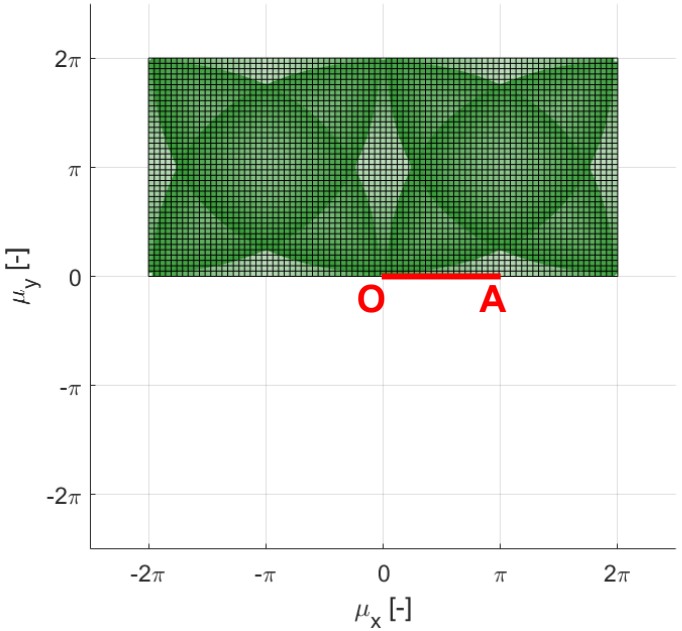
Irreducible Brillouin contour (IBC)

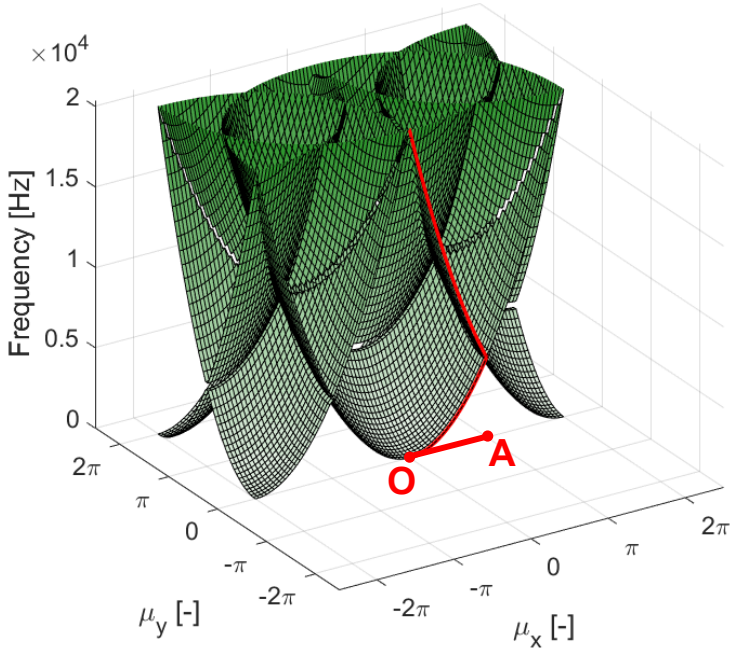
IBC

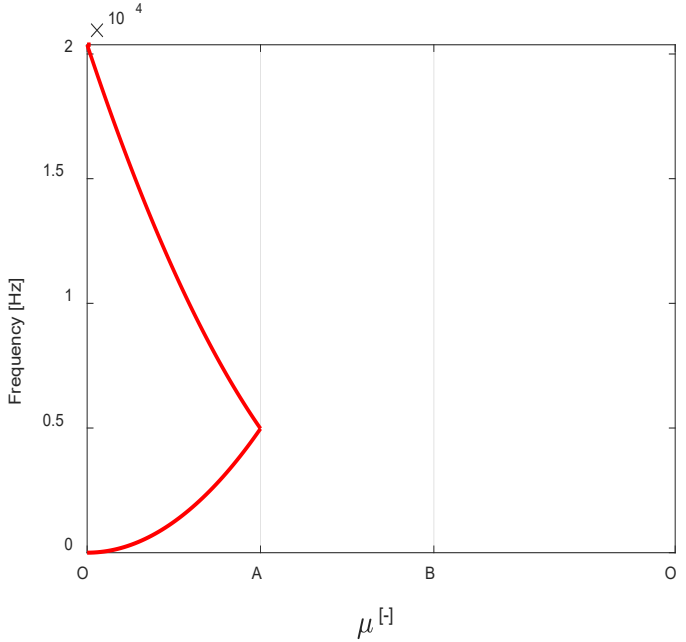

2π
π
0
-π
-2π
μ_y [-]
O
A
-2π
-π
0
π
2π
μ_x [-]


×10^4
2
1.5
1
0.5
0
Frequency [Hz]
O
A
2π
π
0
-π
-2π
μ_y [-]
-2π
-π
0
π
2π
μ_x [-]

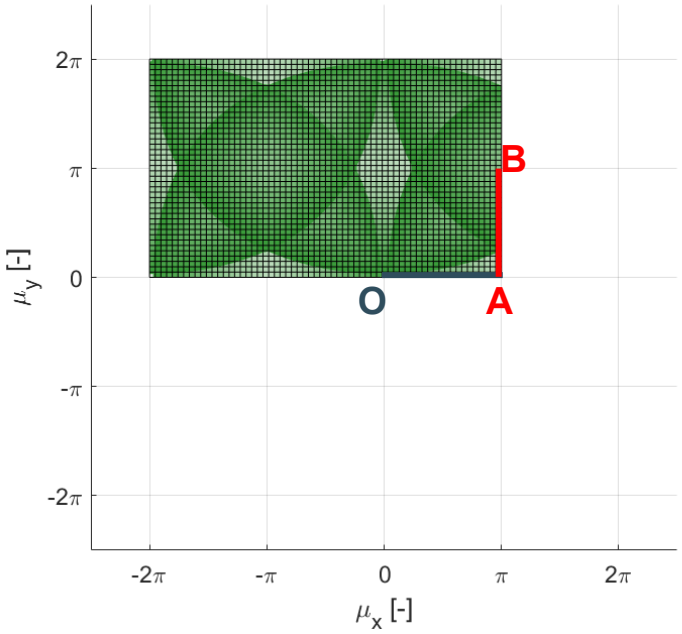
IBC

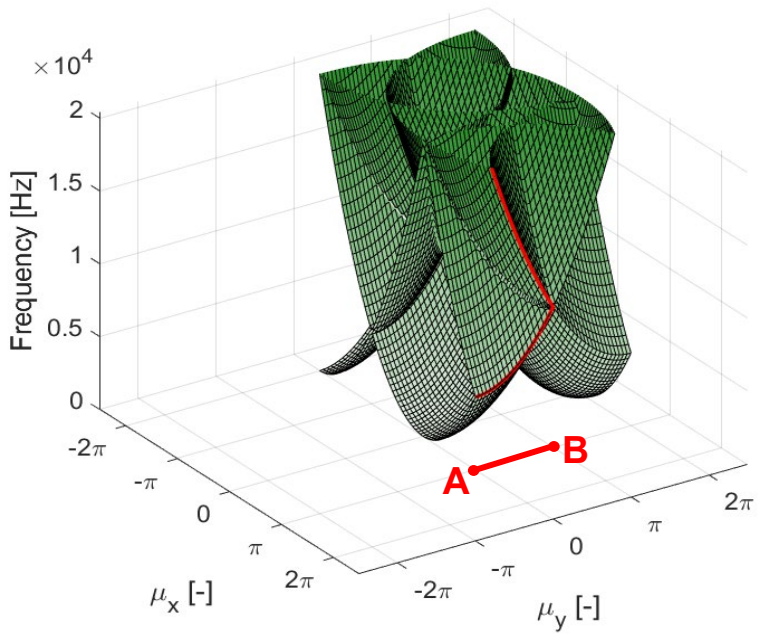

2π
π
0
-π
-2π
μ_y [-]
O
A
-2π
-π
0
π
2π
μ_x [-]

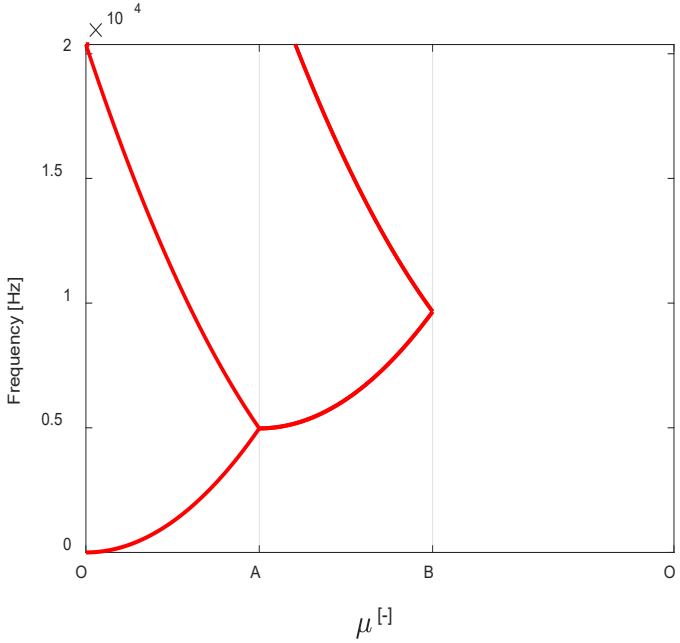

×10^4
2
1.5
1
0.5
0
Frequency [Hz]
O
A
2π
π
0
-π
-2π
μ_y [-]
-2π
-π
0
π
2π
μ_x [-]


×10^4
2
1.5
1
0.5
0
Frequency [Hz]
O
A
B
O
μ [-]

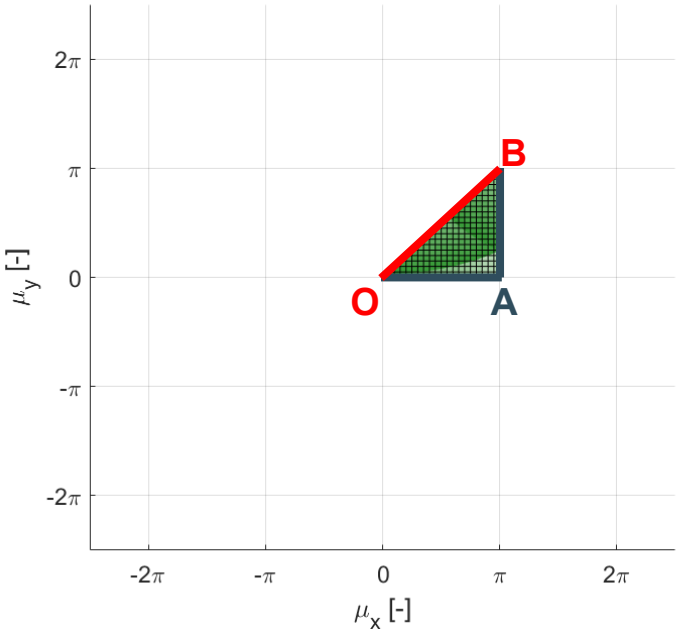
IBC

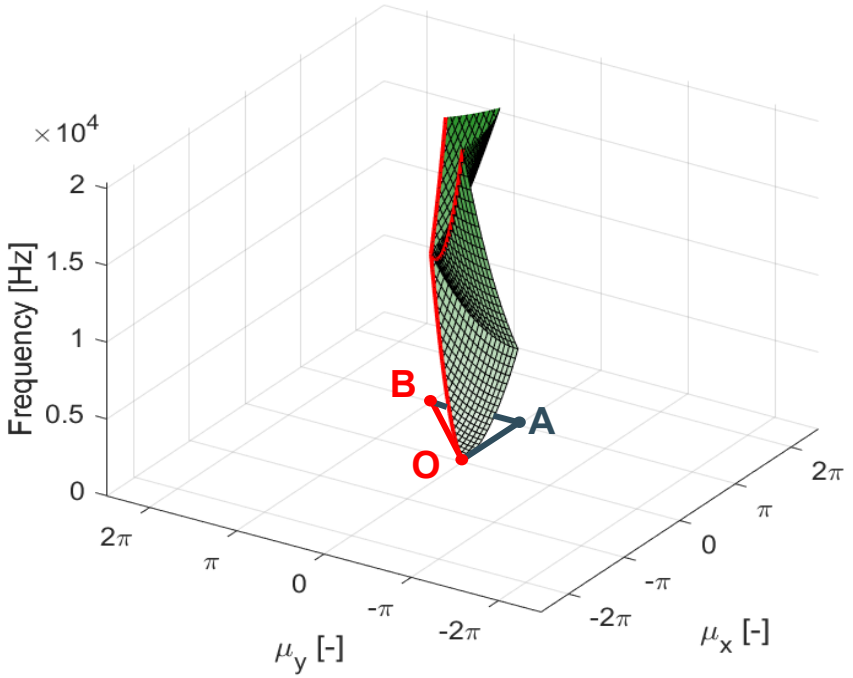

2π
π
0
-π
-2π
μ_y [-]
B
O
A
-2π
-π
0
π
2π
μ_x [-]

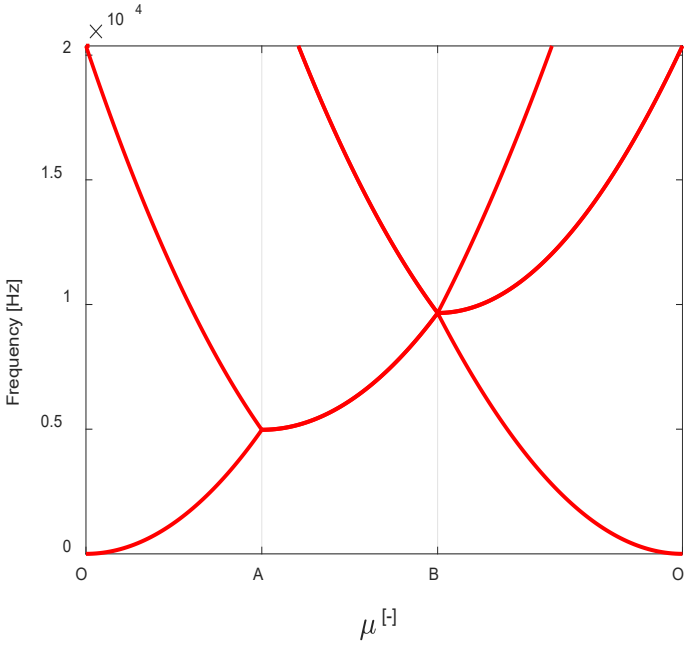

×10^4
2
1.5
1
0.5
0
Frequency [Hz]
-2π
-π
0
π
2π
μ_x [-]
A
B
2π
π
0
-π
-2π
μ_y [-]


×10^4
2
1.5
1
0.5
0
Frequency [Hz]
O
A
B
O
μ [-]

IBC


2π
π
0
-π
-2π
μ_y [-]
-2π
-π
0
π
2π
μ_x [-]
B
O
A

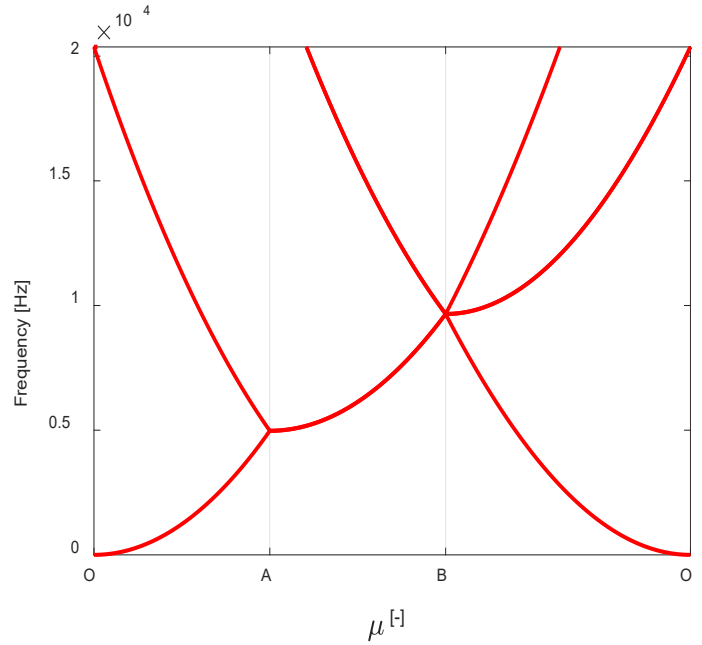

×10^4
2
1.5
1
0.5
0
Frequency [Hz]
2π
π
0
-π
-2π
μ_y [-]
2π
π
0
-π
-2π
μ_x [-]
B
O
A


×10^4
2
1.5
1
0.5
0
Frequency [Hz]
O
A
B
O
μ [-]

IBC

Dispersion curves along the IBC

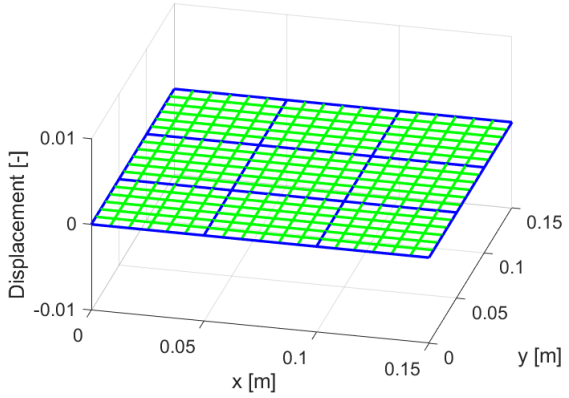
Frequencies and direction of free wave propagation in the infinite periodic structure



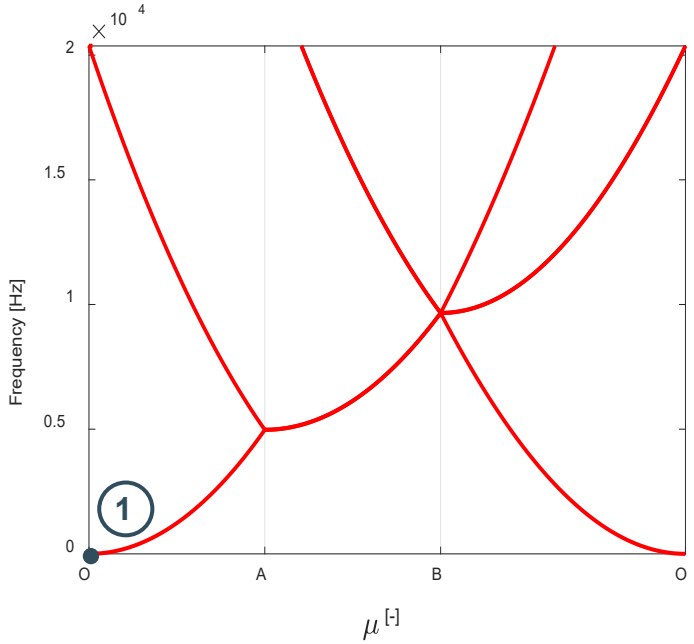
IBC

Dispersion curves along the IBC

Frequencies and direction of free wave propagation in the infinite periodic structure



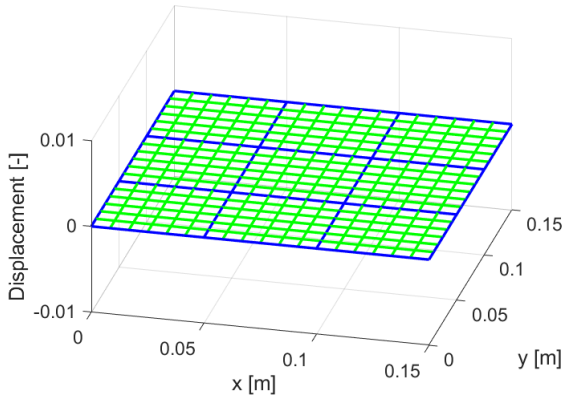
1



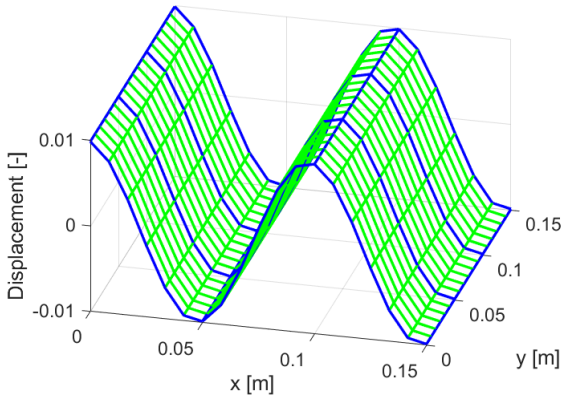
IBC

Dispersion curves along the IBC

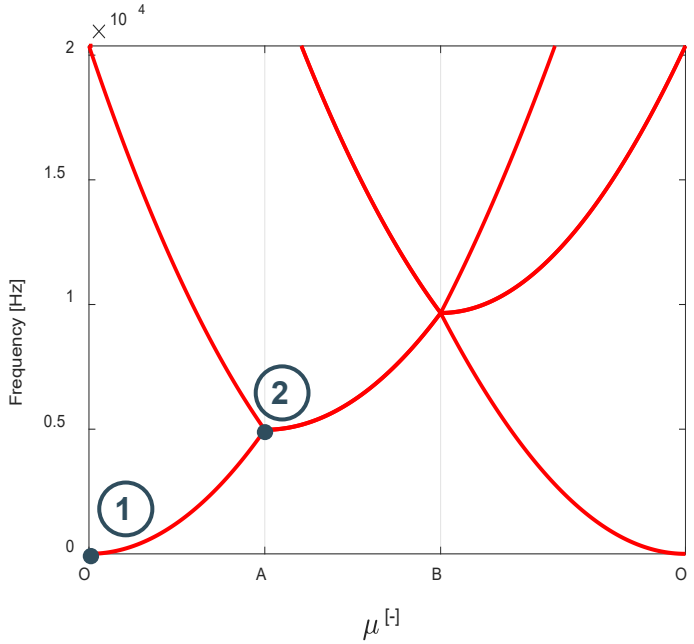
Frequencies and direction of free wave propagation in the infinite periodic structure



1



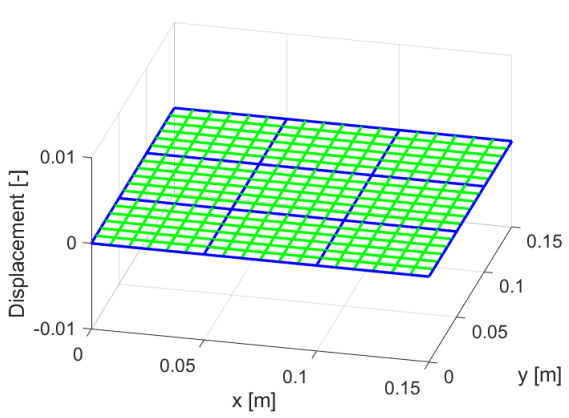
2



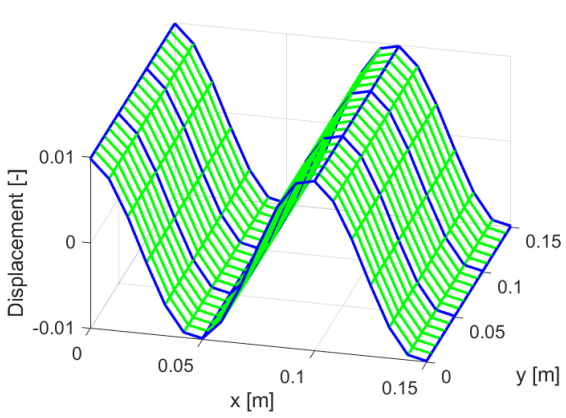
IBC

Dispersion curves along the IBC

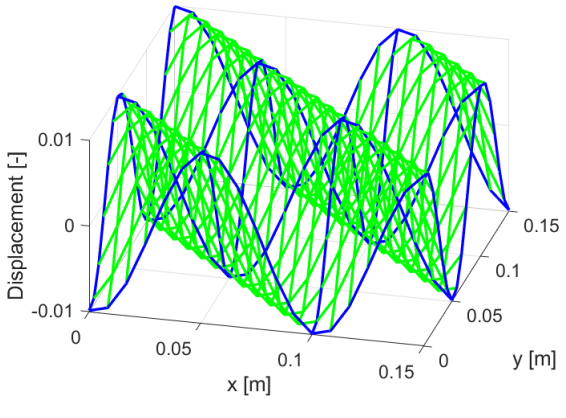
Frequencies and direction of free wave propagation in the infinite periodic structure



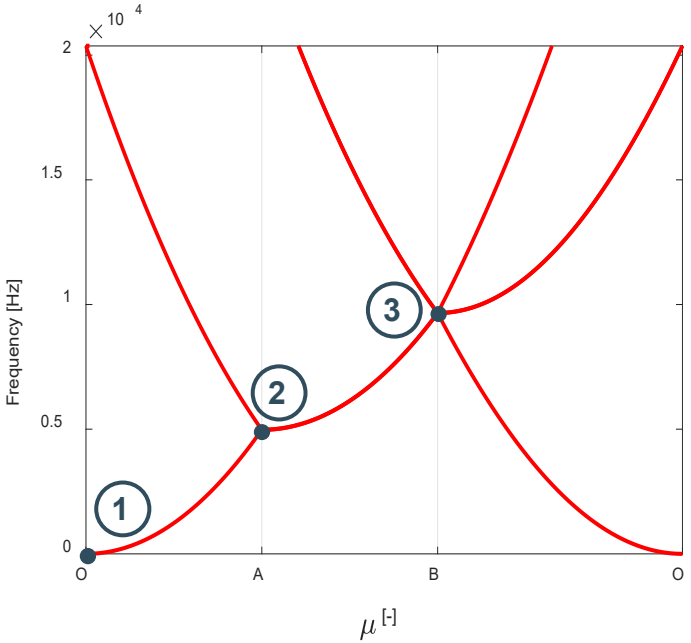
1



2

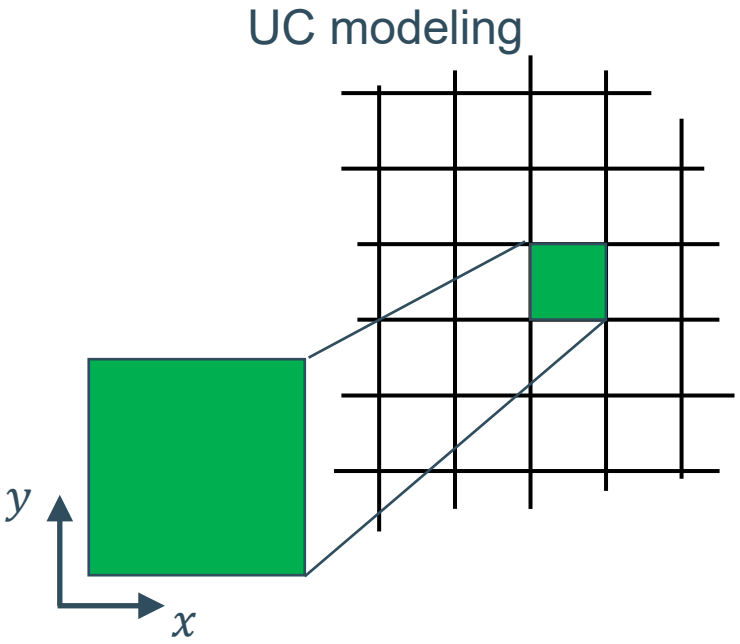


3



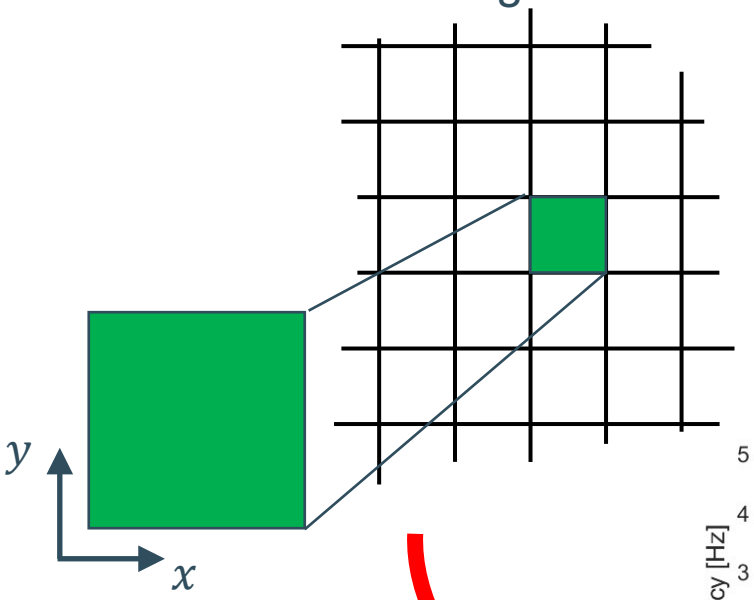
Summary: dispersion surfaces to curves

Summary: dispersion surfaces to curves



Summary: dispersion surfaces to curves

UC modeling

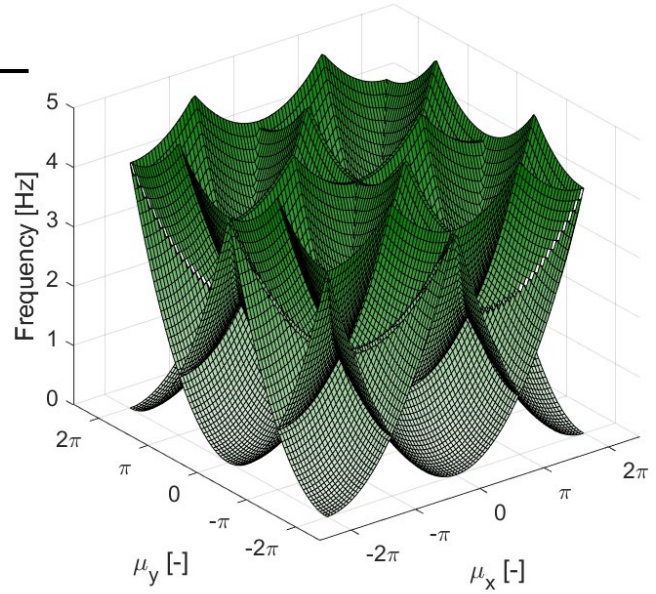


$$\left[\tilde{\mathbf{K}} - \omega^2 \tilde{\mathbf{M}}\right]\tilde{\mathbf{q}} = \mathbf{0}$$

Impose real (μ_x, μ_y)

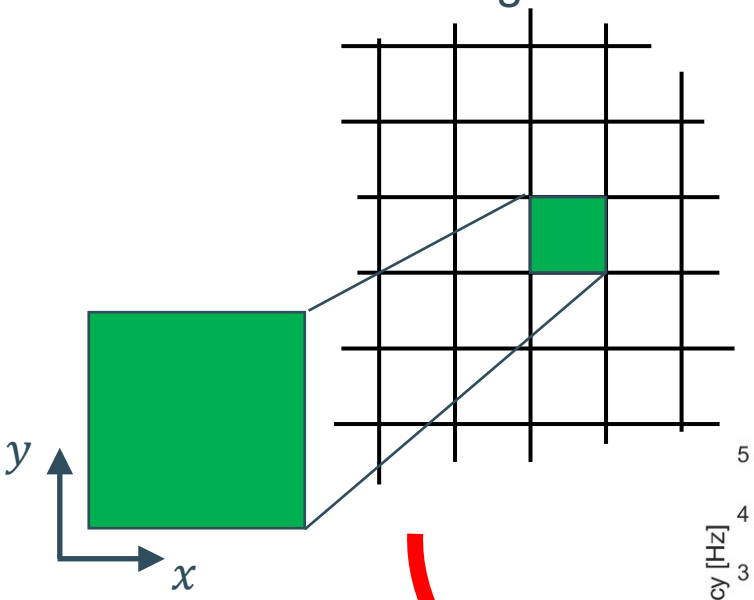
Solve for real ω

Dispersion surfaces

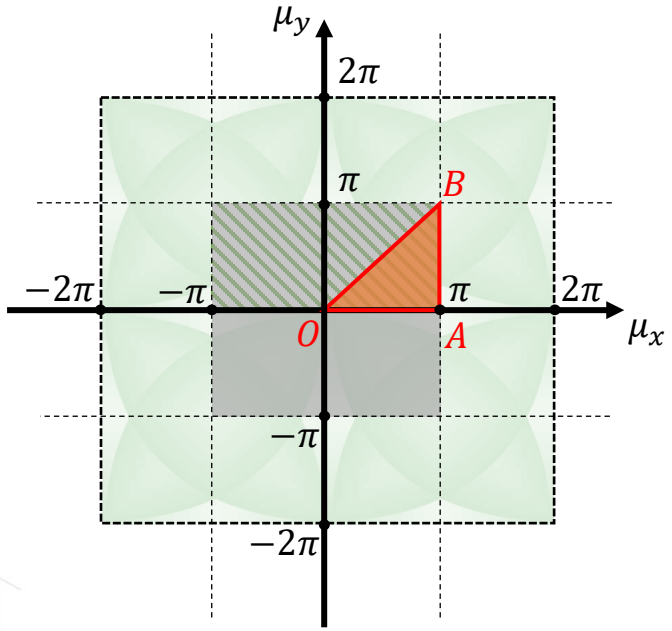


Summary: dispersion surfaces to curves

UC modeling



Periodicity & symmetry



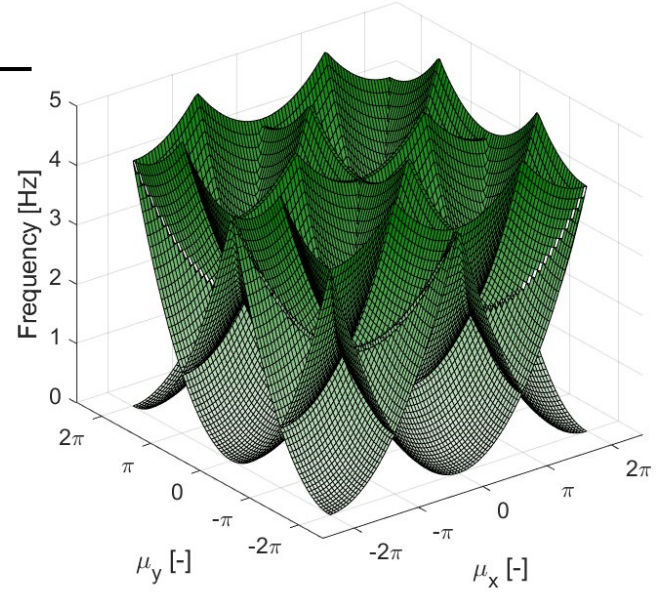
IBC

$\left[\tilde{\mathbf{K}} - \omega^2 \tilde{\mathbf{M}}\right]\tilde{\mathbf{q}} = \mathbf{0}$

Impose real (μ_x, μ_y)

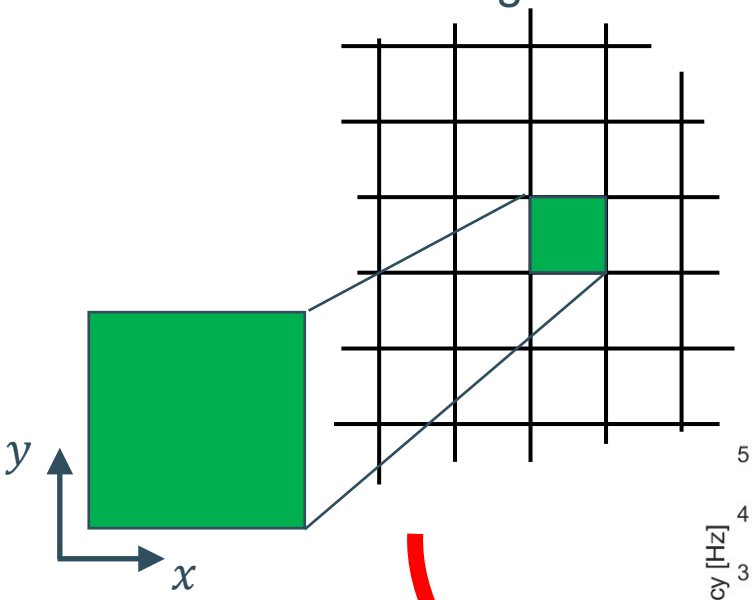
Solve for real ω

Dispersion surfaces



Summary: dispersion surfaces to curves

UC modeling

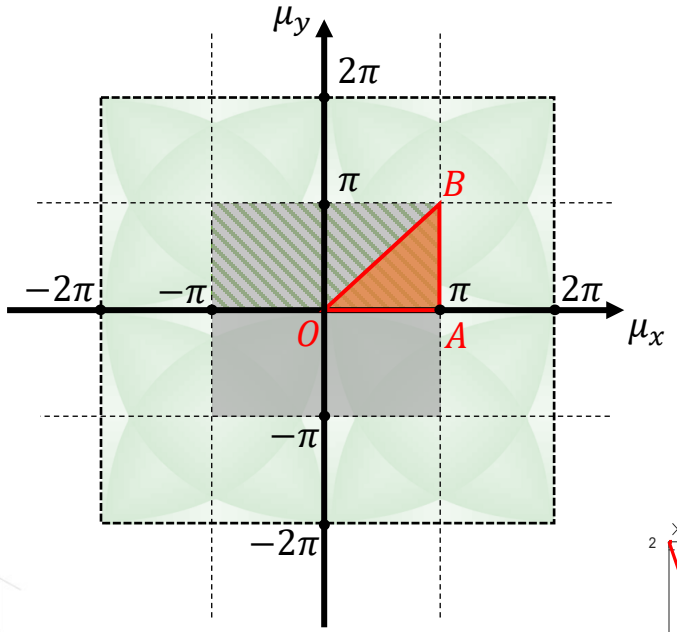


$[\tilde{\mathbf{K}} - \omega^2 \tilde{\mathbf{M}}]\tilde{\mathbf{q}} = \mathbf{0}$

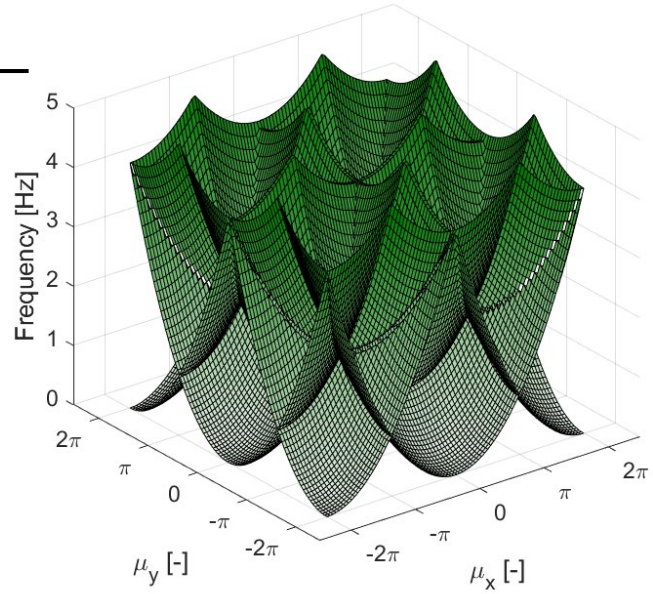
Impose real (μ_x, μ_y)

Solve for real ω

Periodicity & symmetry



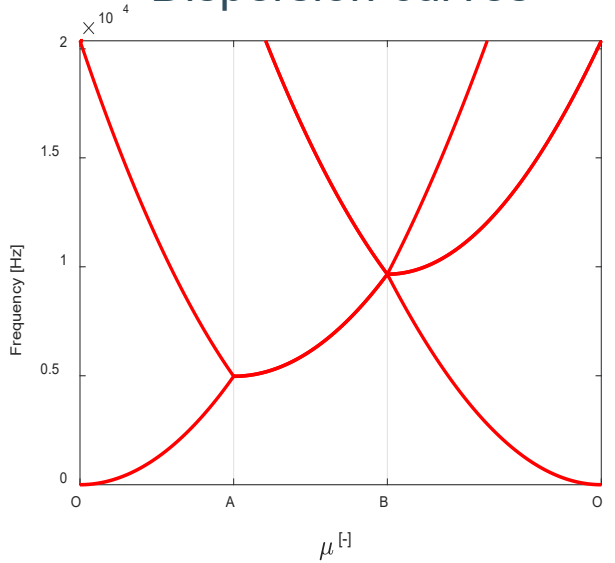
Dispersion surfaces



IBC

Extrema on IBC

Dispersion curves



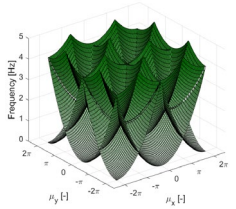
Matlab implementation

```
% Element size
ax = Lx/nelx;
ay = Ly/nely;
az = Lz/nelz;
% Information UC
n_elem = nelx*nely*nelz;
n_nodes = (nelx+1)*(nely+1)*(nelz+1);
nDOF = 3;                          % Number of DOFs per node;
n_DOFs = nDOF*n_nodes;
mass_UC = rho*(nelx*ax)*(nely*ay)*(nelz*az);
% Element matrices
[KE,ME] = KM(E,v,rho,ax,ay,az);
% FEM information (hard-coded for nDOF = 3)
nodeNrs = reshape(1:n_nodes, 1+nely, 1+nelz, 1+nelx); % nodes numbering
cMat = reshape(nDOF * nodeNrs(1:nely, 1:nelz, 1:nelx)+1, n_elem, 1) + ...
    [0,1,2,3*( nely+1)*(nelz+1)+[0,1,2,-3,-2,-1],-3,-2,-1,3*(nely + ...
    1) +[0,1,2],3*( nely+1)*(nelz+2)+[0,1,2,-3,-2,-1],3*(nely+1)+[-3,-2,-1]]; % connectivity matrix DO

%% DOFs partitioning
DofNrs = reshape(1:n_DOFs, nDOF*(nely+1), nelz+1, nelx+1);
dofs.L  = DofNrs(nDOF+1:end-nDOF,:,1); dofs.L = dofs.L(:);
dofs.R  = DofNrs(nDOF+1:end-nDOF,:,end); dofs.R = dofs.R(:);
dofs.T  = DofNrs(1:nDOF,:,2:end-1); dofs.T = dofs.T(:);
dofs.B  = DofNrs(end-(nDOF-1):end,:,2:end-1); dofs.B = dofs.B(:);
dofs.TL = DofNrs(1:nDOF,:,1); dofs.TL = dofs.TL(:);
dofs.TR = DofNrs(1:nDOF,:,end); dofs.TR = dofs.TR(:);
dofs.BL = DofNrs(end-(nDOF-1):end,:,1); dofs.BL = dofs.BL(:);
dofs.BR = DofNrs(end-(nDOF-1):end,:,end); dofs.BR = dofs.BR(:);
dofs.I  = DofNrs(nDOF+1:end-nDOF,:,2:end-1); dofs.I = dofs.I(:);

%% Get system matrices
% Assemble full matrices
iL = reshape(kron(cMat,ones(24,1))',24*24*n_elem,1);
jL = reshape(kron(cMat,ones(1,24))',24*24*n_elem,1);
sK = reshape(KE(:)*(ones(n_elem,1)'),24*24*n_elem,1);
sM = reshape(ME(:)*(ones(n_elem,1)'),24*24*n_elem,1);
K = sparse(iL,jL,sK);
M = sparse(iL,jL,sM);
% Add resonator/ mass
switch scatterer
    case 'mass'
      mass = m_ratio*mass_UC;
      dofM = 3*nodeNrs(floor(end/2),end,floor(end/2));
      M(dofM,dofM) = M(dofM,dofM) + mass;
    case 'resonator'
      % Add extra zero row and column to K, M
      K = [K zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
      M = [M zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
      % Define mass and stiffness
      mass = m_ratio*mass_UC;
      k_res  = (f_res*2*pi)^2*mass;
      dofK = 3*nodeNrs(floor(end/2),end,floor(end/2));
      n_DOFs = n_DOFs+1;
      dofM = n_DOFs;
      % Add resonator matrices to UC mass, stiffness matrices
      K([dofK dofM],[dofK dofM]) = K([dofK dofM],[dofK dofM]) + [k_res -k_res; -k_res k_res];
      M([dofK dofM],[dofK dofM]) = M([dofK dofM],[dofK dofM]) + [0 0; 0 mass];
      dofs.I = [dofs.I; dofM];
end

%% Sampling IBC
mu = 1i*cont_co(1,:)';
```

EVP to dispersion curves

Dispersion eigenvalue problem

Dispersion surfaces

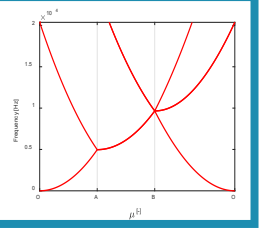


First Brillouin zone

Reduced Brillouin zone

Irreducible Brillouin contour

Dispersion curves



EVP to dispersion curves

Dispersion eigenvalue problem

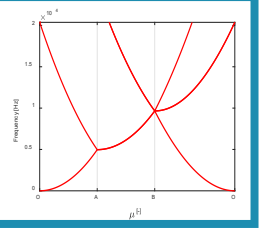
Dispersion surfaces

First Brillouin zone

Reduced Brillouin zone

Irreducible Brillouin contour

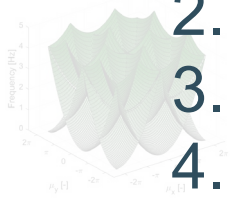
Dispersion curves



EVP to dispersion curves

Dispersion eigenvalue problem

Dispersion surfaces



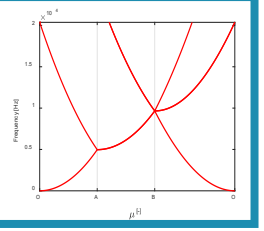
First Brillouin zone

1. Input Data
2. FE preprocessing
3. DOFs partitioning
4. System Matrix assembly
5. Sampling the UC
6. Dispersion curve calculations

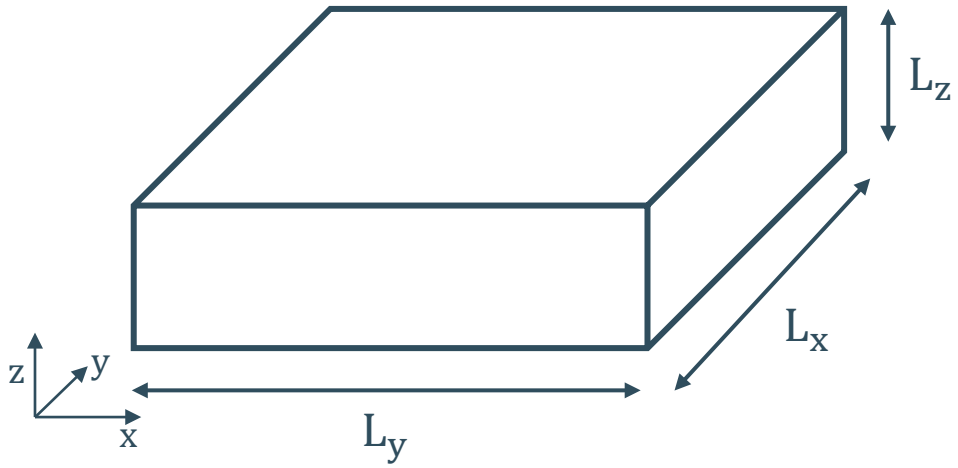
Reduced Brillouin zone

Irreducible Brillouin contour

Dispersion curves

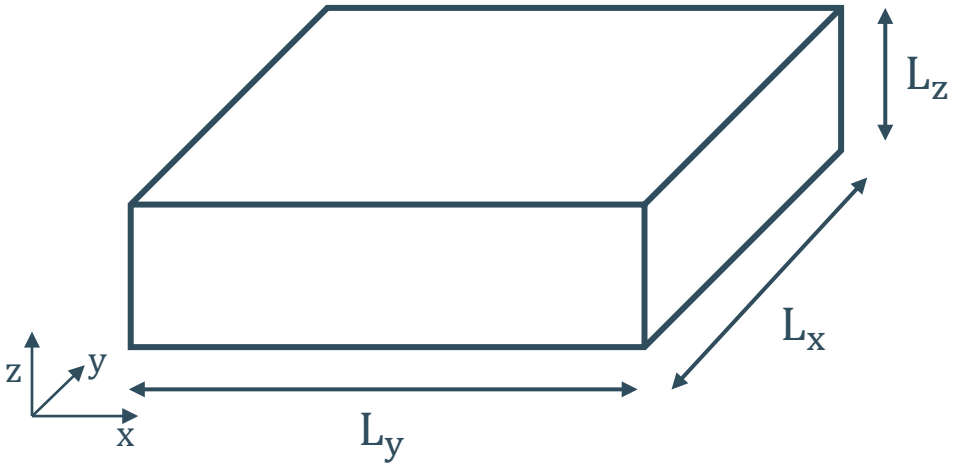


Input data

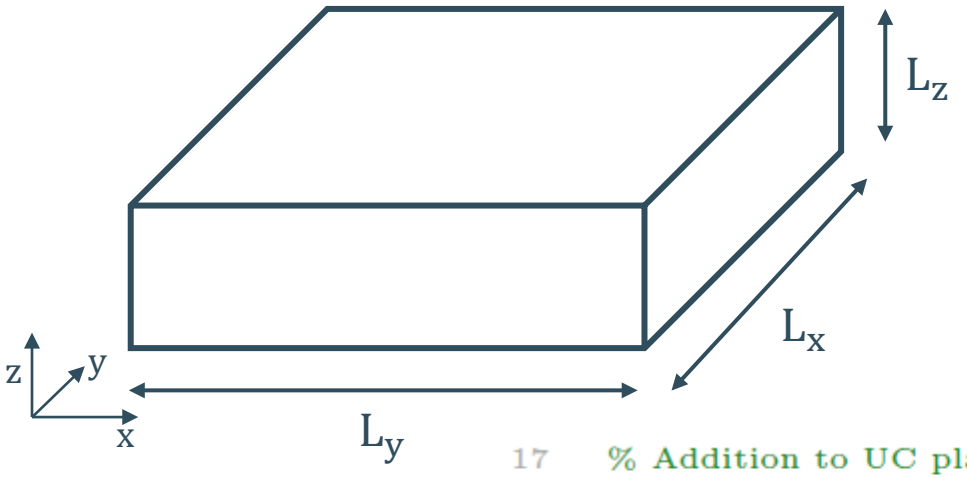


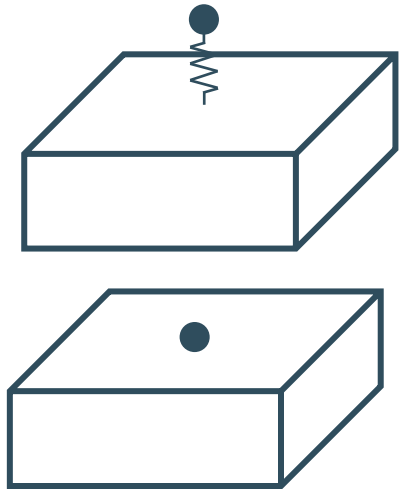

```
5   % UC size
6   Lx = 0.05;          % UC size x-direction [m]
7   Ly = 0.05;          % UC size y-direction [m]
8   Lz = 0.005;         % UC size z-direction [m]
9   % Material
10  E = 210e9;          % Young's modulus [Pa]
11  v = 0.3;            % Poisson ratio [-]
12  rho = 7800;         % Mass density [kg/m^3]
13  % Mesh definition
14  nelx = 10;          % Number of elements in x-dir [-] (>0)
15  nely = 10;          % Number of elements in y-dir [-] (>0)
16  nelz = 3;           % Number of elements in z-dir [-] (>0)
```

Input data



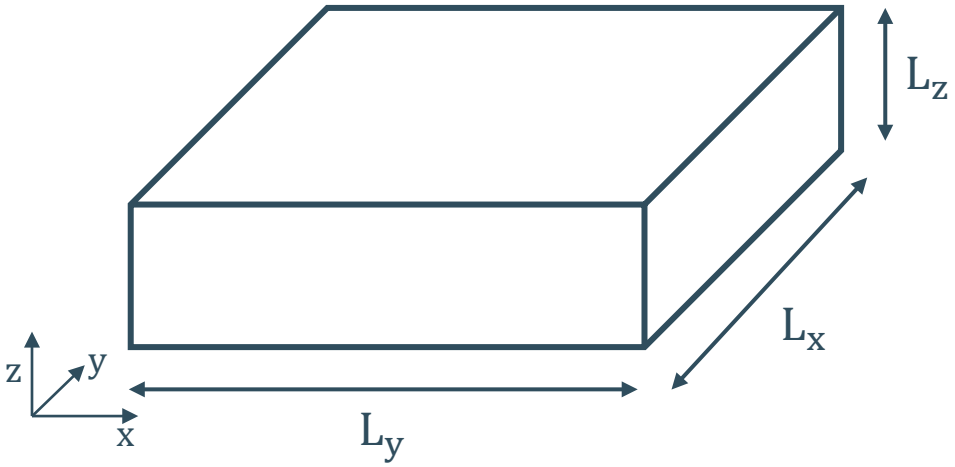
Input data

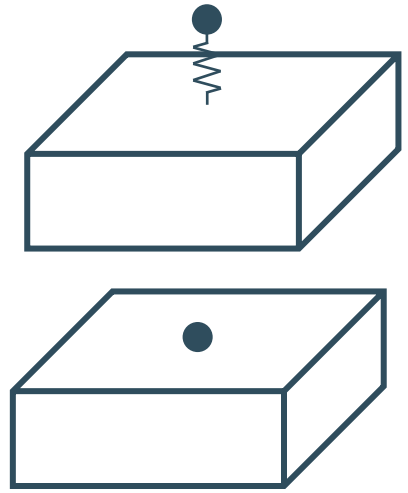


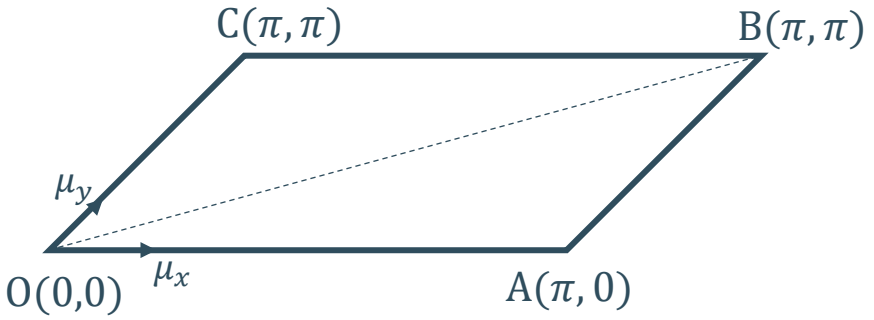


```
17  % Addition to UC plate
18  scatterer  = 'mass';          % 'none', 'resonator', 'mass'
19  f_res  = 2500;                % resonance of resonator [Hz]
20  m_ratio = 0.3;                % Scatterer mass ratio vs plate [-]
```

Input data



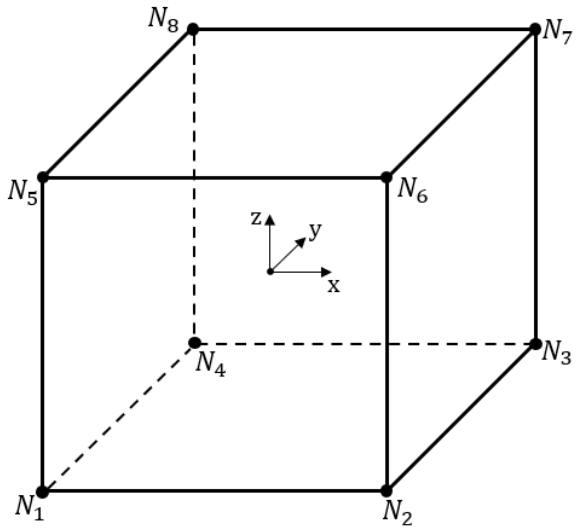


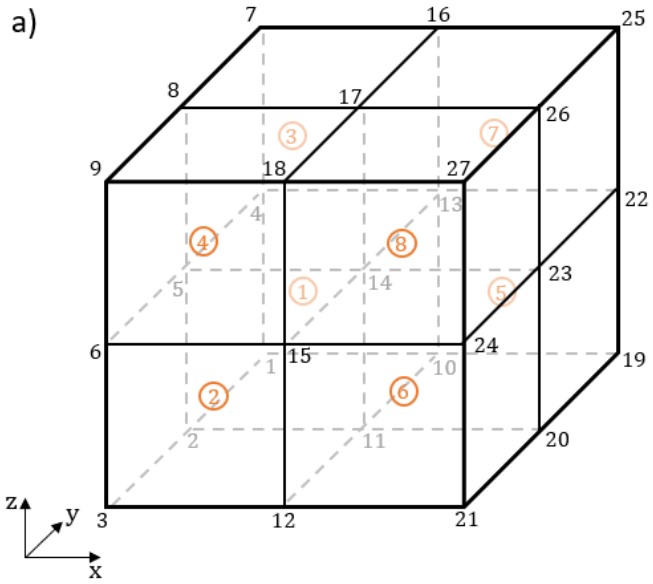


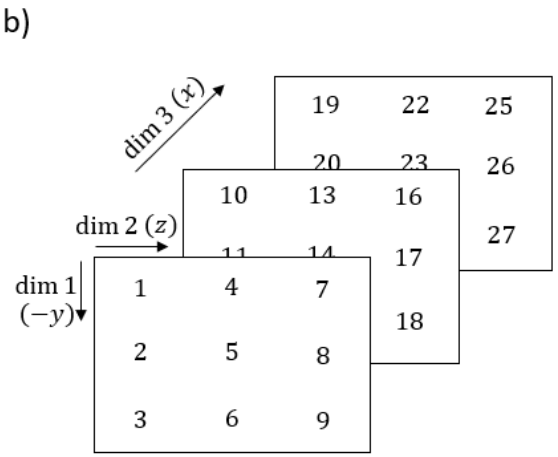

```
21  % Parameters for dispersion curves
22  cont_name = {'O','A','B','O'};          % Name of the IBC contour
23  cont_co =  [0,0;  pi,0;  pi,pi;   0,0];  % IBC definition
24  step_size  = 0.01*pi;                   % Resolution propagation constant
25  n_curves = 10;                          % Number of calculated wave modes
```

FE preprocessing

```
%% Prepare FEM analyis
% Element size
ax = Lx/nelx;
ay = Ly/nely;
az = Lz/nelz;
% Information UC
n_elem = nelx*nely*nelz;
n_nodes = (nelx+1)*(nely+1)*(nelz+1);
nDOF = 3;                          % Number of DOFs per node;
n_DOFs = nDOF*n_nodes;
mass_UC = rho*(nelx*ax)*(nely*ay)*(nelz*az);
% Element matrices
[KE,ME] = KM(E,v,rho,ax,ay,az);
% FEM information (hard-coded for nDOF = 3)
nodeNrs = reshape(1:n_nodes, 1+nely, 1+nelz, 1+nelx); % nodes numbering
cMat = reshape(nDOF * nodeNrs(1:nely, 1:nelz, 1:nelx)+1, n_elem, 1) + ...
    [0,1,2,3*( nely+1)*(nelz+1)+[0,1,2,-3,-2,-1],-3,-2,-1,3*(nely + ...
    1) +[0,1,2],3*( nely+1)*(nelz+2)+[0,1,2,-3,-2,-1],3*(nely+1)+[-3,-2,-1]]; % connectivity matrix DOFs
```



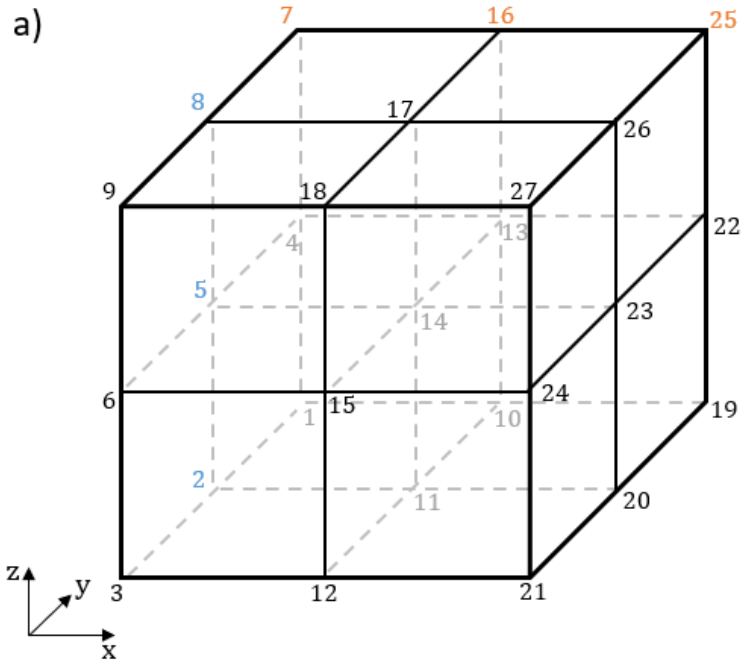




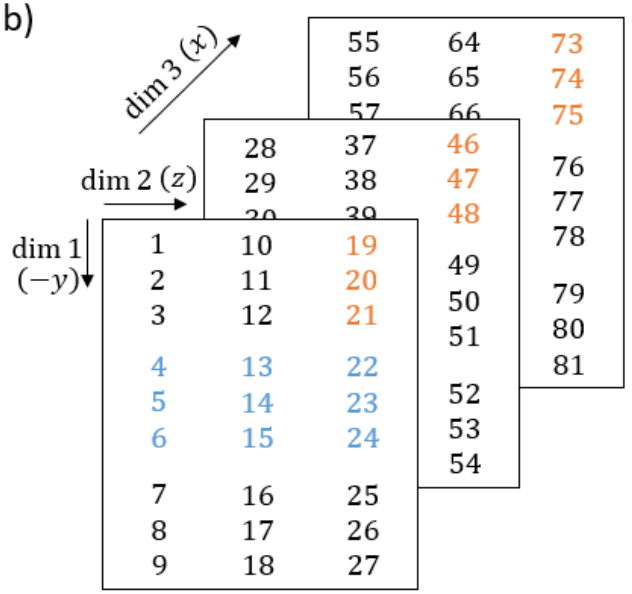
DOFs partitioning

```
46  %% DOFs partitioning
47  DofNrs = reshape(1:n_DOFs, nDOF*(n_ely+1), n_elz+1, n_elx+1);
48  dofs.L  = DofNrs(nDOF+1:end-nDOF,:,1); dofs.L = dofs.L(:);
49  dofs.R  = DofNrs(nDOF+1:end-nDOF,:,end); dofs.R = dofs.R(:);
50  dofs.T  = DofNrs(1:nDOF,:,2:end-1); dofs.T = dofs.T(:);
51  dofs.B  = DofNrs(end-(nDOF-1):end,:,2:end-1); dofs.B = dofs.B(:);
52  dofs.TL = DofNrs(1:nDOF,:,1); dofs.TL = dofs.TL(:);
53  dofs.TR = DofNrs(1:nDOF,:,end); dofs.TR = dofs.TR(:);
54  dofs.BL = DofNrs(end-(nDOF-1):end,:,1); dofs.BL = dofs.BL(:);
55  dofs.BR = DofNrs(end-(nDOF-1):end,:,end); dofs.BR = dofs.BR(:);
56  dofs.I  = DofNrs(nDOF+1:end-nDOF,:,2:end-1); dofs.I = dofs.I(:);
```

a)



b)



System matrix assembly

```
58  %% System matrix assembly
59  % Assemble bare plate system matrices
60  iL = reshape(kron(cMat,ones(24,1))',24*24*n_elem,1);
61  jL = reshape(kron(cMat,ones(1,24))',24*24*n_elem,1);
62  sK = reshape(KE(:)*(ones(n_elem,1)'),24*24*n_elem,1);
63  sM = reshape(ME(:)*(ones(n_elem,1)'),24*24*n_elem,1);
64  K = sparse(iL,jL,sK);
65  M = sparse(iL,jL,sM);
66  % Add scatterer (point mass/resonator)
67  switch scatterer
68      case 'mass'
69        mass = m_ratio*mass_UC;
70        dofM = 3*nodeNrs(floor(end/2),end,floor(end/2));
71        M(dofM,dofM) = M(dofM,dofM) + mass;
72      case 'resonator'
73        % Add extra zero row and column to K, M
74        K = [K zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
75        M = [M zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
76        % Define mass and stiffness
77        mass = m_ratio*mass_UC;
78        k_res = (f_res*2*pi)^2*mass;
79        dofK = 3*nodeNrs(floor(end/2),end,floor(end/2));
80        n_DOFs = n_DOFs+1;
81        dofM = n_DOFs;
82        % Add resonator matrices to bare plate UC system matrices
83        K([dofK dofM],[dofK dofM]) = K([dofK dofM],[dofK dofM]) + [k_res -k_res; -k_res k_res];
84        M([dofK dofM],[dofK dofM]) = M([dofK dofM],[dofK dofM]) + [0 0; 0 mass];
85        dofs.I = [dofs.I; dofM];
86  end
```

System matrix assembly

```
58  %% System matrix assembly
59  % Assemble bare plate system matrices
60  iL = reshape(kron(cMat,ones(24,1))',24*24*n_elem,1);
61  jL = reshape(kron(cMat,ones(1,24))',24*24*n_elem,1);
62  sK = reshape(KE(:)*(ones(n_elem,1)'),24*24*n_elem,1);
63  sM = reshape(ME(:)*(ones(n_elem,1)'),24*24*n_elem,1);
64  K = sparse(iL,jL,sK);
65  M = sparse(iL,jL,sM);
66  % Add scatterer (point mass/resonator)
67  switch scatterer
68      case 'mass'
69        mass = m_ratio*mass_UC;
70        dofM = 3*nodeNrs(floor(end/2),end,floor(end/2));
71        M(dofM,dofM) = M(dofM,dofM) + mass;
72      case 'resonator'
73        % Add extra zero row and column to K, M
74        K = [K zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
75        M = [M zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
76        % Define mass and stiffness
77        mass = m_ratio*mass_UC;
78        k_res = (f_res*2*pi)^2*mass;
79        dofK = 3*nodeNrs(floor(end/2),end,floor(end/2));
80        n_DOFs = n_DOFs+1;
81        dofM = n_DOFs;
82        % Add resonator matrices to bare plate UC system matrices
83        K([dofK dofM],[dofK dofM]) = K([dofK dofM],[dofK dofM]) + [k_res -k_res; -k_res k_res];
84        M([dofK dofM],[dofK dofM]) = M([dofK dofM],[dofK dofM]) + [0 0; 0 mass];
85        dofs.I = [dofs.I; dofM];
86  end
```

System matrices of the bare plate

System matrix assembly

```
58  %% System matrix assembly
59  % Assemble bare plate system matrices
60  iL = reshape(kron(cMat,ones(24,1))',24*24*n_elem,1);
61  jL = reshape(kron(cMat,ones(1,24))',24*24*n_elem,1);
62  sK = reshape(KE(:)*(ones(n_elem,1)'),24*24*n_elem,1);
63  sM = reshape(ME(:)*(ones(n_elem,1)'),24*24*n_elem,1);
64  K = sparse(iL,jL,sK);
65  M = sparse(iL,jL,sM);
66  % Add scatterer (point mass/resonator)
67  switch scatterer
68      case 'mass'
69        mass = m_ratio*mass_UC;
70        dofM = 3*nodeNrs(floor(end/2),end,floor(end/2));
71        M(dofM,dofM) = M(dofM,dofM) + mass;
72      case 'resonator'
73        % Add extra zero row and column to K, M
74        K = [K zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
75        M = [M zeros(n_DOFs,1);zeros(1,n_DOFs) 0];
76        % Define mass and stiffness
77        mass = m_ratio*mass_UC;
78        k_res = (f_res*2*pi)^2*mass;
79        dofK = 3*nodeNrs(floor(end/2),end,floor(end/2));
80        n_DOFs = n_DOFs+1;
81        dofM = n_DOFs;
82        % Add resonator matrices to bare plate UC system matrices
83        K([dofK dofM],[dofK dofM]) = K([dofK dofM],[dofK dofM]) + [k_res -k_res; -k_res k_res];
84        M([dofK dofM],[dofK dofM]) = M([dofK dofM],[dofK dofM]) + [0 0; 0 mass];
85        dofs.I = [dofs.I; dofM];
86  end
```

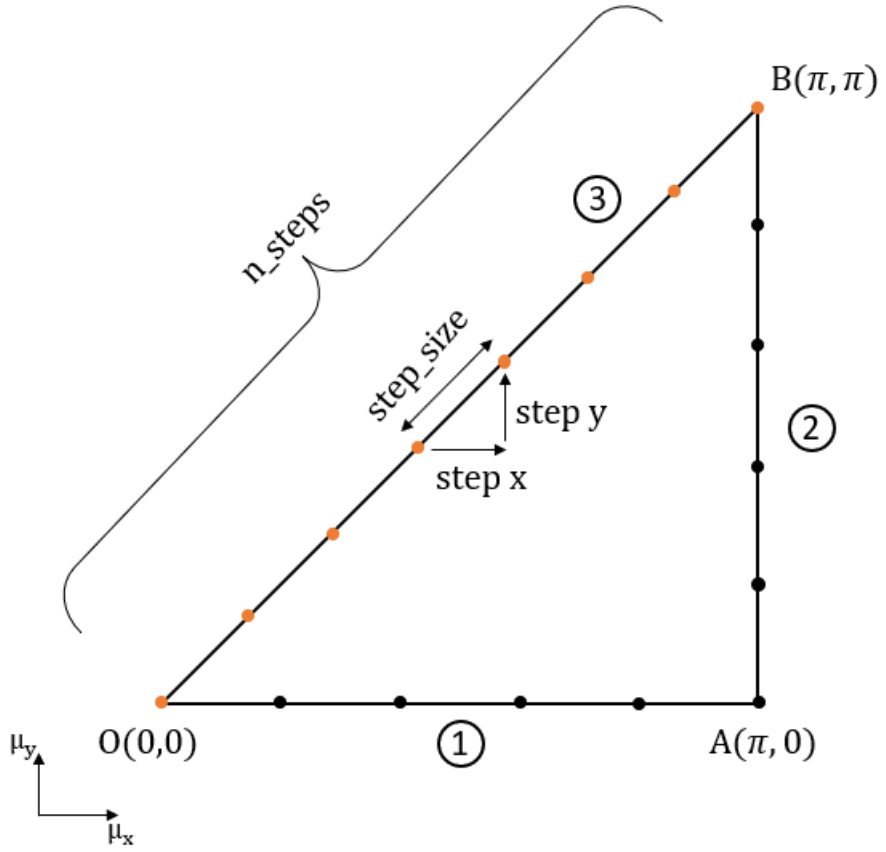
System matrices of the bare plate

Addition of the point mass or the resonator

Sampling the IBC

All (μ_x, μ_y)-pairs of interest need be defined

```
88  %% Sampling the IBC
89  mu = 1i*cont_co(1,:).';
90  tot_steps = zeros(1,size(cont_co,1));
91  for i = 1 : size(cont_co,1)-1
92      n_steps = ceil((sqrt((cont_co(i,1)-cont_co(i+1,1))^2+(cont_co(i,2)-cont_co(i+1,2))^2))/(st
93      ed = zeros(2,n_steps);
94      for j = 1:2 % j = 1 -> x; j = 2 -> y
95          step = (cont_co(i+1,j)-cont_co(i,j))/(n_steps);
96          if step == 0
97              ed(j,:) = cont_co(i,j)*ones(1,n_steps);
98          else
99              ed(j,:) = (cont_co(i,j)+step):step:cont_co(i+1,j);
100         end
101     end
102     mu = [mu, 1i*ed];
103     tot_steps(i+1) = tot_steps(i)+n_steps;
104 end
```



Dispersion curve calculation

```
106  %% Dispersion curve calculation
107  omega = zeros(n_curves,size(mu,2));
108  parfor i = 1:size(mu,2)
109      % Construction of periodicity matrix
110      R = eye(n_DOFs,n_DOFs);
111      R(dofs.R,dofs.L) = exp(mu(1,i))*eye(length(dofs.R),length(dofs.L));
112      R(dofs.T,dofs.B) = exp(mu(2,i))*eye(length(dofs.T),length(dofs.B));
113      R(dofs.TL,dofs.BL) = exp(mu(2,i))*eye(length(dofs.TL),length(dofs.BL));
114      R(dofs.TR,dofs.BL) = exp(mu(1,i)+mu(2,i))*eye(length(dofs.TR),length(dofs.BL));
115      R(dofs.BR,dofs.BL) = exp(mu(1,i))*eye(length(dofs.BR),length(dofs.BL));
116      R = sparse(R(:,setdiff(1:n_DOFs,[dofs.R;dofs.T;dofs.TL;dofs.BR;dofs.TR])));
117      % Impose periodicity boundary conditions
118      K_BF = R'*K*R;
119      M_BF = R'*M*R;
120      % Compute dispersion curves
121      [~,s] = eigs(K_BF,M_BF,n_curves,0);
122      [s,~] = sort(diag(s));
123      omega(:,i) = sqrt(s);
124  end
```

Dispersion curve calculation

```
106  %% Dispersion curve calculation
107  omega = zeros(n_curves,size(mu,2));
108  parfor i = 1:size(mu,2)
109      % Construction of periodicity matrix
110      R = eye(n_DOFs,n_DOFs);
111      R(dofs.R,dofs.L) = exp(mu(1,i))*eye(length(dofs.R),length(dofs.L));
112      R(dofs.T,dofs.B) = exp(mu(2,i))*eye(length(dofs.T),length(dofs.B));
113      R(dofs.TL,dofs.BL) = exp(mu(2,i))*eye(length(dofs.TL),length(dofs.BL));
114      R(dofs.TR,dofs.BL) = exp(mu(1,i)+mu(2,i))*eye(length(dofs.TR),length(dofs.BL));
115      R(dofs.BR,dofs.BL) = exp(mu(1,i))*eye(length(dofs.BR),length(dofs.BL));
116      R = sparse(R(:,setdiff(1:n_DOFs,[dofs.R;dofs.T;dofs.TL;dofs.BR;dofs.TR])));
117      % Impose periodicity boundary conditions
118      K_BF = R'*K*R;
119      M_BF = R'*M*R;
120      % Compute dispersion curves
121      [~,s] = eigs(K_BF,M_BF,n_curves,0);
122      [s,~] = sort(diag(s));
123      omega(:,i) = sqrt(s);
124  end
```

Computation of periodicity matrix $\boldsymbol{R}$

Dispersion curve calculation

```
106  %% Dispersion curve calculation
107  omega = zeros(n_curves,size(mu,2));
108  parfor i = 1:size(mu,2)
109      % Construction of periodicity matrix
110      R = eye(n_DOFs,n_DOFs);
111      R(dofs.R,dofs.L) = exp(mu(1,i))*eye(length(dofs.R),length(dofs.L));
112      R(dofs.T,dofs.B) = exp(mu(2,i))*eye(length(dofs.T),length(dofs.B));
113      R(dofs.TL,dofs.BL) = exp(mu(2,i))*eye(length(dofs.TL),length(dofs.BL));
114      R(dofs.TR,dofs.BL) = exp(mu(1,i)+mu(2,i))*eye(length(dofs.TR),length(dofs.BL));
115      R(dofs.BR,dofs.BL) = exp(mu(1,i))*eye(length(dofs.BR),length(dofs.BL));
116      R = sparse(R(:,setdiff(1:n_DOFs,[dofs.R;dofs.T;dofs.TL;dofs.BR;dofs.TR])));
117      % Impose periodicity boundary conditions
118      K_BF = R'*K*R;
119      M_BF = R'*M*R;
120      % Compute dispersion curves
121      [~,s] = eigs(K_BF,M_BF,n_curves,0);
122      [s,~] = sort(diag(s));
123      omega(:,i) = sqrt(s);
124  end
```

Computation of periodicity matrix $\boldsymbol{R}$

Imposing the periodicity conditions on mass and stiffness matrix

Dispersion curve calculation

```
106  %% Dispersion curve calculation
107  omega = zeros(n_curves,size(mu,2));
108  parfor i = 1:size(mu,2)
109      % Construction of periodicity matrix
110      R = eye(n_DOFs,n_DOFs);
111      R(dofs.R,dofs.L) = exp(mu(1,i))*eye(length(dofs.R),length(dofs.L));
112      R(dofs.T,dofs.B) = exp(mu(2,i))*eye(length(dofs.T),length(dofs.B));
113      R(dofs.TL,dofs.BL) = exp(mu(2,i))*eye(length(dofs.TL),length(dofs.BL));
114      R(dofs.TR,dofs.BL) = exp(mu(1,i)+mu(2,i))*eye(length(dofs.TR),length(dofs.BL));
115      R(dofs.BR,dofs.BL) = exp(mu(1,i))*eye(length(dofs.BR),length(dofs.BL));
116      R = sparse(R(:,setdiff(1:n_DOFs,[dofs.R;dofs.T;dofs.TL;dofs.BR;dofs.TR])));
117      % Impose periodicity boundary conditions
118      K_BF = R'*K*R;
119      M_BF = R'*M*R;
120      % Compute dispersion curves
121      [~,s] = eigs(K_BF,M_BF,n_curves,0);
122      [s,~] = sort(diag(s));
123      omega(:,i) = sqrt(s);
124  end
```

Computation of periodicity matrix $\boldsymbol{R}$

Imposing the periodicity conditions on mass and stiffness matrix

Computes the n_curves eigenvalues closest to zero

Results

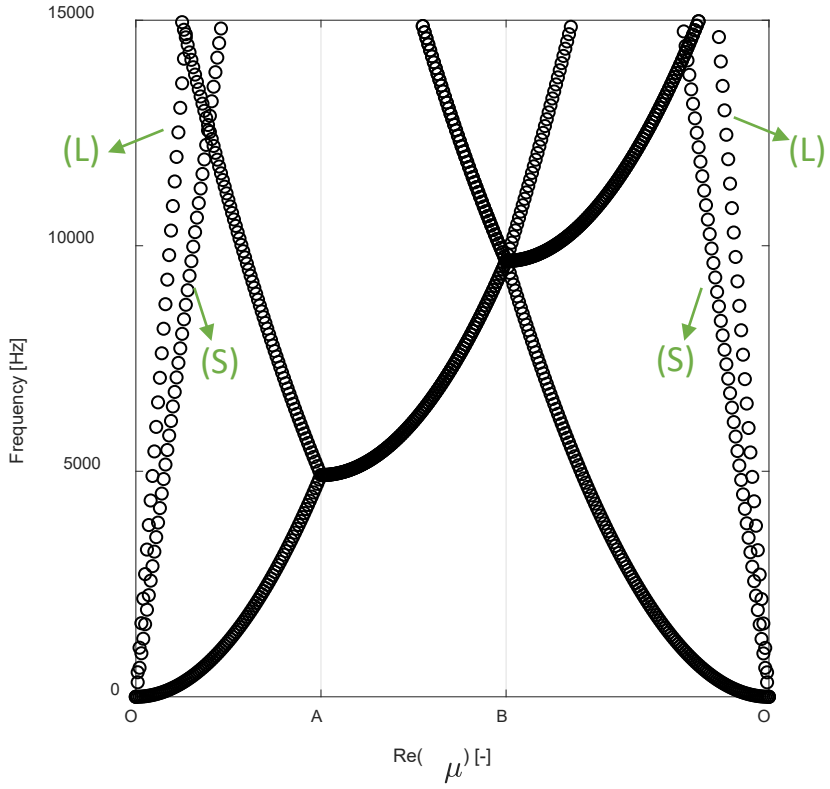
UC size [m]	Material	Discretization	IBC definition
$L_x = 0.05$	$E = 210e9$ GPa	$nelx = 10$	cont_name = O,A,B,O
$L_y = 0.05$	$\nu = 0.3$	$nely = 10$	contco = [0,0; pi,0; pi,pi; 0,0]
$L_z = 0.005$	$rho = 7800$ kg/m^3	$nelz = 3$	step_size = 0.01π
		n_dof = 3	n_curves= 10

f_res = 2500

Results

UC size [m]	Material	Discretization	IBC definition
$L_x = 0.05$	$E = 210e9$ GPa	$nelx = 10$	cont_name = O,A,B,O
$L_y = 0.05$	$\nu = 0.3$	$nely = 10$	contco = [0,0; pi,0; pi,pi; 0,0]
$L_z = 0.005$	$rho = 7800$ kg/m^3	$nelz = 3$	step_size = 0.01π
		n_dof = 3	n_curves= 10

a)

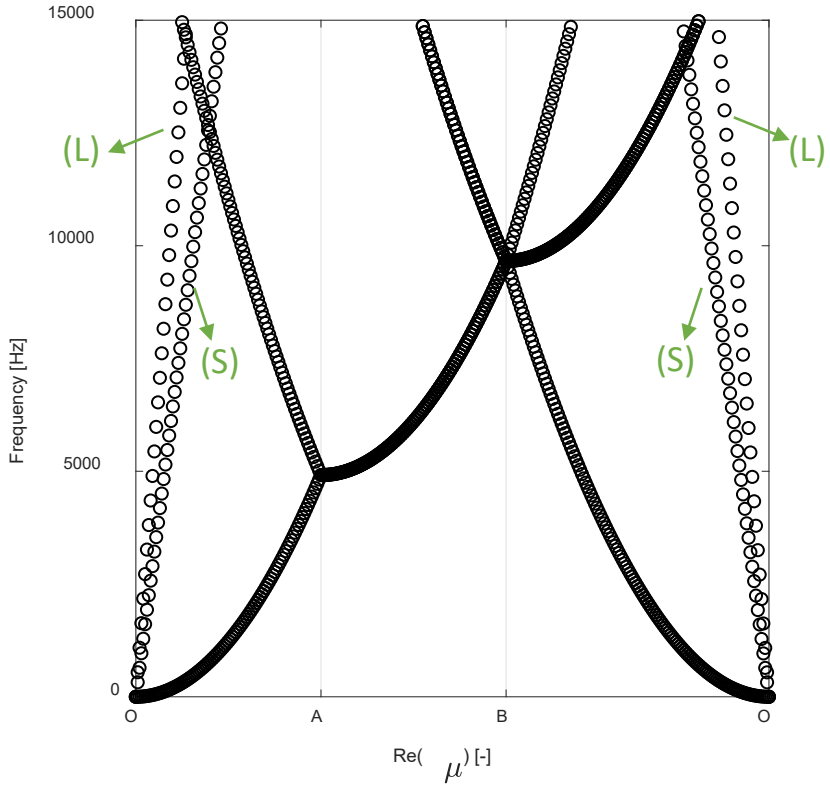


f_res = 2500

Results

UC size [m]	Material	Discretization	IBC definition
$L_x = 0.05$	$E = 210e9$ GPa	$nelx = 10$	cont_name = O,A,B,O
$L_y = 0.05$	$\nu = 0.3$	$nely = 10$	contco = [0,0; pi,0; pi,pi; 0,0]
$L_z = 0.005$	$rho = 7800$ kg/m^3	$nelz = 3$	step_size = 0.01π
		n_dof = 3	n_curves= 10

a)

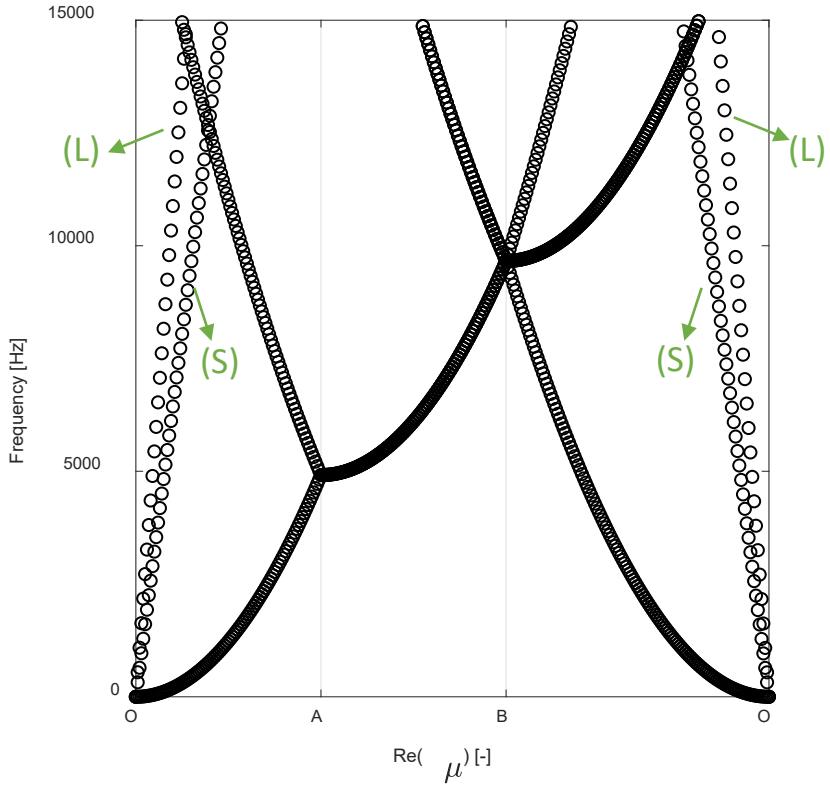


$$f_{\lambda/2} = \frac{\pi}{2L^2}\sqrt{\frac{t^2 E}{12(1-\nu^2)\rho}}$$

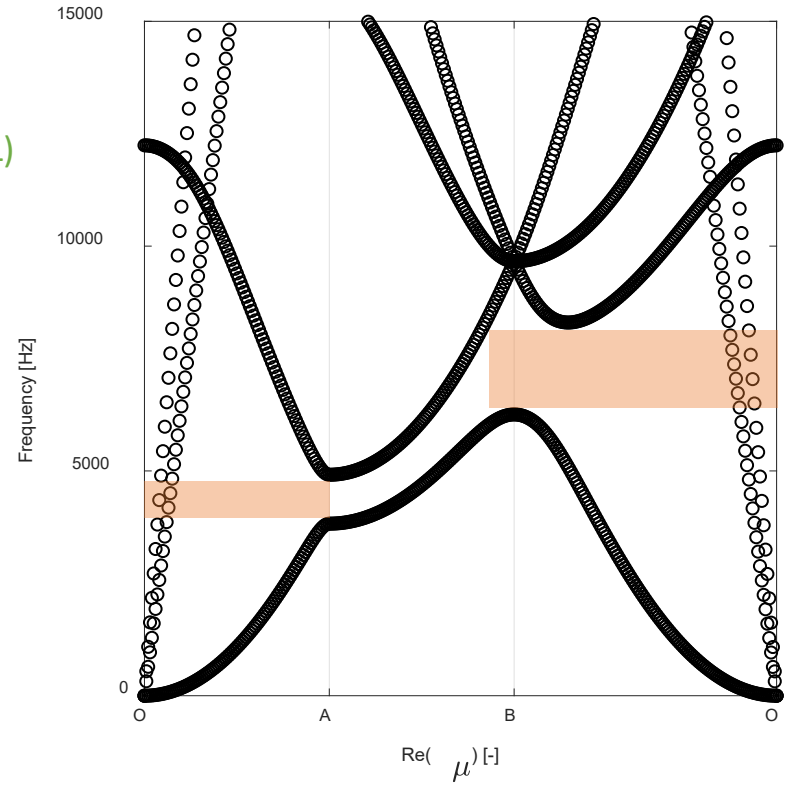
Results

UC size [m]	Material	Discretization	IBC definition
$L_x = 0.05$	$E = 210e9$ GPa	$nelx = 10$	cont_name = O,A,B,O
$L_y = 0.05$	$\nu = 0.3$	$nely = 10$	contco = [0,0; pi,0; pi,pi; 0,0]
$L_z = 0.005$	$rho = 7800$ kg/m^3	$nelz = 3$	step_size = 0.01π
		n_dof = 3	n_curves= 10

a)



b



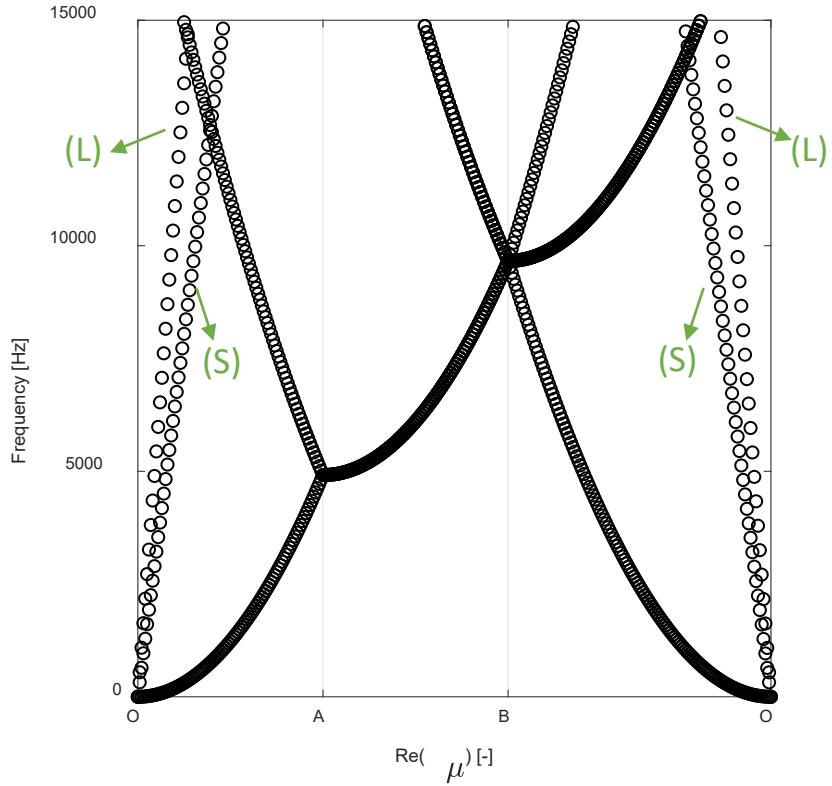
$$f_{\lambda/2} = \frac{\pi}{2L^2}\sqrt{\frac{t^2 E}{12(1-\nu^2)\rho}}$$

m_ratio = 0.3

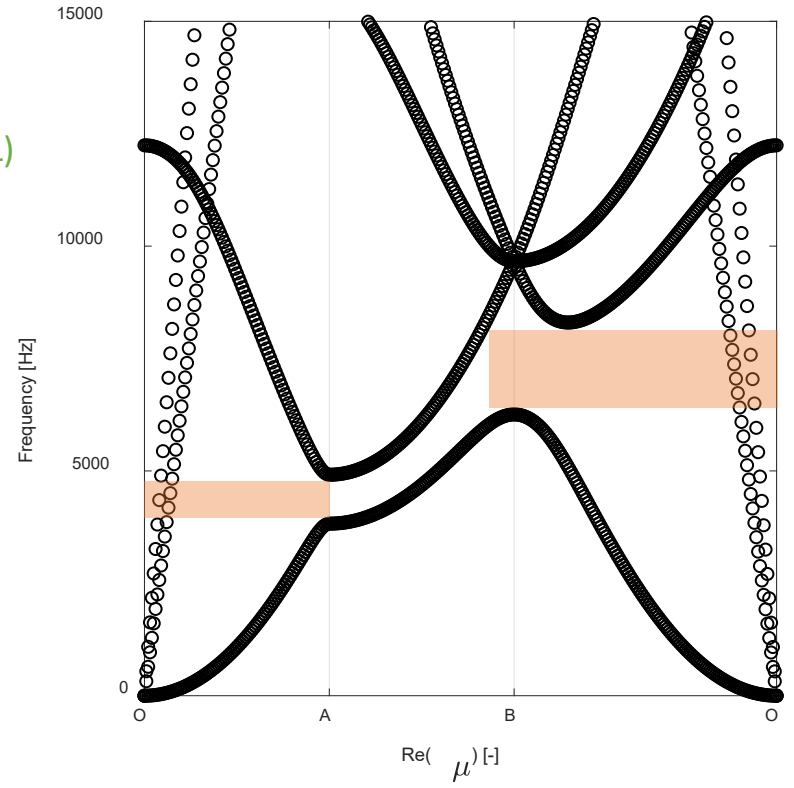
Results

UC size [m]	Material	Discretization	IBC definition
$L_x = 0.05$	$E = 210e9$ GPa	$nelx = 10$	cont_name = O,A,B,O
$L_y = 0.05$	$\nu = 0.3$	$nely = 10$	contco = [0,0; pi,0; pi,pi; 0,0]
$L_z = 0.005$	$rho = 7800$ kg/m^3	$nelz = 3$	step_size = 0.01π
		n_dof = 3	n_curves= 10

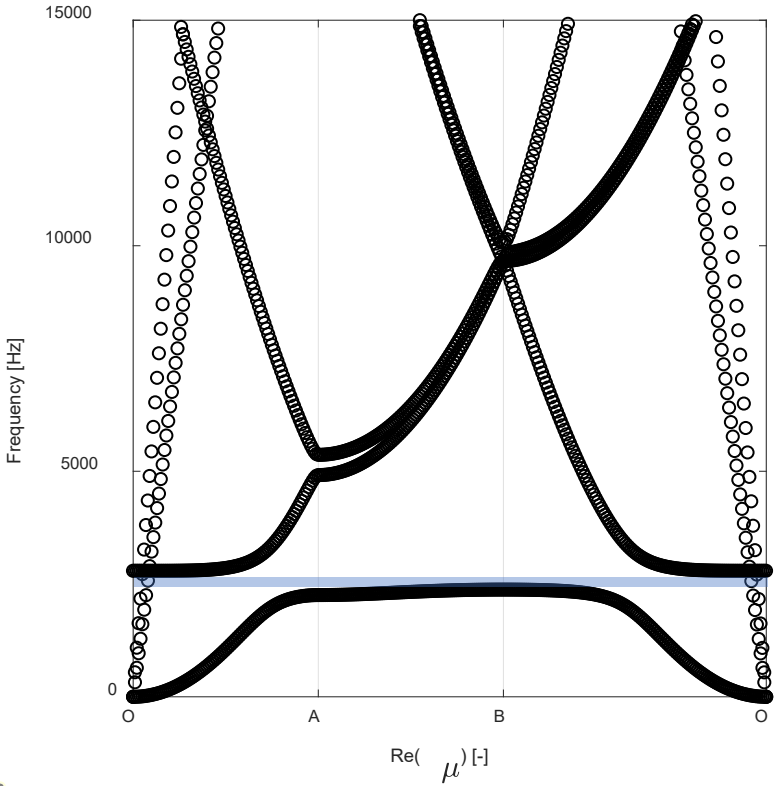
a)



b



c)



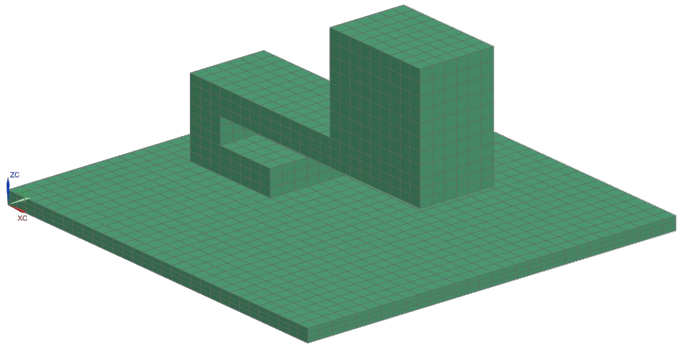
$$f_{\lambda/2} = \frac{\pi}{2L^2}\sqrt{\frac{t^2 E}{12(1-\nu^2)\rho}}$$

m_ratio = 0.3

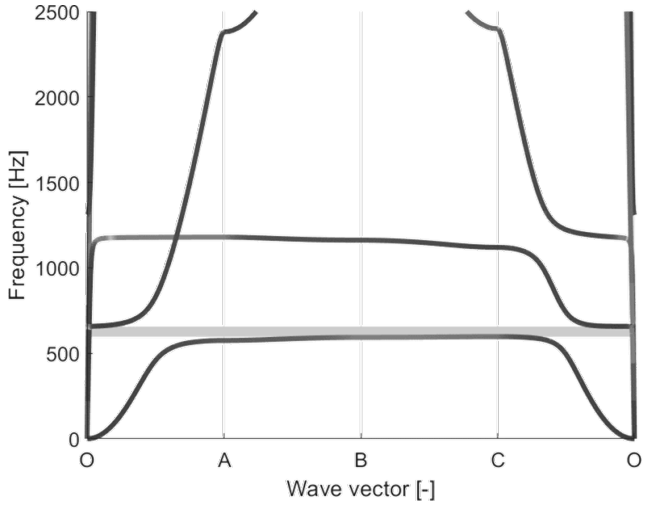
m_ratio = 0.3

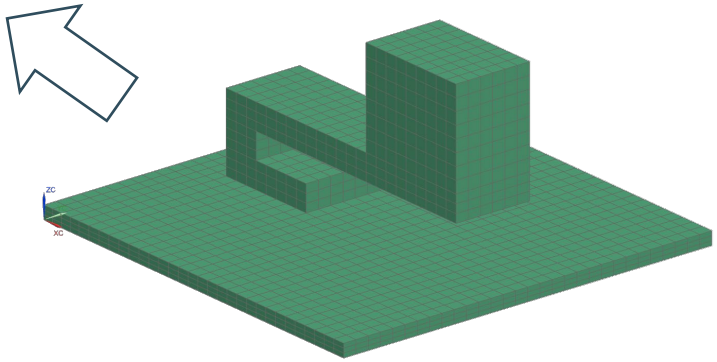
f_res = 2500

What else?

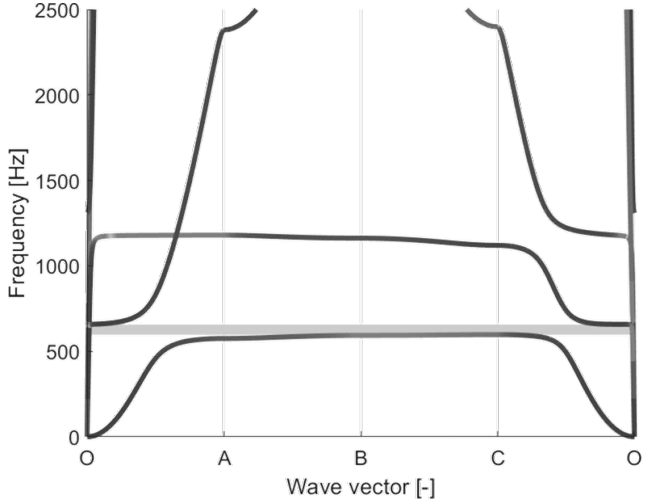

ZC
XC

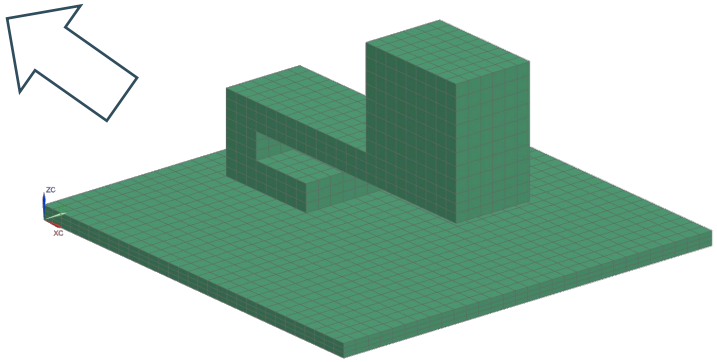
a) (fast) dispersion curve computations

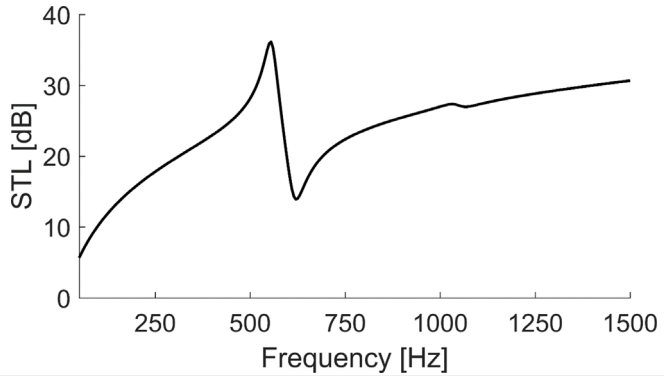




a) (fast) dispersion curve computations

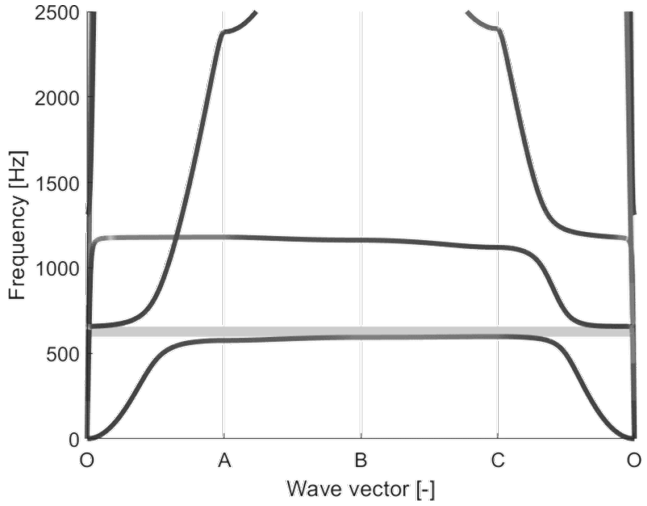




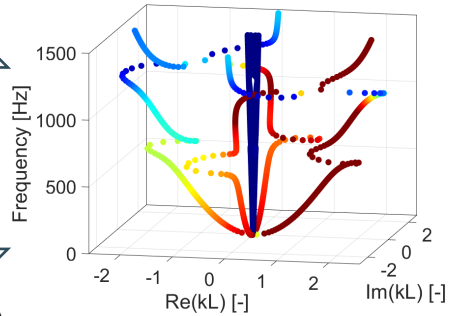


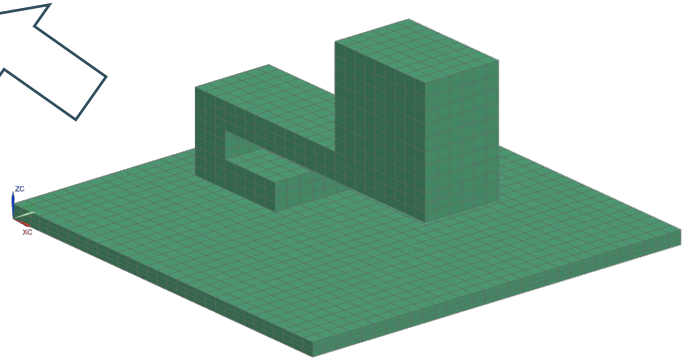
b) infinite structure STL predictions

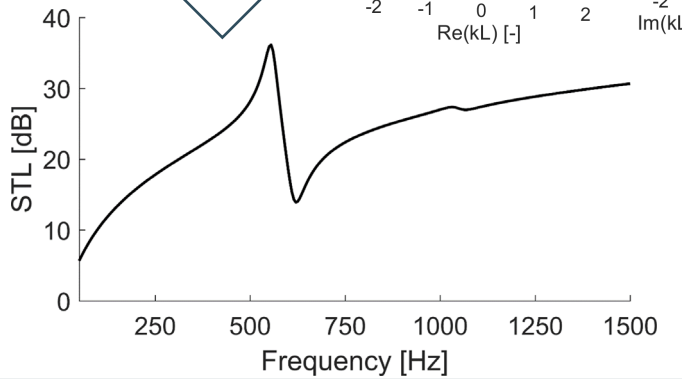
a) (fast) dispersion curve computations



c) wave mode contributions

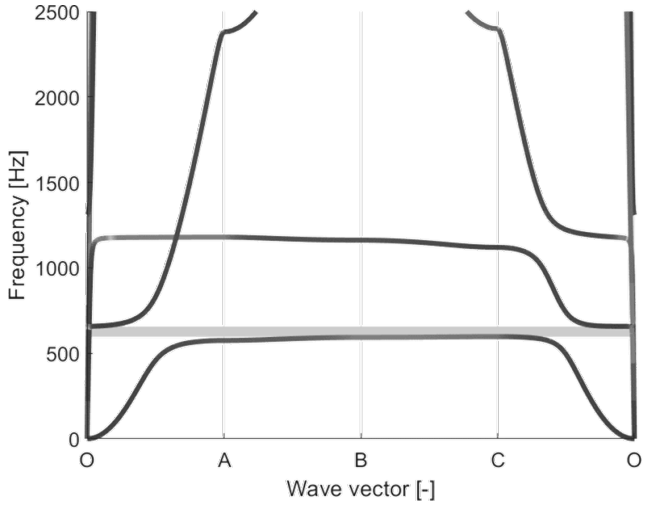




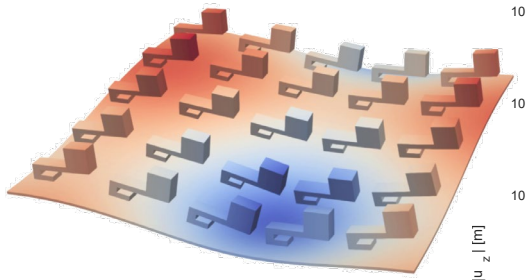


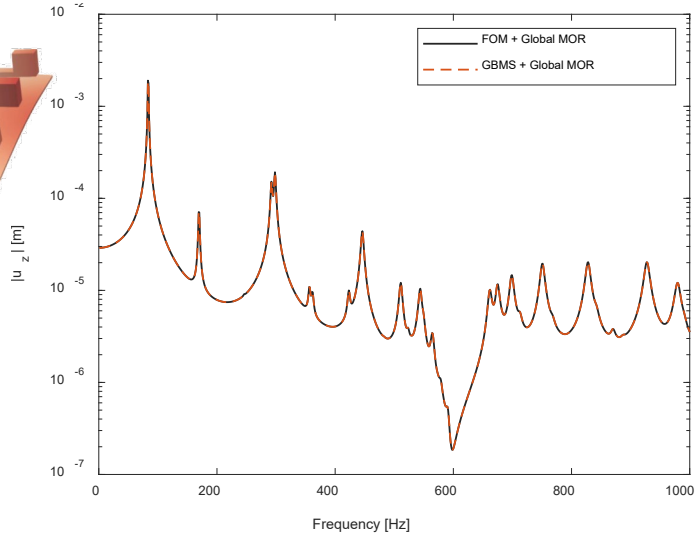
b) infinite structure STL predictions

a) (fast) dispersion curve computations

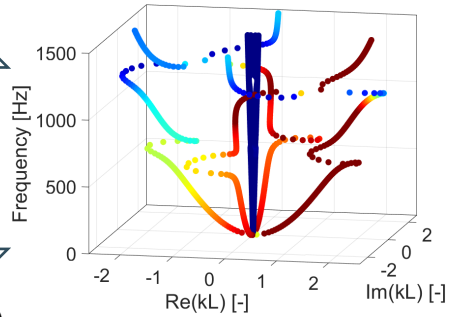


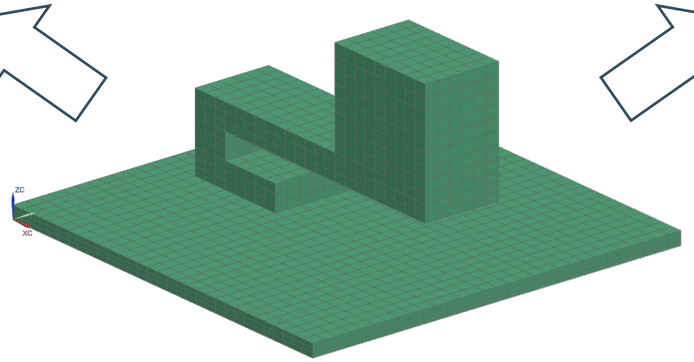
d) finite structure response predictions

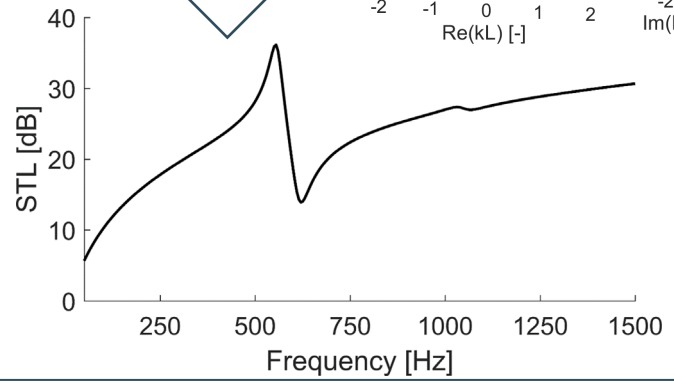




c) wave mode contributions

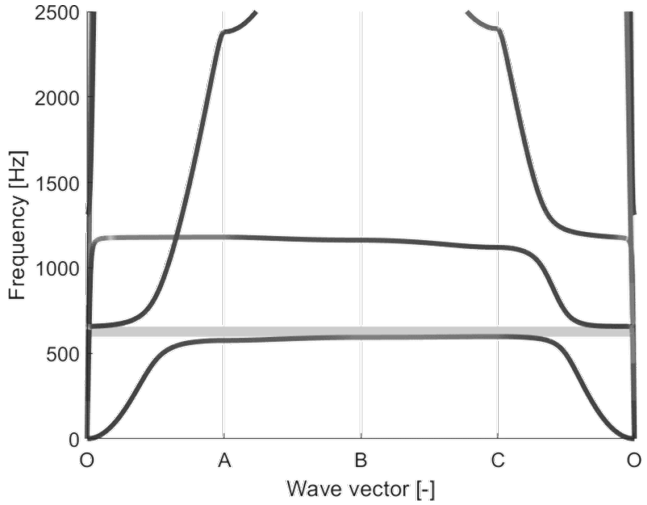




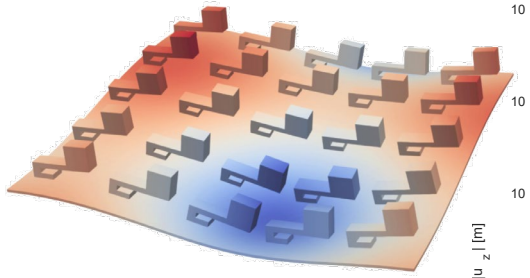


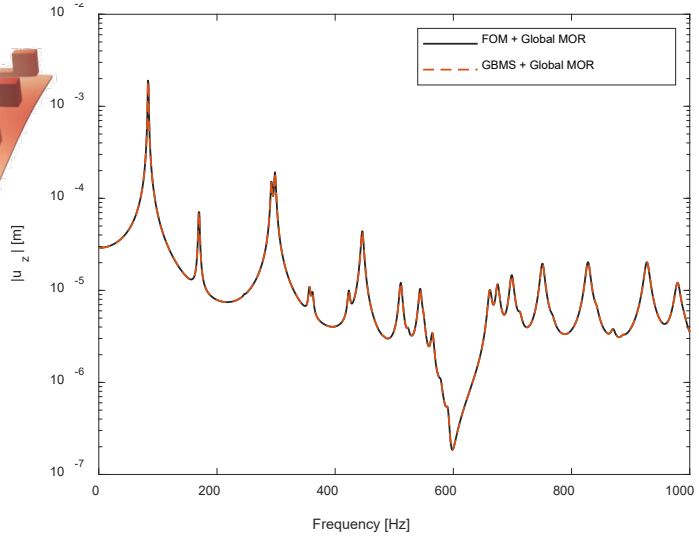
b) infinite structure STL predictions

a) (fast) dispersion curve computations

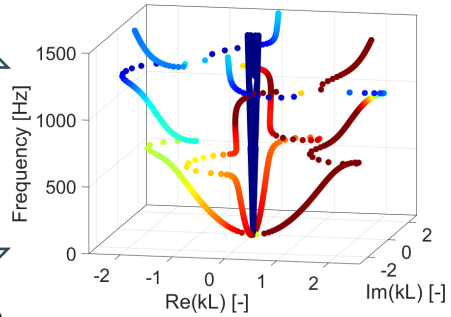


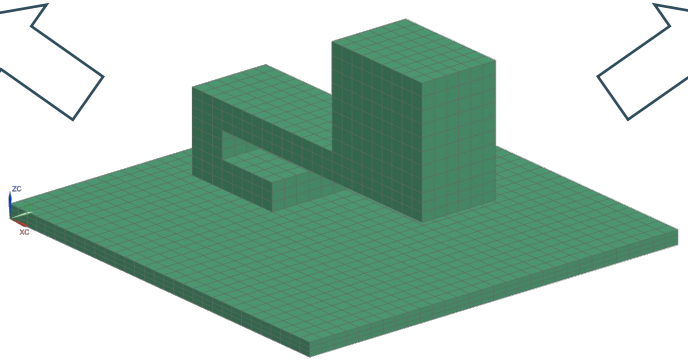
d) finite structure response predictions

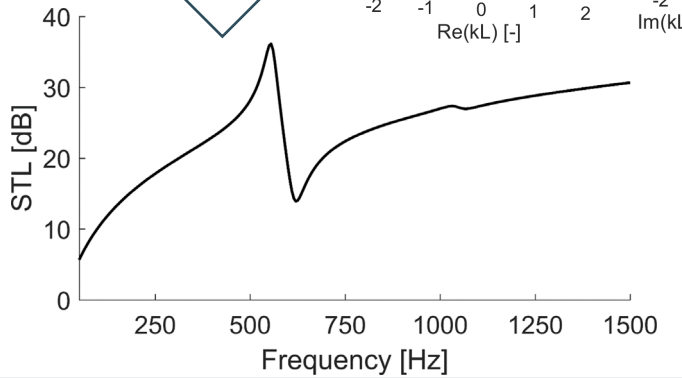




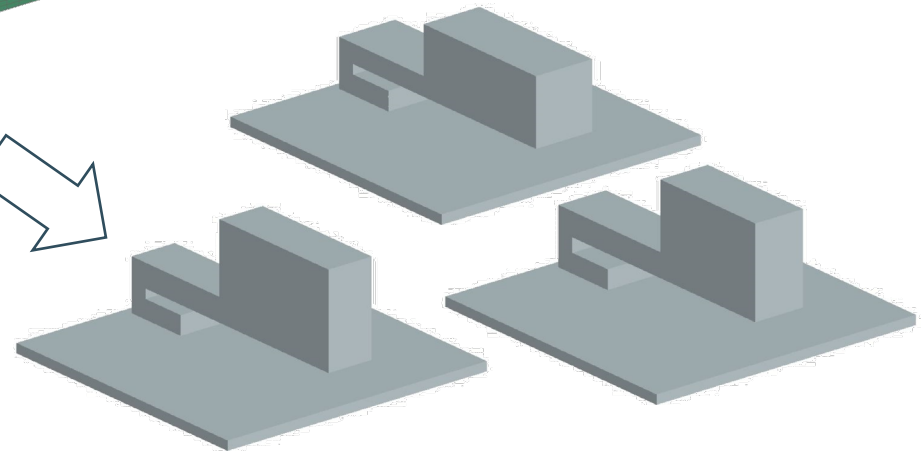
c) wave mode contributions







b) infinite structure STL predictions



e) design/topology optimization

elke.deckers@kuleuven.be

www.mech.kuleuven.be/lmsd

VACANCIES www.mech.kuleuven.be/lmsd-joboffers

lmsd_kuleuven

lmsd-kuleuven

lmsd.kuleuven

lmsd-kuleuven

lmsd_kuleuven

KU Leuven Mecha(tro)nic System Dynamics (LMSD)