CIVE 546 Structural Design Optimization

(3 units)

KKT Conditions w/o Slack
Convexity
GRAND

Instructor: Prof. Yi Shao

Winter 2025

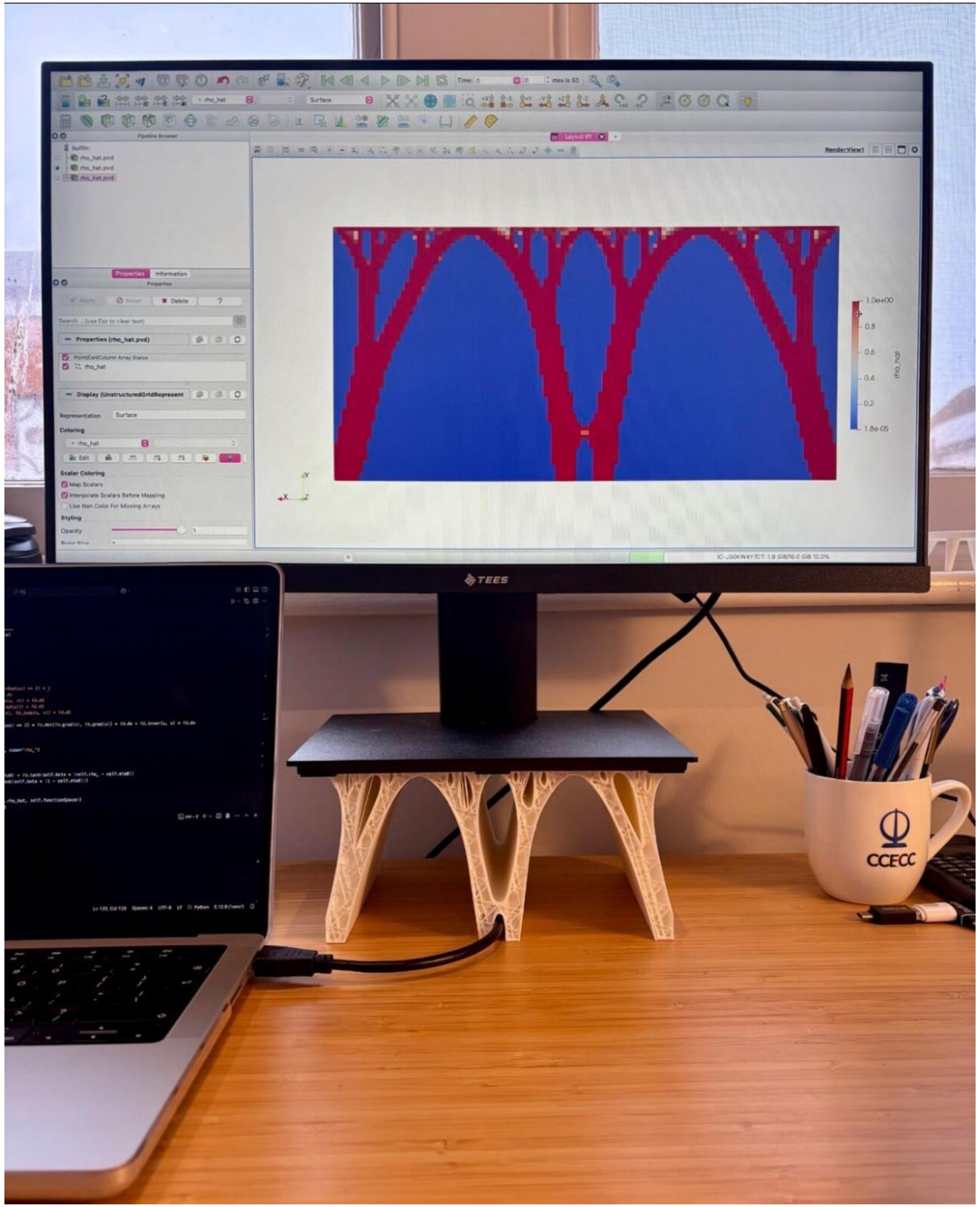
Administrative announcement

Reminder:

- Path 1 should be completed individually
- Path 2&3 needs preapproval by Feb 10

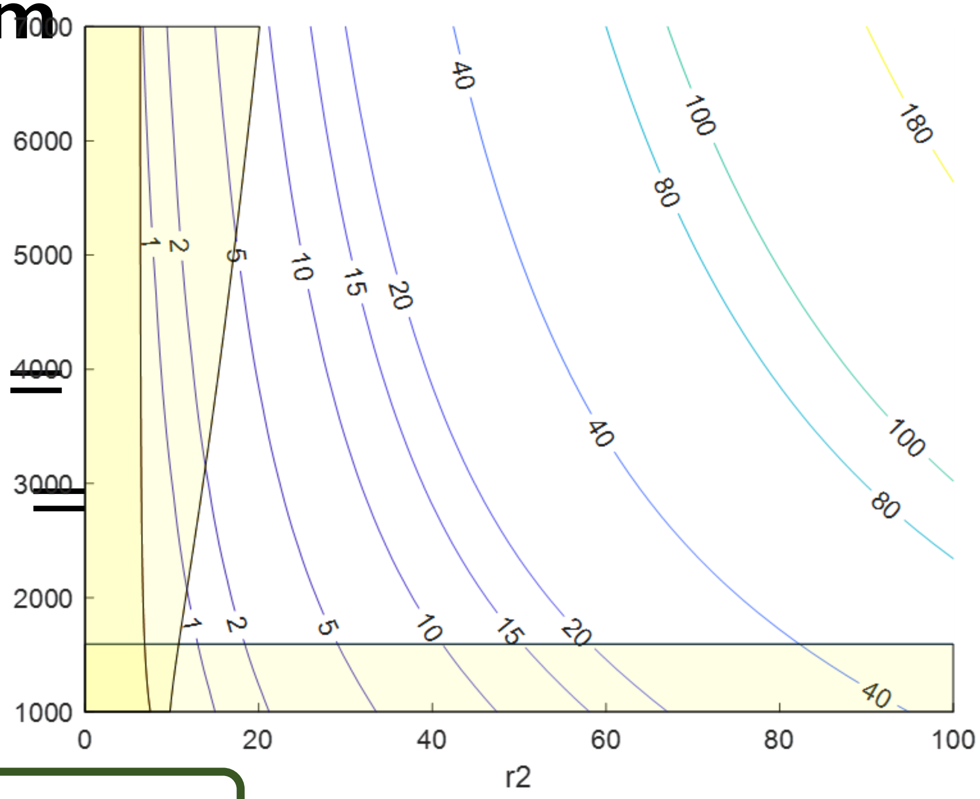
Homework:

Read GRAND paper and play with the software



General form of an optimization problem

Objective function	$\min f(\underline{x})$
	Subject to:
Inequality constraints	$g_j(\underline{x}) \leq 0 \qquad j =$
Equality constraints	$h_k(\underline{x}) = 0 \qquad k =$
Box constraints	$x_i^L \leq x_i \leq x_i^U$



Optimization Method

Optimality Condition Method

Search Method

Karush–Kuhn–Tucker (KKT) optimality conditions

Problem: minimize $f(\boldsymbol{x})$, where the design variable vector $\boldsymbol{x} = (x_1, \cdots, x_n)$, subjected to $(s.t.)$ $h_i(\boldsymbol{x}) = 0$, $i = 1 \cdots m$; $g_j(\boldsymbol{x}) \leq 0$, $j = 1 \cdots p$.

Let $\boldsymbol{x}^*$ be a regular point of the feasible set that is a local min for $f(\boldsymbol{x})$, subjected to the above constraints. Then there exist LMs $\boldsymbol{\lambda}^*$ ($m + p$ vector) such that the Lagrangian function is stationary wrt x_j, λ_j and s_j at the point $\boldsymbol{x}^*$.

KKT 1) Lagrangian function for the problem written in standard form

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^{p} \lambda_j \left(g_j(\boldsymbol{x}) + s_j^2\right)$$
$$= f(\boldsymbol{x}) + \boldsymbol{\lambda}_E^T h(\boldsymbol{x}) + \boldsymbol{\lambda}_I^T (g(\boldsymbol{x}) + \boldsymbol{s}^2)$$

KKT 2) Gradient conditions

$$\frac{\partial L}{\partial x_k} = \frac{\partial f}{\partial x_k} + \sum_{i=1}^{m} \lambda_i^* \frac{\partial h_i}{\partial x_k} + \sum_{j=1}^{p} \lambda_j^* \frac{\partial g_j}{\partial x_k} = 0, k = 1 \cdots n.$$
$$\frac{\partial L}{\partial \lambda_i} = 0 \ \Rightarrow h_i(\boldsymbol{x}^*) = 0; i = 1 \cdots m.$$
$$\frac{\partial L}{\partial \lambda_j} = 0 \ \Rightarrow \left(g_j(\boldsymbol{x}^*) + s_j^2\right) = 0; j = 1 \cdots p.$$

Karush–Kuhn–Tucker (KKT) optimality conditions

KKT 3) Feasibility check for inequalities

$$s_j^2 \geq 0; \text{ or equivalently } g_j \leq 0; j = 1 \cdots p.$$

KKT 4) Switching conditions

$$\frac{\partial L}{\partial s_j} = 0 \ \Rightarrow \lambda_j^* s_j = 0; j = 1 \cdots p.$$

KKT 5) Non-negativity of LMs for inequalities

$$\lambda_j^* \geq 0; \ j = 1 \cdots p.$$

KKT 6) Regularity check

Gradients of active constraints must be linearly independent. In such case, the LMs for the constraints are unique.

Karush–Kuhn–Tucker (KKT) optimality conditions
Remarks

For a given problem, the KKT conditions can be used to find candidate minimum points. Several cases defined by the switching conditions must be considered and solved. Each case can provide multiple solutions.

For each solution, remember to

i. Check all inequality constraints for feasibility

ii. Calculate all the Lagrange Multipliers

iii. Ensure that the Lagrange multipliers for all the inequality constraints are non-negative

Karush–Kuhn–Tucker (KKT) optimality conditions without slack

Rewrite KKT conditions and remove slack variables

Karush–Kuhn–Tucker (KKT) optimality conditions without slack

Example

$$\min f(x, y) = (x - 10)^2 + (y - 8)^2$$

Subject to:

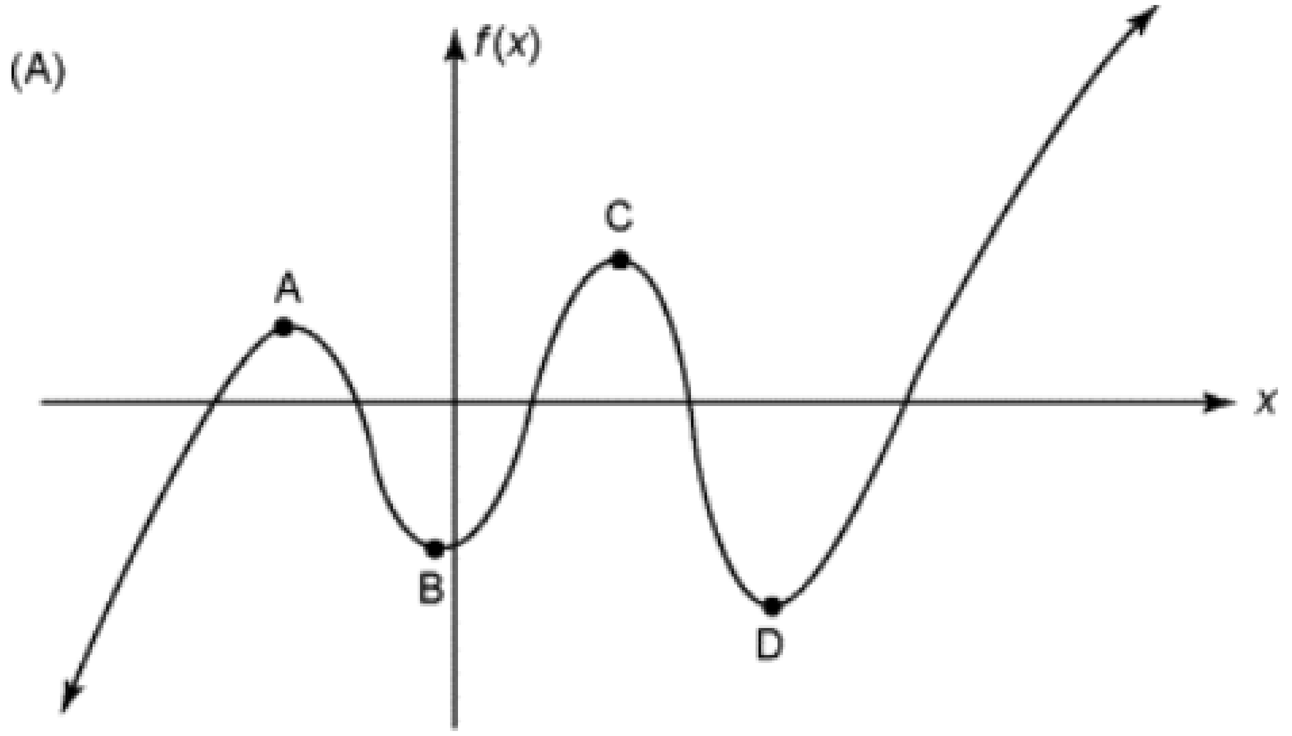
$$g_1(x, y) = x + y - 12 \leq 0$$

$$g_2(x, y) = x - 8 \leq 0$$

Convexity

Convexity
Goal

How can we make sure that the solution is a global minimum?



Convexity

Weierstrass Existence Theorem: Does a global minimum exist?

If f($\underline{x}$) is continuous on a non-empty feasible set S that is closed and bounded, f($\underline{x}$) has a global minimum in S

↳ ok

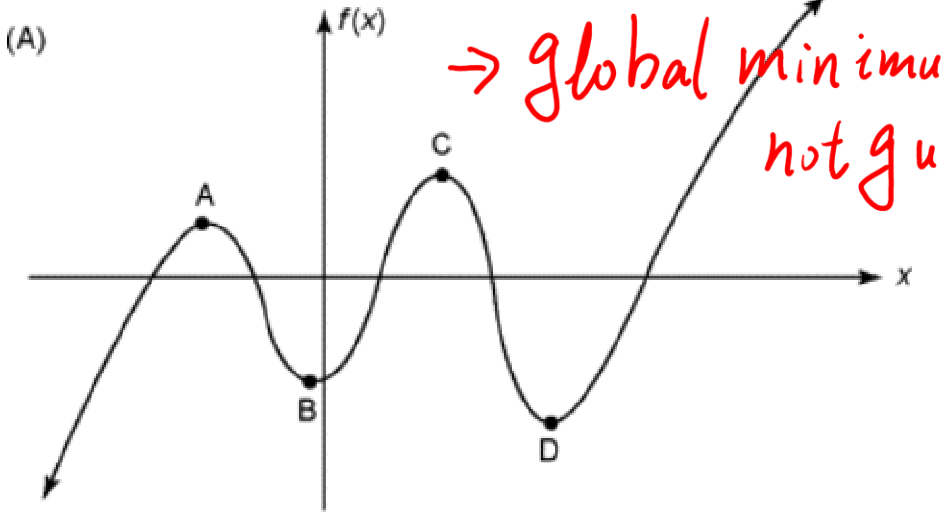
↳ Not easy

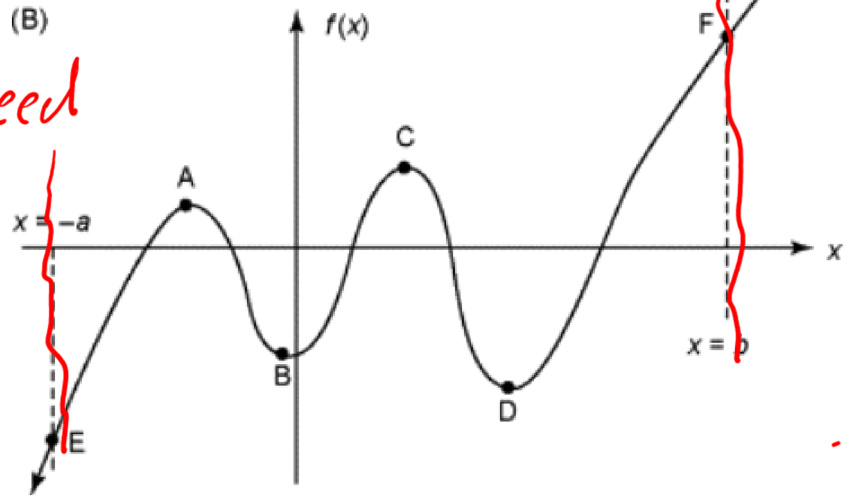
"<" Costraint NOT INCLUDED

- **S is closed if it includes all of its boundary points and every sequence of points have a subsequence that converges to a point in the set**
- **S is bounded if, for any $\underline{x}, \underline{x}^T\underline{x} < c, c$ is a finite number**

$-\infty \le x \le \infty$, S → No Bounded → global minimum not guaranteed

$-a \le x \le b$ → guarantee a global minimum E





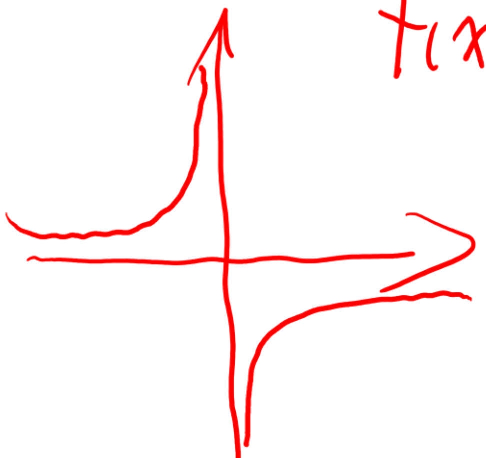
Convexity

Weierstrass Existence Theorem: Example

Check a function $f(x) = -1/x$ defined on the set $S = \{x \mid 0 < x \leq 1\}$, check the existence of a global minimum of the function.

Closed x → does NOT guarantee

Check a function $f(x) = -1/x$ defined on the set $S = \{x \mid 0 \leq x \leq 1\}$, check the existence of a global minimum of the function.



f(x) is NOT definied @ 0, ⇒ NOT Continou

→ NOT guarantee

Convexity

Weierstrass Existence Theorem: Remarks

- If Weierstrass existence theorem is satisfied→ global opt is guaranteed

- If Weierstrass existence theorem is NOT satisfied → global optimum may exist (but can't be guaranteed)

  Examples?

  $\underbrace{f(x) = x^2}$ on $-\infty \le x \le \infty$

  @0 $f_{min} = 0$

- If the numerical process is not converging to a solution, perhaps some conditions of this theorem are violated and the problem formulation needs to be re-examined.

Convexity
Global Optimality

- If Weierstrass existence theorem is satisfied→ global opt is guaranteed

- If the optimization problem can be shown to be convex, any local minimum is also global minimum.

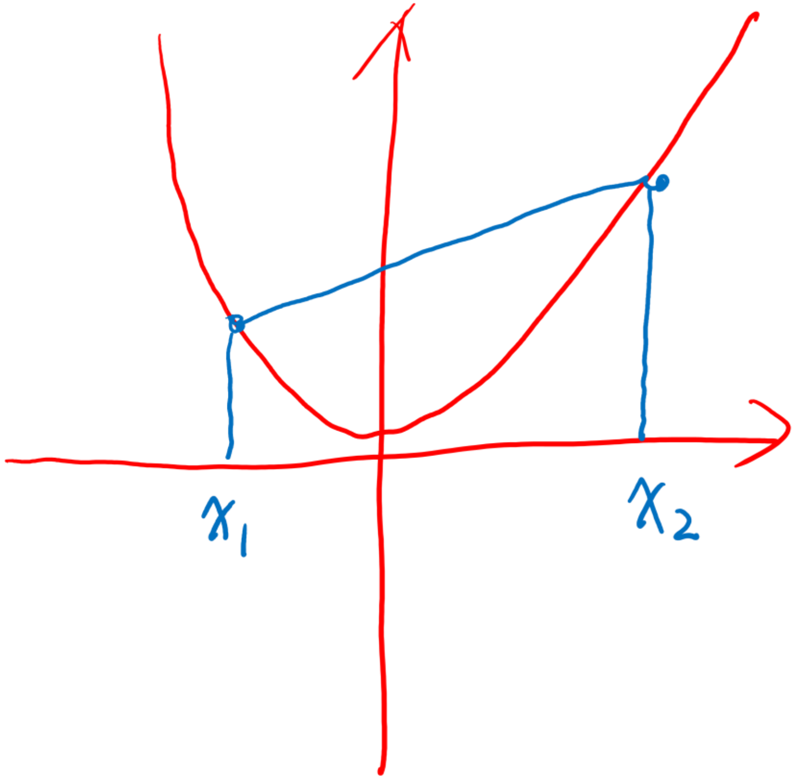
obj → Convex function

S → Convex set

Convexity

Convex function: Definition

$$f(\alpha x_1 + \beta x_2) \leq \alpha f(x_1) + \beta f(x_2)$$
$$\text{for } \alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$$



Convexity

Convex function: Generalization to n dimension

12:45

A function of $f(\underline{x})$ defined on a convex set S is convex if
For any two points $\underline{x_1}, \underline{x_2} \in S$

$$f\left(\alpha \underline{x_1} + \beta \underline{x_2}\right) \leq \alpha f\left(\underline{x_1}\right) + \beta f\left(\underline{x_2}\right)$$

$$\text{for } \alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$$

Rule: sum of convex function is convex.

Proof?

$f(x) = g_1(x) + g_2(x)$, g_1, g_2 both convex

$f(\alpha x_1 + \beta x_2) = g_1(\alpha x_1 + \beta x_2) + g_2(\alpha x_1 + \beta x_2)$

$\leq \alpha g_1(x) + \beta g_1(x_2) + \alpha g_2(x_1) + \beta g_2(x_2)$

$\alpha f(x_1) + \beta f(x_2)$

Convexity

Convex function: Theorem

A function of $f(\underline{x})$ defined on a convex set S is convex iff (if an only if) its hessian matrix $\underline{H}$ is positive semi-definite or positive definite at all points in the set S

Example: Check the convexity of $f(x_1, x_2) = x_1^2 + x_2^2 - 1$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} \qquad \nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \rightarrow \text{P.D.}$$

Convexity

Convex function: Application

A function of $f(\underline{x})$ defined on a convex set S is convex iff (if an only if) its hessian matrix $\underline{H}$ is positive semi-definite or positive definite at all points in the set S

Quadrative function $\underline{x}^T \underline{Q}\, \underline{x}$ is convex iff Q is P.S.D or P.D.

Linear function $\underline{C}^T \underline{x}$ is convex & concave

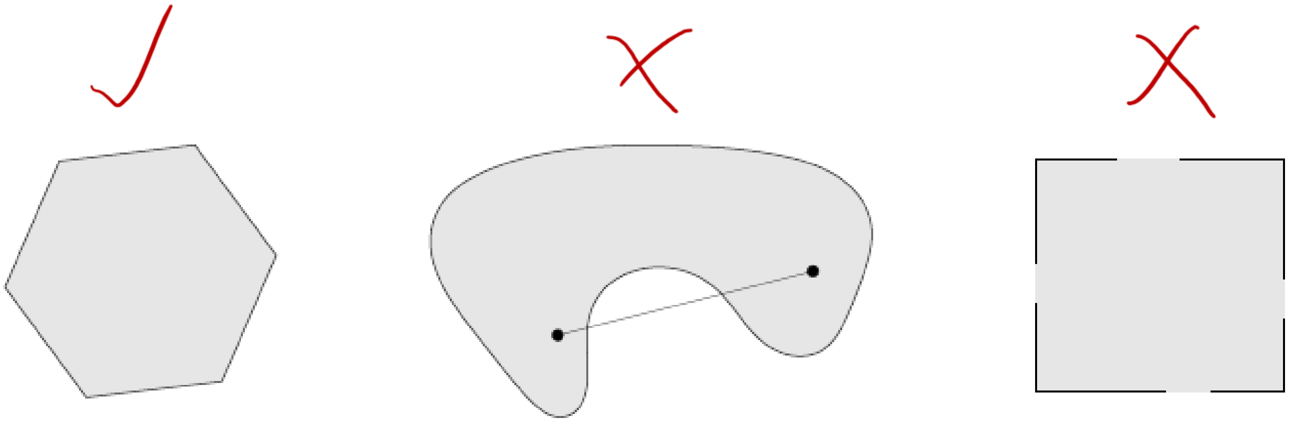
Linear function $\underline{C}^T \underline{x}$ is also called affine function

Convexity
Convex set: Definition

Convex set: contain line segments between any two points in the set

$$x_1, x_2 \in C, \quad 0 \leq \theta \leq 1 \quad \Longrightarrow \quad \theta x_1 + (1-\theta)x_2 \in C$$

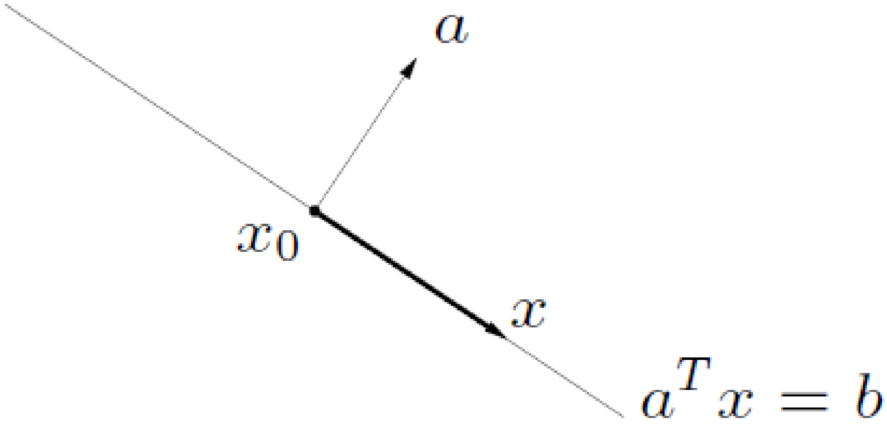
Examples:



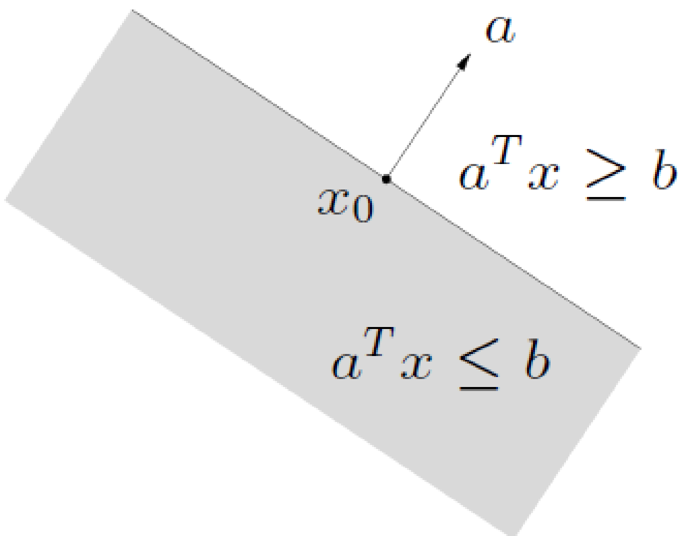
Convexity

Convex set: More Examples

Hyperplane: $\{\underline{x} | \underline{a}^T \underline{x} = \underline{b}\}$, $\underline{a} \neq 0$



Halfspace: set of the form $\{\underline{x} | \underline{a}^T \underline{x} \leq \underline{b}\}$, $\underline{a} \neq 0$



Convexity

Convex set: Operation that preserves convexity

Rule: The intersection of (any number of) convex sets is convex

Application:

Let the feasible domain S be defined by the constraints of the general optimization problem defined in the standard form

$$S = \{\underline{x} | h_i(\underline{x}) = 0, i = 1,,,m; g_j(\underline{x}) \leq 0, j = 1,,,p\}$$

S is a convex set if $g_j(\underline{x})$ are convex AND $h_i(\underline{x})$ are linear

Convexity

Convex set: Convex feasible domain

Let the feasible domain S be defined by the constraints of the general optimization problem defined in the standard form

$$S = \{\underline{x} | h_i(\underline{x}) = 0, i = 1,,, m; g_j(\underline{x}) \le 0, j = 1,,, p\}$$

S is a convex set if $g_j(\underline{x})$ are convex AND $h_i(\underline{x})$ are linear

Remark:

- Feasible domain defined by ANY nonlinear equality constraints is always non-convex
- Feasible domain defined by linear equality or inequality constrains is always convex

GRAND — GRound structure ANalysis and Design

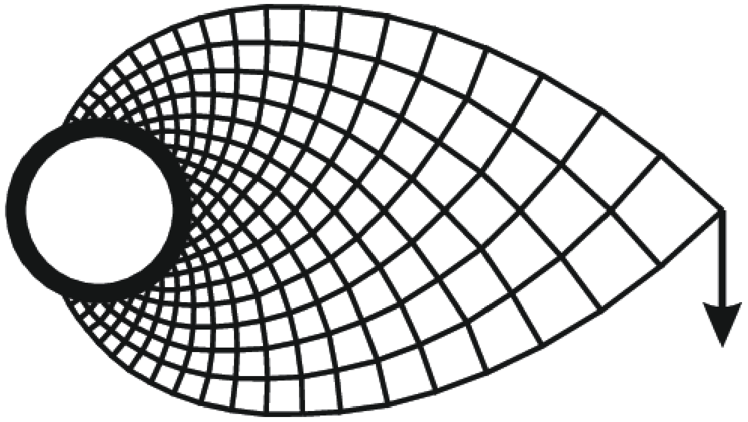
Zegard, T., & Paulino, G. H. (2014). GRAND—Ground structure based topology optimization for arbitrary 2D domains using MATLAB. *Structural and Multidisciplinary Optimization*, *50*, 861-882.

GRAND
Ground Structure

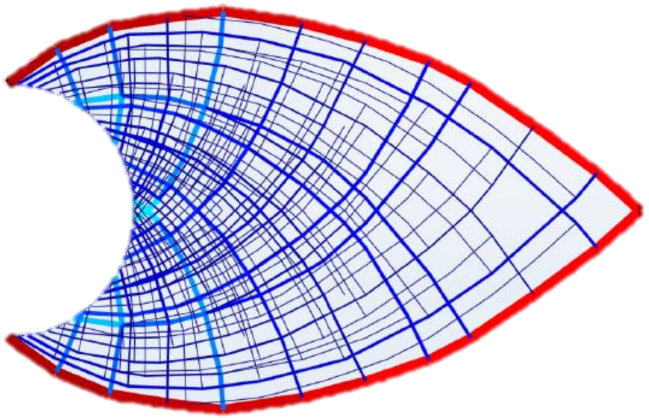
There is no fully automated method to obtain (optimal) Michell trusses

- Michell trusses are infinitely dense
- The GS method provides a close enough approximation using finite number of members

Michell structure



Optimized Ground Structure (GS)



Michell, A. G. M. (1904) The limits of economy of material in frame-structures, Philosophical Magazine, Vol. 8(47), p. 589-597.

GRAND
Ground Structure

- Optimal GS (redundant truss): Minimum Volume

1. Structure includes all load points
2. Structure is rigid (no mechanisms)
3. Structure is safe (stress limits)

Note: GS method does not guarantee stability

- Alternative (easier) conditions

3e. The structure is such that the elastic solution nowhere exceeds the allowable stress

3p. The structure is such that a statically admissible stress distribution can be found

- Structure satisfies 1, 2 and 3e ⇨ elastically admissible structure
- Structure satisfies 1, 2 and 3p ⇨ plastically admissible structure

GRAND

Elastic Formulation

D.V.s: $\underline{a}$*: cross-sectional areas*

$$\min\ V = \underline{a}^T \underline{l}$$

Subject to:

$$\underline{\underline{K}}\,\underline{u} = \underline{f}$$

$$\sigma_c \leq \underline{\sigma} \leq \sigma_T \quad if \quad a_i > 0$$

$$\underline{a} \geq 0$$

GRAND

Plastic Formulation

$$D.V.s: \quad \underline{a}: \text{cross-sectional areas}$$

$$\min \ V = \underline{a}^T \underline{l}$$

$$\text{Subject to:}$$

$$\underline{Bn} = \underline{f}$$

$$\sigma_c \leq \underline{\sigma} \leq \sigma_T \quad \text{if} \quad a_i > 0$$

$$\underline{a} \geq 0$$

Compatibility is NOT enforced!

GRAND

Elastic versus plastic formulation

- Elastic formulation has to comply with

✓Statics

$$\underline{B n} = \underline{f}$$

✓Kinematics

$$\underline{\delta} = \underline{B}^T \underline{u}$$

✓Flexibility (stiffness inverse)

$$\underline{\delta} = \underline{D}\ \underline{n}$$

Where $\underline{D} = \frac{L}{AE}$

Plastic formulation complies only with statics

GRAND

Elastic versus plastic formulation

Elastic Formulation

•Pros

–Multiple load cases

–Material non-linearity

–Geometric non-linearity

•Cons

–Nonlinear optimization

–Vanishing constraints

Plastic Formulation

•Pros

–Linear Programming

–Optimal is global

–No stress discontinuity

–Case $\sigma_T \neq \sigma_C$ is easy

–Useful dual problem

•Cons

–Single static load case

–Linear analysis

GRAND

Plastic Formulation

$$D.V.s: \quad \underline{a}: \text{cross-sectional areas}$$

$$\min\ V = \underline{a}^T \underline{l}$$

$$\text{Subject to:}$$

$$\underline{B}\,\underline{n} = \underline{f}$$

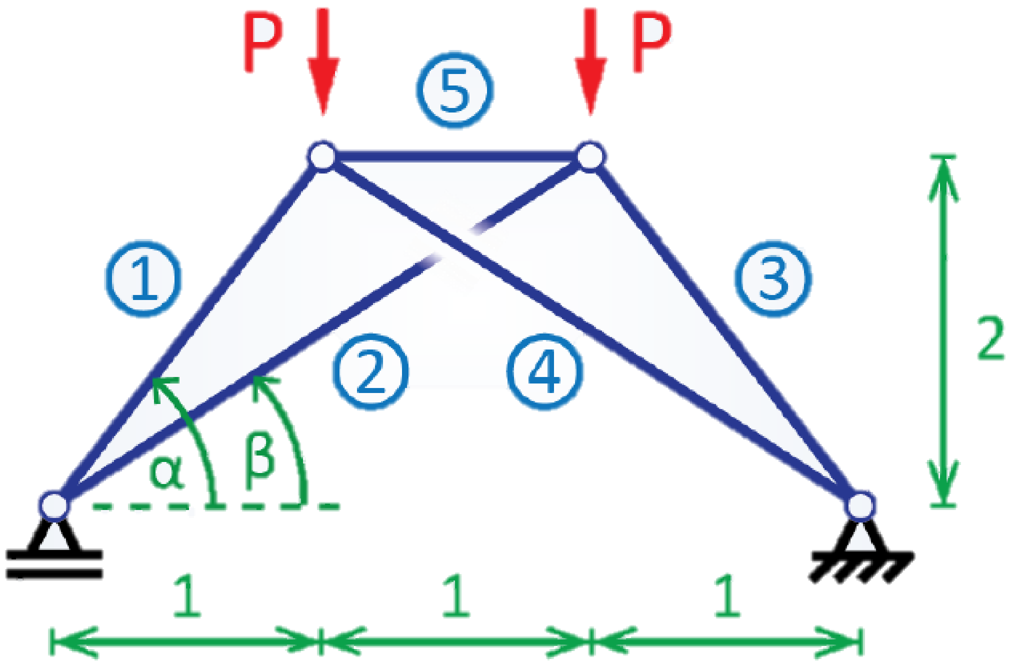
$$\sigma_c \leq \underline{\sigma} \leq \sigma_T \quad if \quad a_i > 0$$

$$\underline{a} \geq 0$$

Compatibility is NOT enforced!

GRAND

Automatic assembly of $\underline{B}$: a powerful example

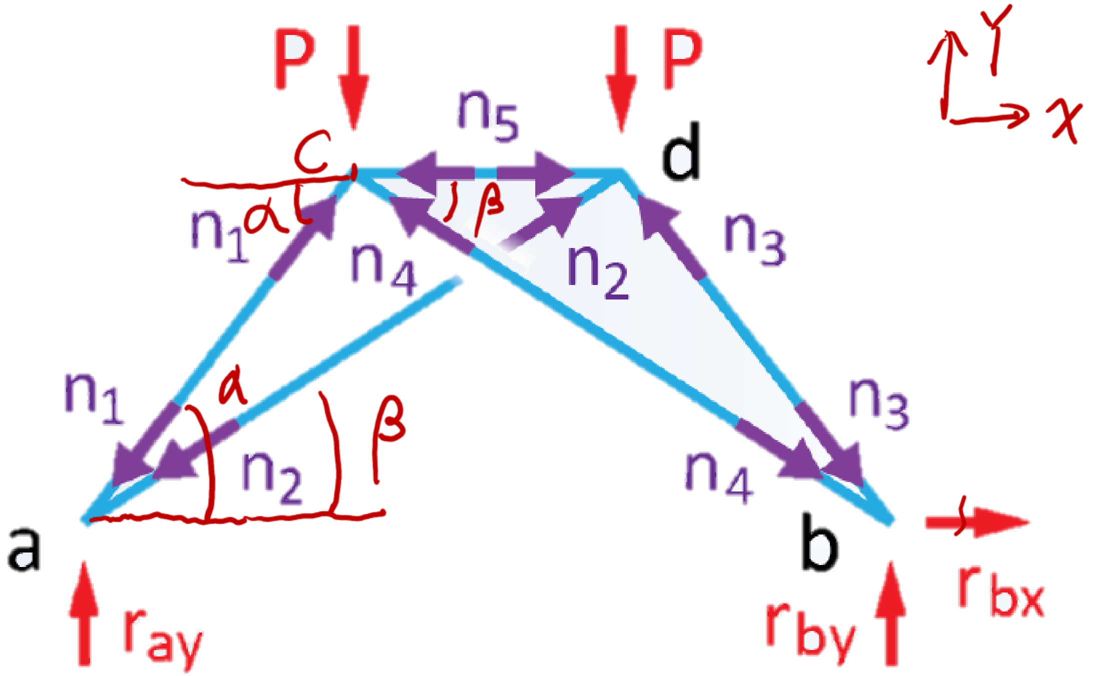


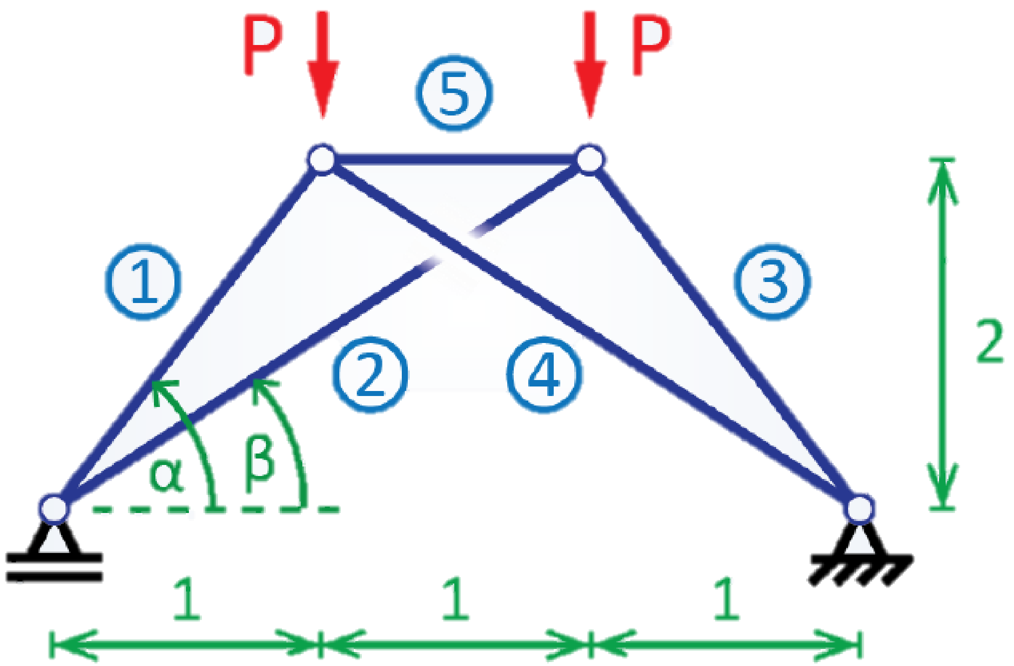
$$\cos\alpha = \frac{1}{\sqrt{5}} \quad , \sin\alpha = \frac{2}{\sqrt{5}}$$

$$\cos\beta = \frac{1}{\sqrt{2}} \quad , \sin\beta = \frac{1}{\sqrt{2}}$$

GRAND

Automatic assembly of B : a powerful example





Take node a&c for example

GRAND

Automatic assembly of B : a powerful example

$$\begin{bmatrix} \mathbf{B} & \mathbf{B}_{nr} \\ \mathbf{B}_{rn} & \mathbf{B}_{rr} \end{bmatrix}^{T} \begin{bmatrix} \mathbf{n} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} -C_\alpha & -C_\beta & 0 & 0 & 0 & 0 & 0 & 0 \\ -S_\alpha & -S_\beta & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & C_\alpha & C_\beta & 0 & 0 & 1 & 0 \\ 0 & 0 & -S_\alpha & -S_\beta & 0 & 0 & 0 & 1 \\ C_\alpha & 0 & 0 & -C_\beta & -1 & 0 & 0 & 0 \\ S_\alpha & 0 & 0 & S_\beta & 0 & 0 & 0 & 0 \\ 0 & C_\beta & -C_\alpha & 0 & 1 & 0 & 0 & 0 \\ 0 & S_\beta & S_\alpha & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ r_{ay} \\ r_{bx} \\ r_{by} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -P \\ 0 \\ -P \end{bmatrix}$$

GRAND

Automatic assembly of B : a powerful example

$$\begin{Bmatrix} \mathbf{B} & \mathbf{B}_{nr} \\ \mathbf{B}_{rn} & \mathbf{B}_{rr} \end{Bmatrix}^{T} \begin{Bmatrix} \mathbf{n} \\ \mathbf{r} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix}$$

We only need $\mathbf{B}^T\mathbf{n}=\mathbf{f}$

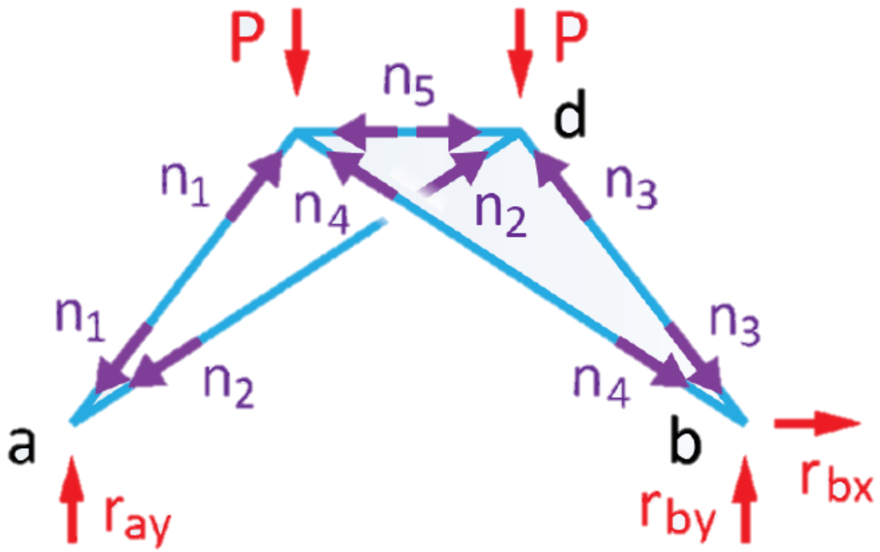
$$\begin{Bmatrix} C_\alpha & C_\beta & 0 & 0 & 0 & 0 & 0 & 0 \\ -S_\alpha & -S_\beta & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & C_\alpha & C_\beta & 0 & 0 & 1 & 0 \\ 0 & 0 & -S_\alpha & -S_\beta & 0 & 0 & 0 & 1 \\ C_\alpha & 0 & 0 & -C_\beta & -1 & 0 & 0 & 0 \\ S_\alpha & 0 & 0 & S_\beta & 0 & 0 & 0 & 0 \\ 0 & C_\beta & -C_\alpha & 0 & 1 & 0 & 0 & 0 \\ 0 & S_\beta & S_\alpha & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ r_{ay} \\ r_{bx} \\ r_{by} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -P \\ 0 \\ -P \end{Bmatrix}$$

GRAND

Automatic assembly of B : a powerful example

$$\begin{bmatrix} \mathbf{B} & \mathbf{B}_{nr} \\ \mathbf{B}_{rn} & \mathbf{B}_{rr} \end{bmatrix}^T \begin{Bmatrix} \mathbf{n} \\ \mathbf{r} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix}$$

We only need $\mathbf{B}^T\mathbf{n}=\mathbf{f}$



$$\begin{bmatrix} C_\alpha & C_\beta & 0 & 0 & 0 \\ C_\alpha & 0 & 0 & -C_\beta & -1 \\ S_\alpha & 0 & 0 & S_\beta & 0 \\ 0 & C_\beta & -C_\alpha & 0 & 1 \\ 0 & S_\beta & S_\alpha & 0 & 0 \end{bmatrix} \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -P \\ 0 \\ -P \end{Bmatrix}$$

GRAND

Plastic Formulation

D.V.s: $\underline{a}$*: cross-sectional areas*

$$\min\ V = \underline{a}^T \underline{l}$$

Subject to:

$$\underline{B}^T \underline{n} = \underline{f}$$

$$-\sigma_c a_i \leq n_i \leq \sigma_t a_i \qquad \forall a_i \geq 0$$

GRAND
Fast ground structure generation

The idea is to “stamp” a pattern in all nodes of a grid

–This pattern has no overlapping bars

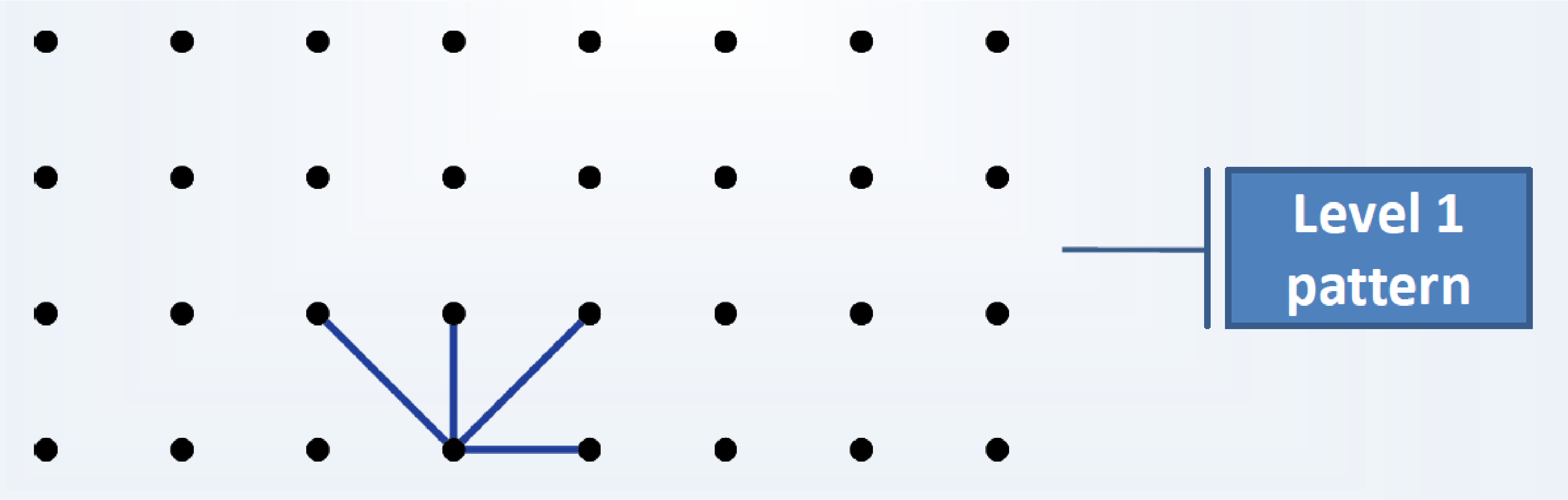
GRAND

Fast ground structure generation

Pattern is created with a user-defined level

- Structure is more redundant with higher levels

Looking at the pattern for a single node



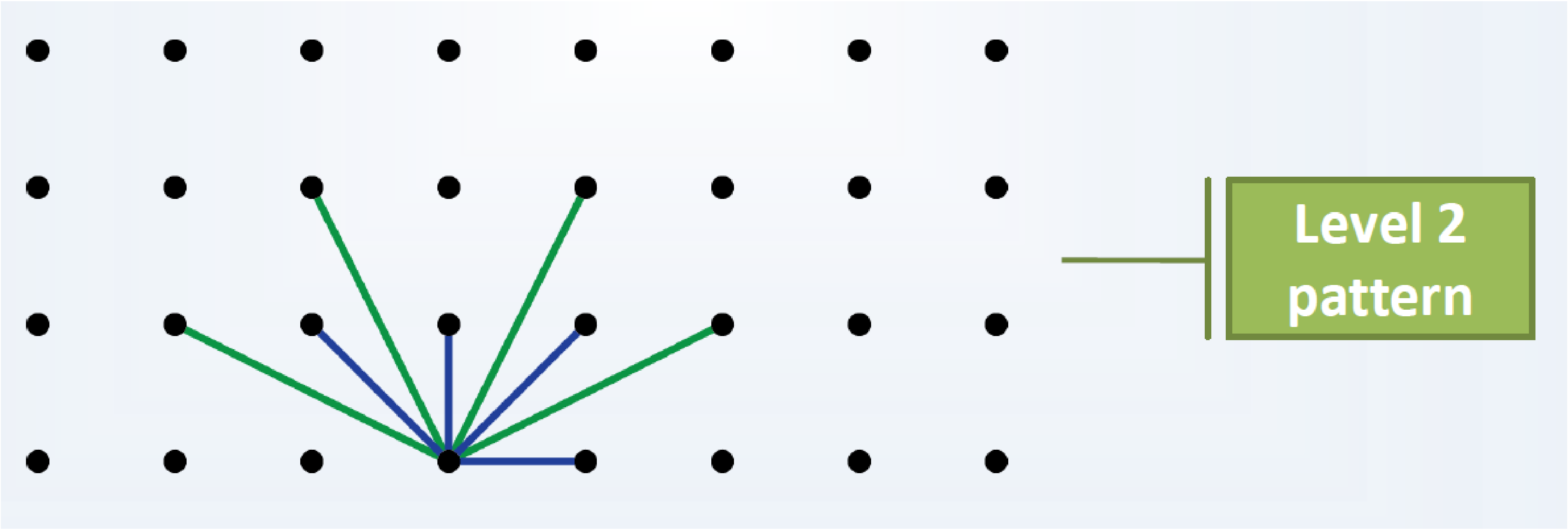
GRAND

Fast ground structure generation

Pattern is created with a user-defined level

- Structure is more redundant with higher levels

Looking at the pattern for a single node



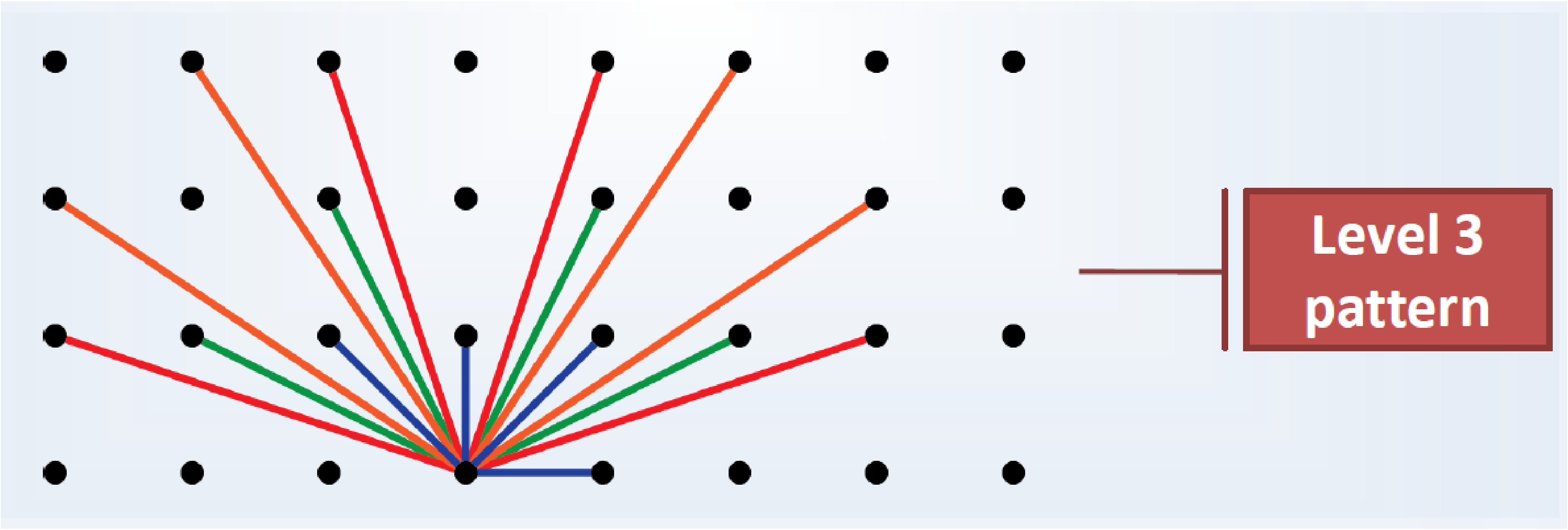
GRAND

Fast ground structure generation

Pattern is created with a user-defined level

- Structure is more redundant with higher levels

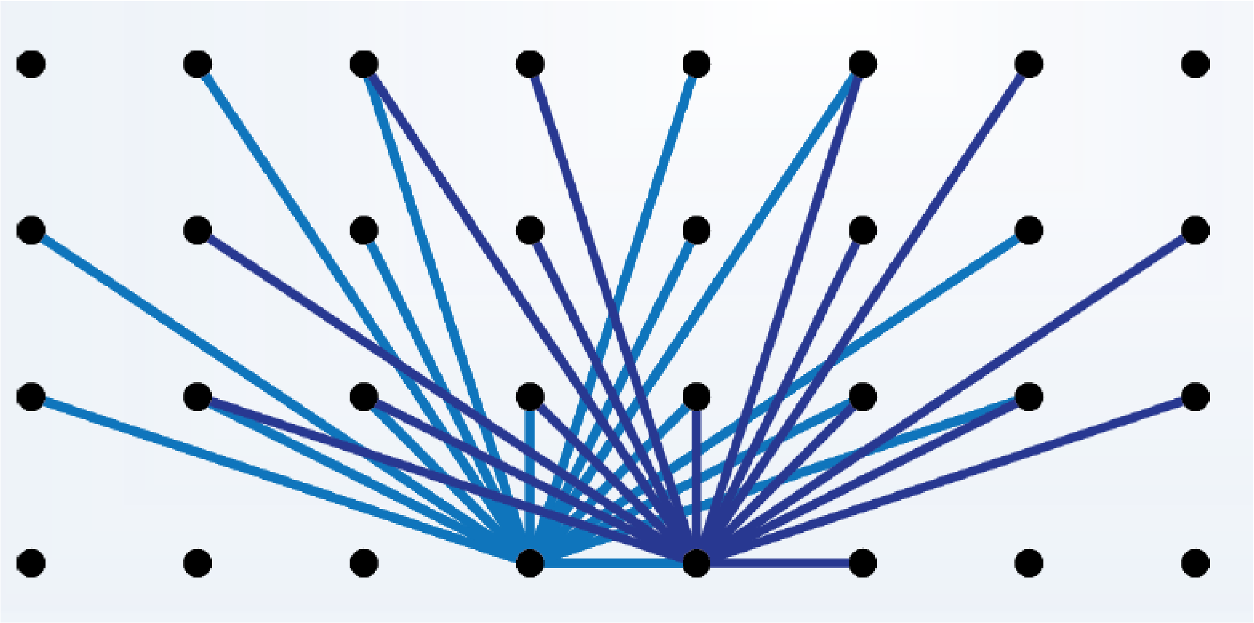
Looking at the pattern for a single node



GRAND

Fast ground structure generation

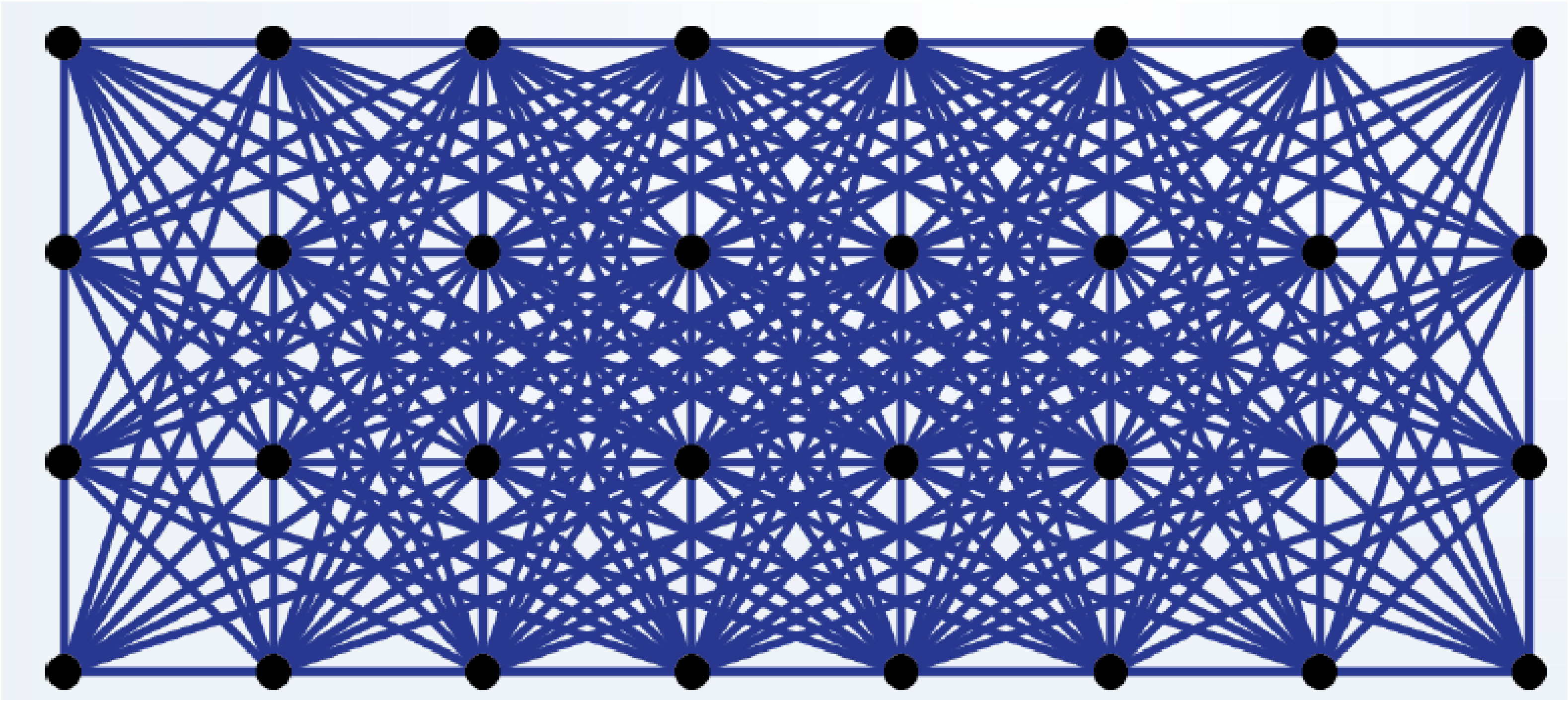
Stamping the pattern in other nodes



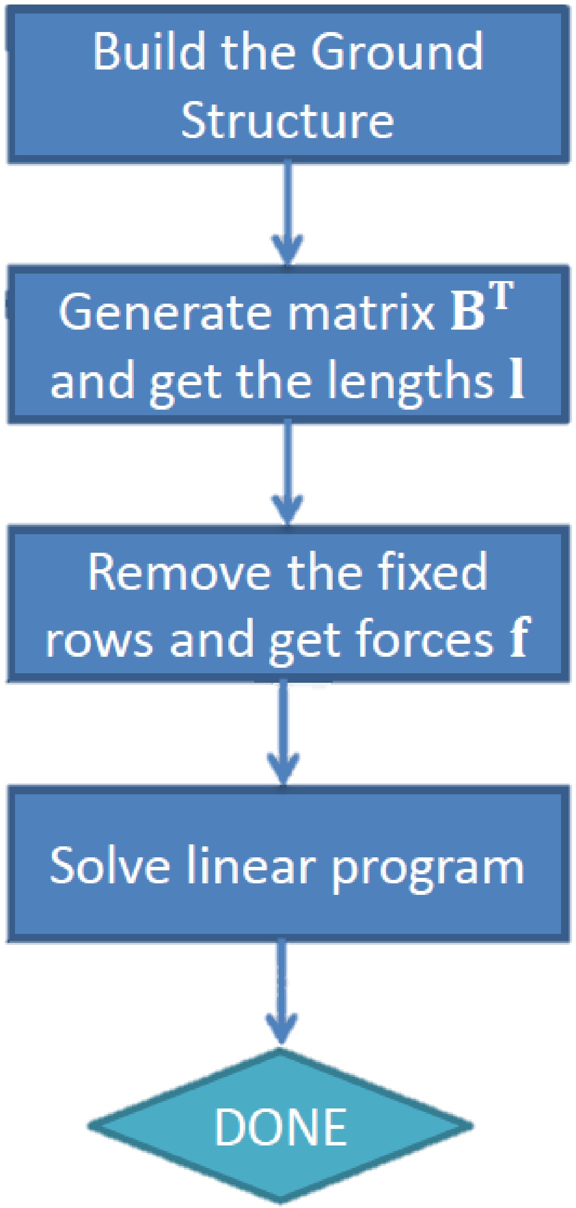
GRAND

Fast ground structure generation

Repeat for all nodes



GRAND
Flow chart



GRAND
MATLAB

Define domain size and the number of grids

Define level of Ground Structure

Generate Ground Structure

Obtain equilibrium matrix and force vector

Call LP optimizer

EDITOR PUBLISH VIEW

```
%GRAND - Ground Structure Analysis and Design Code.
%   Tomas Zegard, Glaucio H Paulino
%% === MESH GENERATION LOADS/BCS ==========================================
kappa = 1.0; ColTol = 0.999999;
Cutoff = 0.002; Ng = 50; % Plot: Member Cutoff & Number of plot groups
% --- OPTION 1: POLYMESHER MESH GENERATION --------------------------------
% addpath('./PolyMesher')
% [NODE,ELEM,SUPP,LOAD] = PolyMesher(@MichellDomain,600,30);
% Lvl = 5; RestrictDomain = @RestrictMichell;
% rmpath('./PolyMesher')
% --- OPTION 2: STRUCTURED-ORTHOGONAL MESH GENERATION ---------------------
[NODE,ELEM,SUPP,LOAD] = StructDomain(30,10,3,1,'Cantilever');

Lvl = 10; RestrictDomain = []; % No restriction for box domain
% --- OPTION 3: LOAD EXTERNALLY GENERATED MESH ----------------------------
% load MeshHook
% Lvl = 10; RestrictDomain = @RestrictHook;
% load MeshSerpentine
% Lvl = 5; RestrictDomain = @RestrictSerpentine;
% load MeshMichell
% Lvl = 4; RestrictDomain = @RestrictMichell;
% load MeshFlower
% Lvl = 4; RestrictDomain = @RestrictFlower;
%% === GROUND STRUCTURE METHOD ============================================
PlotPolyMesh(NODE,ELEM,SUPP,LOAD) % Plot the base mesh
[BARS] = GenerateGS(NODE,ELEM,Lvl,RestrictDomain,ColTol); % Generate the GS
Nn = size(NODE,1); Ne = length(ELEM); Nb = size(BARS,1);
[BC] = GetSupports(SUPP);                 % Get reaction nodes
[BT,L] = GetMatrixBT(NODE,BARS,BC,Nn,Nb); % Get equilibrium matrix
[F] = GetVectorF(LOAD,BC,Nn);             % Get nodal force vector
fprintf('Mesh: Elements %d, Nodes %d, Bars %d, Level %d\n',Ne,Nn,Nb,Lvl)
BTBT = [BT -BT]; LL = [L; kappa*L]; sizeBTBT = whos('BTBT'); clear BT L
fprintf('Matrix [BT -BT]: %d x %d in %gMB (%gGB full)\n',...
        length(F),length(LL),sizeBTBT.bytes/2^20,16*(2*Nn)*Nb/2^30)

tic, [S,vol,exitflag] = linprog(LL,[],[],BTBT,F,zeros(2*Nb,1));
fprintf('Objective V = %f\nlinprog CPU time = %g s\n',vol,toc);

S = reshape(S,numel(S)/2,2);  % Separate slack variables
A = S(:,1) + kappa*S(:,2);    % Get cross-sectional areas
N = S(:,1) - S(:,2);          % Get member forces
%% === PLOTTING ===========================================================
PlotGroundStructure(NODE,BARS,A,Cutoff,Ng)
PlotBoundary(ELEM,NODE)
```

script Ln 13 Col 1

GRAND
MATLAB

```
function [NODE,ELEM,SUPP,LOAD]=StructDomain(Nx,Ny,Lx,Ly,ProblemID)
% Generate structured-orthogonal domains
[X,Y] = meshgrid(linspace(0,Lx,Nx+1),linspace(0,Ly,Ny+1));
NODE = [reshape(X,numel(X),1) reshape(Y,numel(Y),1)];
k = 0; ELEM = cell(Nx*Ny,1);
for j=1:Ny, for i=1:Nx
        k = k+1;
        n1 = (i-1)*(Ny+1)+j; n2 = i*(Ny+1)+j;
        ELEM{k} = [n1 n2 n2+1 n1+1];
end, end

if (nargin==4 || isempty(ProblemID)), ProblemID = 1; end
switch ProblemID
    case {'Cantilever','cantilever',1}
        SUPP = [(1:Ny+1)' ones(Ny+1,2)];
        LOAD = [Nx*(Ny+1)+round((Ny+1)/2) 0 -1];
    case {'MBB','Mbb','mbb',2}
        SUPP = [Nx*(Ny+1)+1 NaN 1;
                (1:Ny+1)' ones(Ny+1,1) nan(Ny+1,1)];
```

GRAND
MATLAB

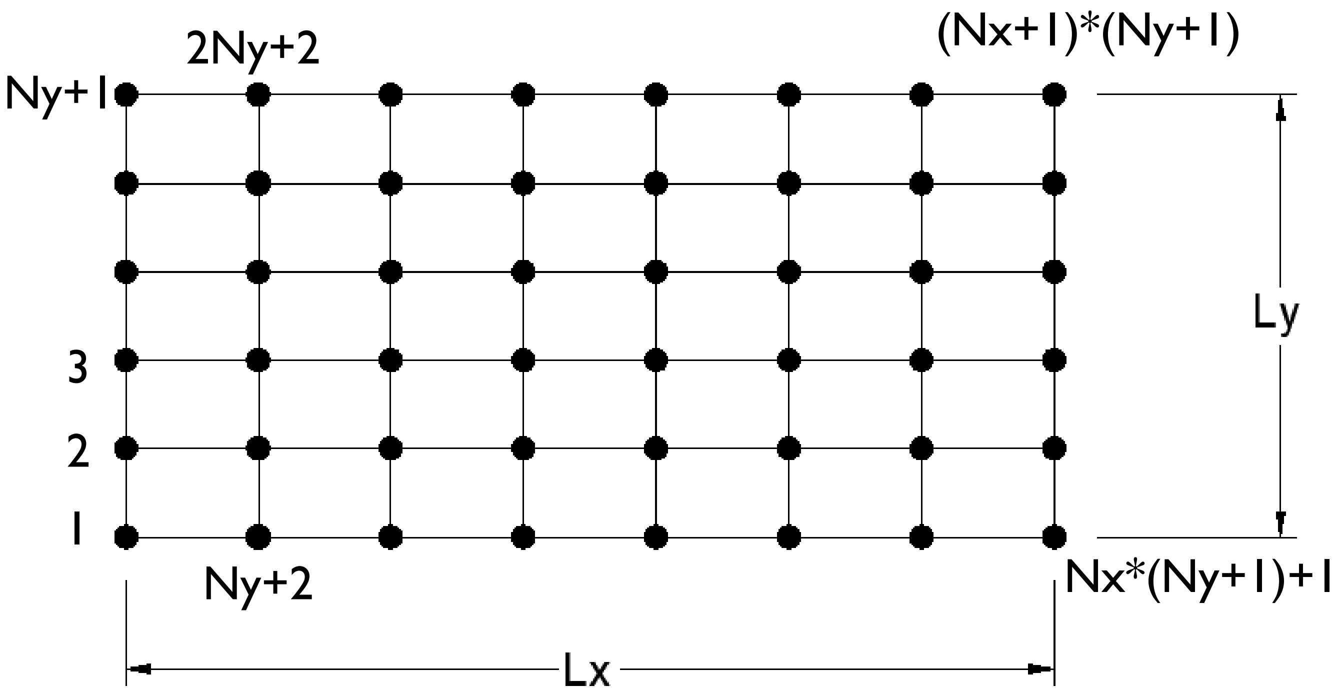
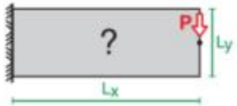
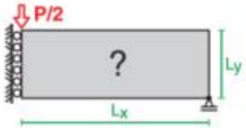
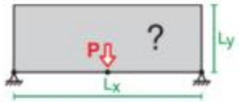
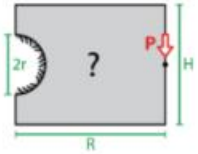
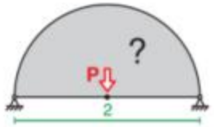
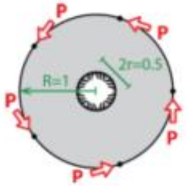


Table 1 Domain definition (base mesh) input variables

Variable Name	Type & Size	Description
`NODE`	`array` $N_n \times 2$	Each row p has the nodal coordinates x and y for node p.
`ELEM`	`cell` $N_e \times 1$	Every element in the list is a row vector containing the node numbers for a particular element.
`SUPP`	`array` $N_f \times 3$	Each row consists of a node number, fixity x and fixity in y. Any value other than `NaN` specifies fixity. The total number of specified fixities is N_{sup}.
`LOAD`	`array` $N_l \times 3$	Each row consists of a node number, load in x and load in y. A zero or `NaN` specify no force in that direction.

GRAND

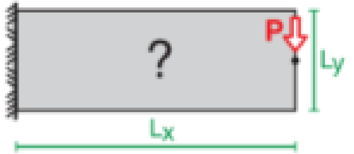
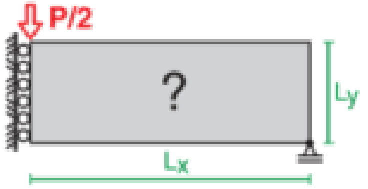
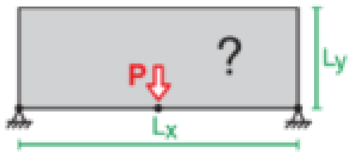
MATLAB-predefined cases

Domain	Base mesh definition	Restriction zone	Comments
	`StructDomain(Nx,Ny,Lx,Ly,'Cantilever')` or PolyMesher with `@CantileverDomain`	—	Horizontal and Vertical lengths `Lx` and `Ly`, using `Nx` and `Ny` elements in each direction. By default the load is $P = 1$
	`StructDomain(Nx,Ny,Lx,Ly,'MBB')` or PolyMesher with `@MbbDomain`	—	Horizontal and Vertical lengths `Lx` and `Ly`, using `Nx` and `Ny` elements in each direction. By default the load is $P/2 = 0.5$
	`StructDomain(Nx,Ny,Lx,Ly,'Bridge')` or PolyMesher with `@BridgeDomain`	—	Dimensions `Lx` and `Ly`, with `Nx` and `Ny` elements in each direction, with a default load $P = 1$. The analytical solution for this problem (if $L_y \geq \sqrt{2}L_x/4$) is: $V_{opt} = P\left(\frac{L_x}{2}\right)\left(\frac{1}{2}+\frac{\pi}{4}\right)\left[\frac{1}{\sigma_T}+\frac{1}{\sigma_C}\right] = 1.2854L_x$
	PolyMesher with `@MichellDomain` or `load MeshMichell`	`@RestrictMichell`	The default parameters are $r = 1$, $R = 5$, $H = 4$ and $P = 1$. The analytical solution for this problem is: $V_{opt} = PR\log\left(\frac{R}{r}\right)\left[\frac{1}{\sigma_T}+\frac{1}{\sigma_C}\right] = 16.0944$
	PolyMesher with `@HalfcircleDomain`	—	The default load is $P = 1$. The analytical solution for this problem is: $V_{opt} = Pr\left(\frac{1}{2}+\frac{\pi}{4}\right)\left[\frac{1}{\sigma_T}+\frac{1}{\sigma_C}\right] = 2.5708$
	PolyMesher with `@FlowerDomain` or `load MeshFlower`	`@RestrictFlower`	The default load is $P = 1$. The analytical solution for this problem is: $V_{opt} = 5PR\log\left(\frac{R}{r}\right)\left[\frac{1}{\sigma_T}+\frac{1}{\sigma_C}\right] = 13.8629$

GRAND
Play!

Try at least two predefined cases (e.g., Cantilever and MBB).

- Different tension to compression ratio (kappa)
- Different domain
- Different mesh size
- Different level of connection
- Compare their load and support matrix
- Try modify the load and support matrix

Domain	Base mesh definition
	`StructDomain(Nx,Ny,Lx,Ly,'Cantilever')` or PolyMesher with `@CantileverDomain`
	`StructDomain(Nx,Ny,Lx,Ly,'MBB')` or PolyMesher with `@MbbDomain`
	`StructDomain(Nx,Ny,Lx,Ly,'Bridge')` or PolyMesher with `@BridgeDomain`