Πανεπιστήμιο Θεσσαλίας
Τμήμα Πολιτικών Μηχανικών

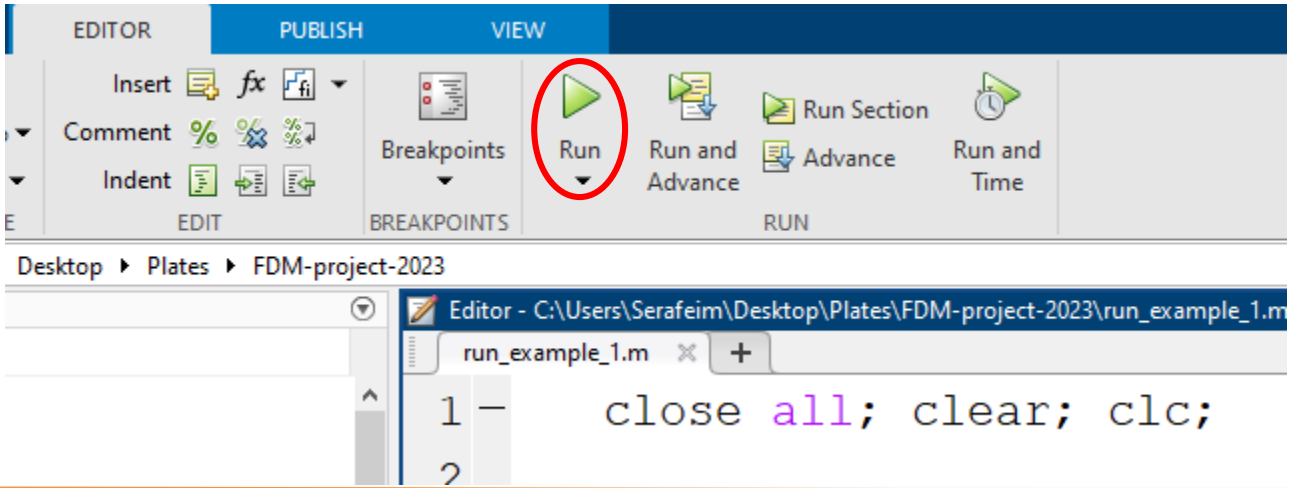
Ανάλυση επιφανειακών φορέων

Ενότητα 8:
Μέθοδος Πεπερασμένων Διαφορών (μέρος Γ)
Προγραμματισμός της ΜΠΔ στο Matlab

Σεραφείμ Μπακαλάκος
Phd Πολιτικός Μηχανικός
2023-2024

Περιγραφή φορέα

- Script: **run_example_1.m**
- Εδώ θα:
  i. Καλέσουμε functions που θα μας βοηθήσουν να περιγράψουμε το φορέα
  ii. Καλέσουμε τη function που εκτελεί ανάλυση Πεπερασμένων Διαφορών
  iii. Καλέσουμε συναρτήσεις που σχεδιάζουν τη βύθιση και τις ροπές
- Για τρέξουμε το script αυτό, ανοίγουμε το αρχείο και πατάμε “Run” από την ομάδα εντολών “Editor”



Περιγραφή φορέα

- Script: **run_example_1.m**
- Δίνουμε τις ιδιότητες της πλάκας και του καννάβου
- Δεν βάζουμε μονάδες. Αντίθετα όλα τα δεδομένα που δίνουμε πρέπει να εχουν τις ίδιες μονάδες μήκους και δύναμης. Έτσι και τα αποτελέσματα θα έχουν αυτές τις μονάδες.
- **nx** και **ny** είναι ο αριθμός των κόμβων του καννάβου κατά τους άξονες x, y αντίστοιχα. Μόνο των πραγματικών κόμβων. Για όλους τους κόμβους θα έχουμε τα $\mathbf{ni} = \mathbf{nx} + 4$, $\mathbf{nj} = \mathbf{ny} + 4$

```
3      % Initialize the plate domain
4 -    D = 5000;
5 -    v = 0.3;
6 -    Lx = 3;
7 -    Ly = 2;
8 -    nx = 25; % number of real nodes along axis x
9 -    ny = 17; % number of real nodes along axis y
10 -   [domain, node_supports] = initialize_domain(D, v, Lx, Ly, nx, ny);
```

Περιγραφή φορέα

- Script: **run_example_1.m**
- Η function **initialize_domain(…)** παίρνει σαν input τις παραπάνω ιδιότητες και δημιουργεί 2 μεταβλητές (**domain** και **node_supports**).
- Η μεταβλητή **domain** είναι struct, δηλαδή μια δομή δεδομένων που αποθηκεύει διάφορα άλλα δεδομένα.
- Το **domain** θα δίνεται σε όποια function χρειάζεται ιδιότητες του φορέα.

- Τα **Lx, Ly, D, v, nx, ny** ορίστηκαν προηγουμένως
- Το **h** είναι η απόσταση μεταξύ κόμβων κατά τον άξονα x και y. Πρέπει

$$dx = \frac{Lx}{nx-1} = dy = \frac{Ly}{ny-1} = h$$

- Τα **ni, nj** είναι το πλήθη κόμβων (πραγματικών και βοηθητικών) κατά τους άξονες x, y.

domain

1x1 struct with 9 fields

Field	Value
ni	29
nj	21
h	0.1250
D	5000
v	0.3000
Lx	3
Ly	2
nx	25
ny	17

Περιγραφή φορέα

- Script: **run_example_1.m**
- Η function **initialize_domain(…)** παίρνει σαν input τις παραπάνω ιδιότητες και δημιουργεί 2 μεταβλητές (**domain** και **node_supports**).
- Η μεταβλητή **node_supports** είναι ένας πίνακας κειμένου (strings) με διαστάσεις (**nj** × **ni**), που για κάθε κόμβο του καννάβου περιέχει ‘free’ (ελεύθερος), ‘fixed’ (πάκτωση), ‘simple’ (απλή έδραση) ή ‘g’ (βοηθητικός/ghost).
- Αρχικά όλοι οι πραγματικοί κόμβοι είναι ‘free’ μέχρι να τους δεσμεύσουμε.
- Π.χ. για nx=4, ny=3

Workspace

node_supports

7x8 cell

	1	2	3	4	5	6	7	8
1	g	g	g	g	g	g	g	g
2	g	g	g	g	g	g	g	g
3	g	g	free	free	free	free	g	g
4	g	g	free	free	free	free	g	g
5	g	g	free	free	free	free	g	g
6	g	g	g	g	g	g	g	g
7	g	g	g	g	g	g	g	g

Περιγραφή φορέα

- Script: **run_example_1.m**
- Έπειτα περιγράφουμε τις συνοριακές συνθήκες.
- Η function **find_nodes_with_x(x0, y_lower, y_upper, domain)** βρίσκει όλους τους κόμβους με $x = x0$, y_lower ≤ y ≤ y_upper.
- Τους επιστρέφει στην μεταβλητή **nodes_index2D**. Αυτή είναι ένας πίνακας με μία γραμμή για κάθε κόμβο και 2 στήλες. Η 1η στήλη είναι το i του κόμβου. Η 2η στήλη είναι το j του κόμβου

➢ Π.χ.

nodes_index2D

3x2 double

	1	2
1	3	3
2	3	4
3	3	5

```
13      % Apply supports to boundary and internal nodes
14      % Supports at edge x=0
15 -    nodes_index2D = find_nodes_with_x(0, 0, Ly, domain);
16 -    node_supports = apply_supports('simple', nodes_index2D, node_supports);
17
18      % Supports at edge x=Lx
19 -    nodes_index2D = find_nodes_with_x(Lx, 0, Ly, domain);
20 -    node_supports = apply_supports('free', nodes_index2D, node_supports);
```

Περιγραφή φορέα

- Script: **run_example_1.m**
- Η function **apply_supports(support_text, nodes_index2D, node_supports)** παίρνει σαν input
  - Τον πίνακα **node_supports** με τις υπαρχουσες συνοριακές συνθήκες
  - Τον πίνακα **nodes_index2D** με τα i, j των κόμβων που μας ενδιαφέρουν
  - Κείμενο 'free', 'fixed' ή 'simple' ανάλογα με το τι θέλουμε να επιβάλουμε στους κόμβους αυτούς.
- Επιστρέφει τον πίνακα **node_supports**, έχοντας εφαρμόσει τις συνοριακές συνθήκες που ζητάμε, στους κόμβους που ζητάμε.

```
13      % Apply supports to boundary and internal nodes
14      % Supports at edge x=0
15 -    nodes_index2D = find_nodes_with_x(0, 0, Ly, domain);
16 -    node_supports = apply_supports('simple', nodes_index2D, node_supports);
17
18      % Supports at edge x=Lx
19 -    nodes_index2D = find_nodes_with_x(Lx, 0, Ly, domain);
20 -    node_supports = apply_supports('free', nodes_index2D, node_supports);
```

Περιγραφή φορέα

- Script: **run_example_1.m**
- Οι functions **find_nodes_with_y(…)** και **find_node_with_xy(…)** λειτουργούν αντίστοιχα με την **find_nodes_with_x(…)**.

```
22      % Supports at edge y=0
23 -    nodes_index2D = find_nodes_with_y(0, 0, Lx, domain);
24 -    node_supports = apply_supports('simple', nodes_index2D, node_supports);
25
26      % Supports at edge y=Ly
27 -    nodes_index2D = find_nodes_with_y(Ly, 0, Lx, domain);
28 -    node_supports = apply_supports('fixed', nodes_index2D, node_supports);
```

- Μετά από την περιγραφή των συνοριακών συνθηκών στις 4 παρειές της πλάκας, ο πίνακας **node_supports** θα είναι

	1	2	3	4	5	6	7	8
1	g	g	g	g	g	g	g	g
2	g	g	g	g	g	g	g	g
3	g	g	simple	simple	simple	simple	g	g
4	g	g	simple	free	free	free	g	g
5	g	g	fixed	fixed	fixed	fixed	g	g
6	g	g	g	g	g	g	g	g
7	g	g	g	g	g	g	g	g

Περιγραφή φορέα

- Script: **run_example_1.m**
- Έπειτα περιγράφουμε τα φορτία.
- Εδώ εφαρμόζουμε φορτίο γραμμικά μεταβαλλόμενο κατά x:

$$q(x,y) = q_{min} + (q_{max} - q_{min}) \cdot \frac{x}{Lx}$$

- Ο υπολογισμός των ισοδύναμων φορτίων $q_{i,j}$ γίνεται στη function **apply_equivalent_loads_linear_x(q_start, q_end, domain)**.
- Θα την εξετάσουμε πιο αναλυτικά αργότερα. Αν θέλαμε άλλο είδος φόρτισης (π.χ. γραμμικά μεταβαλλόμενο κατά y) θα χρησιμοποιούσαμε άλλη function.
- Οι σπουδαστές καλούνται να υλοποιήσουν 2 τέτοιες functions.

```
30      % Apply loads to nodes
31 -    qmin = 10;
32 -    qmax = 11;
33 -    Qji = apply_equivalent_loads_linear_x(qmin, qmax, domain);
```

Περιγραφή φορέα

- Script: **run_example_1.m**
- Ο υπολογισμός των ισοδύναμων φορτίων $q_{i,j}$ γίνεται στη function **apply_equivalent_loads_linear_x(q_start, q_end, domain)** ή κάποια παρόμοια.
- Όλες αυτές οι functions θα επιστρέφουν πίνακα **Qji** με διαστάσεις (**nj** × **ni**), που περιέχει τα ισοδύναμα φορτία.
- Δηλαδή, στη γραμμή j και στήλη i περιέχεται το ισοδύναμο φορτίο $q_{i,j}$.
- Στους βοηθητικούς κόμβους θα έχει τιμή **NaN** (not a number), καθώς δεν υπάρχουν φορτία.

Qji

7x8 double

	1	2	3	4	5	6	7	8
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	10	10.3333	10.6667	11	NaN	NaN
4	NaN	NaN	10	10.3333	10.6667	11	NaN	NaN
5	NaN	NaN	10	10.3333	10.6667	11	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Ανάλυση και αποτελέσματα

- Script: **run_example_1.m**
- Η function **perform_fdm_analysis(node_supports, Qji, domain)** εκτελεί ανάλυση με τη μέθοδο Πεπερασμένων Διαφορών. Θα την εξετάσουμε πιο λεπτομερώς στη συνέχεια.
- Επιστρέφει πίνακα **Wji** με διαστάσεις (**nj** × **ni**), που περιέχει τις βυθίσεις στους πραγματικούς και βοηθητικούς κόμβους. Δηλαδή, στη γραμμή j και στήλη i περιέχεται η βύθιση $w_{i,j}$.

- Στους ανενεργούς βοηθητικούς κόμβους (που δεν εμπλέκονται στις εξισώσεις) θα έχει τιμή **NaN** .

Wji

7x8 double

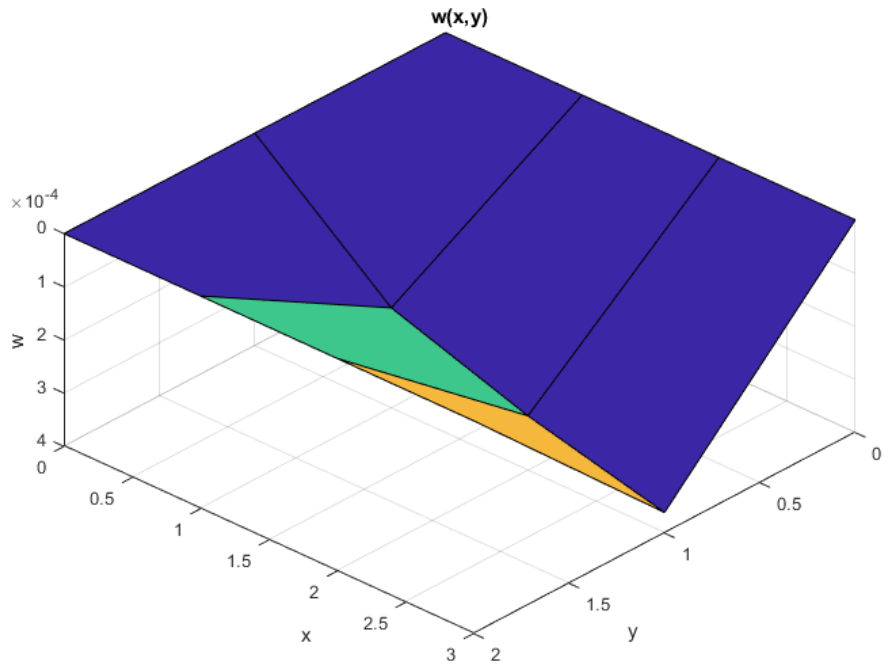
	1	2	3	4	5	6	7	8
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	-4.5996e-20	-2.2154e-04	-3.1669e-04	-3.9103e-04	NaN	NaN
3	NaN	7.4102e-20	-8.2467e-20	-7.4102e-20	-9.0864e-20	2.1281e-19	-2.4715e-19	NaN
4	NaN	-2.2154e-04	4.5996e-20	2.2154e-04	3.1669e-04	3.9103e-04	6.9998e-04	0.0023
5	NaN	0	0	0	0	0	0	NaN
6	NaN	NaN	0	2.2154e-04	3.1669e-04	3.9103e-04	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
36      % Perform analysis using the Finite Difference Method
37 -    Wji = perform_fdm_analysis(node_supports, Qji, domain);
```

Ανάλυση και αποτελέσματα

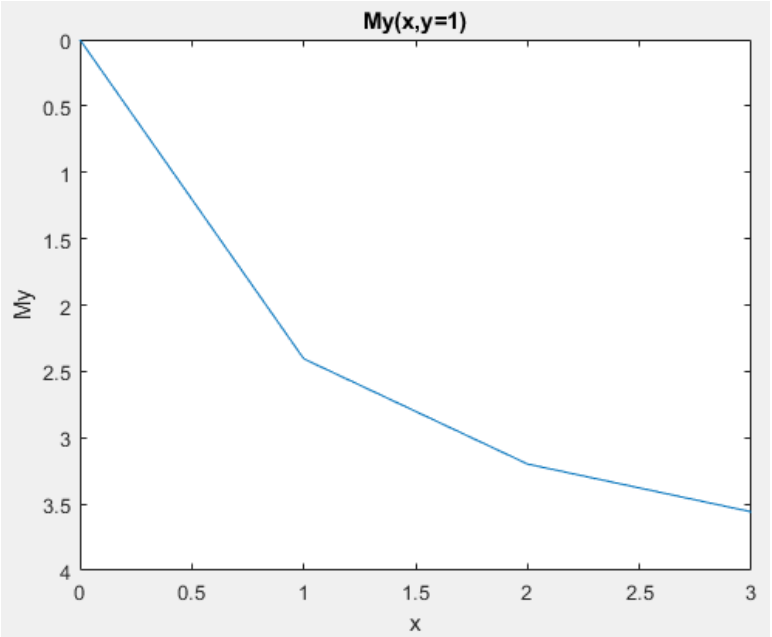
- Script: **run_example_1.m**
- Η function **calc_w_max(Wji, domain)** βρίσκει τη μέγιστη βύθιση **w_max** και τις συντεταγμένες **x_max**, **y_max** του κόμβου όπου εμφανίζεται.
- Η function **plot_w(Wji, domain)** σχεδιάζει το 3D διάγραμμα βυθίσεων

```
39      % Calculate and plot results
40 -    [w_max, x_max, y_max] = calc_w_max(Wji, domain)
41 -    plot_w(Wji, domain);
```



Ανάλυση και αποτελέσματα

- Script: **run_example_1.m**
- Οι functions **plot_Mx_parallel_to_x(y0, Wji, domain)** , **plot_My_parallel_to_x(x0, Wji, domain)**, κτλ, σχεδιάζουν διαγράμματα ροπών πάνω σε συγκεκριμένες ευθείες.
- Για “parallel_to_x” θα σχεδιαστεί διάγραμμα πάνω στην ευθεία y=y0
- Για “parallel_to_y” θα σχεδιαστεί διάγραμμα πάνω στην ευθεία x=x0




```
41 -    plot_Mx_parallel_to_x(Ly/2, Wji, domain);
42 -    plot_My_parallel_to_x(Ly/2, Wji, domain);
43 -    plot_Mxy_parallel_to_x(Ly/2, Wji, domain);
44 -    plot_Mx_parallel_to_y(Lx/2, Wji, domain);
45 -    plot_My_parallel_to_y(Lx/2, Wji, domain);
46 -    plot_Mxy_parallel_to_y(Lx/2, Wji, domain);
```

Οργάνωση του κώδικα

- Μέχρι τώρα περιγράψαμε το φορέα, τον αναλύσαμε και είδαμε αποτελέσματα στο script **run_example_1.m**
- Αυτό είναι top-level κώδικας, που καλεί και συντονίζει τις υπόλοιπες functions.
- Σε ένα εμπορικό λογισμικό, θα υπάρχει GUI, αντί για αυτό το αρχείο. Συχνά όμως υπάρχει και η δυνατότητα εργασίας με script, καθώς είναι πολύ βολικά για αυτοματοποίηση.
- Το **run_example_1.m** λύνει το παράδειγμα 1. Για άλλο πρόβλημα χρειαζόμαστε αντίστοιχο script, το οποίο όμως είναι παρόμοιο. Π.χ. **run_example_2.m**
- Οι σπουδαστές καλούνται να συμπληρώσουν το script **run_project.m** για το θέμα εξαμήνου.
- Στη συνέχεια θα εξετάσουμε πιο αναλυτικά κάποια από τα functions που χρησιμοποιούνται

Ισοδύναμα φορτία

- Function: **apply_equivalent_loads_linear_x(q_start, q_end, domain)**
- Εδώ εφαρμόζουμε φορτίο γραμμικά μεταβαλλόμενο κατά x:

  $q(x,y)=q\_start+(q\_end-q\_start)\cdot x/Lx$

- Το φορτίο είναι ομαλά μεταβαλλόμενο, άρα $q_{i,j} = q(x_i, y_j)$

- Αρχικά διαβάζουμε τις ιδιότητες του φορέα από το struct domain
- Έπειτα υπολογίζουμε το ισοδύναμο φορτίο σε κάθε κόμβο.

```
13      % Domain properties
14 -    ni = domain.ni;
15 -    nj = domain.nj;
16 -    nx = domain.nx;
17 -    ny = domain.ny;
18 -    Lx = domain.Lx;
19 -    h = domain.h;
```

Ισοδύναμα φορτία

- Function: **apply_equivalent_loads_linear_x(q_start, q_end, domain)**
- Τα ισοδύναμα φορτία αποθηκεύονται στον πίνακα Qji με διαστάσεις (nj × ni), ο οποίος και επιστρέφεται
- Δηλαδή, στη γραμμή j και στήλη i περιέχεται το ισοδύναμο φορτίο $q_{i,j}$.
- Αρχικά ο πίνακας έχει παντού τιμές NaN (not a number). Στο τέλος της συνάρτησης, τα NaN θα παραμείνουν μόνο στους κόμβους εκτός της πλάκας όπου δεν ορίζεται φορτίο.

```
21      %Equivalent nodal loads
22 -    Qji = NaN(nj, ni);
23 -    for I=1:nx
24 -        for J=1:ny
25 -            i = I+2;
26 -            j = J+2;
27 -            x = (I-1)*h;
28 -            Qji(j,i) = q_start + (q_end-q_start) * x/Lx;
29 -        end
30 -    end
```

Ισοδύναμα φορτία

- Function: **apply_equivalent_loads_linear_x(q_start, q_end, domain)**
- Οι δείκτες **I,J** αφορούν μόνο τους (**nx** × **ny**) πραγματικούς κόμβους της πλάκας
- Οι δείκτες **i,j** αφορούν όλους τους (**ni** × **nj**) κόμβους της πλάκας, πραγματικούς και βοηθητικούς
- Υπάρχουν 2 στρώσεις βοηθητικών κόμβων πριν την αρχή των πραγματικών, άρα $i = I + 2, j = J + 2$

```
21      %Equivalent nodal loads
22 -    Qji = NaN(nj, ni);
23 -    for I=1:nx
24 -        for J=1:ny
25 -            i = I+2;
26 -            j = J+2;
27 -            x = (I-1)*h;
28 -            Qji(j,i) = q_start + (q_end-q_start) * x/Lx;
29 -        end
30 -    end
```

Ισοδύναμα φορτία

- Function: **apply_equivalent_loads_linear_x(q_start, q_end, domain)**
- Οι σπουδαστές καλούνται να υλοποιήσουν τα functions:
  - **apply_equivalent_loads_linear_y(q_start, q_end, domain)**, για γραμμικά μεταβαλλόμενο φορτίο κατά y:

$$q(x,y)=q\_start+(q\_end-q\_start)\cdot y/Ly$$

  - **apply_equivalent_loads_constant(q0, domain)**, για ομοιόμορφο φορτίο q(x,y)=q0

Υπολογισμός και σχεδιασμός ροπών

- Function: **calc_Mx(i, j, Wji, domain)**
- Υπολογίζει τη ροπή M_x στον κόμβο (i, j).
- Παίρνει σαν input τον πίνακα **Wji** , ο οποίος έχει διαστάσεις (**nj** × **ni**), και περιέχει τις βυθίσεις στους πραγματικούς και βοηθητικούς κόμβους.
- Δηλαδή, στη γραμμή j και στήλη i περιέχεται η βύθιση $w_{i,j}$, άρα **Wji(j, i)**.
- Ο υπολογισμός γίνεται με τη βοήθεια του stencil

$$M_x = \frac{-D}{h^2}\begin{bmatrix} & \nu & \\ 1 & \boxed{-2-2\nu} & 1 \\ & \nu & \end{bmatrix} * \widetilde{W}_{i,j}$$

```
17      % Calculate the moment
18 -    sum = (-2-2*v)*Wji(j,i) + v*Wji(j-1,i) + v*Wji(j+1,i) + ...
19          Wji(j,i-1) + Wji(j,i+1);
20 -    Mx = -D / h^2 * sum ;
```

- Οι σπουδαστές καλούνται να υλοποιήσουν τις functions: **calc_My(...), calc_Mxy(...)**

Υπολογισμός και σχεδιασμός ροπών

- Function: **plot_Mx_parallel_to_x(y0, Wji, domain)**
- Σχεδιάζει τη ροπή M_x σε ευθεία y=y0 παράλληλα στον άξονα x.
- Δεν επιστρέφει κάποιο αποτέλεσμα. Απλά δημιουργεί ένα διάγραμμα.
- Αρχικά βρίσκει τα index (i,j) των κόμβων πάνω σε αυτή την ευθεία.
- Αυτά επιστρέφονται σε πίνακα **index2D** με διαστάσεις (n_points × 2), όπου n_points είναι το πλήθος των κόμβων πάνω στην ευθεία, η 1η στήλη είναι τα i τους και η 2η τα j τους.

```
14      % Find the (i, j) indices of the points parallel to x
15 -    index2D = find_nodes_with_y(y0, 0, Lx, domain);
16 -    n_points = size(index2D, 1);
```

Υπολογισμός και σχεδιασμός ροπών

- Function: **plot_Mx_parallel_to_x(y0, Wji, domain)**
- Έπειτα φτιάχνουμε vectors **X** και **Mx** που περιέχουν τις συντεταγμένες x και τις ροπές M_x των κόμβων αυτών.
- Χρησιμοποιούμε τον πίνακα **index2D** που πήραμε προηγουμένως για να βρούμε τα i,j κάθε κόμβου.
- Χρησιμοποιούμε τη function **calc_Mx(...)** που γράψαμε προηγουμένως για να υπολογίσουμε τη τιμή της ροπής M_x.

```
18      % Store the x-coordinate and Mx of these points into vectors X, Mx.
19 -    X = zeros(n_points, 1);
20 -    Mx = zeros(n_points, 1);
21 -    for k=1:n_points
22 -        i = index2D(k, 1);
23 -        j = index2D(k, 2);
24 -        I = i-2;
25 -        X(k, 1) = (I-1)*h;
26 -        Mx(k, 1) = calc_Mx(i, j, Wji, domain);
27 -    end
```

Υπολογισμός και σχεδιασμός ροπών

- Function: **plot_Mx_parallel_to_x(y0, Wji, domain)**
- Φτιάχνουμε νέο γράφημα (figure) και σχεδιάζουμε το διάγραμμα, δίνοντας τα vectors **X** → τιμές στον x, **Mx** → τιμές στον y.
- Δίνουμε τίτλους στο γράφημα και τους άξονες. Επίσης για ροπές (και τέμνουσες, βυθίσεις) τα θετικά είναι συνήθως προς τα κάτω).

```
29      % Plot the function
30 -    f = figure
31 -    plot(X, Mx);
32 -    title(['Mx(x,y=', num2str(y0), ')']);
33 -    xlabel('x')
34 -    ylabel('Mx')
35 -    f.CurrentAxes.YDir = 'Reverse'
```

- Οι σπουδαστές καλούνται να υλοποιήσουν τις functions:
  plot_Mx_parallel_to_y(...),
  plot_My_parallel_to_y(...), plot_My_parallel_to_y(...),
  plot_Mxy_parallel_to_y(...), plot_Mxy_parallel_to_y(...)

Γραμμικές εξισώσεις

- Function: **equations_internal_free (i, j, domain, qij)**

equations_internal_free.m

```
function [coefficients, rhs] = equations_internal_free(i, j, domain, qij)
%Creates FD equations for the equilibrium of an internal node i,j
```

- Δημιουργεί τη γραμμική εξίσωση για έναν αδέσμευτο εσωτερικό κόμβο **i,j**
- Το input **qij** είναι το ισοδύναμο φορτίο του κόμβου **i,j**
- Το output **coefficients** είναι πίνακας που περιέχει 1 γραμμή του πίνακα $[A]$.
  - Έχει διαστάσεις $(1 \times n_i \cdot nj)$
  - Αυτή η γραμμή περιέχει τους συντελεστές των stencil στις στήλες που αντιστοιχούν στους αγνώστους που w που εμπλέκονται στην εξίσωση
- Το output **rhs** είναι πίνακας που περιέχει 1 στοιχείο του διανύσματος $[b]$
  - Είναι και αυτό πίνακας με διαστάσεις (1×1)
  - Περιέχει το δεξί μέλος της εξίσωσης.

```
% 1 equation per node
coefficients = zeros(1, ni*nj);
rhs = zeros(1,1);
```

Γραμμικές εξισώσεις

- Function: **equations_internal_free (i, j, domain, qij)**
- Δημιουργεί τη γραμμική εξίσωση για έναν αδέσμευτο εσωτερικό κόμβο **i,j**
- Το output **coefficients** είναι μία γραμμή του πίνακα $[A]$.
- Το output **rhs** είναι ένα στοιχείο του διανύσματος $[b]$
- Π.χ. για τον $i = 4, j = 4 \Rightarrow s = 28$, η εξίσωση σαν stencil είναι

$$\begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & \boxed{20} & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix} * \begin{bmatrix} & & w_{12} & & \\ & w_{19} & w_{20} & w_{21} & \\ w_{26} & w_{27} & \boxed{w_{28}} & w_{29} & w_{30} \\ & w_{35} & w_{36} & w_{37} & \\ & & w_{44} & & \end{bmatrix} = 20.67 \cdot 10^{-3}$$

οπότε οι αντίστοιχη γραμμή του $[A]$

12		19	20	21		26	27	28	29	30		35	36	37		44	
[…	1	…	2	−8	2	…	1	−8	20	−8	1	…	2	−8	2	…	1	…]

Γραμμικές εξισώσεις

- Function: **equations_internal_free (i, j, domain, qij)**
- Βρίσκουμε το 1D index απο το 2D index, δηλαδή το **s** από τα **i,j**

$$s = (j-1)\cdot n_i + i$$

- Αυτά τα **s** δείχνουν τις στήλες του [A]!!!
- Υπενθύμιση:
  - c → center (i,j)
  - n → north (i,j-1)
  - w → west (i-1,j)
  - nn → north-north (i,j-2)
  - ne → north-east (i-1,j+1)

  Κτλ.

```
26      % 1D index of each node on the stencil
27 -    s_c = (j-1)*ni+i;
28
29 -    s_n = (j-2)*ni+i;
30 -    s_s = j*ni+i;
31 -    s_w = (j-1)*ni+i-1;
32 -    s_e = (j-1)*ni+i+1;
33
34 -    s_nw = (j-2)*ni+i-1;
35 -    s_ne = (j-2)*ni+i+1;
36 -    s_sw = j*ni+i-1;
37 -    s_se = j*ni+i+1;
38
39 -    s_nn = (j-3)*ni+i;
40 -    s_ss = (j+1)*ni+i;
41 -    s_ww = (j-1)*ni+i-2;
42 -    s_ee = (j-1)*ni+i+2;
```

Γραμμικές εξισώσεις

- Function: **equations_internal_free (i, j, domain, qij)**
- Γράφουμε τους συντελεστές του stencil στις αντίστοιχες στήλες που καθορίζονται απο τα **s**.

```
44      % Left-hand-side(s) (coefficients)
45 -    coefficients(1,s_nn) = 1;
46 -    coefficients(1,s_nw) = 2;
47 -    coefficients(1,s_n) = -8;
48 -    coefficients(1,s_ne) = 2;
49 -    coefficients(1,s_ww) = 1;
50 -    coefficients(1,s_w) = -8;
51 -    coefficients(1,s_c) = 20;
52 -    coefficients(1,s_e) = -8;
53 -    coefficients(1,s_ee) = 1;
54 -    coefficients(1,s_sw) = 2;
55 -    coefficients(1,s_s) = -8;
56 -    coefficients(1,s_se) = 2;
57 -    coefficients(1,s_ss) = 1;
```

- Γράφουμε το στοιχείο για το δεξί μέλος της εξίσωσης

```
59      % Right-hand-side(s) (constants) of the equation(s)
60 -    rhs(1,1) = qij*h^4/D;
```

Γραμμικές εξισώσεις

- Function: **equations_boundary_fixed _y(i, j, domain, qij)**

equations_boundary_fixed_y.m

```
1  function [coefficients, rhs] = equations_boundary_fixed_y(i, j, domain, qij)
2  %Creates FD equations for a boundary node i,j on a fixed (clamped) edge
3  %   perpendicular to axis y.
```

- Δημιουργεί τις γραμμικές εξίσωσεις για ένα συνοριακό κόμβο **i,j** σε πακτωμένη παρειά κάθετη στον άξονα y.
- Το output **coefficients** είναι πίνακας που περιέχει 2 γραμμές του πίνακα $[A]$.
  - Έχει διαστάσεις $(2 \times n_i \cdot nj)$
  - Αυτές οι γραμμές περιέχουν τους συντελεστές των stencil στις στήλες που αντιστοιχούν στους εμπλεκόμενους αγνώστους w
- Το output **rhs** είναι πίνακας που περιέχει 2 στοιχεία του διανύσματος $[b]$
  - Είναι και αυτό πίνακας με διαστάσεις (2×1)
  - Περιέχει τα δεξιά μέλη των εξισώσεων.

```
21    % 2 equations per node
22 -  coefficients = zeros(2, ni*nj);
23 -  rhs = zeros(2,1);
```

Γραμμικές εξισώσεις

- Function: **equations_boundary_fixed _y(i, j, domain, qij)**
- Βρίσκουμε το 1D index $s = (j-1)\cdot n_i + i$. Το s δείχνει τις στήλες του $[A]$

```
25      % 1D index of each node on the stencil
26 -    s_c = (j-1)*ni+i;
27 -    s_n = (j-2)*ni+i;
28 -    s_s = j*ni+i;
```

- Το stencil για την 1η εξίσωση $(w = 0)$ είναι

$$\left[\boxed{1}\right] * \widetilde{W}_{i,j} = 0$$

```
30      % 1st equation: w = 0
31 -    coefficients(1, s_c) = 1;
32 -    rhs(1,1) = 0;
```

- Το stencil για την 2η εξίσωση $\left(\frac{\partial w}{\partial y} = 0\right)$ είναι

$$\begin{bmatrix} -1 \\ \square \\ +1 \end{bmatrix} * \widetilde{W}_{i,j} = 0$$

```
34      % 2nd equation: dx/dy = 0
35 -    coefficients(2,s_n) = -1;
36 -    coefficients(2,s_s) = 1;
37 -    rhs(2,1) = 0;
```

Γραμμικές εξισώσεις

- Function: **equations_boundary_fixed _y(i, j, domain, qij)**

equations_boundary_fixed_y.m

```
function [coefficients, rhs] = equations_boundary_fixed_y(i, j, domain, qij)
%Creates FD equations for a boundary node i,j on a fixed (clamped) edge
%   perpendicular to axis y.
```

- Παρατηρώ ότι δεν χρειάζεται το input **qij.**
- Είναι όμως υποχρεωτικό όλες αυτές οι functions (**equations_internal_...**, **equations_boundary_...**, **equations_corner_...**) να έχουν ακριβώς το ίδιο input και output.
- Αλλιώς δεν θα τρέξει το πρόγραμμα!!!
- Αντίστοιχα υλοποιούνται οι functions:
  - **equations_boundary_free_y(…)** κόμβος σε ελεύθερη παρειά κάθετα στον y: δίνει 3 εξισώσεις
  - **equations_corner_fixed(…)** κόμβος σε γωνία με πάκτωση: δίνει 3 εξισώσεις.
- Οι σπουδαστές καλούνται να υλοποιήσουν τις υπόλοιπες functions που δημιουργούν εξισώσεις.