ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2016-2017

Μάθημα: Εφαρμογές Η/Υ

Δευτέρα, 7/11/2016

Διδάσκοντες:
Ν.Δ. Λαγαρός (Επικ. Καθηγητής), Αθ. Στάμος (ΕΔΙΠ), Χ. Φραγκουδάκης (ΕΔΙΠ)

Παραδείγματα για την 6η παράδοση – Δημιουργία και χειρισμός αριθμητικών δεδομένων

1. Υπολογισμός ολικού μητρώου δυσκαψίας από επί μέρους μητρώα δυσκαμψίας

Το μητρώο δυσκαμψίας ενός στατικού φορέα Kol συνδέει τις μετατοπίσεις των κόμβων Uol του φορέα με τις εξωτερικές δυνάμεις Fol (συμπεριλαμβανομένων των αντιδράσεων) που ασκούνται στους κόμβους:

$$[F_{ol}]=[K_{ol}]\cdot[U_{ol}]$$

Αν υπάρχουν N κόμβοι στο φορέα τότε το μητρώο δυσκαμψίας Kol έχει διαστάσεις N×N και τα μητρώα δυνάμεων και μετατοπίσεων έχουν διαστάσεις N×1. Το μητρώο δυσκαμψίας Kol το οποίο ονομάζουμε ολικό μητρώο δυσκαμψίας, συντίθεται από τα μητρώα δυσκαμψίας των επί μέρους στοιχείων του φορέα τα οποία ονομάζουμε επί μέρους μητρώα δυσκαμψίας. Τα επί μέρους μητρώα δυσκαμψίας συνδέουν τις μετατοπίσεις των κόμβων U ενός στοιχείου του φορέα με τις δυνάμεις f που ασκούνται από το φορέα στο στοιχείο σε στους κόμβους του στοιχείου:

$$[f]=[K]\cdot[U]$$

Σε ένα στατικό φορέα που αποτελείται από υποστυλώματα, το ένα πάνω από το άλλο, τα υποστυλώματα δρουν ως ράβδοι σε μία διάσταση. Το επί μέρους μητρώο δυσκαμψίας ράβδου σε μία διάσταση που συνδέει τον κόμβο i με τον κόμβο j δίνεται:

$$[K]=\frac{AE}{L}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$$

όπου L, A, E το μήκος, το εμβαδόν διατομής και το μέτρο ελαστικότητας της ράβδου αντίστοιχα.
Για να κατασκευαστεί το ολικό μητρώο δυσκαμψίας [Kol], αυτό αρχικά δημιουργείται με μηδενικά στοιχεία, και στη συνέχεια τα 4 στοιχεία του κάθε επί μέρους μητρώου K_{11}, K_{12}, K_{21}, K_{22} προστίθενται αντίστοιχα στα 4 στοιχεία του ολικού μητρώου Kol_{ii}, Kol_{ij}, Kol_{ji}, Kol_{jj}. όπου i, j οι κόμβοι που συνδέει η ράβδος.

Με βάση τα παραπάνω να συνταχθεί πρόγραμμα σε python το οποίο:
1. Να διαβάζει από το αρχείο k.txt τα στοιχεία L, A, E, i, j των υποστυλωμάτων (ένα υποστύλωμα ανά σειρά αρχείου).
2. Να υπολογίζει το πλήθος N των κόμβων.
3. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο k.txt το ολικό μητρώο δυσκαμψίας.
Δοκιμάστε το πρόγραμμά σας με το αρχείο k.txt:

```
3 0.09 200e9 0 1
3 0.16 200e9 1 2
2 0.12 200e9 2 3
2 0.09 200e9 3 4
```

Λύση

```
# -*- coding: iso-8859-7 -*-
from numpy import loadtxt, savetxt, zeros, ones, max

def ypolKol(s):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
    N = max(s[:, 3:5]) + 1
```

```
    Kol = zeros((N, N))
    temp = ones((2, 2))
    temp[0, 1] = temp[1, 0] = -1
    for L, A, E, i, j in s:
        K = A*E/L * temp
        Kol[i, i] += K[0, 0]
        Kol[i, j] += K[0, 1]
        Kol[j, i] += K[1, 0]
        Kol[j, j] += K[1, 1]
    return Kol

def pyMain():
    "Start here."
    s = loadtxt("k.txt")
    Kol = ypolKol(s)
    savetxt("kol.txt", Kol, fmt="%14.6e")

pyMain()
```

Προσοχή: η συνάρτηση max() που έρχεται μαζί με την python (builtin) δεν μπορεί να χειριστεί πολυδιάστατα μητρώα και για αυτό πρέπει να χρησιμοποιηθεί η συνάρτηση max() που είναι ορισμένη στη βιβλιοθήκη numpy.

2. Υπολογισμός ολικού μητρώου δυσκαμψίας επιπέδου δικτυώματος

Το επίπεδο δικτύωμα αποτελείται από ράβδους σε δύο διαστάσεις. Κάθε κόμβος του δικτυώματος έχει δύο μετατοπίσεις και σε κάθε κόμβο ασκούνται δύο δυνάμεις, μία σε διεύθυνση x και μία σε διεύθυνση y. Κάθε ράβδος συνδέει δύο κόμβους και συνεπώς οι δύο κόμβοι έχουν 4 μετατοπίσεις και 4 δυνάμεις. Έτσι το μητρώο δυσκαμψίας του ράβδου συνδέει 4 μετατοπίσεις με 4 ράβδους και συνεπώς έχει διαστάσεις 4×4. Αν η ράβδος σχηματίζει γωνία θ με τον οριζόντιο άξονα x και συνδέει τον κόμβο i με τον κόμβο j το επί μέρους μητρώο δυσκαμψίας της δίνεται:

$$[K]=\frac{AE}{L}\begin{bmatrix} c^2 & & \text{συμ} & \\ cs & s^2 & & \\ -c^2 & -cs & c^2 & \\ -cs & -s^2 & cs & s^2 \end{bmatrix}$$

όπου L, A, E το μήκος, το εμβαδόν διατομής και το μέτρο ελαστικότητας αντίστοιχα και c=cos(θ), s=sin(θ). Οι σειρές 0, 1 και οι στήλες 0, 1 του επί μέρους μητρώου αντιστοιχούν στον κόμβο i (η 0 για τη διεύθυνση x και η 1 για τη διεύθυνση y) και οι γραμμές 2, 3 και οι στήλες 2, 3 στον κόμβο j (η 2 για τη διεύθυνση x και η 3 για τη διεύθυνση y) του δικτυώματος.

Κατ' αντιστοιχία με τη ράβδο, αν υπάρχουν N κόμβοι στο επίπεδο δικτύωμα το ολικό μητρώο δυσκαμψίας [Kol] έχει διαστάσεις 2N×2N. Οι σειρές 2i και 2i+1 και οι στήλες 2i και 2i+1 αντιστοιχούν στον κόμβο i και οι γραμμές 2j, 2j+1 και οι στήλες 2j, 2j+1 στον κόμβο j.

Για να κατασκευαστεί το ολικό μητρώο δυσκαμψίας [Kol], αρχικά το επί μέρους μητρώο κάθε ράβδου χωρίζεται συμμετρικά σε 4 υπομητρώα διαστάσεων 2×2, πάνω αριστερά, πάνω δεξιά, κάτω αριστερά και κάτω δεξιά:

$$[K]=\frac{AE}{L}\left[\begin{array}{cc||cc} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ \hline -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{array}\right]$$

Το ολικό μητρώο δυσκαμψίας επίσης χωρίζεται σε υπομητρώα διαστάσεων 2×2, και τα υπομητρώα του επί μέρους αντιστοιχούν σε υπομητρώα του ολικού ως εξής:

Το πάνω αριστερά αντιστοιχεί στις γραμμές 2i και 2i+1 και στις στήλες 2i και 2i+1 του ολικού
Το πάνω δεξιά αντιστοιχεί στις γραμμές 2i και 2i+1 και στις στήλες 2j και 2j+1 του ολικού
Το κάτω αριστερά αντιστοιχεί στις γραμμές 2j και 2j+1 και στις στήλες 2i και 2i+1 του ολικού
Το κάτω δεξιά αντιστοιχεί στις γραμμές 2j και 2j+1 και στις στήλες 2j και 2j+1 του ολικού

Έτσι το ολικό μητρώο δυσκαμψίας [Kol] αρχικά δημιουργείται με μηδενικά στοιχεία. Στη συνέχεια το επί

μέρους μητρώο κάθε ράβδου χωρίζεται συμμετρικά σε 4 υπομητρώα διαστάσεων 2×2, και το κάθε επιμέρους μητρώο προστίθεται στο αντίστοιχο υπομητρώο του ολικού μητρώου.

Με βάση τα παραπάνω να συνταχθεί πρόγραμμα σε python το οποίο:
1. Να διαβάζει από το αρχείο k.txt τα στοιχεία L, A, E, θ (μοίρες), i, j των ράβδων (μία ράβδος ανά σειρά αρχείου).
2. Να υπολογίζει το πλήθος N των κόμβων.
3. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο kol.txt το ολικό μητρώο δυσκαμψίας.
Δοκιμάστε το πρόγραμμά σας με το αρχείο k.txt:

```
3 0.09 200e9 0 0 1
4 0.16 200e9 90 0 2
5 0.12 200e9 143.13 1 2
```

Λύση

Το επί μέρους μητρώο δυσκαμψίας έχει αρκετούς υπολογισμούς και έτσι θα συνταχθεί συνάρτηση που το υπολογίζει. Αρχικά θα υπολογιστεί το κάτω τριγωνικό μέρος του και στην συνέχεια θα αντιγραφεί συμμετρικά στο άνω τριγωνικό.

```
# -*- coding: iso-8859-7 -*-
from numpy import loadtxt, savetxt, zeros, ones, max, deg2rad, cos, sin


def barMat(L, A, E, theta):
    "Computes the axial stiffness matrix of a bar in 2D."
    theta = deg2rad(theta)
    c = cos(theta)
    s = sin(theta)
    c2 = c*c
    s2 = s*s
    cs = c*s
    k = E*A/L
    K = zeros((4, 4))

    K[0, 0] = k * c2

    K[1, 0] = k * cs
    K[1, 1] = k * s2

    K[2, 0] = -K[0, 0]
    K[2, 1] = -K[1, 0]
    K[2, 2] =  K[0, 0]

    K[3, 0] = -K[1, 0]
    K[3, 1] = -K[1, 1]
    K[3, 2] =  K[1, 0]
    K[3, 3] =  K[1, 1]

    for i in range(3):
        K[i, i+1:] = K[i+1:, i]
    return K


def ypolKol(s):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
    N = max(s[:, 4:6]) + 1
    Kol = zeros((2*N, 2*N))
    for L, A, E, theta, i, j in s:
        K = barMat(L, A, E, theta)
        i *= 2
        j *= 2
        Kol[i:i+2, i:i+2] += K[0:2, 0:2]
        Kol[i:i+2, j:j+2] += K[0:2, 2:4]
        Kol[j:j+2, i:i+2] += K[2:4, 0:2]
        Kol[j:j+2, j:j+2] += K[2:4, 2:4]
    return Kol
```

```
def pyMain():
    "Start here."
    s = loadtxt("k.txt")
    Kol = ypolKol(s)
    savetxt("kol.txt", Kol, fmt="%14.6e")

pyMain()
```

Από αυτό το παράδειγμα μπορεί κάποιος να εκτιμήσει τον τελεστή +=. Αν δεν υπήρχε ο κώδικας για το ολικό μητρώο θα γινόταν:

```
Kol[i:i+2, i:i+2] = Kol[i:i+2, i:i+2] + K[0:2, 0:2]
Kol[i:i+2, j:j+2] = Kol[i:i+2, j:j+2] + K[0:2, 2:4]
Kol[j:j+2, i:i+2] = Kol[j:j+2, i:i+2] + K[2:4, 0:2]
Kol[j:j+2, j:j+2] = Kol[j:j+2, j:j+2] + K[2:4, 2:4]
```

Αυτό ο κώδικας είναι πιο επίπονος στην πληκτρολόγηση, δημιουργεί πιθανότητες λάθους και δυσχεραίνει την αναγνωσιμότητα.

3. Υπολογισμός μετατοπίσεων επιπέδου δικτυώματος

Το μητρώο δυσκαμψίας ενός επιπέδου δικτυώματος Kol συνδέει τις μετατοπίσεις των κόμβων Uol του δικτυώματος με τις εξωτερικές δυνάμεις Fol (συμπεριλαμβανομένων των αντιδράσεων) που ασκούνται στους κόμβους:

$$[F_{ol}]=[K_{ol}]\cdot[U_{ol}]$$

Για παράδειγμα θεωρώντας δικτύωμα με 4 κόμβους η σχέση γίνεται:

$$\begin{bmatrix} F_{1x} \\ F_{1y} \\ F_{2x} \\ F_{2y} \\ F_{3x} \\ U_{3y} \\ F_{4x} \\ U_{4y} \end{bmatrix} = \left[K_{ol}\right] \cdot \begin{bmatrix} U_{1x} \\ U_{1y} \\ U_{2x} \\ U_{2y} \\ U_{3x} \\ U_{3y} \\ U_{4x} \\ U_{4y} \end{bmatrix}$$

Αν όλες οι δυνάμεις ήταν γνωστές η παραπάνω σχέση θα μπορούσε να επιλυθεί ως προς [Uol]. Όμως αποδεικνύεται ότι το μητρώο [Kol] είναι ιδιάζον. Αυτό οφείλεται στο ότι δεν έχουμε στηρίξει το δικτύωμα στο έδαφος ή τουλάχιστον οι στηρίξεις του δικτυώματος δεν έχουν ληφθεί υπόψη στην παραπάνω σχέση. Επιπλέον οι αντιδράσεις στις στηρίξεις δεν είναι γνωστές. Αν το δικτύωμα στηρίζεται με άρθρωση στον κόμβο k τότε οι μετατοπίσεις σε αυτόν τον κόμβο είναι μηδενικές:

$$Uol_{k,x} = Uol_{k,y} = 0$$

ή με αριθμητικούς δείκτες:

$$Uol_{2k} = Uol_{2k+1} = 0$$

Αν στο προηγούμενο παράδειγμα θεωρήσουμε αρθρώσεις στους κόμβους 1 και 3, η σχέση γίνεται:

$$\begin{bmatrix} F_{1x} \\ F_{1y} \\ F_{2x} \\ F_{2y} \\ F_{3x} \\ F_{3y} \\ F_{4x} \\ F_{4y} \end{bmatrix} = \begin{bmatrix} Kol_{11} & Kol_{12} & Kol_{13} & Kol_{14} & Kol_{15} & Kol_{16} & Kol_{17} & Kol_{18} \\ Kol_{21} & Kol_{22} & Kol_{23} & Kol_{24} & Kol_{25} & Kol_{26} & Kol_{27} & Kol_{28} \\ Kol_{31} & Kol_{32} & Kol_{33} & Kol_{34} & Kol_{35} & Kol_{36} & Kol_{37} & Kol_{38} \\ Kol_{41} & Kol_{42} & Kol_{43} & Kol_{44} & Kol_{45} & Kol_{46} & Kol_{47} & Kol_{48} \\ Kol_{51} & Kol_{52} & Kol_{53} & Kol_{54} & Kol_{55} & Kol_{56} & Kol_{57} & Kol_{58} \\ Kol_{61} & Kol_{62} & Kol_{63} & Kol_{64} & Kol_{65} & Kol_{66} & Kol_{67} & Kol_{68} \\ Kol_{71} & Kol_{72} & Kol_{73} & Kol_{74} & Kol_{75} & Kol_{76} & Kol_{77} & Kol_{78} \\ Kol_{81} & Kol_{82} & Kol_{83} & Kol_{84} & Kol_{85} & Kol_{86} & Kol_{87} & Kol_{88} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ U_{2x} \\ U_{2y} \\ 0 \\ 0 \\ U_{4x} \\ U_{4y} \end{bmatrix}$$

Σε αυτή τη σχέση έχουμε 8 εξισώσεις και 4 άγνωστες μετατοπίσεις. Για να επιλυθεί αυτό το σύστημα μπορούμε να αφαιρέσουμε 4 από αυτές (που αποδεικνύεται ότι είναι ταυτότητες). Αφαιρούμε τις εξισώσεις

που αντιστοιχούν στις μηδενικές στηρίξεις, διότι έτσι αφαιρούνται και οι αντίστοιχες δυνάμεις, F_{1x}, F_{1y}, F_{3x}, F_{3y} που είναι οι αντιδράσεις στις στηρίξεις και είναι άγνωστες:

$$\begin{bmatrix} F_{2x} \\ F_{2y} \\ F_{3x} \\ F_{3y} \end{bmatrix} = \begin{bmatrix} Kol_{31} & Kol_{32} & Kol_{33} & Kol_{34} & Kol_{35} & Kol_{36} & Kol_{37} & Kol_{38} \\ Kol_{41} & Kol_{42} & Kol_{43} & Kol_{44} & Kol_{45} & Kol_{46} & Kol_{47} & Kol_{48} \\ Kol_{71} & Kol_{72} & Kol_{73} & Kol_{74} & Kol_{75} & Kol_{76} & Kol_{77} & Kol_{78} \\ Kol_{81} & Kol_{82} & Kol_{83} & Kol_{84} & Kol_{85} & Kol_{86} & Kol_{87} & Kol_{88} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ U_{2x} \\ U_{2y} \\ 0 \\ 0 \\ U_{4x} \\ U_{4y} \end{bmatrix}$$

Στο δεξιό μέλος αυτής της σχέσης οι στήλες 1, 2, 5, 6 κάθε εξίσωσης πολλαπλασιάζονται επί μηδέν. Για παράδειγμα αν εκτελέσουμε το μητρωικό πολλαπλασιασμό για την τρίτη εξίσωση:

$$F_{4x} = Kol_{71}\cdot 0 + Kol_{72}\cdot 0 + Kol_{73}\cdot U_{2x} + Kol_{74}U_{2y} + Kol_{75}\cdot 0 + Kol_{76}\cdot 0 + Kol_{77}\cdot U_{4x} + Kol_{78}\cdot U_{4y}$$ ή

$$F_{4x} = Kol_{73}\cdot U_{2x} + Kol_{74}U_{2y} + Kol_{77}\cdot U_{4x} + Kol_{78}\cdot U_{4y}$$

Συνεπώς οι στήλες 1, 2, 5, 6 δεν συνεισφέρουν τίποτα και απαλείφονται στη μητρωική σχέση:

$$\begin{bmatrix} F_{2x} \\ F_{2y} \\ F_{3x} \\ F_{3y} \end{bmatrix} = \begin{bmatrix} Kol_{33} & Kol_{34} & Kol_{37} & Kol_{38} \\ Kol_{43} & Kol_{44} & Kol_{47} & Kol_{48} \\ Kol_{73} & Kol_{74} & Kol_{77} & Kol_{78} \\ Kol_{83} & Kol_{84} & Kol_{87} & Kol_{88} \end{bmatrix} \cdot \begin{bmatrix} U_{2x} \\ U_{2y} \\ U_{4x} \\ U_{4y} \end{bmatrix}$$ ή

$$[F_m] = [K_m]\cdot[U_m]$$

όπου $[F_m]$, $[K_m]$, $[U_m]$, είναι τα μειωμένα μητρώα, δηλαδή τα ολικά με απαλοιφή των γραμμών και στηλών που αντιστοιχούν σε στηρίξεις. Η σχέση αυτή μπορεί να επιλυθεί ως προς U_m δηλαδή ως προς τις άγνωστες μετατοπίσεις. Τέλος συμπληρώνοντας το μητρώο $[U_m]$ με τις μηδενικές μετατοπίσεις στις στηρίξεις έχουμε το ολικό μητρώο των μετατοπίσεων $[U_{ol}]$.

Με βάση τα παραπάνω να συνταχθεί πρόγραμμα σε python το οποίο:
1. Να διαβάζει από το αρχείο k.txt τα στοιχεία L, A, E, θ (μοίρες), i, j των ράβδων (μία ράβδος ανά σειρά αρχείου).
2. Να διαβάζει από το αρχείο s.txt τους αύξοντες αριθμούς των κόμβων που στηρίζονται με άρθρωση.
3. Να διαβάζει από το αρχείο f.txt τα φορτία το ολικό μητρώο εξωτερικών φορτίων F_{ol}, το οποίο θεωρούμε ότι έχει μηδενικά εξωτερικά φορτία στις στηρίξεις.
4. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο kol.txt το ολικό μητρώο δυσκαμψίας.
5. Να υπολογίζει το μειωμένα μητρώα μετατοπίσεων και δυνάμεων.
6. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο uol.txt το ολικό μητρώο μετατοπίσεων.
Δοκιμάστε το πρόγραμμά σας με:
α) το αρχείο k.txt:

```
3 0.09 200e9 0 0 1
4 0.16 200e9 90 0 2
5 0.12 200e9 143.13 1 2
```

β) το αρχείο s.txt:

```
0
1
```

γ) το αρχείο f.txt:

```
0
0
0
0
100000
0
```

Λύση

Η συνάρτηση numpy.linalg.solve(A, B) επιλύει το μητρωικό σύστημα [A][x]=[B] και επιστρέφει τη λύση [x]. Για τον υπολογισμό του ολικού μητρώου μετατοπίσεων μηδενίζουμε αρχικά όλο το μητρώο και στη συνέχεια στις θέσεις του αρχείου που αντιστοιχούν στο μειωμένο μητρώο τοποθετούμε το μειωμένο μητρώο.

```
import numpy as np

def barMat(L, A, E, theta):
    "Computes the axial stiffness matrix of a bar in 2D."
    theta = np.deg2rad(theta)
    c = np.cos(theta)
    s = np.sin(theta)
    k = E*A/L
    K = np.zeros((4, 4))

    K[0, 0] = k * c**2

    K[1, 0] = k * c*s
    K[1, 1] = k * s**2

    K[2, 0] = -K[0, 0]
    K[2, 1] = -K[1, 0]
    K[2, 2] =  K[0, 0]

    K[3, 0] = -K[1, 0]
    K[3, 1] = -K[1, 1]
    K[3, 2] =  K[1, 0]
    K[3, 3] =  K[1, 1]

    for i in range(3):
        K[i, i+1:] = K[i+1:, i]
    return K

def ypolKol(s):
    "Υπολογισμός ολικού μητρώου."
    N = np.max(s[:, 4:6]) + 1
    Kol = np.zeros((2*N, 2*N))
    for L, A, E, theta, i, j in s:
        K = barMat(L, A, E, theta)
        i *= 2
        j *= 2
        Kol[i:i+2, i:i+2] += K[0:2, 0:2]
        Kol[i:i+2, j:j+2] += K[0:2, 2:4]
        Kol[j:j+2, i:i+2] += K[2:4, 0:2]
        Kol[j:j+2, j:j+2] += K[2:4, 2:4]
    return Kol

def reduceKol(Fol, jst, Kol):
    "Υπολογισμός μειωμένων μητρώων."
    N = len(Kol)
    todel = set()
    for j in jst:
        todel.add(2*j)
        todel.add(2*j+1)
    indexes = np.array([j for j in range(N) if j not in todel])
    i = indexes.reshape(1, len(indexes))
    j = indexes.reshape(len(indexes), 1)
    return indexes, Fol[indexes], Kol[i, j]

def pyMain():
    "Start here."
    stoix = np.loadtxt("k.txt")
    jst = np.loadtxt("s.txt")
    Fol = np.loadtxt("f.txt")
    Kol = ypolKol(stoix)
    np.savetxt("kol.txt", Kol, fmt="%14.6e")
    indexes, Fm, Km = reduceKol(Fol, jst, Kol)
    Um = np.linalg.solve(Km, Fm)
    Uol = np.zeros((len(Kol),))
    Uol[indexes] = Um
```

```
    np.savetxt("uol.txt", Uol, fmt="%14.6e")

pyMain()
```

<u>Λύση με αριθμητική ευστάθεια</u>

Στη σχέση

$$[F_{ol}]=[K_{ol}]\cdot[U_{ol}]$$

το μητρώο δυσκαμψίας περιέχει πολύ μεγάλους αριθμούς (της τάξης του 10^8) ενώ το μητρώο μετατοπίσεων έχει πολύ μικρούς αριθμός της τάξης του 10^{-5}. Η επίλυση αυτού του συστήματος με πεπερασμένο πλήθος σημαντικών ψηφίων εγκυμονεί κίνδυνο απώλειας ακριβείας (σε μεγάλα δικτυώματα). Για να αποφύγουμε αυτόν τον κίνδυνο υπολογίζουμε τη μέγιστη δυσκαμψία k_{max} όλων των ράβδων:

$$k_{max}=max\left\{\frac{A_i E_i}{L_i}\right\}$$

και στη συνέχεια πολλαπλασιάζουμε και διαιρούμε με k_{max} το δεξί μέλος της παραπάνω σχέσης:

$$[F_{ol}]=k_{max}\frac{1}{k_{max}}[K_{ol}]\cdot[U_{ol}]=\left(\frac{1}{k_{max}}[K_{ol}]\right)\cdot\left(k_{max}[U_{ol}]\right)\ \Rightarrow\ [F_{ol}]=[K^*_{ol}]\cdot[U^*_{ol}]$$

Δηλαδή διαιρούμε το μητρώο K_{ol} με ένα μεγάλο νούμερο και πολλαπλασιάζουμε το U_{ol} με ένα μεγάλο νούμερο και έτσι τα μητρώα K^*_{ol} και U^*_{ol} έχουν νούμερα που είναι πολύ πιο κοντά στη μονάδα και το αντίστοιχο σύστημα μπορεί να επιλυθεί χωρίς κίνδυνο απώλειας ακριβείας. Στη συνέχεια για τον υπολογισμό των πραγματικών μετατοπίσεων διαιρούμε το μητρώο U^*_{ol} με το k_{max}.

```
import numpy as np

def barMat(L, A, E, theta, kmax):
    "Computes the axial stiffness matrix of a bar in 2D."
    theta = np.deg2rad(theta)
    c = np.cos(theta)
    s = np.sin(theta)
    k = E*A/L / kmax
    K = np.zeros((4, 4))

    K[0, 0] = k * c**2

    K[1, 0] = k * c*s
    K[1, 1] = k * s**2

    K[2, 0] = -K[0, 0]
    K[2, 1] = -K[1, 0]
    K[2, 2] =  K[0, 0]

    K[3, 0] = -K[1, 0]
    K[3, 1] = -K[1, 1]
    K[3, 2] =  K[1, 0]
    K[3, 3] =  K[1, 1]

    for i in range(3):
        K[i, i+1:] = K[i+1:, i]
    return K

def ypolKol(s):
    "Υπολογισμός ολικού μητρώου."
    kmax = max([A*E/L for L, A, E, theta, i, j in s])
    N = np.max(s[:, 4:6]) + 1
    Kol = np.zeros((2*N, 2*N))
    for L, A, E, theta, i, j in s:
        K = barMat(L, A, E, theta, kmax)
        i *= 2
        j *= 2
        Kol[i:i+2, i:i+2] += K[0:2, 0:2]
        Kol[i:i+2, j:j+2] += K[0:2, 2:4]
        Kol[j:j+2, i:i+2] += K[2:4, 0:2]
        Kol[j:j+2, j:j+2] += K[2:4, 2:4]
    return kmax, Kol

def reduceKol(Fol, jst, Kol):
```

```
    "Υπολογισμός μειωμένων μητρώων."
    N = len(Kol)
    todel = set()
    for j in jst:
        todel.add(2*j)
        todel.add(2*j+1)
    indexes = np.array([j for j in range(N) if j not in todel])
    i = indexes.reshape(1, len(indexes))
    j = indexes.reshape(len(indexes), 1)
    return indexes, Fol[indexes], Kol[i, j]

def pyMain():
    "Start here."
    stoix = np.loadtxt("k.txt")
    jst = np.loadtxt("s.txt")
    Fol = np.loadtxt("f.txt")
    kmax, Kol = ypolKol(stoix)
    np.savetxt("kol.txt", Kol, fmt="%14.6e")
    indexes, Fm, Km = reduceKol(Fol, jst, Kol)
    Um = np.linalg.solve(Km, Fm)
    Uols = np.zeros((len(Kol),))
    Uols[indexes] = Um
    Uol = Uols      / kmax
    np.savetxt("uol.txt", Uol, fmt="%14.6e")

pyMain()
```

4. Υπολογισμός αντιδράσεων και έλεγχος προγράμματος

Να επεκταθεί το πρόγραμμα του προηγούμενο παραδείγματος έτσι ώστε:
1. Να υπολογίζει τις αντιδράσεις στις στηρίξεις και να τις τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο ant.txt.

Λύση

Το μητρώο όλων των εξωτερικών δυνάμεων υπολογίζεται από της σχέση:

$$[F_{ol}]=[K^{*}_{ol}]\cdot[U^{*}_{ol}]$$

Από αυτές αντιδράσεις είναι οι δυνάμεις που αντιστοιχούν στις στηρίξεις. Οι υπόλοιπες είναι τα εξωτερικά φορτία τα οποία πρέπει να συμφωνούν με τα εξωτερικά φορτία που δίνονται στο αρχείο δεδομένων f.txt. Παρακάτω δίνεται μόνο ο νέος κώδικας: η νέα συνάρτηση `ypolAnt()` και η συνάρτηση `pyMain()` που έχει τώρα και την κλήση της `ypolAnt()`.

```
def ypolAnt(Fol, Kol, Uols, jst, indexes):
    "Υπολογισμός αντιδράσεων και έλεγχος αποτελεσμάτων."
    N = len(Kol)
    temp = []
    for j in jst:
        temp.append(2*j)
        temp.append(2*j+1)
    indexes2 = np.array(temp)
    F = np.dot(Kol, Uols)
    np.savetxt("ant.txt", F[indexes2], fmt="%14.6e")

    for i in indexes:
         assert np.fabs(Fol[i]-F[i]) < 0.1, "Programm error: external and computed force
do not agree"


def pyMain():
    "Start here."
    stoix = np.loadtxt("k.txt")
    jst = np.loadtxt("s.txt", np.int)
    Fol = np.loadtxt("f.txt")
    kmax, Kol = ypolKol(stoix)
    np.savetxt("kol.txt", Kol, fmt="%14.6e")
    indexes, Fm, Km = reduceKol(Fol, jst, Kol)
    Um = np.linalg.solve(Km, Fm)
```

```
Uols = np.zeros((len(Kol),))
Uols[indexes] = Um
Uol = Uols / kmax
np.savetxt("uol.txt", Uol, fmt="%14.6e")
ypolAnt(Fol, Kol, Uols, jst, indexes)
```

Το μητρώο `jst` πρέπει να αποτελείται από ακεραίους έτσι ώστε και το μητρώο `indexes2` να αποτελείται από ακεραίους για να μπορεί να χρησιμοποιηθεί ως δείκτης. Έτσι στη `loadtxt()` θέτουμε τον τύπο του μητρώου σε ακέραιο. Η εντολή `assert` χρησιμοποιείται για αποσφαλμάτωση. Η έννοια είναι ότι αν το πρόγραμμα είναι ορθό, τότε κάποιες συνθήκες πρέπει να ισχύουν, για παράδειγμα οι εξωτερικές δυνάμεις που υπολογίζει το πρόγραμμα πρέπει να είναι αυτές που έχει ορίσει ο χρήστης. Η εντολή `assert` εξασφαλίζει ότι ισχύει η συνθήκη. Αν δεν ισχύει, τότε η `assert` σταματάει το πρόγραμμα με το μήνυμα λάθους που φαίνεται στην `assert`. Λόγω αριθμητικών λαθών αποκοπής που οφείλονται στα πεπερασμένα σημαντικά ψηφία, ο έλεγχος γίνεται με ανοχή 0.1 Nt.