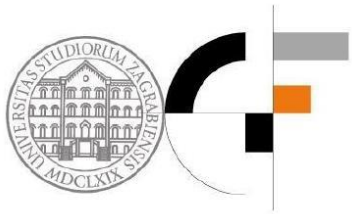
SVEUČILIŠTE U ZAGREBU

GRAĐEVINSKI FAKULTET



Programska realizacija metode pomaka za rešetkaste sisteme

ZAVRŠNI RAD

Studentica: Antonia Tarle, 0082049362

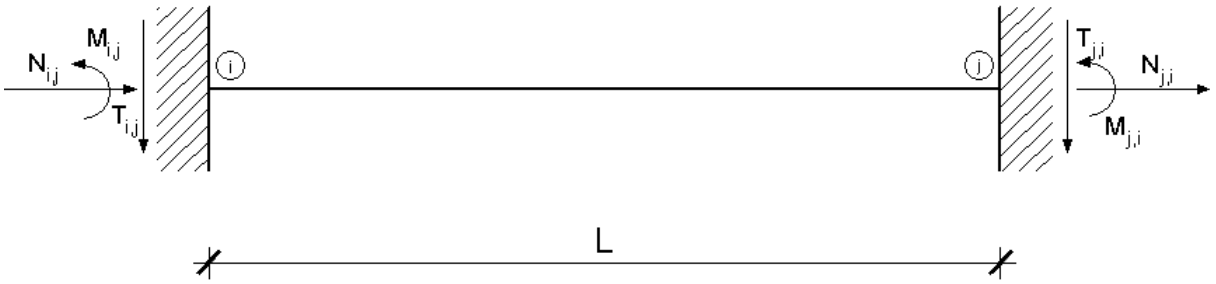
Mentor: prof. dr. sc. Krešimir Fresl

Zagreb, 05. srpnja 2016.

Sadržaj

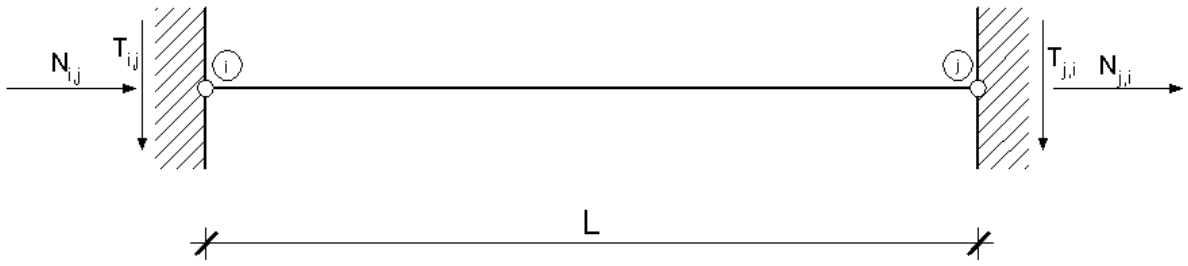
1. Uvod

Metode pomaka su metode proračuna štapnih sistema u kojima su temeljne nepoznanice translacijski pomaci i zaokreti čvorova. Međutim, u rešetkastim sistemima štapovi su povezani čvorovima u kojima nisu spriječeni zaokreti čvorova, pa kao nepoznanice ostaju samo translacijski pomaci. Čvorovi su točke u kojima se štapovi povezuju međusobno ili s podlogom. Ležajnim čvorovima nazivamo čvorove spojene s podlogom u kojima su pomaci spriječeni ili zadani, a slobodnim čvorovima nazivamo čvorove u kojima se spaja više elemenata i u kojima nisu spriječeni pomaci. S obzirom da su glavni elementi rešetke štapovi, a štapovi preuzimaju samo uzdužnu silu, uzima se da vanjsko opterećenje djeluje samo u čvorovima. Na primjerima obostrano upete grede i zglobnog štapnog sistema usporedit ćemo matrice krutosti odnosno broj nepoznanica za naš slučaj.



Slika 1 Gredni element

$$\begin{bmatrix} N_{i,j} \\ T_{i,j} \\ M_{i,j} \\ N_{j,i} \\ T_{j,i} \\ M_{j,i} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} & 0 & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} & 0 & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \cdot \begin{bmatrix} u_{i,j} \\ w_{i,j} \\ \varphi_{i,j} \\ u_{j,i} \\ w_{j,i} \\ \varphi_{j,i} \end{bmatrix} + \begin{bmatrix} \overline{N}_{i,j} \\ \overline{T}_{i,j} \\ \overline{M}_{i,j} \\ \overline{N}_{j,i} \\ \overline{T}_{j,i} \\ \overline{M}_{j,i} \end{bmatrix}$$



Slika 2 Štapni element

$$\begin{bmatrix} N_{i,j} \\ T_{i,j} \\ M_{i,j} \\ N_{j,i} \\ T_{j,i} \\ M_{j,i} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_{i,j} \\ w_{i,j} \\ \varphi_{i,j} \\ u_{j,i} \\ w_{j,i} \\ \varphi_{j,i} \end{bmatrix} + \begin{bmatrix} \overline{N}_{i,j} \\ \overline{T}^{c}_{i,j} \\ 0 \\ \overline{N}_{j,i} \\ \overline{T}^{c}_{j,i} \\ 0 \end{bmatrix}$$

Dvostrukom statičkom kondenzacijom prve matrice izbacivanjem kutova zaokreta čvorova znatno smanjujemo broj nepoznanica, te pritom kao nepoznanice ostaju samo uzdužni translacijski pomaci čvorova. Budući da je opterećenje u čvorovima, nema ni poprečne sile.

Metodom pomaka se mogu osim statički neodređenih proračunavati i statički određeni sistemi. Za razliku od metode sila ili inženjerske metode, vrlo lako ju je formalizirati kao kompjutorske programe za proračun štapnih konstrukcija, kao što će ovaj rad i pokazati. Primjena metode pomaka je široka, te se primjenjuje i u drugim granama građevinarstva kao što je hidrotehnika ili geomehanika. Na zamislima i postupcima metode pomaka utemeljena je i metoda konačnih elemenata koja omogućava proračun plošnih i masivnih sistema.

2. Matrična analiza sistema zglobnih štapova

2.1. Jednadžbe ravnoteže

U daljnjem tekstu broj čvorova označavat ćemo sa *n*, dok ćemo broj štapova označavati sa *b*. Kako će ovaj rad biti izrađen u programu Sage, tako ćemo i brojanje čvorova, elemenata i članova u listi prilagoditi njemu. Brojanje uvijek započinje od nule. Početni čvor štapa označavat će se s *i*, a krajnji s *j*. Štap između čvorova *i* i *j* označit ćemo sa {*i, j*}, što znači da {*i, j*} i {*j, i*} predstavljaju isti štap. Uzdužna sila na kraju *i* (sila kojom čvor *i* djeluje na štap) određena je vektorom:

$$\vec{S}_{i,j} = S_{i,j}\vec{e}_{j,i}.$$

$\vec{e}_{i,j}$ i $\vec{e}_{j,i}$ su jedinični vektori na osi štapa {*i, j*}, pri čemu je orijentacija prvog vektora od čvora *i* do čvora *j*, a drugog od čvora *j* do čvora *i*. Vektor uzdužne sile se računa na način da skalarnu vrijednost $S_{i,j}$ pomnožimo s jediničnom vektorom koji se nalazi na osi štapa. Ako je vrijednost skalara $S_{i,j} > 0$ riječ je o vlačnoj sili, dok je za tlačnu silu $S_{i,j} < 0$. Također, vektor uzdužne sile na kraju *j* iznosi:

$$\vec{S}_{j,i} = S_{j,i}\vec{e}_{i,j}.$$

Kako je $\vec{S}_{j,i} = -\vec{S}_{i,j}$, s druge strane imamo,

$$\vec{S}_{j,i} = -\left(S_{i,j}\vec{e}_{j,i}\right) = S_{i,j}\left(-\vec{e}_{j,i}\right) = S_{i,j}\vec{e}_{i,j},$$

pa je $S_{j,i} = S_{i,j}$; uvodimo oznaku $S_{\{i,j\}} = S_{j,i} = S_{i,j}$.

Ako su (x_i, y_i, z_i) i (x_j, y_j, z_j) koordinate čvorova *i* i *j*, onda su:

$$\vec{e}_{i,j} = \frac{x_j - x_i}{l_{\{i,j\}}}\vec{\imath} + \frac{y_j - y_i}{l_{\{i,j\}}}\vec{\jmath} + \frac{z_j - z_i}{l_{\{i,j\}}}\vec{k},$$

$$\vec{e}_{j,i} = \frac{x_i - x_j}{l_{\{i,j\}}}\vec{\imath} + \frac{y_i - y_j}{l_{\{i,j\}}}\vec{\jmath} + \frac{z_i - z_j}{l_{\{i,j\}}}\vec{k},$$

$$l_{\{i,j\}} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}.$$

$\vec{F_i}$ predstavlja rezultantu vanjskih sila koje djeluju na čvor *i*, te ćemo njezine skalarne komponente označiti s $F_{i,x}$, $F_{i,y}$ *i* $Fi_{,z.}$ Štap {*i, j*} na čvor *i* djeluje silom

$$-\vec{S}_{i,j} = -S_{\{i,ij\}}\vec{e}_{j,i} = S_{\{i,ij\}}\vec{e}_{i,j}.$$

Sada za svaki slobodan čvor vrijedi jednadžba ravnoteže sila koje na njega djeluju. U vektorskom obliku ona glasi:

$$\sum_{j\in N_i} S_{\{i,j\}}\vec{e}_{i,j} + \vec{F}_i = 0,$$

pri čemu je N_i skup čvorova koji su štapovima povezani s čvorom *i.*

Tu jednadžbu možemo raspisati za svaki smjer *x*, *y* i *z*, a one glase:

$$\sum_{j\in N_i} \frac{x_j - x_i}{l_{\{i,j\}}} S_{\{i,j\}} + \vec{F}_{i,x} = 0,$$

$$\sum_{j\in N_i} \frac{y_j - y_i}{l_{\{i,j\}}} S_{\{i,j\}} + \vec{F}_{i,y} = 0,$$

$$\sum_{j\in N_i} \frac{z_j - z_i}{l_{\{i,j\}}} S_{\{i,j\}} + \vec{F}_{i,z} = 0.$$

Što znači da za svaki slobodan čvor možemo napisati tri jednadžbe ravnoteže. Ako broj slobodnih čvorova označimo s n_f, tada dobivamo sustav sa $3n_f$ jednadžbi. Broj nepoznatih sila u štapovima bit će jednak broju štapova. Da bismo provjerili rješivost sustava, to možemo zapisati u matričnom obliku:

$$\mathbf{As} = -\mathbf{f}.$$

Vrijednosti sila u štapovima poredane su u vektor **s** prema brojčanim oznakama štapova, ako se uzme da je κ brojčana oznaka štapa { *i*, *j* }, onda je $S_{\{i,j\}}$ komponenta κ vektora **s,** $S_\kappa = S_{\{i,ij\}}$. Koeficijenti uz $S_{\{i,j\}} = S_\kappa$ u jednadžbama ravnoteže čvora i komponente su matrice sustava **A** u sjecištima redaka 3*i,* 3*i* + 1 *i* 3*i* + 2 sa stupcem κ:

$$a_{3i,\kappa} = \frac{x_j - x_i}{l_{\{i,j\}}} \ , \ a_{3i+1,\kappa} = \frac{y_j - y_i}{l_{\{i,j\}}} \ \text{ i } \ a_{3i+2,\kappa} = \frac{z_j - z_i}{l_{\{i,j\}}} .$$

Vrijednost $S_{\{i,j\}} = S_\kappa$ ulazi i u jednadžbe ravnoteže čvora *j*, kojima odgovaraju redci 3*j,* 3*j* + 1 *i* 3*j* + 2 matrice **A**, pa su, u istom stupcu:

$$a_{3j,\kappa} = \frac{x_i - x_j}{l_{\{i,j\}}} \ , \ a_{3j+1,\kappa} = \frac{y_i - y_j}{l_{\{i,j\}}} \ \text{ i } \ a_{3j+2,\kappa} = \frac{z_i - z_j}{l_{\{i,j\}}} .$$

Matrica **A** ima $3n_f$ redaka i b stupaca. Broj stupaca jednak je broju komponenata vektora **s**, dok je broj redaka jednak broju komponenata vektora **f**. Komponente vektora **f** s indeksima $3i$, $3i + 1$ i $3i + 2$ su skalarne komponente $F_{i,x}$, $F_{i,y}$ i $Fi_{,z}$ vanjske sile $\vec{F}_i$ koja djeluje u čvoru i. Podsjetimo se, u računalnom programu Sage u kojemu će rad biti izrađen počinje brojanje od 0, tako da je i numeracija čvorova i štapova 0, 1, 2, 3…

2.2. Kinematičke jednadžbe

Cilj je povezati promjene duljina štapova s pomacima čvorova. Ako svakom od čvorova pridružimo pripadni vektor pomaka $\vec{u}$, koordinate čvorova i i j će biti biti $(x_i + u_i, y_i + v_i, z_i + w_i)$ i $(x_j + u_j, y_j + v_j, z_j + w_j)$. Promjenu duljine štapa $\{i, j\}$ označit ćemo sa $d_{\{i,j\}}$; za $d_{\{i,j\}} > 0$ riječ je o produljenju, a za $d_{\{i,j\}} < 0$ o skraćenju štapa. Nova duljinu štapa sada iznosi zbroj početne duljine štapa i pripadnog produljenja, odnosno skraćenja $l_{\{i,j\}} + d_{\{i,j\}}$.

Primjenom Pitagorinog poučka slijedi:

$$(l_{\{i,j\}} + d_{\{i,j\}})^2 = [(x_j + u_j) - (x_i + u_i)]^2 + [(y_j + v_j) - (y_i + v_i)]^2 + [(z_j + w_j) - (z_i + w_i)]^2.$$

Kvadriranjem podizraza te promjenom redoslijeda i grupiranjem pribrojnika na desnoj strani dobivamo:

$$l_{\{i,j\}}^2 + 2l_{\{i,j\}}d_{\{i,j\}} + d_{\{i,j\}}^2 = [(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2] + 2\left[(x_j - x_i)(u_j - u_i) + (y_j - y_i)(v_j - v_i) + (z_j - z_i)(w_j - w_i)\right] + [(u_j - u_i)^2 + (v_j - v_i)^2 + (w_j - w_i)^2].$$

Prvi podizraz s desne strane znaka jednakosti, obuhvaćen uglatim zagradama, jednak je $l_{\{i,j\}}^2$, dok je podizraz u posljednjem retku jednak kvadratu duljine razlike pomaka $\vec{u}_j$ i $\vec{u}_i$. Budući da su pomaci mali $d_{\{i,j\}} \ll l_{\{i,j\}}$, $\|\vec{u}_i\| \ll l_{\{i,j\}}$ i $\|\vec{u}_j\| \ll l_{\{i,j\}}$, mogu se $d_{\{i,j\}}^2$ i $\|\vec{u}_j - \vec{u}_i\|^2$ zanemariti u odnosu na $2l_{\{i,j\}}d_{\{i,j\}}$ (drugi pribrojnik slijeva) i na drugi podizraz zdesna. Nakon zanemarivanja malih vrijednosti ostaje:

$$2l_{\{i,j\}}d_{\{i,j\}} = 2[(x_j - x_i)(u_j - u_i) + (y_j - y_i)(v_j - v_i) + (z_j - z_i)(w_j - w_i)].$$

Slijedi

$$d_{\{i,j\}} = \frac{x_j - x_i}{l_{\{i,j\}}}(u_j - u_i) + \frac{y_j - y_i}{l_{\{i,j\}}}(v_j - v_i) + \frac{z_j - z_i}{l_{\{i,j\}}}(w_j - w_i),$$

odnosno

$$d_{\{i,j\}} = -\left(\frac{x_j - x_i}{l_{\{i,j\}}}u_i + \frac{y_j - y_i}{l_{\{i,j\}}}v_i\frac{z_j - z_i}{l_{\{i,j\}}}w_i + \frac{x_i - x_j}{l_{\{i,j\}}}u_j + \frac{y_i - y_j}{l_{\{i,j\}}}v_j\frac{z_i - z_j}{l_{\{i,j\}}}w_j\right).$$

Promjene duljina štapova poredat ćemo u vektor **d** prema brojčanim oznakama štapova, dok ćemo orijentirane duljine komponenata pomaka čvorova svrstati u vektor **u** tako da su u_i, v_i *i* w_i na mjestima 3*i*, 3*i* + 1, 3*i* + 2. Poredak promjena duljina štapova u vektoru **d** odgovara poretku vrijednosti uzdužnih sila u vektoru **s**, a poredak skalarnih komponenata pomaka čvorova u vektoru **u** poretku skalarnih komponenata vanjskih sila vektoru **f**. Prethodni izraz prelazi u

$$\mathbf{d} = -\mathbf{B}\mathbf{u}$$

gdje je **B** matrica čije su komponente:

$$b_{3i} = \frac{x_j - x_i}{l_{\{i,j\}}}, \qquad b_{3i+1} = \frac{y_j - y_i}{l_{\{i,j\}}}, \qquad b_{3i+2} = \frac{z_j - z_i}{l_{\{i,j\}}},$$

$$b_{3j} = \frac{x_i - x_j}{l_{\{i,j\}}}, \qquad b_{3j+1} = \frac{y_i - y_j}{l_{\{i,j\}}}, \qquad b_{3j+2} = \frac{z_i - z_j}{l_{\{i,j\}}}.$$

Vektori **d** i **u** sadrže *b* i $3n_f$ komponenata, pa matrica **B** ima *b* redaka i $3n_f$ stupaca, odnosno matrica **B** odgovara transponiranoj matrici **A**:

$$\mathbf{B} = \mathbf{A}^{\mathrm{T}}.$$

Matricu **B** nazivamo kinematičkom matricom.

2.3. Metoda pomaka

Pretpostavimo da je matrica **A** regularna matrica. Postoje 2 moguća slučaja.

1) Broj stupaca je jednak broju redaka matrice, odnosno $b = 3n_f$. Za ovaj slučaj sustav ima jedinstveno rješenje, a sistem štapova je geometrijski nepromjenjiv i statički određen.

2) Broj stupaca je veći od broja redaka matrica, odnosno $b > 3n_f$. Sada je sustav jednadžbi ravnoteže rješiv za sve vektore koji leže prostoru čvorova, tj. za sve vektore **f**. Sistem štapova je u tom slučaju statički neodređen i geometrijski nepromjenjiv. Kako je broj vrijednosti sila u štapovima veći od broja jednadžbi, matrica **A** prostor veće dimenzije

preslikava u manje dimenzije. Pri tom preslikavanju neki vektori prostora štapova iščezavaju, tj. više vektora se preslikava u jedan vektor prostora čvorova.

Ako na štap duljine $l_{\{i,j\}}$ djeluje sila $S_{\{i,j\}}$, tada produljenje štapa možemo odrediti kao

$$d_{\{i,j\}} = \delta_{\{i,j\}} S_{\{i,j\}}.$$

$\delta_{\{i,j\}}$ nam predstavlja koeficijent u matrici fleksibilnosti, tj. koeficijent uzdužne popustljivosti i računa se kao

$$\delta_{\{i,j\}} = \frac{l_{\{i,j\}}}{E_{\{i,j\}} A_{\{i,j\}}} .$$

Uvrštavanjem koeficijenta uzdužne popustljivosti u izraz za produljenje štapa, te izvlačenjem sile iz jednadžbe dobivamo sljedeći izraz:

$$S_{\{i,j\}} = k_{\{i,j\}}\, d_{\{i,j\}},$$

pri čemu *k* predstavlja uzdužni koeficijent u matrici krutosti i jednak je inverzu koeficijenta uzdužne popustljivosti, tj.

$$k_{\{i,j\}} = \frac{E_{\{i,j\}} A_{\{i,j\}}}{l_{\{i,j\}}} .$$

Koeficijente uzdužne krutosti ćemo smjestiti u dijagonalnu matricu diag (**k**) tako da koeficijent *k* predstavlja dijagonalni element u sjecištu retka *i* i stupca *i*. U matričnom zapisu izraz $S_{\{i,j\}} = k_{\{i,j\}} d_{\{i,j\}}$ za sve štapove glasi:

$$\mathbf{s} = diag\ (\mathbf{k})\ \mathbf{d},$$

uvrštavanjem tog izraza u izraz **As** = **- f** dobivamo:

$$\mathbf{A}\ diag\ (\mathbf{k})\ \mathbf{d} = -\mathbf{f}.$$

Vratimo se sada na izraz za produljenje svih štapova $\mathbf{d} = -\mathbf{B}\,\mathbf{u}$. Sada slijedi:

$$-\mathbf{A}\ diag\ (\mathbf{k})\ \mathbf{B}\,\mathbf{u} = -\mathbf{f}, \text{ odnosno } \mathbf{A}\ diag\ (\mathbf{k})\ \mathbf{B}\,\mathbf{u} = \mathbf{f},$$

gdje je $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ pa je

$$\mathbf{A}\ diag\ (\mathbf{k})\ \mathbf{A}^{\mathrm{T}}\mathbf{u} = \mathbf{f}.$$

Dobivenu matricu $\mathbf{A}\ diag\ (\mathbf{k})\ \mathbf{A}^{\mathrm{T}}$ nazivamo matricom krutosti sistema i označavamo je s **K**.

$$\mathbf{K} = \mathbf{A}\, diag\,(\mathbf{k})\, \mathbf{A}^{\mathrm{T}}.$$

Nepoznate pomake dobivamo rješavanjem sustava jednadžbi $\mathbf{K}\,\mathbf{u} = \mathbf{f}$, tj. $\mathbf{u} = \mathbf{K}^{-1}\mathbf{f}$.

2.4. Neposredna programska realizacija metode pomaka

Algoritmi za ovo poglavlje su preuzeti iz rada studentice Građevinskog fakulteta u Zagrebu, Elene Franjičić *Klasifikacija sklopova zglobnih štapova* (Završni rad, rujan 2015. god.).

2.4.1. Primjer 1. Statički neodređeni štapni sistem

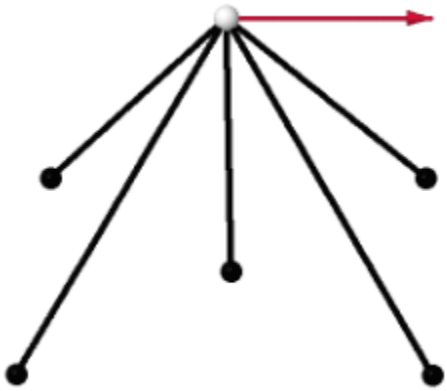
```
joints = { 0: (-2.0, 0.0, 0.0), 1: (0.0, 0.0, 0.0),
2: (2.0, 0.0, 0.0), 3: (0.0, -2.0, 0.0), 4: (0.0, 2.0, 0.0),
5: (0.0, 0.0, 2.0) }
print 'joints:'
print_dict (joints, indent = '   ')
print

bars = { 0: (0, 5), 1: (1, 5), 2: (2, 5), 3: (3, 5), 4: (4, 5) }
print 'bars:'
print_dict (bars, indent = '   ')
print

supports = [0, 1, 2, 3, 4]
print 'supports:', supports
print

loads = { 5: (100.0, 100.0, 0.0) }
print 'loads:'
print_dict (loads, indent = '   ')
```

Za prvi primjer prikazat će se statički neodređen sistem od pet štapova. Naredbom *joints* = { *oznaka*: (x_i, y_i, z_i) } zadajemo koordinate čvorova. Prvi član predstavlja oznaku čvora, drugi su njegove koordinate.

Naredbom *bars* = { oznaka: (*i, j*) } pridružujemo čvorove određenom elementu. *i* predstavlja prvi čvor, dok *j* predstavlja drugi čvor elementa. Naredbom *support* određujemo koji su čvorovi ležajni, te još preostaje zadati opterećenje u čvorove naredbom *loads* = { (*x, y, z*) }.

```
joints:
   0:   (-2.00000000000000, 0.000000000000000, 0.000000000000000)
   1:   (0.000000000000000, 0.000000000000000, 0.000000000000000)
   2:   (2.00000000000000, 0.000000000000000, 0.000000000000000)
   3:   (0.000000000000000, -2.00000000000000, 0.000000000000000)
   4:   (0.000000000000000, 2.00000000000000, 0.000000000000000)
   5:   (0.000000000000000, 0.000000000000000, 2.00000000000000)

bars:
   0:   (0, 5)
   1:   (1, 5)
   2:   (2, 5)
   3:   (3, 5)
   4:   (4, 5)

supports: [0, 1, 2, 3, 4]

loads:
   5:   (100.000000000000, 100.000000000000, 0.000000000000000)
```

```
plot_truss3d (joints, bars, supports, loads)
```



Slika 3 Opterećeni štapni sistem

```
maxwells_rule (joints, supports, bars)
```

Naredbom *maxwells_rule* provjeravamo statičku neodređenost sistema $3n_f - b$, pri čemu n_f predstavlja broj slobodnih čvorova, a b broj štapova u sistemu. Odabrani sistem je dva puta statički neodređen.

```
3 * 1 - 5  ==  -2  !=  0
```

```
AA = equilibrium_matrix (joints, bars, free_joints)
show (AA)
```

Naredbom *equilibrium_matrix* izračunava se ravnotežna matrica. Ta se naredba poziva s čvorovima, elementima i slobodnim čvorovima, koji su već prije definirani. Matrica **AA** je objašnjena u prethodnom poglavlju.

$$\begin{pmatrix} -0.707106781187 & 0.0 & 0.707106781187 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -0.707106781187 & 0.707106781187 \\ -0.707106781187 & -1.0 & -0.707106781187 & -0.707106781187 & -0.707106781187 \end{pmatrix}$$

Naredba *load_vector* sile zadane u čvorovima svrstava u vektor opterećenja.

```
ff = load_vector (loads, free_joints)
ff
```

```
(100.0, 100.0, 0.0)
```

Za primjer je odabran modul elastičnosti $E = 2 \cdot 10^8\ kN/m^2$ i dimenzije presjeka $F = 0.05 \cdot 0.05\ m^2$.

```
E = 2.e8
F = 0.05*0.05
E; F
```

lkk predstavlja listu dijagonalnh elemenata u kojoj svaki element predstavlja uzdužnu krutost štapova, EF/L. n_i i n_j predstavljaju početni i krajnji čvor pojedinog elementa, a *bar_length* duljinu pojedinog elementa.

```
lkk = []
for b in bars :
    ni, nj = bars[b]
    xi = joints[ni]
    xj = joints[nj]
    lkk.append (E*F/bar_length (xi, xj))
lkk
```

Pomoću liste dijagonalnih elemenata oblikuje se dijagonalna matrica **dkk**.

```
dkk = diagonal_matrix (RDF, lkk)
show (dkk)
```

$$\begin{pmatrix} 176776.695297 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 250000.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 176776.695297 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 176776.695297 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 176776.695297 \end{pmatrix}$$

Idući korak je matrica krutosti sistema, koja se, nakon što je definirana ravnotežna matrica **AA** i dijagonalna matrica **dkk** oblikuje prema ranije navedenom izrazu.

```
KK = AA * dkk * AA.T
show (KK)
```

$$\begin{pmatrix} 176776.695297 & 0.0 & 0.0 \\ 0.0 & 176776.695297 & 0.0 \\ 0.0 & 0.0 & 603553.390593 \end{pmatrix}$$

Zatim slijedi računanje pomaka čvorova *uu*, produljenja elemenata *dd*, te sila u elementima *ss*, na isti način kao što je pokazano i objašnjeno u prethodnom poglavlju.

```
uu = KK \ ff
uu
```

```
(0.000565685424949238, 0.000565685424949238, 0.0)
```

```
dd = -AA.T * uu
dd
```

```
(0.00039999999999999996, 0.0, -0.00039999999999999996,
0.00039999999999999996, -0.00039999999999999996)
```

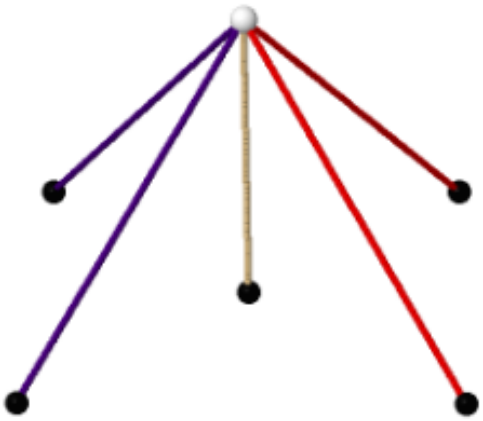
```
ss = dkk * dd
ss
```

```
(70.71067811865476, 0.0, -70.71067811865476, 70.71067811865476,
-70.71067811865476)
```

Naredbom *dict_ss* pridružuju se unutarnje sile pojedinim elementima.

```
dict_ss = dict_bar_force (bars, ss)
print_dict (dict_ss)
```

```
0:   70.7106781187
1:   0.0
2:   -70.7106781187
3:   70.7106781187
4:   -70.7106781187
```



Slika 4 Prikaz tlačnih i vlačnih štapova

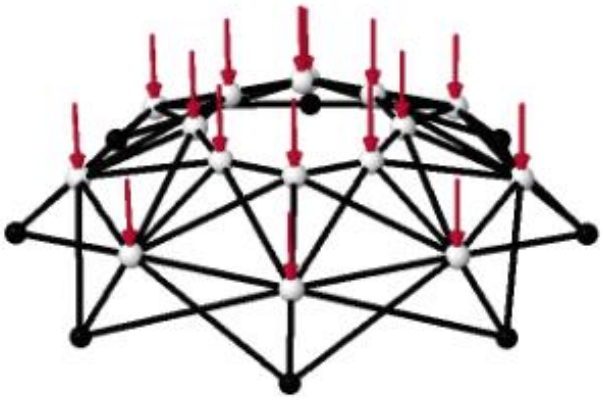
2.4.2. Primjer 2. Statički neodređena „Schwedlerova“ kupola

U primjeru 2. zadan je radijus baze 15 metara, visina prvog kata 3.25 metara, drugog kata 6.25 metara, najviša točka kupole 7 metara, te broj ležajnih čvorova 8. Primjer 2. se neće posebno objašnjavati.

```
joints, bars, supports = schwedler_sind (15., [3.25, 6.25, 7], 8)
```

```
load_gen = [(8, 8, (0.0, 0.0, -100.0)),
            (8, 16, (0.0, 0.0, -100.0))]
loads = make_loads (load_gen)
```

```
plot_truss3d (joints, bars, supports, loads, pull = False, load_scale = 0.0425)
```



Slika 5 Opterećena „Schwedlerova“ kupola

```
maxwells_rule (joints, supports, bars)
```

```
3 * 16 - 64  ==  -16  !=  0
```

```
AA = equilibrium_matrix (joints, bars, free_joints)
AA
```

```
48 x 64 dense matrix over Real Double Field
```

Matrice u ovom primjeru se zbog svojih velikih dimenzija neće ispisivati.

```
ff = load_vector (loads, free_joints)
print ff
```

```
(0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0,
0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0,
0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0,
0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0, 0.0, 0.0, -100.0)
```

```
dkk = diagonal_matrix (RDF, lkk)
dkk
```

```
64 x 64 sparse matrix over Real Double Field
```

```
KK = AA * dkk * AA.T
KK
```

```
48 x 48 dense matrix over Real Double Field
```

```
uu = KK \ ff
print uu
```

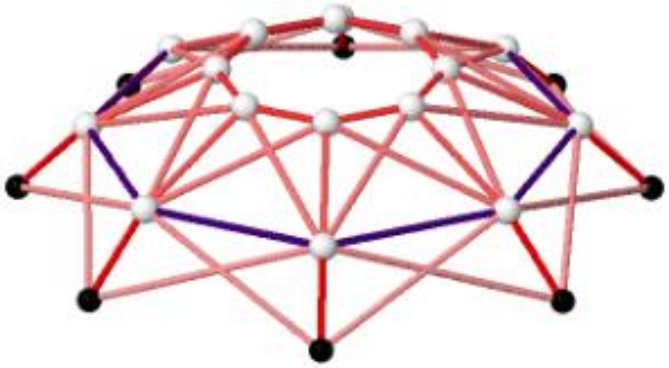
```
(0.0006287193820572289, 1.6074594723909675e-18, -0.0012090077396721627,
0.0004445717385160856, 0.0004445717385160855, -0.0012090077396721573,
5.207390022154409e-19, 0.0006287193820572265, -0.001209007739672164,
-0.0004445717385160276, 0.0004445717385160243, -0.0012090077396721601,
-0.0006287193820572322, 6.656102980853586e-19, -0.00120900773967216,
-0.0004445717385160822, -0.0004445717385160805, -0.0012090077396721638,
-1.2299748196577516e-18, -0.0006287193820572291, -0.0012090077396721627,
0.0004445717385160778, -0.0004445717385160779, -0.0012090077396721681,
-0.0009963147879079098, 3.939035011877538e-18, -0.006110070836254311,
-0.0007045009427261138, -0.0007045009427261142, -0.006110070836254296,
1.6458704750050882e-18, -0.0009963147879079165, -0.00611007083625432,
0.0007045009427261205, -0.0007045009427261231, -0.0061100708362543145,
0.0009963147879079901, 8.150145860911763e-19, -0.006110070836254297,
0.0007045009427261225, 0.0007045009427261262, -0.00611007083625432,
-2.403691249822541e-18, 0.0009963147879079066, -0.006110070836254303,
-0.0007045009427261294, -0.00611007083625433)
```

```
dd = -AA.T * uu
print dd
```

```
(-0.0012847325036767973, -0.0012847325036767971, -0.0012847325036767965,
-0.001284732503676796, -0.001284732503676798, -0.001284732503676797,
-0.0012847325036767976, -0.0012847325036767965, 0.0004812009822402374З,
0.00048120098224023754, 0.000481200982240237l, 0.00048120098224023775,
0.00048120098224023705, 0.00048120098224023683, 0.00048120098224023597,
0.000481200982240236З, -0.00030138105590722674, -0.0003013810559072296,
-0.0003013810559072275, -0.00030138105590722815,
-0.00030138105590723043, -0.0003013810559072297, -0.0003013810559072306,
-0.000301381055907227, -0.00030138105590723065, -0.0003013810559072302,
-0.0003013810559072267, -0.0003013810559072288, -0.00030138105590722815,
-0.000301381055907227, -0.00030138105590722783, -0.00030138105590722756,
-0.0006867404023606144, -0.0006867404023606135, -0.0006867404023606135,
-0.0006867404023606144, -0.0006867404023606148, -0.0006867404023606135,
-0.0006867404023606139, -0.0006867404023606144, -0.0007625463255053913,
-0.0007625463255053913, -0.0007625463255053911, -0.0007625463255053908,
-0.0007625463255053913, -0.0007625463255053918, -0.000762546325505392,
-0.0007625463255053924, -0.0007762263304571578, -0.0007762263304571591,
-0.0007762263304571593, -0.0007762263304571585, -0.0007762263304571604,
-0.0007762263304571602, -0.0007762263304571609, -0.0007762263304571589,
-0.0007762263304571613, -0.0007762263304571604, -0.0007762263304571583,
-0.0007762263304571587, -0.0007762263304571593, -0.0007762263304571583,
-0.000776226330457158, -0.0007762263304571586)
```

```
ss = dkk * dd
print ss
```

```
(-269.83373747778427, -269.8337374777843, -269.8337374777841,
-269.8337374777841, -269.83373747778444, -269.8337374777842,
-269.8337374777843, -269.8337374777841, 54.57370054075599,
54.57370054075598, 54.57370054075596, 54.573700540756015,
54.57370054075594, 54.57370054075591, 54.57370054075581,
54.57370054075584, -27.075976618319544, -27.075976618319796,
-27.075976618319608, -27.075976618319668, -27.07597661831987,
-27.075976618319807, -27.075976618319885, -27.075976618319558,
-27.07597661831989, -27.075976618319853, -27.075976618319533,
-27.07597661831973, -27.075976618319665, -27.07597661831956,
-27.07597661831964, -27.07597661831961, -100.30188829847502,
-100.30188829847489, -100.30188829847489, -100.30188829847502,
-100.30188829847508, -100.30188829847489, -100.30188829847496,
-100.30188829847502, -185.66956857402346, -185.66956857402346,
-185.6695685740234, -185.66956857402332, -185.66956857402346,
-185.66956857402354, -185.6695685740236, -185.66956857402366,
-85.15518072929603, -85.15518072929616, -85.1551807292962,
-85.15518072929609, -85.15518072929632, -85.1551807292963,
-85.15518072929638, -85.15518072929613, -85.15518072929642,
-85.15518072929632, -85.15518072929609, -85.15518072929613,
-85.1551807292962, -85.15518072929609, -85.15518072929606,
-85.1551807292961)
```



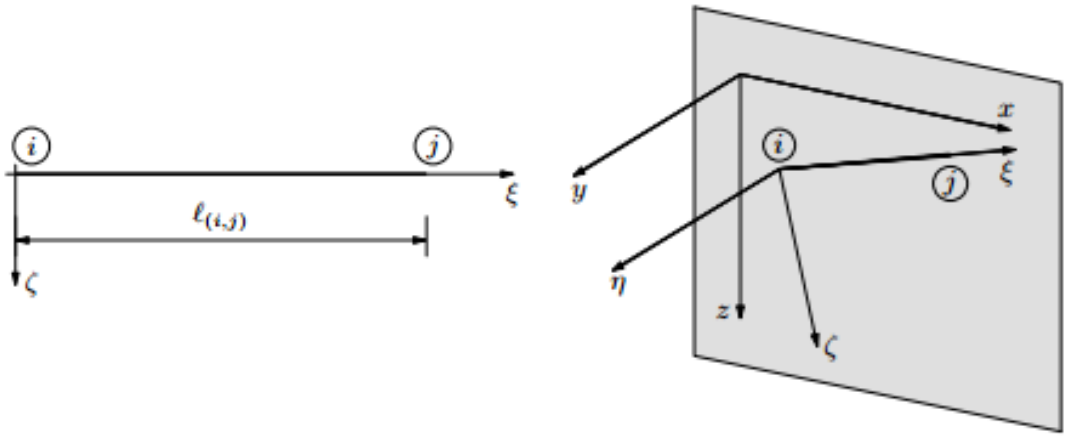
Slika 6 Prikaz vlačnih i tlačnih elemenata

3. Klasična programska realizacija metode pomaka

Ovaj rad će biti izrađen u programu Sage. Sage je matematički softver sa značajkama koje pokrivaju mnoge aspekte matematike . Klasična realizacija metode pomaka je jednostavnija za izvedbu te se lako može proširiti i prilagoditi na okvirne sisteme, te na konačne elemente.

Nakon što smo objasnili osnovne pojmove, možemo početi s izradom računalnog programa koji rješava rešetkaste sisteme metodom pomaka, te objašnjenjima pripadnih algoritama.

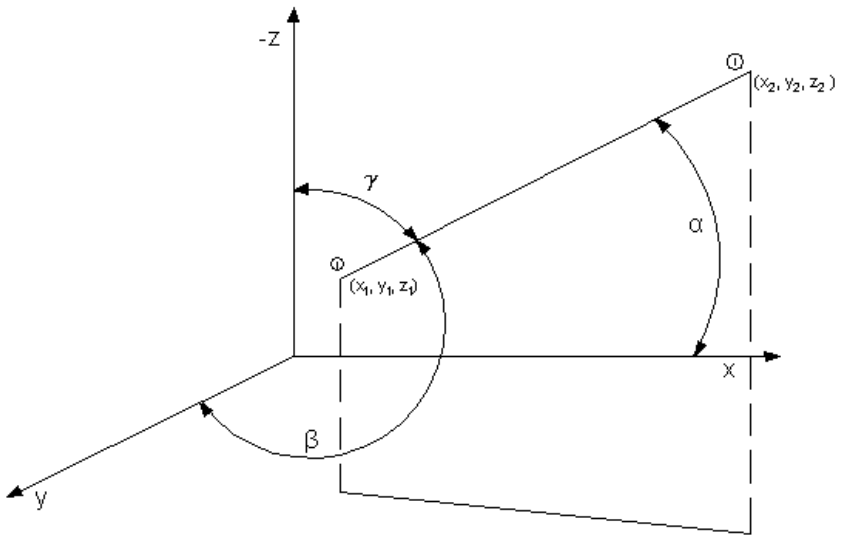
3.1. Lokalni i globalni koordinatni sustav



Slika 7 Lokalni i globalni koordinatni sustav

Štapovi konstrukcije mogu biti u raznim položajima u prostoru, odnosno osi štapova zatvaraju različite kutove s osima *x*, *y*, *z* koordinatnog sustava. Sustav *xyz* nazivamo globalnim koordinatnim sustavom. Naš cilj je reakcije, zadana opterećenja, pomak na krajevima i sile u štapovima izraziti u obliku koji je primjenjiv na štap u bilo kojem položaju. Zbog toga uvodimo pojam lokalnog koordinatnog sustava u kojem uzdužna os štapa leži na osi ξ i orijentirana je od čvora *i* prema čvoru *j*. Ostale dvije osi su orijentirane na način kao *y* i *z* osi, te su okomite na os ξ. Sada svaki štap ima svoj lokalni koordinatni sustav.

```
def matrica_elementa (e, EA, cvorovi):
    [x1, y1, z1] = cvorovi [e[0]]
    [x2, y2, z2] = cvorovi [e[1]]

    L = sqrt ((z2 - z1)^2 + (x2 - x1)^2 + (y2 - y1)^2)
    x21 = x2 - x1
    y21 = y2 - y1
    z21 = z2 - z1

    l = x21/L
    m = y21/L
    n = z21/L
    k = (EA/L) * matrix([[ 1, 0, 0, -1, 0, 0 ],
                         [ 0, 0, 0, 0, 0, 0 ],
                         [ 0, 0, 0, 0, 0, 0 ],
                         [ -1, 0, 0, 1, 0, 0 ],
                         [ 0, 0, 0, 0, 0, 0 ],
                         [ 0, 0, 0, 0, 0, 0 ]])
    R = matrix([[ l, m, n, 0, 0, 0 ],
                [ l, m, n, 0, 0, 0 ],
                [ l, m, n, 0, 0, 0 ],
                [ 0, 0, 0, l, m, n ],
                [ 0, 0, 0, l, m, n ],
                [ 0, 0, 0, l, m, n ]])

    Ke = matrix(R.transpose()*k*R)
    return Ke
```



Slika 8 Kutovi koje konstrukcija zatvara sa globalnim osima

Kut između globalne osi *x* i lokalne osi *ξ* označili smo s *α*, između osi *y* i osi *ξ* označili smo s *β*, te između osi *z* i osi *ξ* s *γ*.

Kosinuse kutova, smo zapisali kao *l*, *m* i *n* da bi nam zapis matrice transformacije bio kraći.

$$l = \frac{x_2 - x_1}{L},\ m = \frac{y_2 - y_1}{L},\ n = \frac{z_2 - z_1}{L}\ ;$$

pri čemu je *L* duljina štapa. Također smo u kodu definirali način na koji se uzimaju koordinate *x*, *y*, *z* prvog, te drugog čvora elementa. Naredbom *matrica_elementa* smo definirali lokalnu matricu za štapove, te smo ju pomoću matrice transformacije **R**, transformirali u globalnu:

$$\mathbf{K}^{lok}\mathbf{w}^{lok} = \mathbf{q}^{lok} \quad , \quad \mathbf{w}^{lok} = \mathbf{R}^{gl-lok}\mathbf{w}^{gl} \ , \quad \mathbf{q}^{lok} = \mathbf{R}^{gl-lok}\mathbf{q}^{gl}$$

$$\mathbf{K}^{lok}\mathbf{R}^{gl-lok}\mathbf{w}^{gl} = \mathbf{R}^{gl-lok}\mathbf{q}^{gl} \ ,$$

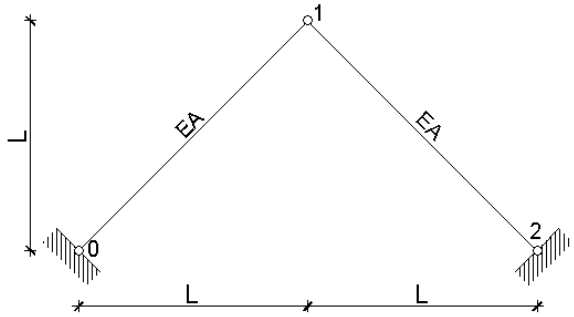
Da bi se riješili $\mathbf{R}^{gl-lok}$ s desne strane, potrebno je pomnožiti sve s lijeve strane s $(\mathbf{R}^{gl-lok})^{-1}$ pri čemu slijedi:

$$(\mathbf{R}^{gl-lok})^{-1}\ \mathbf{K}^{lok}\mathbf{R}^{gl-lok}\mathbf{w}^{gl} = \mathbf{q}^{gl}$$

Pritom je $\mathbf{w^{gl}}$ vektor pomaka u globalnim koordinatama, a $\mathbf{q^{gl}}$ vektor opterećenja također u globalnim koordinatama. Prva tri člana jednadžbe predstavljaju transformaciju lokalne matrice u globalnu za svaki štap naše konstrukcije. Jednostavnije nam je zadavati vektor upetosti u globalnim koordinatama, nego da smo za svaki štap zadavali vektore u pripadnom lokalnom

koordinatnom sustavu. Matrica transformacije je ortogonalna što znači da je inverz te matrice jednak transponiranoj matrici.

Nakon što smo dobili globalnu matricu za svaki štap, slijedi spajanje svih štapova, odnosno svih pojedinih globalnih matrica u jednu. Način na koji smo to realizirali u programskom kodu objasnit ćemo na jednostavnom ravninskom primjeru s dva međusobno spojena štapa.



Slika 9 Trozglobni sustav

```
def matrica_sistema (elementi, cvorovi, EA):
    Kgl = zero_matrix(RDF, 3*len(cvorovi), 3*len(cvorovi))
    q_gl = zero_matrix(RDF, 3*len(cvorovi), 1)
    for i in srange (len (elementi)):
        [ni, nj] = elementi[i]
        nn = [3*ni, 3*ni + 1, 3*ni + 2, 3*nj, 3*nj + 1, 3*nj + 2]
        for k in srange (6):
            ii= nn[k]
            for l in srange (k, 6):
                jj=nn[l]
                Ke = matrix(matrica_elementa (elementi[i], EA, cvorovi))
                if jj <> ii:
                    Kgl[ii, jj]= Kgl[ii, jj] + Ke[k, l]
                    Kgl[jj, ii]= Kgl[jj, ii] + Ke[l, k]
                else:
                    Kgl[ii, jj]= Kgl[ii, jj] + Ke[k, l]
    return Kgl
```

Globalna matrica za oba elementa i pretpostavljena nul - matrica sustava su

$$\mathbf{K}_e = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \quad \mathbf{K}_{gl} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

U prvom koraku, ***i = 0***, a element 0 ima početni *čvor* 0, a krajnji *čvor* 1.

$$[ni, nj] = [0, 1] \qquad nn = [0 \quad 1 \quad 2 \quad 3]$$

Za ***k = 0*** dobivamo $ii = nn[0] = 0$. *k* predstavlja redak u globalnoj matrici elementa, a ***ii*** predstavlja redak u globalnoj matrici sustava. Sada brojač ***l*** broji stupce globalne matrice elementa i pridružuje ih doprinosu krutosti čvora u globalnoj matrici sustava.

Kada se brojač ***l*** provrti od 0 do 3, ***jj*** (predstavlja stupac u gl. matrici sustava) nam poprima vrijednosti 0, 1, 2 i 3, te dobivamo sljedeću matricu sustava

$$\mathbf{K}_{gl} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Kada se brojač ***k*** provrti od 0 do 3 dobivamo sljedeću matricu sustava

$$\mathbf{K}_{gl} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 0 & 0 \\ k_{21} & k_{22} & k_{23} & k_{24} & 0 & 0 \\ k_{31} & k_{32} & k_{33} & k_{34} & 0 & 0 \\ k_{41} & k_{42} & k_{43} & k_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sada algoritam prelazi na drugi element, tj. ***i = 1***.

Sada je početni *čvor* 1, a krajnji *čvor* 2.

$$[ni, nj] = [1, 2] \qquad nn = [2 \quad 3 \quad 4 \quad 5]$$

Za *k* = 0 i *l* od 0 do 3 dobivamo matricu

$$\mathbf{K}_{gl} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 0 & 0 \\ k_{21} & k_{22} & k_{23} & k_{24} & 0 & 0 \\ k_{31} & k_{32} & k_{33} + k_{11} & k_{34} + k_{12} & k_{13} & k_{14} \\ k_{41} & k_{42} & k_{43} & k_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Nakon što ***k*** prođe cijelu petlju dobivamo konačnu matricu sustava

$$\mathbf{K}_{gl} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 0 & 0 \\ k_{21} & k_{22} & k_{23} & k_{24} & 0 & 0 \\ k_{31} & k_{32} & k_{33}+k_{11} & k_{34}+k_{12} & k_{13} & k_{14} \\ k_{41} & k_{42} & k_{43}+k_{21} & k_{44}+k_{22} & k_{23} & k_{24} \\ 0 & 0 & k_{31} & k_{32} & k_{33} & k_{34} \\ 0 & 0 & k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}$$

Uklapanje pojedinih globalnih matrica elementa svodi se na pridruživanje zajedničkih pomaka pojedinih krajeva elementa koji se spajaju u isti čvor, što algoritam zapravo i radi.

3.2. Definiranje ležajeva

```
def lezajevi (rubni_uvjeti):
    w_gl = range(3 * len(cvorovi))
    for i in srange(len(cvorovi)):
        w_gl[3 * i] = 'ux_' + str(i)
        w_gl[3 * i + 1] = 'uy_' + str(i)
        w_gl[3 * i + 2] = 'uz_' + str(i)
    for i in srange(len(rubni_uvjeti)):
        br_cv = rubni_uvjeti[i][0]
        lezaj = rubni_uvjeti[i][1]
        if lezaj == 'zglobni':
            w_gl[3 * br_cv] = 0
            w_gl[3 * br_cv + 1] = 0
            w_gl[3 * br_cv + 2] = 0
        if lezaj == 'klizni_x':
            w_gl[3 * br_cv + 1] = 0
            w_gl[3 * br_cv + 2] = 0
        if lezaj == 'klizni_y':
            w_gl[3 * br_cv] = 0
            w_gl[3 * br_cv + 2] = 0
        if lezaj == 'klizni_z':
            w_gl[3 * br_cv] = 0
            w_gl[3 * br_cv + 1] = 0
    return w_gl
```

U štapnim konstrukcijama postoje *"zglobni"* te *"klizni"* ležajevi. Kako se rešetka nalazi u trodimenzionalnom prostoru, postoje tri moguća klizna oslonca, u *x*, *y* te *z* smjeru.

Definirali smo već **w_gl** kao vektor pomaka, ali nismo rekli da je njegova dimenzija ovisna o broju čvorova te dimenziji prostora u kojem se nalazimo. U trodimenzionalnom prostoru dimenzija vektora upetosti iznosi tri puta broj čvorova. Brojač *i* broji čvorove i to od nultog prema zadnjem i za svaki čvor ispisuje koje su nepoznanice pomaka. Ako je oslonac *''zglobni''*, pomak *u* je 0 u sva tri smjera *x*, *y* i *z*, ako je oslonac klizni, s dopuštenim pomakom u *x* smjeru, pomaci u ostala dva smjera iznose 0, a kao nepoznanica ostaje pomak u smjeru *x* za zadani čvor. U stvarnosti ležajevi mogu biti pod bilo kojim kutom u odnosu na lokalnu os čvora, ali zbog kompliciranosti programiranja to nećemo uzeti u obzir.

3.3. Definiranje vektora vanjskog opterećenja

```
def opt (opt_cv, cvorovi):
    q_gl= zero_vector(RDF, 3*len(cvorovi))
    for i in srange(len(opt_cv)):
        br_cv = opt_cv[i][0]
        [qx, qy, qz] = opt_cv[i][1]
        q_gl[3*br_cv] = qx
        q_gl[3*br_cv + 1] = qy
        q_gl[3*br_cv + 2] = qz
    return q_gl
```

Vektor opterećenja **q_gl** zadajemo kao nul - vektor, čija je dimenzija kao i vektoru pomaka, tri puta broj čvorova štapne konstrukcije. U svakom od čvorova moguće je zadati tri sile, u tri smjera *x*, *y* i *z.* Kod je napravljen tako da kada zadamo silu u nekom čvoru, prazni odnosno nul - vektor se puni i umjesto nula sada imamo iznose sila koje djeluju na pripadni čvor. U prvom od tri slobodna mjesta se nalazi sila u *x* smjeru, na drugom mjestu sila u *y* smjeru, te na zadnjem, sila u *z* smjeru. Odnosno kada je *br_cv* = 0 imamo *q_gl* (3*0) = *qx*, *q_gl* (3*0+1) = *qy*, te *q_gl* (3*0+2) = *qz*.

3.4. Ispis pomaka čvorova

```
def pomaci (cvorovi, w_gl):
    pom = range(3 * len(cvorovi))
    for i in srange(len(cvorovi)):
        pom[3 * i] = 'ux_' + str(i)
        pom[3 * i + 1] = 'uy_' + str(i)
        pom[3 * i + 2] = 'uz_' + str(i)
        print pom[3 * i], '=', w_gl[3 * i]
        print pom[3 * i + 1], '=', w_gl[3 * i + 1]
        print pom[3 * i + 2], '=', w_gl[3 * i + 2]
    return
```

Izradu koda za ispisivanje pomaka započinjemo definiranjem broja pomaka, koji iznosi tri puta broj čvorova, te pozivamo naredbu koja nam ispisuje iznose pomaka za svaki čvor za sva tri smjera, *x*, *y* i *z*, no kako još nismo definirali način izračuna vektora pomaka, taj dio ćemo trenutno izostaviti. Nakon što definiramo reduciranu matricu krutosti idućim algoritmom, možemo relativno jednostavno izračunati pomake svih čvorova. To dobivamo tako da množimo reduciranu matricu krutosti sa vektorom vanjskog opterećenja u čvorovima, tj. $\mathbf{w}_{gl} = \mathbf{K}_{gl,red}^{-1} \cdot \mathbf{q}_{gl}$.

3.5. Reduciranje matrice krutosti zbog rješavanja sustava jednadžbi

```
def elim (Kgl, w_gl, q_gl):
    for i in srange(3 * len(cvorovi)):
            for j in srange(3 * len(cvorovi)):
                if w_gl[i] == 0:
                    if i == j:
                        Kgl[i, i] = 1
                        q_gl[i] = 0
                    else:
                        Kgl[i, j] = 0
                        Kgl[j, i] = 0
    return Kgl
```

Matrica krutosti $\mathbf{K}_{gl}$ je singularna, tj. nema inverz, pa je nemoguće dobiti rješenja u obliku pomaka. Kako bismo mogli riješiti sustav jednadžbi potrebno je iz matrice krutosti eliminirati nepotrebne redove i stupce ovisno o rubnim uvjetima u globalnom vektoru pomaka. To postižemo tako da dijagonalni element matrice zamijenimo jedinicom, a ostale elemente koji

se nalaze u retku i stupcu pripadnog elementa zamijenimo nulama. Također, na pripadnom mjestu globalnog vektora čvornih sila stavljamo nulu. Na taj način postižemo da je pripadni pomak na dijagonali pomnožen s pripadnim elementom vektora pomaka jednak nuli.

3.6. Definiranje unutarnjih sila u štapovima

```
def lok_sile (elementi, cvorovi, w_glob):
    w1 = zero_vector (RDF, 6)
    for i in srange(len(elementi)):
        e = elementi[i]
        [x1,y1,z1] = cvorovi [elementi[i][0]]
        [x2,y2,z2] = cvorovi [elementi[i][1]]
        L=sqrt ((x2-x1)^2+(y2-y1)^2+(z2-z1)^2)

        [def_x1, def_y1, def_z1] = vector(cvorovi[e[0]]) +
        vector([w_glob[3*e[0]], w_glob[3*e[0] + 1], w_glob[3*e[0] + 2]])
        [def_x2, def_y2, def_z2] = vector(cvorovi[e[1]]) +
        vector([w_glob[3*e[1]], w_glob[3*e[1] + 1], w_glob[3*e[1] + 2]])
        L_def = sqrt((def_x2 - def_x1)^2 + (def_y2 - def_y1)^2 + (def_z2 - def_z1)^2)
        delta = L_def - L
```

Dijelovi koda *[def_x1, def_y1, def_z1] i [def_x2, def_y2, def_z2]* su prelomljeni da bi se prilagodili stranici. Prvim dijelom koda želimo da algoritam napravi razliku između vlačne i tlačne sile u elementu. To postižemo tako da izračunamo početnu duljinu elementa, te deformiranu. Ako se element produljio, u elementu dolazi do vlačnih, dok kod skraćenja elementa dolazi do tlačnih sila. *[x1, y1, z1]* predstavljaju koordinate prvog, a *[x2, y2, z2]* drugog čvora pojedinog elementa.

```
    x21 = x2 - x1
    y21 = y2 - y1
    z21 = z2 - z1
    l = x21/L
    m = y21/L
    n = z21/L
    ii=elementi[i][0]
    kk=elementi[i][1]
    w1[0]=w_glob [3*ii]
    w1[1]=w_glob [3*ii+1]
    w1[2]=w_glob [3*ii+2]
    w1[3]=w_glob [3*kk]
    w1[4]=w_glob [3*kk+1]
    w1[5]=w_glob [3*kk+2]
    R = matrix([[ l, m, n, 0, 0, 0 ],
                [ l, m, n, 0, 0, 0 ],
                [ l, m, n, 0, 0, 0 ],
                [ 0, 0, 0, l, m, n ],
                [ 0, 0, 0, l, m, n ],
                [ 0, 0, 0, l, m, n ]])
    Ke = matrica_elementa (elementi[i], EA, cvorovi)
    N_gl = Ke*w1
    N_lok = R*N_gl
    if abs(N_lok[0]) > 10^-5 and delta < 0:
        print 'Sila u elementu', str(i)
        print -abs(N_lok[0].n()), '(tlak)'
    if abs(N_lok[0]) > 10^-5 and delta > 0:
        print 'Sila u elementu', str(i)
        print abs(N_lok[0].n()), '(vlak)'
print 'U ostalim štapovima sile su jednake 0!'
```

U drugom dijelu koda cilj nam je iz globalnog vektora pomaka za cijeli sustav izvući pomake za pojedine čvorove svakog elementa, te pomoću njih i globalne matrice krutosti za pojedini element odrediti globalne sile svakog elementa. Zatim te sile matricom transformacije **R** prebacujemo u lokalni sustav i provjerom produljenja ili skraćenja štapa zaključujemo jesu li te sile vlačne ili tlačne.

3.7. Funkcije za grafički prikaz rešetkastog sistema

```
def plot_resetka (elementi, cvorovi, color = 'black', thickness = 5 ) :
    return sum ([line3d ((matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*vector(cvorovi[b[0]]),
    matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*vector(cvorovi[b[1]])), thickness = thickness,
                color = color, aspect_ratio = 1, frame = false) for b in elementi])

def plot_joints3d (cvorovi, lezajevi, rr) :
    cvor = range(len(cvorovi))
    for i in srange(len(cvorovi)):
        if lezajevi[3*i] == 0 or lezajevi[3*i + 1] == 0 or lezajevi[3*i + 2] == 0:
            cvor[i] = 'red'
        else:
            cvor[i] = 'white'
    return sum ([sphere (center = matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*
    vector(cvorovi[j]), size = rr, edgecolor = 'black',
    color = str(cvor[j]),
    fill = True, aspect_ratio = 1, frame = false) for j in srange(len(cvorovi))])

def plot_loads3d (opt, cvorovi, load_scale) :
    pls = []
    for li in srange(len(opt)):
        e = opt[li][0]
        tail = matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]]) * vector(cvorovi[e])
        ff = opt[li][1]
        head = [tail[0] + load_scale*ff[0], tail[1] + load_scale*ff[1],
        tail[2] + load_scale*ff[2]]
        pls.append (arrow3d (head, tail, width = 7, color = 'crimson',
        aspect_ratio = 1, head_radius = 7*load_scale, frame = false))
    return sum (pls)
```

```
def num_cv (cvorovi, scale_cv):
    for i in srange(len(cvorovi)):
        return sum ([text3d (str(j), matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*
        vector(cvorovi[j])
        + vector((scale_cv, scale_cv, scale_cv)), fontweight='bold', frame = false)
        for j in srange(len(cvorovi))])

def num_elem (cvorovi, elementi, scale_elem):
    koor_cv = range (len(elementi))
    for i in srange(len(elementi)):
        elem = elementi[i]
        koor_cv[i] = [(vector(cvorovi[elem[0]], RDF) + vector(cvorovi[elem[1]]))/2]
    return sum ([text3d (str(j), matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*
    vector(koor_cv[j][0]) + vector((scale_elem, scale_elem, scale_elem)),
    fontweight='bold', color='red', frame = false) for j in srange(len(elementi))])

def plot_cv_elem (cvorovi, elementi, scale_cv, scale_elem):
    return num_elem (cvorovi, elementi, scale_elem) + num_cv(cvorovi, scale_cv)
```

```
def plot_sve (cvorovi, elementi, rubni_uvjeti, scale_joints, opt_cvor, scale_loads):
    return plot_joints3d (cvorovi, rubni_uvjeti, scale_joints) +
    plot_loads3d (opt_cvor, cvorovi, scale_loads) + plot_resetka( elementi, cvorovi )

def deformacija (cvorovi, elementi, w_glob, scale, color = 'green', thickness = 5):
    koor_def = range(len(cvorovi))
    for i in srange(len(cvorovi)):
        koor_def[i] = vector([cvorovi[i][0] + scale*w_glob[3*i], cvorovi[i][1] +
        scale*w_glob[3*i + 1], cvorovi[i][2] + scale*w_glob[3*i + 2]])
    return sum ([line3d ((matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*koor_def[b[0]],
    matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*koor_def[b[1]]),
    thickness = thickness, color = color, aspect_ratio = 1, frame = false) for b in elementi])

def plot_lezajevi (cvorovi, lezajevi, w_gl, scale, rr) :
    boja = range(len(cvorovi))
    for i in srange(len(cvorovi)):
        if lezajevi[3*i] == 0 or lezajevi[3*i + 1] == 0 or lezajevi[3*i + 2] == 0:
            boja[i] = 'red'
        else:
            boja[i] = 'white'
    return sum ([sphere (center = matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]]) *
    vector(cvorovi[j]) + matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]]) *
    vector ([scale * w_gl[3*j], scale * w_gl[3*j + 1], scale * w_gl[3*j + 2]]),
    size = rr, edgecolor = 'black',
    color = boja[j],
    fill = True, aspect_ratio = 1) for j in srange(len(cvorovi))])
```

Izrada algoritma za crtanje rešetkastog sustava neće se posebno objašnjavati u ovom radu. Algoritam je dan samo radi potpunosti koda. Važno je napomenuti da Sage koristi *xyz* koordinatni sustav, gdje je pozitivna *os z* usmjerena prema gore, dok se u ovom radu koristi pozitivna *os z* u smjeru gravitacije, pa se sve koordinate množe s matricom [[1, 0, 0], [0, 1, 0], [0, 0, -1]].

3.8. Produljenje - skraćenje elemenata

```
def produljenja (cvorovi, elementi, w_glob):
    for i in srange(len(elementi)):
        e = elementi[i]
        [x1,y1,z1] = cvorovi [e[0]]
        [x2,y2,z2] = cvorovi [e[1]]
        L_1 = sqrt ((x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2)
        [def_x1, def_y1, def_z1] = vector(cvorovi[e[0]]) +
        vector([w_glob[3*e[0]], w_glob[3*e[0] + 1], w_glob[3*e[0] + 2]])
        [def_x2, def_y2, def_z2] = vector(cvorovi[e[1]]) +
        vector([w_glob[3*e[1]], w_glob[3*e[1] + 1], w_glob[3*e[1] + 2]])
        L_2 = sqrt((def_x2 - def_x1)^2 + (def_y2 - def_y1)^2 + (def_z2 - def_z1)^2)
        delta = L_2 - L_1
        N = aks_sile (e, cvorovi, w_glob)
        if N > 10^-5:
            print 'element ', str(i)
            print delta.n()
            print ''
```

* Dijelovi koda [def_x1, def_y1, def_z1] i [def_x2, def_y2, def_z2] su prelomljeni da bi stali na stranicu.

Da bismo smo izračunali produljenje/skraćenje elementa, potrebno je uz početnu duljinu elementa izračunati i duljinu elementa nakon deformacije. Duljinu produljenog odnosno skraćenog elementa računamo pomoću vektora pomaka, tako da za svaki čvor pripadnog elementa izračunavamo nove koordinate. Sada jednostavno oduzmemo od duljine *L_2*, koja nam označava novu duljinu elementa, početnu duljinu elementa *L_1*. Ako nam je *delta > 0* došlo je do produljenja, a ako je *delta < 0* do skraćenja elementa.

4. Primjeri rešetkastih sistema

4.1. Primjer 3. Ravninska rešetka

U prvom primjeru riješit ćemo ravninsku statički određenu rešetku. Kao ulazne podatke unijet ćemo koordinate čvorova, te ih spojiti u elemente. Zadat ćemo vrste ležajeva u pripadnim čvorovima. Kao izlazne podatke program daje pomake čvorova i produljenje elemenata u globalnom koordinatnom sustavu, te unutarnje sile u štapovima.

```
cvorovi  = [[0,0],[1,0],[2,0],[0,-1],[1,-1],[2,-1]]
elementi = [[0,1],[1,2],[3,4],[4,5],[0,3],[0,4],[1,4],[4,2],[2,5]]

print 'čvorovi:'
print cvorovi
print ''
print 'elementi:'
print elementi
```

```
čvorovi:
[[0, 0], [1, 0], [2, 0], [0, -1], [1, -1], [2, -1]]

elementi:
[[0, 1], [1, 2], [3, 4], [4, 5], [0, 3], [0, 4], [1, 4], [4, 2], [2, 5]]
```

cvorovi = [[*x1*, *z1*], [*x2*, *z2*], … [*xn*, *zn*]] – Prvo polje označava prvi čvor. Prvi član označava koordinatu *x* čvora, a drugi koordinatu *z* čvora.

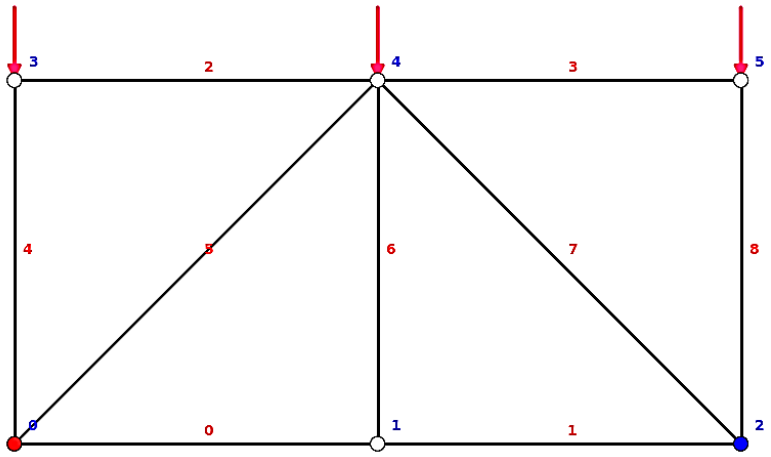
elementi [[ ], [ ], [ ], …] – Prvi član označava početni, a drugi krajnji čvor elementa.

Za naš primjer uzeli smo da su štapovi čelični, promjera 5 centimetara. Modul elastičnosti za čelik iznosi $E = 2{,}1 \cdot 10^8\ KN/m^2$, iz čega slijedi da je $EA = 131250\ KN$.

Nakon toga zadajemo opterećenje u čvorove u kojima djeluju vanjske sile tako da prvo upisujemo broj čvora, a zatim iznos sile u *x* i *z* smjeru. Također zadajemo rubne uvjete pri čemu smo zadali da nam program crta zglobne ležajeve crvenom, klizne u *x* smjeru plavom, te klizne u *y* smjeru narančastom bojom. Ležajeve unosimo tako da za prvi član napišemo broj čvora, a zatim vrstu ležaja. Još smo dodatno ispisali iznose opterećenja za svaki čvor.

```
opt_cvorovi = [[3, [0,100]],[4, [0,100]],[5, [0,100]]]
p = opt (opt_cvorovi, cvorovi)
rubni_uvjeti = lezajevi ([[0, 'zglobni'], [2, 'klizni_x']])
EA = 131250

print 'opterećeni čvorovi:'
print opt_cvorovi
print ''
print 'rubni uvjeti:'
print rubni_uvjeti
print ''
print 'vektor upetosti:'
print p
print ''
print 'aksijalna krutost:'
print 'EA = ', EA, 'kN'
```

```
opterećeni čvorovi:
[[3, [0, 100]], [4, [0, 100]], [5, [0, 100]]]

rubni uvjeti:
[0, 0, 'u1', 'w1', 'u2', 0, 'u3', 'w3', 'u4', 'w4', 'u5', 'w5']

vektor upetosti:
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0)

aksijalna krutost:
EA =  131250 kN
```



Slika 10 Opterećena ravninska rešetka

```
kk = elim(K, rubni_uvjeti, p)
w_glob = kk.inverse()*p
print 'Pomaci čvorova:'
print ''
pomaci(cvorovi, w_glob)
print ''
print 'Sile u štapovima:'
print ''
lok_sile (elementi, cvorovi, w_glob)
```

Nakon toga računamo pomake čvorova množenjem reducirane matrice krutosti s vektorom opterećenja u čvorovima elemenata. Naredbom *print* ispisujemo pomake svakog čvora te sile u štapovima.

```
Sile u štapovima:

Sila u elementu 0
(-50.0, 0.000, 50.0, 0.000)
Sila u elementu 1
(-50.0, 0.000, 50.0, 0.000)
Sila u elementu 2
(0.000, 0.000, 0.000, 0.000)
Sila u elementu 3
(0.000, 0.000, 0.000, 0.000)
Sila u elementu 4
(100., 0.000, -100., 0.000)
Sila u elementu 5
(70.7, 0.000, -70.7, 0.000)
Sila u elementu 6
(0.000, 0.000, 0.000, 0.000)
Sila u elementu 7
(70.7, 0.000, -70.7, 0.000)
Sila u elementu 8
(100., 0.000, -100., 0.000)
```

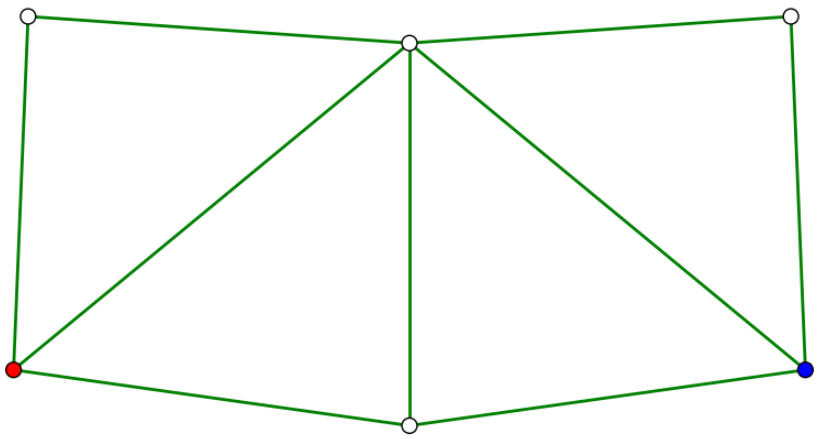
```
Pomaci čvorova:

u_0 = 0.0
w_0 = 0.0
u_1 = 0.000380952380952
w_1 = 0.00145844842847
u_2 = 0.000761904761905
w_2 = 0.0
u_3 = 0.000380952380952
w_3 = 0.000761904761905
u_4 = 0.000380952380952
w_4 = 0.00145844842847
u_5 = 0.000380952380952
w_5 = 0.000761904761905
```

Još samo ostaje da ispišemo produljenje svakog štapa u konstrukciji, pozivajući funkciju *produljenja*, te prikažemo deformirani sustav.

```
print 'Produljenje ili skraćenje štapova:'
print ''
produljenja (cvorovi, elementi, w_glob)
```

```
Produljenje ili skraćenje štapova:

element  0
0.00038202

element  1
0.00038202

element  2
2.4259e-7

element  3
2.4259e-7

element  4
-0.00076183

element  5
-0.00076103

element  6
0.00000

element  7
-0.00076103

element  8
-0.00076183
```

```
def_oblik (cvorovi, elementi, rubni_uvjeti, w_glob, 100, 0.02)
```



Slika 11 Deformirana ravninska rešetka

4.2. Primjer 4. Schwedlerova kupola

```
def schwedler_2 (r0, hh, n, ccw = True) :
    jsl = []
    bsl = []
    ssl = []
    rsl = [r0]

    h0 = hh[-1]
    rr = r0^2/(2*h0) + h0/2
    aa = rr - h0
    for h in hh[:-1] :
        rsl.append (sqrt (rr^2 - (aa+h)^2))
    hh.insert (0, 0.0)
    alpha = RDF (2*pi/n)
    for i in xrange (len (rsl)) :
        for j in xrange (n) :
            x = rsl[i] * cos (j*alpha)
            y = rsl[i] * sin (j*alpha)
            z = hh[i]
            jsl.append ((x, y, z))

    for i in xrange (0, len (hh) - 2) :
        n0 = i*n
        for j in xrange (n) :
            bsl.append ((n0+j, n0+j+n))
        for j in xrange (n) :
            bsl.append ((n0+n+j, n*(i+1)+(n0+j+1)%n))
        if ccw == True :
            for j in xrange (0, n) :
                bsl.append ((n0+j, (n*(i+1)+(n0+j+1)%n)))
        else :
            for j in xrange (0, n) :
                bsl.append ((n0+j, (n*(i+1)+(n0+j-1)%n)))

    ssl = range (n)
    return jsl, bsl, ssl
```

Prije početka rješavanja Schwedlerove kupole, moramo dodatno definirati način brojanja čvorova i njihovog spajanja u elemente. Naredbu *schwedler_2* pozivamo s *r0* što predstavlja radijus baze, te listom visina i brojem čvorova. Nakon toga unosimo ulazne podatke na isti način kao i za *Primjer 1*.

```
čvorovi:
[(10.0000000000000, 0.000000000000000, 0.000000000000000),
(7.07106781186548, 7.07106781186547, 0.000000000000000),
(6.12323399573676e-16, 10.0000000000000, 0.000000000000000),
(-7.07106781186547, 7.07106781186548, 0.000000000000000),
(-10.0000000000000, 1.22464679914735e-15, 0.000000000000000),
(-7.07106781186548, -7.07106781186547, 0.000000000000000),
(-1.83697019872103e-15, -10.0000000000000, 0.000000000000000),
(7.07106781186547, -7.07106781186548, 0.000000000000000),
(8.51102226527460, 0.000000000000000, -5.25000000000000),
(6.01820155860536, 6.01820155860536, -5.25000000000000),
(5.21149808732019e-16, 8.51102226527460, -5.25000000000000),
(-6.01820155860536, 6.01820155860536, -5.25000000000000),
(-8.51102226527460, 1.04229961746404e-15, -5.25000000000000),
(-6.01820155860536, -6.01820155860536, -5.25000000000000),
(-1.56344942619606e-15, -8.51102226527460, -5.25000000000000),
(6.01820155860536, -6.01820155860536, -5.25000000000000),
(3.79967103839267, 0.000000000000000, -9.25000000000000),
(2.68677315752558, 2.68677315752558, -9.25000000000000),
(2.32662748749024e-16, 3.79967103839267, -9.25000000000000),
(-2.68677315752558, 2.68677315752558, -9.25000000000000),
(-3.79967103839267, 4.65325497498047e-16, -9.25000000000000),
(-2.68677315752559, -2.68677315752558, -9.25000000000000),
(-6.97988246247071e-16, -3.79967103839267, -9.25000000000000),
(2.68677315752558, -2.68677315752559, -9.25000000000000)]
```

```
elementi:
[(0, 8), (1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15),
(8, 9), (9, 10), (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15,
8), (0, 9), (1, 10), (2, 11), (3, 12), (4, 13), (5, 14), (6, 15), (7,
8), (8, 16), (9, 17), (10, 18), (11, 19), (12, 20), (13, 21), (14, 22),
(15, 23), (16, 17), (17, 18), (18, 19), (19, 20), (20, 21), (21, 22),
(22, 23), (23, 16), (8, 17), (9, 18), (10, 19), (11, 20), (12, 21), (13,
22), (14, 23), (15, 16)]
```

```
def lezl (l) :
    zg = ['zglobni'] * len (l)
    return zip (l, zg)

def joints_transf (joints):
    joints_t = range(len(joints))
    for i in srange(len(joints)):
        joints_t[i] = matrix([[1, 0, 0], [0, 1, 0], [0, 0, -1]])*vector(joints[i])
    return joints_t
```

U Schwedlerovoj kupoli imamo samo zglobne ležajeve. Naredbom *lezl* definiramo čvorove kojima su spriječeni pomaci, tj. ležajeve.

```
opt_cvorovi = sile_cv (cvorovi, supports, 0, 0, 100)
p = opt (opt_cvorovi, cvorovi)
rubni_uvjeti = lezajevi (lezl (supports))
EA = 1000000

print 'opterećeni čvorovi:'
print opt_cvorovi
print ''
print 'rubni uvjeti:'
print rubni_uvjeti
print ''
print 'vektor upetosti:'
print p
print ''
print 'aksijalna krutost:'
print 'EA = ', EA, 'kN'
```

```
opterećeni čvorovi:
[[8, [0, 0, 100]], [9, [0, 0, 100]], [10, [0, 0, 100]], [11, [0, 0,
100]], [12, [0, 0, 100]], [13, [0, 0, 100]], [14, [0, 0, 100]], [15, [0,
0, 100]], [16, [0, 0, 100]], [17, [0, 0, 100]], [18, [0, 0, 100]], [19,
[0, 0, 100]], [20, [0, 0, 100]], [21, [0, 0, 100]], [22, [0, 0, 100]],
[23, [0, 0, 100]]]

rubni uvjeti:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
'ux_8', 'uy_8', 'uz_8', 'ux_9', 'uy_9', 'uz_9', 'ux_10', 'uy_10',
'uz_10', 'ux_11', 'uy_11', 'uz_11', 'ux_12', 'uy_12', 'uz_12', 'ux_13',
'uy_13', 'uz_13', 'ux_14', 'uy_14', 'uz_14', 'ux_15', 'uy_15', 'uz_15',
'ux_16', 'uy_16', 'uz_16', 'ux_17', 'uy_17', 'uz_17', 'ux_18', 'uy_18',
'uz_18', 'ux_19', 'uy_19', 'uz_19', 'ux_20', 'uy_20', 'uz_20', 'ux_21',
'uy_21', 'uz_21', 'ux_22', 'uy_22', 'uz_22', 'ux_23', 'uy_23', 'uz_23']

vektor upetosti:
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0,
0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0)

aksijalna krutost:
EA =  1000000 kN
```

```
kk = elim(K, rubni_uvjeti, p)
w_glob = kk.inverse()*p
print 'Pomaci čvorova:'
print ''
pomaci(cvorovi, w_glob)
```

```
ux_8 = 0.000679007698093
uy_8 = 0.00059425958863
uz_8 = 0.000986626668156
ux_9 = 5.9925962894e-05
uy_9 = 0.000900335932704
uz_9 = 0.000986626668156
ux_10 = -0.00059425958863
uy_10 = 0.000679007698093
uz_10 = 0.000986626668156
ux_11 = -0.000900335932704
uy_11 = 5.99259628939e-05
uz_11 = 0.000986626668156
ux_12 = -0.000679007698093
uy_12 = -0.00059425958863
uz_12 = 0.000986626668156
ux_13 = -5.99259628939e-05
uy_13 = -0.000900335932704
uz_13 = 0.000986626668156
ux_14 = 0.00059425958863
uy_14 = -0.000679007698093
uz_14 = 0.000986626668156
ux_15 = 0.000900335932704
uy_15 = -5.9925962894e-05
uz_15 = 0.000986626668156
ux_16 = -0.000584738693084
uy_16 = 0.00136259573501
uz_16 = 0.00395055308855
ux_17 = -0.00137697337934
uy_17 = 0.00055002798914
uz_17 = 0.00395055308855
ux_18 = -0.00136259573501
uy_18 = -0.000584738693083
uz_18 = 0.00395055308855
ux_19 = -0.000550027989139
uy_19 = -0.00137697337934
uz_19 = 0.00395055308855
ux_20 = 0.000584738693083
uy_20 = -0.00136259573501
uz_20 = 0.00395055308855
ux_21 = 0.00137697337934
uy_21 = -0.00055002798914
uz_21 = 0.00395055308855
ux_22 = 0.00136259573501
uy_22 = 0.000584738693084
uz_22 = 0.00395055308855
ux_23 = 0.00055002798914
uy_23 = 0.00137697337934
uz_23 = 0.00395055308855
```

```
print 'Sile u štapovima:'
print ''
lok_sile (elementi, cvorovi, w_glob)
```

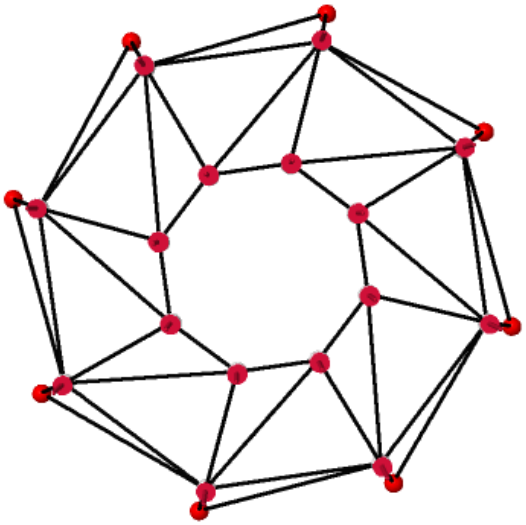
```
Sila u elementu 0
-207.888177492699 (tlak)
Sila u elementu 1
-207.888177492699 (tlak)
Sila u elementu 2
-207.888177492701 (tlak)
Sila u elementu 3
-207.888177492698 (tlak)
Sila u elementu 4
-207.888177492701 (tlak)
Sila u elementu 5
-207.888177492697 (tlak)
Sila u elementu 6
-207.888177492701 (tlak)
Sila u elementu 7
-207.888177492699 (tlak)
Sila u elementu 8
79.7798051666405 (vlak)
Sila u elementu 9
79.7798051666397 (vlak)
Sila u elementu 10
79.7798051666408 (vlak)
Sila u elementu 11
79.7798051666396 (vlak)
Sila u elementu 12
79.7798051666411 (vlak)
Sila u elementu 13
79.7798051666394 (vlak)
Sila u elementu 14
79.7798051666411 (vlak)
Sila u elementu 15
Sila u elementu 24
-154.508960870887 (tlak)
Sila u elementu 25
-154.508960870883 (tlak)
Sila u elementu 26
-154.508960870910 (tlak)
Sila u elementu 27
-154.508960870873 (tlak)
Sila u elementu 28
-154.508960870885 (tlak)
Sila u elementu 29
-154.508960870882 (tlak)
Sila u elementu 30
-154.508960870886 (tlak)
Sila u elementu 31
-154.508960870882 (tlak)
Sila u elementu 32
-153.891925689224 (tlak)
Sila u elementu 33
-153.891925689106 (tlak)
Sila u elementu 34
-153.891925689208 (tlak)
Sila u elementu 35
-153.891925689229 (tlak)
Sila u elementu 36
-153.891925689222 (tlak)
Sila u elementu 37
-153.891925689227 (tlak)
Sila u elementu 38
-153.891925689221 (tlak)
Sila u elementu 39
-153.891925689226 (tlak)
```

Sile u ostalim štapovima iznose 0.

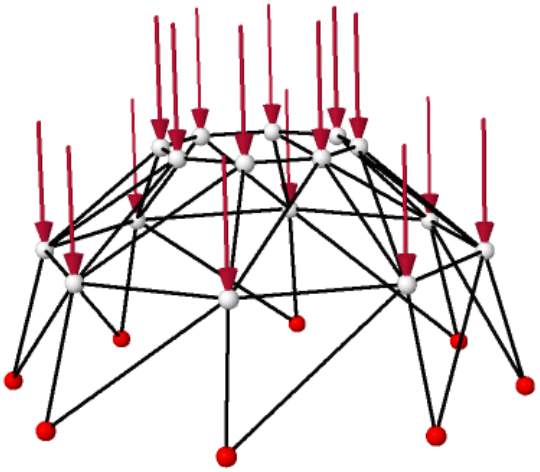
```
print 'Produljenja štapova:'
print ''
produljenja (cvorovi, elementi, w_glob)
```

```
Produljenja štapova:

element  0
-0.00113441334698816

element  1
-0.00113441334698905

element  2
-0.00113441334698816

element  3
-0.00113441334698905

element  4
-0.00113441334698816

element  5
-0.00113441334698905

element  6
-0.00113441334698816

element  7
-0.00113441334698905

element  8
0.000519705870281939

element  9
0.000519705870281939

element  10
0.000519705870281939

element  11
0.000519705870281051

element  12
0.000519705870281939

element  13
0.000519705870281939

element  14
0.000519705870281051

element  15
0.000519705870281051

element  24
-0.000954106734332605

element  25
-0.000954106734334381

element  26
-0.000954106734334381

element  27
-0.000954106734334381

element  28
-0.000954106734334381

element  29
-0.000954106734334381

element  30
-0.000954106734334381

element  31
-0.000954106734333493

element  32
-0.000447352597419126

element  33
-0.000447352597418682

element  34
-0.000447352597419570

element  35
-0.000447352597419570

element  36
-0.000447352597419126

element  37
-0.000447352597418682

element  38
-0.000447352597419126

element  39
-0.000447352597419126
```

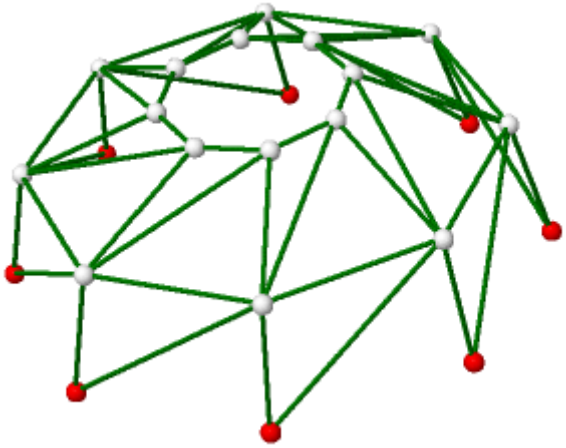
U ostalim štapovima produljenja iznose 0.



Slika 12 Tlocrt Schwedlerove kupole



Slika 13 Opterećena Schwedlerova kupola



Slika 14 Deformirana Schwedlerova kupola

4.3. Primjer 5. Statički neodređena „Schwedlerova“ kupola

U svrhu usporedbe rješenja, te načina izvedbe, prikazat će se ponovno „Schwedlerova“ kupola iz *Primjera 2*.

```
def schwedler_sind (r0, hh, n) :
    jsl = []
    bsl = []
    ssl = []
    rsl = [r0]

    h0 = hh[-1]
    rr = r0^2/(2*h0) + h0/2
    aa = rr - h0
    for h in hh[:-1] :
        rsl.append (sqrt (rr^2 - (aa+h)^2))
    hh.insert (0, 0.0)
    alpha = RDF (2*pi/n)
    for i in xrange (len (rsl)) :
        for j in xrange (n) :
            x = rsl[i] * cos (j*alpha)
            y = rsl[i] * sin (j*alpha)
            z = hh[i]
            jsl.append ((x, y, z))

    for i in xrange (0, len (hh) - 2) :
        n0 = i*n
        for j in xrange (n) :
            bsl.append ((n0+j, n0+j+n))
        for j in xrange (n) :
            bsl.append ((n0+n+j, n*(i+1)+(n0+j+1)%n))
        for j in xrange (0, n) :
            bsl.append ((n0+j, (n*(i+1)+(n0+j+1)%n)))
        for j in xrange (0, n) :
            bsl.append ((n0+j, (n*(i+1)+(n0+j-1)%n)))

    ssl = range (n)
    return jsl, bsl, ssl
```

Zadatak započinjemo na isti način kao i u *Primjeru 2,* definiranjem načina brojanja čvorova, te njihovog spajanja u elemente.

```
joints, bars, supports = schwedler_sind (15., [3.25, 6.25, 7.], 8)
```

Naredbu *schwedler_sind* pozivamo s radijusom baze (15 m), visinama katova mjerenim od baze (3.25, 6.25 m), najvišom točkom kupole (7.0 m), te brojem zglobnih čvorova u bazi.

```
čvorovi:
[(15.0000000000000, 0.000000000000000, 0.000000000000000),
(10.6066017177982, 10.6066017177982, 0.000000000000000),
(9.18485099360514e-16, 15.0000000000000, 0.000000000000000),
(-10.6066017177982, 10.6066017177982, 0.000000000000000),
(-15.0000000000000, 1.83697019872103e-15, 0.000000000000000),
(-10.6066017177982, -10.6066017177982, 0.000000000000000),
(-2.75545529808154e-15, -15.0000000000000, 0.000000000000000),
(10.6066017177982, -10.6066017177982, 0.000000000000000),
(11.5205561621700, 0.000000000000000, -3.25000000000000),
(8.14626338531091, 8.14626338531091, -3.25000000000000),
(7.05430611419942e-16, 11.5205561621700, -3.25000000000000),
(-8.14626338531091, 8.14626338531091, -3.25000000000000),
(-11.5205561621700, 1.41086122283988e-15, -3.25000000000000),
(-8.14626338531091, -8.14626338531091, -3.25000000000000),
(-2.11629183425983e-15, -11.5205561621700, -3.25000000000000),
(8.14626338531091, -8.14626338531091, -3.25000000000000),
(5.36606400047025, 0.000000000000000, -6.25000000000000),
(3.79438024301353, 3.79438024301353, -6.25000000000000),
(3.28576655109786e-16, 5.36606400047025, -6.25000000000000),
(-3.79438024301353, 3.79438024301353, -6.25000000000000),
(-5.36606400047025, 6.57153310219573e-16, -6.25000000000000),
(-3.79438024301353, -3.79438024301353, -6.25000000000000),
(-9.85729965329359e-16, -5.36606400047025, -6.25000000000000),
(3.79438024301353, -3.79438024301353, -6.25000000000000)]
```

```
elementi:
[(0, 8), (1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15),
(8, 9), (9, 10), (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15,
8), (0, 9), (1, 10), (2, 11), (3, 12), (4, 13), (5, 14), (6, 15), (7,
8), (0, 15), (1, 8), (2, 9), (3, 10), (4, 11), (5, 12), (6, 13), (7,
14), (8, 16), (9, 17), (10, 18), (11, 19), (12, 20), (13, 21), (14, 22),
(15, 23), (16, 17), (17, 18), (18, 19), (19, 20), (20, 21), (21, 22),
(22, 23), (23, 16), (8, 17), (9, 18), (10, 19), (11, 20), (12, 21), (13,
22), (14, 23), (15, 16), (8, 23), (9, 16), (10, 17), (11, 18), (12, 19),
(13, 20), (14, 21), (15, 22)]
```

```
opt_cvorovi = sile_cv (cvorovi, supports, 0, 0, 100)
p = opt (opt_cvorovi, cvorovi)
rubni_uvjeti = lezajevi (lezl (supports))
EA = 1000000

print 'opterećeni čvorovi:'
print opt_cvorovi
print ''
print 'rubni uvjeti:'
print rubni_uvjeti
print ''
print 'vektor upetosti:'
print p
print ''
print 'aksijalna krutost:'
print 'EA = ', EA, 'kN'
```

```
opterećeni čvorovi:
[[8, [0, 0, 100]], [9, [0, 0, 100]], [10, [0, 0, 100]], [11, [0, 0,
100]], [12, [0, 0, 100]], [13, [0, 0, 100]], [14, [0, 0, 100]], [15, [0,
0, 100]], [16, [0, 0, 100]], [17, [0, 0, 100]], [18, [0, 0, 100]], [19,
[0, 0, 100]], [20, [0, 0, 100]], [21, [0, 0, 100]], [22, [0, 0, 100]],
[23, [0, 0, 100]]]

rubni uvjeti:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
'ux_8', 'uy_8', 'uz_8', 'ux_9', 'uy_9', 'uz_9', 'ux_10', 'uy_10',
'uz_10', 'ux_11', 'uy_11', 'uz_11', 'ux_12', 'uy_12', 'uz_12', 'ux_13',
'uy_13', 'uz_13', 'ux_14', 'uy_14', 'uz_14', 'ux_15', 'uy_15', 'uz_15',
'ux_16', 'uy_16', 'uz_16', 'ux_17', 'uy_17', 'uz_17', 'ux_18', 'uy_18',
'uz_18', 'ux_19', 'uy_19', 'uz_19', 'ux_20', 'uy_20', 'uz_20', 'ux_21',
'uy_21', 'uz_21', 'ux_22', 'uy_22', 'uz_22', 'ux_23', 'uy_23', 'uz_23']

vektor upetosti:
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0,
0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0,
100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0)

aksijalna krutost:
EA =  1000000 kN
```

```
kk = elim(K, rubni_uvjeti, p)
w_glob = kk.inverse()*p
print 'Pomaci čvorova:'
print ''
pomaci(cvorovi, w_glob)
```

```
ux_8 = 0.000628719382057
uy_8 = -1.08420217249e-19
uz_8 = 0.00120900773967
ux_9 = 0.000444571738516
uy_9 = 0.000444571738516
uz_9 = 0.00120900773967
ux_10 = -3.87941089842e-19
uy_10 = 0.000628719382057
uz_10 = 0.00120900773967
ux_11 = -0.000444571738516
uy_11 = 0.000444571738516
uz_11 = 0.00120900773967
ux_12 = -0.000628719382057
uy_12 = -6.42050974019e-19
uz_12 = 0.00120900773967
ux_13 = -0.000444571738516
uy_13 = -0.000444571738516
uz_13 = 0.00120900773967
ux_14 = -1.62630325873e-18
uy_14 = -0.000628719382057
uz_14 = 0.00120900773967
ux_15 = 0.000444571738516
uy_15 = -0.000444571738516
uz_15 = 0.00120900773967
```

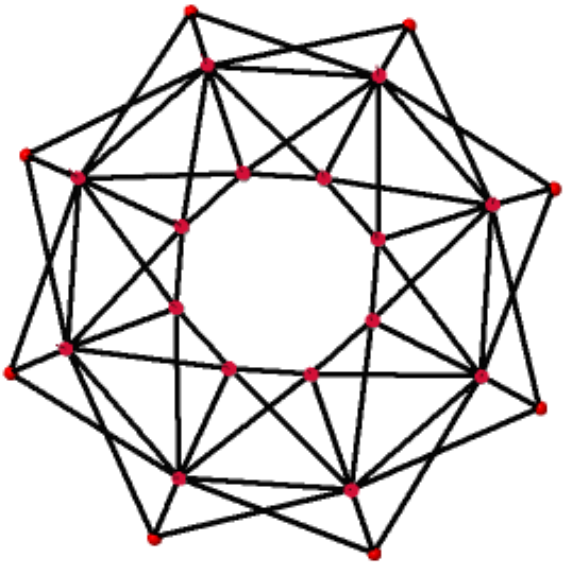
```
ux_16 = -0.000996314787908
uy_16 = -1.30104260698e-18
uz_16 = 0.00611007083625
ux_17 = -0.000704500942726
uy_17 = -0.000704500942726
uz_17 = 0.00611007083625
ux_18 = -1.67780286192e-17
uy_18 = -0.000996314787908
uz_18 = 0.00611007083625
ux_19 = 0.000704500942726
uy_19 = -0.000704500942726
uz_19 = 0.00611007083625
ux_20 = 0.000996314787908
uy_20 = 2.73761048553e-18
uz_20 = 0.00611007083625
ux_21 = 0.000704500942726
uy_21 = 0.000704500942726
uz_21 = 0.00611007083625
ux_22 = -3.03576608296e-18
uy_22 = 0.000996314787908
uz_22 = 0.00611007083625
ux_23 = -0.000704500942726
uy_23 = 0.000704500942726
uz_23 = 0.00611007083625
```

```
print 'Sile u štapovima:'
print ''
lok_sile (elementi, cvorovi, w_glob)
```

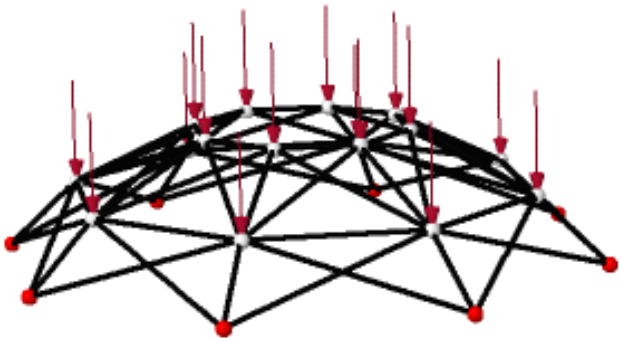
```
Sila u elementu 0
-269.833737477785 (tlak)
Sila u elementu 1
-269.833737477784 (tlak)
Sila u elementu 2
-269.833737477785 (tlak)
Sila u elementu 3
-269.833737477784 (tlak)
Sila u elementu 4
-269.833737477784 (tlak)
Sila u elementu 5
-269.833737477784 (tlak)
Sila u elementu 6
-269.833737477783 (tlak)
Sila u elementu 7
-269.833737477784 (tlak)
Sila u elementu 8
54.5737005407560 (vlak)
Sila u elementu 9
54.5737005407561 (vlak)
Sila u elementu 10
54.5737005407561 (vlak)
Sila u elementu 11
54.5737005407559 (vlak)
Sila u elementu 12
54.5737005407558 (vlak)
Sila u elementu 13
54.5737005407560 (vlak)
Sila u elementu 14
54.5737005407559 (vlak)
Sila u elementu 15
54.5737005407557 (vlak)
Sila u elementu 16
-27.0759766183199 (tlak)
Sila u elementu 17
-27.0759766183195 (tlak)
Sila u elementu 18
-27.0759766183197 (tlak)
Sila u elementu 19
-27.0759766183197 (tlak)
Sila u elementu 20
-27.0759766183198 (tlak)
Sila u elementu 21
-27.0759766183199 (tlak)
Sila u elementu 22
-27.0759766183196 (tlak)
Sila u elementu 23
-27.0759766183195 (tlak)
Sila u elementu 24
-27.0759766183202 (tlak)
Sila u elementu 25
-27.0759766183195 (tlak)
Sila u elementu 26
-27.0759766183199 (tlak)
Sila u elementu 27
-27.0759766183196 (tlak)
Sila u elementu 28
-27.0759766183197 (tlak)
Sila u elementu 29
-27.0759766183198 (tlak)
Sila u elementu 30
-27.0759766183194 (tlak)
Sila u elementu 31
-27.0759766183196 (tlak)
Sila u elementu 32
-100.301888298475 (tlak)
Sila u elementu 33
-100.301888298474 (tlak)
Sila u elementu 34
-100.301888298474 (tlak)
Sila u elementu 35
-100.301888298474 (tlak)
Sila u elementu 36
-100.301888298474 (tlak)
Sila u elementu 37
-100.301888298475 (tlak)
Sila u elementu 38
-100.301888298474 (tlak)
Sila u elementu 39
-100.301888298475 (tlak)
Sila u elementu 40
-185.669568574023 (tlak)
Sila u elementu 41
-185.669568574015 (tlak)
Sila u elementu 42
-185.669568574021 (tlak)
Sila u elementu 43
-185.669568574021 (tlak)
Sila u elementu 44
-185.669568574024 (tlak)
Sila u elementu 45
-185.669568574024 (tlak)
Sila u elementu 46
-185.669568574024 (tlak)
Sila u elementu 47
-185.669568574024 (tlak)
Sila u elementu 48
-85.1551807292965 (tlak)
Sila u elementu 49
-85.1551807292924 (tlak)
Sila u elementu 50
-85.1551807292947 (tlak)
Sila u elementu 51
-85.1551807292966 (tlak)
Sila u elementu 52
-85.1551807292962 (tlak)
Sila u elementu 53
-85.1551807292959 (tlak)
Sila u elementu 54
-85.1551807292965 (tlak)
Sila u elementu 55
-85.1551807292960 (tlak)
Sila u elementu 56
-85.1551807292971 (tlak)
Sila u elementu 57
-85.1551807292958 (tlak)
Sila u elementu 58
-85.1551807292955 (tlak)
Sila u elementu 59
-85.1551807292955 (tlak)
Sila u elementu 60
-85.1551807292935 (tlak)
Sila u elementu 61
-85.1551807292957 (tlak)
Sila u elementu 62
-85.1551807292963 (tlak)
Sila u elementu 63
-85.1551807292959 (tlak)
```

```
print 'Produljenja štapova:'
print ''
produljenja (cvorovi, elementi, w_glob)
```

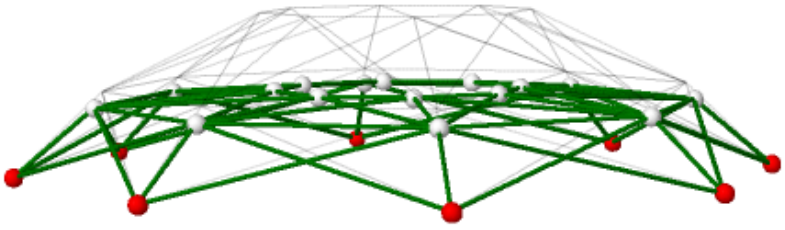
```
     element  0                element  21                element  42
-0.00128471081734993      -0.000301301718261371      -0.000762546325505653

     element  1                element  22                element  43
-0.00128471081734993      -0.000301301718261371      -0.000762546325505653

     element  2                element  23                element  44
-0.00128471081734993      -0.000301301718261371      -0.000762546325505653

     element  3                element  24                element  45
-0.00128471081734993      -0.000301301718261371      -0.000762546325505653

     element  4                element  25                element  46
-0.00128471081734993      -0.000301301718263147      -0.000762546325505653

     element  5                element  26                element  47
-0.00128471081734993      -0.000301301718263147      -0.000762546325505653

     element  6                element  27                element  48
-0.00128471081734993      -0.000301301718263147      -0.000774816969178360

     element  7                element  28                element  49
-0.00128471081734993      -0.000301301718263147      -0.000774816969178360

     element  8                element  29                element  50
0.000481200982241248      -0.000301301718263147      -0.000774816969178360

     element  9                element  30                element  51
0.000481200982239471      -0.000301301718263147      -0.000774816969178360

     element  10               element  31                element  52
0.000481200982241248      -0.000301301718263147      -0.000774816969178360

     element  11               element  32                element  53
0.000481200982239471      -0.000684827653554265      -0.000774816969178360

     element  12               element  33                element  54
0.000481200982241248      -0.000684827653553377      -0.000774816969178360

     element  13               element  34                element  55
0.000481200982239471      -0.000684827653554265      -0.000774816969178360

     element  14               element  35                element  56
0.000481200982241248      -0.000684827653553377      -0.000774816969178360

     element  15               element  36                element  57
0.000481200982239471      -0.000684827653554265      -0.000774816969178360

     element  16               element  37                element  58
-0.000301301718263147     -0.000684827653554265      -0.000774816969178360

     element  17               element  38                element  59
-0.000301301718261371     -0.000684827653554265      -0.000774816969178360

     element  18               element  39                element  60
-0.000301301718263147     -0.000684827653554265      -0.000774816969178360

     element  19               element  40                element  61
-0.000301301718263147     -0.000762546325505653      -0.000774816969178360

     element  20               element  41                element  62
-0.000301301718263147     -0.000762546325505653      -0.000774816969178360

                                                       element  63
                                                   -0.000774816969178360
```



Slika 15 Tlocrt „Schwedlerove“ kupole



Slika 16 „Schwedlerova“ kupola



Slika 17 Deformirana „Schwedlerova“ kupola

5. Literatura

[1] Programski paket *Sage*, dostupno na adresi www.sagemath.org

[2] Franjičić, E.: *Klasifikacija sklopova zglobnih štapova:* završni rad, Građevinski fakultet, Zagreb, 2015.

[3] Fresl K.: *Građevna statika 1.: bilješke i skice s predavanja*, dostupno na mrežnoj stranici predmeta