SVEUČILIŠTE U ZAGREBU

GRAĐEVINSKI FAKULTET

PROGRAMSKA REALIZACIJA METODE SILA ZA REŠETKASTE SISTEME

ZAVRŠNI RAD

Student: Adriana Panižić, 0246038966
Mentor: izv. prof. dr. sc. Krešimir Fresl

Rujan, 2016

Zahvaljujem se mentoru izv. prof. dr. sc. Krešimiru Freslu
na predloženoj temi, stručnoj pomoći i savjetima tijekom izrade završnog rada.

Sadržaj

1. UVOD

Kada se postave jednadžbe ravnoteže svih slobodnih čvorova nekog rešetkastog sistema pogodno ih je zapisati u matričnom obliku. Rešetkasti sistem može biti statički određen sistem, mehanizam ili statički neodređen sistem te s obzirom na to odabire se metoda rješavanja takvog sistema.

Matrica koeficijenata jednadžbi ravnoteže svih slobodnih čvorova nekog rešetkastog sistema može biti kvadratna. Ako je takva matrica regularna, tada postoji jedinstveno rješenje i rešetka je statički određen sistem. Ako je matrica singularna, rešetkasti je sistem mehanizam. Ako je matrica pravokutna, s većim brojem stupaca od broja retaka, onda sistem ima višak štapova. Tada ne postoji jedinstveno rješenje i takav je sistem statički neodređen (a može sadržavati mehanizam). Ako je matrica pravokutna, s manjim brojem stupaca od broja retaka, sustav ima manjak štapova, pa je mehanizam.

U ovom ćemo se radu baviti mehanikom statički neodređenih geometrijski nepromjenjivih rešetkastih sistema. Prikazat ćemo detaljan izvod jednadžbi ravnoteže i kinematičkih jednadžbi koje povezuju produljenje i pomake. Izvest ćemo i objasniti programsku realizaciju određivanja nepoznatih sila u štapovima statički neodređenih rešetkastih sistema postupkom koji se zove metoda sila. Također, prikazat ćemo nekoliko primjera programskog proračuna ravninskih i prostornih štapnih sustava metodom sila.

2. KLASIFIKACIJA REŠETKASTIH SISTEMA

Rešetkasti sistemi

Štapni element je konstrukcijski element kojem je jedna karakteristična dimenzija, duljina, istaknuta u odnosu na druge dvije, visinu (ili debljinu) i širinu. Za njegovu se proračunsku shemu može stoga uzeti dio krivulje ili dio pravca.
Sistemi sastavljeni od štapnih elemenata nazivaju se **štapnim sistemima.**
Zglobni štap je štap kod kojeg su veze štapa s podlogom ili drugim dijelovima konstrukcija zglobne i uz to veze smiju biti samo na njegovim krajevima.

Slobodni čvorovi su zglobni čvorovi u kojima se štapovi spajaju sa drugim štapovima sistema. **Ležajni čvorovi** su čvorovi u kojima se štapovi sistema spajaju sa podlogom.

Unutarnje su sile u zglobnim štapovima uzdužna, a mogu biti **vlačne** ili **tlačne**. Posljedice djelovanja tih sila su produljenje ili skraćenje štapa. Matematički se vlačne i tlačne sile u štapovima razliku po predznacima.

Štapni se sistemi mogu podijeliti na **ravninske** i **prostorne**.

Sistem je **ravninski** ako osi svih elemenata leže u istoj ravnini, u odnosu na koju su uz to svi poprečni presjeci simetrični, ako su statičke i kinematičke karaktersitike svih spojeva simetrične u odnosu na tu ravninu te ako su pravci djelovanja rezultanata zadanih vanjskih sila u svim ravninama poprečih presjeka u toj ravnini, a vektori svih zadanih vanjskih momenata okomiti na nju. U tom će slučaju i sile u vezama s podlogom i rezultirajuće sile unutarnjih sila u poprečnim presjecima biti u toj ravnini, dok će vektori momenata u vezama i rezultirajućih momenata unutarnjih sila biti okomiti na nju. **Prostorni sistemi** su sistemi kod kojih su osi (nekih) elemenata prostorne krivulje, sistemi u kojima osi, iako su ravninske krivulje, ne leže u jednoj ravnini te sistemi čije su osi u jednoj ravnini, ali pravci djelovanja vanjskih sila nisu u toj ravnini.

Rešetkasti sistemi su sistemi kod kojih su zglobnim čvorovima povezani zglobni štapovi, a vanjske sile djeluju samo u čvorovima pa u štapovima postoje samo uzdužne sile.

Statička i kinematička određenost rešetkastih sistema

Sistem je **statički** određen ako je broj jednadžbi ravnoteže jednak broju nepoznanica. Uzdužne sile u svim štapovima mogu se odrediti pomoću jednažbi ravnoteže za dani skup vanjskih sila koje djeluju u čvorovima. **Kinematički određen** sistem je geometrijski nepromijenjiv sistem , odnosno onaj sustem kod kojeg su položaji čvorova jedinstveno određeni duljinama štapova. Kada se jednadžbe ravnoteže takvog sistema napišu u matričnom zapisu matrica koeficijenata je kvadratna, a rang matrice je jednak broju redaka matrice, odnosno broju stupaca. Takva matrica koeficijenata je regularna. Rješenje sustava jednadžbi je jedinstveno, a sustav je statički određen i geometrijski nepromijenjiv.

Ako statički određenom sistemu uklonimo jednu ili više veza dobijemo **mehanizam** koji ima jedan stupanj slobode. Ako se ukloni više veza sistem može imati i više stupnjeva slobode. Broj jednadžbi ravnoteže je veći od broja nepoznanica. Sustav ima manje veza nego što je potrebno da bi bio geometrijski nepromijenjiv.

Ako je broj nepoznatih sila u spojevima veći od broja jednadžbi ravnoteže sistem je **statički neodređen.** Takav sustav ima višak štapova (veza). Može ostati u stanju ravnoteže pri bilo kakvom opterećenju, ali jednadžbama ravnoteže se ne mogu izračunati vrijednosti uzdužnih sila u štapovima. Broj viška štapova (veza) određuje stupanj statičke neodređenosti sistema. Matrica koeficijenata je pravokutna (ima više stupaca nego redaka). Ako je rang matrice koeficijenata statički neodređenog rešetkastog nosača jednak broju redaka sistem je geometrijski nepromijenjiv. Takav, statički neodređen i kinematički određen sistem ima rješenje, ali ono nije jedinstveno.

3. METODA SILA ZA REŠETKASTE SISTEME

Metoda sila je jedna od metoda proračuna statički neodređenih sistema.

Nepoznanice su sile i momenti u prekobrojnim vezama. Za proračun metodom sila potrebno je odrediti stupanj statičke neodređenosti zadanog sistema. Zadani se statički neodređeni sistem u proračunu zamjenjuje osnovnim sistemom koji nastaje tako da se zadanom sistemu ukloni broj prekobrojnih štapova. Osnovni sistem je statički određen i geometrijski nepromijenjiv.Na mjestima raskinutih veza sistem se optereti odgoovarajućim silama i momentima.

Osnovne jednadžbe metode sila dobivaju se iz uvjeta kompatibilnosti koji govori da na osnovnom sistemu pomaci hvatišta zamjenjujućih sila po pravcima njihovih djelovanja, koji su nastali zbog zadanih vanjskih djelovanja i zbog svih zamjenjujućih sila, moraju biti jednaki nuli. Drugim riječima: Na mjestu uklonjene prekobrojne veze zadano opterećenje i zamijenjujuća sila daju pomak koji je jednak nuli. Koeficijent popustljivosti je pomak hvatišta sile po pravcu i u smjeru njena djelovanja uslijed njena djelovanja. Koeficijenti fleksibilnosti se mogu izračunati pomoći izraza koji su izvedeni metodom jedinične sile. Postavljene jednadžbe daju nepozante vrijednosti zamijenjujućih sila.

Matrična analiza sistemâ zglobnih štapova

Jednadžbe ravnoteže

Čvorove nekog rešetkastog sistema označiti ćemo uzastopnim brojevima od 0 do n. Prvo ćemo prebrojiti slobodne čvorove i označiti ih brojevima $0,1,\ldots,n_f$, a zatim ćemo prebrojiti sve ležajne čvorove i označiti ih brojevima $n_f+1,\ldots,n$. Štap između čvorova i i j označit ćemo sa $\{i,j\}$. Uvijek je $i \neq j$ te $\{i,j\}$ i $\{j,i\}$ označavaju isti štap. Ukupan broj štapova je b. Pogodno je štapove označiti i uzastopnim brojevima od 0 do b.

$\vec{e}_{i,j}$ i $\vec{e}_{j,i}$ jedinični vektori na osi štapa $\{i\,,j\}$, pri čemu je vektor $\vec{e}_{i,j}$ orijentiran od čvora i prema čvoru j, dok je vektor $\vec{e}_{j,i}$ orijentiran od čvora j prema čvoru i, tako da je $\vec{e}_{i,j} = -\vec{e}_{j,i}$. Vanjska je normala na ravninu poprečnog presjeka na kraju i orijentirana kao vektor $\vec{e}_{j,i}$, a na kraju j kao vektor $\vec{e}_{i,j}$.

Uzdužna sila na kraju ***i*** (sila kojom čvor ***i*** djeluje na štap) određena je vektorom

$$\vec{S}_{i,j} = S_{i,j}\,\vec{e}_{j,i}\;. \tag{1}$$

Vektor uzdužne sile na kraju *j* (sila kojom čvor j djeluje na štap) određena je vektorom

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

Ako je vrijednost skalara $S_{i,j} > 0$ sila u štapu je vlačna, a ako je vrijdnost skalara $S_{i,j} < 0$ sila je tlačna.

Vrijedi da je $\vec{S}_{j,i} = -\vec{S}_{i,j}$. Ako uvrstimo gornje izraze dobivamo

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

Što znači da je da je $S_{j,i} = S_{i,j}$ pa možemo zapisati kao $S_{\{i,j\}} = S_{j,i} = S_{i,j}$.

Ako su (x_i, y_i, z_i) i (x_j, y_j, z_j) koordinate čvorova *i* i *j*, vektori $\vec{e}_{i,j}$ i $\vec{e}_{j,i}$ dani su izrazima

$$\vec{e}_{i,j} = \frac{x_j - x_i}{l_{\{i,j\}}}\vec{\imath} + \frac{y_j - y_i}{l_{\{i,j\}}}\vec{\jmath} + \frac{z_j - z_i}{l_{\{i,j\}}}\vec{k} \tag{2}$$

$$\vec{e}_{j,i} = \frac{x_i - x_j}{l_{\{i,j\}}}\vec{\imath} + \frac{y_i - y_j}{l_{\{i,j\}}}\vec{\jmath} + \frac{z_i - z_j}{l_{\{i,j\}}}\vec{k} \tag{3}$$

gdje je

$$l_{\{i,j\}} = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2 + \left(z_j - z_i\right)^2} \tag{4}$$

duljina štapa $\{i,j\}$.

Rezultatnu vanjskih sila koje djeluju na slobodni čvor i označit ćemo sa $\vec{F}_i$; njezine su skalarne komponente $F_{i,x}$, $F_{i,y}$ i $F_{i,z}$. Štap {i,j} na čvor i djeluje silom

$$-\vec{S}_{i,j} = -S_{\{i,j\}}\vec{e}_{j,i} = S_{\{i,j\}}\vec{e}_{j,i} \tag{5}$$

Za svaki se slobodan čvor i=1,...,nf može napisati vektorska jednadžba ravnoteže sila koje na nj djeluju:

$$\sum_{j \in N_i}\left(-\vec{S}_{i,j}\right) + \vec{F}_i = \vec{0}$$

ili

$$\sum_{j \in N_i} S_{\{i,j\}} \vec{e}_{i,j} + \vec{F}_i = \vec{0} \qquad (6)$$

pri čemu je Ni skup čvorova koji su štapovima povezani sa čvorom i.

Tu jednadžbu možemo raspisati s obzirom na tri koordinatne osi :

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

$$\sum_{j \in N_i} \frac{y_j - y_i}{l_{\{i,j\}}} S_{\{i,j\}} + \vec{F}_{i,y} = 0, \qquad (7)$$

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

za $i = 1, \dots, n_f.$, odnosno za svaki slobodni čvor. Što znači da u svakom slobodnom čvoru možemo napisati $3n_f$ jedandžbe. Broj nepoznatih vrijednosti u štapovima je **b**.

Za analizu uvjeta rješivosti pogodno je sustav (7) zapisati u matričnom obliku:

As + f =0

ili

As = -f. (8)

Vrijednosti sila u štapovima poredane su pritom u vektor **s** prema brojčanim oznakama štapova: ako je **K** brojčana oznaka štapa *{i,j}* onda $S_{\{i,j\}}$ komponenta K vektora s, $S_K = S_{\{i,j\}}$.

Koeficijenti uz $S_{\{i,j\}} = S_K$ u jednadžbama ravnoteže čvora i komponente su matrice sustava **A** u sjecištima redaka $3(i-1)+1$, $3(i-1)+2$, $3(i-1)+3$ sa stupcem K:

$$a_{3(i-1)+1,K} = \frac{x_j - x_i}{l_{\{i,j\}}}, \quad a_{3(i-1)+2,K} = \frac{y_j - y_i}{l_{\{i,j\}}} \quad \text{i} \quad a_{3(i-1)+3,K} = \frac{z_j - z_i}{l_{\{i,j\}}} . \qquad (9)$$

Vrijednost $S_{\{i,j\}} = S_K$ ulazi i u jednadžbe ravnoteže čvora j, kojima odogvaraju redci $3(j-1)+1,\ 3(j-1)+2\ , 3(j-1)+3$ matrice **A**, pa su, u istom stupcu,

$$a_{3(j-1)+1,K}=\frac{x_i-x_j}{l_{\{i,j\}}}\ ,\quad a_{3(j-1)+2,K}=\frac{y_i-y_j}{l_{\{i,j\}}}\quad \text{i}\quad a_{3(j-1)+3,K}=\frac{z_i-z_j}{l_{\{i,j\}}}\ . \tag{10}$$

Matrica **A** ima 3nf 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 $3(i-1)+1,\ 3(i-1)+2\ , 3(i-1)+3$ skalarne su komponente $F_{i,x}\ , F_{i,y}\ , F_{i,z}$ vanjske sile $\vec{F_i}$ koja djeluje u čvoru i.

Matricu **A** nazivamo ***ravnotežnom*** ili ***statičkom matricom.***

Kinematičke jednadžbe

Kinematičke jednadžbe povezuju promjene duljina štapova i pomake čvorova. Pomake čvorova i i j označimo vektorima $\vec{u}_i$ i $\vec{u}_j$. Njihove će nove koordinate biti

$$\vec{u}_i = (x_i+u_i, y_i+v_i, z_i+w_i)$$
$$\vec{u}_j = (x_j+u_j, y_j+v_j, z_j+w_j).$$

Promjenu duljine štapa $\{i,j\}$ označit ćemo sa $d_{\{i,j\}}$. Ako je $d_{\{i,j\}} > 0$ riječ je o produljenju, a za $d_{\{i,j\}} < 0$ o skraćenju štapa. Nova je duljina štapa $l_{\{i,j\}} + d_{\{i,j\}}$. Primjenom pitagorina poučka dobijemo da je:

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

ili , možemo napisati:

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

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

$$l_{\{i,j\}}^2 + 2l_{\{i,j\}}d_{\{i,j\}} + d_{\{i,j\}}^2$$

$$= \left[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2\right]$$
$$+ 2\left[(x_j - x_i)(u_j - u_i) + (y_j - y_i)(v_j - v_i) + (z_j - z_i)(w_j - w_i)\right]$$
$$+ \left[(u_j - u_i)^2 + (v_j - v_i)^2 + (w_j - w_i)^2\right].$$

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 $\vec{u}_i$ i $\vec{u}_j$. Budući da su pomaci mali, $d_{\{i,j\}} < l_{\{i,j\}}$, $\|\vec{u}_i\| \ll l_{\{i,j\}}$ i $\|\vec{u}_j\| \ll l_{\{i,j\}}$, mogu se $d_{\{i,j\}}^2$ i $\|\vec{u}_i - \vec{u}_j\|^2$ zanemariti u odnosu na $2l_{\{i,j\}}d_{\{i,j\}}$ (drugi pribrojnik slijeva) i na drugi podizraz zdesna, pa ostaje

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

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

$$d_{\{i,j\}} = -\left(\frac{x_j - x_i}{l_{\{i,j\}}}u_i + \frac{y_j - y_i}{l_{\{i,j\}}}v_i + \frac{z_j - z_i}{l_{\{i,j\}}}w_i + \frac{x_i - x_j}{l_{\{i,j\}}}u_j + \frac{y_i - y_j}{l_{\{i,j\}}}v_j + \frac{z_i - z_j}{l_{\{i,j\}}}w_j\right). \tag{11}$$

Promjene duljine 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 , w_i na mjestima $3(i-1)+1,\ 3(i-1)+2$, $3(i-1)+3$. Odnosno, poredak promjene duljina štapva u vektor **d** odgovara poretku vrijednosti uzdužnih sila u vektoru **s**, a poredak skalarnih komponenata pomaka čvorova u vektoru u poretku skalarnih komponenata vanjskih sila u vektoru f. Sada izraze (11) za sve štapove možemo sažeto zapisati u obliku

$$\mathbf{d} = -\mathbf{Bu} \quad , \tag{12}$$

gdje je B matrica komponente koje su

$$b_{K,3(i-1)+1=}\frac{x_j - x_i}{l_{\{i,j\}}}, \quad b_{K,3(i-1)+2=}\frac{y_j - y_i}{l_{\{i,j\}}} \quad \text{i} \quad b_{K,3(i-1)+3=}\frac{z_j - z_i}{l_{\{i,j\}}} ,$$
$$b_{K,3(j-1)+1,K=}\frac{x_i - x_j}{l_{\{i,j\}}}, \quad b_{K,3(j-1)+2,K=}\frac{y_i - y_j}{l_{\{i,j\}}} \quad \text{i} \quad b_{K,3(j-1)+3,K=}\frac{z_i - z_j}{l_{\{i,j\}}} . \tag{13}$$

Vektori d i u sadrže b i 3nf komponenata, pa matrica B ima b redaka i 3nf stupaca.

Usporedba izraza (13) s izrazima (9) i (10) pokazuje da je

$$\mathbf{B} = \mathbf{A}^{\mathrm{T}}. \tag{14}$$

Matricu **B** nazivamo *kinematičkom matricom.*

Metoda sila

Kao što je već navedeno, ako je broj nepoznatih sila jednak broju jednadžbi ravnoteže $b = 3nf$ i ako je matrica **A** regularna (ako joj je rang $3nf$) sustav jednadžbi ravnoteže (8) ima jedinstveno rješenje za bilo koji vektor **f;**

$$\mathbf{s} = -\mathbf{A}^{-1}\ . \tag{15}$$

Sistem je tada geometrijski nepormijenjiv i statički određen. Nepoznate sile u štapovima se mogu odrediti iz jednadžbi ravnoteže.

Ako je pak broj vrijednosti sila veći od broja jednadžbi, b>3nf, i ako je rang ravnotežne matrice *3nf*, sistem je štapova geometrijski nepormijenjiv, ali statički neodređen.
Budući da je rang matrice jednak broju redaka, slika je matrice A cijeli prostor čvorova. Za bazu prostora čvorova možemo uzeti tri vektora za svaki slobodni čvor koji su međusobno okomiti. Dakle, ukupno *3nf* vektora. Vektori jednog čvora su okomiti na vektore drugih čvorova. U jednadžbama ravnoteže te vektore ćemo interpretirati kao vektorske komponente sila u čvorovima, dok ćemo ih u kinematičkim jednadžbama interpretirati kao komponente pomaka čvorova.
Sustav jednadžbi ranvoteže rješiv za sve vektore koji leže u tom prostoru – dakle, za sve vektore f. Ali, kako je broj vrijednosti sila u štapovima veći od broja jednadžbi, matrica A prostor veće dimenzije preslikava u prostor manje dimenzije. Pri tom preslikavanju neki vektori prostora štapova iščezavaju, odnosno više se vektora tog protora preslikava u jedan vektor prostora čvorova.

Ako je $\mathbf{s}_0$ neki vektor prostora redaka matrice **A** i ako su $\bar{\mathbf{s}}_i$ bazni vektori njezine jezgre, tada se svi vektori

$$\mathbf{s} = \mathbf{s}_0 + \textstyle\sum_i x_{\boldsymbol{i}} \bar{\mathbf{s}}_i \tag{16}$$

s bilo kojim koeficijentima $x_{\boldsymbol{i}}$ preslikavaju u isti vektor prostora čvorova. Rješenje sustava jednadžbi stoga ne može biti jedinstveno. Za izdvajanje jednog riješenja treba pored uvijeta ravnoteže primjeniti i kinematičke uvjete. Povezvivanje kinemaatičkih i ravnotežnih uvjeta ostavaruje se uvođenjem uvijeta koje za zglobne štapove izražavaju međuovistnost vrijednosti uzdužnih sila u njima i promjena njihovih duljina.

Djeluje li u štapu uzdužna sila vrijednosti $S_{\{i,j\}}$ **,** promjena je njegove duljine

$$d_{\{i,j\}} = \delta_{\{i,j\}} S_{\{i,j\}} \tag{17}$$

pri čemu je

$$\delta_{\{i,j\}} = \frac{l_{\{i,j\}}}{E_{\{i,j\}} A_{\{i,j\}}} \tag{18}$$

koeficijent (uzdužne) popustljivosti. I obratno, za promjenu duljine štapa za $d_{\{i,j\}}$ treba u nj unijeti uzdužnu silu vrijednosti

$$S_{\{i,j\}} = k_{\{i,j\}} d_{\{i,j\}}\,, \tag{19}$$

pri čemu je

$$k_{\{i,j\}} = \frac{E_{\{i,j\}} A_{\{i,j\}}}{l_{\{i,j\}}} \tag{20}$$

koeficijent (uzdužne) krutosti.

Izraz (16) zapisat ćemo u obliku

$$\mathbf{s} = \mathbf{s}_0 + \bar{\mathbf{S}}\mathbf{x}, \tag{21}$$

gdje je $\bar{\mathbf{S}}$ matrica stupaca kod koje su bazni vektori $\bar{\mathbf{s}}_i$ jezgre matrice **A**, a **x** je vektor koji sadrži koeficijente x_i. Uzet ćemo uz to da je $\mathbf{s}_0$ neko rješenje sustava jednadžbi ravnoteže (8), tako da je $\mathbf{A}\mathbf{s}_\mathbf{0} = -\mathbf{f}$ **.** Tada je i s rješenje tog sustava:

$$\mathbf{A}\mathbf{s} = \mathbf{A}(\mathbf{s}_0 + \bar{\mathbf{S}}\mathbf{x}) = \mathbf{A}\mathbf{s}_0 + \mathbf{A}\bar{\mathbf{S}}\mathbf{x} = \mathbf{A}\mathbf{s}_0 = -\mathbf{f}$$

treća jednakost slijedi iz činjenice da vektori $\bar{\mathbf{S}}\mathbf{x}$ leže u matrici **A** , pa je $\mathbf{A}\bar{\mathbf{S}}\mathbf{x} = \mathbf{0}$ **.**

Da nađemo vektor $\mathbf{s}_0$ i matricu $\bar{\mathbf{S}}$ Gaussovim ćemo eliminacijskim postupkom matricu **A** i vektor **f** provesti u $\widetilde{\mathbf{A}}$ i $\tilde{\mathbf{f}}$**.** Uporišni stupci matrice $\widetilde{\mathbf{A}}$ odgovaraju štapovima sistema čiji sklop daje statički određeni sistem; ostali su štapovi „prekobrojni“. Uzmemo li da su vrijednosti sila u prekobrojnim štapovima jednake nuli, vrijednosti sila u štapovima određenog sistema (i, time, vektor $\mathbf{s}_0$) možemo izračunati uvrštavanjem unazad u sustav $\widetilde{\mathbf{A}}\varsigma_0 = -\tilde{\mathbf{f}}$ **,** gdje je ς_0 vektor koji na mjestima štapova određenog sistema sadrži nepoznanice, a na mjestima prekobrojinih štapova nule.

Vektore $\bar{\mathbf{s}}_i$ odredit ćemo uvrštavanjem unazad u sustave $\widetilde{\mathbf{A}}\varsigma_0 = 0$, pri čemu su ς_i vektori koji sadrže nepoznanice na mjestima štapova statički određenog sistema, 1 na mjestu prekobrojnog štapa $\boldsymbol{i}$ i *0* na mjestima ostalih prekobrojnih štapova.
Fizički prihvatljivo rješenje mora pored uvjeta ravnoteže zadovoljiti i određene kinmatičke uvjete: vrijednosti sila u štapovima moraju uzrokogvati komatibilne

promjene njihovih duljina – sistem štapova se mora moći sklopiti bez dodatnih promjena duljina. Uzajamna ovisnost vrijednosti uzdužne sile i promjene duljine štapa dana je konstitucijskom relacijom (17).

Koeficijente popustljivosti $\delta_{\{i,j\}}$ definirane izrazom (18) smjestit ćemo u dijagonalnu matricu $\mathrm{diag}(\delta)$ na isti način kao što smo koeficijente krutosti $k_{\{i,j\}}$ smjestili u matricu $\mathrm{diag}(\mathrm{k})$: ako je K brojčana oznaka štapa $\{i,j\}$, koeficijent $\delta_{\{i,j\}}$ bit će dijagonalna komponenta $\delta_{K,K}$ matrice $\mathrm{diag}(\delta)$. Iz $\delta_{\{i,j\}} = \frac{1}{k_{\{i,j\}}} = k_{\{i,j\}}^{-1}$ neposredno $\mathrm{diag}(\delta) = [\mathrm{diag}(\mathrm{k})]^{-1}$.

Sažeti je matrični zapis izraza (17) za sve štapove

$$\mathrm{d} = \mathrm{diag}(\delta)\mathrm{s} \quad , \tag{22}$$

a uvrštavanje izraza (21) daje

$$\mathrm{d} = \mathrm{diag}(\delta)(\mathrm{s}_0 + \bar{\mathrm{S}}\mathrm{x}). \tag{23}$$

Traženi je vektor kompatibilnih promjena duljine okomit na prostor nekompatibilnih promjena duljina, koji je lijeva jezgra kinematičke matrice **B** . Budući da se lijeva jezgra matrice **B** poduduara sa jezgrom matrice **A**, uvijet je okomitosti

$$\bar{\mathbf{S}}^{\mathbf{T}}\mathbf{d} = \mathbf{0}. \tag{24}$$

Uz (23) taj uvjet

$$\bar{\mathbf{S}}^{\mathbf{T}}\mathbf{diag}(\boldsymbol{\delta})(\mathbf{s_0} + \bar{\mathbf{S}}\mathbf{x}) = \mathbf{0}$$

ili

$$\bar{\mathbf{S}}^{\mathbf{T}}\mathbf{diag}(\boldsymbol{\delta})\bar{\mathbf{S}}\mathbf{x} = -\bar{\mathbf{S}}^{\mathbf{T}}\mathbf{diag}(\boldsymbol{\delta})\mathbf{s_0}. \tag{25}$$

Matrica

$$\mathbf{D} = \bar{\mathbf{S}}^{\mathbf{T}}\mathbf{diag}(\boldsymbol{\delta})\bar{\mathbf{S}} \tag{26}$$

je matrica popustljivosti sistema, dok je vektor

$$\mathbf{d_0} = \bar{\mathbf{S}}^{\mathbf{T}}\mathbf{diag}(\boldsymbol{\delta})\mathbf{s_0} \tag{27}$$

vektor promjena duljina štapova izazvanih ravnotežnim silama u osnovnom sistemu, pa je (25) *sustav jednadžbi kompatibilnosti*

$$\mathbf{Dx} = -\mathbf{d_0} \tag{28}$$

poznat iz metode sila.

4.Računalni program

Program za izračunavanje sila u štapovima statički neodređenih rešetkastih sistema izveden je u programskom paketu *Sage*.

Sustav se zadaje popisom čvorova - **joints**, popisom elemenata (štapova) – **bars**, popisom ležajnih čvorova - **supports** i opterećenjima - **loads**. Označavanje je najbolje uvijek započeti nulom.

Za prikaz zadanog sistema primjenjuje se naredba **plot_truss2d** ili **plot_truss3d,** ovisno o tome je li sustav ravninski ili prostorni.

```
def plot_truss2d (joints, bars, supports, loads = [], load_scale =
0.01) :
    prr = point2d (joints.values(), aspect_ratio = 1)
    rrx = (prr.xmax() - prr.xmin())
    rry = (prr.ymax() - prr.ymin())
    rr = rrx if rrx > rry else rry*3/2
    pfjs = plot_free_joints2d (list_indexed (others (supports,
joints), joints), rr/120)
    psjs = plot_supports2d (list_indexed (supports, joints), rr/150)
    pbs = plot_bars2d (bars.values(), joints)
    pt = pfjs + psjs + pbs
    if loads != [] :
        pls = plot_loads2d (loads, joints, load_scale)
        pt += pls
    return pt
```

```
def plot_truss3d (joints, bars, supports, loads = [], load_scale =
0.01, pull = True) :
    pfjs = plot_free_joints3d (list_indexed (others (supports,
joints), joints))
    psjs = plot_supports3d (list_indexed (supports, joints))
    pbs = plot_bars3d (bars.values(), joints)
    pt = pfjs + psjs + pbs
    if loads != [] :
        pls = plot_loads3d (loads, joints, load_scale, pull)
        pt += pls
    return pt
```

Naredbom **free_joints** na zadanom sustavu se traže slobodni čvorovi te se ispisuje se lista s oznakama slobodnih čvorova.

Naredbom **maxwells_rule** pomoću Maxwellovog pravila određuje se nužan uvijet za statičku i kinematičku određenost sustava.

```
def maxwells_rule (joints, supports, bars) :
    dim = len (joints.values()[0])
    free_joints = others (supports, joints)
    mxw = dim*len (free_joints) - len (bars)
    print dim, '*', len (free_joints), '-', len (bars), ' == ', mxw,
    if mxw != 0 :
        print ' !=  0'
    else :
        print
    return
```

U fukciji **auqmented_equil_matrix** su spojene funkcije **equilibrium_matrix** i **load_vector** kako bi se napravila proširena ravnotežna matrica koja sadrži ravnotežnu matricu i dodatni stupac koji predstavlja vektor sila u čvorovima.

```
def augmented_equil_matrix (nodes, elems, loads, free_nodes, field = 
RDF) :
    nfj = len (free_nodes)
    nb = len (elems)
    ne_inc = dict_joint_bars (free_nodes, elems)
    ukws = dict_unknown_index (elems)
    eqs = dict_equation_index (free_nodes)
    dim = len (nodes.values()[0])
    A = zero_matrix (field, dim*nfj, nb + 1)

    for ni in free_nodes :
        ei = eqs[ni]
        bni = ne_inc[ni]
        for bi in bni :
            if ni == elems[bi][0] :
                nj = elems[bi][1]
            else :
                nj = elems[bi][0]
            xxi = map (field, nodes[ni])
            xxj = map (field, nodes[nj])
            l = sqrt (sum (map (lambda xi, xj : (xj - xi)^2, xxi, 
xxj)))
            cc = map (lambda xi, xj : (xj - xi) / l, xxi, xxj)
            ui = ukws[bi]
            A[ei : (ei+dim), ui] += matrix (dim, 1, cc)
    for li, ff in loads.iteritems() :
        ei = eqs[li]
        ej = nb
        A[ei : (ei+dim), ej] = matrix (dim, 1, map (lambda x : -
field(x), ff))

    A.subdivide ([], [nb])
    return A
```

Funkcija **equilibrium_matrix** radi ravnotežnu matricu. Ima četiri parametara, a to su lista čvorova, lista elemnata, lista slobodnih čvorova i polje (način prikaza) realnih brojeva. Funkcija len daje brojeve slobodnih čvorova i elemenata.
Nakon toga se poziva funkcija **dict_joint_bars** koja radi rječnik ne_inc koji govori koji su sve elementi priključeni u određeni slobodni čvor.
Funkcija **dict_unknown_index** u rječniku ukws elemente označava brojeći od nule bez obzira kako ih mi u početku u popisu elemenata zadali.
Funkcija **dict_equation_index** radi rječnik eqs koji govori koja je prva jednadžba ravnoteže od tri jednadžbe ravnoteže za svaki čvor.
Funkcija **zero_matrix** radi matricu koja je ispunjena nulama. Funkcija ima tri argumenta: polje realnih brojeva, broj redaka koji je jednak trostrukom broju čvorova i broj stupaca koji odgovara broju štapova.Petlja prolazi po slobodnim čvorovima. Prvo iz rječnika eqs izdvaja indeks prve jednadžbe za čvor u kojem se nalazimo. Zatim se za taj čvor iz rječnika ne_inc izdvaja lista štapova koji ulazi u njega. Petlja u petlji prolazi po svakom priključenom elementu i određuje nam koji je njegov drugi čvor. Nakon toga se određuju nagibi elemenata preko koordinata koje smo zadali u popisu čvorova.SR označava simbolički tip podataka. Ako se radi sa simboličkim varijablama onda se duljina elemenata ne računa nego se napiše da je duljina simbolička varijabla.
U slučaju da se radi s brojevima, računaju se stvarna duljina elementa po formuli za duljinu elemenata i kosinusi kutova koje štapovi zatvaraju s koordinatnim osima kao razlike koordinata podijeljene duljinom elementa. Nakon toga se u matricu upisuju kosinusi tako da ih upisujemo u stupac koji je jednak indeksu elementa za koji petlja računa duljinu i kosinuse, odnosno štapu na kojem jesmo, te počevši od prvog retka kojeg smo dobili na početku funkcijom koja nam pokazuje koja je prva jednadžba ravnoteže za taj čvor u kojem jesmo.

```
def equilibrium_matrix (nodes, elems, free_nodes, field = RDF) :
    nfj = len (free_nodes)
    nb = len (elems)
    ne_inc = dict_joint_bars (free_nodes, elems)
    ukws = dict_unknown_index (elems)
    eqs = dict_equation_index (free_nodes)
    dim = len (nodes.values()[0])
    A = zero_matrix (field, dim*nfj, nb)

    for ni in free_nodes :
        ei = eqs[ni]
        bni = ne_inc[ni]
        for bi in bni :
            if ni == elems[bi][0] :
                nj = elems[bi][1]
            else :
                nj = elems[bi][0]
            xxi = map (field, nodes[ni])
            xxj = map (field, nodes[nj])
            if field == SR :
                if ni < nj :
                    l = var ('l_%d%d' % (ni, nj))
                else :
                    l = var ('l_%d%d' % (nj, ni))
            else :
                l = sqrt (sum (map (lambda xi, xj : (xj - xi)^2, xxi,
xxj)))
            cc = map (lambda xi, xj : (xj - xi) / l, xxi, xxj)
            ui = ukws[bi]
            A[ei : (ei+dim), ui] += matrix (dim, 1, cc)

    return A
```

Funkcija **load_vector** radi vektor sila, odnosno ono što nam je u jednadžbama ravnoteže na desnoj strani. Parametri su opterećenja koja su zadana po čvorovima, popis slobodnih čvorova i polje realnih brojeva.
Prvo se funkcijom **len** dobivaju broj komponenata opterećenja i broj slobodnih čvorova. Pozivom funkcije **dict_equation_index** dobiva se rječnik eqs koji nam govori koja je prva jednadžba ravnoteže od tri jednadžbe ravnoteže za svaki čvor. Funkcija **zero_vector** ima dva argumenta. Prvi je polje realnih brojeva, a drugi je broj komponenata opterećenja pomnožen s brojem slobodnih čvorova. Ta funkcija prvo vektor popunjava nulama. Petlja prolazi po zadanim opterećenjima koja se zadaju komponentama usporednima s koordinatnim osima.Iz rječnika eqs izdvaja se indeks prve jednadžbe za čvor u kojem se nalazimo.Potom se za sve tri jednadžbe tog čvora, počevši od prve, upisuje u vektor opterećanja po redu trojka brojeva koje su komponente zadanih opterećenja.

```
def load_vector (loads, free_nodes, field = RDF) :
    dim = len (loads.values()[0])
    nfj = len (free_nodes)
    eqs = dict_equation_index (free_nodes)
    b = zero_vector (field, dim*nfj)
    for li, ff in loads.iteritems() :
        ei = eqs[li]
        b[ei : (ei+dim)] = map (field, ff)
    return b
```

Kada smo napravili proširenu ravnotežnu matricu naredbom **subdivide** dijelimo matricu na dva dijela tako da se ne prikaže dodatni stupac koji predstavlja vektore sila u čvorovima nego samo ravnotežna matrica koeficijenata.

Nakon što smo dobili ravnotežnu matricu, prevodimo ju u stepeničasti ili u reduirani stepeničasti oblik kako bismo saznali koji su uporišni stupci. U stepeničastom obliku matrica ima ispod uporišnih komponenata nule, dok su u reduciranom stepeničastom obliku nule i ispod i iznad uporišnih komponenata. To radimo funkcijama **ref_wpp** i **rref_wpp.** Parametri tih funkcija su matrica koju želimo preoblikovati, funkcija kojom provjeravamo je li neka vrijednost nula te parametar in_place kojim određujemo hoće li preoblikovana matrica biti nova ili će biti upisana preko polazne.

Pivot označava uporišni stupac u kojem se nalazimo, a kreće se od prvog.

Ako je in_place == true, preoblikovana se matrica piše preko zadane. To je bitno kod velikih sustava kako bi se sačuvala memorija računala.

pivotlist je lista indeksa uporišnih stupaca, u početku prazna. Nakon toga petlja prolazi po stupcima matrice. Na početku se provjerava je li indeks retka s trenutnim uporišnim elementom jednak broju redaka; ako jest, petlja se prekida.

U petlji, koja prolazi po retcima u stupcu u kojem se nalazimo, počevši od prvog retka iza retka s trenutnim uporišnim elementom, traži se redak s najvećom vrijednošću Ako je nađen redak s većom vrijednošću od vrijednosti u retku s trenutnim uporišnim elementom, onda zamjenjujemo retke. Ako ta najveća vrijednost nije nula, onda je trenutni stupac novi uporišni stupac, te se provodi jedan korak Gaussove eliminacije (u **ref_wpp**) ili Gauss-Jordanove eliminacije (u **rref_wpp**). Na kraju se u listu pivotlist upisuje indeks stupca u kojem jesmo, a broj se uporišnih elemenata povećava za 1.

```
def ref_wpp (m, is_zero_f = is_zz, in_place = False) :
    pivot = 0
    if in_place == True :
        r = m
    else :
        r = copy (m)
    pivotlist = []
    for i in xrange (r.ncols()) :
        if pivot == r.nrows() :
            break
        maxrow = pivot
        for k in xrange (pivot+1, r.nrows()):
            if abs (r[k][i]) > abs (r[maxrow][i]) :
                maxrow = k
        if pivot != maxrow :
            r.swap_rows (pivot, maxrow)
        if not is_zero_f (r[pivot, i]) :
            scale = r[pivot, i]
            r.rescale_row (pivot, 1/scale, i)
            for j in xrange (pivot+1, r.nrows()) :
                scale = r[j, i]
                r.add_multiple_of_row (j, pivot, -scale, i)
            pivotlist.append (i)
            pivot += 1
    return r, pivotlist
```

```
def rref_wpp (m, is_zero_f = is_zz, in_place = False) :
    pivot = 0
    if in_place == True :
        r = m
    else :
        r = copy (m)
    pivotlist = []
    for i in xrange (r.ncols()) :
        if pivot == r.nrows() :
            break
        maxrow = pivot
        for k in xrange (pivot+1, r.nrows()):
            if abs (r[k][i]) > abs (r[maxrow][i]) :
                maxrow = k
        if pivot != maxrow :
            r.swap_rows (pivot, maxrow)
        if not is_zero_f (r[pivot, i]) :
            scale = r[pivot, i]
            r.rescale_row (pivot, 1/scale, i)
            for j in xrange (r.nrows()) :
                if j == pivot :
                    continue
                scale = r[j, i]
                r.add_multiple_of_row (j, pivot, -scale, i)
            pivotlist.append (i)
            pivot += 1
    return r, pivotlist
```

Uporišni stupci odgovaraju štapovima statički određenog sistema, odnosno osnovnog sistema postupka metode sila. Stupci koji nisu uporišni odgovaraju prekobrojnim štapovima. Naredbom **pivots** ispisujemo uporišne stupce. Naredba **len (pivots**) ih broji i ispisuje ukupan broj uporišnih stupaca odnosno štapova zadanog osnovnog sistema.

U postupku metode sila mora se definirati osnovni sistem koji je statički određen sistem. Pomoću uporišnih stupaca se odabire osnovni sistem. **primary_system_bars** ispisuje listu štapova koji pripadaju osnovnom sistemu te listu čvorova koji pripadaju tim štapovima. Program bira osnovni sistem s obzirom kojim su redoslijedom zadani elementi.
redundant_bars ispisuje listu prekobrojnih štapova i pripadajuće čvorove tih štapova.

Zatim se računaju sile u štapovima osnovnog sistema pozivom fukcije **ref_solution.** Tom funkcijom se proširena ravnotežna matrica koja je u stepeničastom obliku rješava „uvrštavanjem unazad“ u njezin posljenji stupac. Ako je proširena ravnotežna matrica u reduciranom stepeničastom obliku upotrebljava se funkcija **rref_solution**. Ispisuju se vrijednosti sila u štapovima statički određenog osnovnog sistema.

```
def ref_solution (Ar, piv) :
    r = Ar.column (-1)
    nu = Ar.ncols() - 1
    if nu < Ar.nrows() :
        r = r[0:nu]
    np = len (piv)
    for j in xrange (np-1, -1, -1) :
        for jj in xrange (np-1, j, -1) :
            r[j] -= Ar[j, piv[jj]] * r[jj]
    return r
```

```
def rref_solution (Arr) :
    r = Arr.column (-1)
    nu = Arr.ncols() - 1
    if nu < Arr.nrows() :
        r = r[0:nu]
    return r
```

Kada smo dobili vrijednosti sila u svim štapovima zadanog sistema naredbom **all_bar_forces** pridružujemo te izračunate vrijednosti štapovima osnovnog sistema, a prekobrojnim štapovima pridružujemo nule. Dakle ta fukcija za statički neodređen sistem daje vektor sila u svim štapovima. Naime, kada se napravi proširena matrica i stepeničasti oblik ta matrica ima manje jednadžbi nego štapova.

```
def all_bar_forces (primary_bars, redundant_bars, primary_forces) :
    bfs = vector (primary_forces.base_ring(), len (primary_bars) + len
(redundant_bars))
    for i in xrange (len (primary_bars)) :
        bfs[primary_bars[i]] = primary_forces[i]
    return bfs
```

Funkcijom **dict_bar_forces** sređuje se prikaz izračunatih sila u štapovima. Ta funkcija omogućava ispis sila u štapovima kad se riješi sustav jednadžbi. Ona radi rječnik koji pridružuje vrijednosti rješenja dobivenih rješavanjem jednadžbi oznakama elementa kako su zadane prilikom zadavanja sustava.

```
def dict_bar_force (elems, forces) :
    ukws = dict_label_index (elems.keys())
    return dict ([(b, forces[i]) for b, i in ukws.iteritems()])
```

Funkcijama **plot_truss_with_forces2d** ili **plot_truss_with_forces3d** radimo prikaz zadanog sustava kojemu su pridružene izračunate sile u štapovima osnovnog sistema.

```
def plot_truss_with_forces2d (joints, bars, supports, forces) :
    prr = point2d (joints.values(), aspect_ratio = 1)
    rrx = (prr.xmax() - prr.xmin())
    rry = (prr.ymax() - prr.ymin())
    rr = rrx if rrx > rry else rry*3/2
    pfjs = plot_free_joints2d (list_indexed (others (supports,
joints), joints), rr/120)
    psjs = plot_supports2d (list_indexed (supports, joints), rr/150)
    pt = pfjs + psjs
    tt, cc, zi = tension_compression (forces)
    ti = unique_lp (tt)
    ti.reverse()
    ci = unique_lp (cc)
    if zi != [] :
        cl = Color ('orange')
        cl = cl.lighter (0.425)
        pbz = plot_bars2d (list_indexed (zi, bars), joints, color =
cl, thickness = 3)
        pt += pbz
    if ti != [] :
        step = 1./len (ti)
        cl = Color ('darkblue')
        pbt = plot_bars2d (list_indexed (ti[0], bars), joints, color =
cl, thickness = 3)
        for i in xrange (1, len (ti)) :
            cl = cl.lighter (step)
            pbt += plot_bars2d (list_indexed (ti[i], bars), joints,
color = cl, thickness = 3)
        pt += pbt
    if ci != [] :
        step = 1./len (ci)
        cl = Color ('red')
        pbt = plot_bars2d (list_indexed (ci[0], bars), joints, color =
cl, thickness = 3)
        for i in xrange (1, len (ci)) :
            cl = cl.lighter (step)
            pbt += plot_bars2d (list_indexed (ci[i], bars), joints,
color = cl, thickness = 3)
        pt += pbt
    return pt
```

```
def plot_truss_with_forces3d (joints, bars, supports, forces) :
    pfjs = plot_free_joints3d (list_indexed (others (supports,
joints), joints))
    psjs = plot_supports3d (list_indexed (supports, joints))
    pt = pfjs + psjs
    tt, cc, zi = tension_compression (forces)
    ti = unique_lp (tt)
    ti.reverse()
    ci = unique_lp (cc)
    if zi != [] :
        cl = Color ('orange')
        cl = cl.lighter (0.625)
        pbz = plot_bars3d (list_indexed (zi, bars), joints, color =
cl, thickness = 7)
        pt += pbz
    if ti != [] :
        step = 1./len (ti)
        cl = Color ('indigo')
        pbt = plot_bars3d (list_indexed (ti[0], bars), joints, color =
cl, thickness = 7)
        for i in xrange (1, len (ti)) :
            cl = cl.lighter (step)
            pbt += plot_bars3d (list_indexed (ti[i], bars), joints,
color = cl, thickness = 7)
        pt += pbt
    if ci != [] :
        step = 1./len (ci)
        cl = Color ('red')
        pbt = plot_bars3d (list_indexed (ci[0], bars), joints, color =
cl, thickness = 7)
        for i in xrange (1, len (ci)) :
            cl = cl.lighter (step)
            pbt += plot_bars3d (list_indexed (ci[i], bars), joints,
color = cl, thickness = 7)
        pt += pbt
    return pt
```

Sada treba pronaći matricu $\bar{\mathbf{S}}$ koja je matrica stupci koje su bazni vektori jezgre ravnotežne matrice. Vektori jezgre dobivaju se rješanjem niza sustava jednadžbi koji se sastoje od matrice u stepeničastom obliku, nepoznanica u štapovima statički određenog sistema, jedinice u jednom prekobrojnom štapu *i*, dok su u ostalim prekobrojnim štapovima nule, te nul-vektora na desnoj strani. Uvrštavanjem unazad dobiva se vektor $\mathbf{s}_i$ koji sadrži sile u štapovima statički određenog sistema za slučaj kada je jedinica postavljena u prekobrojnom štapu ***i***, dok su u ostalim prekobrojnim štapovima nule. Dakle matrica $\bar{\mathbf{S}}$ je matrica stupci koje su vektori $\mathbf{s}_i$. Vektori $\mathbf{s}_i$ sadrže vrijednosti sila u jediničnim stanjima metode sila.
Bazne vektore pronalazi funkcija **kernel.** Parametri fukncije **kernel** su matrica u (reduciranom) stepeničastom obliku, lista indeksa uporišnih stupaca i funkcija koja rješava sustav s jediničnom silom. Funkcija nalazi bazu jegre matrice kao komponete liste. U petlji koja prolazi po stupcima matrice za svaki se stupac koji nije uporišni (ako indeks stupca nije u listi indeksa uporišnih stupaca) rješava se homogeni sustav s jediničnom silom u štapu koji odgovara tom stupcu. Rješenje je sustava bazni vektor jezgre koji se dodaje u listu rješenja.

```
def kernel (mr, piv, backsubs = backsubs_) :
    k = []
    m = mr
    nc = m.ncols()
    np = len (piv)
    piv_s = set (piv)
    for i in xrange (nc) :
        if not i in piv_s :
            k.append (backsubs (m, nc, piv, np, i))
    return k
```

Naredba **matrix** listu baznih vektora jezgre koje je pronašla funkcija kernel pretvara u matricu SST u kojoj je svaki redak jedan vektor.
Matrica $\bar{\mathbf{S}}$ dobiva se transponiranjem **SS=SST.T**.

Zatim se ispisuju liste vrijednosti sila u štapovima za pojedinačna stanja. Funkcijom **dict_bar_force** ispisuju se ta rješenja tako da ih se pridruži oznakama štapova.

```
def dict_bar_force (elems, forces) :
    ukws = dict_label_index (elems.keys())
    return dict ([(b, forces[i]) for b, i in ukws.iteritems()])
```

Funkcijama ***plot_truss_with_forces2d*** ili ***plot_truss_with_forces3d*** radimo prikaz zadanog sustava kojemu su pridružene izračunate sile u štapovima.

Kako bi u svim štapovima zadali istu površinu poprečnog presjeka štapa i koristimo fukncjiu **dict_same_value** . Ta funkcija pojednostavljuje pridruživanje zadanih vrijednosti štapovima kako ne bismo za svaki štap posebno zadavali površinu i modul elastičnosti.

```
def dict_same_value (dct, vl) :
    dct2 = {}
    for d in dct :
        dct2[d] = vl
    return dct2
```

Za rješavanje sustava jednadžbi koje izražavaju uvjete kompatibilnosti treba je za sve štapove u sustavu izračunati koeficijente popustljivosti i za matrični zapis ih svrstati u dijagonalnu matricu. Uvrštavanjem u uvjet kompatibilnosti promjene duljina štapova dobiva se matrica popustljivosti sistema. Dijagonalna matrica fleksibilnosti zadaje se funkcijom **diag_flex.** Parametri te funkcije su lista elemenata zadanog sustava, lista čvorova, vrijednost modula elastičnosti koja je u svim štapovima jednaka, lista površine poprečnog presjeka štapova i polje realnih brojeva. Funkcija radi listu koeficijenata fleksibilnosti tako da petlja ide po svim elementima i izdvaja rječnik početnih i krajnjih čvorova elemenata. Pomoću tog rječnika se računa duljina štapova. Po formuli se računaju koeficjenti fleksibilnosti za svaki pojedini element. Funkcija

Sage-a **diagonal_matrix** radi matricu u kojoj su parametri realni brojevi i lista komponenata koja dolazi na dijagonalu.

```
def diag_flex (elems, nodes, E, Fs, field = RDF) :
    ldelta = []
    for el in elems :
        ni, nj = elems[el]
        xi = nodes[ni]
        xj = nodes[nj]
        ldelta.append (bar_length (xi, xj) / (E * Fs[el]))
    return diagonal_matrix (RDF, ldelta)
```

Računa se matrica **D** koja se dobiva umnoškom transponirane matrice **SST=**$\bar{\mathbf{S}}^{\mathbf{T}}$, dijagnalne matrice fleksibilnosti i matrice $\bar{\mathbf{S}}$.

Zatim računamo vektor $\mathbf{d}_0$ koji se dobiva umnoškom $\bar{\mathbf{S}}^{\mathbf{T}}$, dijagonalne matrice koeficijenata fleksibilnosti i vektora rješenja osnovnog sistema $\mathbf{s}_0$.

Rješava se sustav jednadžbi koji izražavaju uvjete kompatibilnosti. $\mathbf{D}\mathbf{x} = -\mathbf{d}_0$ iz kojeg se dobiva nepoznanica **x** koju uvrštavamo u formulu za sile u štapovima **s** te se dobiva lista sila u štapovima.

Funkcijom **dict_bar_force** prikazuju se ta rješenja tako da ih pribroji oznakama štapova.

```
def dict_bar_force (elems, forces) :
    ukws = dict_label_index (elems.keys())
    return dict ([(b, forces[i]) for b, i in ukws.iteritems()])
```

Funkcijama **plot_truss_with_forces2d** i **plot_truss_with_forces3d** radimo prikaz zadanog sustava kojemu su pridružene izračunate sile u štapovima. Na prikazu sustava se po bojama može vidjeti da li je sila u štapu tlačna ili vlačna i usporedba veličina vrijednosti sila u štapovima s obzirom na intenzitet boja.

5.Primjeri

Primjer 1. – ravninski sistem sa tri štapa

```
joints = { 0: (-3.0, 0.0), 1: (1.0, 0.0), 2: (4.0, 0.0), 3: (0.0, 5.0)
}
print 'joints:'
print_dict (joints, indent = '   ')
print

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

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

loads = { 3: (125.0, 25.0) }
print 'loads:'
print_dict (loads, indent = '   ')
```

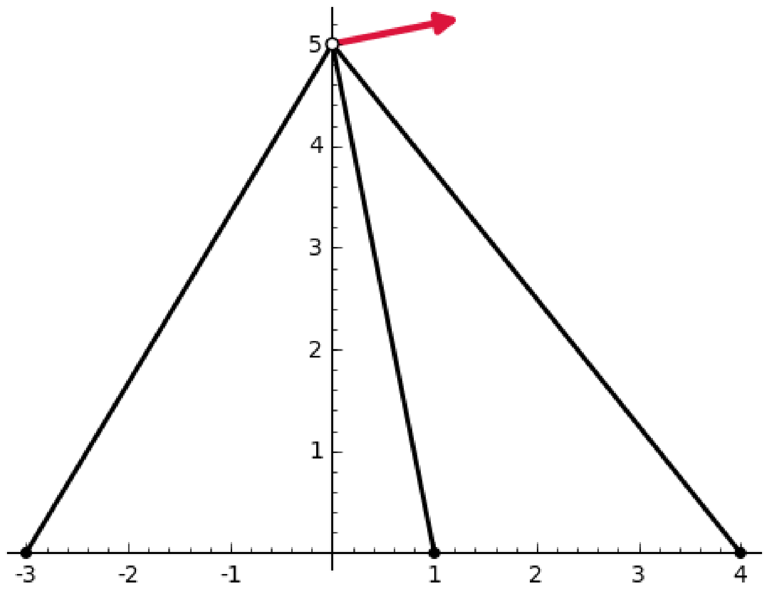
```
joints:
   0:   (-3.00000000000000, 0.000000000000000)
   1:   (1.00000000000000, 0.000000000000000)
   2:   (4.00000000000000, 0.000000000000000)
   3:   (0.000000000000000, 5.00000000000000)

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

supports: [0, 1, 2]

loads:
   3:   (125.000000000000, 25.0000000000000)
```

```
plot_truss2d (joints, bars, supports, loads) . show (figsize = 5)
```



```
free_joints = others (supports, joints)
free_joints
```

```
[3]
```

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

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

```
AAf = augmented_equil_matrix (joints, bars, loads, free_joints)
show (AAf)
```

$$\left(\begin{array}{rrr|r} -0.514495755428 & 0.196116135138 & 0.624695047554 & -125.0 \\ -0.857492925713 & -0.980580675691 & -0.780868809443 & -25.0 \end{array}\right)$$

```
AA = AAf.subdivision (0, 0)
show (AA)
```

$$\begin{pmatrix} -0.514495755428 & 0.196116135138 & 0.624695047554 \\ -0.857492925713 & -0.980580675691 & -0.780868809443 \end{pmatrix}$$

```
AAfr, pivots = ref_wpp (AAf)
show (AAfr)
```

$$\left(\begin{array}{ccc|c} 1.0 & 1.14354374979 & 0.910641692809 & 29.1547594742 \\ 0.0 & 1.0 & 1.39358285392 & -140.223036624 \end{array}\right)$$

```
AAr = AAfr.subdivision (0, 0)
show (AAr)
```

$$\begin{pmatrix} 1.0 & 1.14354374979 & 0.910641692809 \\ 0.0 & 1.0 & 1.39358285392 \end{pmatrix}$$

```
pivots
```

```
[0, 1]
```

```
len (pivots)
```

```
2
```

```
primary_system_bars = pivots
print primary_system_bars
print_list (list_indexed (primary_system_bars, bars), indent = '    ')
```

```
      [0, 1]
      (0, 3)
      (1, 3)
```

```
redundant_bars = others (primary_system_bars, all_indices (bars))
print redundant_bars
print_list (list_indexed (redundant_bars, bars), indent = '    ')
```

```
      [2]
      (2, 3)
```

```
prim_bar_forces = ref_solution (AAfr, primary_system_bars)
prim_bar_forces
```

```
(189.50593658247226, -140.22303662380156)
```

```
ss0 = all_bar_forces (primary_system_bars, redundant_bars,
prim_bar_forces)
ss0
```

```
(189.50593658247226, -140.22303662380156, 0.0)
```

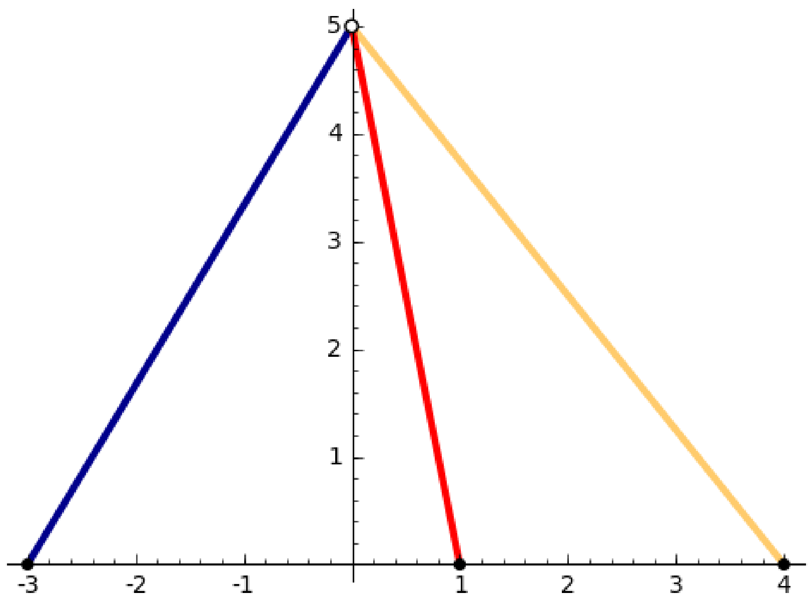
```
AA * ss0
```

```
(-124.99999999999999, -24.999999999999997)
```

```
dict_ss0 = dict_bar_force (bars, ss0)
print_dict (dict_ss0)
```

```
0:   189.505936582
1:   -140.223036624
2:   0.0
```

```
plot_truss_with_forces2d (joints, bars, supports, dict_ss0) . show
(figsize = 5)
```



```
AAk = kernel (AAr, pivots)
print_list (AAk)
```

```
(0.6829812696071147, -1.393582853917092, 1.0)
```

```
SST = matrix (AAk)
SS = SST.T
show (SS)
```

$$\begin{pmatrix} 0.682981269607 \\ -1.39358285392 \\ 1.0 \end{pmatrix}$$

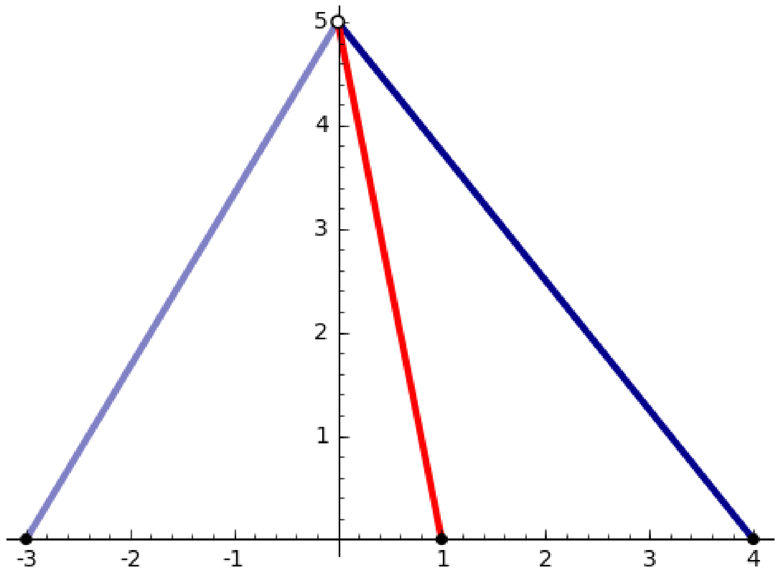
```
SS0 = SS.column(0)
SS0
```

```
(0.6829812696071147, -1.393582853917092, 1.0)
```

```
dict_SS0 = dict_bar_force (bars, SS0)
print_dict (dict_SS0)
```

```
0:   0.682981269607
1:   -1.39358285392
2:   1.0
```

```
plot_truss_with_forces2d (joints, bars, supports, dict_SS0) . show
(figsize = 5)
```



```
E = 2.e8
Fs = dict_same_value (bars, 0.05*0.05)
print E
print
print_dict (Fs)
```

```
2.00000000000000e8

0:   0.00250000000000000
1:   0.00250000000000000
2:   0.00250000000000000
```

```
ddelta = diag_flex (bars, joints, E, Fs)
show (ddelta)
```

$$\begin{pmatrix} 1.16619037897 \times 10^{-05} & 0.0 & 0.0 \\ 0.0 & 1.01980390272 \times 10^{-05} & 0.0 \\ 0.0 & 0.0 & 1.28062484749 \times 10^{-05} \end{pmatrix}$$

```
DD = SST * ddelta * SS
show (DD)
```

$$\left(3.80514379265 \times 10^{-05}\right)$$

```
dd0 = SST * ddelta * ss0
dd0
```

```
(0.003502212086933164)
```

```
xx = DD \ (-dd0)
xx
```

```
(-92.038889402733)
```

```
ss = ss0 + SS*xx
ss
```

```
(126.64509904496487, -11.95921845858129, -92.038889402733)
```

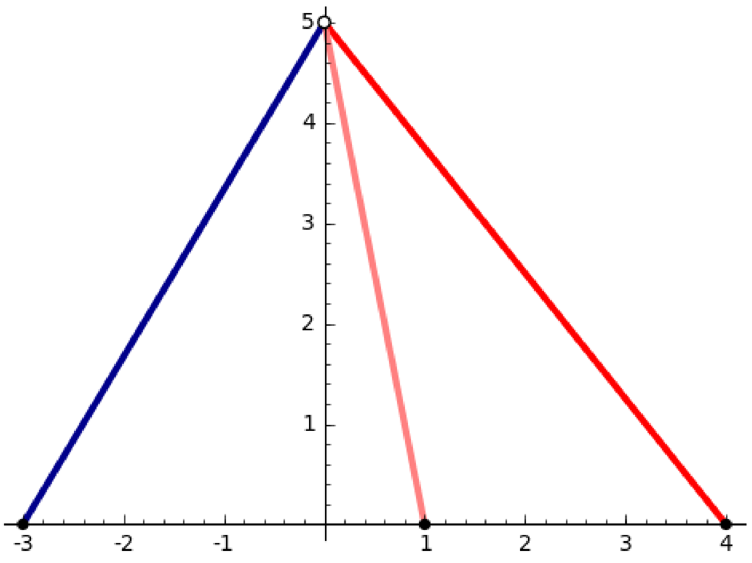
```
AA * ss
```

```
(-124.99999999999999, -25.000000000000014)
```

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

```
0:   126.645099045
1:   -11.9592184586
2:   -92.0388894027
```

```
plot_truss_with_forces2d (joints, bars, supports, dict_ss) . show
(figsize = 5)
```



Metoda sila — promjena redoslijeda štapova

Promjenom redoslijeda štapova program odabire druge štapove za osnovni sistem.

```
bars2 = { 0: (2, 3), 1: (1, 3), 2: (0, 3) }
print 'bars:'
print_dict (bars2, indent = '    ')
```

```
bars:
   0:    (2, 3)
   1:    (1, 3)
   2:    (0, 3)
```

```
A2Af = augmented_equil_matrix (joints, bars2, loads, free_joints)
show (A2Af)
```

$$\left(\begin{array}{rrr|r} 0.624695047554 & 0.196116135138 & -0.514495755428 & -125.0 \\ -0.780868809443 & -0.980580675691 & -0.857492925713 & -25.0 \end{array}\right)$$

```
A2A = A2Af.subdivision (0, 0)
show (A2A)
```

$$\left(\begin{array}{rrr} 0.624695047554 & 0.196116135138 & -0.514495755428 \\ -0.780868809443 & -0.980580675691 & -0.857492925713 \end{array}\right)$$

```
A2Afr, pivots2 = ref_wpp (A2Af)
show (A2Afr)
```

$$\left(\begin{array}{rrr|r} 1.0 & 1.25575597825 & 1.09812674721 & 32.0156211872 \\ 0.0 & 1.0 & 2.04044080846 & 246.452609824 \end{array}\right)$$

```
primary_system_bars2 = pivots2
print primary_system_bars2
print_list (list_indexed (primary_system_bars2, bars2), indent = '
')
print
#
redundant_bars2 = others (primary_system_bars2, all_indices (bars2))
print redundant_bars2
print_list (list_indexed (redundant_bars2, bars2), indent = '    ')
```

```
 [0, 1]
(2, 3)
(1, 3)

[2]
(0, 3)
```

```
prim_bar_forces2 = ref_solution (A2Afr, primary_system_bars2)
prim_bar_forces2
```

```
(-277.4687169554234, 246.45260982365124)
```

```
s2s0 = all_bar_forces (primary_system_bars2, redundant_bars2,
prim_bar_forces2)
s2s0
```

```
(-277.4687169554234, 246.45260982365124, 0.0)
```

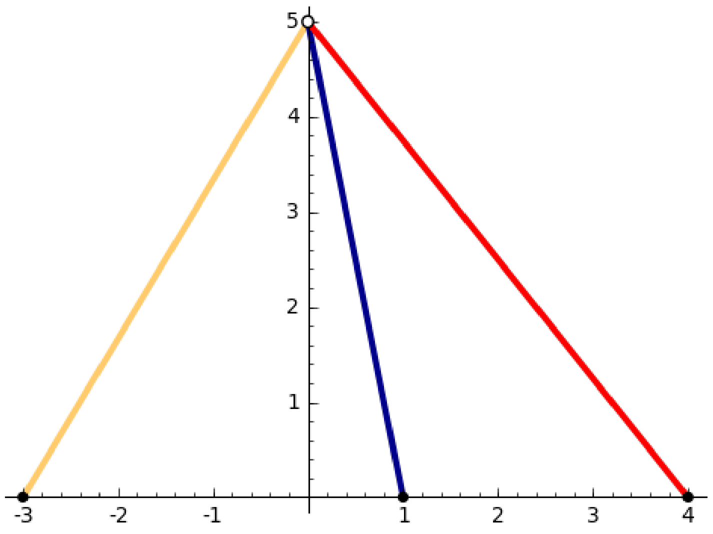
```
A2A * s2s0
```

```
(-124.99999999999999, -25.0)
```

```
dict_s2s0 = dict_bar_force (bars2, s2s0)
print_dict (dict_s2s0)
```

```
0:   -277.468716955
1:   246.452609824
2:   0.0
```

```
plot_truss_with_forces2d (joints, bars2, supports, dict_s2s0) . show
(figsize = 5)
```



```
A2Ar = A2Afr.subdivision (0, 0)
show (A2Ar)
```

$$\begin{pmatrix} 1.0 & 1.25575597825 & 1.09812674721 \\ 0.0 & 1.0 & 2.04044080846 \end{pmatrix}$$

```
S2ST = matrix (kernel (A2Ar, pivots2))
S2S = S2ST.T
show (S2S)
```

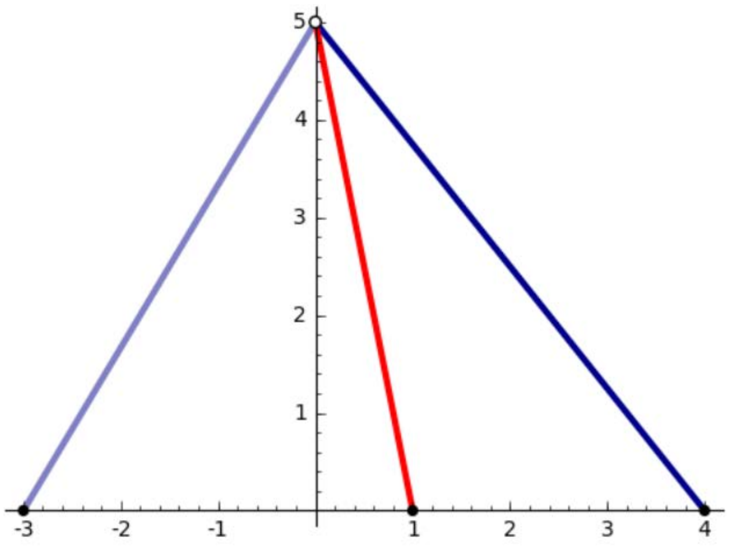
$$\begin{pmatrix} 1.46416899628 \\ -2.04044080846 \\ 1.0 \end{pmatrix}$$

```
S2S0 = S2S.column(0)
```

```
dict_S2S0 = dict_bar_force (bars2, S2S0)
print_dict (dict_S2S0)
```

```
0:   1.46416899628
1:   -2.04044080846
2:   1.0
```

```
plot_truss_with_forces2d (joints, bars2, supports, dict_S2S0) . show
(figsize = 5)
```



```
ddelta2 = diag_flex (bars2, joints, E, Fs)
show (ddelta2)
```

$$\begin{pmatrix} 1.28062484749 \times 10^{-05} & 0.0 & 0.0 \\ 0.0 & 1.01980390272 \times 10^{-05} & 0.0 \\ 0.0 & 0.0 & 1.16619037897 \times 10^{-05} \end{pmatrix}$$

```
D2D = S2ST * ddelta2 * S2S
show (D2D)
```

$$\begin{pmatrix} 8.15743244438 \times 10^{-05} \end{pmatrix}$$

```
d2d0 = S2ST * ddelta2 * s2s0
d2d0
```

```
(-0.010330988398706522)
```

```
x2x = D2D \ (-d2d0)
x2x
```

```
(126.6450990449649)
```

```
s2s = s2s0 + S2S*x2x
s2s
```

```
(-92.03888940273299, -11.959218458581347, 126.6450990449649)
```

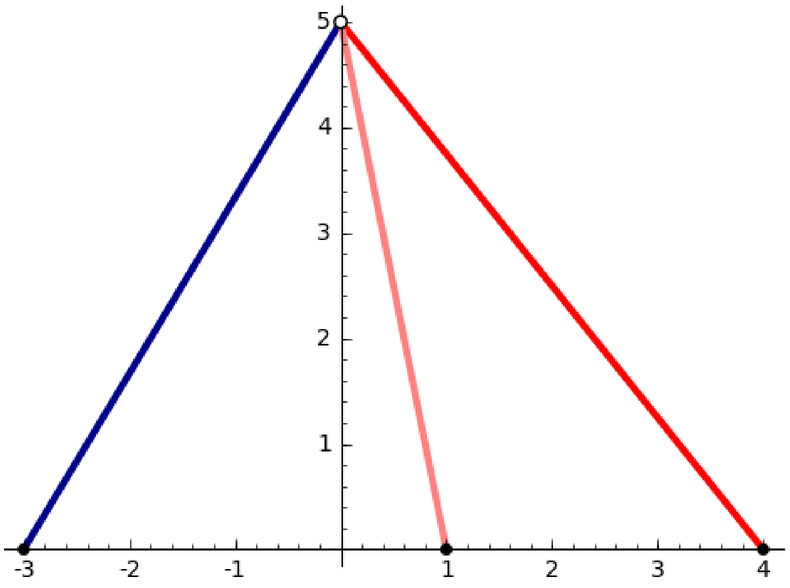
```
A2A * s2s
```

```
(-125.00000000000001, -25.0)
```

```
dict_s2s = dict_bar_force (bars2, s2s)
print_dict (dict_s2s)
```

```
0:   -92.0388894027
1:   -11.9592184586
2:   126.645099045
```

```
plot_truss_with_forces2d (joints, bars2, supports, dict_s2s) . show
(figsize = 5)
```



Metoda sila — još jedna promjena redoslijeda štapova

```
bars3 = { 0: (0, 3), 1: (2, 3), 2: (1, 3) }
print 'bars:'
print_dict (bars3, indent = '    ')
```

```
bars:
   0:   (0, 3)
   1:   (2, 3)
   2:   (1, 3)
```

```
A3Af = augmented_equil_matrix (joints, bars3, loads, free_joints)
show (A3Af)
```

$$\left(\begin{array}{rrr|r} -0.514495755428 & 0.624695047554 & 0.196116135138 & -125.0 \\ -0.857492925713 & -0.780868809443 & -0.980580675691 & -25.0 \end{array}\right)$$

```
A3A = A3Af.subdivision (0, 0)
show (A3A)
```

$$\left(\begin{array}{rrr} -0.514495755428 & 0.624695047554 & 0.196116135138 \\ -0.857492925713 & -0.780868809443 & -0.980580675691 \end{array}\right)$$

```
A3Afr, pivots3 = ref_wpp (A3Af)
show (A3Afr)
```

$$\left(\begin{array}{rrr|r} 1.0 & 0.910641692809 & 1.14354374979 & 29.1547594742 \\ 0.0 & 1.0 & 0.717574844717 & -100.620523731 \end{array}\right)$$

```
primary_system_bars3 = pivots3
print primary_system_bars3
print_list (list_indexed (primary_system_bars3, bars3), indent = '    ')
print
#
redundant_bars3 = others (primary_system_bars3, all_indices (bars3))
print redundant_bars3
print_list (list_indexed (redundant_bars3, bars3), indent = '    ')
```

```
[0, 1]
   (0, 3)
   (2, 3)

[2]
   (1, 3)
```

```
prim_bar_forces3 = ref_solution (A3Afr, primary_system_bars3)
prim_bar_forces3
```

```
(120.78400353608126, -100.62052373108763)
```

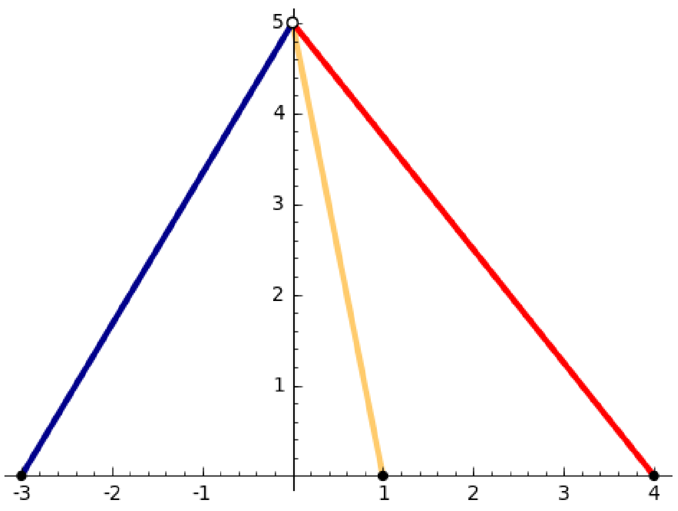
```
s3s0 = all_bar_forces (primary_system_bars3, redundant_bars3,
prim_bar_forces3)
dict_s3s0 = dict_bar_force (bars3, s3s0)
print_dict (dict_s3s0)
```

```
0:   120.784003536
1:   -100.620523731
2:   0.0
```

```
A3A * s3s0
```

```
(-125.00000000000001, -25.0)
```

```
plot_truss_with_forces2d (joints, bars3, supports, dict_s3s0) . show
(figsize = 5)
```



```
A3Ar = A3Afr.subdivision (0, 0)
show (A3Ar)
```

$$\begin{pmatrix} 1.0 & 0.910641692809 & 1.14354374979 \\ 0.0 & 1.0 & 0.717574844717 \end{pmatrix}$$

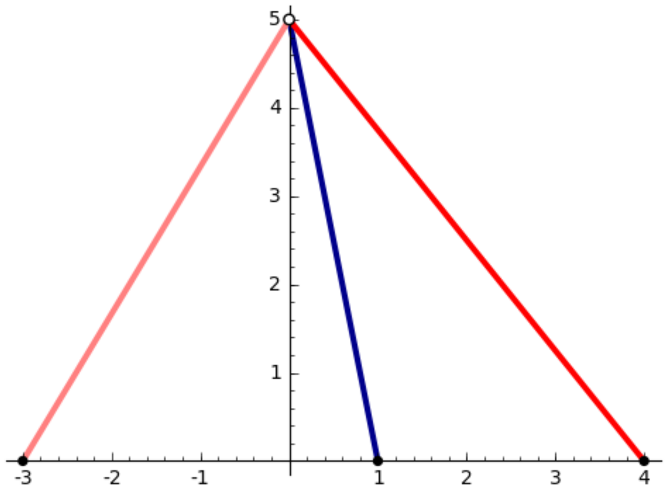
```
S3ST = matrix (kernel (A3Ar, pivots3))
S3S = S3ST.T
show (S3S)
```

$$\begin{pmatrix} -0.490090178483 \\ -0.717574844717 \\ 1.0 \end{pmatrix}$$

```
S3S0 = S3S.column(0)
dict_S3S0 = dict_bar_force (bars, S3S0)
print_dict (dict_S3S0)
```

```
0:   -0.490090178483
1:   -0.717574844717
2:   1.0
```

```
plot_truss_with_forces2d (joints, bars3, supports, dict_S3S0) . show
(figsize = 5)
```



```
ddelta3 = diag_flex (bars3, joints, E, Fs)
show (ddelta3)
```

$$\begin{pmatrix} 1.16619037897 \times 10^{-05} & 0.0 & 0.0 \\ 0.0 & 1.28062484749 \times 10^{-05} & 0.0 \\ 0.0 & 0.0 & 1.01980390272 \times 10^{-05} \end{pmatrix}$$

```
D3D = S3ST * ddelta3 * S3S
show (D3D)
```

$\left(1.95932050862 \times 10^{-05}\right)$

```
d3d0 = S3ST * ddelta3 * s3s0
d3d0
```

```
(0.000234319419929397 22)
```

```
x3x = D3D \ (-d3d0)
x3x
```

```
(-11.95921845858131)
```

```
s3s = s3s0 + S3S*x3x
s3s
```

```
(126.6450990449649, -92.03888940273302, -11.95921845858131)
```

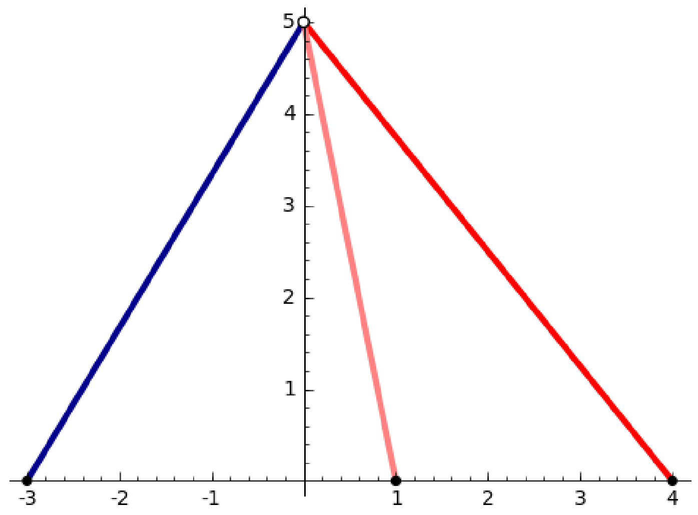
```
A3A * s3s
```

```
(-125.00000000000003, -25.000000000000002)
```

```
dict_s3s = dict_bar_force (bars3, s3s)
print_dict (dict_s3s)
```

```
0:   126.645099045
1:   -92.0388894027
2:   -11.9592184586
```

```
plot_truss_with_forces2d (joints, bars3, supports, dict_s3s) . show
(figsize = 5)
```


5
4
3
2
1
-3
-2
-1
1
2
3
4

Primjer 2.

Primjer prikazuje prostorni rešetkasti sistem sa pet štapova. Program bira osnovni sistem ovisno o tome kojim su redoslijedom zadani štapovi. U ovom primjeru je program za osnovni sistem odabrao sile u tri štapa koja leže gotovo u istoj ravnini pa su stoga dobivene jako velike sile u štapovima odabranog osnovnog sistema. Da bi sustav štapova koji leže u gotovo jednoj ravnini, a na koji djeluje sila izvan te ravnine, bio u ravnoteži potrebne su jako velike sile u štapovima tog sustava.

```
joints = { 0: (-2.0, 0.0, 0.0), 1: (0.0, 0.0, 0.0), 2: (2.425, 0.425,
0.0), 3: (0.625, -2.325, 0.0), 4: (0.0, 2.625, 0.0), 5: (0.0, 0.0,
3.025) }
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, 125.0, -25.0) }
print 'loads:'
print_dict (loads, indent = '   ')
```

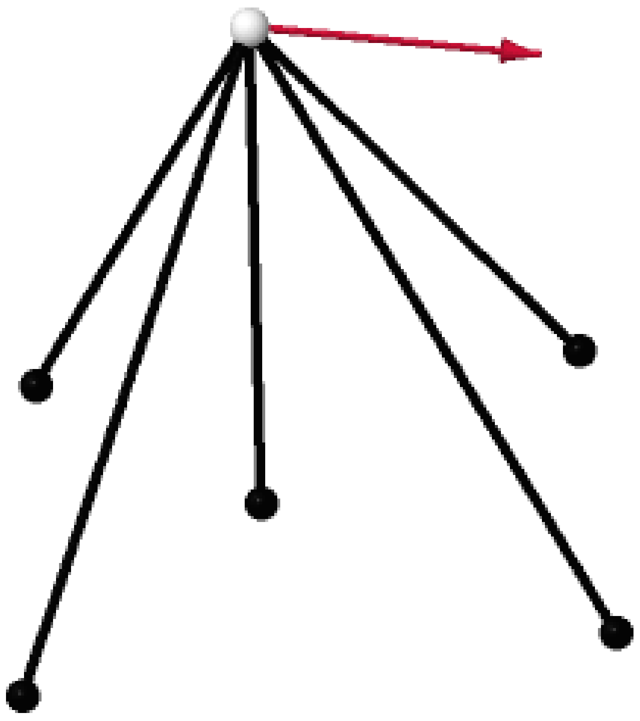
```
joints:
   0:   (-2.00000000000000, 0.000000000000000,
0.000000000000000)
   1:   (0.000000000000000, 0.000000000000000,
0.000000000000000)
   2:   (2.42500000000000, 0.425000000000000, 0.000000000000000)
   3:   (0.625000000000000, -2.32500000000000,
0.000000000000000)
   4:   (0.000000000000000, 2.62500000000000, 0.000000000000000)
   5:   (0.000000000000000, 0.000000000000000, 3.02500000000000)

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, 125.000000000000, -25.0000000000000)
```

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



```
free_joints = others (supports, joints)
free_joints
```

```
[5]
```

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

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

```
AAf = augmented_equil_matrix (joints, bars, loads, free_joints)
show (AAf)
```

$$\left(\begin{array}{rrrrr|r}
-0.551514326966 & 0.0 & 0.621756549717 & 0.161660834408 & 0.0 & -100.0 \\
0.0 & 0.0 & 0.108967642734 & -0.601378303999 & 0.655405684946 & -125.0 \\
-0.834165419537 & -1.0 & -0.775593221812 & -0.782438438536 & -0.755277027414 & 25.0
\end{array}\right)$$

```
AA = AAf.subdivision (0, 0)
show (AA)
```

$$\left(\begin{array}{rrrrr}
-0.551514326966 & 0.0 & 0.621756549717 & 0.161660834408 & 0.0 \\
0.0 & 0.0 & 0.108967642734 & -0.601378303999 & 0.655405684946 \\
-0.834165419537 & -1.0 & -0.775593221812 & -0.782438438536 & -0.755277027414
\end{array}\right)$$

```
AAfr, pivots = ref_wpp (AAf)
show (AAfr)
```

$$\left(\begin{array}{rrrrr|r}
1.0 & 1.19880299108 & 0.929783474173 & 0.937989540456 & 0.905428359562 & -29.9700747771 \\
0.0 & 1.0 & 1.71600000326 & 1.02695045058 & 0.755277027414 & -176.25 \\
0.0 & 0.0 & 1.0 & -5.51887045466 & 6.01468168442 & -1147.12952271
\end{array}\right)$$

```
pivots
```

```
[0, 1, 2]
```

Metoda sila

```
primary_system_bars = pivots
print primary_system_bars
print_list (list_indexed (primary_system_bars, bars), indent = '
')
print
#
redundant_bars = others (primary_system_bars, all_indices (bars))
print redundant_bars
print_list (list_indexed (redundant_bars, bars), indent = '    ')
```

```
[0, 1, 2]
   (0, 5)
   (1, 5)
   (2, 5)

[3, 4]
   (3, 5)
   (4, 5)
```

```
prim_bar_forces = ref_solution (AAfr, primary_system_bars)
prim_bar_forces
```

```
(-1111.9118110507566, 1792.224264705882, -1147.1295227080996)
```

```
ss0 = all_bar_forces (primary_system_bars, redundant_bars,
prim_bar_forces)
ss0
```

```
(-1111.9118110507566, 1792.224264705882, -1147.1295227080996,
0.0, 0.0)
```

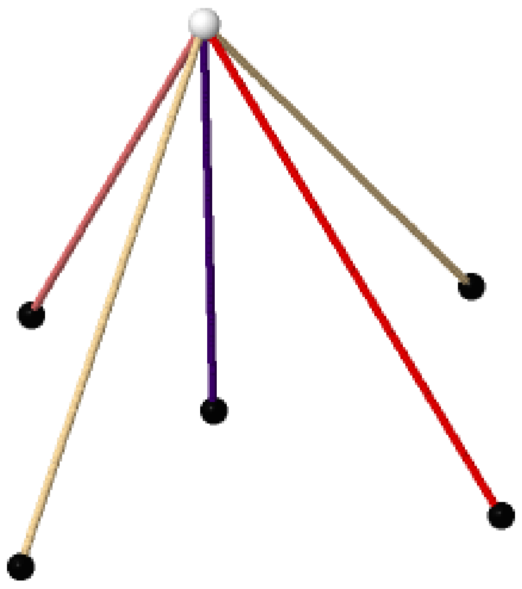
```
AA * ss0
```

```
(-100.00000000000011, -125.0, 25.0)
```

```
dict_ss0 = dict_bar_force (bars, ss0)
print_dict (dict_ss0)
```

```
0:   -1111.91181105
1:   1792.22426471
2:   -1147.12952271
3:   0.0
4:   0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_ss0)
```



```
AAr = AAfr.subdivision (0, 0)
```

```
SST = matrix (kernel (AAr, pivots))
SS = SST.T
show (SS)
```

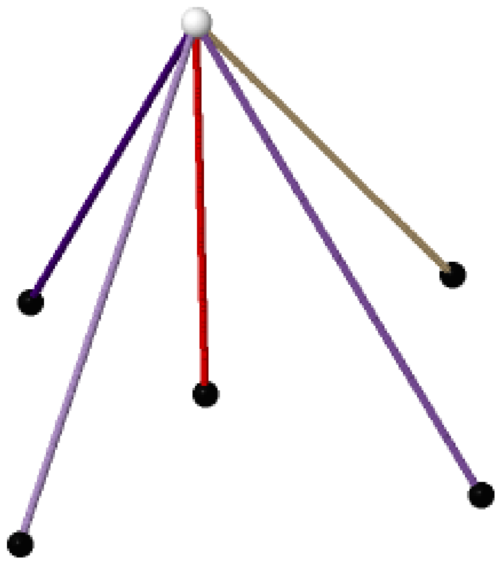
$$\left(\begin{array}{rr} 6.51488911702 & -6.78072635451 \\ -10.4973321688 & 9.56591676266 \\ 5.51887045466 & -6.01468168442 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{array}\right)$$

```
SS0 = SS.column(0)
SS1 = SS.column(1)
```

```
dict_SS0 = dict_bar_force (bars, SS0)
print_dict (dict_SS0)
```

```
0:   6.51488911702
1:   -10.4973321688
2:   5.51887045466
3:   1.0
4:   0.0
```

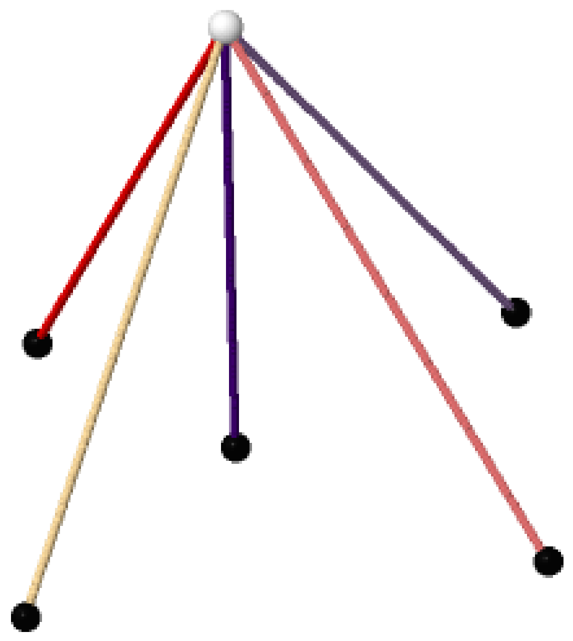
```
plot_truss_with_forces3d (joints, bars, supports, dict_SS0)
```



```
dict_SS1 = dict_bar_force (bars, SS1)
print_dict (dict_SS1)
```

```
0:   -6.78072635451
1:   9.56591676266
2:   -6.01468168442
3:   0.0
4:   1.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_SS1)
```



```
E = 2.e8
Fs = dict_same_value (bars, 0.05*0.05)
print E
print
print_dict (Fs)
```

```
2.00000000000000e8

0:   0.00250000000000000
1:   0.00250000000000000
2:   0.00250000000000000
3:   0.00250000000000000
4:   0.00250000000000000
```

```
ddelta = diag_flex (bars, joints, E, Fs)
show (ddelta)
```

$$\left(\begin{array}{rrrrr}
7.25275809606 \times 10^{-06} & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 6.05 \times 10^{-06} & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 7.80048075442 \times 10^{-06} & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 7.73223770974 \times 10^{-06} & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 8.01030586183 \times 10^{-06}
\end{array}\right)$$

```
DD = SST * ddelta * SS
show (DD)
```

$$\begin{pmatrix} 0.00121982680866 & -0.00118684708882 \\ -0.00118684708882 & 0.00117728862819 \end{pmatrix}$$

```
dd0 = SST * ddelta * ss0
dd0
```

```
(-0.21574471505962056, 0.21222584223481203)
```

```
xx = DD \ (-dd0)
xx
```

```
(76.92889294554278, -102.7131382232714)
```

```
ss = ss0 + SS*xx
ss
```

```
(85.74107969007537, 2.1307914059143513, -104.78209710001738,
76.92889294554278, -102.7131382232714)
```

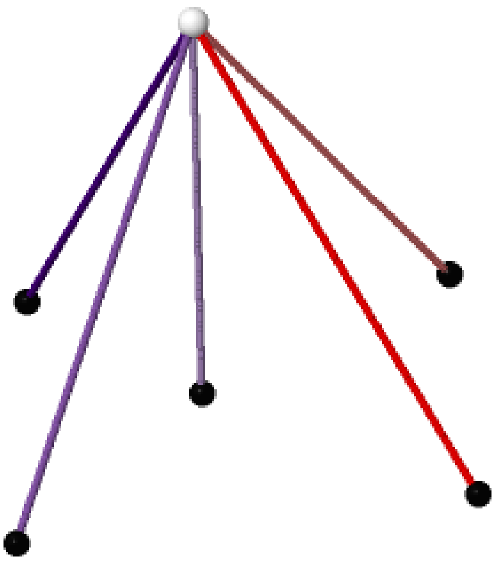
```
AA * ss
```

```
(-99.99999999999987, -125.0, 24.999999999999744)
```

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

```
0:   85.7410796901
1:   2.13079140591
2:   -104.7820971
3:   76.9288929455
4:   -102.713138223
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_ss)
```



Varijacija — drugi osnovni sistem

Zadan je drugi redoslijed štapova te program odabire drugi osnovni sistem. Štapovi drugog odabranog osnovnog sistema ne daju prevelike sile osnovnog sistema.

```
bars2 = { 0: (2, 5), 1: (4, 5), 2: (0, 5), 3: (1, 5), 4: (3, 5) }
print 'bars:'
print_dict (bars2, indent = '    ')
print
```

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

```
A2Af = augmented_equil_matrix (joints, bars2, loads, free_joints)
show (A2Af)
```

$$\left(\begin{array}{rrrrr|r}
0.621756549717 & 0.0 & -0.551514326966 & 0.0 & 0.161660834408 & -100.0 \\
0.108967642734 & 0.655405684946 & 0.0 & 0.0 & -0.601378303999 & -125.0 \\
-0.775593221812 & -0.755277027414 & -0.834165419537 & -1.0 & -0.782438438536 & 25.0
\end{array}\right)$$

```
A2A = A2Af.subdivision (0, 0)
show (A2A)
```

$$\left(\begin{array}{rrrrr}
0.621756549717 & 0.0 & -0.551514326966 & 0.0 & 0.161660834408 \\
0.108967642734 & 0.655405684946 & 0.0 & 0.0 & -0.601378303999 \\
-0.775593221812 & -0.755277027414 & -0.834165419537 & -1.0 & -0.782438438536
\end{array}\right)$$

```
A2Afr, pivots2 = ref_wpp (A2Af)
show (A2Afr)
```

$$\left(\begin{array}{rrrrr|r}
1.0 & 0.973805606049 & 1.07551922332 & 1.28933566189 & 1.00882578204 & -32.2333915472 \\
0.0 & 1.0 & 2.01533588911 & 1.32401749782 & 0.768961620809 & 132.060508159 \\
0.0 & 0.0 & 1.0 & 0.7088422911 & 0.926063867925 & 158.492542908
\end{array}\right)$$

```
pivots2
```

```
[0, 1, 2]
```

```
primary_system_bars2 = pivots2
print primary_system_bars2
print_list (list_indexed (primary_system_bars2, bars2), indent = '    ')
print
#
redundant_bars2 = others (primary_system_bars2, all_indices (bars2))
print redundant_bars2
print_list (list_indexed (redundant_bars2, bars2), indent = '    ')
```

```
[0, 1, 2]
   (2, 5)
   (4, 5)
   (0, 5)

[3, 4]
   (1, 5)
   (3, 5)
```

```
prim_bar_forces2 = ref_solution (A2Afr, primary_system_bars2)
prim_bar_forces2
```

```
(-20.24762244066045, -187.35520172016587, 158.49254290840108)
```

```
s2s0 = all_bar_forces (primary_system_bars2, redundant_bars2,
prim_bar_forces2)
s2s0
```

```
(-20.24762244066045, -187.35520172016587, 158.49254290840108,
0.0, 0.0)
```

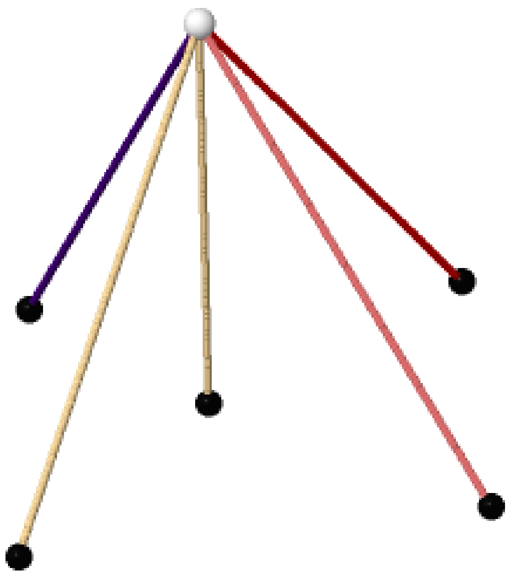
```
A2A * s2s0
```

```
(-100.0, -125.00000000000003, 25.0)
```

```
dict_s2s0 = dict_bar_force (bars2, s2s0)
print_dict (dict_s2s0)
```

```
0:   -20.2476224407
1:   -187.35520172
2:   158.492542908
3:   0.0
4:   0.0
```

```
plot_truss_with_forces3d (joints, bars2, supports, dict_s2s0)
```



```
A2Ar = A2Afr.subdivision (0, 0)
```

```
S2ST = matrix (kernel (A2Ar, pivots2))
S2S = S2ST.T
show (S2S)
```

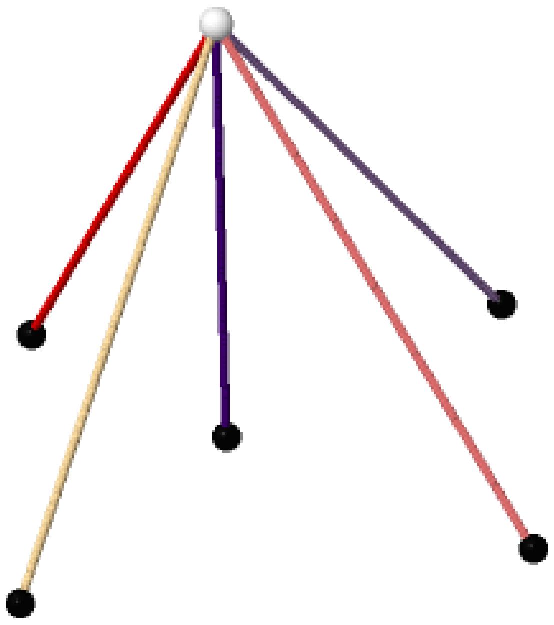
$$\left(\begin{array}{rr}
-0.628761658046 & -1.08144952483 \\
0.104537811149 & 1.09736812783 \\
-0.7088422911 & -0.926063867925 \\
1.0 & 0.0 \\
0.0 & 1.0
\end{array}\right)$$

```
S2S0 = S2S.column(0)
S2S1 = S2S.column(1)
```

```
dict_S2S0 = dict_bar_force (bars2, S2S0)
print_dict (dict_S2S0)
```

```
0:   -0.628761658046
1:   0.104537811149
2:   -0.7088422911
3:   1.0
4:   0.0
```

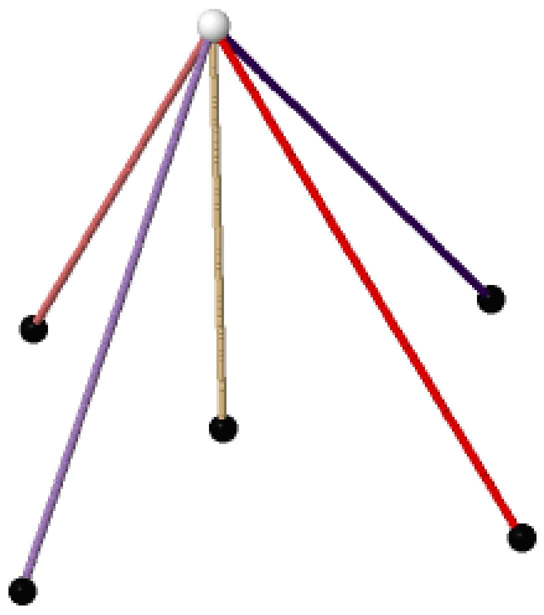
```
plot_truss_with_forces3d (joints, bars2, supports, dict_S2S0)
```



```
dict_S2S1 = dict_bar_force (bars2, S2S1)
print_dict (dict_S2S1)
```

```
0:   -1.08144952483
1:   1.09736812783
2:   -0.926063867925
3:   0.0
4:   1.0
```

```
plot_truss_with_forces3d (joints, bars2, supports, dict_S2S1)
```



```
ddelta2 = diag_flex (bars2, joints, E, Fs)
show (ddelta2)
```

$$\begin{pmatrix} 7.80048075442 \times 10^{-06} & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 8.01030586183 \times 10^{-06} & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 7.25275809606 \times 10^{-06} & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 6.05 \times 10^{-06} & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 7.73223770974 \times 10^{-06} \end{pmatrix}$$

```
D2D = S2ST * ddelta2 * S2S
show (D2D)
```

$$\begin{pmatrix} 1.2865591384 \times 10^{-05} & 1.0983989472 \times 10^{-05} \\ 1.0983989472 \times 10^{-05} & 3.27212268185 \times 10^{-05} \end{pmatrix}$$

```
d2d0 = S2ST * ddelta2 * s2s0
d2d0
```

```
(-0.000872400041761093, -0.002540612345336011)
```

```
x2x = D2D \ (-d2d0)
x2x
```

```
(2.130791405914571, 76.92889294554227)
```

```
s2s = s2s0 + S2S*x2x
s2s
```

```
(-104.78209710001724, -102.71313822327193, 85.74107969007562,
2.130791405914571, 76.92889294554227)
```

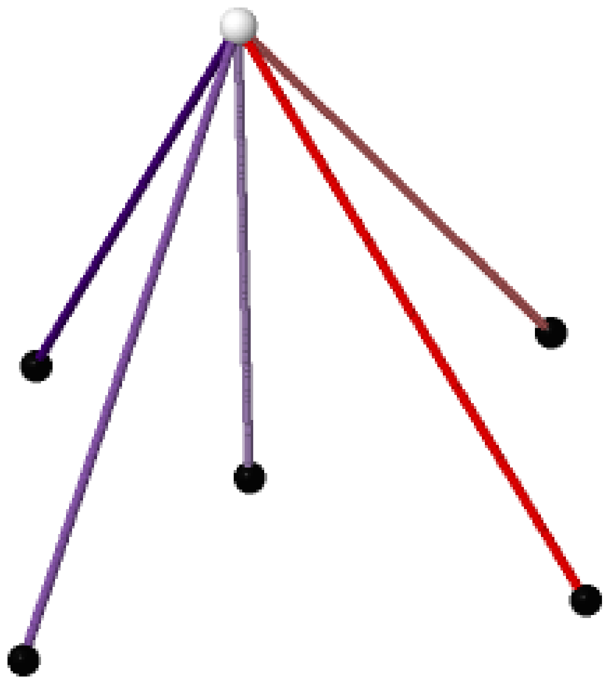
```
A2A * s2s
```

```
(-100.00000000000001, -125.00000000000001, 25.0)
```

```
dict_s2s = dict_bar_force (bars2, s2s)
print_dict (dict_s2s)
```

```
0:   -104.7820971
1:   -102.713138223
2:   85.7410796901
3:   2.13079140591
4:   76.9288929455
```

```
plot_truss_with_forces3d (joints, bars2, supports, dict_s2s)
```



Primjer 3 – „Schwedlerova“ kupola

Statički neodređena „Schwedlerova kupola“ se zadaje funkcijom ***schwdler_sind*** .Radi Schwedlerovu kupolu koja ima obje dijagonale u svakom polju. Takva Schwedlerova kupola je statički neodređena jer ima višak dijagonala. Prvi parametar je radijus ležajnog prstena. Drugi parametar su visine prstenova te visina „tjemena“, a treći je broj stranica poligona. Kupola mora biti polukugla ili njezin odsječak, odnosno visina mora biti manja ili jednaka radijusu.

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

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

    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)))
    bsd = dict (zip (xrange (len (bsl)), bsl))

    ssl = range (n)
    return jsd, bsd, ssl
```

Opterećenja u čvorovima se zadaju sa funkcijom **make_loads** i **load_gen**. Tim funkcijama se zadaju opterećenja u čvorovima kada se više čvorova želi opteretiti jednakom silom, tako da se ne mora za svaki čvor pisati posebno. Parametri funkcije make_loads su broj opterećenih čvorova, prvi opterećeni čvor, korak i trojka komponenata sila u x, y, z smjeru. Generator je lista opterećenja opisanih navedenih parametrima koju zadajemo.

```
def make_loads__ (nl, n0, step, ld) :
    return zip (range (n0, n0 + nl*step, step), [ld for i in xrange
(nl)])
```

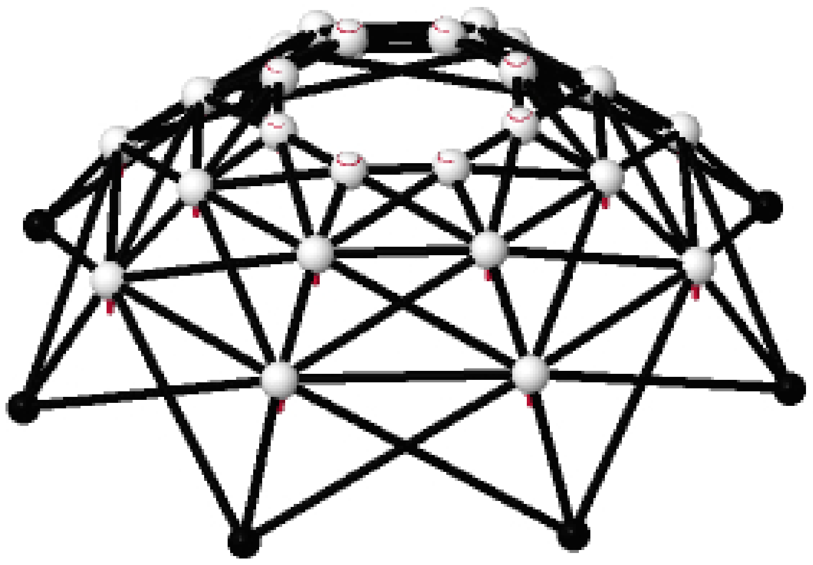
```
def make_loads (generator) :
    ll = []
    for gg in generator :
        if len (gg) == 3 :
            nl, n0, ld = gg
            step = 1
        else :
            nl, n0, step, ld = gg
        ll.extend (make_loads__ (nl, n0, step, ld))
    return dict (ll)
```

U ovom primjeru je prikazana statički neodređena „Schwedlerova“ kupola koja ima dvije ukrižene dijagonale u svakom polju. Zadano opterećenje u čvorovima je vertikalno te se zbog toga u osnovnom sistemu, koji je dobiven tako da se jedan niz dijagonala smatra prekobrojnim, ni u dijagonalama drugog niza ne javljaju se uzdužne sile. U ovom primjeru jednostavnosti radi uzeli smo da kupola ima sve štapove iste površine.

```
joints, bars, supports = schwedler_sind (10., [3, 5, 6, 6.5], 8)
```

```
load_gen = [(8, 8, (0.0, 0.0, -90.0)),
            (8, 16, (0.0, 0.0, -80.0)),
            (8, 24, (0.0, 0.0, -50.0))]
loads = make_loads (load_gen)
```

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



```
free_joints = others (supports, joints)
free_joints
```

```
[8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,
29,30,31]
```

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

```
3 * 24 - 96  ==  -24  !=  0
```

```
AAf = augmented_equil_matrix (joints, bars, loads, free_joints)
AAf
```

```
72 x 97 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
AA = AAf.subdivision (0, 0)
AA
```

```
72 x 96 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
AAfr, pivots = ref_wpp (AAf)
AAfr
```

```
72 x 97 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
pivots
```

```
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
 32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,
 53,54,55,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,
 82,83,84,85,86,87]
```

```
primary_system_bars = pivots
print primary_system_bars
print_list (list_indexed (primary_system_bars, bars), indent = '   ')
print
#
redundant_bars = others (primary_system_bars, all_indices (bars))
print redundant_bars
print_list (list_indexed (redundant_bars, bars), indent = '   ')
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19,20, 21, 22, 23, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45,46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 64, 65,
66, 67, 68, 69, 70, 71,72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87]
   (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)
(16, 24)
(17, 25)
(18, 26)
(19, 27)
(20, 28)
(21, 29)
(22, 30)
(23, 31)
(24, 25)
(25, 26)
(26, 27)
(27, 28)
(28, 29)
(29, 30)
(30, 31)
(31, 24)
(16, 25)
```

```
   (17, 26)
   (18, 27)
   (19, 28)
   (20, 29)
   (21, 30)
   (22, 31)
   (23, 24)

[24, 25, 26, 27, 28, 29, 30, 31, 56, 57, 58, 59, 60, 61, 62, 63,
88, 89,90, 91, 92, 93, 94, 95]
   (0, 15)
   (1, 8)
   (2, 9)
   (3, 10)
   (4, 11)
   (5, 12)
   (6, 13)
   (7, 14)
   (8, 23)
   (9, 16)
   (10, 17)
   (11, 18)
   (12, 19)
   (13, 20)
   (14, 21)
   (15, 22)
   (16, 31)
   (17, 24)
   (18, 25)
   (19, 26)
   (20, 27)
   (21, 28)
   (22, 29)
   (23, 30)
```

```
prim_bar_forces = ref_solution (AAfr, primary_system_bars)
prim_bar_forces
```

```
(-281.76760906656517, -281.76760906656534, -281.7676090665652,
-281.76760906656534, -281.76760906656517, -281.76760906656557,
-281.767609066565, -281.7676090665657, -32.01047864505287,
-32.01047864505281, -32.010478645052864, -32.0104786450527,
-32.010478645052935, -32.010478645052764, -32.010478645053055,
-32.01047864505274, 2.3155759809037474e-14, -2.842170943040401e-
14,
4.752613235570464e-14, -1.5099033134902131e-13,
6.765843308643685e-14,
-1.2434497875801756e-13, 2.265549618901036e-13, -
9.237055564881302e-14,
-199.6670834718586, -199.66708347185866, -199.66708347185863,
-199.6670834718587, -199.66708347185863, -199.66708347185872,
-199.66708347185852, -199.6670834718588, -57.85734423920378,
-57.85734423920373, -57.857344239203755, -57.857344239203705,
-57.85734423920382, -57.85734423920363, -57.8573442392039,
-57.85734423920362, 3.380307101168053e-14, -3.059598761762708e-
14,
3.6168136954704156e-14, -1.0627312656085139e-13,
1.2304806640629603e-13,
-1.8318658920346015e-13, 2.3445183221310427e-13,
-1.5631940186722204e-13, -118.34728240234361, -
118.34728240234365,
-118.34728240234365, -118.34728240234367, -118.34728240234361,
-118.3472824023437, -118.34728240234364, -118.34728240234375,
-140.15029759509366, -140.15029759509366, -140.1502975950937,
-140.15029759509366, -140.1502975950937, -140.15029759509366,
-140.15029759509375, -140.15029759509363, 0.0, -
1.7083870736439967e-14,
2.1482228976552236e-14, -3.8566099712992266e-14,
3.8566099712992266e-14,
-3.8566099712992285e-14, 1.3498134899547292e-13, -
1.928304985649612e-14)
```

```
ss0 = all_bar_forces (primary_system_bars, redundant_bars,
prim_bar_forces)
ss0
```

```
(-281.76760906656517, -281.76760906656534, -281.7676090665652,
-281.76760906656534, -281.76760906656517, -281.76760906656557,
-281.767609066565, -281.7676090665657, -32.01047864505287,
-32.01047864505281, -32.010478645052864, -32.0104786450527,
-32.010478645052935, -32.010478645052764, -32.010478645053055,
-32.01047864505274, 2.3155759809037474e-14, -2.842170943040401e-
14,
4.752613235570464e-14, -1.5099033134902131e-13,
6.765843308643685e-14,
-1.2434497875801756e-13, 2.265549618901036e-13, -
9.237055564881302e-14,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -199.6670834718586,
-199.66708347185866, -199.66708347185863, -199.6670834718587,
-199.66708347185863, -199.66708347185872, -199.66708347185852,
-199.6670834718588, -57.85734423920378, -57.85734423920373,
-57.857344239203755, -57.857344239203705, -57.85734423920382,
-57.85734423920363, -57.8573442392039, -57.85734423920362,
3.380307101168053e-14, -3.059598761762708e-14,
3.6168136954704156e-14,
-1.0627312656085139e-13, 1.2304806640629603e-13,
-1.8318658920346015e-13, 2.3445183221310427e-13,
-1.5631940186722204e-13, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
-118.34728240234361, -118.34728240234365, -118.34728240234365,
-118.34728240234367, -118.34728240234361, -118.3472824023437,
-118.34728240234364, -118.34728240234375, -140.15029759509366,
-140.15029759509366, -140.1502975950937, -140.15029759509366,
-140.1502975950937, -140.15029759509366, -140.15029759509375,
-140.15029759509363, 0.0, -1.7083870736439967e-14,
2.1482228976552236e-14, -3.8566099712992266e-14,
3.8566099712992266e-14,
-3.8566099712992285e-14, 1.3498134899547292e-13, -
1.928304985649612e-14,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
```

```
AA * ss0
```

```
(-2.2837935000767606e-14, -1.0449315790553649e-14,
90.00000000000001,
-1.3963152487757814e-14, 5.640435437732197e-16,
90.00000000000004,
2.3167263414638915e-15, 3.985895282483222e-15,
89.99999999999999,
-3.0380877137698674e-14, 2.852701961892914e-14,
89.99999999999996,
2.6289966907191645e-14, -1.0843500747547877e-15,
90.00000000000004,
1.3089462012560565e-14, 1.0833770953848911e-14,
90.00000000000006,
6.154550229182829e-16, 1.629113502607419e-14, 90.00000000000003,
-1.1329073270797818e-14, 9.843160006735557e-15,
90.00000000000003,
-4.263256414560601e-14, 4.367694708466446e-15,
80.00000000000001,
2.0911165108829402e-15, -1.1574396317835024e-14, 80.0,
-1.2935732907155725e-14, 1.1582487091825363e-14,
79.99999999999997,
2.2470018562998218e-14, 8.379285936676103e-15,
80.00000000000001,
7.286327849595176e-14, -1.5888145362895355e-14,
80.00000000000003,
-8.379285936676122e-15, -5.95169086740579e-15,
79.99999999999997,
9.21590380204023e-16, 3.4753226650200076e-14, 79.99999999999997,
2.975845433702891e-15, 1.0021211746863965e-14, 80.0,
1.8340436639100116e-14, -1.0021211746863965e-14,
49.99999999999999,
-2.842170943040401e-14, 0.0, 50.0, 1.2119738204319063e-14,
4.468968960234022e-15, 50.00000000000001, 5.04534825942912e-15,
-4.000419652222937e-14, 50.0, -1.5364591205397216e-14,
-8.379285936676103e-15, 49.99999999999999, 2.6612704510462273e-
14,
7.602897691406385e-15, 50.00000000000001, -2.0042423493727885e-
14,
-1.1537364901952118e-15, 50.0, -2.661014191992235e-14,
7.879337495405936e-15, 50.0)
```

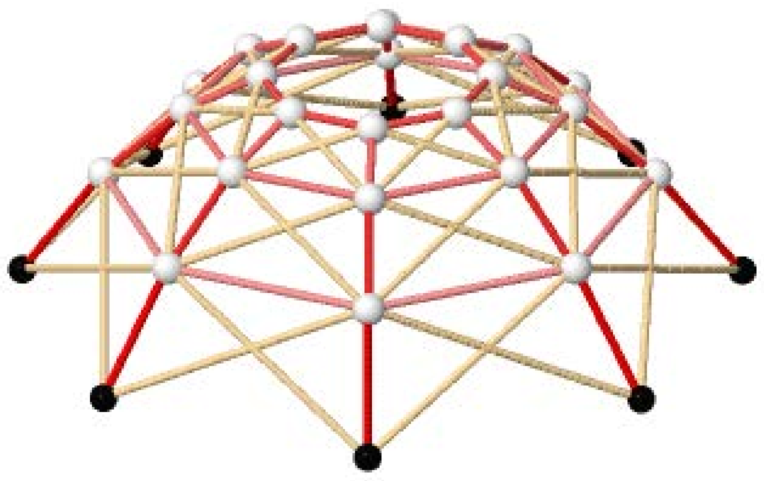
```
dict_ss0 = dict_bar_force (bars, ss0)
print_dict (dict_ss0)
```

```
0:   -281.767609067
1:   -281.767609067
2:   -281.767609067
3:   -281.767609067
4:   -281.767609067
5:   -281.767609067
6:   -281.767609067
7:   -281.767609067
8:   -32.0104786451
9:   -32.0104786451
10:  -32.0104786451
11:  -32.0104786451
12:  -32.0104786451
13:  -32.0104786451
14:  -32.0104786451
15:  -32.0104786451
16:  2.3155759809e-14
17:  -2.84217094304e-14
18:  4.75261323557e-14
19:  -1.50990331349e-13
20:  6.76584330864e-14
21:  -1.24344978758e-13
22:  2.2655496189e-13
23:  -9.23705556488e-14
24:  0.0
25:  0.0
26:  0.0
27:  0.0
28:  0.0
29:  0.0
30:  0.0
31:  0.0
32:  -199.667083472
33:  -199.667083472
34:  -199.667083472
35:  -199.667083472
36:  -199.667083472
37:  -199.667083472
```

```
38:  -199.667083472
39:  -199.667083472
40:  -57.8573442392
41:  -57.8573442392
42:  -57.8573442392
43:  -57.8573442392
44:  -57.8573442392
45:  -57.8573442392
46:  -57.8573442392
47:  -57.8573442392
48:  3.38030710117e-14
49:  -3.05959876176e-14
50:  3.61681369547e-14
51:  -1.06273126561e-13
52:  1.23048066406e-13
53:  -1.83186589203e-13
54:  2.34451832213e-13
55:  -1.56319401867e-13
56:  0.0
57:  0.0
58:  0.0
59:  0.0
60:  0.0
61:  0.0
62:  0.0
63:  0.0
64:  -118.347282402
65:  -118.347282402
66:  -118.347282402
67:  -118.347282402
68:  -118.347282402
69:  -118.347282402
70:  -118.347282402
71:  -118.347282402
72:  -140.150297595
73:  -140.150297595
74:  -140.150297595
75:  -140.150297595
76:  -140.150297595
77:  -140.150297595
78:  -140.150297595
79:  -140.150297595
80:  0.0
81:  -1.70838707364e-14
82:  2.14822289766e-14
83:  -3.8566099713e-14
84:  3.8566099713e-14
85:  -3.8566099713e-14
86:  1.34981348995e-13
87:  -1.92830498565e-14
88:  0.0
```

```
89:  0.0
90:  0.0
91:  0.0
92:  0.0
93:  0.0
94:  0.0
95:  0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_ss0)
```



```
AAr = AAfr.subdivision (0, 0)
```

```
SST = matrix (kernel (AAr, pivots))
SS = SST.T
SS
```

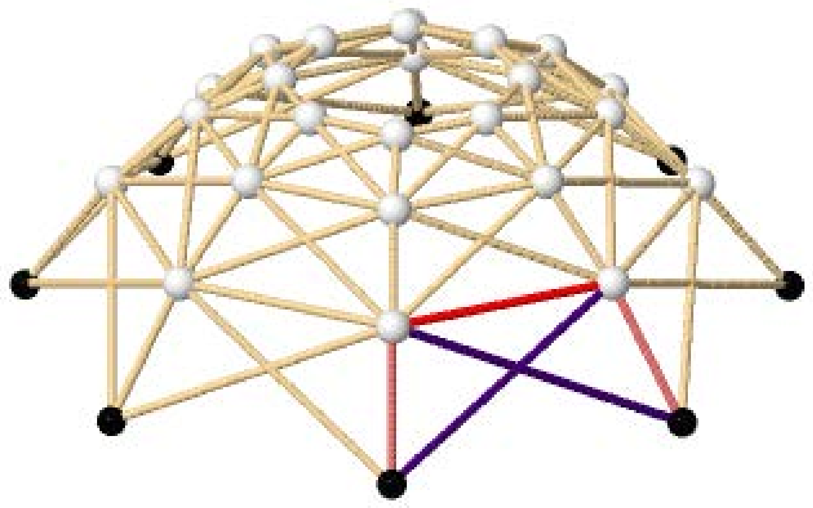
```
96 x 24 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
SS0 = SS.column(0)
SS1 = SS.column(1)
```

```
dict_SS0 = dict_bar_force (bars, SS0)
print_dict (dict_SS0)
```

```
0:   -0.49904411251
1:   -3.46281415008e-17
2:   0.0
3:   0.0
4:   0.0
5:   1.38512566003e-17
6:   0.0
7:   -0.49904411251
8:   -6.02988354968e-17
9:   2.09528202549e-17
10:  2.09528202549e-17
11:  5.05845828295e-17
12:  5.05845828295e-17
13:  5.55111512313e-17
14:  5.55111512313e-17
15:  -0.994074542306
16:  6.93889390391e-17
17:  0.0
18:  0.0
19:  0.0
20:  -2.77555756156e-17
21:  0.0
22:  -1.11022302463e-16
23:  1.0
24:  1.0
25 - 95:  0.0
```

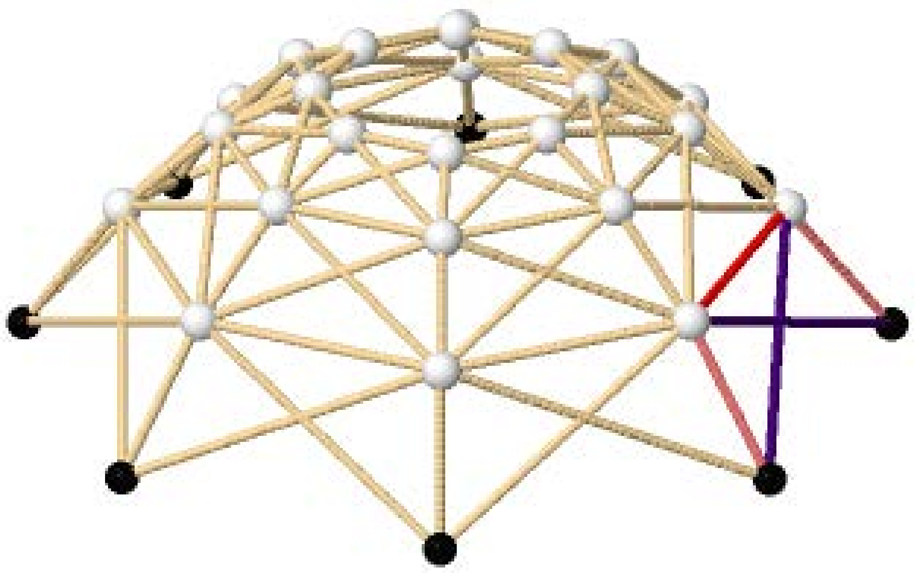
```
plot_truss_with_forces3d (joints, bars, supports, dict_SS0)
```



```
dict_SS1 = dict_bar_force (bars, SS1)
print_dict (dict_SS1)
```

```
0:   -0.49904411251
1:   -0.49904411251
2:   -3.64108286064e-17
3:   5.07756755679e-33
4:   -1.74998167042e-48
5:   4.28132324248e-33
6:   -7.26850480829e-50
7:   3.01637180521e-17
8:   -0.994074542306
9:   -7.80001102808e-17
10:  -5.47129594438e-18
11:  -1.32088768727e-17
12:  -1.32088768727e-17
13:  -3.18890496898e-17
14:  -3.18890496898e-17
15:  -2.22044604925e-16
16:  1.0
17:  7.29611424996e-17
18:  -1.01745866337e-32
19:  3.50666729964e-48
20:  -8.5790476937e-33
21:  1.45648543407e-49
22:  -6.04429894992e-17
23:  1.45922284999e-16
24:  0.0
25:  1.0
26 - 95:  0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_SS1)
```



```
E = 2.e8
Fs = dict_same_value (bars, 0.05*0.05)
print E
print
print_dict (Fs)
```

```
2.00000000000000e8

0 - 95:   0.00250000000000000
```

```
ddelta = diag_flex (bars, joints, E, Fs)
ddelta
```

```
96 x 96 sparse matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
DD = SST * ddelta * SS
DD
```

```
24 x 24 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
dd0 = SST * ddelta * ss0
dd0
```

```
(0.002531281426763446, 0.002531281426763447,
0.002531281426763446,
0.0025312814267634486, 0.0025312814267634425,
0.00253128142676345,
0.0025312814267634447, 0.002531281426763454,
0.002049131209634814,
0.0020491312096348174, 0.002049131209634815,
0.002049131209634817,
0.0020491312096348126, 0.0020491312096348187,
0.002049131209634812,
0.0020491312096348204, 0.0016569231225757074,
0.0016569231225757094,
0.0016569231225757061, 0.0016569231225757083,
0.0016569231225757048,
0.0016569231225757053, 0.0016569231225757098,
0.0016569231225757094)
```

```
xx = DD \ (-dd0)
xx
```

```
(-45.36827094416766, -45.36827094416766, -45.36827094416771,
-45.3682709441677, -45.368270944167655, -45.36827094416775,
-45.368270944167655, -45.36827094416777, -32.62615383616719,
-32.6261538361672, -32.626153836167234, -32.626153836167234,
-32.62615383616723, -32.62615383616733, -32.62615383616712,
-32.62615383616731, -48.654158205960925, -48.65415820596099,
-48.654158205960876, -48.65415820596094, -48.654158205960854,
-48.65415820596088, -48.654158205961004, -48.65415820596095)
```

```
ss = ss0 + SS*xx
ss
```

```
(-236.4860720476919, -236.48607204769186, -236.4860720476919,
-236.4860720476919, -236.4860720476919, -236.48607204769218,
-236.4860720476916, -236.4860720476922, 36.027217462178456,
36.027217462178506, 36.02721746217849, 36.027217462178484,
36.027217462178534, 36.027217462178406, 36.02721746217844,
36.027217462178484, -45.3682709441677, -45.36827094416769,
-45.36827094416772, -45.3682709441677, -45.368270944167655,
-45.368270944167776, -45.36827094416758, -45.36827094416769,
-45.36827094416766, -45.36827094416766, -45.36827094416771,
-45.3682709441677, -45.368270944167655, -45.36827094416775,
-45.368270944167655, -45.36827094416777, -164.71425702409903,
-164.71425702409906, -164.71425702409908, -164.71425702409908,
-164.71425702409903, -164.71425702409914, -164.7142570240989,
-164.71425702409917, 5.020442932858828, 5.020442932858764,
5.020442932858828, 5.020442932858735, 5.020442932858813,
5.020442932858863, 5.020442932858799, 5.0204429328588915,
-32.62615383616722, -32.62615383616722, -32.626153836167234,
-32.62615383616722, -32.62615383616722, -32.62615383616726,
-32.62615383616713, -32.626153836167276, -32.62615383616719,
-32.6261538361672, -32.626153836167234, -32.626153836167234,
-32.62615383616723, -32.62615383616733, -32.62615383616712,
-32.62615383616731, -59.34407863995311, -59.34407863995317,
-59.34407863995323, -59.344078639953246, -59.34407863995324,
-59.3440786399532, -59.3440786399531, -59.34407863995326,
-89.89770614442216, -89.89770614442227, -89.89770614442226,
-89.89770614442233, -89.89770614442232, -89.89770614442213,
-89.89770614442229, -89.89770614442219, -48.654158205961004,
-48.65415820596089, -48.654158205960925, -48.654158205960876,
-48.65415820596082, -48.65415820596107, -48.65415820596082,
-48.65415820596097, -48.654158205960925, -48.65415820596099,
-48.654158205960876, -48.65415820596094, -48.654158205960854,
-48.65415820596088, -48.654158205961004, -48.65415820596095)
```

```
AA * ss
```

```
(-6.039613253960852e-14, -7.105427357601002e-15,
90.00000000000004,
-7.66053886991358e-15, 1.4210854715202004e-14,
89.99999999999999,
2.1316282072803006e-14, -7.105427357601002e-15, 90.0,
-4.618527782440651e-14, -3.5416114485542494e-14,
89.99999999999996,
6.394884621840902e-14, 3.552713678800501e-15, 90.00000000000006,
3.3306690738754696e-16, 6.394884621840902e-14,
90.00000000000003,
2.842170943040401e-14, 6.039613253960852e-14, 90.00000000000003,
-1.0658141036401503e-14, 4.340972026284362e-14,
90.00000000000006,
-2.1316282072803006e-14, -2.4868995751603507e-14,
79.99999999999997,
1.0658141036401503e-14, -7.105427357601002e-15, 80.0,
-1.7763568394002505e-14, 0.0, 80.0, 2.842170943040401e-14,
4.618527782440651e-14, 79.99999999999997, 4.973799150320701e-14,
7.105427357601002e-15, 80.00000000000003, 9.769962616701378e-15,
-7.105427357601002e-15, 80.0, 3.552713678800501e-15,
-1.4210854715202004e-14, 79.99999999999999, 0.0, 0.0,
79.99999999999996,
3.552713678800501e-15, 7.105427357601002e-15, 50.0,
-3.197442310920451e-14, 0.0, 50.0, 3.552713678800501e-14,
-1.509903313490213e-14, 50.000000000000014, 7.105427357601002e-
15,
-7.105427357601002e-15, 50.00000000000001, -7.105427357601002e-
15,
-1.4210854715202004e-14, 50.0, 1.0658141036401503e-14,
7.105427357601002e-15, 50.0, -7.105427357601002e-15,
8.881784197001252e-15, 50.00000000000001, -1.4210854715202004e-
14,
7.105427357601002e-15, 50.0)
```

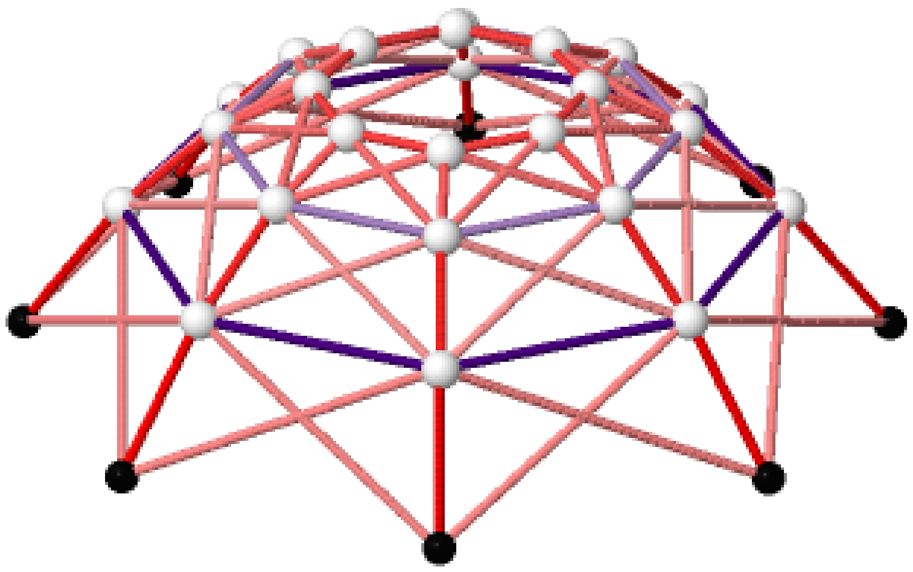
```
dict_ss = dict_bar_force (bars, ss)
print_dict (dict_ss)
```

```
0:   -236.486072048
1:   -236.486072048
2:   -236.486072048
3:   -236.486072048
4:   -236.486072048
5:   -236.486072048
6:   -236.486072048
7:   -236.486072048
8:   36.0272174622
9:   36.0272174622
10:  36.0272174622
11:  36.0272174622
12:  36.0272174622
13:  36.0272174622
14:  36.0272174622
15:  36.0272174622
16:  -45.3682709442
17:  -45.3682709442
18:  -45.3682709442
19:  -45.3682709442
20:  -45.3682709442
21:  -45.3682709442
22:  -45.3682709442
23:  -45.3682709442
24:  -45.3682709442
25:  -45.3682709442
26:  -45.3682709442
27:  -45.3682709442
28:  -45.3682709442
29:  -45.3682709442
30:  -45.3682709442
```

```
31:  -45.3682709442
32:  -164.714257024
33:  -164.714257024
34:  -164.714257024
35:  -164.714257024
36:  -164.714257024
37:  -164.714257024
38:  -164.714257024
39:  -164.714257024
40:  5.02044293286
41:  5.02044293286
42:  5.02044293286
43:  5.02044293286
44:  5.02044293286
45:  5.02044293286
46:  5.02044293286
47:  5.02044293286
48:  -32.6261538362
49:  -32.6261538362
50:  -32.6261538362
51:  -32.6261538362
52:  -32.6261538362
53:  -32.6261538362
54:  -32.6261538362
55:  -32.6261538362
56:  -32.6261538362
57:  -32.6261538362
58:  -32.6261538362
59:  -32.6261538362
60:  -32.6261538362
61:  -32.6261538362
62:  -32.6261538362
63:  -32.6261538362
64:  -59.34407864
65:  -59.34407864
66:  -59.34407864
67:  -59.34407864
68:  -59.34407864
69:  -59.34407864
70:  -59.34407864
71:  -59.34407864
72:  -89.8977061444
73:  -89.8977061444
74:  -89.8977061444
75:  -89.8977061444
76:  -89.8977061444
77:  -89.8977061444
78:  -89.8977061444
79:  -89.8977061444
80:  -48.654158206
81:  -48.654158206
```

```
82:  -48.654158206
83:  -48.654158206
84:  -48.654158206
85:  -48.654158206
86:  -48.654158206
87:  -48.654158206
88:  -48.654158206
89:  -48.654158206
90:  -48.654158206
91:  -48.654158206
92:  -48.654158206
93:  -48.654158206
94:  -48.654158206
95:  -48.654158206
```

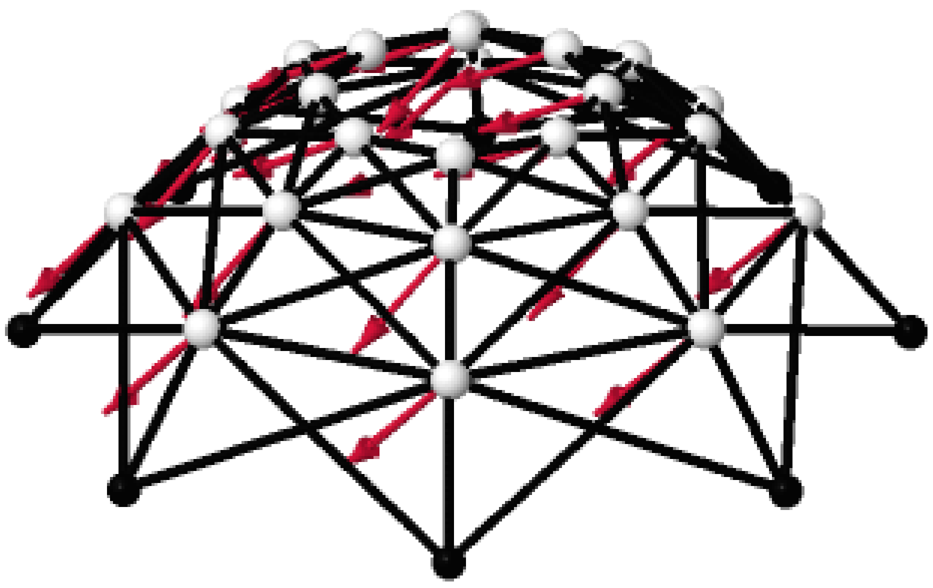
```
plot_truss_with_forces3d (joints, bars, supports, dict_ss)
```



Varijacija – zadano opterećenje u čvorovima nije vertikalno i odabran je drugi osnovni sistem

```
load_gen = [(8, 8, (-80.0, -30.0, -90.0)),
            (8, 16, (-40.0, -60.0, -80.0)),
            (8, 24, (-90.0, -40.0, -50.0))]
loads22 = make_loads (load_gen)
```

```
plot_truss3d (joints, bars, supports, loads22, load_scale = 0.03)
```



```
AAf22= augmented_equil_matrix (joints, bars, loads22, free_joints)
AAf22
```

```
72 x 97 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
AA22 = AAf22.subdivision (0, 0)
AA22
```

```
72 x 96 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
AAfr22, pivots = ref_wpp (AAf22)
AAfr22
```

```
72 x 97 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
pivots
```

```
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
 32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,
 53,54,55,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,
 82,83,84,85,86,87]
```

```
primary_system_bars = pivots
print primary_system_bars
print_list (list_indexed (primary_system_bars, bars), indent = '   ')
print
#
redundant_bars = others (primary_system_bars, all_indices (bars))
print redundant_bars
print_list (list_indexed (redundant_bars, bars), indent = '   ')
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19,20, 21, 22, 23, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45,46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 64, 65,
66, 67, 68, 69, 70, 71,72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87]
   (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)
(16, 24)
(17, 25)
(18, 26)
(19, 27)
(20, 28)
(21, 29)
(22, 30)
(23, 31)
(24, 25)
(25, 26)
(26, 27)
(27, 28)
(28, 29)
(29, 30)
(30, 31)
(31, 24)
(16, 25)
(17, 26)
(18, 27)
(19, 28)
```

```
   (20, 29)
   (21, 30)
   (22, 31)
   (23, 24)

[24, 25, 26, 27, 28, 29, 30, 31, 56, 57, 58, 59, 60, 61, 62, 63,
88, 89, 90, 91, 92, 93, 94, 95]
   (0, 15)
   (1, 8)
   (2, 9)
   (3, 10)
   (4, 11)
   (5, 12)
   (6, 13)
   (7, 14)
   (8, 23)
   (9, 16)
   (10, 17)
   (11, 18)
   (12, 19)
   (13, 20)
   (14, 21)
   (15, 22)
   (16, 31)
   (17, 24)
   (18, 25)
   (19, 26)
   (20, 27)
   (21, 28)
   (22, 29)
   (23, 30)
```

```
prim_bar_forces22 = ref_solution (AAfr22, primary_system_bars)
prim_bar_forces
```

```
(-281.76760906656517, -281.76760906656534, -281.7676090665652,
-281.76760906656534, -281.76760906656517, -281.76760906656557,
-281.767609066565, -281.7676090665657, -32.01047864505287,
-32.01047864505281, -32.010478645052864, -32.0104786450527,
-32.010478645052935, -32.010478645052764, -32.010478645053055,
-32.01047864505274, 2.3155759809037474e-14, -2.842170943040401e-
14,
4.752613235570464e-14, -1.5099033134902131e-13,
6.765843308643685e-14,
-1.2434497875801756e-13, 2.265549618901036e-13, -
9.237055564881302e-14,
-199.6670834718586, -199.66708347185866, -199.66708347185863,
-199.6670834718587, -199.66708347185863, -199.66708347185872,
-199.66708347185852, -199.6670834718588, -57.85734423920378,
-57.85734423920373, -57.857344239203755, -57.857344239203705,
-57.85734423920382, -57.85734423920363, -57.8573442392039,
-57.85734423920362, 3.380307101168053e-14, -3.059598761762708e-
14,
3.6168136954704156e-14, -1.0627312656085139e-13,
1.2304806640629603e-13,
-1.8318658920346015e-13, 2.3445183221310427e-13,
-1.5631940186722204e-13, -118.34728240234361, -
118.34728240234365,
-118.34728240234365, -118.34728240234367, -118.34728240234361,
-118.3472824023437, -118.34728240234364, -118.34728240234375,
-140.15029759509366, -140.15029759509366, -140.1502975950937,
-140.15029759509366, -140.1502975950937, -140.15029759509366,
-140.15029759509375, -140.15029759509363, 0.0, -
1.7083870736439967e-14,
2.1482228976552236e-14, -3.8566099712992266e-14,
3.8566099712992266e-14,
-3.8566099712992285e-14, 1.3498134899547292e-13, -
1.928304985649612e-14)
```

```
ss022 = all_bar_forces (primary_system_bars, redundant_bars,
prim_bar_forces22)
ss022
```

```
(-0.11891922598309179, -110.52425854845121, -321.24163013822044,
-508.835655545616, -563.4162989071474, -453.0109595846794,
-242.29358799491004, -54.69956258751503, -105.94956869929044,
-297.8528046054388, -334.0292114168027, -193.28714066121057,
41.92861140918471, 233.83184731533285, 270.0082541266966,
129.266183371105, -61.35602972143781, 241.20362094674522,
402.46946175782455, 327.9741503121695, 61.35602972143761,
-241.20362094674516, -402.46946175782415, -327.97415031216957,
0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -13.739206932119245, -
101.61287839801471,
-246.92537334845522, -364.5546030237447, -385.59496001159806,
-297.7212885457026, -152.40879359526213, -34.779563919972844,
-118.40091011908595, -267.6855871697918, -294.0557252806759,
-182.06405518803354, 2.686221640678326, 151.9708986913843,
178.3410368022682, 66.34936670962612, -82.98432208859985,
131.76967030291507, 269.33477694041414, 249.12722406495035,
82.98432208859994, -131.7696703029152, -269.3347769404138,
-249.1272240649505, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
-34.51810386632033, -115.39589781111002, -198.00257282166544,
-233.94825901934976, -202.17646093836692, -121.29866699357727,
-38.69199198302188, -2.7463057853376007, -236.09312042811968,
-279.3888084398154, -241.1204652033065, -143.70516717699098,
-44.207474762067676, -0.911786750371931, -39.1801299868809,
-136.59542801319625, -4.8674351110420275, 131.36781919878683,
190.64958668134318, 138.2514119467935, 4.867435111042056,
-131.36781919878686, -190.64958668134298, -138.25141194679358,
0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
```

```
AA22 * ss022
```

```
(79.99999999999999, 30.00000000000002, 90.0, 79.99999999999996,
30.000000000000014, 90.0, 79.99999999999997, 30.000000000000114,
90.0,
80.00000000000003, 30.0, 89.99999999999997, 79.99999999999997,
29.99999999999997, 90.00000000000003, 79.99999999999983,
29.99999999999998, 90.00000000000006, 80.00000000000003,
29.999999999999943, 90.00000000000004, 80.00000000000006, 30.0,
90.00000000000004, 39.99999999999999, 59.999999999999986,
80.00000000000001, 40.0, 59.999999999999986, 80.0,
40.000000000000014,
60.0, 80.00000000000006, 40.00000000000011, 60.000000000000085,
80.0,
40.00000000000006, 59.99999999999999, 80.0, 39.99999999999997,
60.000000000000014, 80.0, 40.00000000000003, 60.00000000000003,
80.00000000000001, 40.0, 60.00000000000003, 80.0, 90.0, 40.0,
49.99999999999999, 90.00000000000001, 40.00000000000003,
49.99999999999997, 90.0, 40.0, 50.00000000000001,
89.99999999999997,
40.0, 49.99999999999998, 89.99999999999997, 39.99999999999997,
50.0,
90.0, 40.00000000000002, 50.00000000000001, 89.99999999999999,
39.999999999999986, 50.0, 89.99999999999999, 40.000000000000014,
50.00000000000001)
```

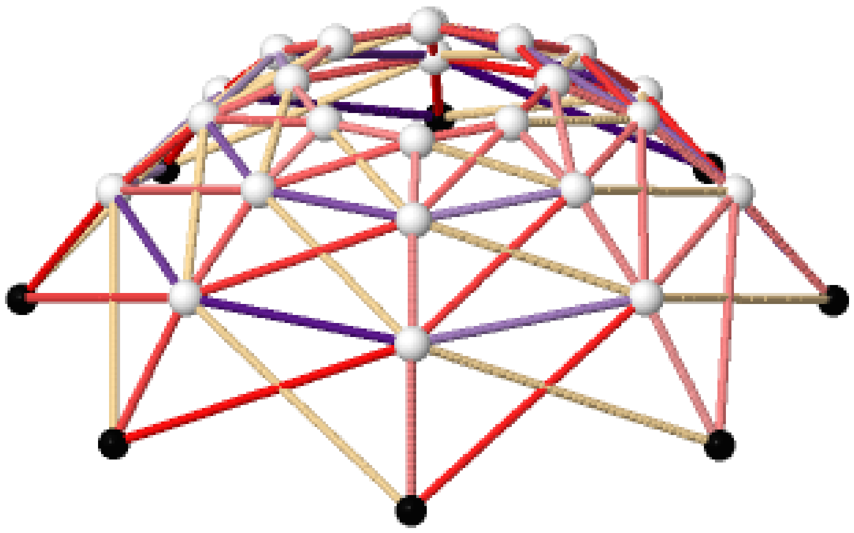
```
dict_ss022 = dict_bar_force (bars, ss022)
print_dict (dict_ss022)
```

```
0:   -0.118919225983
1:   -110.524258548
2:   -321.241630138
3:   -508.835655546
4:   -563.416298907
5:   -453.010959585
6:   -242.293587995
7:   -54.6995625875
8:   -105.949568699
9:   -297.852804605
10:  -334.029211417
```

```
11:  -193.287140661
12:  41.9286114092
13:  233.831847315
14:  270.008254127
15:  129.266183371
16:  -61.3560297214
17:  241.203620947
18:  402.469461758
19:  327.974150312
20:  61.3560297214
21:  -241.203620947
22:  -402.469461758
23:  -327.974150312
24:  0.0
25:  0.0
26:  0.0
27:  0.0
28:  0.0
29:  0.0
30:  0.0
31:  0.0
32:  -13.7392069321
33:  -101.612878398
34:  -246.925373348
35:  -364.554603024
36:  -385.594960012
37:  -297.721288546
38:  -152.408793595
39:  -34.77956392
40:  -118.400910119
41:  -267.68558717
42:  -294.055725281
43:  -182.064055188
44:  2.68622164068
45:  151.970898691
46:  178.341036802
47:  66.3493667096
48:  -82.9843220886
49:  131.769670303
50:  269.33477694
51:  249.127224065
52:  82.9843220886
53:  -131.769670303
54:  -269.33477694
55:  -249.127224065
56:  0.0
57:  0.0
58:  0.0
59:  0.0
60:  0.0
61:  0.0
```

```
62:  0.0
63:  0.0
64:  -34.5181038663
65:  -115.395897811
66:  -198.002572822
67:  -233.948259019
68:  -202.176460938
69:  -121.298666994
70:  -38.691991983
71:  -2.74630578534
72:  -236.093120428
73:  -279.38880844
74:  -241.120465203
75:  -143.705167177
76:  -44.2074747621
77:  -0.911786750372
78:  -39.1801299869
79:  -136.595428013
80:  -4.86743511104
81:  131.367819199
82:  190.649586681
83:  138.251411947
84:  4.86743511104
85:  -131.367819199
86:  -190.649586681
87:  -138.251411947
88:  0.0
89:  0.0
90:  0.0
91:  0.0
92:  0.0
93:  0.0
94:  0.0
95:  0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_ss022)
```



```
AAr22 = AAfr22.subdivision (0, 0)
```

```
SST22 = matrix (kernel (AAr22, pivots))
SS22 = SST.T
SS22
```

```
96 x 24 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

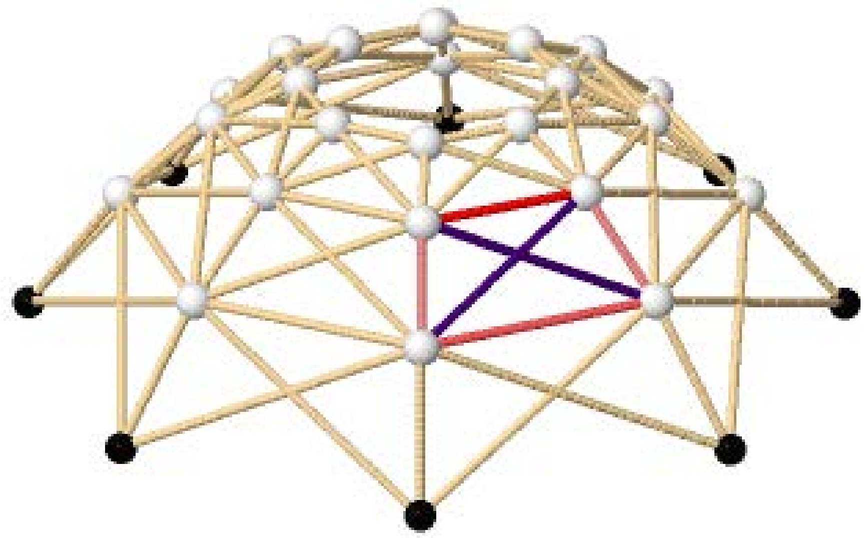
```
SS022 = SS22.column(8)
SS122 = SS22.column(9)
```

```
dict_SS022 = dict_bar_force (bars, SS022)
print_dict (dict_SS022)
```

```
0:   3.88578058619e-16
1:   -2.24504923676e-16
2:   1.47804175225e-17
3:   9.70195206167e-18
4:   -4.51013542211e-17
5:   -2.45334204944e-17
6:   2.35561280869e-17
7:   -2.21938480095e-16
8:   -2.11955094882e-16
```

```
9:   3.61805144111e-17
10:  -2.11981729676e-18
11:  -5.1270981481e-17
12:  -4.56867737247e-18
13:  5.82814234553e-17
14:  -5.55111512313e-17
15:  -0.703063347534
16:  2.49800180541e-16
17:  -4.16333634234e-17
18:  -2.08166817117e-17
19:  5.55111512313e-17
20:  2.77555756156e-17
21:  -5.55111512313e-17
22:  1.11022302463e-16
23:  -4.4408920985e-16
24:  0.0
25:  0.0
26:  0.0
27:  0.0
28:  0.0
29:  0.0
30:  0.0
31:  0.0
32:  -0.53565655675
33:  -1.18939648519e-16
34:  -4.83956917228e-18
35:  2.35144610397e-18
36:  -2.73256001468e-17
37:  -6.46076028674e-18
38:  -4.96269906846e-17
39:  -0.53565655675
40:  -2.26897101667e-16
41:  -4.08364860324e-18
42:  -5.58642387928e-18
43:  -2.42357011059e-17
44:  -3.024680221e-17
45:  -4.3488949023e-17
46:  -5.55111512313e-17
47:  -1.01423585188
48:  2.22044604925e-16
49:  -1.48168226675e-18
50:  -4.38983911302e-18
51:  -5.92672906701e-18
52:  1.20613856123e-17
53:  -1.1853458134e-17
54:  8.33642553535e-17
55:  1.0
56:  1.0
57-95:  0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_SS022)
```

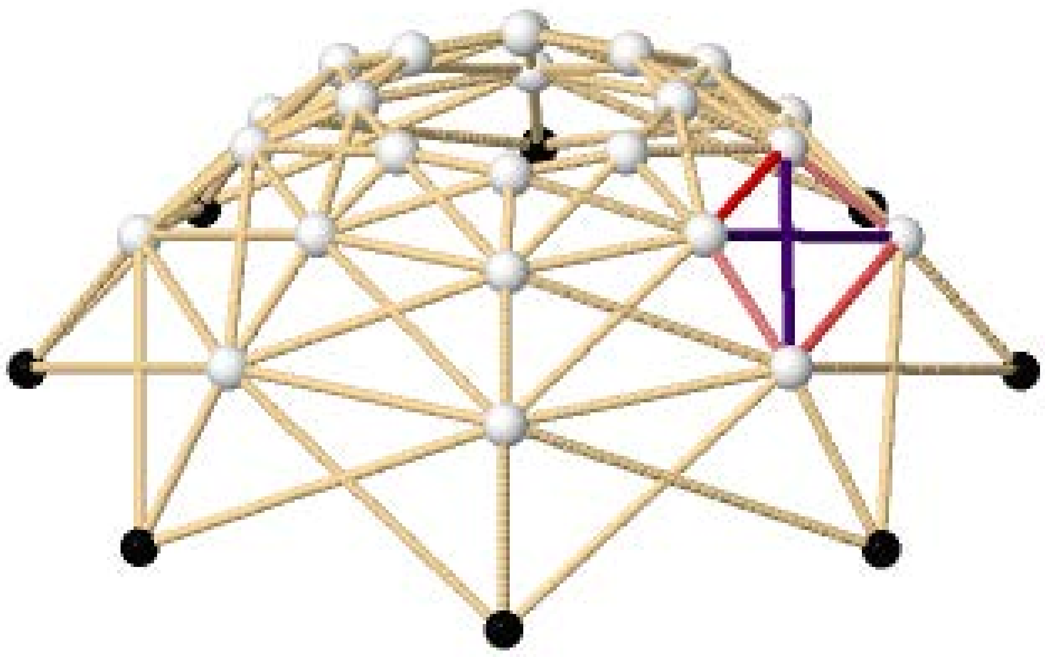


```
dict_SS122 = dict_bar_force (bars, SS122)
print_dict (dict_SS122)
```

```
0:   1.74283750874e-16
1:   -1.11022302463e-16
2:   1.05437946858e-16
3:   -9.27331332671e-18
4:   4.4567177642e-19
5:   -7.1009609044e-20
6:   -6.92562830015e-18
7:   -7.76837431404e-17
8:   -0.703063347534
9:   6.34969657634e-17
10:  -1.92763676162e-17
11:  -1.94180948265e-18
12:  -2.82956861267e-18
13:  -6.4896969841e-18
14:  7.30585857917e-18
15:  1.11022302463e-16
16:  1.11022302463e-16
17:  -8.32667268469e-17
18:  1.85821515458e-17
19:  -8.93050865141e-19
20:  1.42291246934e-19
21:  1.38777878078e-17
22:  6.43285457969e-18
23:  -1.55302847711e-17
24:  0.0
25:  0.0
26:  0.0
```

```
27:  0.0
28:  0.0
29:  0.0
30:  0.0
31:  0.0
32:  -0.53565655675
33:  -0.53565655675
34:  7.66104548181e-17
35:  7.20171956576e-33
36:  -1.41073336848e-33
37:  2.98171293936e-33
38:  -1.20295069921e-32
39:  -3.1932867591e-33
40:  -1.01423585188
41:  1.45057628673e-16
42:  1.30989655062e-32
43:  -1.29664833365e-33
44:  -1.51157764971e-33
45:  -1.72792145322e-32
46:  -6.04631059883e-33
47:  0.0
48:  1.0
49:  -1.43021594439e-16
50:  -1.34446586624e-32
51:  -2.11912560243e-34
52:  -5.56646399971e-33
53:  1.10752384789e-32
54:  5.96144435994e-33
55:  -1.66728510707e-16
56:  0.0
57:  1.0
58-95:  0.0
```

```
plot_truss_with_forces3d (joints, bars, supports, dict_SS122)
```



```
E = 2.e8
Fs = dict_same_value (bars, 0.05*0.05)
print E
print
print_dict (Fs)
```

```
2.00000000000000e8

0 - 95:   0.00250000000000000
```

```
ddelta = diag_flex (bars, joints, E, Fs)
ddelta
```

```
96 x 96 sparse matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
DD22 = SST22 * ddelta * SS22
DD22
```

```
24 x 24 dense matrix over Real Double Field (use the '.str()'
method to see the entries)
```

```
dd022 = SST22 * ddelta * ss022
dd022
```

```
(-0.00633489864016572l, 0.000704676928258184l,
0.00881425263881475l,
0.0132433490253514l, 0.0113974614936926O9, 0.00435788592526870 7,
-0.0037516897852878 59, -0.008180786171824503, -
0.004297473545932 9,
0.0012626714607223875, 0.007283513922030032,
0.010238125981067844,
0.008395735965202529, 0.0028355909585472 46, -0.0031852515027604,
-0.006139863561798194, -0.0006255989816569684,
0.002142359508068103,
0.004625955946841101, 0.005370333224290334,
0.003939445226808381,
0.0011714867370833033, -0.00131210970168969 4, -
0.0020564869791389 25)
```

```
xx22 = DD22 \ (-dd022)
xx22
```

```
(117.91444528698324, -10.26051020492249, -159.00111579332835,
-241.17714200013373, -208.65098717531873, -80.47603168341277,
68.26457390499291, 150.44060011179812, 77.29669581246935,
-18.798523076621375, -122.99378052916671, -174.25290786442199,
-142.54900348480388, -46.453784595713124, 57.74147285683218,
109.00060019208729, 9.71016326041193, -67.34381735206195,
-133.44964911288005, -149.88343232850966, -107.01847967233368,
-29.964499059859758, 36.14133270095829, 52.57511591658786)
```

```
ss22 = ss022 + SS22*xx22
ss22
```

```
(-53.84298171719866, -26.05524062018688, -121.53502663196107,
-284.35157603930463, -419.1291623781853, -446.9169034751969,
-351.4371174634226, -188.6205680560793, -82.53330415056658,
-53.32142413242113, 28.229678352228007, 114.34847349435137,
154.5877390749235, 125.37585905677791, 43.824756572129075,
-42.29403856999437, -71.61653992636037, 82.20250515341701,
161.29231975769068, 119.32316313685101, -19.120001961975163,
-172.9390470417523, -252.02886164602603, -210.05970502518625,
117.91444528698324, -10.26051020492249, -159.00111579332835,
-241.17714200013373, -208.65098717531873, -80.47603168341277,
68.26457390499291, 150.44060011179812, -45.07413671597506,
-25.660901274876068, -87.70323573818123, -194.8575820185738,
-284.3543773322231, -303.7676127733219, -241.72527831001685,
-134.57093202962434, -58.10537124488581, -61.24000246521163,
-25.56016342022957, 28.03338008079851, 68.14625711060332,
71.28088833092923, 35.60104928594723, -17.992494215080896,
-101.7828451652213, 8.775889773748474, 95.08186907599202,
106.57822058014668, 36.53053749288677, -74.02819744608306,
-160.33417674832646, -171.83052825248114, 77.29669581246935,
-18.798523076621375, -122.99378052916671, -174.25290786442199,
-142.54900348480388, -46.453784595713124, 57.74147285683218,
109.00060019208729, 0.4282435922626604, 6.355852222622843,
-26.202667599339435, -78.17497653271167, -119.11640087216912,
-125.04400950252929, -92.4854896805671, -40.513180747194866,
-166.5368591565561, -141.55494298238366, -86.31291806325223,
-33.17081338383704, -13.258553132288533, -38.240469306460945,
-93.48249422559238, -146.62459890500745, -72.21125246310403,
-2.0818299140932197, 40.766154352833524, 31.232932274459827,
-25.097063948817716, -95.22648649782856, -138.07447076475512,
-128.54124868638166, 9.71016326041193, -67.34381735206195,
-133.44964911288005, -149.88343232850966, -107.01847967233368,
-29.964499059859758, 36.14133270095829, 52.57511591658786)
```

```
AA22 * ss22
```

```
(80.0, 29.999999999999986, 90.0, 79.99999999999991,
30.000000000000036,
90.0, 79.99999999999993, 30.00000000000007, 89.99999999999997,
79.99999999999994, 30.000000000000107, 89.99999999999993, 80.0,
30.000000000000043, 89.99999999999991, 79.99999999999984,
29.99999999999997, 90.00000000000003, 80.00000000000004,
29.999999999999915, 90.0, 79.99999999999997, 30.00000000000004,
90.00000000000009, 40.0, 59.99999999999992, 80.0,
39.99999999999999,
60.0, 80.0, 39.999999999999986, 59.999999999999964,
80.00000000000009,
40.00000000000014, 60.000000000000114, 80.00000000000003,
40.00000000000007, 60.000000000000014, 80.00000000000003,
40.00000000000001, 59.999999999999986, 79.99999999999997,
40.00000000000003, 60.00000000000005, 80.00000000000003,
40.000000000000014, 60.000000000000014, 79.99999999999999,
89.99999999999997, 40.000000000000014, 50.0, 90.00000000000001,
40.00000000000004, 49.99999999999997, 89.99999999999997,
39.999999999999964, 50.000000000000036, 90.0,
40.000000000000036,
49.99999999999998, 89.99999999999999, 39.99999999999999,
49.99999999999999, 90.00000000000001, 39.99999999999999,
50.000000000000014, 89.99999999999997, 40.000000000000014,
50.00000000000001, 89.99999999999999, 40.00000000000001,
50.00000000000001)
```

```
dict_ss22 = dict_bar_force (bars, ss22)
print_dict (dict_ss22)
```

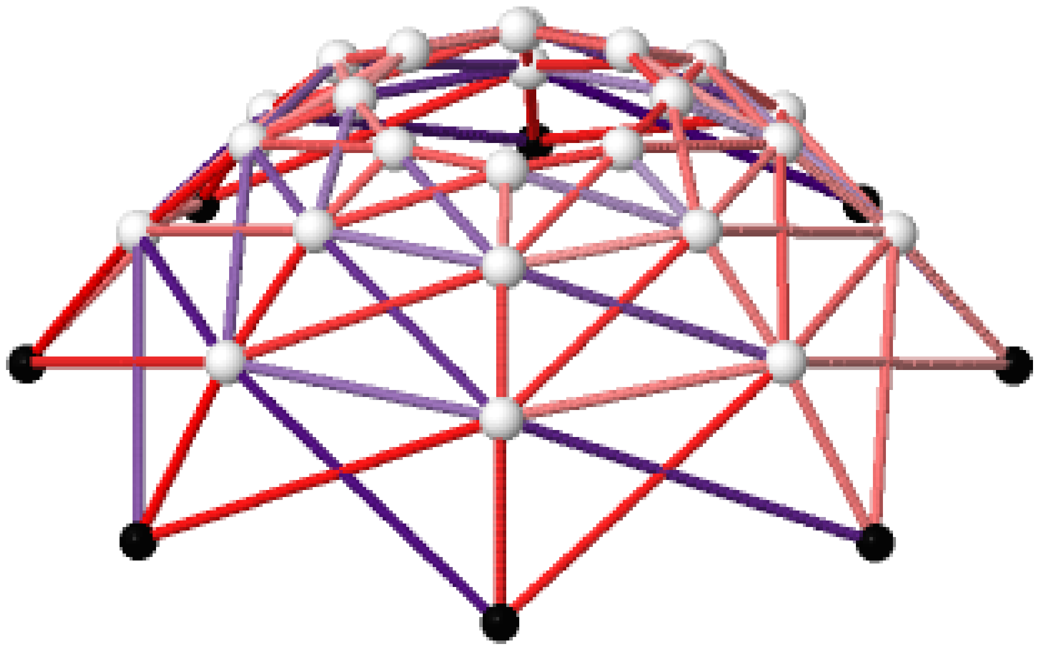
```
0:   -53.8429817172
1:   -26.0552406202
2:   -121.535026632
3:   -284.351576039
4:   -419.129162378
5:   -446.916903475
6:   -351.437117463
7:   -188.620568056
8:   -82.5333041506
9:   -53.3214241324
10:  28.2296783522
11:  114.348473494
12:  154.587739075
13:  125.375859057
14:  43.8247565721
15:  -42.29403857
16:  -71.6165399264
17:  82.2025051534
18:  161.292319758
19:  119.323163137
20:  -19.120001962
21:  -172.939047042
22:  -252.028861646
23:  -210.059705025
24:  117.914445287
25:  -10.2605102049
26:  -159.001115793
27:  -241.177142
28:  -208.650987175

29:  -80.4760316834
30:  68.264573905
31:  150.440600112
32:  -45.074136716
33:  -25.6609012749
34:  -87.7032357382
35:  -194.857582019
36:  -284.354377332
37:  -303.767612773
38:  -241.72527831
39:  -134.57093203
40:  -58.1053712449
41:  -61.2400024652
42:  -25.5601634202
43:  28.0333800808
```

```
44:  68.1462571106
45:  71.2808883309
46:  35.6010492859
47:  -17.9924942151
48:  -101.782845165
49:  8.77588977375
50:  95.081869076
51:  106.57822058
52:  36.5305374929
53:  -74.0281974461
54:  -160.334176748
55:  -171.830528252
56:  77.2966958125
57:  -18.7985230766
58:  -122.993780529
59:  -174.252907864
60:  -142.549003485
61:  -46.4537845957
62:  57.7414728568
63:  109.000600192
64:  0.428243592263
65:  6.35585222262
66:  -26.2026675993
67:  -78.1749765327
68:  -119.116400872
69:  -125.044009503
70:  -92.4854896806
71:  -40.5131807472
72:  -166.536859157
73:  -141.554942982
74:  -86.3129180633
75:  -33.1708133838
76:  -13.2585531323
77:  -38.2404693065
78:  -93.4824942256
79:  -146.624598905
80:  -72.2112524631
81:  -2.08182991409
82:  40.7661543528
83:  31.2329322745
84:  -25.0970639488
85:  -95.2264864978
86:  -138.074470765
87:  -128.541248686
88:  9.71016326041
89:  -67.3438173521
90:  -133.449649113
91:  -149.883432329
92:  -107.018479672
93:  -29.9644990599
94:  36.141332701
```

95: 52.5751159166

```
plot_truss_with_forces3d (joints, bars, supports, dict_ss22)
```



6. Zaključak

U radu smo se bavili analizom statički neodređenih rešetkastih sistemima i rješavali ih pomoću programskog paketa *Sage-a* postupkom metode sila.

Na primjerima smo vidjeli kako je jednostavnim slijedom naredbi moguće riješiti velike ravninske ili prostorne sisteme te se slikovnim prikazima mogu prikazati rješenja pojedinih koraka postupka metode sila i konačnog rješenja sustava s pripadnim silama u štapovima.

Uvjerili smo se da je pomoću računalnih programa moguće računati statiku složenih sistema čije ravnotežne matrice sadrže mnogo jednadžbi koje bi bilo nemoguće računati ručno.

7. Literatura

[1] Gilbert Strang: *Linear Algebra and Its Applications (4th Edition)*: Brooks/Cole, 2004

[2] Fresl K : Mehanika 1. : Bilješke i skice s predavanja, dostupno na mrežnoj stranici predmeta

[3] Fresl K; Simović V.: Građevna statika 1. :Bilješke i skice s predavanja, dostupno na mrežnoj stranici predmeta

[4] Fresl K; Simović V.: Građevna statika 2. :Bilješke i skice s predavanja, dostupno na mrežnoj stranici predmeta

[5] Franjčić, E: Klasifikacija sklopova zglobnih štapova: završni rad , Građevinski fakultet, Zagreb, 2015.

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