Sveučilište u Zagrebu
Građevinski fakultet

Nenad Bijelić

Kompjutorska grafostatika

Zagreb, 2008.

Ovaj rad izrađen je na Katedri za statiku, dinamiku i stabilnost konstrukcija u Zavodu za tehničku mehaniku Građevinskog fakulteta Sveučilišta u Zagrebu pod vodstvom prof. dr. sc. Krešimira Fresla, dipl. ing. građ., i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2007./2008.

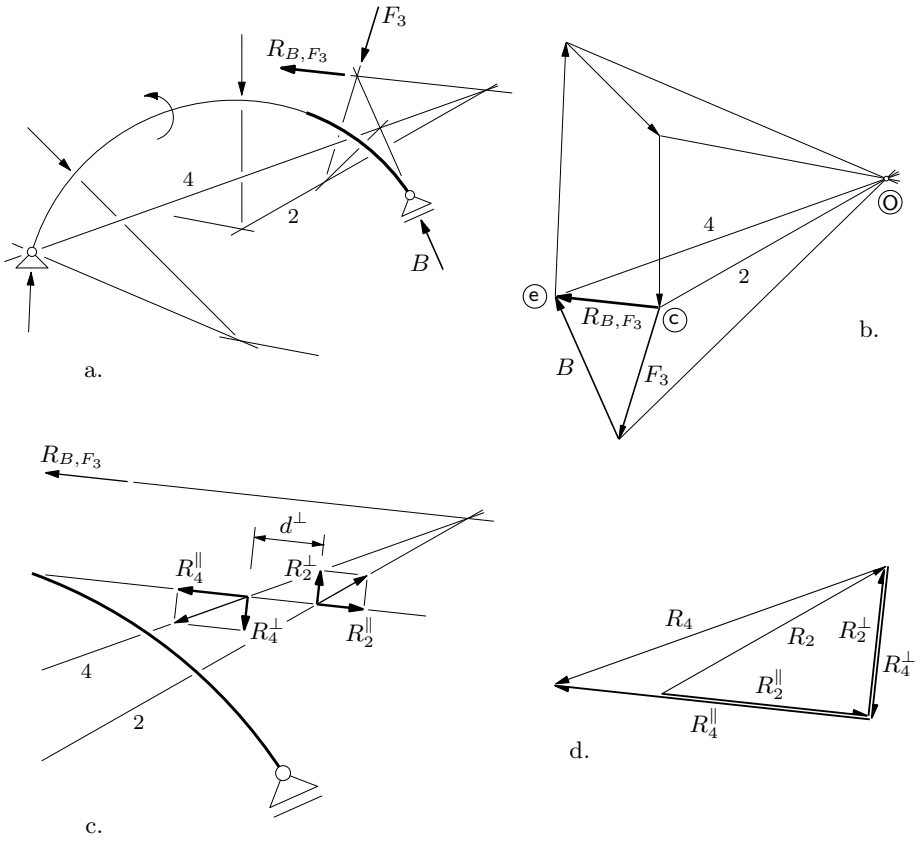
Sadržaj

1. Uvod

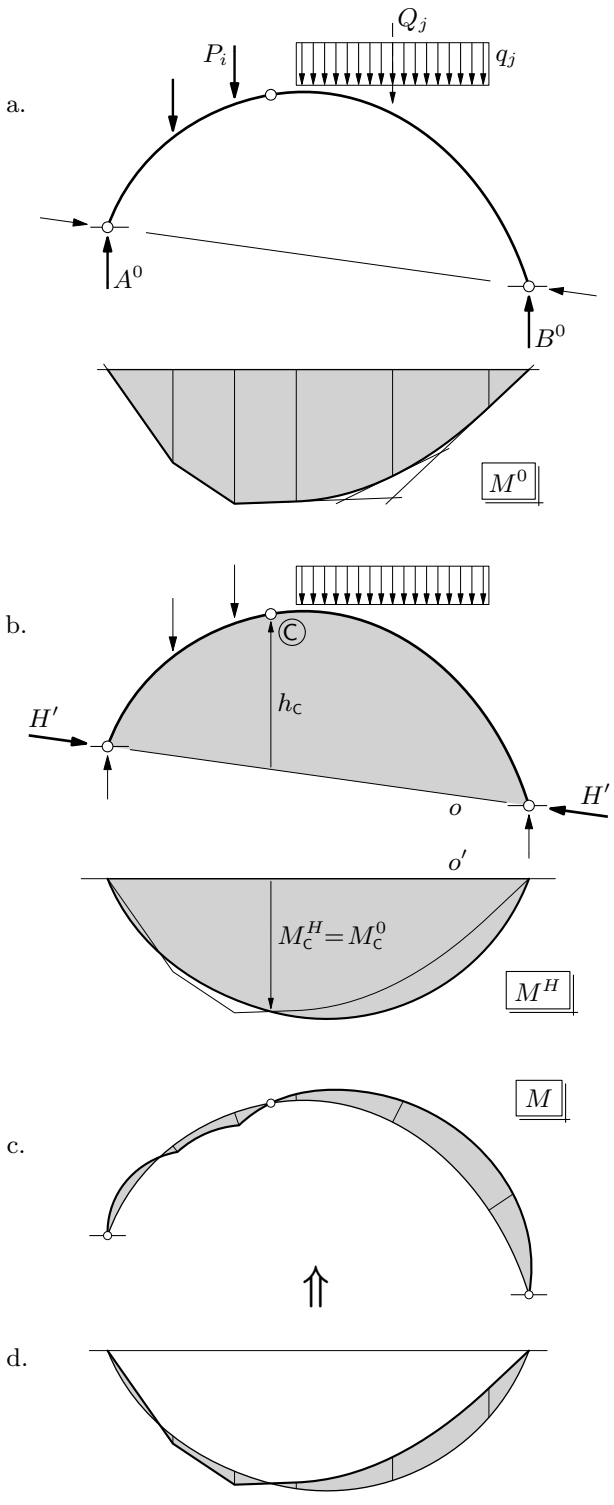
Predmet ovoga rada su grafički postupci u statici, te njihova realizacija na računalu. Izrađen je računalni program nazvan GS koji određene statičke probleme rješava ‚grafički', što znači da svi algoritmi slijede korake grafičkih postupaka (primjerice, slike 1. i 2.), a da su samo elementarne radnje, poput povlačenja pravca kroz dvije točke ili nalaženja sjecišta dvaju pravaca, realizirane algebarski/numerički. Naravno, i prikaz je rješenja grafički, uz mogućnost numeričkoga prikaza rezultata.

Program GS, kodiran u programskom jeziku `Asymptote`, oblikovan je prema načelima objektno usmjerene programske paradigme. Posljedično tome, program je otvoren daljnjim proširenjima i dopunama uz minimalno zadiranje u izrađeni kôd koji će biti prikazan, opisan i obrazložen u nastavku.

U okviru ovog rada prikazat će se elementi i mogućnosti kompjutorskog programa GS, izrađenoga u potpunosti ‚od nule', s primjerima i obrazloženjima, te uz prikaz fragmenata programskog kôda potrebnih za razumijevanje i uporabu. Odjeljak 4. tako sadrži



Slika 1. Određivanje momenta savijanja u presjeku pomoću verižnog poligona (iz [5])



Slika 2. Crtanje momentnog dijagrama na trozglobnom luku primjenom principa superpozicije (iz [5])

geometrijske i statičke elemente grafičkih zahvata. U odjeljku 5. obrađeni su temeljni postupci — poligon sila i verižni poligon — kao osnova za grafičko rješavanje statičkih problema. Konačno, odjeljkom 6. obuhvaćeno je grafičko rješavanje odabranih tipova statički određenih sustava. U odjeljku 7. dani su primjeri s pripadnim kodom.

U sljedećem su odjeljku ukratko prikazani osnovni grafički postupci i zahvati statike u obliku u kojem su realizirani u samom programu. Dan je i sažeti opis vektorskog grafičkog programskog jezika `Asymptote` (odjeljak 3.) kojim je oblikovan program GS, te programskih jezika potrebnih za razumijevanje i uporabu programa.

2. Grafostatika

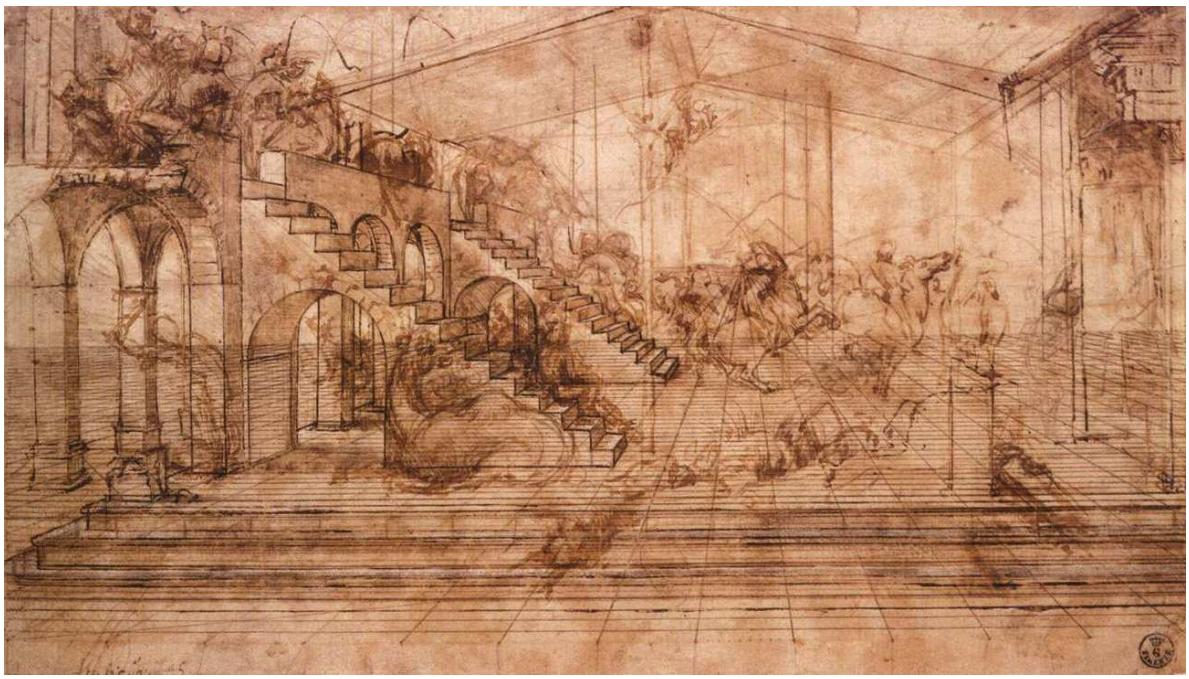
2.1. O grafičkim postupcima

Postupci koji se primjenjuju pri rješavanju problema građevne statike dijele se na geometrijske, analitičke, numeričke i kombinirane ili grafoanalitičke [17, 5]. Geometrijski postupci mogu biti modelski i grafički.

U današnjem se računalnom dobu grafički postupci često smatraju zastarjelima: to je bio vrijedan sadržaj knjiga iz devetnaestog i s početka dvadesetoga stoljeća [12, 13, 14]. Neosporni su nedostatci grafičkih postupaka to što ne daju opća rješenja i to što su primjenjivi gotovo isključivo u rješavanju statički određenih zadataka. No, njihove su prednosti preglednost (pogreške se mogu lako uočiti) te razmjerna jednostavnost i brzina rješavanja uz točnost dovoljnu za praktične primjene ili za laku i brzu provjeru rezultata dobivenih drugim, točnijim postupcima. Nadalje, primjenom grafičkih postupaka najlakše se može predočiti i najslikovitije objasniti (sebi i drugima) tok sila u konstrukciji, što je ključno za dobro poznavanje i razumijevanje ponašanja konstrukcija [17, 5]. Naime, to da se danas na računalu sve ‚može izračunati' ne znači da inženjeri mogu i smiju projektirati bez poznavanja i razumijevanja načina na koje konstrukcijski elementi preuzimaju i prenose sile, jer proračun konstrukcije mora biti samo provjera zamišljene konstruktorske koncepcije.

Primjena grafičkih postupaka u mehanici ima dugu i bogatu povijest [3]. Oni nisu imali samo ulogu alata za rješavanje praktičnih problema, nego su tijekom razvitka mehanike ponajprije bili sredstvom otkrivanja novih spoznaja. Primjerice, u povijesti je statike i dinamike Leonardo da Vinci značajan zato što je u ponajčešće neodređenim i nedorađenim skicama, razbacanim po njegovim bilježnicama, ‚naslutio' zakone i principe koje su kasnije jasno izrazili Galileo Galilei, Stevin, Newton i Lagrange — princip inercije, zakon akcije i reakcije, zakon paralelograma brzina i paralelograma sila, zakon poluge i princip virtualnih pomaka [3, 4] — ali je možda čak i važnija njegova temeljna teorijska zamisao, koju je Galileo neposredno preuzeo i razrađivao — zamisao postojanja prirodnih zakona, koji se mogu spoznati jedino mišlju, razumom [1]. Obojica su se suprostavljala skolastičkim nasljednicima starogrčke prirodne filozofije za koje je ta filozofija tek knjiga, proizvod mašte. Za našu je temu, međutim, posebno zanimljivija metodološka razlika u pristupima Leonarda i Galileja, istaknuta u [1]. Odvajajući objektivnu istinu prirode od svijeta mašte i bajki, Galileo je i poeziju i likovne umjetnosti smještao u svijet mašte. Za Leonarda pak djela likovne umjetnosti nisu bila samo proizvodi mašte; likovna je umjetnost nepatvoreno i neophodno sredstvo razumijevanja stvarnosti, čija istina nije manje vrijedna od istine znanosti. „Znanost je drugo stvaranje razumijevanjem, slikarstvo je drugo stvaranjem maštom". Ali vrijednost tih stvaranja nije u odvajanju od prirode i od empirijske istine, nego u njezinu dohvaćanju i otkrivanju. I kao umjetnik i kao znanstvenik, Leonardo se bavio ‚svijetom vida', smatrajući mogućnost predodžbe granicom spoznaje. Viđena kroz umjetnost, priroda više nije kaotična i bezoblična; slikarstvo viđenom daje oblik i proporciju, pa umjetnička ‚vizija'

otvara put znanstvenoj apstrakciji. Proporcija ne postoji samo između brojeva i glazbenih tonova, nego i između prirodnih oblika. Zahvaljujući toj unutarnjoj harmoniji, priroda više nije suprostavljena ljudskoj misli: iako je priroda neiscrpna i beskrajna, to više nije nepojmljivi beskraj, već beskraj geometrije i matematike (slika 3.).



Slika 3. Leonardova studija za sliku „Poklonstvo kraljeva"

Iz svega rečenoga proizlazi da grafički postupci imaju nezaobilaznu i neprijepornu obrazovnu i spoznajnu vrijednost. No, kada je riječ o grafičkim postupcima i, posebice, o njihovu didaktičkom značenju i značaju, treba istaknuti da nije važan samo gotov crtež, nego da je jednako važan, pa i važniji, proces nastajanja crteža, jer često upravo taj proces iskazuje fizikalnu interpraciju grafičkog rješenja. U tom slijedu koraka u kojem crtež postupno nastaje tražit ćemo moguću primjenu kompjutorske grafike i kompjutorske animacije.

2.2. Postupak s dva plana

Grafički se postupak najčešće, preglednosti radi, provodi s pomoću dva povezana crteža — planom položaja i planom sila [17, 5, 20]. Na *planu položaja* prikazan je dio konstrukcije koji se analizira, te poopćene sile s hvatištima ili pravcima djelovanja te pretpostavljenim smislom djelovanja. Konstrukcija i položaj sila prikazuje se u određenom mjerilu (*mjerilo duljina*), a sile su samo označene i duljine im ne moraju biti u nekom mjerilu. Na

planu sila konstruira se rješenje i na njemu su sile ucrtane u mjerilu (*mjerilo sila*), usporedno s pripadnim pravcima u planu položaja. Program GS omogućava zadavanje mjerila sila i mjerila duljina po volji. Prešutno se podrazumijeva prikazivanje sila tehničkom notacijom — sile se prikazuju strelicama, a apsolutni iznos veličine sile proporcionalan je duljini sile, odnosno, njezine strelice. Koncentrirani se momenti prikazuju načinom uobičajenim za ravninske probleme i tehničku notaciju.

Osnovni postupci statike, čiji sklop čini alat za rješavanje složenijih problema, a koje podržava program jesu [20]:

- rastavljanje sila na komponente i rastavljanje momenata na spreg sila,
- konstrukcija poligona sila za zadani niz sila i momenata,
- uravnoteženje sustava sila i momenata,
- nalaženje statički ekvivalentnog djelovanja te
- općeniti verižni poligon sa svim mogućnostima primjene.

Posebno se ističe važnost mogućnosti kritičke interpretacije rezultata grafičkih operacija.

2.3. Jednadžbe ravnoteže i poopćene sile

U trenutnom je obliku program GS namijenjen rješavanju ravninskih problema statike statički određenih štapnih konstrukcija. Pritom se barata s *poopćenim* ili *generaliziranim* silama. Program podržava *koncentriranu silu*, *koncentrirani moment*, te opće *distribuirano* trapezno *opterećenje* koje obuhvaća trapezno, trokutasto i jednoliko raspodijeljeno opterećenje. Njihova je programska realizacija detaljnije opisana u odjeljku 4. Iz poznatih su razloga distribuirani momenti izostavljeni iz prikaza i zasad nisu podržani, ali njihovo naknadno uključivanje ne bi predstavljalo pretežak programski zadatak.

Ishodište za rješavanje problema statike poznate su jednadžbe ravnoteže koje u općoj *vektorskoj formulaciji* glase [19, 20]:

$$\sum_i \vec{F}_i \; + \sum_j \int \vec{q}_j(s)\,\mathrm{d}s \;=\; \vec{0},$$

$$\sum_i \vec{r}_i \times \vec{F}_i \; + \sum_j \int \vec{r}_j(s) \times \vec{q}_j(s)\,\mathrm{d}s \; + \sum_k M_k \; + \sum_\ell \int \vec{m}_\ell(s)\,\mathrm{d}s \;=\; \vec{0}$$

(no, rekli smo već da ćemo zasad zanemariti distribuirane momente m_ℓ).

Uvjeti ravnoteže mogu se izraziti i u *mješovitoj formulaciji* [20]. Tijelo na koje djeluju poopćene sile bit će u ravnoteži ako iščezava vektorski zbroj svih sila i ako zasebice iščezavaju zbrojevi momenata oko tri osi koje nisu usporedne s istom ravninom:

$$\begin{aligned}
\sum_i \vec{F}_i &= \vec{0}, \\
\sum_j M_j^{o_1} &= 0, \\
\sum_j M_j^{o_2} &= 0, \\
\sum_j M_j^{o_3} &= 0.
\end{aligned}$$

Navedenu formulaciju nazivamo mješovitom jer sadrži jedan vektorski i tri skalarna uvjeta.

Osnovna *skalarna formulacija* uvjetâ ravnoteže tijela proizlazi iz činjenice da se vektor u prostoru može odrediti s pomoću tri skalarna podatka. To mogu biti tri koordinate u odabranoj bazi, a bazu čine tri nekomplanarna vektora. Ti podaci mogu biti i (ortogonalne) projekcije na tri nekomplanarne osi. Prema tome, zahtjevu za isčezavanjem vektorskoga zbroja svih sila jednakovrijedan je zahtjev za isčezavanjem zbroja projekcija svih sila za svaku od tri nekomplanarne osi zasebno. Formulacija je potpuna ako se tim uvjetima dodaju još i tri uvjeta za iščezavanje zbroja momenata oko tri nekomplanarne osi. Stoga se u općem slućaju za osnovnu formulaciju može uvesti šest osi. Uvjeti ravnoteže dani su izrazima [19, 20]:

$$\begin{aligned}
\sum_i F_i^{o_1} &= 0, & \sum_j M_j^{o_4} &= 0, \\
\sum_i F_i^{o_2} &= 0, & \sum_j M_j^{o_5} &= 0, \\
\sum_i F_i^{o_3} &= 0, & \sum_j M_j^{o_6} &= 0.
\end{aligned}$$

Za formulaciju uvjetâ projekcija sila i uvjetâ momenata oko osî mogu se upotrijebiti iste osi, pa je dovoljno odabrati tri nekomplanarne osi. Pritom se najčešće uvode tri međusobno okomite osi, što je i najjednostavniji oblik osnovne formulacije:

$$\begin{aligned}
\sum_i F_i^x &= 0, & \sum_j M_j^x &= 0, \\
\sum_i F_i^y &= 0, & \sum_j M_j^y &= 0, \\
\sum_i F_i^z &= 0, & \sum_j M_j^z &= 0.
\end{aligned}$$

Može se uočiti da je skalarna formulacija uvjetâ ravnoteže općenitija od vektorske — točka redukcije koja se uvodi pri postavljanju momentnoga uvjeta ravnoteže u vektorskoj formulaciji u biti je sjecište triju osi baze rezultirajućeg vektora momenta, dok u skalarnoj formulaciji tri osi, za koje se postavljaju momentni uvjeti, ne moraju prolaziti istom točkom. Drugim riječima, vektorska formulacija odgovara mješovitoj formulaciji kod koje momentne osi prolaze istom točkom. Taj se slučaj notacijom vektorske algebre mora opisati pomoću tri zasebne jednadžbe [20]. Uvjet isčezavanja zbroja momenata oko odabrane osi prikazuje se u vektorskoj algebri izrazom

$$\left(\sum_i \vec{r}_i \times \vec{F}_i + \sum_j \vec{M}_j \right) \cdot \vec{e}_0^{\,o} \;=\; 0,$$

gdje je $\vec{e}_0^{\,o}$ jedinični vektor na osi.

Uz osnovnu skalarnu formulaciju uvjeta ravnoteže, moguće su još tri daljnje skalarne formulacije, ovisno o načinu postavljanja uvjeta [19, 20]:

- dvije osi za projekcije sila i četiri momentne osi,
- jedna os za projekciju sila i pet momentnih osi te
- šest momentnih osi.

Raspored tih osi mora zadovoljiti određene geometrijske uvjete. Kriterij ispravnosti je nemogućnost zadovoljenja uvjeta ravnoteže uz djelovanje jedne sile ili sprega, koji naravno ne smije biti nulvektor.

Za ravninski se slučaj može napraviti restrikciju navedenih uvjeta ravnoteže. Djelovanja se smatraju *ravninskima* ako sve sile djeluju u istoj ravnini, a vektori su koncentriranih momenata okomiti na tu ravninu. I za ovaj poseban slučaj vrijedi na prethodnoj stranici navedena opća vektorska formulacija uvjeta ravnoteže, ali ako se točka redukcije odabere u ravnini silâ, a baza za prikazivanje vektora tako da dvije osi leže u toj ravnini, različite od nule mogu biti samo projekcije sila na osi u ravnini i momenti oko osi koja ne leži u ravnini. Odabere li se k tome još momentna os okomito na ravninu silâ, momenti oko osi bit će jednaki vrijednostima momenata sila na točku redukcije i vrijednostima koncentriranih momenata.

Moguće su tri skalarne formulacije uvjeta ravnoteže za djelovanja u ravnini, pri čemu moraju biti zadovoljeni određeni geometrijski uvjeti [19, 20]:

a) Osnovna formulacija, koja sadrži dva uvjeta projekcija silâ i jedan momentni uvjet:

$$\sum_i F_i^{o_1} \;=\; 0, \qquad \sum_i F_i^{o_2} \;=\; 0 \qquad \text{i} \qquad \sum_j M_j^{\mathsf{T}} \;=\; 0.$$

Osi o_1 i o_2 ne smiju biti paralelne.

b) Formulacija s pomoću dvije točke redukcije, koja sadrži jedan uvjet projekcija silâ i dva momentna uvjeta:

$$\sum_i F_i^o \;=\; 0, \qquad \sum_j M_j^{\mathsf{T}_1} \qquad \text{i} \qquad \sum_j M_j^{\mathsf{T}_2} \;=\; 0.$$

Točke redukcije T_1 i T_2 moraju biti različite, a os o ne smije biti okomita na njihovu spojnicu.

c) Formulacija s pomoću tri točke redukcije, koja sadrži tri momentna uvjeta:

$$\sum_j M_j^{\mathsf{T}_1} \qquad \sum_j M_j^{\mathsf{T}_2} \qquad \text{i} \qquad \sum_j M_j^{\mathsf{T}_3} \;=\; 0.$$

Točke redukcije moraju, naravno, biti različite i ne smiju biti kolinearne.

Sve su navedene formulacije uvjeta ravnoteže jednako vrijedne i vode na jednaka rješenja pri rješavanju istoga zadatka. Razlog za uvođenje različitih formulacija je omogućavanje efikasnijeg rješavanja pojedinih zadataka, te bolje razumijevanje i usvajanje zakonitosti ravnoteže.

2.4. Primjena zakonitosti ravnoteže

Uvjeti ravnoteže tijela neposredno se primjenjuju na dva načina: za kontrolu ravnoteže i za postupke uravnotežavanja, a upotrebljava se i statička ekvivalencija koja se izvodi iz uvjeta ravnoteže [20].

Uravnotežavanje je postupak u kojem se uz poznate podatke o nekim silama iz uvjeta ravnoteže određuju neki daljnji podaci o silama. Sam postupak može se provesti za jednu točku, za jedno tijelo, te za sustav koji sadrži više tijela i/ili točaka. Primjena *statičke ekvivalencije* je postupak zamjene zadanih djelovanja (ili samo nekih od njih) zamišljenim djelovanjima čiji su doprinosi uvjetima ravnoteže jednaki doprinosima zadanih djelovanja. Postupak se provodi postavljanjem uvjeta ekvivalencije, koji se za koncentrirana djelovanja u vektorskom obliku izražavaju jednadžbama:

$$\sum_i \vec{F}_i \;=\; \sum_\ell \vec{P}_\ell,$$

$$\sum_i \vec{r}_i \times \vec{F}_i \;+\sum_k \overrightarrow{M}_k \;=\; \sum_\ell \vec{r}_\ell \times \vec{P}_\ell \;+\sum_j \vec{K}_j.$$

Bilo koji skup koncentriranih djelovanja u prostoru može se uvijek zamijeniti statički ekvivalentnim skupom (nazvanim *dinamom* ili *rezultirajućim djelovanjem*) koji sadrži jednu

silu $\vec{F}_R$ (*rezultirajuća sila* ili *glavni vektor sila*) i jedan spreg ili koncentrirani moment $\overrightarrow{M}_R$ (*rezultirajući moment* ili *glavni vektor momenta*). Ako su ispunjeni uvjeti

$$\vec{F}_R \neq \vec{0},$$
$$\overrightarrow{M}_R \cdot \vec{F}_R = 0,$$

onda postoji *rezultanta* — sila koja je sama ekvivalentna zadanom djelovanju. Drugi uvjet izražava ortogonalnost glavnoga vektora sila i glavnog vektora momenta. Taj je uvjet za slučaj djelovanja u ravnini automatski ispunjen (naravno, ako je $\vec{F}_R \neq \vec{0}$).

Ako isčezava rezultirajuća sila, a postoji rezultirajući moment, zadani sustav sila i momenata svodi se na spreg ili na koncentrirani moment. Ako pak isčezavaju i rezultirajuća sila i rezultirajući moment, za zadani sustav kažemo da je *statički neutralan* ili, kad djeluje na tijelo, *uravnotežen*.

Veza uravnotežavajuće grupe i ekvivalentne grupe može se u matematičkoj notaciji izraziti:

$$\vec{F} + \vec{P} = \vec{0},$$

gdje $\vec{P}$ označava ukupnost uravnotežavajućih djelovanja, a $\vec{F}$ ukupnost ekvivalentnih djelovanja.

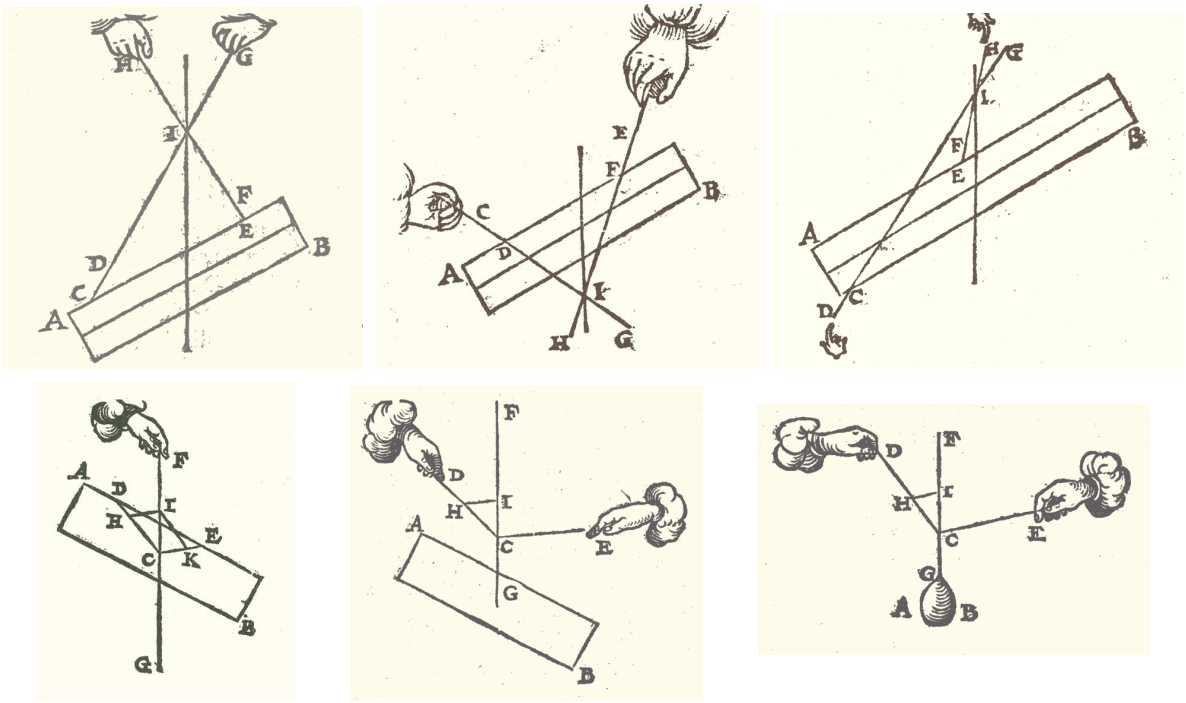
Uz statičku ekvivalenciju, u primjenama i analizama upotrebljava se i suprotan postupak — sila se zamjenjuje s više zamišljenih sila; najjednostavniji je slučaj rastavljanje sile na komponente.

Zakonitosti koje proizlaze iz uvjeta ravnoteže su [20]:

- doprinos sile uvjetima ravnoteže ne ovisi o položaju sile na pravcu djelovanja — sila je *klizni vektor*;
- doprinos koncentriranoga momenta uvjetima ravnoteže ne ovisi o položaju hvatišta momenta — moment je *slobodni vektor*;
- u uvjetima ravnoteže se sile i momenti pojavljuju kao linearni faktori, pa se mogu primijeniti svojstva linearnosti — vrijedi *princip superpozicije*.

Sile su, kao što je opće poznato, vektorske veličine s definiranim operacijama nad njima. Operacija zbrajanja više sila u grafičkoj se realizaciji provodi pomoću *poligona sila* (poopćenje postupka zbrajanja dviju sila pomoću trokuta sila): zbroj niza sila $\left\{\vec{F}\right\}_i$ je sila $\vec{F}$ čiji se početak (‚rep') poklapa sa početkom prve sile, a kraj (‚vrh' ili ‚šiljak') s krajem zadnje sile. Stranice tako dobivenoga poligona usporedne su s pravcima djelovanja sila naznačenih u planu položaja, a duljine tih stranica u određenom mjerilu odgovaraju intenzitetima pripadnih sila. Sile koje se zbrajaju pritom ne moraju ležati u istoj ravnini, te se razlikuju ravninski i prostorni poligoni sila.

Zbrajanje pak dviju sila pomoću trokuta sila drugi je oblik *pravila paralelograma sila*: zbroj $\vec{F}$ sila $\vec{F}_1$ i $\vec{F}_2$ dijagonala je paralelograma čije su susjedne stranice $\vec{F}_1$ i $\vec{F}_2$. To je pravilo izveo i detaljno obradio Simon Stevin u djelu *Elementi umijeća vaganja* (*De beghinselen der weeghconst*) izdanom 1586. godine [2]. Slika 4. sadrži nekoliko crteža koji se odnose na poligon sila iz spomenute Stevinove knjige te iz *Dopune umijeću vaganja* (*Byvough der weeghconst*, 1608.).

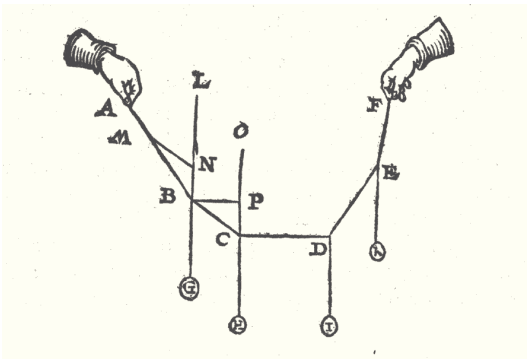


Slika 4. Uvođenje pravila paralelograma sila (iz [2])

Verižni poligon je geometrijska konstrukcija koja ima iznimno veliko polje moguće primjene (određivanje težišta tijela, određivanje momenata tromosti, određivanje rezultirajućeg djelovanja, određivanje reakcija i unutarnjih sila štapova i sustava štapova u ravnini, ...), a zbog svoje ‚posebne' veze s dijagramom momenata savijanja (afina slika) naročito je interesantan za razmatrano područje ovog rada. Također, za skupove sila na paralelnim pravcima verižni je poligon jedini praktični geometrijski postupak. Zametak verižnoga poligona može se naći u Stevinoj *Dopuni umijeću vaganja* (slika 5.).

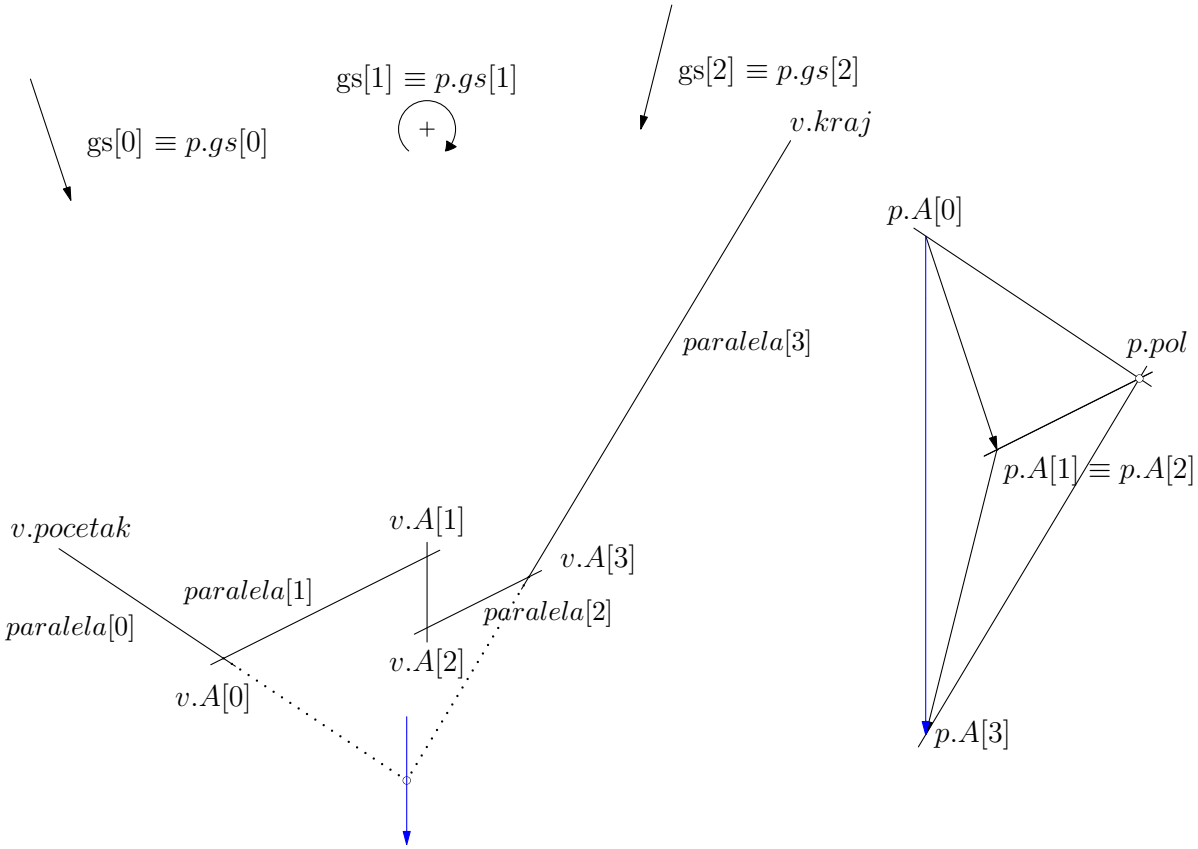
Verižni poligon je ravninska konstrukcija. Za rješavanje prostornih zadataka mogu se primijeniti nacrtnogeometrijski postupci projiciranja na dvije ili tri ravnine.

Verižni je poligon geometrijska konstrukcija u kojoj se nizom pogodnih rastavljanja zadani sustav djelovanja (generaliziranih sila) svodi na dvije sile. Svođenje svih sila i momenata na samo dvije sile ostvaruje se rastavljanjem svake zadane sile u dvije komponente i zamjenjivanjem svakoga zadanog momenta spregom sila, pri čemu te komponente i sile tih



Slika 5. Slutnja verižnoga poligona (iz [2])

spregova biramo tako da jedna komponenta svake sile ili jedna sila svakog sprega ponište jednu komponentu neke druge sile ili jednu silu nekog drugog sprega. Dvije pak tražene sile sastavljaju se od po dvije komponente, pri čemu u svakom paru komponenata jednu zadajemo tako da poništi po jednu od dvije sile na koje su netom svedene zadane sile i

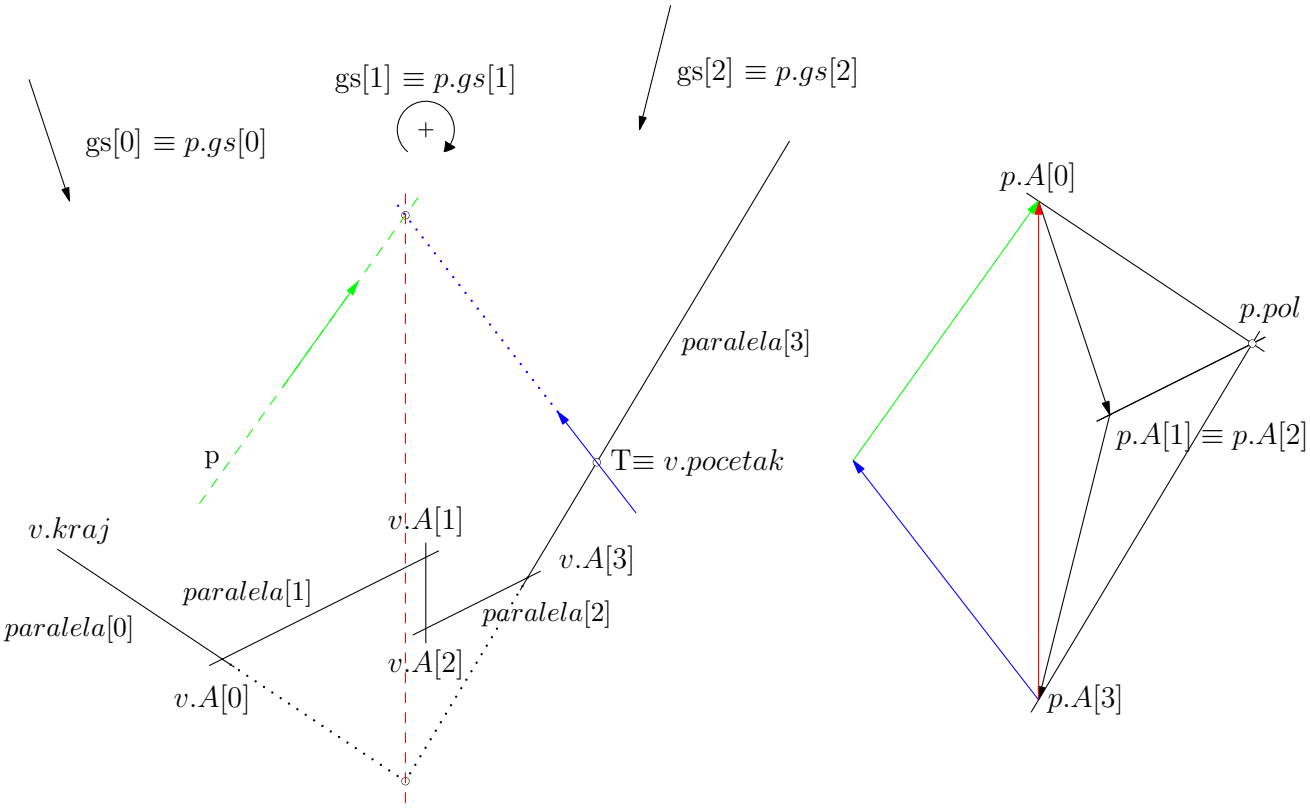


Slika 6. Rezultantna sila

momenti. Verižni je poligon sastavljen od pravaca koji se nanose u planu položaja. Ti su pravci usporedni s odgovarajućim zrakama koje se konstruiraju u poligonu sila (zrake izlaze iz pola te sačinjavaju pramen pravaca).

Nakon definiranja poligona sila i verižnoga poligona može se izreći grafički uvjet ravnoteže sustava sila i momenata u ravnini. Nužan su i dovoljan uvjet ravnotežnog stanja zatvoreni poligon sila i ‚zatvoreni' verižni poligon, pod čim se podrazumijeva poklapanje početne i završne zrake verižnoga poligona.

Slika 6. na prethodnoj stranici prikazuje određivanje rezultirajućeg djelovanja, a slika 7. uravnoteženje niza sila i momenata uporabom verižnoga poligona i poligona sila. Na slikama su označeni elementi navedenih poligona pomoću kojih su oni realizirani u programu GS, a koji će biti detaljnije opisani u odjeljku 5.



Slika 7. Uravnoteženje silom kroz točku i silom na pravcu

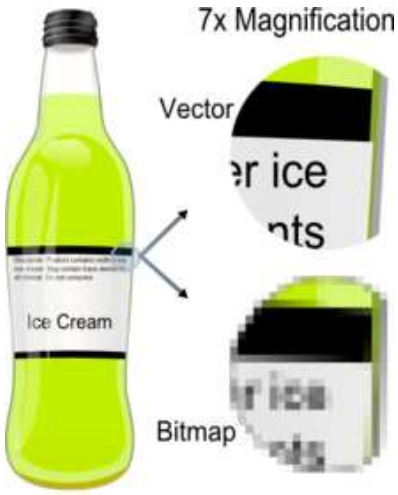
3. Programski jezik `Asymptote`

3.1. Općenito

`Asymptote` je deskriptivni vektorski grafički programski jezik koji pruža matematički okvir za tehničke crteže zasnovan na koordinatama [6]. Riječ je o višem programskom jeziku sa sintaksom i semantikom prilagođenom svom području primjene.

Vektorska grafika podrazumijeva upotrebu osnovnih geometrijskih elemenata poput točaka, pravaca, krivulja i poligona opisanih matematičkim jednadžbama za prikaz grafike na računalu. Alternativa vektorskoj grafici je rasterska grafika koja sliku prikazuje kao skup piksela (njezin poznati predstavnik je format bitmap). Prednosti — motivacija za upotrebu — vektorske grafike su:

- minimalna količina potrebnih podataka implicira manje veličine datoteka;
- teorijski neizmjerna mogućnost uvećavanja slike uz zadržavanje oštrine i ‚glatkoće' elemenata (slika 8..);



Slika 8. Vektorska i rasterska grafika

- pri uvećavanju slike debljina krivulja ne mora se proporcionalno povećavati (najćešće se ne povećava ili se ponešto povećava);
- parametri grafičkih objekata se čuvaju i mogu se kasnije mijenjati bez ustupaka u kvaliteti prikaza (omogućeno je pomicanje, rastezanje, afine transformacije, ...).

Izlazna datoteka koju `Asymptote` ‚izbaci' nakon izvršenja programa je u formatu PostScript, ali su mogući i drugi formati (svi formati koje podržava paket ImageMagick, među kojima su DPX, EXR, GIF, JPEG, PDF, PNG, TIFF itd.).

Asymptoteovi uzori i ‚rođaci' su programski jezici C++, te METAPOST i METAFONT. Grafičku je stranu Asymptote naslijedio od grafičkoga jezika METAPOST, proširivši je i poopćivši na prikaz prostornih tvorevina. Grafički je pak jezik METAPOST Johna D. Hobbyja [8, 7] zasnovan na grafičkom jeziku METAFONT Donalda E. Knutha [9]. METAPOST je od METAFONT-a preuzeo elegantnu deklarativnu sintaksu za baratanje točkama, pravcima, krivuljama i geometrijskim transformacijama. No, dok je METAFONT ponajprije zamišljen za oblikovanje slova (fontova), METAPOST je namijenjen izradi dijagrama, grafova i drugih crteža za znanstvene i tehničke publikacije.

S druge strane, Asymptote je programski jezik sa sintaksom i semantikom uvelike nalik onoj programskoga jezika C++. To, međutim, znači da je deskriptivna programska paradigma dijelom ‚žrtvovana' i zamijenjena objektno usmjerenom paradigmom [18].

Mogućnosti programa Asymptote mogu se proširivati i prilagođavati pojedinim područjima dodatnim modulima. Od postojećih modula, za izradu programa GS najznačajniji je modul math koji matematičke sposobnosti Asymptotea proširuje velikim brojem algoritama prvenstveno vezanih za geometrijske manipulacije i linearnu algebru. I sam se program GS može shvatiti kao modul jezika Asymptote za primjenu u području statike štapnih konstrukcija.

Iz navedenoga slijedi da je programski jezik Asymptote „blizak jeziku problema", te je njegova primjena opravdana. Također, Asymptote je u potpunosti kompatibilan s LaTeX-om (upotrebljava ga za prikaz teksta na crtežima). LaTeX Leslieja Lamporta je kompjutorski jezik za obradu i prikaz teksta [10]. Iznimno prikladan za slog i prijelom matematičkoga teksta, a među njegovim se pogodnostima ističu automatsko numeriranje poglavlja, ulomaka, jednadžbi, teorema itd. Zasnovan je na jeziku TeX Donalda E. Knutha.

3.2. Programiranje u Asymptoteu

Ukratko ćemo opisati elemente programskog jezika Asymptote nužne za razumijevanje programa GS, te one najčešće upotrebljavane pri njegovu oblikovanju.

Tip podataka je konkretna reprezentacija (apstraktnoga) pojma [16]. Svaki tip predstavlja jedan jasno određeni i razgraničeni pojam. Definicija tipa sastoji se od opisa strukture podataka i skupa vrijednosti koje ti podaci mogu poprimiti, te od opisa skupa operatora i funkcija koje barataju tim podacima, ali koje su i jedine koje im mogu i smiju pristupiti. Tipovi se obično svrstavaju u osnovne, složene (ili izvedene) i korisničke.

Osnovni tipovi podataka koje podržava Asymptote su:

`void` — tip koji odgovara praznom skupu; primjenjuje se za tip vrijednosti funkcije koja ne vraća nikakvu vrijednosti i za tip parametra funkcije koja nema parametara;

`bool` — logički tip: može poprimiti dvije vrijednosti: `true` i `false`;

`int` — cijeli broj s predznakom;

`real` — realni broj; uz prikaz realnih brojeva u računalu vezana su pitanja strojne točnosti, značajnih znamenaka, te najmanjega i najvećeg prikazivog pozitivnog broja.

Složeni tipovi cjeline su sastavljene od osnovnih podataka. Jednostavniji su složeni tipovi:

`pair` — uređeni par realnih brojeva, pogodan za prikaz koordinata točaka u ravnini i komponenata vektora u ravnini. Od funkcija koje barataju varijablama tipa `pair` spomenut ćemo dvije:

`real length (pair z)` — udaljenost od ishodišta do točke Z i duljina vektora: `d = length (z2 - z1);`

`pair unit (pair z)` — jedinični vektor orijentiran od ishodišta prema točki Z; naravno, i jedinični vektor na pravcu vektora $\overrightarrow{Z_1Z_2}$: `z0 = unit (z2 - z1);`

`triple` — uređena trojka realnih brojeva, pogodna za prikaz koordinata točaka i komponenata vektora u prostoru;

`string` — niz znakova.

Niz određenog broja varijabli nekog tipa prikazuje se varijablom složenoga tipa `array`. Podržani su jednodimenzionalni, dvodimenzionalni i trodimenzionalni nizovi, a za baratanje nizovima (kao cjelinama) i njihovim komponentama definiran je velik broj funkcija. Jednodimenzionalnim nizovima mogu se prikazati vektori, a dvodimenzionalnima matrice. Spomenut ćemo funkcije za linearnoalgebarske operacije:

`T[][] transpose (T[][] a)` — transponiranje ‚matrice' $\mathbf{A}$, pri čemu komponente matrice mogu biti bilo kojeg tipa;

`real[] solve (real[][] a, real[] b)` — rješavanje sustava linearnih jednadžbi $\mathbf{A}\,\mathbf{x} = \mathbf{b}$;

`real[][] inverse (real[][] a)` — invertiranje matrice $\mathbf{A}$.

Struktura — tip `struct` — složeni je tip podataka koji omogućava definiranje vlastitih/korisničkih tipova podataka. Za razliku od niza, komponente strukture mogu biti različitih tipova. Poligon sila i verižni poligon, primjerice, definirat ćemo kao strukture.

Varijablama različitih tipova, osnovnih, složenih i korisničkih, barata se pomoću operatorâ i funkcija. Uporaba operatora slična je onoj u jeziku C++, uz poneke iznimke i dodatke. *Operatori* su aritmetički (`+`, `-`, `*`, `/`, `%` i `^` s uvriježenim značenjima), relacijski

(==, !=, <, <=, >, >=), logički (&, |, !) i ‚grafički' (.., ::, -- i --- za povezivanje točaka i krivulja krivuljama, preuzeti iz jezika METAFONT).

Funkcije su imenovane programske cjeline namijenjene rješavanju jasno definiranih zadaća — već bi sam naziv funkcije trebao jezgrovito izražavati njezinu svrhu [16]. (I operatori su funkcije.) Funkcije se u pravilu definiraju pomoću drugih funkcija i pomoću vrijednosti, a moguće je rekurzivno definiranje funkcija.

Asymptote, kao i C++, omogućava i preopterećivanje naziva funkcija i simbola operatora. Pod *preopterećivanjem* se podrazumjeva primjena različitih funkcija istih naziva (s različitim tipovima parametara, odnosno različitim sučeljem iako s istim ili sličnim izvornim kodom) na varijable različitih, ali najčešće srodnih tipova. Izbor jedne od funkcija preopterećenog naziva vrši prevodilac na temelju tipova parametara (pravila za izbor preopterećene funkcije u Asymptoteu slična su onima u jeziku C++).

Posebna skupina funkcija jezika Asymptote, preuzetih iz jezika METAFONT i prilagođenih ‚geometrijskom' programiranju, su *transformacije*. Riječ je o afinim transformacijama t = (t.x, t.y, t.xx, t.xy, t.yx, t.yy) kojima se par (x, y) preslikava u par (x', y'), pri čemu je

```
x' = t.x + t.xx * x + t.xy * y
y' = t.y + t.yz * z + t.yy * y
```

Osnovne ugrađene transformacije — specijalizacije navedene opće afine transformacije — su:

identity() — identitet,

shift() — translacija za zadani parametar,

xscale(), yscale() i scale() — ‚rastezanje' ili ‚stezanje' elemenata,

slant() — ‚posmična deformacija',

rotate() — rotacija za zadani kut oko zadane točke te

reflect() — zrcaljenje.

Također je podržano komponiranje transformacija, te pronalaženje transformacije inverzne zadanoj transformaciji. Uz to je moguće i definiranje vlastitih transformacija.

Iz jezika METAFONT Asymptote je preuzeo i neke ‚geometrijske' i ‚grafičke' tipove podataka:

guide i path — Bezierove krivulje kroz zadane točke definirane parametarski; razlika je u vremenu izračunavanja same krivulje (guide se izračunava u trenutku crtanja, a path općenito u svakoj iteraciji), što je važno s gledišta brzine izvođenja programa;

`pen` — olovka kojom se crta; mogu se mijenjati vrsta linije koja se ‚izvlači' (npr. dotted, dashed), debljina, boja i slično;

`picture` — može se shvatiti kao slika na koju se crta.

Jezik `Asymptote` omogućava i *nasljeđivanje svojstava* i simulaciju *dinamičkoga polimorfizma* pomoću operatora `cast()`. Postupak apstrakcije obuhvaća primarno formiranje pojmova iz niza predodžbi, te uz to i nastajanje novog, općenitijeg pojma iz niza po nečemu sličnih, srodnih pojmova zadržavanjem samo zajedničkih svojstava. Takva apstrakcija naziva se poopćenjem (generalizacijom). Nasuprot tome, dodavanjem svojstava nekom pojmu, njegovim profinjivanjem i preciznijim određivanjem, dobiva se sadržajno bogatiji, ali opsegom uži pojam, ‚posebni slučaj'. Odavde naziv uposebljenje (specijalizacija), ali i ograničenje (determinacija). Mehanizam kojim se u objektno usmjerenim programskim jezicima prikazuje JE hijerarhija je nasljeđivanje [16]. Nasljeđivanje može biti jednostavno (svaki tip kao pretka može imati samo jedan tip), te višestruko (pri oblikovanju tipa povezuju se svojstva dvaju ili više tipa). Nasljeđivanje se upotrebljava u oba smjera, za poopćenje i za uposebljenje. Nasljeđivanje omogućava opis i izražavanje sličnosti i razlika među tipovima, odnosno odvajanje zajedničkih svojstava nekoliko tipova od njihovih posebnosti. Primjer nasljeđivanja i dinamičkoga polimorfizma bit će dan u odjeljku o generaliziranim silama (odjeljak 4.2. na stranici 21).

4. Geometrijski i statički elementi grafičkih zahvata

Da bi se mogli definirati problemi čije rješenje tražimo potrebno je definirati elemente pomoću kojih ih opisujemo i pomoću kojih ih dovodimo u vezu sa zakonitostima koje vode do rješenja, te pomoću kojih iskazujemo i interpretiramo i samo rješenje.

I geometrijski i statički elementi su u samom programu definirani kao strukture, a za njihovo uvođenje, te za operacije nad njima definirane su funkcije. Funkcije će se prikazati u obliku u kojem su uvedene u program GS, a sintaksa kojeg odgovara sintaksi jezika C++ za definiranje funkcija. To znači, prva riječ označava tip vrijednosti funkcije, druga je riječ naziv funkcije, a parovi riječi u oblim zagradama, odvojeni zarezima, tipovi su i nazivi parametara. Radni dio — ‚tijelo' funkcije — nalazi se između vitičastih zagrada. Funkcija se poziva nazivom i argumentima.

4.1. Točke i pravci

Kao osnovne geometrijske elemente uvest ćemo točke i pravce, a kao osnovne geometrijske postupke sječenje i spajanje [15]. Gotovo da se može reći da je u programu GS jedino u tipovima i funkcijama kojima su realizirani ti osnovni elementi i postupci sadržan numerički kôd.

Točka. Osnovni geometrijski element, definiran kao uređeni par realnih brojeva. Uveden je radi čitljivosti koda, te radi omogućavanja promjene koordinatnoga sustava (podrazumijeva se desni koordinatni sustav u ravnini xy; prijelaz u ravninu xz moguć je dodjeljivanjem vrijednosti `false` logičkoj varijabli `is_xy_plane` na početku programa). Položaj točke na ‚papiru' zadaje se u odnosu na koordinatni sustav. Funkcije za zadavanje točaka su:

`tocka tocka (real x1, real x2)` — zadavanje točke pomoću dva realna broja, koordinata x i y ili x i z,

`tocka tocka (pair p)` — zadavanje tocke pomoću uređenoga para realnih brojeva.

Moguće je i relativno zadavanje točaka:

```
tocka A = tocka (3cm, 5cm);
tocka B = tocka (A + (2cm, -3cm));
```

Uz spomenutu jedinicu mjere `cm`, podržane su još i `mm`, `in` te `pt` (1 `pt` = 1/72,27 `in`).

U primjeru je vidljiva i u 3. odjeljku spominjana mogućnost preopterećivanja naziva funkcija.

Pravac. Osnovni geometrijski element. U strukturi je definiran dvjema točkama (`p.a`, `p.b`) kojima prolazi, ali po svojoj je (geometrijskoj) biti neograničen.

Pravac kroz točke `A` i `B` zadaje se funkcijom:

```
pravac pravac (tocka a, tocka b) {
  pravac p = new pravac;
  p.a = a; p.b = b;
  return p;
}
```

Crtanje segmenta pravca (između dviju točaka kojima je definiran u strukturi) obavlja se naredbom:

```
void draw (picture pic = currentpicture, pravac l,
           real off1 = 0, real off2 = off1, pen p = currentpen)
```

Produljenje segmenta pravca koji se crta ‚izvan' točaka `l.a` i `l.b` definira se parametrima `off1` i `off2`.

Temeljni geometrijski postupci. Za temeljne su postupke s točkama i pravcima uvedene funkcije:

- ‚povlačenje' pravca kroz dvije zadane točke: već spomenuta funkcija za zadavanje pravca;
- ‚povlačenje' pravca kroz zadanu točku usporedno sa zadanim pravcem:
  - pravac kroz točku `K`, usporedan s pravcem `p`:

    ```
    pravac paralela (pravac l, tocka k) {
      return paralela (l.a, l.b, k);
    }
    ```

  - pravac kroz točku `K`, usporedan s pravcem kroz točke `A` i `B`:

    ```
    pravac paralela (tocka a, tocka b, tocka k) {
      tocka k2 = tocka (k + (b - a));
      return pravac (k, k2);
    }
    ```

- ‚povlačenje' pravca kroz zadanu točku okomito na zadani pravac:
  - pravac kroz točku `K`, okomit na pravac `p`,

```
pravac okomica (pravac l, tocka k) {
  return okomica (l.a, l.b, k);
}
```

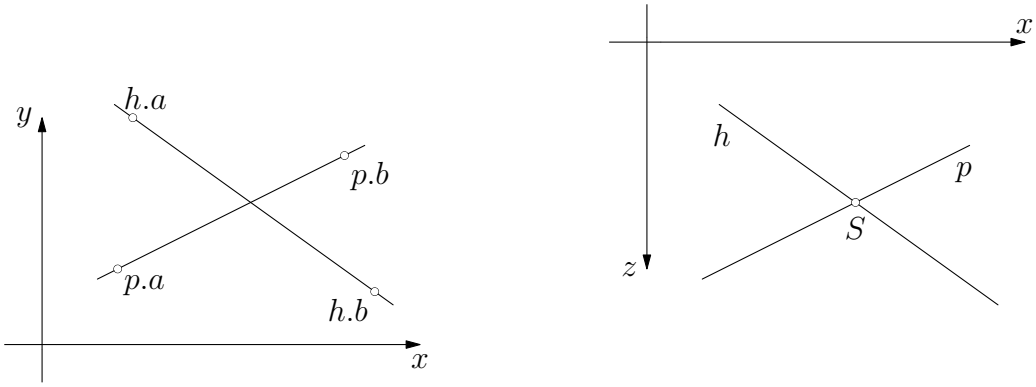
- pravac kroz točku K, okomit na pravac kroz točke A i B:

```
pravac okomica (tocka a, tocka b, tocka k) {
  tocka k2 = tocka (k + rotate (90) * (b - a));
  return pravac (k, k2);
}
```

- nalaženje sjecišta dvaju pravaca:

```
tocka sjeciste (pravac l1, pravac l2) {
  return tocka (extension (l1.a, l1.b, l2.a, l2.b));
}
```

Na lijevom su dijelu slike 9. prikazani elementi strukture `pravac` — točke kojima su pravci definirani. Na desnom je pak dijelu nađeno sjecište pravaca, a pravci su na slici prikazani u oba koordinatna sustava.



Slika 9. Pravci kroz zadane točke i sjecište pravaca

4.2. Sile i momenti

Generalizirane sile. Prilikom rješavanja statičkih problema program GS barata s apstraktnim generaliziranim silama, ali kada se za to ukaže potreba, GS pojedinu generaliziranu silu prepoznaje, izdvajajući njezina posebna obilježja, kao koncentriranu silu ili kao koncentrirani moment i u daljnjim se postupcima ‚ponaša' u skladu s tim. To je u jeziku `Asymptote` omogućeno uvođenjem odnosa nasljeđivanja simulacijom dinamičkoga polimorfizma pomoću operatora `cast()`.

Generalizirane sile kojima se može baratati su koncentrirana sila, koncentrirani moment, te opće trapezno distribuirano opterećenje (slika 10.). Generalizirane su sile definirane strukturom:

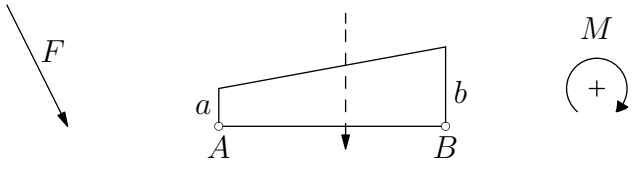
```
struct gen_sila {
  tocka hvat;
  real fx, fy;
  real m;
  bool iz;
  tocka A;
  tocka B;
  real a;
  real b;
  real mjeriloD;
  real h_intenzitet();
  real intenzitet() { return h_intenzitet(); }
  pair h_smisao();
  pair smisao_djelovanja() { return h_smisao(); }
};
```

Elementi strukture su varijable i funkcije kojima su opisane generalizirane sile. Varijabla `hvat`, tipa `tocka`, hvatište je generalizirane sile (hvatište rezultante za distribuirano opterećenje).

Realne varijable `fx` i `fy` sadrže vrijednosti projekcija koncentrirane sile na koordinatne osi (vrijednosti projekcija rezultante za distribuirano opterećenje); za koncentrirani moment te su vrijednosti jednake nuli (takva je, ‚debela', struktura, koja sadrži ‚sve i svašta', nužna za simulaciju dinamičkog polimorfizma).

Realna varijabla `m` sadrži vrijednost koncentriranoga momenta (za razliku od intenziteta, koji je uvijek pozitivan ili jednak nuli, vrijednost sadrži i predznak, pa je time određen i smisao vrtnje momenta).

Vrijednošću logičke varijable `iz` definira se ‚ulazi' li koncentrirana sila u ili ‚izlazi' iz hvatišta (podatak je bitan samo radi crtanja).



Slika 10. Generalizirane sile

Točkama `A` i `B` i realnim brojevima `a` i `b` definira se opće trapezno distribuirano opterećenje (značenje sadržaja varijabli vidljivo je sa slike 10.). Mjerilo duljina definirano je realnom varijablom `mjeriloD`, a potrebno je radi distribuiranoga opterećenja.

Funkcija `intenzitet()` virtualno je sučelje koje se ‚poziva' u programskom kôdu, ali koje poziv samo prosljeđuje ‚skrivenoj' funkciji `h_intenzitet()` čije se, različite, implementacije definiraju u izvedenim tipovima — koncentriranoj sili i koncentriranom momentu:

```
struct sila {
  gen_sila gs;
  // ...
  real h_intenzitet() { return sqrt (gs.fx^2 + gs.fy^2); }
  gs.h_intenzitet = h_intenzitet;
  real intenzitet() = gs.intenzitet;
};

struct moment {
  gen_sila gs;
  // ...
  real h_intenzitet() { return abs (gs.m); }
  gs.h_intenzitet = h_intenzitet;
  real intenzitet() = gs.intenzitet;
};
```

Takvo preusmjeravanje poziva omogućava dinamički polimorfizam [16]. Pokušat ćemo to objasniti pomoću sljedećega programskog fragmenta:

```
gen_sila s[2];
real f;
s[0] = sila (-60., 80.);
s[1] = moment (-50.);
f = s[0].intenzitet();   // 1
f = s[1].intenzitet();   // 2
s[1] = sila (0., -100.);
f = s[0].intenzitet();   // 3
f = s[1].intenzitet();   // 4
```

Program, osim u trenucima stvaranja pojedinih konkretnih sila ili momenata, ‚vidi' samo generalizirane sile sadržane u nizu `s`. U prvom je dijelu programa `s[0]` koncentrirana sila, a `s[1]` koncentrirani moment. Stoga poziv funkcije `intenzitet()` u retku `//1` izračunava intenzitet sile (prema izrazu $\sqrt{F_x^2 + F_y^2}$), a poziv u retku `//2` izračunava intenzitet momenta (prema izrazu $|M|$). Nakon toga i `s[1]` postaje koncentriranom silom, pa pozivi u

recima `// 3` i `// 4` izračunavaju intenzitete sila. Drugim riječima, tek se u trenutku poziva odlučuje koja će funkcija biti pozvana.

Između virtualnoga sučelja `smisao_djelovanja()` i skrivene funkcije `h_smisao()` postoji sličan odnos.

Pretvaranje koncentriranih sila i momenata u generalizirane sile i obratno ostvareno je operatorom `cast`:

```
gen_sila operator cast (sila f) { return f.gs; }
sila operator cast (gen_sila gs) {
  sila f;
  f.gs = gs;
  return f;
}

gen_sila operator cast (moment m) { return m.gs; }
moment operator cast (gen_sila gs) {
  moment m;
  m.gs = gs;
  return m;
}
```

Kao što kôd na prethodnoj stranici pokazuje, koncentrirane sile i koncentrirani momenti također su definirani strukturama. Za distribuirano djelovanje nije uvođena zasebna struktura.

Koncentrirane sile. Za definiranje koncentrirane sile potrebno je poznavati njezin intenzitet, pravac djelovanja i orijentaciju (smisao) na njemu te hvatište. Koncentrirane sile zadaju se funkcijom:

`sila sila (real fx, real fy, tocka hv = tocka (0, 0), bool iz_hv = true)`
— parametri su vrijednosti projekcija sile na koordinatne osi (odnosno, vrijednosti njezinih komponenata u rastavu na pravce usporedne s koordinatnim osima), hvatište `hv` te logička vrijednost `iz_hv` koja određuje ‚izlazi' li sila iz hvatišta ili u njega ‚ulazi', odnosno, smješta hvatište u rep ili šiljak (taj je podatak potreban samo za crtanje). Hvatište ne treba zadavati; u tom se slučaju podrazumijeva ishodište. Isto tako ne treba zadavati ni vrijednost `iz_hv`; podrazumijeva se da sila izlazi iz hvatišta.

Podatke o zadanim silama daju funkcije:

`real fx (sila f)` i `real fy (sila f)` — vrijednosti projekcija koncentrirane sile na koordinatne osi, odnosno, vrijednosti komponenata;

`real intenzitet (gen_sila gs)` — intenzitet koncentrirane sile: $\sqrt{F_x^2 + F_y^2}$.

Treba uočiti da je parameter funkcije tipa `gen_sila`, tako da je funkcija primjenjiva na sve generalizirane sile. Funkcija u stvari samo poziva funkciju–članicu strukture `gen_sila`:

```
pair intenzitet (gen_sila gs) { return gs.intenzitet(); }
```

a uvedena je ponajviše iz sintaktičkih razloga — njezinim uvođenjem zapis poziva poprima oblik uobičajen u matematici, pa primjer sa stranice 23 prelazi u:

```
gen_sila s[2];
real f;
s[0] = sila (-60., 80.);
s[1] = moment (-50.);
f = intenzitet (s[0]);   // 1
f = intenzitet (s[1]);   // 2
s[1] = sila (0., -100.);
f = intenzitet (s[0]);   // 3
f = intenzitet (s[1]);   // 4
```

`pravac pravac_djelovanja (sila f)` — pravac djelovanja koncentrirane sile ili rezultante distribuiranoga djelovanja;

`pair smisao_djelovanja (gen_sila gs)` — jedinični vektor u smjeru i u smislu djelovanja koncentrirane sile ili rezultante distribuiranoga djelovanja;

`tocka hvatiste (gen_sila gs)` — hvatište sile;

`tocka rep (sila f)` — za crtanje sile: točka u kojoj je početak strelice;

`tocka strl (sila f)` — za crtanje sile: točka u kojoj je šiljak strelice.

Koncentrirani momenti. Za ravninske je slučajeve vektor momenta okomit na razmatranu ravninu, te je za njegovo definiranje potrebno poznavati vrijednost momenta predznak koje određuje smisao vrtnje, te njegovo hvatište. Zadavanje momenta vrši se funkcijom:

```
moment moment (real m, tocka h = tocka (0, 0))
```

Podaci o koncentriranom momentu mogu se dobiti funkcijama:

`real m (moment mom)` — vrijednost M čiji predznak određuje smisao vrtnje;

`real intenzitet (gen_sila gs)` — intenzitet: $|M|$;

`tocka hvatiste (gen_sila gs)`

`pair smisao_djelovanja (gen_sila gs)` — rezultat je jedinični vektor $(0, 1)$ ako je smisao vrtnje momenta pozitivan (suprotan smislu vrtnje kazaljke na satu) ili vektor $(0, -1)$ ako je smisao negativan (u smislu vrtnje kazaljke na satu).

Distribuirano djelovanje. Linijske sile se na jednodimenzionalnom prostoru definiraju kao vektorske funkcije dimenzije [sila/duljina]. U programu GS definirana su trokutna, trapezna i jednolika distribuirana djelovanja međusobno paralelnih sila usporednih sa vertikalnom koordinatnom osi, te s ‚osnovicom' djelovanja usporednom s horizontalnom osi koordinatnog sustava. Distribuirano se djelovanje zadaje funkcijom:

```
sila distribuirano (tocka A, tocka B, real a, real b = a,
                    real mjeriloD = cm/100)
```

Značenje prva četiri parametra vidljivo je sa slike 10.. Pretpostavlja se da se vrijednosti `a` i `b` zadaju po metru dužnom. Realni parametar `mjeriloD` mjerilo je duljina (odjeljak 5).

Podaci o zadanom distribuiranom djelovanju mogu se dobiti funkcijama navedenim za koncentrirane sile.

Crtanje. Crtanje niza generaliziranih sila obavlja se funkcijom:

```
void draw (picture pic = currentpicture, gen_sila[] gs,
           real asize = 5.25pt, real aangle = 15, pen p = currentpen)
```

Niz generaliziranih sila može sadržavati i samo jednu ili niti jednu generaliziranu silu (funkcija prihvaća prazan niz sila). Ova se funkcija upotrebljava za crtanje sila u planu položaja (duljine strelica nisu u mjerilu sila). Parametrima `asize` i `aangle` određena je duljina i kut šiljka strelice kojom se prikazuje sila.

5. Alat za rješavanje problema

Može se reći da se problemi statike štapnih konstrukcija grafičkim putem općenito rješavaju primjenom verižnog poligona [17, 5]. (Katkada se upotrebljava i *rezultantni poligon* [20], ali verižni je poligon primjenjiv u znatno većem broju zadataka i mnogo je prilagodljiviji.) Primjena verižnoga poligona implicira primjenu poligona sila, dok obrat naravno ne vrijedi — pojedini problemi ne zahtijevaju konstrukciju verižnoga poligona. Konstrukcije poligona sila i verižnoga poligona realizirane su u programu GS ‚čistim grafičkim' postupcima: kao i pri ‚ručnom' crtanju tih poligona, ‚povlače' se pravci kroz točke i paralele sa zadanim pravcima i ‚nalaze' sjecišta pravaca. Naravno, pri dnu funkcija, kojima su ti temeljni geometrijski postupci realizirani, leže neka izračunavanja (primjerice, izračunavanje koordinata sjecišta dvaju pravaca), ali viša, korisnička razina, na kojoj se rješavaju zadaci statike, geometrijska je, a ne algebarska. Bitno je naglasiti da se sva izračunavanja odvijaju neposredno prije crtanja, tako da i nakon promjene koordinata neke točke ili promjene intenziteta ili nagiba pravca djelovanja neke sile cijela geometrijska konstrukcija ostaje valjanom.

5.1. Mjerila

Zbog same prirode grafičkog rješavanja danih problema postavlja se pitanje mjerila. Značajna su mjerila mjerilo sila i mjerilo duljina [20, 5]. U programu GS realna varijabla `mjeriloD` odnosi se na mjerilo duljina, a realna varijabla `mjerilo` na mjerilo sila.

Mjerilo se zadaje u pozivu funkcija koje ga zahtijevaju. Ako ga se ne zada, prešutno se uzima: za mjerilo sila 1 cm na papiru predstavlja 100 jedinica sile; za mjerilo duljina 1 cm na papiru predstavlja 1 m u prirodi. Mjerila se zadaju u obliku razlomka. Na primjer: `cm/50`; za mjerilo sila to znači da 1 cm na papiru predstavlja 50 jedinica sile, a za mjerilo duljina da 1 cm na papiru predstavlja 50 cm u prirodi. Umjesto `cm` mogu se još upotrebljavati i `mm`, `pt` i `in`.

5.2. Poligon sila

Poligon sila definiran je strukturom:

```
struct poligonsila {
  tocka[] A;
  tocka pol;
  real mjerilo;
  gen_sila[] gs;
  gen_sila RAVN;
  gen_sila EKV;
};
```

Pojedini elementi strukture vidljivi su na slikama 6. i 7. na stranicama 12 i 13. Točke niza `A[]` na slikama su označene sa `p.A[]` da bi se označila pripadnost poligonu sila. Niz generaliziranih sila `gs[]` predstavlja niz sila koje sačinjavaju poligon sila. Sile `EKV` i `RAVN` rezultantna su i uravnotežujuća sila zadanoga niza sila — sile, dakle, koje ‚zatvaraju' poligon.

Poligon sila konstruira se na sljedeći način: počevši iz zadane početne točke, kreira se niz tocaka `A[]` (`A[0]` je ujedno i početna točka) tako da se svaka sljedeća točka niza dobiva iz prethodne translacijom za vektor sile prikazan u odgovarajućem mjerilu. To ostvaruje funkcija:

```
poligonsila poligonsila (gen_sila[] f,
                         tocka poc = (0, 0), tocka pol = (0, 0),
                         real mjerilo = cm/100)
{
  poligonsila p;
  p.gs = f;
  p.pol = pol;
  p.mjerilo = mjerilo;
  p.A[0] = poc;
  for (int i = 1; i <= f.length; ++i)
    p.A[i] = p.A[i-1] + scale (p.mjerilo) * (f[i-1].fx, f[i-1].fy);
  return p;
}
```

Parametar `pol` treba zadati ako će se poligon sila upotrebljavati za konstrukciju verižnoga poligona. Niz sila sadrži generalizirane sile, ali konstrukciji poligona ne smeta ako je argument niz sila među kojima se nalaze momenti.

Iz izloženoga je postupka nastajanja poligona sila vidljivo da je on, iako se koordinate vrhova poligona izračunavaju, analogan crtanju — postupku koji se provodi ‚ručno' na papiru. To je bila zamisao–vodilja pri oblikovanju programa GS.

Prikaz poligona sila na ekranu računala ili na papiru dobiva se funkcijom

```
void draw (picture pic = currentpicture, poligonsila p,
           real asize = 5.25pt, real aangle = 15, pen p = currentpen)
```

Rezultantnu i uravnotežujuću silu crtaju funkcije

```
void ekvivalencija (poligonsila p)
void ravnoteza (poligonsila p)
```

5.3. Verižni poligon

Verižni je poligon definiran strukturom:

```
struct verizni {
  tocka[] A;
  tocka pocetak;
  tocka kraj;
  poligonsila p;
  real mjeriloD;
  pravac[] paralela;
};
```

Slike 6. i 7. na stranicama 12 i 13 prikazuju značenje elemenata navedene strukture. Varijabla `p` tipa `poligonsila` predstavlja poligon sila (s podacima o silama) za koji se verižni poligon konstruira.

Samo formiranje poligona odvija se na sljedeći način. Za zadani poligon sila pri njegovom su kreiranju definirane i zrake pomoću kojih se konstruira verižni poligon (pravci definirani polom poligona sila, te pripadnom točkom iz niza `A[]` poligona sila). Time su definirani pravci na koje se pojedine sile rastavljaju. Zatim se usporedno formiraju niz pravaca `paralela[]` i niz točaka `A[]` (na slikama označeno sa `v.A[]` radi naznačavanja pripadnosti strukturi verižnog poligona), povlačeći paralelu s pripadnom zrakom kroz prethodnu točku niza (počevši iz zadanog početka), te definirajući novu točku niza kao sjecište povučene paralele i pravca djelovanja pripadne sile niza.

Specifičnost pritom predstavlja koncentrirani moment za koji treba postojeću paralelu translatirati ovisno o intenzitetu i smislu vrtnje momenta (na mjestu presjecišta definiraju se dvije nove točke niza `v.A[]`), te tako formirati novu paralelu s kojom se nastavlja postupak.

Verižni poligon prestaje na mjestu presjecišta vertikale kroz zadanu krajnju točku s posljednjim pravcem niza `paralela[]`.

Opisani postupak obavlja funkcija:

```
verizni verizni (poligonsila p,
                 tocka pocetak, tocka kraj,
                 real mjeriloD = cm/100)
{
  verizni v;
  v.p = p;
  v.pocetak = pocetak;
  v.mjeriloD = mjeriloD*100;
```

```
  pravac[] paralela;
  if (v.p.gs.length == 0) {
    v.A[0] = v.pocetak;
    v.kraj = sjeciste (paralela (v.p.A[0], v.p.pol, v.pocetak),
                       okomica ((0, 0), (1, 0), kraj));
  }
  else {
    paralela[0] = paralela (v.p.A[0], v.p.pol, v.pocetak);
    int j = 0;
    for (int i=0; i < v.p.A.length-1; ++i) {
      pravac a = pravac (v.p.gs[i].hvat,
                 v.p.gs[i].hvat + smisao_djelovanja (v.p.gs[i]));
      v.A[j] = sjeciste (a, paralela[i]);
      real m = v.p.gs[i].m * v.p.mjerilo * v.mjeriloD;
      real h = abs (xpart (v.p.pol - v.p.A[i]));
      v.A[j+1] = v.A[j] + (0, m/h);
      paralela[i+1] = paralela (v.p.A[i+1], v.p.pol, v.A[j+1]);
      j = j+2;
    }
    pravac zs = paralela (v.p.A[v.p.A.length-1], v.p.pol, v.A[v.A.length-1]);
    v.kraj = sjeciste (zs, okomica ((0, 0), (1, 0), kraj));
  }
  v.paralela = paralela;
  return v;
}
```

(Navedena funkcija prihvaća i prazan niz sila.)

Redoslijed kojim se sile uvode u verižni poligon općenito nije bitan [20]. Njihovo uvođenje bit će po onom redoslijedu prema kojem su unesene u niz kojim se formira poligon. Ako se želi verižni poligon konstruirati ‚s lijeva na desno', a pri unosu sila se nije pazilo na poredak, može se upotrijebiti funkcija:

```
gen_sila[] poredaj (gen_sila[] gs)
```

koja oblikuje novi niza sila u kojem su sile poredane prema apscisama hvatišta, od hvatišta s najmanjom apscisom prema najvećoj.

Određivanje ravnotežnoga ili rezultirajućeg djelovanja te pravaca tih djelovanja omogućuju funkcije:

```
void ravnoteza (verizni v)
void ekvivalencija (verizni v)
```

Crtanje verižnog poligona na papiru zadaje se funkcijom:

```
void draw (picture pic = currentpicture,
           verizni v, int n = 0, bool krivulja = false, real off = 2mm,
           bool crtajPS = true, bool crtajZRAKE = true,
           real asize = 5.25pt, real aangle = 15, pen p = currentpen)
```

Logički parametar `crtajPS` definira da li se prilikom crtanja verižnoga poligona crta i pripadni poligon sila. Logički parametar `crtajZRAKE` definira crtaju li se pol i zrake u poligonu sila. Iznos produljenja paralela u verižnom poligonu preko njihovih sjecišta definira se realnim parametrom `off`. Značenje parametara `n` i `krivulja` bit će objašnjeno u odjeljku 6. (funkcija `Gerber()`).

6. Grafičko rješavanje statički određenih sustava

6.1. Veze i spojevi

Veze su mjesta na tijelima u krutom agregatnom stanju na kojima je omogućen prijenos kontaktnih sila [20]. U mehanici je u pravilu riječ o idealnim vezama sljedećih svojstava:

- kontaktne plohe su dijelovi ravnine ili kružnog valjka ili kugle,
- kontaktne plohe se ne deformiraju pri djelovanju sila,
- u kontaktu se ne pojavljuje trenje.

Veze mogu biti dvostrane, jednostrane i kombinirane. Dvostrane veze nazivaju se i *spojevima*. Svaka veza ima pripadna statička i kinematička svojstva koja su međusobno komplementarna. Ako se kontakt među tijelima ostvaruje direktno, radi se o neposrednim vezama. U protivnom se radi o posrednim vezama — kontakt je ostvaren indirektno pomoću spojnih elemenata. Poseban slučaj spojeva čine spojevi u ravnini (za spojeve i tijela koja spajaju postoji zajednička ravnina simetrije).

U programu GS kao strukture definirani su nepomični zglobni ležaj, pomični zglobni ležaj, upeti ležaj i zglobna veza (slika 11.). Oni se kreiraju sljedećim funkcijama:

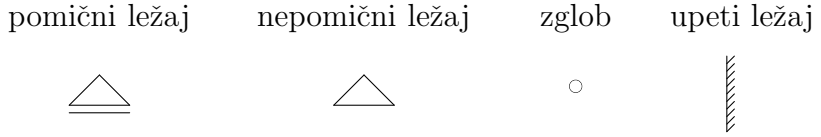
`lezaj pomicni (tocka A)` — pomični ležaj u zadanoj točki,

`lezaj nepomicni (tocka A)` — nepomični ležaj u zadanoj točki,

`lezaj upeti (tocka A)` — upeti ležaj u zadanoj točki,

`zglob zglob (tocka A)` — zglobna veza u zadanoj točki.

Prikaz spojeva na crtežu ostvaruje se, kao i uvijek, funkcijom `draw()`.



Slika 11. Spojevi

6.2. Ravninski sustavi s ravnom osi

Konstrukcije će se prikazivati proračunskim shemama. (*Proračunska shema* je pojednostavljeni prikaz stvarne konstrukcije koji dovoljno vjerno oslikava ponašanje konstrukcije pod opterećenjem i ostalim djelovanjima omogućujući istodobno što jednostavniji proračun [17, 5].) Razmatrat će se konstrukcije i konstrukcijski elementi koji su s geometrijskog stajališta *štapni*. S kinematičkog će se stajališta razmatrati *geometrijski nepromjenjivi* sistemi s najmanjim mogućim brojem ispravno raspoređenih veza (*statički određeni* sistemi).

Posebnu skupinu ravninskih sistema čine *ulančani sistemi* — određeni ravninski spojeni sistemi koji se mogu sastaviti nizom elementarnih spajanja pojedinih tijela i/ili određenih podsustava [20]. Podrobnije ćemo se pozabaviti podskupinom ulančanih sistema s ravnom osi — Gerberovim nosačima.

Gerberovi nosači su statički određeni ravninski ravni nosači horizontalno položeni iznad dva ili više otvora [17, 5]. U osnovnom obliku imaju samo zglobne ležajeve, pri čemu je jedan nepomičan, a svi ostali su uzdužno pomični; na krajevima nosača nema prepusta. No, u Gerberove nosače u širem smislu spadaju i razne varijacije na temu s prepustima i upetim ležajevima. Program GS Gerberove nosače rješava *grafičkim superpozicijskim postupkom*.

6.2.1. Zadavanje grednih sustava

Osnovni elementi ravninskih sistema su štapni elementi. Štapni elementi s pripadnim pravilno raspoređenim vezama čine nosače. Gredni nosači definirani su sljedećom strukturom:

```
struct greda {
  tocka pocetak;
  tocka kraj;
  lezaj[] lezaj;
  zglob[] zglob;
};
```

Točke `pocetak` i `kraj` početna su i krajnja točka grede. Ležajevi su sadržani u nizu `lezaj[]`, a zglobovi u nizu `zglob[]`. Relativnim odnosom položaja početne/krajnje točke grede i položaja početnog/krajnjeg ležaja definiraju se prepusti.

Gredni se nosači zadaju funkcijom

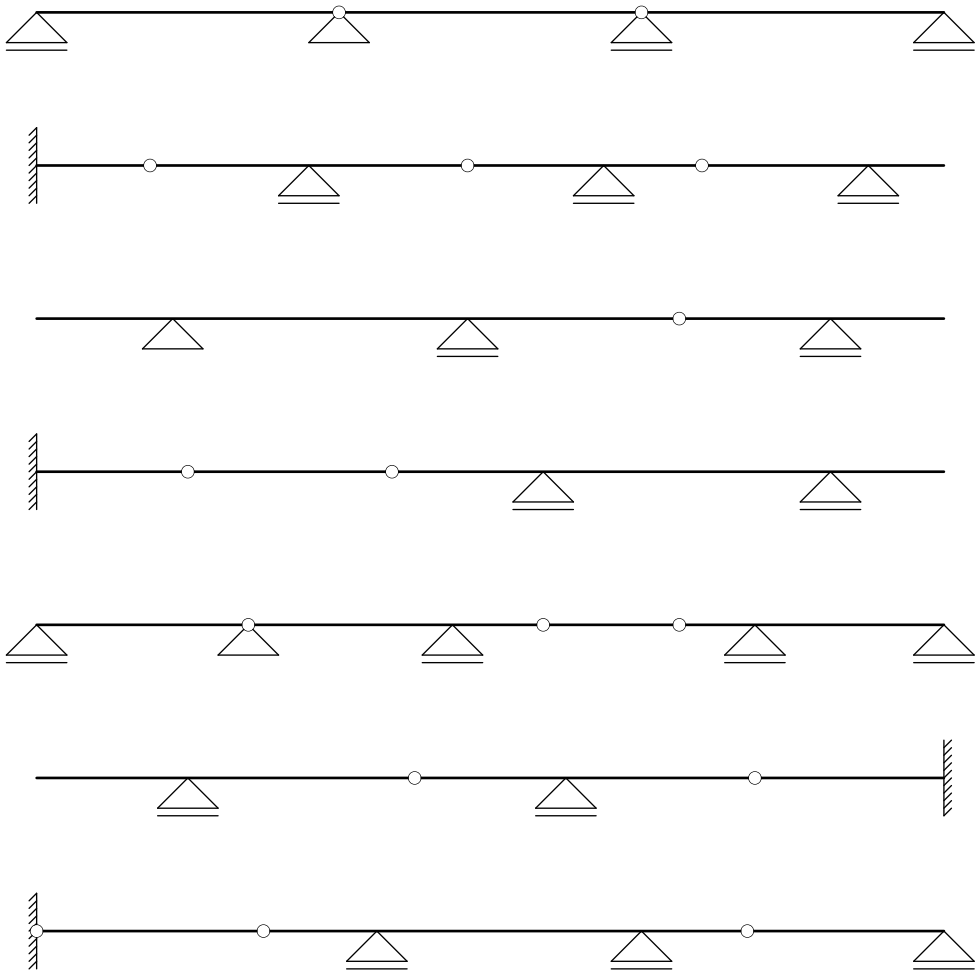
```
greda greda (tocka pocetak, tocka kraj,
             lezaj[] lezaj, zglob[] zglob = new zglob[])
```

a grafički prikaz omogućuje funkcija

```
void draw (picture pic = currentpicture, greda g,
           pen p = currentpen)
```

Na slici 12. prikazano je nekoliko primjera Gerberovih nosača. Drugi je nosač zadan i nacrtan nizom naredbi:

```
import gs;
tocka A = (0, 0);
pair t = (0, -2.025cm);
tocka B = A + t;
```



Slika 12. Gerberovi nosači

```
lezaj[] L2 = new lezaj[];
L2[0] = upeti(B);
L2[1] = pomicni (B + (3.6cm, 0));
L2[2] = pomicni (B + (7.5cm, 0));
L2[3] = pomicni (B + (11cm, 0));

zglob[] Z2 = new zglob[];
Z2[0] = zglob (B + (1.5cm, 0));
Z2[1] = zglob (B + (5.7cm, 0));
Z2[2] = zglob (B + (8.8cm, 0));

greda G2 = greda (B, B + (12cm, 0), L2, Z2);
draw (G2);
```

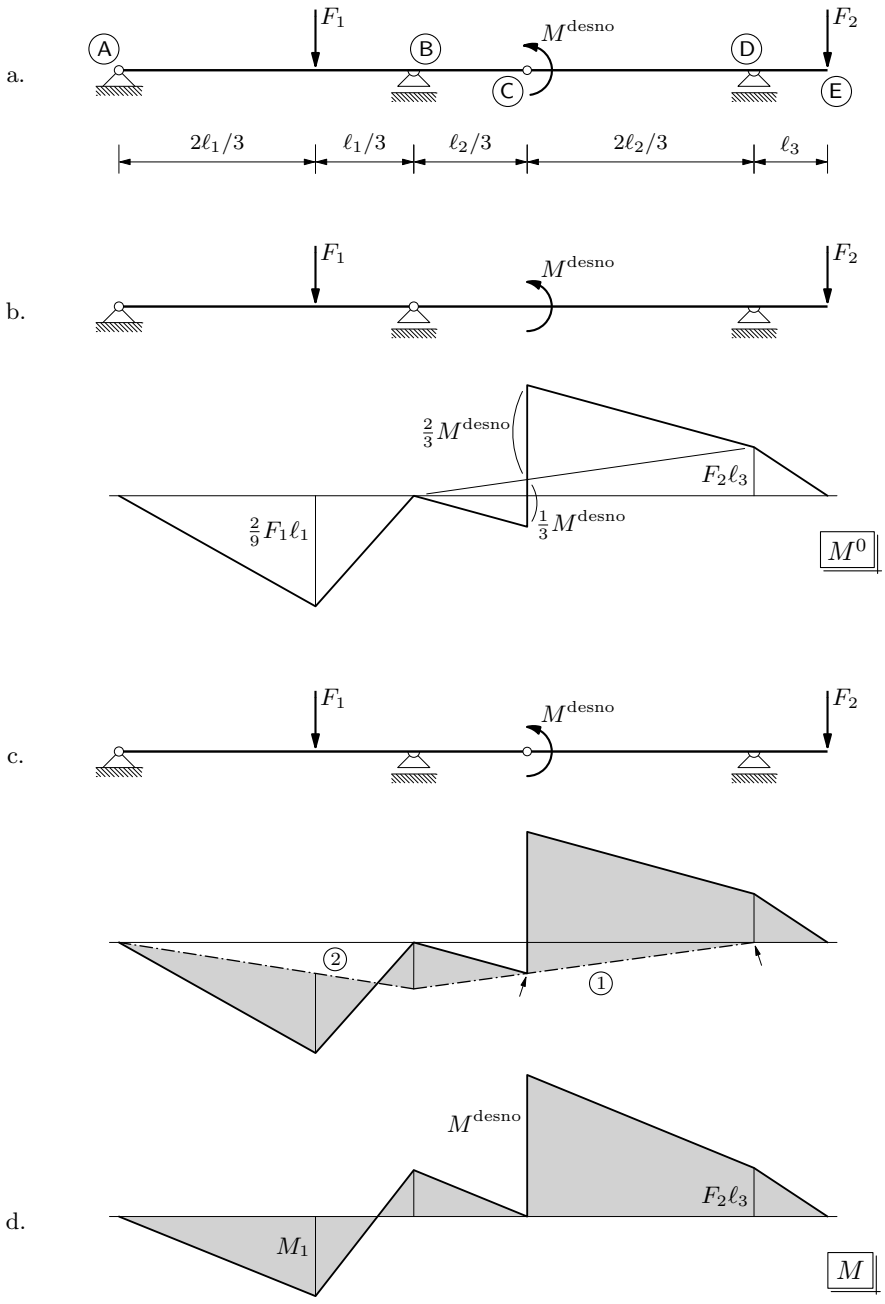
6.2.2. Gerberovi nosači i princip superpozicije

Rješavanje Gerberovih nosača može se provesti primjenom principa superpozicije [17, 5]. Pri ‚čistom' grafičkom rješavanju postupak se provodi pomoću verižnoga poligona [5].

Djelovanja na Gerberove nosače općenito se mogu rastaviti u dvije posebne skupine. Prvoj skupini djelovanja pripadaju sile okomite na os i koncentrirani momenti. U drugu skupinu ulaze opterećenja usmjerena uzduž osi nosača. Djelovanja uzduž osi nosača prenose se do nepomičnog ležaja gdje se uravnotežuju, ne utječući pritom na momente savijanja niti na poprečne sile [5].

Prvu skupinu, rekosmo, čine sile na paralelnim pravcima okomitima na os nosača i koncentrirani momenti (slika 13.a.). Za takav poseban slučaj opterećenja verižni je poligon afina slika momentnog dijagrama [5, 15]. Pritom je ta afina transformacija kompozicija dviju transformacija: ‚rastezanja' dijagrama (vezano uz mjerilo), te ‚posmične deformacije' dijagrama. Iz navedenoga slijedi da se unutarnje sile na Gerberovim nosačima uvijek mogu naći primjenom principa superpozicije ‚čistim' grafičkim postupkom.

Princip superpozicije realizira se uvođenjem zamišljenoga zamjenjujućeg sistema (slika 13.b.), te dovođenjem tog sistema u mehaničko stanje jednako onomu izvornoga Gerberovog nosača. Zamjenjujući sistem nastaje iz početnog pravilnim razmještanjem zglobova tako da se dobije niz prostih greda i greda s prepustima. Za zadana se djelovanja nalaze momenti savijanja na zamjenjujućem sistemu (M^0). Ta djelovanja i pripadne unutarnje sile čine prvo mehaničko stanje zamjenjujućega nosača. Da bi se zamjenjujući sistem doveo u mehaničko stanje početnoga sistema, uvode se dodatna djelovanja — parovi koncentriranih momenata (M^B) nad ležajevima u koje su pri oblikovanju zamjenjućega sistema ‚premješteni' zglobovi. Ta (zasad nepoznata) djelovanja i pripadne (također nepoznate) unutarnje sile čine drugo mehaničko stanje zamjenjujućega nosača. Konačno stanje nosača



Slika 13. Rješavanje Gerberova nosača primjenom principa superpozicije (iz [5])

(M) dobiva se superpozicijom opisanih dvaju stanja, odnosno:

$$M(x) \;=\; M^0(x) \;+\; M^B(x).$$

Uvjet za nalaženje nepoznatih unutarnjih sila iščezavanje je momenata savijanja na mjestima zglobova Gerberova nosača (slika 13.c.) što definira matematički uvjet:

$$M(x_z) \;=\; M^0(x_z) \;+\; M^B(x_z) \;=\; 0.$$

Navedeni uvjet osnova je grafičkoga nalaženja momenata savijanja u Gerberovim nosačima, pri čemu se superponiranje provodi grafičkim putem.

Program GS omogućava rješavanje Gerberovih nosača primjenom principa superpozicije pozivom funkcije:

```
gerber Gerber (greda G, gen_sila[] ggs,
               tocka pocetakPsila = (0,0), real mjerilo = cm/100,
               tocka pocetakVP = (0,0), real mjeriloD = cm/100,
               real polnaudaljenost = 2,
               int n = 0, bool krivulja = true, bool zakljucna = true)
```

Greda `G` predstavlja ranije definirani Gerberov nosač. Niz `ggs` niz je generaliziranih sila koje djeluju na nosač. Točka u kojoj počinje crtanje poligona sila zadaje se parametrom `pocetakPsila`, a početna točka verižnog poligona točkom `pocetakVP`. Realni parametar `polnaudaljenost` predstavlja polnu udaljenost. Cjelobrojna varijabla `n` broj je dodatnih tangenata koje se konstruiraju ispod distribuiranoga djelovanja. Logičke varijable `krivulja` i `zakljucna` određuju crta li se verižna krivulja ispod distribuiranoga djelovanja te crta li se zaključna linija.

Funkcija konstruira rješenje, ali ga ne prikazuje. Rezultat funkcije pohranjuje se u strukturi:

```
struct gerber {
    verizni[] verizni;
    polje[] polje;
    greda G;
    gen_sila[] gs;
    int n;
    bool krivulja;
    bool zakljucna;
    real polnaudaljenost;
    real mjerilo;
    real mjeriloD;
};
```

Funkciju ćemo obrazložiti po segmentima.

Postupak rješavanja grafičkim putem počinje definiranjem zamjenskog sistema. S tom je svrhom definirana struktura:

```
struct polje {
    lezaj pocetni;
    lezaj krajnji;
    gen_sila[] sile;
    zglob[] zglob;
    path linija;
    pravac zakljucna;
    bool zak=false;
    tocka[] B;
    B[0] = (10000.2347cm, 10000.2347cm);
    B[1] = (10000.2347cm, 10000.2347cm);
    tocka ZP = (10000.2347cm, 10000.2347cm);
    tocka ZK = (10000.2347cm, 10000.2347cm);
    int M;
};
```

Ta struktura sadrži podatke o nizu greda zamjenjujućega sistema (položaj, opterećenja, veze). Točkama `B[0]`, `B[1]`, `ZP` i `ZK` treba pridružiti početne vrijednosti kako bi algoritam u kojem se koriste funkcionirao. Točke su inicijalizirane ‚slučajno' odabranim brojem (uređenim parom) za koji postoji vrlo mala vjerojatnost da će izazvati konflikt prilikom rješavanja (točka se nalazi daleko izvan okvira ‚papira' na kojemu se provodi postupak rješavanja).

Prije kreiranja polja, zadano opterećenje sistema prilagođava se grafičkom postupku. Prilagodba se sastoji u kreiranju novoga skupa opterećenja rastavljanjem distribuiranih djelovanja na mjestima ležajeva, zglobova i koncentriranih djelovanja. Prilikom zadavanja opterećenja ne mora se voditi računa o potrebnim podjelama, jer sam program GS djelovanja prilagođava grafičkome postupku. Opterećenja se također mogu zadavati proizvoljnim redoslijedom. Prilagodba se vrši pozivom funkcije `prilagodi()`:

```
gen_sila[] gs = prilagodi (ggs, G.lezaj, G.zglob);
```

Kreiranje poljâ i raspodjela sila po njima provode se u petlji:

```
for (int i=1; i < brojpolja - 1; ++i) {
  gen_sila[] sile = new gen_sila[];
  int k = 0;
  for (int j = 0; j < gs.length; ++j)
    if (gs[j].hvat.p.x > G.lezaj[i].polozaj.p.x
        & gs[j].hvat.p.x <= G.lezaj[i+1].polozaj.p.x) {
      sile[k] = gs[j];
```

```
      k = k + 1;
    }
  zglob[] zglob = new zglob[];
  k = 0;
  for (int j = 0; j < G.zglob.length; ++j)
    if (G.zglob[j].polozaj.p.x >= G.lezaj[i].polozaj.p.x
        & G.zglob[j].polozaj.p.x <= G.lezaj[i+1].polozaj.p.x) {
      zglob[k ] = G.zglob[j];
      k = k + 1;
    }
  polje[i] = polje (G.lezaj[i], G.lezaj[i+1], zglob, sile);
}
```

Nakon što su polja definirana, za svako se polje konstruira verižni poligon uz pomoć poligona sila za to polje. Pritom se distribuirana djelovanja zamjenjuju koncentriranima. Verižni poligoni se u točkama koje odgovaraju zajedničkim ležajevima susjednih greda nadovezuju jedan na drugi. Kako bi mjerila vrijednosti momenata u svim poljima bila jednaka, svi polovi definiraju se s jednakom zadanom polnom udaljenosti. Položaj pola za svako polje nalazi funkcija:

```
tocka pol (tocka pocetakPsila, gen_sila[] gs,
           real polnaudaljenost, real mjerilo)
{
  tocka pol;
  real y = 0;
  for (int i = 0; i < gs.length; ++i)
    y = y + 0.5 * gs[i].fy * mjerilo;
  pol = pocetakPsila + (polnaudaljenost, y);
  return pol;
}
```

Kreiranje niza verižnih poligona započinje konstruiranjem verižnog poligona u prvome polju. U ostalim se poljima verižni poligoni konstruiraju unutar petlje `for`:

```
verizni[] verizni = new verizni[];
poligonsila p = poligonsila (polje[0].sile, pocetakPsila,
                             pol (pocetakPsila, polje[0].sile,
                                  polnaudaljenost,mjerilo),
                             mjerilo);
verizni[0] = verizni (p, mjeriloD, pocetakVP, polje[0].krajnji.polozaj);
```

```
for (int i = 1; i < polje.length; ++i) {
  poligonsila p = new poligonsila;
  p = poligonsila (polje[i].sile,
                   verizni[i-1].p.A[verizni[i-1].p.A.length-1],
                   pol (verizni[i-1].p.A[verizni[i-1].p.A.length-1],
                        polje[i].sile, polnaudaljenost,mjerilo),
                   mjerilo);
  verizni[i] = verizni (p, mjeriloD, verizni[i-1].kraj,
                        polje[i].krajnji.polozaj);
}
```

Potom slijedi definiranje zaključnih linija. Prvo se određuju postojeći rubni uvjeti, odnosno uzimaju se u obzir mogući prepusti, zglobni i upeti krajevi greda, te se, ako je moguće, iz njih nalaze zaključne linije za odgovarajuća polja. Zatim se u petlji ‚provrte' sva polja i, ako je moguće, nađu se zaključne linije iz položaja zglobova:

```
for (int i = 1; i < brojpolja - 1; ++i) {
  if (polje[i].M == 2) {
    polje[i].zakljucna = pravac (polje[i].B[0], polje[i].B[1]);
    polje[i].ZP = sjeciste (polje[i].zakljucna,
                            pravac (polje[i].pocetni.polozaj,
                                    polje[i].pocetni.polozaj + (0,1)));
    polje[i].ZK = sjeciste (polje[i].zakljucna,
                            pravac (polje[i].krajnji.polozaj,
                                    polje[i].krajnji.polozaj + (0,1)));
    polje[i].zak = true;
  }
}
```

Nadalje se po poljima traže rubni uvjeti definirani susjednim poljima, te se definiraju zaključne linije:

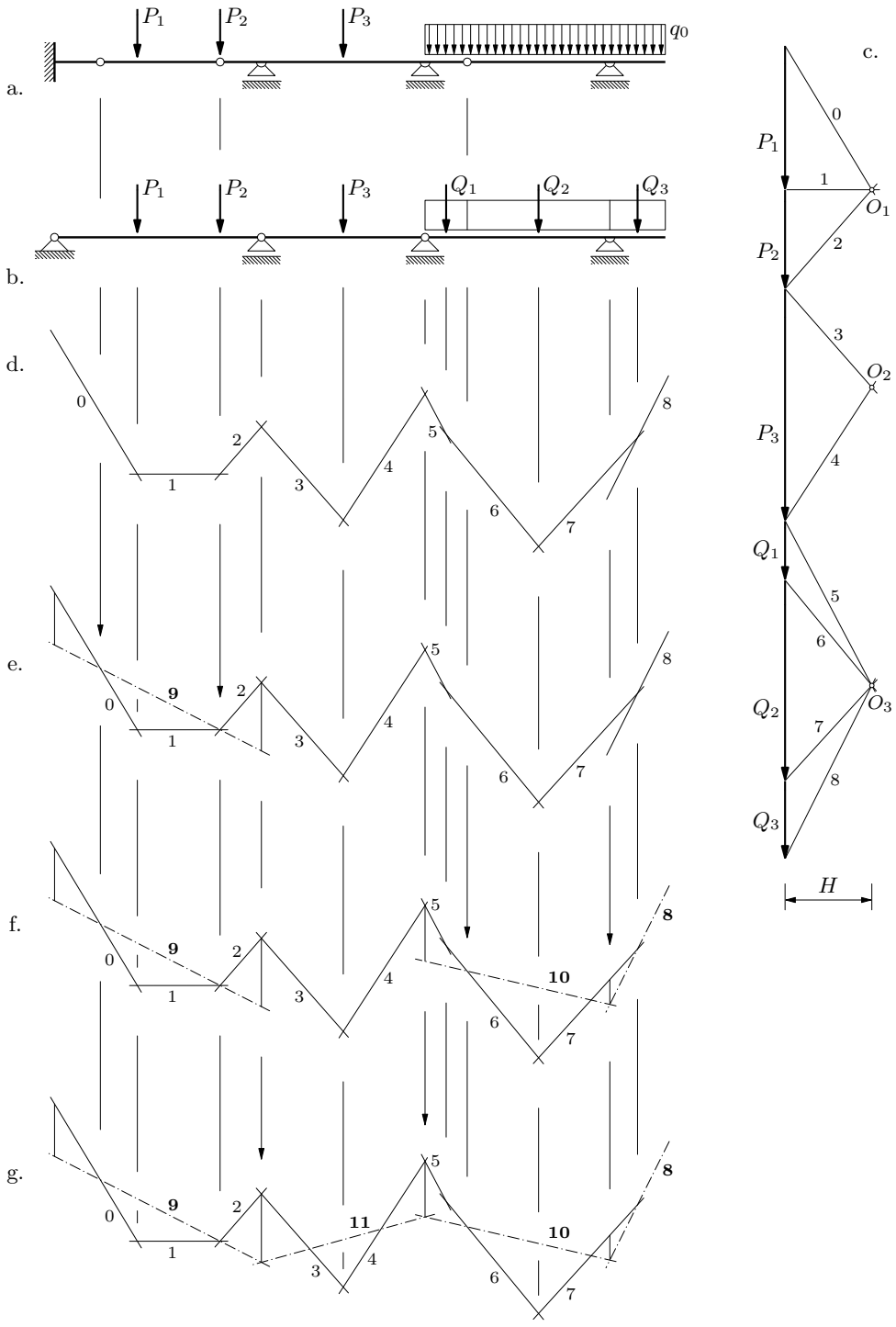
```
for (int i = 1; i < brojpolja - 1; ++i) {
  if (polje[i].M < 2) {
    if (polje[i+1].zak)
      polje[i].ZK = polje[i+1].ZP;
    if (polje[i-1].zak)
      polje[i].ZP = polje[i-1].ZK;
    if (polje[i].ZP.p != (10000.2347cm, 10000.2347cm)
        & polje[i].ZK.p != (10000.2347cm, 10000.2347cm)) {
      polje[i].zakljucna = pravac (polje[i].ZP, polje[i].ZK);
```

```
      polje[i].zak = true;
    }
    if (polje[i].ZP.p != (10000.2347cm, 10000.2347cm)
        & polje[i].B[0].p != (10000.2347cm, 10000.2347cm)) {
      polje[i].zakljucna = pravac (polje[i].ZP, polje[i].B[0]);
      polje[i].zak = true;
      polje[i].ZK = sjeciste (polje[i].zakljucna,
                              pravac (polje[i].krajnji.polozaj,
                                      polje[i].krajnji.polozaj + (0,1)));
    }
    if (polje[i].ZK.p != (10000.2347cm, 10000.2347cm)
        & polje[i].B[0].p != (10000.2347cm, 10000.2347cm)) {
      polje[i].zakljucna = pravac (polje[i].ZK, polje[i].B[0]);
      polje[i].zak = true;
      polje[i].ZP = sjeciste (polje[i].zakljucna,
                              pravac (polje[i].pocetni.polozaj,
                                      polje[i].pocetni.polozaj + (0,1)));
    }
  }
}
```

Nakon što je rješenje nađeno, ono se grafički prikazuje: crtaju se opterećenja, nosači, verižni poligoni, poligoni sila, te zaključne linije. Upotrebljava se funkcija:

```
void draw (picture pic = currentpicture, gerber gerb, pen p = currentpen)
{
  for (int i = 0; i < gerb.polje.length; ++i)
    draw (gerb.polje[i].sile);
  draw (gerb.G);
  for (int i = 0; i < gerb.verizni.length; ++i)
    draw (gerb.verizni[i], gerb.n, gerb.krivulja);
  if (gerb.zakljucna)
    for (int i = 0; i<gerb.polje.length; ++i)
      draw (pravac (gerb.polje[i].ZP, gerb.polje[i].ZK), 0.2cm, dashdotted);
  if (gerb.polje[0].pocetni.upeti)
    draw (pravac (gerb.polje[0].ZP, gerb.verizni[0].pocetak), 0.2cm);
  if (gerb.polje[gerb.polje.length-1].krajnji.upeti)
    draw (pravac (gerb.polje[gerb.polje.length-1].ZK,
                  gerb.verizni[gerb.polje.length-1].kraj), 0.2cm);
}
```

Postupak rješavanja kako ga provodi program GS prikazan je na slici 14.

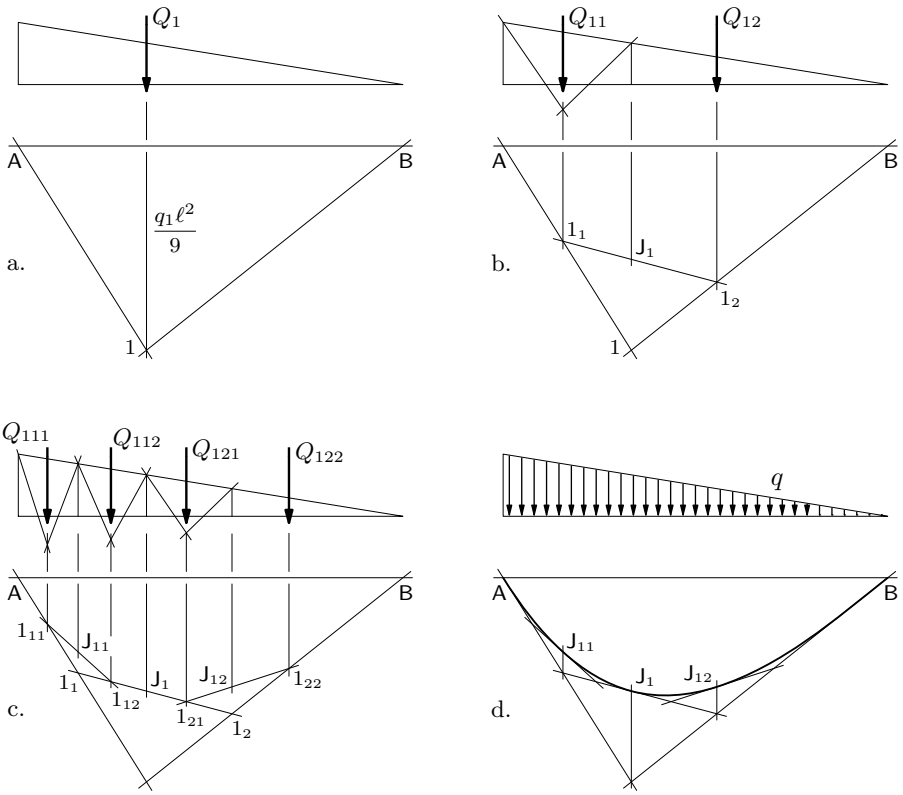


Slika 14. Grafičko rješavanje Gerberova nosača primjenom principa superpozicije (iz [5])

Verižna krivulja. Čitav provedeni postupak rješavanja Gerberovih nosača primjenom principa superpozicije je grafički. Iznimka pritom nije ni konstruiranje verižne krivulje ispod distribuiranoga djelovanja. Oblik krivulje proizlazi iz diferencijalnog odnosa [20]

$$\frac{\mathrm{d}^2 M(x)}{\mathrm{d}x^2} \;=\; -q(x).$$

Krivulja se konstruira rekurzivno sve bolje aproksimirajući distribuiranu silu nizovima koncentriranih sila, odnosno sve boljom aproksimacijom tražene krivulje tangentnim poligonom (slika 15.) [5]. Teorijska osnova za navedenu konstrukciju poznati je Varignonov teorem.



Slika 15. Rekurzivna konstrukcija verižne krivulje (iz [5])

Konstrukcija se provodi primjenom verižnoga poligona. Pritom treba distribuirano opterećenje podijeliti na određeni broj dijelova (formirati niz distribuiranih djelovanja) ovisno o broju koncentriranih sila kojima se ono aproksimira. Pod dodatnim tangentama u programu GS podrazumijevaju se sve tangente osim početne i krajnje, koje su definirane već prvom aproksimacijom. Za n dodatnih tangenata potrebno je distribuirano djelovanje podijeliti na $n + 1$ dio. Mjesta podjele distribuiranih djelovanja su na pravcima djelovanja

rezultanata distribuiranih djelovanja prethodne razine podjele. Pravci djelovanja rezultanata također se određuju grafičkom konstrukcijom. Podjelu distribuiranog opterećenja na $n+1$ dio obavlja funkcija:

```
gen_sila[] podjelanaN1 (gen_sila dist, int n)
{
  gen_sila[] f = new gen_sila[];
  if (n == 0) {
    f[0]=dist;
    return f;
  }
  gen_sila[] A = podjelanadva (dist);
  f[0] = A[0];
  f[1] = A[1];
  int g = 2;
  while (g < n+1) {
    int j = 0;
    g = g * 2;
    for (int i = 0; i < g; i = i + 2) {
      gen_sila[] B = new gen_sila[];
      B = podjelanadva (f[j]);
      f.insert (j, B[0]);
      f.insert (j+1, B[1]);
      f.delete (j+2);
      j=j+2;
    }
  }
  return f;
}
```

Sama podjela odvija se u petlji `while` navedene funkcije. Za broj dodatnih tangenti može se zadati bilo koji član niza $n_k = 2 \cdot n_{k-1} + 1$, $k \in \mathbb{N}$, $n_0 = 0$. Program GS verižnu krivulju aproksimira sa 63 dodatne tangente.

‚Ravnanje' osi. Dijagram momenata savijanja dobiven grafičkim rješavanjem općenito ima os izlomljenu po poljima. Dijagram se može transformirati u dijagram s horizontalnom osi. Za to je potrebna ‚posmična' transformacija koja će osi po poljima svesti na horizontalu. Transformacija se može direktno provesti u poligonu sila [5]. Pogodnim odabirom polova mogu se dobiti horizontalne zaključne linije čime se automatski dobiva i dijagram momenata savijanja sveden na horizontalnu os. Kako su poznate zaključne linije u izlomljenom dijagramu, traženi se položaj pola može konstruirati. Polom u poligonu sila povuče

se zraka usporedno s odgovarajućom zaključnom linijom. Zatim se nađe sjecište te zrake i pravca djelovanja sila u poligonu sila. Rotacijom zaključne zrake oko konstruiranog sjecišta zraka se dovodi u horizontalan položaj (slika 16.b.). Položaj novog pola moguće je odabrati bilo gdje na konstruiranoj zraci. Međutim, da bi izlomljeni dijagram i dijagram sveden na horizontalnu os bili u istom mjerilu, traženi pol odabire se u sjecištu horizontalne zrake i vertikale kroz početni pol. Vektor translacije početnog u traženi pol može se odrediti i pomoću sličnosti trokuta (slika 17.).

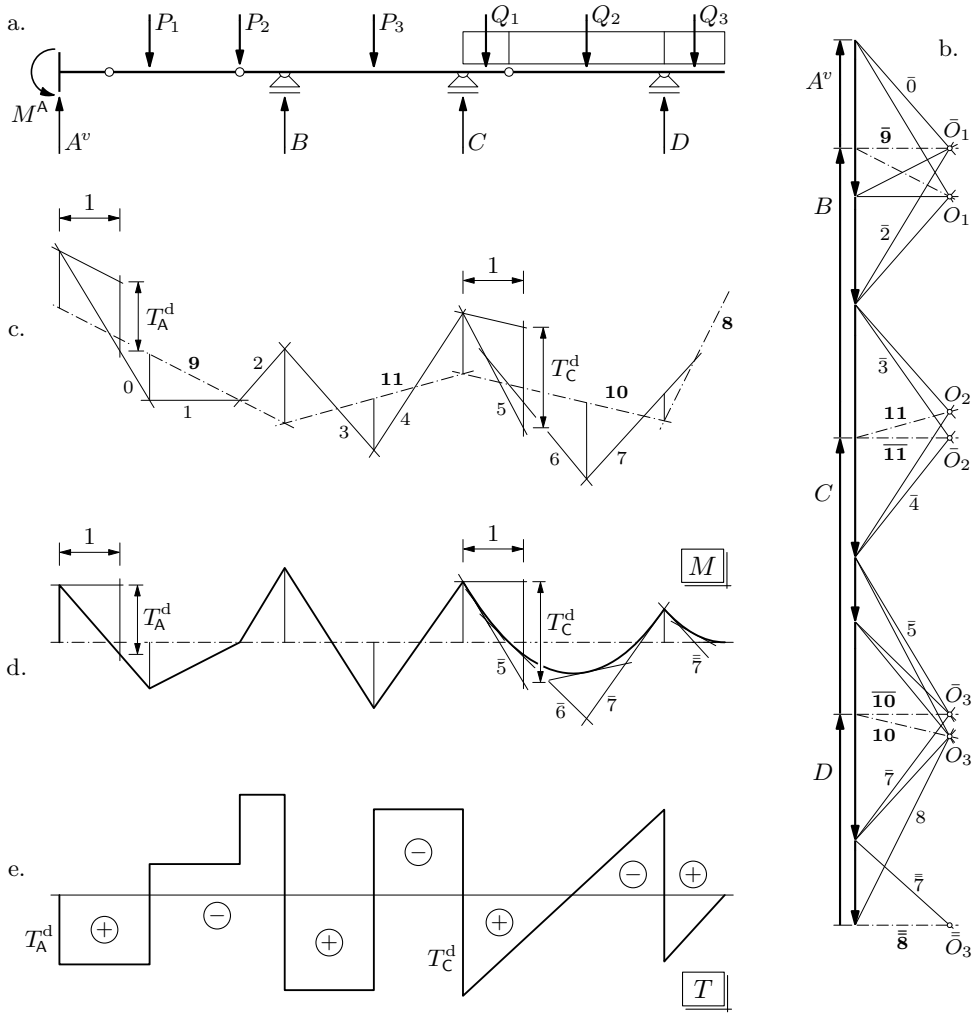
Navedeni postupak provodi funkcija:

```
Mdijagram Mdijagram (gerber G, tocka pocetakVP,
                     tocka pocetakPS = G.verizni[0].p.A[0],
                     bool crtajPSila = false)
```

Parametar `G` tipa `gerber` predstavlja strukturu u kojoj se nalazi rješenje s izlomljenim dijagramom. Parametar `pocetakVP` predstavlja točku iz koje počinje konstruiranje dijagrama s horizontalnom osi. Treba je odabrati na vertikali kroz početnu točku izlomljenog dijagrama. Točka iz koje počinje konstruiranje poligona sila s novim polovima definira se parametrom `pocetakPS`. Ako se ne zada, poligoni sila konstruiraju se na istom mjestu kao i početni. Logičkim parametrom `crtajPSila` određuje se crtanje novih poligona sila (ako se parametar ne zada, novi poligoni sila se ne crtaju).

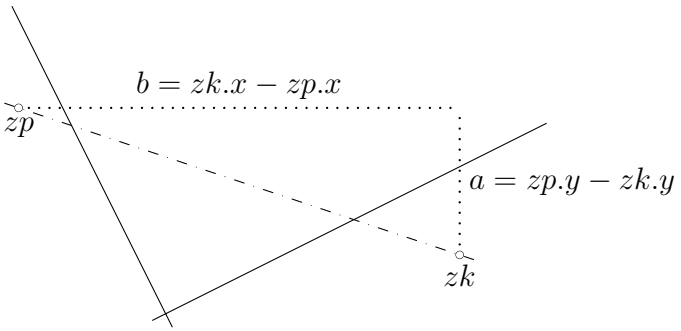
Funkcija prvo pronalazi pogodne polove te konstruira verižne poligone koje zatim i crta:

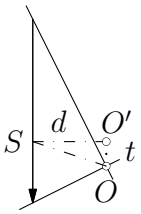
```
verizni[] verizni = new verizni[];

poligonsila p = new poligonsila;
p = poligonsila (
  G.polje[0].sile,
  pocetakPS,
  pol (pocetakPS, G.polje[0].sile, G.polnaudaljenost,G.mjerilo)
    + (0, G.polnaudaljenost
      * ((G.polje[0].ZP.p.y - G.polje[0].ZK.p.y)
        / abs (G.polje[0].ZK.p.x - G.polje[0].ZP.p.x))),
  G.mjerilo
);
verizni[0] = verizni (p, G.mjeriloD, pocetakVP, G.polje[0].krajnji.polozaj);

for(int i = 1; i < G.polje.length; ++i) {
  poligonsila p = new poligonsila;
  p = poligonsila (
    G.polje[i].sile,
```



Slika 16. ‚Ravnanje' osi momentnoga dijagrama i konstrukcija dijagrama poprečnih sila (iz [5])

$$\frac{t}{d} = \frac{a}{b} \Rightarrow t = d \cdot \frac{a}{b}$$





Slika 17. Prilagođavanje položaja pola

```
    verizni[i-1].p.A[verizni[i-1].p.A.length-1],
    pol (verizni[i-1].p.A[verizni[i-1].p.A.length-1],G.polje[i].sile,
         G.polnaudaljenost, G.mjerilo)
      + (0, G.polnaudaljenost
        * ((G.polje[i].ZP.p.y - G.polje[i].ZK.p.y)
          / abs (G.polje[i].ZK.p.x - G.polje[i].ZP.p.x))),
    G.mjerilo
  );
  verizni[i] = verizni (p, G.mjeriloD, verizni[i-1].kraj,
                        G.polje[i].krajnji.polozaj);
}

for (int i=0; i < verizni.length; ++i)
  draw (verizni[i], n=0, krivulja = false, off = 0,
        crtajPS = crtajPSila, crtajZRAKE = crtajPSila,
        dodatnetangente = false);
```

Zatim se na analogan način kao u funkciji `Gerber()` nalaze zaključne linije crtanjem kojih se dobiva konačan dijagram momenata savijanja sveden na horizontalnu os. Rezultat funkcije je struktura tipa:

```
struct Mdijagram {
  verizni[] verizni;
};
```

Dijagram poprečnih sila. Nakon što su određene vrijednosti momenata savijanja uzduž osi nosača mogu se primjenom diferencijalnih odnosa odrediti vrijednosti poprečnih sila. Diferencijalna veza momenata savijanja i poprečnih sila glasi [20]:

$$\frac{\mathrm{d}M(x)}{\mathrm{d}x} = T(x).$$

Deriviranje funkcije momenata savijanja može se provesti čisto grafički [20, 5]. Postupak se svodi na određivanje nagiba tangenata na verižnu krivulju u odabranim točkama. Ako se iz odabrane točke na verižnoj krivulji napravi jedinični pomak po horizontali (pomak od jednog metra prikazan u mjerilu duljina) te se povuče vertikala dobivenom točkom, tada je segment te vertikale između povučenoga horizontalnog pravca (ako je os momentnoga dijagrama svedena na horizontalu) i tangente na verižnu krivulju jednak nagibu te tangente, odnosno derivaciji funkcije momenata savijanja u odabranoj točki (slike 16.d. i e.). Na taj se način određuju iznosi poprečnih sila u odabranim točkama (vrijednosti su u mjerilu koje odgovara mjerilu momenata savijanja).

Poprečne sile određuju se na mjestima hvatišta koncentriranih sila (strogo gledano, vrijednosti se određuju neposredno ispred i neposredno nakon hvatišta pošto je samo hvatište mjesto singulariteta funkcije poprečnih sila) što obuhvaća i ležajeve, te na početku i na kraju distribuiranog djelovanja. Oblik dijagrama poprečnih sila ‚ispod' distribuiranoga opterećenja proizlazi iz diferencijalnog odnosa [20]

$$\frac{\mathrm{d}T(x)}{\mathrm{d}x} \;=\; -q(x).$$

Program GS vrijednosti poprečne sile ispod distribuiranog djelovanja određuje u konačnom broju točaka. Broj točaka ovisi o tome s koliko je tangenata aproksimirana verižna krivulja (momentni dijagram) ispod distribuiranog djelovanja prilikom formiranja dijagrama momenata savijanja svedenoga na horizontalnu os – taj je broj jednak dvostrukom broju hvatišta koncentriranih sila s kojima je aproksimirano distribuirano djelovanje. Navedenu podjelu distribuiranog opterećenja za funkciju `Mdijagram()` vrši funkcija:

```
gen_sila[] MDmod (gen_sila[] gs, int n = 127)
{
  gen_sila[] MD = new gen_sila[];
  for (int i=0; i < gs.length; ++i)
    if (gs[i].mjeriloD == 0)
      MD.push (gs[i]);
    else
      MD.append (podjelanaN1 (gs[i], n));
  return poredaj(MD);
}
```

Funkcija za dani niz sila kreira novi niz pri čemu svako distribuirano djelovanje zamjenjuje nizom od $n + 1$ novih distribuiranih djelovanja (dijeli distribuirana djelovanja na $n + 1$ dio). O broju n ovisi kvaliteta aproksimacije verižne krivulje tangentnim poligonom, ali također i aproksimacija dijagrama poprečnih sila koja je još ‚osjetljivija' na broj n. Kao broj koji se podrazumijeva odabran je $n = 127$ kao optimum između kvalitete prikaza (‚glatkoća' krivulja) i brzine izvođenja. Po potrebi se može zadati bilo koji element niza $n_k = 2 \cdot n_{k-1} + 1$, $k \in \mathbb{N}$, $n_0 = 0$.

Za poziv funkcije koja konstruira dijagram poprečnih sila koristi se rezultat funkcije koja je dijagram momenata savijanja svela na horizontalnu os (struktura tipa `Mdijagram`). Također treba zadati početnu točku dijagrama poprečnih sila na vertikali položenoj početnom točkom momentnog dijagrama. Dijagram poprečnih sila određuje funkcija:

```
Tdijagram Tdijagram (Mdijagram MD, tocka pocetak, real k = 1)
{
```

```
Tdijagram TD = new Tdijagram;
pravac os = pravac (pocetak, pocetak+(1,0));
tocka[] T = new tocka[];
real[] t = new real[];

for (int i = 0; i < MD.verizni.length; ++i) {
  real a = MD.verizni[i].mjeriloD;
  tocka C = MD.verizni[i].pocetak.p + (a,0);
  pravac s = pravac (C, C+(0,1));
  tocka D = sjeciste (
    s,
    pravac (MD.verizni[i].pocetak,
            MD.verizni[i].pocetak
              + (MD.verizni[i].p.pol - MD.verizni[i].p.A[0]))
  );
  t.push (-ypart (D.p - C.p) * (k * cm) / a);
  T.push (sjeciste (pravac (MD.verizni[i].pocetak,
                             MD.verizni[i].pocetak+(0,1)),
                    os) + (0, t[t.length - 1]));
  draw (pravac (T[T.length-1],
                sjeciste (os, pravac (T[T.length-1], T[T.length-1]+(0,1)))));

  for(int j = 0; j < MD.verizni[i].A.length-2; ++j) {
    tocka SO = sjeciste (
      pravac (MD.verizni[i].A[j], MD.verizni[i].A[j]+(0,1)), os);
    t.push (t[t.length-1]);
    T.push (SO + (0, t[t.length-1]));
    j=j+1;
    tocka C = MD.verizni[i].A[j] + (a,0);
    pravac s = pravac (C, C+(0,1));
    tocka D = sjeciste (s, pravac (MD.verizni[i].A[j], MD.verizni[i].A[j+1]));
    t.push (-ypart (D.p - C.p) * (k * cm) / a);
    T.push (SO + (0, t[t.length-1]));
  }
  tocka SO = sjeciste (
    pravac (MD.verizni[i].A[MD.verizni[i].A.length-1],
            MD.verizni[i].A[MD.verizni[i].A.length-1] + (0,1)),
    os);
  t.push (t[t.length-1]);
  T.push (SO + (0, t[t.length-1]));
```

```
    tocka C = MD.verizni[i].A[MD.verizni[i].A.length-1] + (a,0);
    pravac s = pravac (C, C+(0,1));
    tocka D = sjeciste (
      s,
      pravac (MD.verizni[i].A[MD.verizni[i].A.length-1],
              MD.verizni[i].A[MD.verizni[i].A.length-1]
                + (MD.verizni[i].p.pol
                  - MD.verizni[i].p.A[MD.verizni[i].p.A.length-1]))
    );
    t.push (-ypart (D.p - C.p) * (k * cm) / a);
    T.push (SO + (0, t[t.length-1]));
    tocka SO = sjeciste (pravac(MD.verizni[i].kraj,MD.verizni[i].kraj + (0,1)),
      os);
    t.push (t[t.length-1]);
    T.push (SO + (0, t[t.length-1]));
    draw (pravac (T[T.length-1],
                  sjeciste (os, pravac (T[T.length-1], T[T.length-1]+(0,1)))));
  }
  for (int i = 0; i < T.length-1; ++i)
    draw (pravac (T[i],T[i+1]));

  draw (pravac (pocetak,
                sjeciste (os, pravac (T[T.length-1], T[T.length-1] + (0,1)))),
        0.3cm, scale (0.9mm) * blue + squarecap);
  TD.T = T;
  TD.t = t;
  return TD;
}
```

Rezultat funkcije je struktura

```
struct Tdijagram {
  tocka[] T;
  real[] t;
};
```

Niz realnih brojeva `t` predstavlja vrijednosti poprečne sile u odabranim točkama. Vrijednosti su prikazane u mjerilu koje odgovara mjerilu momenata savijanja. Niz točaka `T` dobiva se nanošenjem vrijednostî poprečne sile okomito na os dijagrama u odgovarajućim točkama.

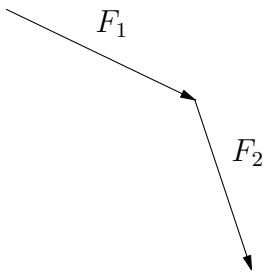
Iznos poprečne sile koji u dijagramu nacrta program GS treba pomnožiti s polnom udaljenosti pripadnoga verižnog poligona da bi ih se dobilo u mjerilu sila. Faktor `k` koji se javlja u pozivu funkcije `Tdijagram()` afino transformira dijagram poprečnih sila, što je ponekad zgodno radi ‚uljepšavanja' crteža. Ako je $k > 1$, dijagram se ‚rasteže', a ako je $k < 1$, dijagram se ‚skuplja'.

7. Primjeri

Primjer 1.

Definirati niz generaliziranih sila: $F_1 = 250 \cdot \vec{\imath} - 120 \cdot \vec{\jmath}$ [kN], $F_2 = 75 \cdot \vec{\imath} - 225 \cdot \vec{\jmath}$ [kN], $M = -200$ kNm. Za definirani niz sila konstruirati poligon sila. Hvatišta sila i početak poligona odabrati po volji. Mjerilo sila neka je 1 cm :: 100 kN.

```
import gs;

gen_sila[] gs = new gen_sila[];
gs[0] = sila (250, -120, (0, 0));
gs[1] = moment (-200, (5, 1));
gs[2] = sila (75, -225, (8, 1));

poligonsila P1 = poligonsila (gs, (12cm, 3cm), cm/100);

draw (P1);
```



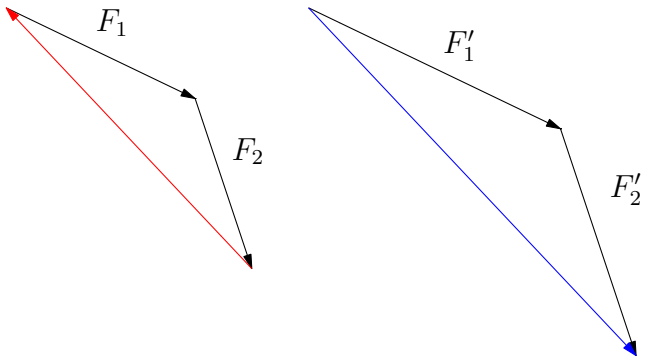
Slika 18. Primjer 1.: poligon sila

Primjer 2.

U poligonu sila iz primjera 1. konstruirati ravnotežnu silu. Za sile iz primjera 1. konstruirati novi poligon sila i u njemu odrediti rezultantu. Mjerilo sila drugoga poligona sila neka je 1 cm :: 75 kN.

```
import gs;

gen_sila[] gs = new gen_sila[];
```

```
gs[0] = sila (250, -120, (0, 0));
gs[1] = moment (-200, (5, 1));
gs[2] = sila (75, -225, (8, 1));

poligonsila P1 = poligonsila (gs, (12cm, 3cm), cm/100);
poligonsila P2 = poligonsila (gs, (16cm, 3cm), cm/75);

draw (P1);
ravnoteza (P1);

draw (P2);
ekvivalencija (P2);
```



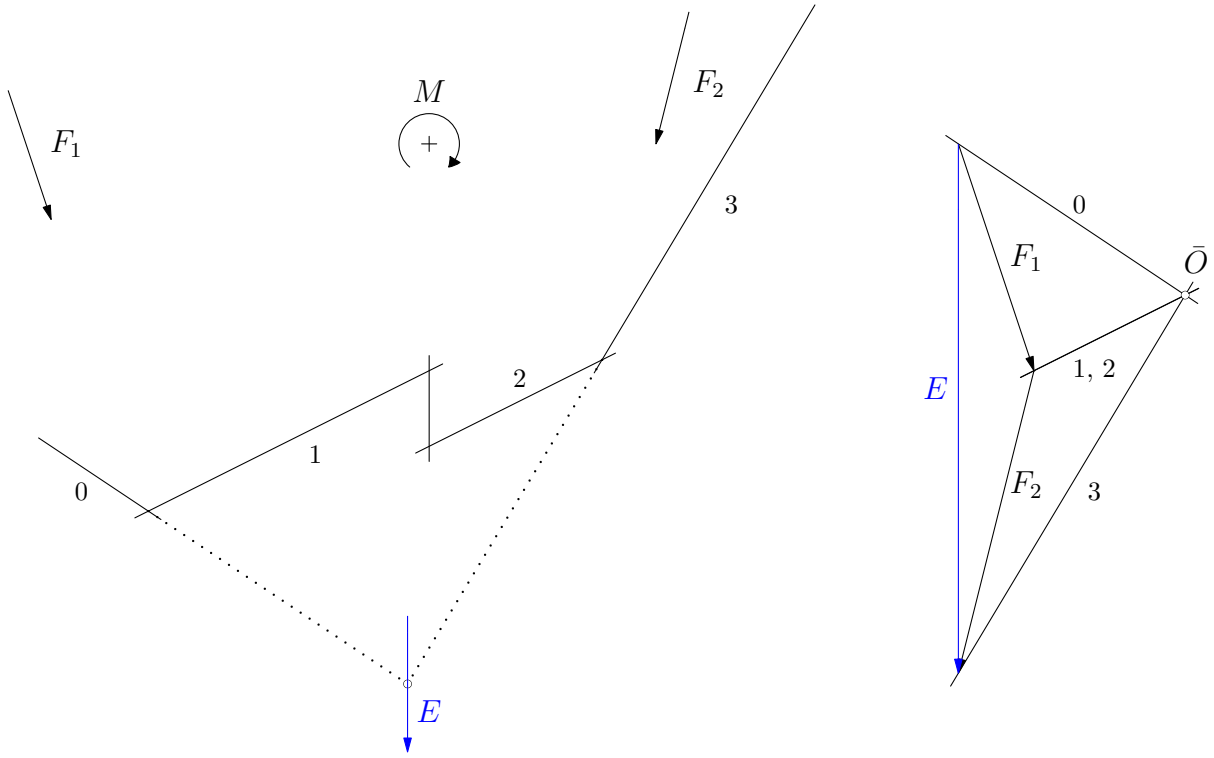
Slika 19. Primjer 2.: ravnotežna i rezultantna sila

Primjer 3.

Zadan je niz generaliziranih sila: koncentrirana sila $F_1 = 100 \cdot \vec{\imath} - 300 \cdot \vec{\jmath}$ [kN] s hvatištem $(0, 0)$, koncentrirani moment $M = -200\,\text{kNm}$ s hvatištem $(5, 1)$ i koncentrirana sila $F_2 = -100 \cdot \vec{\imath} - 400 \cdot \vec{\jmath}$ [kN] s hvatištem $(8, 1)$. Naći rezultantu i uravnotežujuću silu. Mjerilo sila neka je 1 cm :: 100 kN, a mjerilo duljina 1 : 100.

```
import gs;

gen_sila[] gs = new gen_sila[];
gs[0] = sila (100, -300, (0, 0));
gs[1] = moment (-200, (5, 1));
gs[2] = sila (-100, -400, (8, 1));
```

```
poligonsila G = poligonsila (gs, (12cm, 1cm), (15cm, -1cm), cm/100);
verizni V = verizni (G, cm/100, (0, -3cm), (10cm, 0));

draw (V);
draw (gs);
ekvivalencija (V);
```

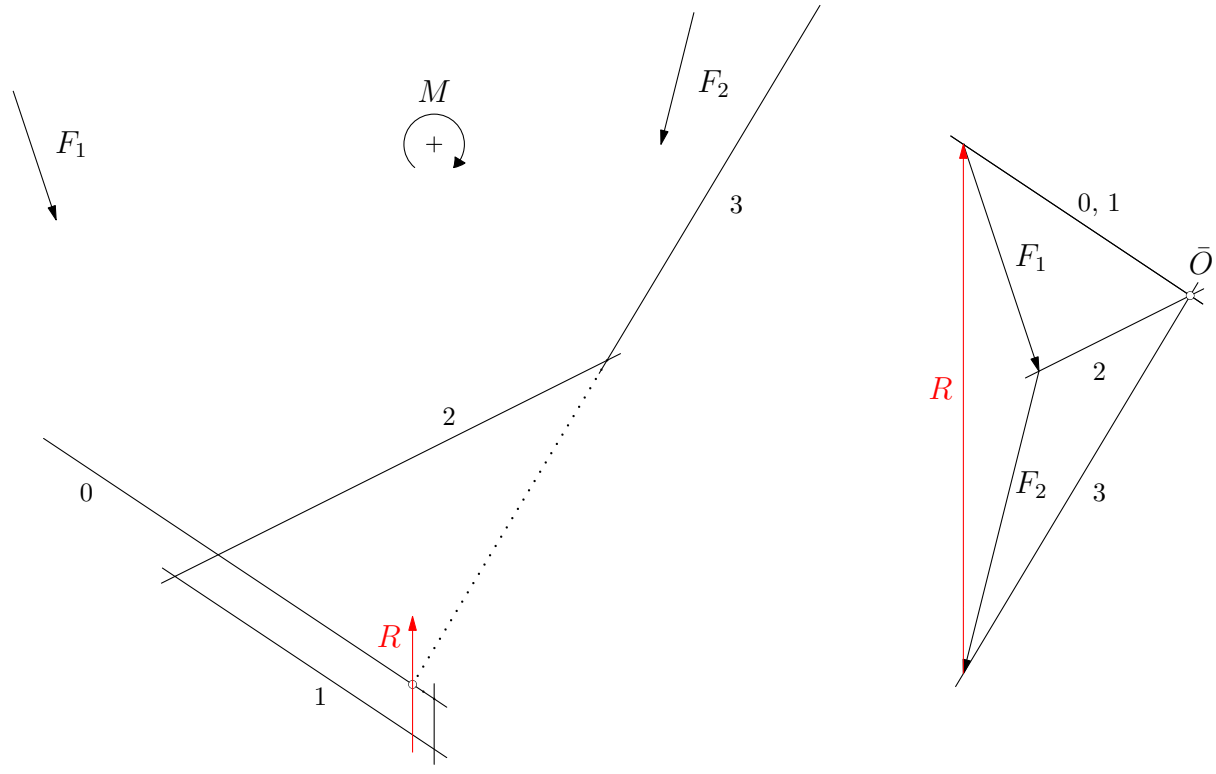


Slika 20. Primjer 3.: konstrukcija rezultante pomoću verižnog poligona

Program konstruira verižni poligon od sile do sile redom kojim su definirane u nizu generaliziranih sila. U verižnom je poligonu naznačen samo pravac djelovanja ekvivalentne sile (strelica nije u mjerilu), dok je njen intenzitet moguće očitati iz poligona sila.

Ravnotežna sila dobiva se umetanjem u kôd naredbe `ravnoteza()` umjesto naredbe `ekvivalencija()`. Uz to je, primjera radi, promijenjen redoslijed sila za koji se konstruira verižni poligon. Pravac djelovanja ravnotežne sile se, naravno, podudara s pravcem djelovanja ekvivalentne sile.

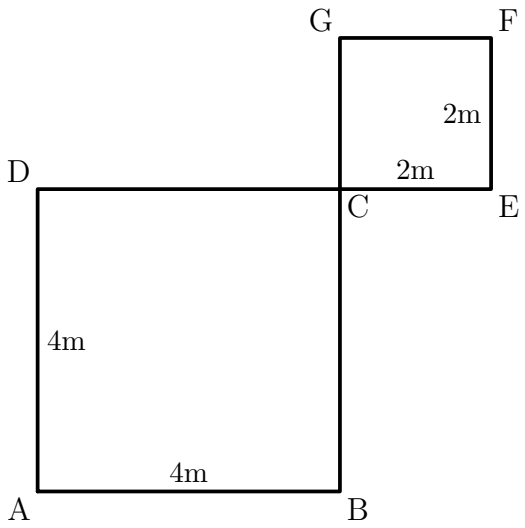
```
import gs;

gen_sila[] gs = new gen_sila[];
gs[1] = sila (100, -300, (0, 0));
gs[0] = moment (-200, (5cm, 1cm));
gs[2] = sila (-100, -400, (8cm, 1cm));

poligonsila G = poligonsila (gs, (12cm, 1cm), (15cm, -1cm), cm/100);
verizni V = verizni (G, cm/100, (0, -3cm), (10cm, 0));

draw (V);
draw (gs);
ravnoteza (V);
```



Slika 21. Primjer 3.: uravnoteženje pomoću verižnog poligona

Primjer 4.

Grafički odrediti položaj težišta homogene ploče sa slike 22. Duljine stranica pravokutnika ABCD su 4 m, a pravokutnika CEFG 2 m. Mjerilo duljina neka je 1 : 100.



Slika 22. Primjer 4.: homogena ploča

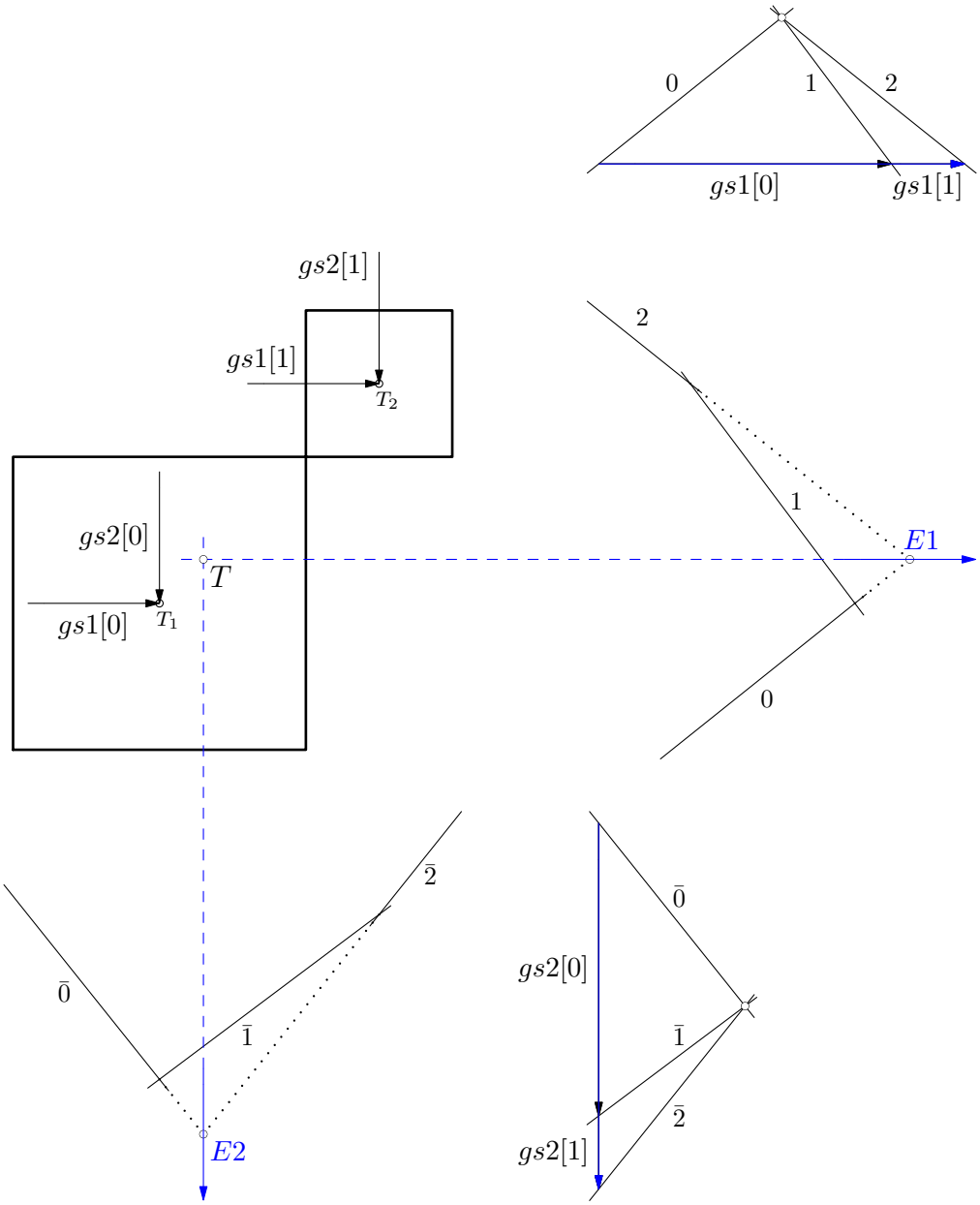
Ploče su zadane i nacrtane sljedećim fragmentom kôda:

```
import gs;

tocka A = (0, 0);
tocka B = A + (4, 0);
tocka C = A + (4, 4);
tocka D = A + (0, 4);
tocka E = A + (6, 4);
tocka F = A + (6, 6);
tocka G = A + (4, 6);

draw (A.p--B.p--C.p--E.p--F.p--G.p--C.p--D.p--A.p, defaultpen + 1.5);
```

Hvatišta sila pomoću kojih se određuje težište lika nalaze se u težištima podsegmenata, a intenzitete sila treba odabrati u odgovarajućim omjerima. Za određivanje težišta potrebna su dva sustava sila:

```
tocka T1 = sjeciste (pravac (A, C), pravac (B, D));
```

```
tocka T2 = sjeciste (pravac (C, F), pravac (E, G));

gen_sila[] gs1 = new gen_sila[];
gs1[0] = sila (400, 0, T1);
gs1[1] = sila (100, 0, T2);
```



Slika 23. Primjer 4.: nalaženje težišta

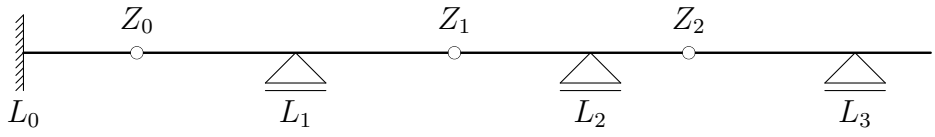
```
gen_sila[] gs2 = new gen_sila[];
gs2[0] = sila (0, -400, T1);
gs2[1] = sila (0, -100, T2);
```

Pravci djelovanja ekvivalentnih sila danih nizova sila određuju se verižnim poligonom. Njihovo sjecište definira traženo težište:

```
poligonsila p1 = poligonsila (gs1, A+(8cm, 8cm), A+(10.5cm, 10cm), cm/100);
verizni v1 = verizni (p1, cm/100, A+(9cm, 0cm), F+(2cm,0));
draw (v1);
draw (p1);
draw (gs1);
ekvivalencija (v1);

poligonsila p2 = poligonsila (gs2, A+(8cm, -1cm), A+(10cm, -3.5cm), cm/100);
verizni v2 = verizni (p2, cm/100, A+(0cm, -2cm), F);
draw (v2);
draw (p2);
draw (gs2);
ekvivalencija (v2);

tocka T = sjeciste (pravac_djelovanja (p1.EKV), pravac_djelovanja (p2.EKV));
```

Primjer 5.

Treba definirati Gerberov nosač sa slike 24.



Slika 24. Primjer 5.: Gerberov nosač

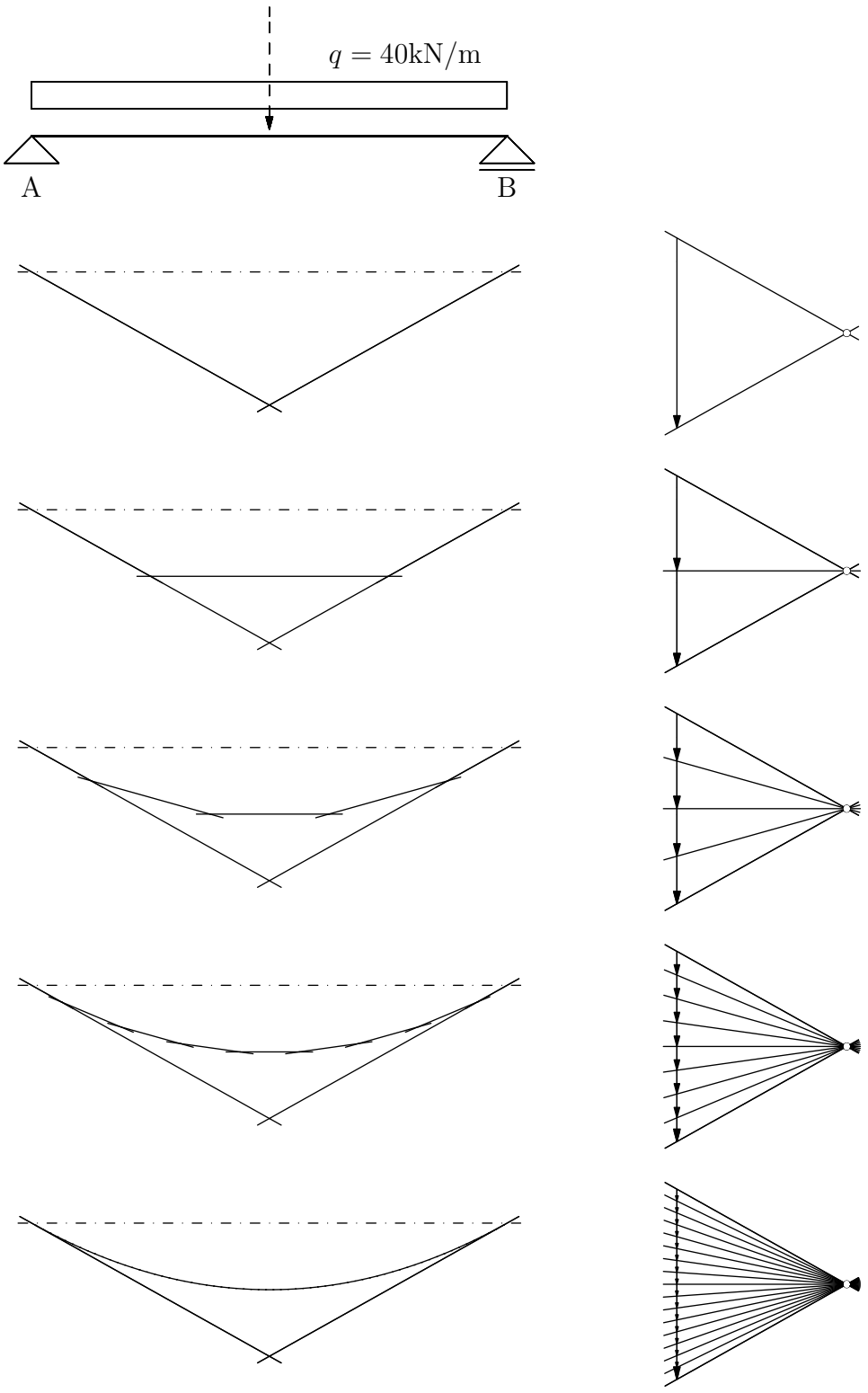
```
import gs;

tocka B = (0,0);
```

```
lezaj[] L1 = new lezaj[];
L1[0] = upeti (B);
L1[1] = pomicni (B + (3.6, 0));
L1[2] = pomicni (B + (7.5, 0));
L1[3] = pomicni (B + (11, 0));

zglob[] Z1 = new zglob[];
Z1[0] = zglob (B + (1.5, 0));
Z1[1] = zglob (B + (5.7, 0));
Z1[2] = zglob (B + (8.8, 0));

greda G1 = greda(B, B + (12, 0), L1, Z1);
draw (G1);
```

Primjer 6.

Definirati prostu gredu raspona 7 m. Neka je greda opterećena jednoliko raspodijeljenim linijskim opterećenjem $q = 40$ kN/m. Konstruirati momentni dijagram pomoću funkcije `Gerber()`.

```
import gs;

tocka A = (0, 0);
tocka B = (7, 0);

lezaj[] L = new lezaj[];
L[0] = nepomicni(A);
L[1] = pomicni(B);

zglob[] Z = new zglob[];

greda G = greda(A, B, L, Z);
draw (G);

gen_sila[] gs = new gen_sila[];
gs[0] = distribuirano (A + (0, 0.4cm), B + (0, 0.4cm), 40, 40, cm/100);
draw (gs);
```



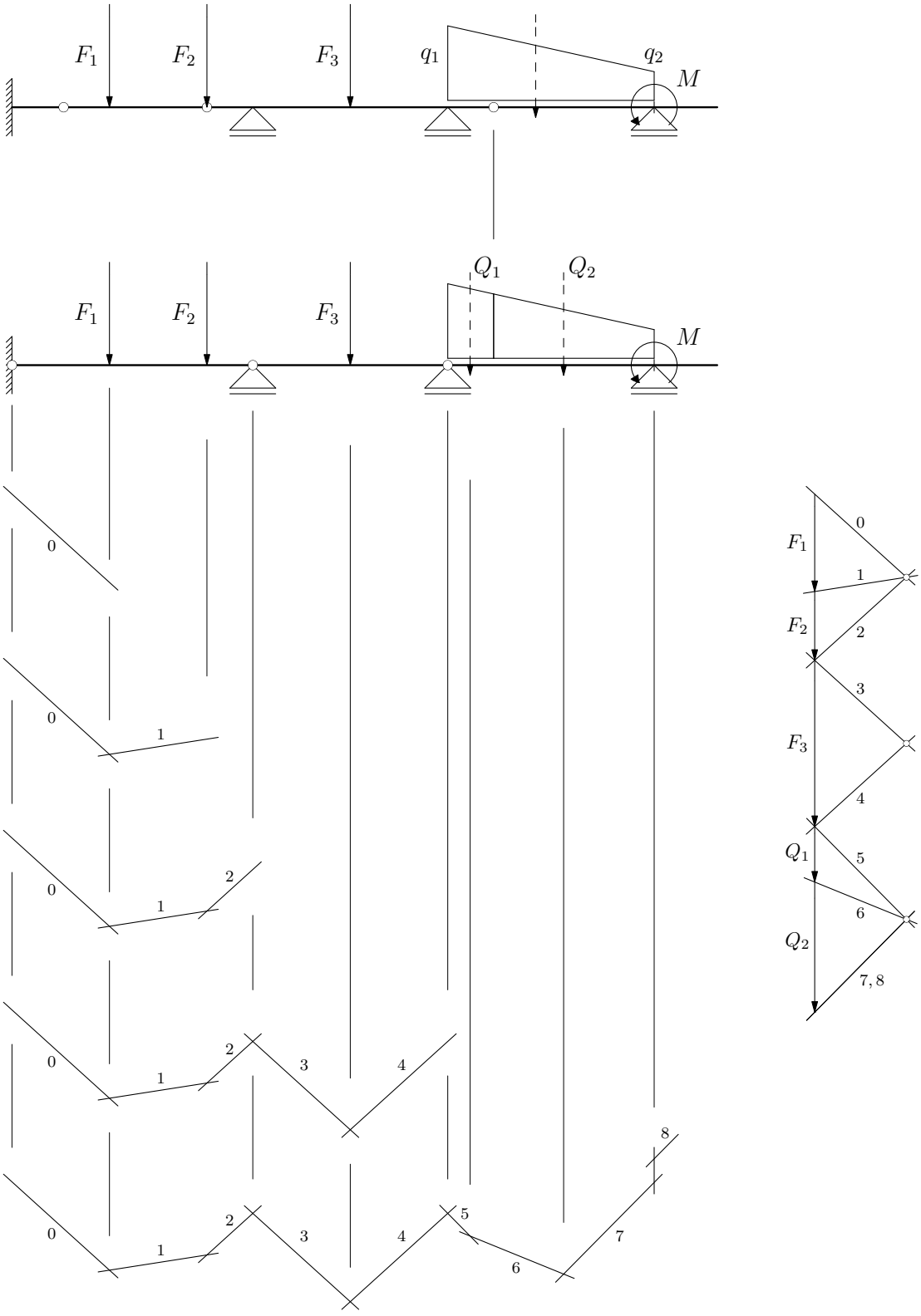
Slika 25. Primjer 6.: Rekurzivno crtanje momentnog dijagrama

Koristeći mogućnosti funkcije `Gerber()`, momentni je dijagram postupno aproksimiran verižnom krivuljom:

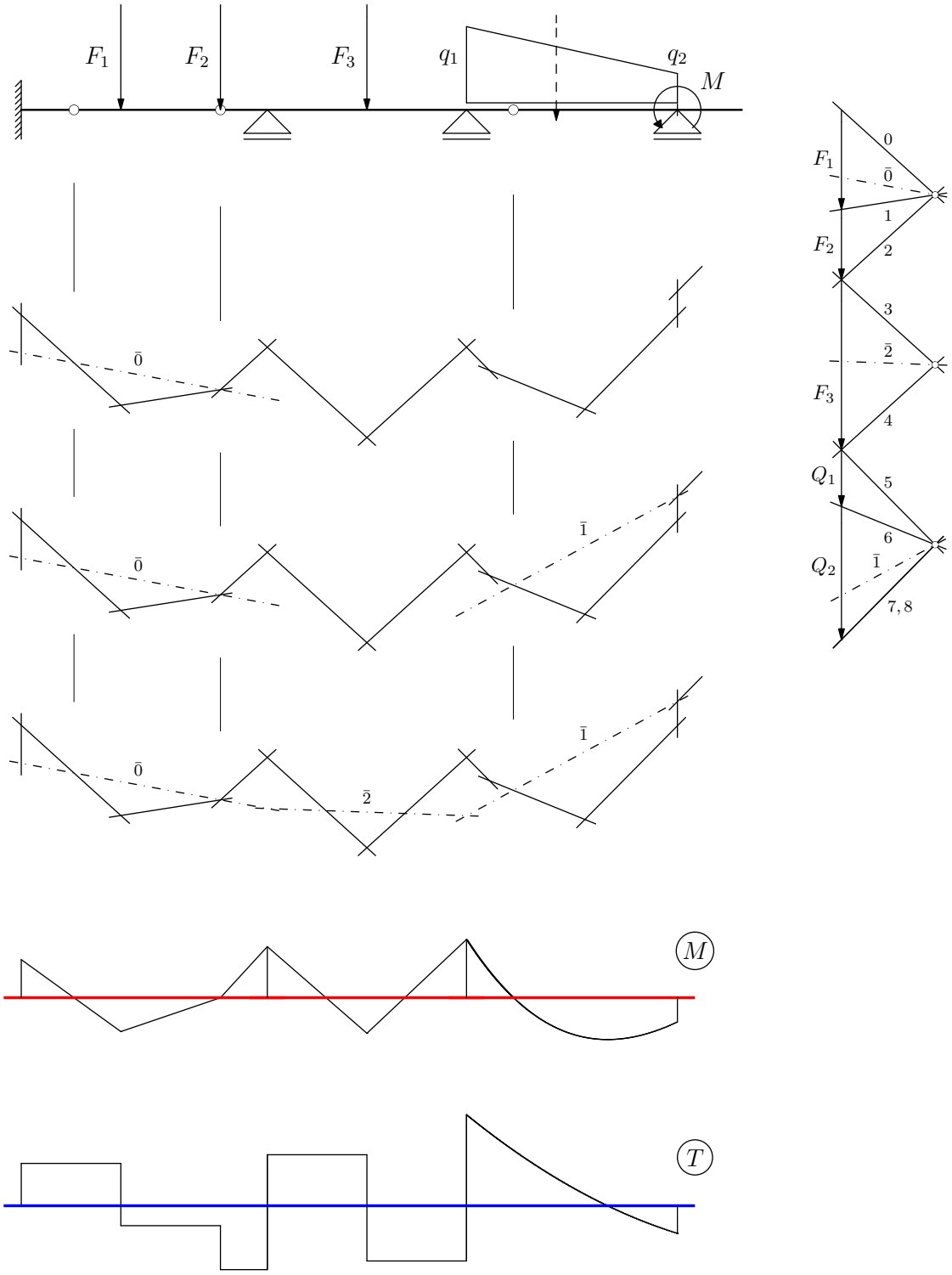
```
gerber PG = new gerber;

PG = Gerber (G, gs, pocetakPsila = A + (9.5cm, -1.5cm), mjerilo = cm/100,
             pocetakVP = A + (0, -2cm), mjeriloD = cm/100,
             polnaudaljenost = 2.5, n = 0, krivulja = false, zakljucna = true);
draw(PG);

PG = Gerber (G, gs, pocetakPsila = A + (9.5cm, -5cm), mjerilo = cm/100,
             pocetakVP = A + (0, -5.5cm), mjeriloD = cm/100,
             polnaudaljenost = 2.5, n = 1, krivulja = false, zakljucna = true);
draw(PG);

PG = Gerber (G, gs, pocetakPsila = A + (9.5cm, -8.5cm), mjerilo = cm/100,
             pocetakVP = A + (0, -9cm), mjeriloD = cm/100,
             polnaudaljenost = 2.5, n = 3, krivulja = false, zakljucna = true);
draw(PG);

PG = Gerber (G, gs, pocetakPsila = A + (9.5cm, -12cm), mjerilo = cm/100,
             pocetakVP = A + (0, -12.5cm), mjeriloD = cm/100,
             polnaudaljenost = 2.5, n = 7, krivulja = false, zakljucna = true);
draw(PG);

PG = Gerber (G, gs, pocetakPsila=A + (9.5cm, -15.5cm), mjerilo = cm/100,
             pocetakVP = A + (0, -16cm), mjeriloD = cm/100,
             polnaudaljenost = 2.5, n = 0, krivulja = true, zakljucna = true);
draw(PG);
```

Primjer 7.

Na slikama 26. i 27. prikazani su koraci postupka rješavanja Gerberovoga nosača primjenom principa superpozicije. Niz crteža, na kojima se postupno sastavljaju poligon sila i verižni poligon te, potom, određuje i ‚ravna' zaključna linija i u tangentni poligon (ako treba) interpolira verižna krivulja, može se lako povezati u animirani prikaz. Nekoliko je animiranih primjera uključeno na priloženi CD. (Jednostavnosti radi, programski kôd koji slijedi daje samo konačni crtež. Pojedini se crteži sastavljanja poligonâ crtaju, naravno, jedan preko drugoga.)



Slika 26. Primjer 7.: koraci u rješavanju Gerberovog nosača



Slika 27. Primjer 7.: koraci u rješavanju Gerberovog nosača (nastavak)

```
import gs;

tocka A = (0,0);
lezaj[] L = new lezaj[];
L[0] = upeti (A + (0, 0));
L[1] = pomicni (A + (4.2, 0));
L[2] = pomicni (A + (7.6, 0));
L[3] = pomicni (A + (11.2, 0));
zglob[] Z = new zglob[];
Z[0] = zglob (A + (0.9, 0));
Z[1] = zglob (A + (3.4, 0));
Z[2] = zglob (A + (8.4, 0));
greda gr = greda (A + (0, 0), A + (12.3, 0), L, Z);
draw (gr);

gen_sila[] gs = new gen_sila[];
gs[0] = sila (0, -170, A + (1.7, 0));
gs[1] = sila (0, -120, A + (3.4, 0));
gs[2] = sila (0, -290, A + (5.9, 0));
gs[3] = distribuirano(A + (7.6, 0.12), A + (11.2, 0.12), 130, 50,
                      mjeriloD = cm/100);
gs[4] = moment (66, A + (11.2, 0));
draw (gs);

tocka t = (0, -4.5cm);
gerber G = Gerber (gr, gs,
                   pocetakPsila = A + (14cm, 0cm), mjerilo = cm/100,
                   pocetakVP = A + 1.5t, mjeriloD = cm/100,
                   polnaudaljenost = 1.6, n = 0,
                   krivulja = true, zakljucna = true);

Mdijagram M = Mdijagram (G, pocetakVP = A + 2t,
              pocetakPS = G.verizni[0].p.A[0], crtajPSila = false);

Tdijagram T = Tdijagram (M, pocetak = A + 3t, k = 1);
```

Pri zadavanju linijskih opterećenja ne treba voditi računa o zglobovima, ležajevima i koncentriranim djelovanjima. Program opterećenja sam prilagođava grafičkom postupku.

8. Zaključak

„The images provide a physical basis for thinking." [1]

„Slika govori tisuću riječi."

Iako o broju i kategorizaciji ljudskih osjetila u znanstvenoj zajednici postoje nesuglasice i oprečna mišljenja, nedvojbeno je da osjetilo vida, sa stanovišta prikupljanja i primanja informacija o okolini i o pojavnostima oko nas u svim svojim oblicima ima vodeću ulogu. Direktno u vezi s tim je i način na koji čovjek uči, usvaja spoznaje i razmišlja [11].

Moglo bi se ustvrditi da je inženjerski način razmišljanja, ‚inženjerski mozak' po svojoj biti zapravo grafički. Opis fizikalnih pojava i procesa gotovo uvijek prati i grafička reprezentacija i interpretacija danog problema, a katkad je takva reprezentacija i jedina. Grafičko komuniciranje dio je inženjerske svakodnevnice i praktično je nezamjenjivo.

Jedna od prednosti slika nad tekstom je i u količini informacija koje je ljudski mozak u stanju primiti i obraditi u gradijentu vremena. Na primjer, jedan pogled na momentni dijagram može otkriti potencijalne nedosljednosti u rješenju, pa i slabosti u koncepciji konstrukcije, dok jedan pogled na tablični prikaz numeričkih vrijednosti unutarnjih sila u čvorovima elemenata bez drugog (drugih) pogleda i ne otkriva baš previše. Iz navedenoga slijedi da je ‚bijeg' u grafiku prirodan i nužan.

Primarna namjena oblikovanoga programa GS je implementacija grafičkih postupaka statike štapnih konstrukcija na računalu, što je njime uvelike olakšano. Uporabom programa moguća je elegantna provedba grafičkih postupaka, s prikazom kako rješenja tako i postupka kojim se do njega došlo. Kao mogućnost kontrole ostavljen je i numerički prikaz rezultata.

Uočavanje bitnih elemenata u postupku nalaženja rješenja, te njihovog utjecaja na konačan oblik rješenja od iznimne je važnosti. Nalaženjem rješenja grafičkim putem, postupno korak po korak, vodeći pritom računa o utjecaju pojedinih radnji na oblik prikaza rješenja i u skladu s tim organizirajući radni prostor, tj. papir na kojem nastaje rješenje, iskazuje se fizikalna interpretacija rješenja, te se dobiva dublji uvid i omogućava se bolje razumijevanje ponašanja konstrukcije i toka sila u njoj — *conditio sine qua non* za imalo ozbiljnija daljnja razmatranja i ingeniozna rješenja kompleksnijih problema.

Prikaz rješenja korak po korak omogućava primjena kompjutorske animacije. Primjenom programa GS animacije se mogu jednostavno izraditi; primjeri su na priloženom CD-u.

Na priloženom CD-u nalazi se i sav sadržaj potreban za primjenu programa GS — programski jezik `Asymptote`, te izvorni kod programa GS. Također su priloženi i svi primjeri iz ovog rada koji mogu poslužiti kao osnova za rješavanje daljnjih problema.

[1] D. McCabe, A. Castel: Seeing is believing: The effect of brain images on judgments of scientific reasoning, *Cognition* (in press)

Literatura

[1] E. Cassirer: *The Individual and the Cosmos in Renaissance Philosophy*, Harper and Row, New York, 1963.; ponovljeno izdanje: Dover, New York, 2000.

[2] E. J. Dijksterhuis (ed.): *The Principal Works of Simon Stevin. Volume I. General Introduction. Mechanics*, C. V. Swets & Zeitlinger, Amsterdam, 1955.; dostupno i na `http://www.library.tudelft.nl/ws/services/tresor/digitalworks/stevin/index.htm`, pristupljeno: 2. svibnja 2008.

[3] R. Dugas: *A History of Mechanics*, Éditions du Griffon, Neuchâtel, 1955.; ponovljeno izdanje: Dover, New York, 1988.

[4] P. Duhem: *History of Physics*, Catholic Encyclopedia, R. Appleton, New York, 1911., str. 47–67; dostupno i na `http://www.newadvent.org/cathen/12047a.htm`, pristupljeno: 2. svibnja 2008.

[5] K. Fresl: *Građevna statika 1.: bilješke i skice s predavanja*, `http://www.grad.hr/nastava/gs/bilj1/index.html`, pristupljeno: 2. svibnja 2008.

[6] A. Hammerlindl, J. Bowman, T. Prince: *Asymptote: the Vector Graphics Language*, version 1.34svn, `http://asymptote.sourceforge.net/`, pristupljeno: 2. svibnja 2008.

[7] J. D. Hobby: *A User's Manual for METAPOST*, `http://www.tug.org/metapost.html`, pristupljeno: 2. svibnja 2008.

[8] J. D. Hobby: *The METAPOST System*, `http://www.tug.org/metapost.html`, pristupljeno: 2. svibnja 2008.

[9] D. E. Knuth: *The METAFONTbook*, Addison–Wesley Publishing Company, Reading, Massachusetts, 1991.

[10] L. Lamport: *LaTeX — A Document Preparation System*, Second Edition for LaTeX 2ε, Addison–Wesley Publishing Company, Reading, Massachusetts, 1994.

[11] L. F. Lowery: *The Biological Basis for Thinking and Learning*, Lawrence Hall of Science, University of California, Berkeley, 1998.; dostupno i na `www.lhs.berkeley.edu/foss/newsletters/archive/pdfs/FOSS_BBTL.pdf`, pristupljeno: 6. svibnja 2008.

[12] H. Müller–Breslau: *Die Graphische Statik der Baukonstruktionen. Erster Band*, Fünfte, vermehrte Auflage, Alfred Kröner Verlag, Leipzig, 1912.

[13] H. Müller–Breslau: *Die Graphische Statik der Baukonstruktionen. Zweiter Band. I. Abteilung*, Fünfte, vermehrte Auflage, Alfred Kröner Verlag, Stuttgart, 1922.

[14] H. Müller–Breslau: *Die Graphische Statik der Baukonstruktionen. Zweiter Band. II. Abteilung*, Fünfte, vermehrte Auflage, Alfred Kröner Verlag, Stuttgart, 1925.

[15] V. Niče: *Uvod u sintetičku geometriju*, Školska knjiga, Zagreb, 1956.

[16] R. Sethi: *Programming Languages*, Addison–Wesley Publishing Company, Reading, Massachusetts, 1989.

[17] V. Simović: *Građevna statika I*, Građevinski institut, Zagreb, 1988.

[18] B. Stroustrup: *The C++ Programming Language*, Third Edition, Addison–Wesley Publishing Company, Reading, Massachusetts, 1997.

[19] S. Timoshenko, D. H. Young: *Enginnering Mechanics*, Fourth Edition, McGraw–Hill Book Company, New York, 1956.

[20] H. Werner: *Mehanika I — Statika*, Hrvatski savez građevinskih inženjera, Zagreb, 2007.

Sažetak

Nenad Bijelić

Kompjutorska grafostatika

Predmet rada su grafički postupci u statici i njihova realizacija na računalu. Izrađen je računalni program nazvan GS koji određene statičke probleme rješava primjenom grafičkih postupaka. Rješenja se prikazuju u grafičkom obliku, uz mogućnost numeričkog prikaza rezultata. Obrađeni su svi osnovni postupci te njihova primjena na rješavanje složenijih problema. Poseban naglasak stavljen je na primjenu principa superpozicije.

Ključne riječi: grafostatika, objektno usmjereno programiranje, vektorska grafika, superpozicija, statički određeni nosač

Summary

Nenad Bijelić

Computational Graphostatics

Paper deals with graphical procedures in statical analysis and their computer realization. Computer programme GS which solves various problems by graphical methods was designed and implemented. Solutions are presented graphically with the possibility of numerical output. All simple procedures are covered, including their application on more complex problems. A special emphasis is placed on the usage of superposition principle.

Key words: graphostatics, object–oriented programming, vector graphics, superposition, statically determined structure

O autoru

Nenad Bijelić rođen je 19. rujna 1985. u Zagrebu gdje je 2004. godine završio Prirodoslovno–matematičku gimnaziju (XV. gimnazija). Iste je godine upisao studij na Građevinskom fakultetu Sveučilišta u Zagrebu. Redoviti je student četvrte godine. Dobitnik je nagrade najuspješnijim studentima Građevinskoga fakulteta za akademske godine 2004./2005., 2005./2006. i 2006./2007. te nagrade Konstruktor–inženjeringa d. d. iz Splita najboljem studentu četvrte godine Građevinskog fakulteta u Zagrebu za 2007. godinu.