Laboratorijske vaje pri predmetu Postopki konstruiranja

Leon Kos, Laboratorij za CAD - LECAD

1 Naloga

Izdelajte računalniški program za izračun statičnih karakteristik (ISK) prerezov poljubnih oblik popisanih z daljicami in loki.

Program naj omogoča izračun veličin:

- površina $A = \int_A dxdy$
- težišči
$$x_T = \frac{1}{A}\int_A xdxdy$$
$$y_T = \frac{1}{A}\int_A ydxdy$$
- Težiščna vztrajnostna momenta
$$I_{x_T} = \int_A y^2 dxdy$$
$$I_{y_T} = \int_A x^2 dxdy$$
- Težiščna odpornostna momenta
$$W_{x_T} = \frac{I_{x_T}}{e_{y_max}}$$
$$W_{y_T} = \frac{I_{y_T}}{e_{x_max}}$$
- Vztrajnostna momenta za poljubno os z uporabo *Steinerjevega pravila*
$$I_x = I_{x_T} + a^2 A$$
$$I_y = I_{y_T} + b^2 A$$

Za izračun karakteristik uporabite metodo **Monte Carlo** s transformacijo spektralne gostote generatorja naključnega šuma. Natančnost metode kontrolirajte z oceno napake (1%) in maksimalnim dovoljenim številom naključnih točk.

Obliko prereza skonstruirajte s programom Autocad iz katerega posredujete podatke o geometriji s predprocesorjem (program v Autolispu) v program za izračun statičnih karakteristik.

V drugem delu naloge modificirajte obstoječi program ISK tako, da bo pred izračunom s homogenimi transformacijami postavil lik v poljubno orientacijo in zaj izračunal potrebne karakteristike. Transformacija mora biti izvedena s kompozicijo matrik za rotacijo, skaliranje in translacijo.

Po transformaciji in izračunu naj program omogoči prikaz prereza v novi legi za kar uporabite grafični jezik DGL ali standardni grafični jezik GKS (level 1a).

Programa morata biti ločeni zaključeni celoti (osnovni in modificirani program). Za vsak program izdelajte poročilo z opisom delovanja programa, omejitvami, teoretičnimi osnovami in rezultati izračuna.

2 Metoda Monte Carlo

Uporaba metode Monte Carlo se omejuje na probleme, kjer imamo komplicirane robne pogoje in zahtevne funkcije, ki se ne dajo enostavno integrirati. Sama metoda je zelo enostavna vendar računsko zelo potratna za zahtevano natančnost izračuna. Tako se za tak pristop k reševanju odločamo le v skrajni sili, ko želimo približno oceno in nimamo časa (znanje) za primernejše metode.

Oglejmo si primer reševanja površine kroga z radijem $r = 1$.

$$A = \int_A dxdy \qquad x^2 + y^2 \leq r^2$$

Analitična rešitev je seveda $A = \pi r^2$. Naš krog premera $d = 2$ omejimo z najmanjšim pravokotnikom za katerega seveda znamo izračunati površimo in v katerem še stoji krog; torej $A_{kv} = 2 \times 2 = 4$. Imejmo generator naključnih točk, ki zagotavlja, da vsaka generirana točka gotovo pade v pravokotnik.

Generator tako izbere recimo N točk (x, y). Za vsako izbrano točko pa lahko ugotovimo ali je točka v notranjosti kroga ali zunaj. Po razsipanju n točk smo tako ugotovili, da je sn točk padlo znotraj kroga, ostale pa so padle zunaj kroga vendar še vedno v kvadrat znanih dimenzij. Površino kroga laho tako brez poznavanja števila π ocenimo kot

$$A \simeq A_{kv}\frac{sn}{N} \tag{1}$$

Primer: Program je posejal 100000 točk v kvadrat in ugotovil, da jih je 78589 padlo v krog. Površina kroga je tako

$$A = 4\frac{78589}{100000} = 3.1436$$

Evidentno je, da z večanjem števila točk dosežemo tudi dovolj natančne rezultate, vendar je za pridobitev dodatne točne cifre potrebno vsaj deset krat več časa. Napako metode je možno oceniti z

$$Error = \pm A\sqrt{\frac{<f^2> - <f>^2}{N}} \tag{2}$$

kjer puščičasti oklepaji predstavljajo aritmetičo sredino preko N testnih točk.

$$<f> = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \qquad <f^2> = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i) \tag{3}$$

Primer programa v jeziku Fortran je sledeč

```
program Povrsina kroga
n = 100000
a = 2.0 * 2.0
sn = 0.0
varn = 0.0
do 10 i = 1, n
```

```
      x = 2 * ran2(idum) - 1.0
      y = 2 * ran2(idum) - 1.0
      if ((x*x + y*y) .lt. 1.0) then
         sn = sn + 1
         varn = varn + 1
      endif
10    continue
      print *, 'Povrsina je kroga je:', sn/n*a
      print *, 'Ocenjena napaka je:',
     *   a*sqrt((varn/n-(sn/n)**2)/n)
      end
```

V zanki 10 programa generirano z naključnim generatorjem `ran2()` števila v obsegu $[0..1]$ in jih z množenjem in translacijo prestavimo v željeno območje $[-1..1]$. Tako generirane točke (x, y) testiramo na robne pogoje (krožnica) in ugotavljamo zadetek točke v notranjst kroga. Istočasno računamo še oceno napake za izbrano število točk.

Kot generator naključnih števil lahko uporabimo funkcijo random, če je ta prikjučena v jeziku Fortran ali pa uporabimo generator `ran2()`, ki deluje v vsaki izvedenki jezika:

```
      function ran2(idum)
      parameter(m=714025, ia=1366)
      parameter(ic=150899, rm=1./m)
      dimension ir(97)
      if (idum.lt.0.or.iff.eq.0) then
         iff = 1
         idum = mod(ic - idum, m)
         do 11 j = 1, 97
            idum = mod(ia*idum + ic, m)
            ir(j) = idum
11       continue
         idum = mod(ia*idum + ic, m)
         iy = idum
      endif
      j = 1 + (97 * iy)/m
      if (j.gt.97.or.j.lt.1) pause
      iy = ir(j)
      ran2 = iy*rm
      idum = mod(ia*idum + ic, m)
      ir(j) = idum
      return
      end
```

2.1 Vztrajnostni moment kroga

$$I_x = \int_A y^2 dxdy \qquad x^2 + y^2 \leq r^2 \tag{4}$$

Če uporabimo analogijo pri izračunu površine kroga in enačbo za vztranjostni moment opazimo, da je potrebno vse točke, ki padejo znotraj kroga množiti še s kvadratom oddaljenosti od osi in tako dobimo popravljen program, ki nam računa vztrajnostni moment kroga

$$I_x = \frac{\pi d^4}{64} \qquad I_x(2) = \frac{\pi}{4} \tag{5}$$

```
      program Ix Kroga
      n = 100000
      a = 2.0 * 2.0
      sn = 0.0
      varx = 0.0
      do 10 i = 1, n
         x = 2.0 * ran2(idum) - 1.0
         y = 2.0 * ran2(idum) - 1.0
         if ((x*x + y*y) .lt. 1.0) then
            sn = sn + y*y
            varn = varn + y**4
         endif
10    continue
      print *, 'Vztr. moment kroga je:', sn/n*a
      print *, 'Ocenjena napaka je:',
     *   a*sqrt((varn/n-(sn/n)**2)/n)
      end
```

Ugotovimo lahko, da točke, ki so bliže težišču ne prispevajo bistveno k natančnosti izračuna, saj bistveni delež k vztrajnstnem momentu prispevajo točke, ki so bolj oddaljene od težišča. Ugodno bi torej bilo, če bi pozornost posvetili oddaljenim točkam in ne zgubljali časa okoli težišča. V ta namen moramo bolj gosto sejati točke ob spodnjem in zgornjem robu.

Za integral (4) uvedimo substitucijo

$$sd = y^2 dy \qquad s = \frac{y^3}{3} \qquad y = \sqrt[3]{3s}$$

Integral se zdaj prevede na

$$A = \int_A dsdx$$

kar je podobno izračunu površine, le da so se meje za y spremenile iz $y = [-1..1]$ v $s = [-\frac{1}{3}..\frac{1}{3}]$ in s tem tudi celotna površina omejitvenega pravokotnika, ki je zdaj $2 \times 2/3]$. S to substitucijo smo spremenili spektralno gostoto generatorja belega šuma v funkcijsko gostoto, pri kateri so točke ki so bolj oddaljene od težišca tudi bolj gosto posejane. Za testiranje meje pa uporabimo isto formulo kot prej, le da smo y izračunali z inverzno transformacijo. Program s transformacijo gostote je sledeč:

```
      program Ix Kroga
      n = 10000
      a = (1.0 - (-1.0)) * (0.333 - (-0.333))
      sn = 0.0
      varx = 0.0
      ss = 1.0/3.0 - (-1.0/3.0)
      print *, -1.0**(1.0/3.0)
      do 10 i = 1, n
         x = 2.0 * ran2(idum) - 1.0
         s = -1.0/3.0 + ss * ran2(idum)
         y = (abs(3.0 * s))**(1.0/3.0)
         if ((x*x + y*y) .lt. 1.0) then
            sn = sn + 1
            varn = varn + y**4
         endif
10    continue
      print *, 'Vztr. moment kroga je:', a*sn/n
      print *, 'Ocenjena napaka je:',
     *   a*sqrt((varn/n-(sn/n)**2)/n)
      end
```

3 Konstrukcija prereza

S programom Autocad bomo narisali prereze, ki jih bomo kasneje izračunali s programom ISK. V Autocadu bomo uporabljali le daljice (`LINE`) in loke (`ARC`). Liki zgrajeni iz teh dveh primitivov so lahko zelo kompleksni z večkratnimi luknjami v prerezu.

S programom v Autolispu pa podatke risbe izpišemo v dve datoteki `lines.txt` in `arcs.txt`, ki jih uporabimo pri izračunu karakteristik.

Vsaka od navedenih datotek ima v prvi vrstici število gradnikov, ki sledijo. Daljice imajo tako v vsaki vrstici koordinato x in y prve in druge točke daljice. Loki pa imajo najprej točko središča, nato pa še radij, začetni in kočni kot loka.

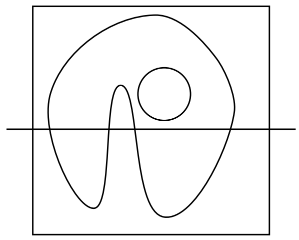
Program v lispu naložim v Autocad z ukazom `(load "post")` ter ga izvedemo z ukazom `post`. Če program preimenujemo v `acad.lsp` se bo ob vsakem zagonu Austocad-a avtomatsko naložil.

```
;; Program naredi dve datoteki LINES.TXT
;; in ARCS.TXT, ki vsebujejo podatke o
;; crtah in lokih pobranih iz baza Acad-a
;; (C) 1994 Leon Kos

(defun c:post( / f i ss ent lines arcs)
   (defun pp(dxf ent / p)
     (setq p (assoc dxf (entget ent)))
     (princ (cadr p)  f) (princ " " f)
     (princ (caddr p) f) (princ "    " f)
   )
   (defun pv(dxf ent / p)
     (setq p (assoc dxf (entget ent)))
     (princ (cdr p) f) (princ " " f)
   )
   (setq i 0 arcs 0 lines 0)
   (setq f (open "LINES.TXT" "w"));dat.daljic
   (setq ss (ssget "X" (list ’(0 . "LINE"))))
   (if (not (null ss))
     (progn
        (princ (setq lines (sslength ss)) f)
        (princ " \n" f)
        (while (< i lines)
           (setq ent (ssname ss i))
           (pp 10 ent) (pp 11 ent)
           (princ "\n" f)
           (setq i (1+ i))
        )
     )
     (princ "0 \n" f)
   )
   (close f)
   (setq f (open "ARCS.TXT" "w"));dat. lokov
   (setq ss (ssget "X" (list ’(0 . "ARC"))))
   (if (not (null ss))
     (progn
       (princ (setq arcs (sslength ss) i 0)f)
       (princ " \n" f)
       (while (< i arcs)
          (setq ent (ssname ss i))
          (pp 10 ent) ; center
          (pv 40 ent) ; radij
          (pv 50 ent) ; zac. kot
          (pv 51 ent) ; koncni kot
          (princ "\n" f)
          (setq i (1+ i))
       )
     )
     (princ "0 \n" f)
   )
   (close f)
   (princ "Risba vsebuje ") (princ lines)
   (princ " daljic in ")    (princ arcs)
   (princ " lokov.")        (princ)
)
```

4 Kontrola notranjosti lika

Prerez sestavljen iz daljic in lokov mora biti sestavljen iz zaprtih likov poljubnih oblik, ki pa se lahko tudi prekrivajo. Za podano geometrijo s programom ugotovimo najmanjši pravokotnik, ki še omejuje celoten lik. To naredimo tako, da poiščemo minimalne in maksimalne koordinate vseh točk daljic in lokov na osi x in y. Nortranjost



Slika 1: Ugotavljanje notranjosti lika

lika ugotovimo tako, da za izbrano točko (x, y) štejemo koliko daljic od vseh možnih prereže daljico od robu zunanjega pravokotnika do izbrane točke. Če je število takih prerezov daljice liho potem je točka v notranjosti lika, sicer je v zunanjosti.

5 Grafični prikaz prereza

Za prikaz transformacije je možno uporabiti jezik DGL ali standardni GKS. V ta namen uporabite homogene matrike in homogene transformacije.

Osnovni program, ki demonstrira uporabo jezika GKS je:

```
      program GKS skeleton
      implicit none
      real x(3)  /-200.0, 350.0, 250.0/
      real y(3)  /-200.0, 300.0,-250.0/
      real kx(2) /0.0, 100.0/
      real ky(2) /50.0, 120.0/
      character*80 datrec(1)

c     Odpre GKS (Gks OPen gKS)
      call gopks()
c     Nastavitev zaslonskih koordinat
c     (Gks Set WorKstation ViewPort)
      call gswkvp(1, 20.0, 50.0, 600.0, 450.0)
c     Nastavitev uporabniskega koord. sistema
c     (Gks Set WiNdow)
      call gswn(1, -300.0, -300.0, 400.0, 400.0)
c     Izris lika z dvema crtama
c     (GksSetPolyLineColorIndex, GksPolyLine)
c     Indeksi:1 modra, 2 zelena, 3 cyan, 4 rdeca,
c             5 violicna, 6 rjava, 7 bela, 8 siva,
c             9 svetlomodra, ..., 14 rumena, 15
      call gsplci(3)
      call gpl(3, x, y)
c     Izris teksta (GksSetCharacterHeight,
c     ...TextColorIndex, GksTeXt)
      call gstxci(14)
      call gschh(20)
      call gtx(-250.0, 300.0, ’GKS skeleton’)
c     Izris kroga
c     (Generalized Drawing Primitive)
      call gdp(2, kx, ky, -1, 1, datrec)
c     Zapre GKS (Gks CLose gKS)
```

```
call gclks()
end
```

6 Poročilo

Poročilo se oddaja v dveh delih, enega za prvi program, drugega na koncu. V poročilu opišite problem, ki ga rešujete (zahteve), podajte teoretične osnove in bistvene dele rešitve, ki ste jih uporabili v programu.

Izpisa programa ni potrebno prilagati poročilu, potrebno pa je predstaviti pomembne dele programa na način, ki čimbolj jasno predstavlja rešitev. To je lahko z besednim opisom, diagramom poteka ali z delom kode v fortranu. Priložite tudi rezultate izračuna s skico prerezov, ki ste jih izračunali.

Celotno poročilo naj ima klasično obliko (uvod, zahteve, teoretične osnove, problemi, rezultati, možne izboljšave, zaključek, literatura, priloge). Izberite si primer mehanskega in na njem pokažite reševanje problema. Diskutirajte izboljšave in slabosti programa, kot tudi omejitve, ki so posledica vgrajenih algoritmov (stabilnost, ...). Torej: poročilo mora biti napisano tako, da je možno iz njega rekonstrurirati program.

Pišite čitljivo; priporočen je izpis s tiskalnikom. Ovojnico za poročilo dobite pri vratarju (Računalniško podprto konstruiranje).

7 Delo na računalniku

Na računalniku PC je instalirano programsko okolje s fortranskim prevajalnikom po standardu F77. Okolje je integrirano in lahko deluje v okolju DOS ali Windows *Programmers WorkBench*. Integrirano okolje z požene z ukazom pwb ime.for v katerem lahko program popravljamo, prevajamo, zaganjamo in iščemo napake. Osnovni ukazi so dosegljiv s pritiskom na tipko skupaj s tipko **ALT**. Tako je izhod iz **PWB ALT+F,X**, Prevajanje in zaganjanje programa z ukazom **Alt+R,E,B,R**. Za pomoč lahko uporabite tipko **F1**, ki je kontekstno občutljiva.

8 Fortran

Tu bom prikazal standardne posebnosti v Fortranu, ki jih boste s pridom uporabili pri programiranju. Priporoča se tipkanje programa z malimi črkami, saj je tako kodiran program bolj čitljiv.

Formatni stavek v funkciji write

```
character ime*65
write(*, '('' Ime vhodne datoteke: '',$)')
read(*,'(A)') ime
```

Pri vseh branjih in pisanjih se lahko testira napaka ob konverziji ali napaka na vhodno/izhodno enoti s parametrom err. Primer:

```
    ...
    read(line(6:80), '(I10)', err=102) lines
    ...
    return
102 stop 'Napaka pri branju parametra lines!'
    end
```

Fortran omogoča pisanje in branje v svoje interne enote v obliki `character` polja. To nam omogoča pretvarjanje numeričnih spremenljivk v znakovne. Največkrat se to uporablja za podprograme, ki zahtevajo kot parameter znakovno spremenljivko, na voljo pa imamo le numerično spremenljivko. Tipičen primer uporabe je izpis teksta v jeziku GKS. Primer:

```
character cnode*3
integer node
...
write(cnode, '(I3)') node
call gtx(x, y, cnode)
...
end
```

8.1 Spolšna priporočila za delo

Noben program ne deluje na mah. Ker je problem kompleksen se ga je potrebno lotiti sistematično. Posamezne dele program moramo sprotno kontrolirati. Začnemo z majhnim (enodimenzionalnim) sistemom, preverimo delovanje in šele nato dodamo več dimenzij. Pomembno je tudi dokazovanje pravilnosti delovanja, kjer si logično razlagamo obnašanje programa za različne vhodne parametre. Švejkova metoda pri problemih kot je naš nikoli ni bila uspešna.

Fortran ima to (ne)ugodno lastnost, da so vse spremenljivke ob začetku postavljene na nič. Ker je naš program do neke mere iterativen, ne pozabite brisati spremenljivk!

Po končanem delu brišite `del` vse nepotrebne datoteke, ki jih imate na svojem področju. Dateteke simulacij znajo biti velike preko 1 MB, kar pri velikem številu uporabnikov in majhnem disku prav rado zapolni disk mrežnega serverja.

Literatura

[1] Kraut Bojan. *Strojniški priročnik.* Strojniški vestnik, Ljubljana, 1981.

[2] Divjak J. Dovedan Z. *Fortran 77 s tehnikama programiranja.* Zveza organizacij za tehnično kulturo Slovenije, Ljubljana, 1988.

[3] Kansy K. Plaff G. Enderle G. *Computer Graphics Programming.* Springer-Verlag, New York, 1984.

[4] William H. et.al. *Numerical Recipes in Fortran; The Art of Scientific programing.* Cambridge university press, Cambridge, 1989.

Laboratorij za CAD, Soba N-17. Govorilne ure za študente so od ponedeljka do četrtka od 13^{00} do 14^{00}. Prosimo, da se držite razporeda urnika. Nadomeščanje zamujenih laboratorijskih vaj načeloma ni možno zaradi zasedenosti računalniške učilnice. Pred vajami na računalniku si pripravite gradivo za delo, da ne boste gledali v zrak (k sosedu).

Ljubljana 9. oktober 1995

Typeset in LaTeX