Laboratorijske vaje pri predmetu Računalniško Podprto Konstruiranje

Leon Kos in Simon Kulovec, Laboratorij za CAD - LECAD

Naloga

Izdelajte računalniški program za preračun prostorskih statično nedoločenih paličnih konstrukcij in program za prikaz rezultatov izračuna z uporabo grafičnega standarda OpenGL. Programa morata biti ločeni zaključeni celoti (*processor, post-processor*), ki si posredujeta potrebne podatke preko formatirane datoteke podatkov. Prikaz konstrukcije v prostoru izvedite s homogenimi transformacijami poljubnega zaporedja. Poleg šolskega primera je potrebno prikazati delovanje na vsaj še dveh dodatnih zahtevnejših primerih. Za vsak program izdelajte poročilo z opisom delovanja programa, teoretičnimi osnovami in rezultati izračuna na wiki strani.

1 Deformacijska metoda končnih elementov

Mehaniko konstrukcij delimo na mehaniko linijskih konstrukcij (enodimenzionalni elementi), ploskovnih konstrukcij (dvodimenzionalni elementi) in mehaniko teles (tridimenzionalne konstrukcije). Metoda končnih elementov je splošna in enaka za vse tri tipe konstrukcij, kar omogoča kombiniranje zgoraj naštetih tipov konstrukcij pri postavljanju problema z metodo končni elementov. Pri reševanju problemov z uporabo MKE konstrukcijo razdelimo na končne elemente. Za linijske konstrukcije so to nosilci ali deli nosilcev, za ploskovne trikotniki, pravokotniki in za konstrukcije v obliki teles tetraedri, heksaedri, ... Vozlišča končnih elementov so povezana med seboj in tvorijo konstrukcijo. S pomočjo enačb elastomehanike poiščemo zveze med pomiki v vozliščih in v poljih elementov. Tako dobljeno enačbo imenujemo enačba končnega elementa, v katerih nastopajo kot neznanke pomiki v vozliščih. Vse enačbe končnih elementov združimo v enačbo konstrukcije, ki je sistem linearnih enačb. Enačbo rešimo ob upoštevnju robnih pogojev in obremenitev. S pomočjo rešitev sistema enačb (pomikov) izračunamo specifične deformacije in napetosti. Opisano metodo imenujemo deformacijska metoda končnih elementov. V primeru da, so neznanke v vozliščih sile pa govorimo o metodi sil. V praksi se izkaže, da ima deformacijska metoda več prednosti pred metodo sil.

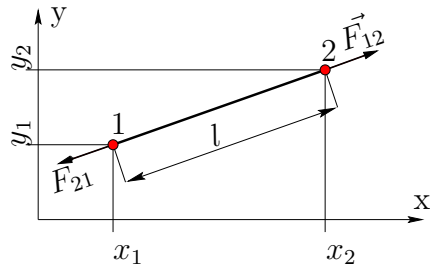
1.1 Enačba končnega elementa

Slika 1 kaže osnovni element, ki se uporablja pri sestavljanju paličij. Obremenitev F je enakomerno porazdeljena po prerezu ploščine A. Zvezo med silo in raztezkom podaja enačba

$$F = \frac{AE}{l}\Delta l \tag{1}$$

Dolžina palice v 3D je

$$l = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2} \tag{2}$$



Slika 1: Obremenjena palica konstantnega prereza

Če so raztezki palice majhni (v primerjavi z dolžino palice) lahko Δl nadomestimo s totalnim diferencialom dl, ki ga zapišemo kot

$$\begin{aligned} dl &= \frac{\partial l}{\partial x_1}dx_1 + \frac{\partial l}{\partial y_1}dy_1 + \frac{\partial l}{\partial z_1}dz_1 \\ &\quad + \frac{\partial l}{\partial x_2}dx_2 + \frac{\partial l}{\partial y_2}dy_2 + \frac{\partial l}{\partial z_2}dz_2 \end{aligned} \tag{3}$$

$$\begin{aligned} dl &= \frac{1}{l}(x_2-x_1)(dx_2-dx_1) \\ &\quad + \frac{1}{l}(y_2-y_1)(dy_2-dy_1) \\ &\quad + \frac{1}{l}(z_2-z_1)(dz_2-dz_1) \end{aligned} \tag{4}$$

Komponentne sile, ki delujejo v palici vozlišča 1 izračunamo tako, da enačbo (1) množimo s smernim kosinusom:

$$\begin{aligned} F_{12x} &= \frac{(x_2-x_1)}{l}\ \frac{AE}{l}dl \\ F_{12y} &= \frac{(y_2-y_1)}{l}\ \frac{AE}{l}dl \\ F_{12z} &= \frac{(z_2-z_1)}{l}\ \frac{AE}{l}dl \end{aligned} \tag{5}$$

Enačbe (5) so povezava med poljubnim pomikom vozlišča (dq_i) in silo v palici. V prostoru ima vsaka palica šest možnih pomikov in šest komponentnih sil.

1.2 Enačba konstrukcije

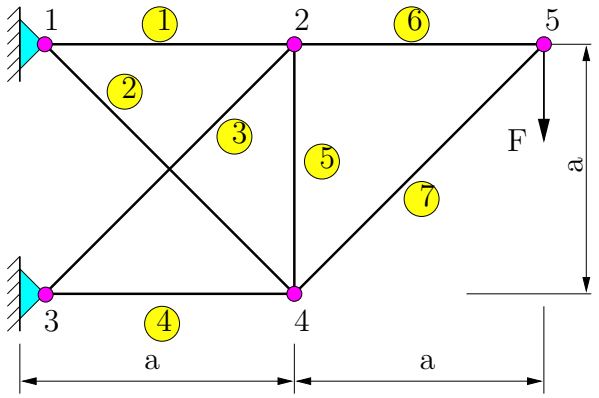
Ker imajo palice tudi skupna vozlišča lahko za vsako vozlišče napišemo ravnotežno enačbo

$$\sum_i \vec{F}_i = 0 \tag{6}$$

Če napišemo ravnotežno enačbo za vsa vozlišča, dobimo enačbo konstrukcije, ki ima obliko

$$\begin{Bmatrix} F_{1x} \\ F_{1y} \\ F_{1z} \\ F_{2x} \\ \vdots \\ F_n \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ k_{31} & k_{32} & \cdots & k_{3n} \\ k_{41} & k_{42} & \cdots & k_{4n} \\ \vdots & & \ddots & \vdots \\ k_{n1} & k_{b2} & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} dx_1 \\ dy_1 \\ dz_1 \\ dx_2 \\ \vdots \\ d_n \end{Bmatrix} \tag{7}$$

Na levi strani enačbe so zunanje sile in neznane reakcije v podporah. Elementi k_{ij} tvorijo matriko togosti. Na desni strani so pomiki vozlišč. Nekateri pomiki v vozliščih so že vnaprej znani in jih zato ni potrebno računati v sistemu linearnih enačb.



Slika 2: Statično nedoločeno paličje; a = 1 m, F = 1 kN, AE = 4e6 N

Slika 2 prikazuje primer ravninskega statično nedoločenega paličja s petimi vozlišči. Velikost sistema enačb je tako $n = 2 \cdot 5 = 10$. Pripadajoča togostna matrika je

$$k_{ij} = \begin{bmatrix} -5.4 & 1.4 & 4 & 0 & 0 & 0 & 1.4 & -1.4 & 0 & 0 \\ 1.4 & -1.4 & 0 & 0 & 0 & 0 & -1.4 & 1.4 & 0 & 0 \\ 4 & 0 & -9.4 & -1.4 & 1.4 & 1.4 & 0 & 0 & 4 & 0 \\ 0 & 0 & -1.4 & -5.4 & 1.4 & 1.4 & 0 & 4 & 0 & 0 \\ 0 & 0 & 1.4 & 1.4 & -5.4 & -1.4 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1.4 & 1.4 & -1.4 & -1.4 & 0 & 0 & 0 & 0 \\ 1.4 & -1.4 & 0 & 0 & 4 & 0 & -6.8 & 0 & 1.4 & 1.4 \\ -1.4 & 1.4 & 0 & 4 & 0 & 0 & 0 & -6.8 & 1.4 & 1.4 \\ 0 & 0 & 4 & 0 & 0 & 0 & 1.4 & 1.4 & -5.4 & -1.4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.4 & 1.4 & -1.4 & -1.4 \end{bmatrix} \cdot 10^3 N \tag{8}$$

Togostno matriko k_{ij} zgradimo tako, da za vse palice izračunamo sile v vozliščih in jih postavimo na ustrezno mesto (trojna zanka).

$$\begin{Bmatrix} R_{1x} \\ R_{1y} \\ 0 \\ 0 \\ R_{3x} \\ R_{3y} \\ 0 \\ 0 \\ 0 \\ 1000 \end{Bmatrix} = \begin{bmatrix} -5.4 & 1.4 & \cdots & 0 & 0 \\ 1.4 & -1.4 & \cdots & 0 & 0 \\ 4 & 0 & \cdots & 4 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 1.4 & -1.4 & \cdots & 1.4 & 1.4 \\ -1.4 & 1.4 & \cdots & 1.4 & 1.4 \\ 0 & 0 & \cdots & -5.4 & -1.4 \\ 0 & 0 & \cdots & -1.4 & -1.4 \end{bmatrix} \begin{Bmatrix} dx_1 \\ dy_1 \\ dx_2 \\ dy_2 \\ dx_3 \\ dy_3 \\ dx_4 \\ dy_4 \\ dx_5 \\ dy_5 \end{Bmatrix} \tag{9}$$

1.3 Robni pogoji

Pogoj stabilnosti konstrukcije zahteva da je del vozliščnih pomikov ali zasukov poznan. Poznani vozliščni pomiki predstavljajo robne pogoje. Ker so pomiki in sile ter zasuku in momenti konjugirane veličine, lahko z poznavanjem ene veličine izrazimo njen konjugirani par.

Linearni sistem enačb (9) je predoločen, saj so nakateri pomiki že vnaprej znani ($dx_1 = 0$, $dy_1 = 0$, $dx_3 = 0$, $dy_3 = 0$). Niso pa poznane velikosti reakcij. Enačbo (9) lahko preuredimo s permutacijo vrtic in kolon v obliko

$$\begin{Bmatrix} \mathbf{F}_m \\ \mathbf{F}_r \end{Bmatrix} = \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{mr} \\ \mathbf{K}_{rm} & \mathbf{K}_{rr} \end{bmatrix} \begin{Bmatrix} \mathbf{U}_m \\ \mathbf{U}_r \end{Bmatrix} \tag{10}$$

v kateri so v spodnjem delu enačb (10) poznani pomiki $\mathbf{U}_r = 0$. Zunanje sile v vozliščih so $\mathbf{F}_m$, reakcije v podporah pa $\mathbf{F}_r$. Zaradi ničnosti vektorja $\mathbf{U}_r$ se enačba (10) poenostavi v

$$\mathbf{F}_m = \mathbf{K}_{mm} \cdot \mathbf{U}_m \tag{11}$$

iz katere lahko izračunamo neznane pomike $\mathbf{U}_m$ z enačbo

$$\mathbf{U}_m = \mathbf{K}_{mm}^{-1} \cdot \mathbf{F}_m \tag{12}$$

Po izračunanih pomikih se reakcije v podporah izračuna kot

$$\mathbf{F}_r = \mathbf{K}_{rm} \cdot \mathbf{U}_m \tag{13}$$

Sile v palicah pa izračunamo iz osnovne enačbe elementa (1 ali 5).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1000 \end{bmatrix} = \begin{bmatrix} -6828 & 0 & 0 & 0 & 1414 & 1414 \\ 0 & -6828 & 0 & 4000 & 1414 & 1414 \\ 0 & 0 & -9414 & -1414 & 4000 & 0 \\ 0 & 4000 & -1414 & -5414 & 0 & 0 \\ 1414 & 1414 & 4000 & 0 & -5414 & -1414 \\ 1414 & 1414 & 0 & 0 & -1414 & -1414 \end{bmatrix} \begin{bmatrix} dx4 \\ dy4 \\ dx2 \\ dy2 \\ dx5 \\ dy5 \end{bmatrix} \tag{14}$$

Za primer na sliki 2 je sistem linearnih enačb (14) dobljen s permutacijo vrstic in kolon. Po izračunu smo dobili naslednje rezultate pomikov in reakcij

$$\begin{array}{ll|ll} dx_4 = -0.3894\,mm & & dy_4 = -0.7838\,mm \\ dx_2 = 0.3606\,mm & & dy_3 = -0.6733\,mm \\ dx_5 = 0.6106\,mm & & dy_5 = -2.491\,mm \\ R_{1x} = -2000\,N & & R_{1y} = 557.8\,N \\ R_{3x} = 2000\,N & & R_{3y} = 442.2\,N \end{array} \tag{15}$$

2 Programska modula

2.1 Procesor

Prebere podatke iz tekstne datoteke, ki jo uporabnik napiše z editorjem v predpisani obliki. Pravilnost podatkov se sproti kontrolira in izpisujejo se napake ter opozorila pri interpretiranju vhodne datoteke. Ob napaki v interpretaciji mora program izpisati vrstico v kateri se je pojavlja napaka in vrsto napake. Vhodna datoteka naj ima končnico `.txt`. Po preračunu pa naj program naredi dve datoteki; eno v tekstovni (`.prn`) s tabelaričnim izpisom izračuna in drugo v okrnjeni obliki (`.bin`) za posredovanje podatkov postprocesorju. Program naj omogoča izračun ravninskega in prostorskega paličja z najmanj 100 vozlišči.

2.1.1 Reševanje sistema enačb

Sistem linearnih enačb lahko rešujemo na različne načine. Klasična Gaussova eliminacijska metoda se pri konkretnih problemih pokaže kot počasna (N^3 operacij). Predlagana metoda, ki se v praksi tudi največ uporablja za

reševanje sistema enačb je *Lower/Upper* dekompozicija, ki ima časovno zahtevnost $1/3N^3$. Reševanje sistema po tej metodi se sestoji iz dveh korakov:

1. **decomposition** razdeli matriko **M** na dve matriki (zgornja / spodnja), katerih produkt je **M**.) Obe matriki sta shranjeni v matriki M, le da je zgornji del matrike **m** matrika **U**, spodnji pa matrika **L**.
2. **backsubstitution** množi desno stran enačbe z zgornjo matriko in pri tem izračuna neznane linearne spremenljivke.

Rešujemo sistem enačb velikosti `N`. Matrika `m` predstavlja levo stran sistema enačb. V C-ju so lahko matrike statične ali dinamične z uporabo funkcije `malloc()`. `N` opisuje velikost matrike `m`. `indx` je celoštevilčni vektor permutacij v matriki `m` in se prenaša naprej v podprogram *lubksb*, kateri zahteva še desno stran sistema enačb v vektorju `u`. Po izračunu se rezultat nahaja v vektorju `u`. Prejšnje vrednosti matrike `m` in vektorja `u` se ne ohranijo!

Potek izračuna sistema linearnih enačb (11) je prikazan na naslednjem primeru, ki izpiše rezultat $u = [1, 2, 3, 4, 5]$.

```
/* m*u = b */
#include <stdio.h>
#include <stdlib.h>
#include "lupack.h"

#define N 5

float m[N*N] = {
  2,  3,  0, 0, 0,
  3,  0,  4, 0, 6,
  0, -1, -3, 2, 0,
  0,  0,  1, 0, 0,
  0,  4,  2, 0, 1};

float b[N] =
  { 8., 45., -3., 3., 19.};

int main()
{
  int i, *indx;
  float d;

  indx = (int *)malloc(N*sizeof(int));
  ludcmp(m, N, indx, &d);
  lubksb(m, N, indx, b);
  free(indx);

  for (i = 0; i < N; i++)
    printf("u [%d] = %g\n", i, b [i]);

  return 0;
}
```

2.1.2 Vhodna datoteka

Format datoteke je podoben programom STRESS (*STRuctural Engineering System Solver*). Vhodna datoteka je tekstovna s ključnimi besedami katerim sledijo numerični podatki.

structure ***naziv*** Naziv je le kot komentar.

plane ali space določa dimenzijo koordinatnega sistema ravnina ali prostor. Če imamo ravninski problem, potem ne podajamo koordinate z.

joints ***število vozlov***

members ***število palic***

reactions ***število reakcij***

loads ***število obremenitev (sil)***

coordinates sledi seznam koordinat vozlišč z zacetnim vozliščem št. 1 v obliki:
`št.  vozlišča koord-x koord-y (koord-z)`

incidences sledi seznam palic (palice se začnejo šteti z 1) z navedenimi povezavami vozlišč v obliki:
`št.-palice začetni-vozel končni-vozel lastnost-`$A \cdot E$ AE je zmnožek površine prereza palice in elastičnega modula materiala palice.

supports sledi seznam vozlišč v katerih so podpore in smeri delovanja reakcij v obliki:
`številka vozlišča črka x ali y ali z`

forces sledi seznam sil v vozliščih in komponente sil v posameznih smereh v obliki:
`številka vozlišča sila-x sila-y (sila-z)`

solve pove programu, da je konec podatkov in naj konča z interpretiranjem.

Primer vhodne datoteke za statično nedoločen nosilec v ravnini (slika 2).

```
structure MDT2 p.479/16
plane
joints 5
members 7
reactions 4
loads 1

coordinates
1     0.0  1000.0
2  1000.0  1000.0
3     0.0     0.0
4  1000.0     0.0
5  2000.0  1000.0

incidences
1 1 2 4e6
2 1 4 4e6
3 2 3 4e6
4 3 4 4e6
5 2 4 4e6
6 2 5 4e6
7 4 5 4e6

forces
5 0.0 -1000.0

supports
1 x
1 y
3 x
3 y

solve
```

Za interpretiranje vhodne datoteke je najbolje, da se podatki berejo po vrsticah, ki se nato interpretirajo glede na ključne besede. Delček kode, ki najprej prebere vrstico v znakovni niz in nato preveri ključno besedo je naslednji:

```
/* Branje vhodnih podatkov */
while (!feof (f))
  {
    char s[BUFLEN];
    fgets (s, BUFLEN, f);

    if (strncmp (s, "space", 5) == 0)
      dim = 3;

    if (strncmp (s, "joints", 6) == 0)
      sscanf(s + 6, "%d", &joints);
   ...
  }
```

2.1.3 Tekstovna izhodna datoteka

Tekstovna izhodna datoteka naj izpiše vse rezultate preračuna v tabelirani obliki. Palice, ki so obremenjene na tlak imajo po dogovoru negativen predznak sile. Reakcije v podporah naj bodo tabelirane tako, da se bo vedelo v katerem vozlišču je podpora in kakšna je velikost. Predznak (usmerjenost) sile v podporah se določi glede na globalni koordinatni sistem.

2.1.4 Grafična izhodna datoteka

Grafična izhodna datoteka se bo uporabila za posredovanje podatkov postprocesorju. Kateri podatki so potrebni za nadaljno obdelavo ocenite sami. (npr. kordiante vozlišč, pomiki, št. palic, vozlišč, sile v palicah, ...). Ime datoteke naj bo fiksno (npr. `palicje.bin`), tako da poprocesor ne bo spraševal po vhodni datoteki ampak samo preverjal če obstaja.

2.2 Postprocesor

Postprocesor je program, ki omogoča prikaz in nadaljno obdelavo izračunanih podatkov (za razliko od preprocesorja, ki bi npr. omogočal interaktiven vnos podatkov za preračun).

Za grafičen prikaz rezultatov v programu uporabite grafični jezik OpenGL. Po-procesor naj omogoča prikaz konstrukcije v treh dimenzijah s poljubnim zaporedjem transformacj v prostoru. V ta namen uporabi homogene matrike in iz njih sestavi kompozicijsko matriko. Projekcija na zaslon naj bo ortogonalna projekcija ravnine x-y. Ravninski nosilci imajo koordinato z=0. Na zahtevo (menu) mora program izrisati na konstrukcijo še označbe vozlišč, palic, sile v palicah oziroma pomike. Ker so pomiki vozlišč majhni, jih je potrebno množiti s konstanto, da dobimo pretiran prikaz deformiranega paličja. Palice, ki so obremenjene na tlak izriši v rdeči barvi, natezno obremenjene palice pa v zeleni. Risanje sil in reakcij v vozliščih je neobvezno.

Program za prikaz mora uporabljati homogene transformacije v dvodimenzionalnem prostoru s kompozicijo matrik poljubnega vrstnega reda. Osnovni program, ki demonstrira uporabo jezika OpenGL in knjižnice uporabniškega vmesnika GLUT ter omogoča vnos z miško je:

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glu.h>
#include <GL/glut.h>

#define MAXN 100
GLint n;
GLfloat *vertex;

void redraw()
{
  int i;
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_LINE_STRIP);
  for (i = 0; i < n; i++)
    glVertex2fv(&vertex[2*i]);
  glEnd();
  glutSwapBuffers();
}

void mouse(int button, int state, int x, int y)
{
  GLint viewport[4];
  GLdouble mvmatrix[16], projmatrix[16];
  GLdouble wx, wy, wz; /*x, y, z coords*/
  GLdouble px, py, pz; /*picked window coords*/
  int status;

  if (button == GLUT_LEFT_BUTTON )
    if (state ==  GLUT_DOWN) {
      glGetIntegerv (GL_VIEWPORT, viewport);
      glGetDoublev (GL_MODELVIEW_MATRIX, mvmatrix);
      glGetDoublev (GL_PROJECTION_MATRIX, projmatrix);
      /* note viewport(4) is height  in pixels */
      px = x;
      py = viewport[3] - y - 1;
      pz = 0.0;
      fprintf (stderr,
               "Coordinates at cursor are %f, %f\n",
                px, py);
      status = gluUnProject (px, py, pz, mvmatrix,
                projmatrix, viewport, &wx, &wy, &wz);
      fprintf(stderr,
            "World coords at z=0.0 are %f, %f, %f\n",
             wx, wy, wz);
      if (n < MAXN) {
        vertex[n*2] = wx; vertex[n*2+1] = wy;
        n ++;
      } else
        fprintf(stderr,
          "Doseženo maksimalno stevilo točk!\n");
      glutPostRedisplay();
    }
}
int main(int argc, char *argv[])
{
  vertex = (GLfloat *)
           malloc(2 * MAXN * sizeof (GLfloat));
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow("Click in window");
  glutDisplayFunc(redraw);
  glutMouseFunc(mouse);
  glutMainLoop();
  return 0;
}
```

Za označevanje in izpis rezultatov v grafični obliki v prostoru je najenostavneje uporabiti rastrske (bitmap) fonte. Ker pa GLUT omogoča izpis le enega znaka v prostoru, se predlaga naslednji podprogram, ki omogoča izpis niza znakov, podobno kot funkcija `printf(format, ...)`, le da podamo še začetno točko v prostoru.

```
#include <stdarg.h>
#include <GL/glut.h>
```

```
void print3D(GLfloat x, GLfloat y, GLfloat z,
             const char *format, ...)
{
  glRasterPos3f(x, y, z);
  char    buf[80];
  char*   ch = buf;
  va_list args;
  va_start(args, format);
  vsnprintf(buf, 80, format, args);
  va_end(args);
  while (*ch)
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,*ch++);
}
```

Uvod v OpenGL s primeri se nahaja na spletni strani `http://rpk.lecad.si/vaje/wiki/opengl-intro`

3 Poročilo

Poročilo se oddaja v dveh delih, prvi za numerični program, drugi za grafični del v wiki formatu. 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 programske kode. Priložite tudi rezultate primerov, ki ste jih izračunali.

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

4 Delo na računalniku

Na vsakem računalniku v učilnici N17 je nameščen navidezni računalnik (*virtualbox*) z operacijskim sistemom Linux, ki vsebuje vsa potrebna orodja za izdelavo naloge. Navidezni računalnik deluje v nespremenljivi (*immutable*) obliki, kar pomeni, da se datoteke na navideznem računalniku ne hranijo stalno. Navidezni računalnik se ob ponovnem zagonu vedno postavi v prvotno stanje ne glede na spremembe datotečnega sistema.

Delo (datoteke) je tako potrebno shranjevati na strežnik. Za hranjenje in sledenje spremembam je na strežniku nameščen sistem *subversion*. Prikaz, primerjanje in komunikacija pri izvedbi vaj pa poteka s sistemom *TRAC* na namenskem strežniku `http://rpk.lecad.fs.uni-lj.si/`. Vsa potrebna navodila in dodatna so dostopna na projektu `http://rpk.lecad.si/vaje`

Vsakemu študentu bo dodeljeno uporabniško ime in geslo za dostop do njegovih datotek na strežniku.

Kljub temu, da obstaja množica razvojnih vmesnikov (IDE) je na vajah predvidena uporaba osnovnih orodij za razvoj programov. Navidezni računalnik je možno namestiti na poljuben operacijski sistem in nalogo opravljamo tudi od doma, če je na voljo dovolj zmogljiv računalnik in internetna povezava.

Predvidena so naslednja orodja za razvoj programske opreme:

1. Terminalsko okno z ukazno lupino
2. Urejevalnik gedit
3. C prevajalnik gcc
4. Razhroščevalnik gdb in ddd
5. Prenos datotek in komunikacija s strežnikom - subversion
6. Brskalnik

Delovni cikel v ukazni lupini za vzorčnega uporabnika `vaje` je naslednji:

1. Zadnjo inačico datotek uporabnika `vaje` poberemo s strežnika z ukazom `svn co svn://lecad.fs.uni-lj.si/rpk/vaje`
2. Premik v delovni imenik `cd vaje`
3. Urejanje in prevajanje datotek (gedit, cc, ddd, ...)
4. Shranitev delovne verzije datotek na strežnik z ukazom `svn ci -m "opis narejenega dela"`

Podrobnejša navodila in predstavitev delovnega cikla so predstavljena na spletni strani `http://rpk.lecad.si/vaje/wiki/lab-intro` so predvidena za uvodne vaje.

SVN Dostop do primerov v projektu vaje Primere lahko poleg brskanja na WWW strani poberete tudi z ukazom

```
svn cp --username vaje \
    svn://lecad.si/rpk/vaje/lupack.c .
```

Podajanje uporabniškega imena vaje v navideznem računalniku ni potrebno, saj je to ime že privzeto. Geslo za dostop je ravno tako vaje.

Primer za reševanje linearnih enačb si skopirate v svoj imenik z ukazom:

```
cd
cp vaje/lupack.c ipriimek/
cp vaje/lupack.h ipriimek/
cp vaje/example-lin.c ipriimek/
```

kjer je ipriimek seveda vaš imenik projekta, ki ste ga popreje ravno tako izvozili

```
svn co svn://lecad.fs.uni-lj.si/rpk/ipriimek
```

Ne pozabite si datoteke prijaviti z ukazi

```
cd ipriimek
svn add lupack.* example-lin.c
svn ci -m "Primer reševanja linearnih enačb"
```

5 Terminski načrt

Po sedmih uvodnih vajah na katerih so prikazani potrebni postopki za uspešno izdelavo naloge, se predvideva samostojno delo študentov za katerega je do konca semestra predvideno še sedem terminov na katerih bo prisoten asistent s katerim lahko rešujete težave pri izdelavi aplikacije. Na vajah bodo prikazane osnove programskega jezika C s primeri, katere bo možno razširiti ali uporabiti v končnem programu.

Na uvodnih vajah, na kateri so bile prikazane značilnosti jezika C in orodja potrebna za izdelavo projekta, so z namenom ponovitve in utrditve znanja C-ja izdelana besedila domačih nalog dostopna na spletni strani `http://rpk.lecad.si/vaje/wiki/naloge`. Besedila nalog

so razporejena po skupinah in tematiki, ki je potrebna za izvedbo projekta. Domače naloge bodo preko zahtevkov dodeljene vsakemu študentu posebej. Vsak študent mora do naslednjega tedna izdelati dodeljene vaje in preveriti ali mu pravilno delujejo. Pravilnost delovanja izbranih nalog preverimo z "Ocenjevalcem nalog" na spletni strani `http://lecad.si/cgi-bin/cclass.cgi`.

Po uvodnih vajah je obisk v laboratoriju LECAD je še vedno obvezen, vendar je pričakovati različen napredek, ki je odvisen od posameznikove usposobljenosti reševanja večjih problemov. Obisk laboratorija je možen tudi izven predvidenega urnika, če so računalniki učilnice prosti. Možna je tudi namestitev navideznega računalnika z vsemi potrebnimi orodji in s tem delo od doma.

Vaje se zaključijo 14. januarja in takrat je možno oddati končno verzijo tako, da zaprosite asistenta za pregled in oceno vašega izdelek, ki se je oddajal sproti, kar bo tudi razvidno iz poteka oddajanj (*TracTimeline*). V dogovoru s študenti bo določen tudi rok za oddajo, ki mora biti pred pričetkom letnega semestra.

6 Ocenjevanje

Programska koda naj vsebuje komentarje, ki bodo opisovali pomen posameznih odsekov programa. Uporaba MakeFile-a je obvezna, saj je v osnovi program napisan na virtualnem računalniku, kjer je operacijski sistem Linux. Pri oddaji poročila in programa mora biti tudi testni primer, s katerim se dokaže pravilnost delovanja programa.

Ocena naloge bo sestavljena iz naslednjih kriterijev:

- 10% Sprotnost dela in sledenje razvoju s sistemom TRAC iz katerega mora biti jasno razviden postopek, s katerim ste prišli do končne verzije izdelka. Iz vsake opombe ob shranitvi na strežnik mora biti jasno predstavljena vsebina sprememb! To velja tako za izvorno kodo, kot za poročilo.
- 10% Jasnost sporočil pri shranjevanju dela in opis le teh v poročilu.
- 5% Prisotnost in izvedba domačih nalog s katerimi si pridobimo delovni cikel.
- 35% Delovanje programa za preračun sil v palični konstrukciji in jasnost kode. Uporaba programa `make` in datoteke `Makefile` je obvezna.
- 20% Oblika in vsebina poročila. Poročilo v elektronski obliki se vnaša s sistemom Trac na strani WIKI.
- 20% Neobvezni del funkcionalnosti programa, ki obsega različne dopolnitve programa in se individualno oceni glede na zahtevnost in izvirnost rešitve, ki mora biti podrobneje opisana v poročilu. Npr. kontrola uklona 1%, risanje sil in reakcij v vozliščih 2%, enovita aplikacija 2%, element z momenti 15%, interaktivnost in vrtenje z miško 4%, preprocesor za grafični vnos, uvoz iz CAD programa, ...

Rang ocen:

```
 50% - 60% =  6
 61% - 70% =  7
 71% - 80% =  8
 81% - 90% =  9
91% - 100% = 10
```

Predstavljeni kriteriji niso dokončni in se bodo lahko dopolnili ali popravili do konca koledarskega leta, če bodo v tem času ugotovljena nova dejstva. Strani "Časovni pregled" in "WikiStart" posameznih projektov so vidne le z geslom, kar onemogoča neavtoriziran dostop do vašega izdelka. Sodelovanje študentov na način, kjer bi se programska koda prepisovala na kakršen koli način, ni dovoljeno. Če se problematika diskutira, potem naj se to izvaja tako, da se v kodo ne gleda oziroma se preverja kodo na abstraktnem primeru. Izdelki vseh študentov (poročila in programi) so ob ocenjevanju primerjani med seboj s sistemom MOSS (*Measure Of Software Similarity*), s katerim je možno enostavno analizirati kodo in detektirati morebitni plagiarizem, ki je povod za negativno oceno.

Kako zaprositi za oceno? Poleg osebnega kontakta in e-pošte je za vse najbolj priročno, če zaprosite kar v Tracu, tako da odprete nov Zahtevek. Tam napišite kar želite in asistenta bosta preko e-pošte obveščena, da je odprt nov listek. Če želite tudi Vi spremljati dogodke na listku je najbolje, da pod Settings napišete svoje e-poštni naslov in boste tako obveščeni, ko se bo pojavil odgovor. Vsi projekti bodo imeli spremljanje ocenjevanja preko listka. Možne so seveda tudi pripombe.

Datotek ni potrebno prilagati v "Zahtevek". Dovolj je, da zaprosite za oceno. Pogledala bova SVN verzijo, ker je lažje dobiti izdelek naenkrat, kot pa vsako datoteko pobirati v začasni imenik.

Literatura

[1] Kraut Bojan. *Strojniški priročnik.* Strojniški vestnik, Ljubljana, 2003.

[2] Jože Duhovnik, Milan Kljajin, and Milan Opalić. *Inženirska grafika.* Univerza v Ljubljani, Fakulteta za strojništvo, 2009.

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

[4] Anthony Porter. *The Best C/C++ Tips Ever.* Osborne McGraw-Hill, U.S.A., 1993.

[5] Ervin Prelog. *Metoda končnih elementov.* FAGG Ljubljana, 1975. FS 16528.

[6] Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 3.0 and 3.1.* Upper Saddle River [etc.], Addison-Wesley, 2010.

[7] Marko Škerlj. *Mehanika - Trdnost.* Fakulteta za strojništvo, Ljubljana, 1984.

[8] Žiga Turk. *Programski jezik C.* ZOTKS, Ljubljana, 1987.

[9] Richard S. Wright. *OpenGL SuperBible.* SAMS, cop., Indianapolis, 2005.

Ljubljana 6. oktober 2010