Laboratorijske vaje pri predmetu Računalniško Podprto Konstruiranje

Leon Kos, Laboratorij za CAD - LECAD

Naloga

Izdelajte program za izračun ravninskega stacionarnega toka nestisljivega in neviskoznega fluida. Fluid naj se pretaka v mejah poljubne oblike. Rešitve enačbe in hitrostno polje v izbranih točkah prikažite z uporabo grafičnega jezika *OpenGL*.

1 Tok idealnega fluida

1.1 Kontinuitetna enačba

Integralska oblika zakona o ohranitvi mase za mirujoči kontrolni volumen V_k je

$$\frac{\partial}{\partial t}\int_{V_k}\rho dV + \int_{A_k}\rho\vec{v}_r d\vec{A} = 0 \tag{1}$$

V primeru stacionarnega toka $\partial\rho/\partial t = 0$ se poenostavi v obliko

$$\int_{A_k}\rho\vec{v}d\vec{A} = 0 \tag{2}$$

Enačbo (2), ki je površinski integral po kontrolni površini, lahko z Gaussovim stavkom prevedemo na volumskega

$$\int_{V_k}[\vec{\nabla}\cdot(\rho\vec{v})]dV = 0 \tag{3}$$

Ker lahko kontrolni volumen v enačbi (3) poljubno izberemo, mora enačba veljati za vse primere. Sledi, da mora biti integrand enak nič

$$\vec{\nabla}\cdot(\rho\vec{v}) = 0 \tag{4}$$

Za nestisljiv fluid se enačba (4) poenostavi na

$$\vec{\nabla}\cdot\vec{v} = 0 \tag{5}$$

Enačbo (5) imenujemo kontinuitetna enačba, ki v komponentni obliki zapišemo kot

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0 \tag{6}$$

1.2 Hitrostni potencial

Za nevrtinčen oz. potencialen tok lahko podamo hitrostno polje $\vec{v}(\vec{r},t)$ z gradientom skalarne funkcije $\phi(\vec{r},t)$

$$\vec{v} = \vec{\nabla}\phi \tag{7}$$

Za kartezičen koordinatni sistem je komponenta oblika enačbe 7

$$v_x = \frac{\partial\phi}{\partial x}\ ,\quad v_y = \frac{\partial\phi}{\partial y}\ ,\quad v_z = \frac{\partial\phi}{\partial z} \tag{8}$$

Z vstavitvijo enačb (8) v kontinuitetno enačbo (7), dobimo enačbo hitrostnega potenciala

$$\frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial\phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial\phi}{\partial z}\right) = 0 \tag{9}$$

ki jo imenujemo *Laplaceova enačba* in opisuje potencialno polje v volumnu, ki nima notranjih izvorov. Zapišemo jo

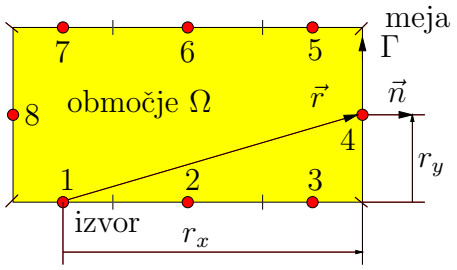
$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{10}$$

ali krajše

$$\vec{\nabla}^2\phi = 0 \tag{11}$$

2 Metoda robnih elementov - BEM

Potencialne probleme ki jo predstavlja enačba (11) najlažje rešujemo z metodo robnih elementov. Vse ostale metode (MKD, MKE, MKV, ...) zahtevajo diskretizacijo območja kjer integriramo diferencialno enačbo. Metoda robnih elementov[2, 7] (*Boundary Elemenh Method*) omogoča, da prevedemo reševanje diferencialne enačbe iz notranjosti na rob, ki omejuje območje.



Slika 1: Definicija območja in smer obhodne krivulje za dvodimenzionalni primer toka idealnega fluida

Primer s slike 1 kaže pravokotnik, ki ima območje Ω v katerem je potrebno zadovoljiti enačbo 10 v vsaki točki. Pri metodi končnih diferenc se običajno izbere točke v katerih se enačba, ob upoštevanju pogojev na robu, zadovolji. Z BEM pa se prevede reševanje enačbe na rob, kje se hkrati zadovolji tudi robne elemente. Ko je enačba (10) na robu rešena, lahko za vsako poljubno točko v območju dobimo rešitev z enostavno integracijo po robu.

2.1 Divergenčni teorem

Za vektorsko funkcijo $\vec{F} = (F_x, F_y, F_z)$ obstaja divergenčni teorem, ki povezuje volumski in površinski integral funkcije

$$\int_V\left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}\right)dV = \int_S(\vec{F}\cdot\vec{n})dS \tag{12}$$

kjer je $\vec{n}$ normala na površini območja. Za dvodimenzionalni primer s slike 1 lahko uporabimo hitrostno polje iz enačbe (8) in napišemo divergenčni teorem kot

$$\int_S\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right)dS = \int_l(v_x n_x + v_y n_y)dl \tag{13}$$

Leva stran enačbe (13) je v vsaki točki območja nič zaradi enačbe (10). Tako velja, da je integral po krivulji tudi enak nič. Če torej območje diskretiziramo kot na sliki 1, na ravne elemente, nam enačba (13) pove, da mora biti vsota vseh pretokov $\vec{v} \cdot \vec{n}$ skozi elemente enaka nič. Povedani drugače, vsota zmnožkov normalnih hitrosti in dolžin mora biti nič.

2.2 Greenova druga identiteta

če za vektorsko funkcijo $\vec{F} = u\vec{\nabla}u^*$ uporabimo divergenčni teorem iz enačbe (12) lahko izpeljemo Greenovo drugo identiteto

$$\int_V u\vec{\nabla}^2 u^* dV = \int_S q^* u dS - \int_S u^* q dS \tag{14}$$

kjer je u potencial, ki je v našem primeru $u = \phi$. Funkcija u^* je poljubna funkcija, ki jo bomo kasneje izbrali tako, da nam bo pomagala odstraniti volumski integral. Funkcija q je pravokotna hitrost na elementu

$$q = \frac{\partial u}{\partial n} = \frac{\partial \phi}{\partial n} \tag{15}$$

Funkcija q^* je prav tako poljubna, vendar povezana z u^* z enačbo

$$q^* = \frac{\partial u^*}{\partial n} \tag{16}$$

2.3 Osnovni rešitvi

Ker lahko za u^* izberemo poljubno funkcijo, ki nima zveze z realnim problemom, je ugodno če izberemo za u^* v volumskem integralu enačbe 14, funkcijo, ki odpravi ali poenostavi volumski integral. Impulzna (*Diracova delta*) funkcija ima naslednje lastnosti:

$$\delta(p,x) = \begin{cases} 0 & x \neq p \\ \infty & x = p \end{cases} \tag{17}$$

$$\int_V \delta(p,x) dV = 1 \tag{18}$$

če izberemo $\vec{\nabla}^2 u^* = -\delta(p,x)$ za impulzno funkcijo odstranimo volumski integral in dobimo znani funkciji u^* in q^*. Volumski integral enačbe (14) je torej

$$\int_V u\delta(p,x) dV = u(p) \tag{19}$$

kar reducira enačbo (14) na

$$u(p) + \int_S q^* u dS = \int_S u^* q dS \tag{20}$$

kjer je izraz $u(p)$ vrednost potenciala v poljubno izbrani točki. Za dvodimenzionalne primere dobimo *osnovni rešitvi*:

$$u^* = \frac{1}{2\pi} \ln\left(\frac{1}{r}\right) \tag{21}$$

$$q^* = \frac{\partial u^*}{\partial r}\frac{\partial r}{\partial n} = \frac{-1}{2\pi r^2}(r_x n_x + r_y n_y) \tag{22}$$

Radij $r = \sqrt{r_x^2 + r_y^2}$ je razdalja, med izbrano točko (izvor) in točko na ograjo po kateri integriramo. $\vec{n} = (n_x, n_y)$ je normala v točki na ograji po kateri integriramo.

2.4 Integriranje na ograji

Točko p lahko izberemo v območju Ω, na ograji Γ ali celo izven območja, kjer ni potenciala. Če torej izberemo točko na gladki ograji, je le polovica točke v območju; Ostala polovica pa je zunaj. Če izberemo točko v vogalu (slika 1) je le četrtina točke v območju. Če integriramo na ograji, je potrebno v enačbi 20 upoštevati, da je le del točke v območju in da zaradi tega je potencial v točki na ograji le tolikšen, kolikor točke je v območju. Enačbo (20) na ograji zapišemo kot

$$c(p)u(p) + \int_S q^* u dS = \int_S u^* q dS \tag{23}$$

kjer je $c(p)$ delež točke v območju. Če je točka na ravnem (gladkem) robu, je delež $c(p) = 1/2$. Če je točka v vogalu, kot na sliki 1 je $c(p) = 1/4$.

2.5 Robni elementi

Ograjo Γ je potrebno diskretizirati na enostavne elemente, katere lahko enostavno integriramo. Diskterizirana enačba na meji je

$$c(p)u(p) + \sum_{i=1}^{n} \int_S q^* u dS_i = \sum_{i=1}^{n} \int_S u^* q dS_i \tag{24}$$

Najenostavnejši elementi na ograji so konstantni. Pri teh elementih sta funkciji u in q po elementu konstantni. Enačba 24 za konstantne ravne elemente pri katerih smo izbrali, da je izvorna točka vedno na sredini, je tako

$$0.5u(p) + \sum_{i=1}^{n} \int_S q^* dS_i u = \sum_{i=1}^{n} \int_S u^* dS_i q \tag{25}$$

Za vsak element lahko napišemo enačbo (25), kar vodi do sistema n enačb. Integrale osnovnih rešitev na ograji lahko pišemo krajše kot

$$\hat{h}_{ij} = \int_{S_{elem}} \frac{-1}{2\pi r^2}[r_x n_x + r_y n_y] dS \tag{26}$$

$$g_{ij} = \int_{S_{elem}} \frac{1}{2\pi} \ln\left(\frac{1}{r}\right) dS \tag{27}$$

Vzemimo za primer prvi element, ki ima točko na sredini. Enačba prvega elementa z upoštevanjem okrajšav (26) in (27) je

$$0.5u_1 + \hat{h}_{11}u_1 + \hat{h}_{12}u_2 + \cdots + \hat{h}_{1n}u_n = g_{11}q_1 + \cdots + g_{1n}q_{1n} \tag{28}$$

Člen $c(p)$ lahko združimo z definicijo

$$h_{ij} = \begin{cases} \hat{h}_{ij} + c(p) & i = j \\ \hat{h}_{ij} & i \neq j \end{cases} \tag{29}$$

Sedaj lahko napišemo sistem n enačb, ki ga zapišemo

$$\mathbf{H}\,\mathbf{u} = \mathbf{G}\,\mathbf{q} \tag{30}$$

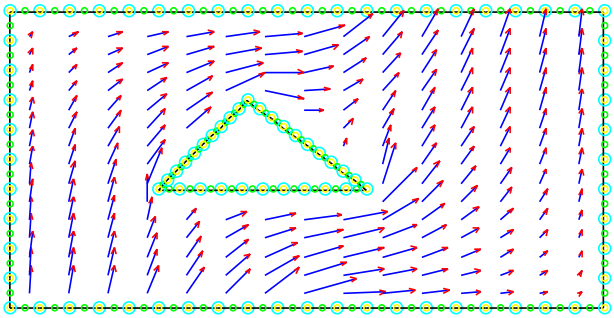
Matriki $\mathbf{H}$ in $\mathbf{G}$ sta velikosti $(n \times n)$. Vektor u je potencial in vektor q je pravokotna hitrost na elementih.

2.6 Robni pogoji

V naši nalogi imamo izračun toka idealnega fluida. Definirano je hitrostno polje s potencialom $u = \phi$ in hitrostmi na robu $q = v$. Če skozi rob ne teče fluid, je to ovira, za fluid. Pogoj za obtekanje je torej podan z normalno hitrostjo na površini:

$$\vec{v} \cdot \vec{n} = 0 \tag{31}$$

Telesa, ki jih fluid obteka imajo na ograji $q = 0$ in smer obhoda v smeri urinega kazalca.



Slika 2: Obtekanje telesa

Primer obtekanja telesa kaže slika 2. Ograja, v kateri zajamemo fluid pa ima smer krivulje naproti obtekajočem trikotniku. Če želimo videti gibanje fluida, moramo na ograjo podati hitrost q na elementih v katerih fluid priteka in odteka, s tem, da upoštevamo enačbo (13). Za elemente skozi katere pretoka ni, podamo nično hitrost $q_i = 0$. Laplaceovo enačbo (10) rešujemo z znanimi hitrostmi na vseh elementih. Sistem enačb (30) se tako poenostavi na

$$\mathbf{H}\,\mathbf{u} = \mathbf{b} \tag{32}$$

kjer je $\mathbf{b} = \mathbf{G}\,\mathbf{q}$ vektor izračunan z enostavnim množenjem matrike $\mathbf{G}$ in vektorja hitrosti na ograji $\mathbf{q}$.

3 Gaussova kvadratura

Numerična integracija je metoda izračuna vrednosti integrala splošnih funkcij takrat, ko ni znana točna analitična rešitev. Gauss–Legendrova kvadraturna formula omogoča hiter in natančen izračun integrala za splošne funkcije. Za razliko od ostalih metod (Simpsonova, trapezna, ...) ima ta integracija majhno število izbranih točk, kar pospeši programe, ki računajo veliko število integralov.

Splošna enačba Gaussove kvadrature je

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} \{f(x_i)w_i\} \tag{33}$$

Integral funkcije $f(x)$ se izračuna tako, da se v n točkah vzame vrednost funkcije $f(x_i)$, ki se jo pomnoži z z utežjo w_i. Vsota tako obteženih vrednosti funkcije je vrednost integrala. Pomembno je poudariti, da so meje integrala od -1 do 1, zato je potrebno pri drugačnih mejah uporabiti translacijo in skaliranje osnovne funkcije, predno uporabimo Gaussovo kvadraturo za izračun integrala. Integracija poteka torej simetrično. Red kvadrature določa število točk, ki so uporabljene pri izračunu integrala. V naslednji tabeli so podane kvadrature za posamezno število točk:

red(n)	$\pm x_i$	w_i
1	0.00000	2.00000
2	0.57735	1.00000
3	0.00000	0.88889
	0.77459	0.55556
4	0.33998	0.65214
	0.86134	0.34785
5	0.00000	0.56889
	0.53847	0.47863
	0.90618	0.23693
6	0.23862	0.46791
	0.66121	0.36076
	0.93247	0.17132
7	0.00000	0.41796
	0.40585	0.38183
	0.74153	0.27972
	0.94911	0.12948

Tabela 1: Točke za Gaussovo kvadraturo in uteži

Primer izračuna za funkcijo

$$y = f(x) = 2x^4 - x^3 + 1$$

za katero želimo izračunati integral

$$I = \int_{-1}^{1} (2x^4 - x^3 + 1)dx$$

in ima analitično rešitev 2.80.

- n = 2
  Imamo dve integracijski točki pri $x = \pm 0.57735$.
  Integral I izračunamo kot
  $$\begin{aligned} I &= f(-0.057736) \cdot 1.00000 + f(0.057736) \cdot 1.00000 \\ &= 2.44444 \end{aligned}$$
  kar je 12.7% odstopanja od točne rešitve.
- n = 3
  Imamo tri integracijske točke pri $x = 0$ in $x = \pm 0.77459$
  $$\begin{aligned} I &= f(-0.77459)0.55556 + f(0)0.88889 \\ &+ f(0.77459)0.55556 = 2.8000 \end{aligned}$$
  kar kaže na natančnost in hitrost kvadrature.

3.1 Translacija in skaliranje

Če želimo integrirati funkcijo z Gaussovo kvadraturo po poljubnem območju moramo prevesti integracijsko območje na $[-1, 1]$ s skaliranjem s in translacijo t.

$$s = \frac{x_{max} - x_{min}}{2} \quad , \qquad t = x_{min}$$

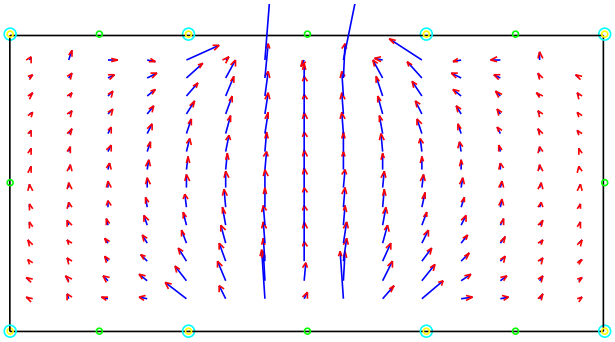
Integral z novimi mejami izračunamo kot

$$I = s \sum_{i=1}^{n} f(sx_i + t)w_i$$

4 Primer

Na primeru s slike 1 bomo pokazali reševanje metode robnih elementov. Dimenzija območja je 20×10. Na srednjem spodnjem elementu dolžine 8 priteka fluid s hitrostjo 1. Na zgornjem elementu pa fluid odteka.

4.1 Vhodna datoteka



Slika 3: Primer ravninskega potencialnega toka z osmimi konstantnimi robnimi elementi

Vhodna datoteka popisuje problem s slike 3. V prvi vrstici je podano število elementov in število nadaljnjih delitev elementov. Sledijo elementi z začetno, končno koordinato in hitrost na robu.

```
8 1
00 00 06 00 0
06 00 14 00 -1
14 00 20 00 0
20 00 20 10 0
20 10 14 10 0
14 10 06 10 1
06 10 00 10 0
00 10 00 00 0
```

4.2 Integracija

Elemente sistema enačb računamo z Gaussovo kvadraturo s štirimi točkami.

$$h_{ij} = \frac{-l}{4\pi} \sum_{k=1}^{4} \frac{w_k}{r^2} (r_x n_x + r_y n_y) \tag{34}$$

$$h_{ii} = 0.5 \tag{35}$$

$$g_{ij} = \frac{l}{4\pi} \sum_{k=1}^{4} w_k \ln(1/r) \tag{36}$$

$$g_{ii} = \frac{l}{2\pi} \left[\ln\left(\frac{2}{l}\right) + 1 \right] \tag{37}$$

Razdalja $r = \sqrt{r_x^2 + r_y^2}$ od sredine elementa izvora do vsake k-te točke kvadrature lahko izračunamo z linearno interpolacijo med začetno $\vec{e}_0$ in končno točko $\vec{e}_1$ elementa j

$$\vec{e}_j(x_k) = \vec{e}_0 + \frac{x+1}{2}(\vec{e}_1 - \vec{e}_0)\,. \tag{38}$$

Iz enačbe (38) je razvidno, da pri $x_k = -1$ dobimo začetno točko elementa $\vec{e}_o$, pri $x_k = 0$ dobimo srednjo točko elementa in da je $e_j(1) = \vec{e}_1$. Vektor $\vec{r}$ od sredine vira i do k-te točke j-tega elementa je

$$\vec{r} = \vec{e}_j(x_k) - \vec{e}_i(0) \tag{39}$$

V enačbah (34–37) je $l = |\vec{e}_1 - \vec{e}_0|$ dolžina j-tega robnega elementa.

Za dani primer ima matrika H naslednjo rešitev

$$\begin{bmatrix} 0.50 & -0.00 & -0.00 & -0.08 & -0.03 & -0.09 & -0.09 & -0.20 \\ -0.00 & 0.50 & -0.00 & -0.13 & -0.06 & -0.12 & -0.06 & -0.13 \\ -0.00 & -0.00 & 0.50 & -0.20 & -0.09 & -0.09 & -0.03 & -0.08 \\ -0.02 & -0.06 & -0.14 & 0.50 & -0.14 & -0.06 & -0.02 & -0.08 \\ -0.03 & -0.09 & -0.09 & -0.20 & 0.50 & -0.00 & -0.00 & -0.08 \\ -0.06 & -0.12 & -0.06 & -0.13 & -0.00 & 0.50 & -0.00 & -0.13 \\ -0.09 & -0.09 & -0.03 & -0.08 & -0.00 & -0.00 & 0.50 & -0.20 \\ -0.14 & -0.06 & -0.02 & -0.08 & -0.02 & -0.06 & -0.14 & 0.50 \end{bmatrix} \tag{40}$$

Matrika G je:

$$\begin{bmatrix} -0.09 & -2.40 & -2.51 & -4.59 & -2.72 & -3.19 & -2.21 & -2.75 \\ -1.83 & -0.49 & -1.83 & -3.87 & -2.39 & -2.96 & -2.39 & -3.87 \\ -2.51 & -2.40 & -0.09 & -2.75 & -2.21 & -3.19 & -2.72 & -4.59 \\ -2.74 & -3.06 & -1.71 & -0.97 & -1.71 & -3.06 & -2.74 & -4.78 \\ -2.72 & -3.19 & -2.21 & -2.75 & -0.09 & -2.40 & -2.51 & -4.59 \\ -2.39 & -2.96 & -2.39 & -3.87 & -1.83 & -0.49 & -1.83 & -3.87 \\ -2.21 & -3.19 & -2.72 & -4.59 & -2.51 & -2.40 & -0.09 & -2.75 \\ -1.71 & -3.06 & -2.74 & -4.78 & -2.74 & -3.06 & -1.71 & -0.97 \end{bmatrix} \tag{41}$$

Rešitev na ograji je

$$\mathbf{u} = \begin{bmatrix} -0.741832 \\ -3.82685 \\ -0.741832 \\ -0.00026735 \\ 0.741297 \\ 3.82631 \\ 0.741297 \\ -0.000268133 \end{bmatrix} \tag{42}$$

4.3 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 (32) 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;
}
```

4.4 Integracija v notranjosti

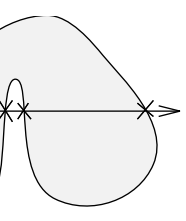
Za vsako točko v notranjosti p lahko izračunamo krivuljni integral iz enačbe (24) potem, ko imamo znane vrednosti potenciala na ograji. Ker so točke vse v notranjosti je izraz $c(p) = 1$

$$u(p) = \sum_{i=1}^{n} \int_S u^* dS_i q - \sum_{i=1}^{n} \int_S q^* dS_i u \qquad (43)$$

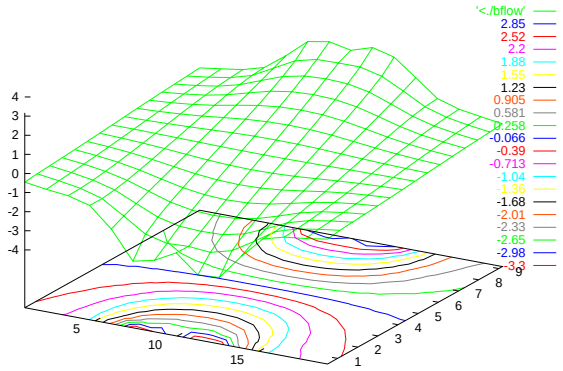
Integracija za notranjo točko p je podobna kot v enačbah (34) in (36), le da namesto točke v sredini elementa i uporabimo notranjo točko p.

$$u(p) = \sum_{j=1} g_j(p) q_j - \sum_{j=1} h_j(p) u_j \qquad (44)$$

Kontrola notranjosti. Pri računanju vrednosti potenciala v poljubni točki je potrebno zagotoviti, da se točka nahaja v notranjosti lika skozi katerega računamo pretok. Notranjost lika ugotovimo tako, da za izbrano točko (x, y) štejemo koliko daljic od vseh možnih prereže poltrak desno od točke. Če je število takih prerezov liho potem je točka v notranjosti lika, sicer je v zunanjosti. Dodatne težavi pri BEM lahko nastopijo tudi v bližini robu, zato moramo poleg kontrole notranjosti zagotoviti še, da se točka p, za katero računamo integral, ne nahaja v bližini katerega od elementov. Točke v notranjosti lahko računamo na poljubnih mestih kar pomeni, da lahko uporabimo ekvidistantno mrežo in kontroliramo notranjost. Za točke lahko uporabimo tudi generator naključnih položajev v mejah območja ali kakšen drug prikladen način kot je npr zaporedno nastacljanje računskega mesta in s tem sledenje delcu (*particle simulation*).



Slika 4: Ugotavljanje notranjosti lika



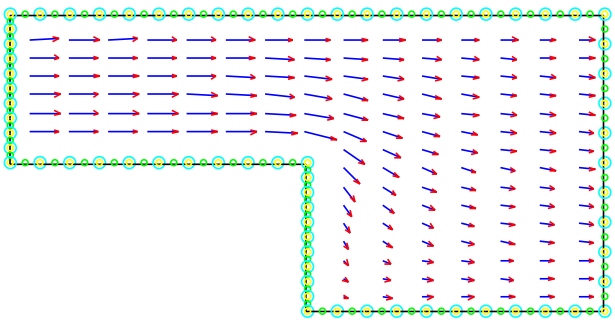
Slika 5: Hitrostni potencial in ekvipotencialne krivulje toka s slike 3

4.5 Hitrost v točki

Slika 3 kaže površino hitrostnega potenciala na v izbranih točkah. Če želimo izračunati hitrost v točki enostavno izberemo diferenčni enačbo, ki je npr. za smer x enaka

$$v_x(p) = \frac{u(p+h) - u(p-h)}{2h} \qquad (45)$$

kjer je $\pm h$ majhen odmik v smer x. Hitrost v smeri y izračunamo s premikom iz točke p v smeri y za $\pm h$.



Slika 6: Grafičen prikaz hitrosti v razširitvi kanala.

5 Grafični prikaz rezultatov

Za grafičen prikaz rezultatov v programu uporabite grafični jezik OpenGL [5, 1, 6]. Za uporabniški vmesnik uporabite knjižnico GLUT [4]. Osnovni program, ki demonstrira uporabo jezika OpenGL in knjižnice uporabniškega vmesnika GLUT 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; /* world x, y, z coords */
  GLdouble px, py, pz; /* picked window coordinates */
  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,
          "Dosezeno maksimalno stevilo tock!\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;
}
```

6 Algoritem izračuna hitrosti v točkah

1. Generiramo vhodno .txt datoteko (določitev definicijskega območja sistema), ki popisuje naš problem.
2. Določimo elemente matrike $\mathbf{H}$ z kvadraturno integracijo po enačbi (34) in (35) in $\mathbf{G}$ po enačbi (36) in (37).
3. Izračunamo hitrostni potencial u na ograji. Ko izračunamo matriki $\mathbf{H}$ in $\mathbf{G}$ s pomočjo že znanega vektorja hitrosti na ograji q in enačbe (30) izračunamo $\mathbf{u} = \mathbf{H}^{-1}\mathbf{G}\,\mathbf{q}$

   To operacijo naredimo s pomočjo LU razcepa, ki nam izvede inverzno transformacijo matrike $\mathbf{H}$ in zmnoži vrednost inverzne matrike $\mathbf{H}$ z vektorjem produkta $\mathbf{G}\,\mathbf{q}$.
4. Pravokotno območje, ki zajema celoten lik (min/max), enakomerno razdelimo po širini in višini. Pri tem koraku je potrebno ugotoviti katere točke so v notranjosti (slika 4) in dovolj oddaljene od meje, da lahko izračunamo odvod v točki z enačbo (45).
5. Če je točka v notranjosti lahko za njo izračunamo hitrostni potenciala $u(p)$ po enačbi (44). Funkcija $h_j(p)$ v notranjosti ima obliko

$$h_j(p) = \frac{-l}{4\pi} \sum_{k=1}^{4} \frac{w_k}{r_j^2(p)} (r_x(p)n_x + r_y(p)n_y)_j \,, \qquad (46)$$

   kjer je $r_j(p)$ razdalja med točko p in točko j-tega elementa na ograji, ki jo izračunamo po enačbi (38). Funkcija $g_j(p)$ ustreza enačbi (36) s tem, da namesto i-te točke izvora uporabimo točko p.
6. Hitrost v poljubni točki izračunamo s pomočjo diskretnega odvoda, ki ga podaja enačba (45). V izbranih točkah (mreža) lahko narišemo puščice, ki ponazarjajo smer in hitrost toka idealnega fluida.

7 Poročilo

Zaradi lažjega ocenjevanja in doseganja enotnih standardov, se predvideva enoten način izdelave poročila o nalogi. Poročilo o vajah se oddaja na WIKI strani vašega projekta. Prvi del poročila naj opisuje numerični program, drugi pa naj predstavi grafični del. 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 uporabljati 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.

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, kot tudi omejitve, ki so posledica vgrajenih algoritmov (velikost matrik, stabilnost, ...). Torej: poročilo mora biti napisano tako, da je možno iz njega rekonstruirati program.

Spletni vmesnik TRAC omogoča strukturirano pisanje dokumentov, ki spremljajo projekt. Vsak študent je tako na strani WikiStart svojega projekta, dolžan predstaviti svoje delo v obliki poročila v katerem predstavi projekt, komentira svoje delo in delovanje programa. Strukturirano obliko poročila je možno doseči z logičnimi ukazi, ki so podrobneje predstavljeni na strani *WikiFormatting*. Predvideno je samostojno učenje opisnega jezika za oblikovanje strani, za kar je predviden prostor v peskovniku (*SandBox*). Učenje na strani

SandBox se ne shranjuje in je predviden ravno zaradi tega, da se z njim ne smeti časovni potek dela.

Poročilo je ravno tako, kot kodo možno slediti v Timeline. Ravno tako mora biti iz Timeline razviden potek izdelave poročila.

8 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* [3]. Prikaz, primerjanje in komunikacija pri izvedbi vaj pa poteka s sistemom *TRAC* na namenskem strežniku `http://www.lecad.fs.uni-lj.si:8000/`. Vsa potrebna navodila in dodatna so dostopna na projektu 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://www.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 je predvideno v uvodnih vajah.

Za izvedbo naloge je predvideno znanje C-ja. Podrobnosti in stanje znanja bo preverjeno na uvodnih vajah.

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

```
svn co --username vaje \
    svn://www.lecad.fs.uni-lj.si/rpk/vaje
```

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://www.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"
```

9 Domače vaje

Študentom se na vajah v laboratoriju razloži osnove vektorske analize. Seštevanje, množenje s skalarjem, rotacija. Kaj je to gradient, divergenca? Operatorja nabla in laplace. Naslednje domače vaje utrjujejo in pojasnjujejo nekatere probleme pri izdelavi programa.

1. Izračunaj dolžino vektorja podanega s točkama $\vec{r}_1 = (1.2, 3.4)$, $\vec{r}_2 = (3.5, -4.3)$
   R: $l = \sqrt{(\vec{r}_2 - \vec{r}_1) \cdot (\vec{r}_2 - \vec{r}_1)}$
2. Napiši parametrično enačbo premice $\vec{p}$(t), ki gre skozi točki $\vec{r}_1$ in $\vec{r}_2$ s parametrom t, ki bo zavzel vrednosti od nič do ena v področju daljice: $\vec{p}(0) = \vec{r}_1$ in $\vec{p}(1) = \vec{r}_2$ R: $\vec{p}(t) = \vec{r}_1 + t\frac{\vec{r}_2 - \vec{r}_1}{l}$
3. Uporabi skaliranje in translacijo parametra t z uvedbo novega parametra ξ, ki bo spremenil območje daljice $\xi = [-1, 1]$
   R: Glej Gaussovo kvadraturo
4. Kaj je to normala in norma vektorja?
   R: Normala je pravokotnica na krivuljo ali površino v dani točki z normo 1.
5. Izračunaj normalo premice $\vec{n}$ z skalarnim produktom $\vec{p} \cdot \vec{n} = 0$
6. Izračunaj normalo daljice z rotacijo smeri vektorja za $-\pi/2$
   R: $\vec{n} = (p_y, -p_x)/l$
7. Napiši enačbo ravnine v vektorski obliki in preizkusi na primeru
   R: $(\vec{r} - \vec{r}_0) \cdot \vec{n} = 0$ kjer je $\vec{r}_0$ točka, ki zanesljivo leži na ravnini in $\vec{r}$ testna točka. Ali je potrebno, da je normala normirana.
8. Podana je daljica $\vec{d} = \{(2.0, 0.0), (2.0, 1.0)\}$. Izračunaj normalo $\vec{n} = (n_x, n_y)$ in integrala z Gaussovo kvadraturo na daljici za funkciji

$$h = \frac{1}{2\pi r^2}(r_x n_x + r_y n_y)$$

$$g = \frac{1}{2\pi} ln\left(\frac{1}{r}\right)$$

   kjer je $\vec{r}$ vektor od točke (0.5, 0.0) do točke na daljici.
   R: $h = -0.09358$ in $g = 0.07502$

10 Terminski načrt

Po petih 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.

Po uvodnih vajah je obisk v laboratoriju LECAD je še vedno obvezen, vendar je pričakovatirazličen 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 16. 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.

11 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:

- 20% 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.
- 5% Prisotnost in izvedba domačih nalog s katerimi si pridobimo delovni cikel.
- 35% Delovanje programa za preračun, izris hitrosti v točkah notranjosti, 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:
  - Interaktivni program za risanje lika in določanje vtoka in iztoka (5%).
  - Integracija v zaključeno aplikacijo. S tako aplikacijo je možno iterativno ponavljati vnos, preračun in prikaz brez izhoda iz aplikacije (5%).
  - Povečana stopnja interaktivnosti z uporabo miške in menujev GLUT (5%).
  - Za poljubno začetno točko v notranjosti sledite potovanju delca (5%). Animacija potovanja delca (5%).
  - Različne tehnike grafičnega vnosa podatkov (snap to grid, align, simetrija, popravljanje posameznih vozlišč, ...). 5%
  - Prostorski izris potenciala hitrosti (7%).
  - Ekvipotencialne krivulje toka (3%).
  - Merilo. Različen obseg vhodnih podatkov, ki je v osnovi -1..1. 3%

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. Timeline in WikiStart strani posameznih projektov so po uvodnih vajah vidni 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 in 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 Ticket. 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 "Ticket". 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] OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *The Official Guide to Learning OpenGL*. Addison-Wesley Publishing Company, 2007. `http://www.opengl.org/documentation/red_book/`.

[2] C.A. Brebbia. *The boundary elementh method for engineers*. Pentech press, 1978.

[3] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. Version control with subversion. O'Reilly Media., 2004. `http://svnbook.red-bean.com/`.

[4] Mark J. Kilgard. *The OpenGL Utility Toolkit (GLUT)*. Silicon Graphics, Inc., 11 1996. `http://www.opengl.org/documentation/specs/glut/glut-3.spec.pdf`.

[5] Leon Kos. Računalniška grafika. Gradivo za vaje v PDF, Fakulteta za strojništvo v Ljubljani, 4 2002. `http://www.lecad.si/education/gradivo/software/opengl-intro.pdf`.

[6] David Rogelberg, editor. *OpenGL Reference Manual*. Addison-Wesley Publishing Company, 1992. `http://www.opengl.org/documentation/blue_book/`.

[7] J. Trevelyan. *Boundary elements for engineers*. Computational Mechanic Publications, 1994.