SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 612

SIMULACIJA FLUIDA METODOM LBM

Kim Staničić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 612

SIMULACIJA FLUIDA METODOM LBM

Kim Staničić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 612

Pristupnica:	**Kim Staničić (0036522105)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Željka Mihajlović

Zadatak: **Simulacija fluida metodom LBM**

Opis zadatka:

Proučiti načine simulacije fluida u dvodimenzionalnom prostoru. Proučiti rešetkaste Boltzmanove metode LBM (engl. Lattice Boltzmann Methods) u domeni simulacije fluida. Razraditi prikaz simulacije dinamičkog ponašanja fluida u 2D prostoru temeljen na proučenoj metodi. Načiniti testiranje na nizu primjera. Analizirati i ocijeniti ostvarene rezultate. Diskutirati upotrebljivost ostvarenih rezultata kao i moguća proširenja. Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 10. lipnja 2022.

Sadržaj

Uvod

Simulacija fluida je područje računalne grafike koje nastoji na računalu prikazati dinamičko ponašanje fluida pod utjecajem različitih uvjeta. Ovisno o primjeni, mogu se koristiti algoritmi kojima je fokus na fizikalnoj dosljednosti (npr. u području biološkog inženjerstva, aerodinamike i sl.) ili vizualnoj dosljednosti (npr. vizualni efekti u filmovima i video igrama). Kod fizikalne dosljednosti se minimizira odstupanje od fizikalnih pravila koja opisuju ponašanje fluida, tj. pokušava se ostvariti matematička točnost. Kod vizualne dosljednosti nije bitno ostvariti ponašanje koje je realistično, već je dovoljno da ono samo izgleda realistično. Razlog odbacivanja točnosti je što takav pristup omogućuje jednostavniji izračun. Time se dobiva na brzini koja je ključna za animacije u stvarnom vremenu (engl. real-time animations).

Većina metoda simulacije fluida temelji se na rješavanju Navier-Stokes jednadžbi. One opisuju ponašanje fluida ovisno o predefiniranim parametrima i daju precizne rezultate za širok spektar ulaznih vrijednosti. Problem je što je rješavanje Navier-Stokes jednadžbi računalno zahtjevno te se uvijek još pokušavaju pronaći načini optimiziranja ovih algoritama.

Alternativni pristup simulaciji fluida je korištenje rešetkastih Boltzmannovih metoda. One za simulaciju ne koriste Navier-Stokes jednadžbe, već koriste statističku distribuciju fluida unutar rešetke. Ovakav pristup učinkovitiji je od rješavanja Navier-Stokes jednadžbi zbog mogućnosti analitičkog izračuna i paralelizacije procesa, ali je ograničen na male vrijednosti Machovog broja (engl. Mach number).

U ovom radu opisane su Navier-Stokes jednadžbe i metode simulacije fluida temeljene na njima. Nakon toga su obrađene rešetkaste Boltzmannove metode te je implementirana simulacija dinamičkog ponašanja fluida u dvodimenzionalnom prostoru koristeći rešetkastu Boltzmannovu metodu. Ponašanje simulacije testirano je i prikazano kroz nekoliko primjera. Za implementaciju je korišten programski jezik Java.

1 Navier-Stokes jednadžbe (NSE)

Navier-Stokes jednadžbe opisuju precizan matematički model toka za većinu fluida koji se pojavljuju u prirodi [1]. Njihovim rješavanjem možemo predvidjeti brzine i gustoće fluida kroz vrijeme (uz unaprijed definirane granične uvjete) [2]. Zbog njihove kompleksnosti, jednadžbe je moguće analitički riješiti samo za jednostavne slučajeve. Njihova izravna implementacija se stoga ne preporučuje.

Većina metoda simulacije fluida temelji se na diskretizaciji Navier-Stokes jednadžbi, stoga je njihovo razumijevanje ključno za razumijevanje najkorištenijih algoritama u prikazu fluida. Treba imati na umu da ove jednadžbe opisuju koja pravila treba poštivati pri implementaciji, a ne kako implementirati simulaciju. Pri osmišljavanju metoda se često ubacuju dodatne pretpostavke koje sužavaju područje fluida pokrivenih pretpostavkama, ali omogućuju simulaciju s prihvatljivim vremenom izvođenja.

Za nestlačive Newtonovske fluide (engl. incompressible Newtonian fluids) vrijede jednadžbe (1) i (2). Prva jednadžba (1) opisuje očuvanje mase fluida, gdje varijabla u predstavlja brzinu fluida po komponentama. Druga jednadžba (2) opisuje Newtonov drugi zakon (brzina promjene količine gibanja fluida jednaka je sili koja djeluje na fluid). Varijable ρ, u, t, p predstavljaju gustoću, brzinu, vrijeme i tlak (tim redoslijedom). Jednadžba opisuje ovisnost akceleracije (lijeva strana jednadžbe) o unutarnjim i vanjskim silama (desna strana jednadžbe). Unutarnja sila je posljedica sudaranja čestica (engl. particles) fluida, a ona ovisi o promjeni tlaka i viskoznosti fluida (prvi i drugi član na desnoj strani jednadžbe). Fluid se giba od područja većeg tlaka prema području manjeg tlaka te je unutarnja sila veća što je razlika u tlakovima unutar fluida veća. Veća viskoznost uzrokuje veće trenje među česticama te se time brzine lakše prenose s čestice na česticu. Vanjska sila F je zbroj svih ostalih sila koje djeluju na fluid, a to su najčešće gravitacijska sila i prepreke unutar fluida.

$$\nabla \cdot u = 0 \tag{1}$$

$$\rho \frac{Du}{Dt} = -\nabla p + \mu \nabla^2 u + \rho F \tag{2}$$

2 Načini simuliranja fluida

Zbog kompleksnosti izračuna ne postoji jedna metoda simuliranja fluida koja je najbolja. Razne metode su često specijalizirane za određene slučajeve te su zadovoljavajuće u svojem području primjene, ali van toga nisu upotrebljive.

Metode simuliranja fluida mogu se klasificirati u dvije grupe – Eulerove i Langrageove [3].

Eulerove metode kao osnovu opisa fluida koriste mrežu. Područje simulacije se dijeli na ćelije te se izračuni obavljaju za svaku ćeliju. Fluid se promatra na makroskopskoj razini – prati se kretanje gustoće i brzine kroz rešetke te se zanemaruju pojedine čestice.

Lagrangeove metode za opis fluida koriste pojedine čestice. One se slobodno gibaju kroz prostor te se za svaku od njih mogu pamtiti masa, volumen, brzina i slično. Ove metode fluid promatraju na mikroskopskoj razini.

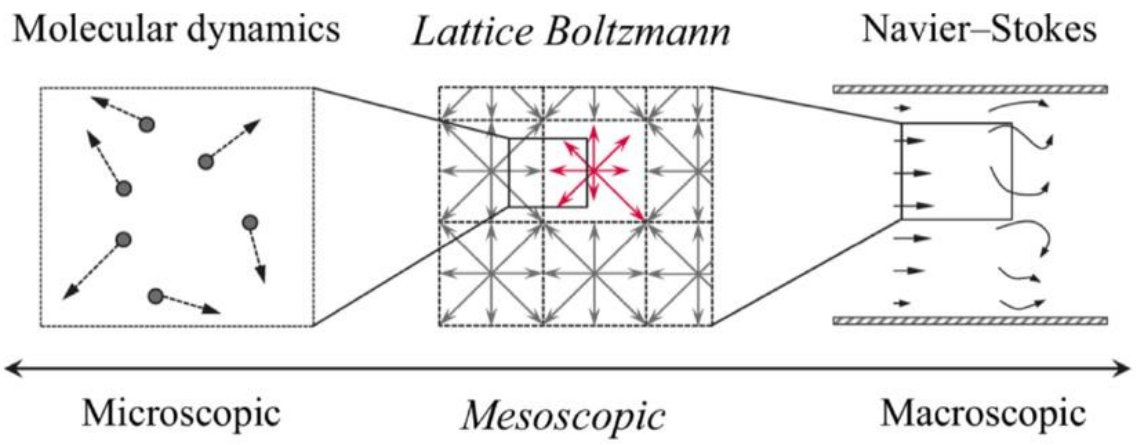
Primjer poznate metode je FDM (engl. Finite Difference Method) koja diskretizira NSE na skupu točaka koje pamte vrijednosti tlaka i brzine fluida u tim točkama [3]. Pri izračunu derivacija koristi se Taylorov red. Ovakav pristup čest je pri simulacijama vremenskih prilika.

Još neke značajne metode su Eulerove metode FEM (engl. Finite Element Method) i FVM (engl. Finite Volume Method) te Langrangeova metoda SPH (engl. Smoothed Particle Hydrodynamics) [3]. Ove se metode također temelje na diskretizaciji Navier-Stokes jednadžbi.

3 Rešetkaste Boltzmannove metode (LBM)

Rešetkaste Boltzmannove metode (engl. Lattice Boltzmann Methods) su alternativa tradicionalnim metodama simulacije fluida jer se LBM ne temelji na numeričkom rješavanju Navier-Stokes jednadžbi. Osnovna je ideja ovih metoda računanje statističkih distribucija po rešetkama [4]. Fluid se sastoji od fiktivnih čestica koje se sudaraju i propagiraju u odgovarajuće susjedne ćelije. LBM ne promatra svaku česticu zasebno, već unutar svake ćelije pamti podjelu u grupe čestica koje imaju slična svojstva.

Zbog navedenih svojstava, LBM fluid promatra na mezoskopskoj (engl. mesoscopic) razini - razina između mikroskopskog i makroskopskog pogleda. Usporedba navedenih pogleda vidljiva je na slici (Sl. 3.1).



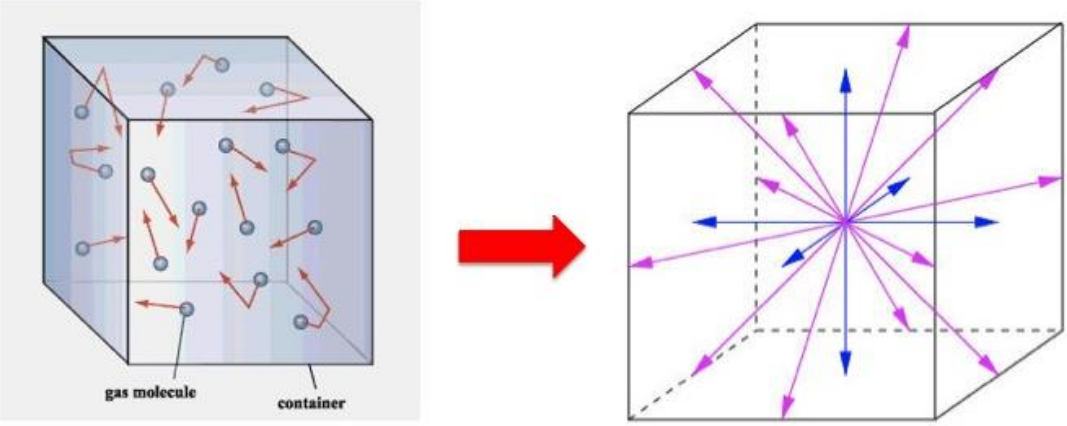
Sl. 3.1 Usporedba različitih pogleda na fluid [5]

Glavna prednost LBM nad ostalim metodama je potpuna prilagođenost paralelizaciji. Glavni nedostatak je ograničenost algoritma na relativno male Machove brojeve.

3.1 Matematičke osnove LBM

LBM ima statistički pristup rješavanju problema simulacije fluida. Osnovna ideja je prikaz fluida pomoću vjerojatnosne funkcije $f(\vec{x}, \vec{v}; t)$ koja opisuje vjerojatnost pronalaska dane čestice na poziciji $\vec{x}$ u trenutku t s brzinom $\vec{v}$ [6].

Zbog efikasnije implementacije, LBM diskretizira brzinu tako da unaprijed definira konačan broj brzina te time ograničava moguće smjerove gibanja čestica. Metode se klasificiraju prema odabranom broju dimenzija *n* i broju brzina *m*. Zapis je oblika 'D*n*Q*m*' (npr. D2Q9, D3Q15 i sl.). Primjer takve diskretizacije vidljiv je na slici (Sl. 3.2).

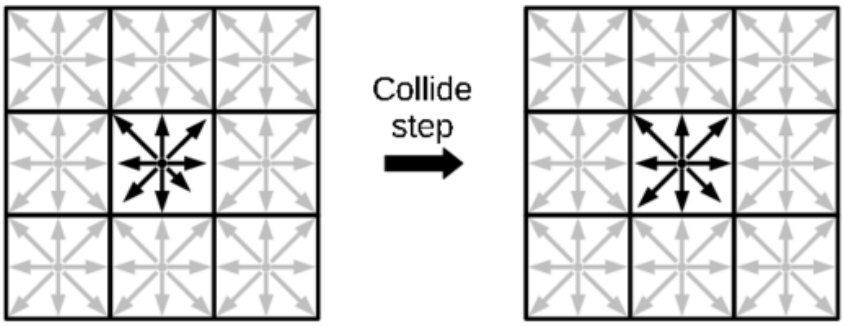


Sl. 3.2 Prikaz diskretizacije pravih čestica u LBM čestice [6]

LBM ima dva glavna koraka – sudar čestica (engl. collision step) i propagacija čestica (engl. streaming step). Oni se iterativno ponavljaju tijekom cijele simulacije, a prije prelaska na iduću iteraciju treba osigurati pravilno ponašanje za granične slučajeve. Tu se uvodi i treći korak – odbijanje čestica (engl. bounce-back step).

3.1.1 Sudar čestica

Ideja ovog koraka je preraspodjela gustoće po komponentama (smjerovima) unutar svake ćelije. Imaginarne čestice se sudaraju te time mijenjaju smjer, tj. brzinu. Zbroj svih gustoća unutar ćelije ostaje konstantan u ovom koraku - mora biti jednak prije i nakon sudara. Primjer sudara prikazan je na slici (Sl. 3.3).



Sl. 3.3 Prikaz raspodjele gustoće po komponentama prije i nakon sudara [7]

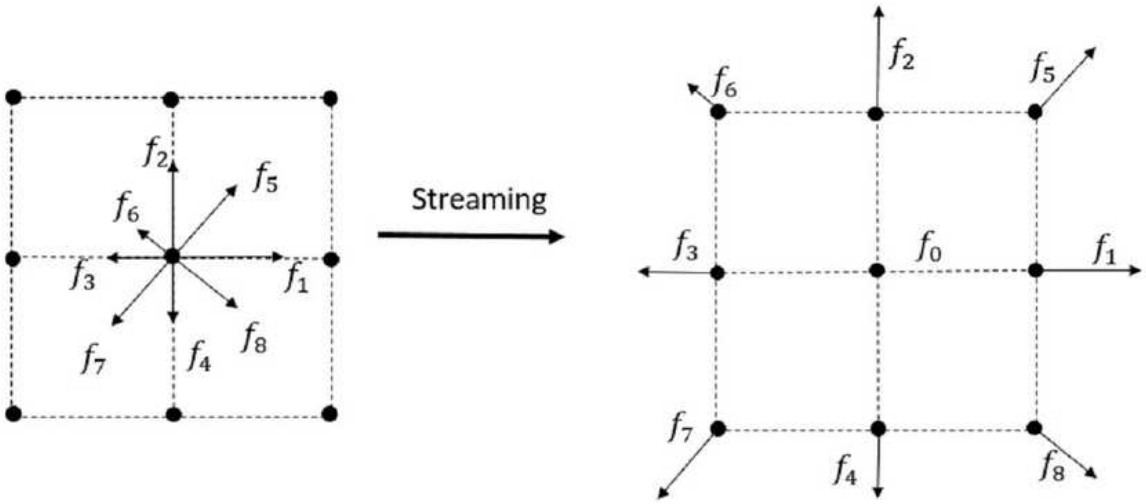
Jednadžba (3) prikazuje općeniti oblik Lattice Boltzmannove jednadžbe za Bhatnagar-Gross-Krook (BGK) model relaksacije. Funkcija f_i predstavlja stvarnu gustoću zadane točke u određenom trenutku, a f_i^{eq} predstavlja gustoću u ravnotežnom stanju. Indeks *i* označava o kojoj komponenti je riječ te poprima vrijednosti od 1 do *m*. Parametar ω opisuje kojom će se brzinom fluid približavati ravnotežnom stanju. Viskoznost utječe na ovaj parametar.

$$f_i(\vec{x}, t + \delta_t) = f_i(\vec{x}, t) + \omega \left(f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t) \right) \tag{3}$$

3.1.2 Propagacija čestica

Cilj ovog koraka je propagirati novoizračunate gustoće u susjedne ćelije u odgovarajućem smjeru. Vektor $\overrightarrow{e_i}$ u jednadžbi (4) definira taj smjer. Nakon ovog koraka sve su gustoće ili u istoj ćeliji kao na početku koraka ili u susjednoj ćeliji, ali nikad dalje od toga. Primjer propagacije prikazan je na slici (Sl. 3.4).

$$f_i(\vec{x} + \overrightarrow{e_i}, t + \delta_t) = f_i(\vec{x}, t) \tag{4}$$

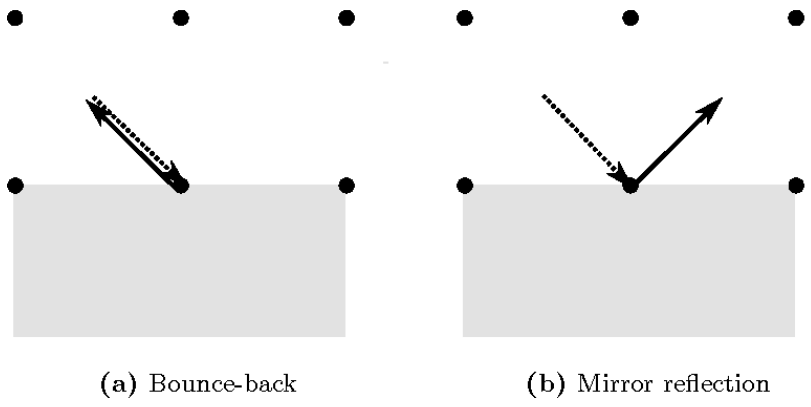


Sl. 3.4 Prikaz komponenata prije i nakon propagacije [8]

3.1.3 Odbijanje čestica

Pri simulaciji treba voditi računa o mogućim graničnim uvjetima. To su najčešće prepreke unutar rešetke fluida (npr. zid). U prethodnom koraku se moglo dogoditi da su se čestice pomakle u prepreku umjesto da se od nje odbiju. Ovaj korak osigurava da se taj problem riješi tako da svaku dolaznu česticu odbijemo u suprotnom smjeru (5). Primjer takvog odbijanja prikazan je na slici (Sl. 3.5 a) gdje je upadni vektor prikazan isprekidanom, a odbijeni punom linijom. Alternativni pristup je zrcalno odbijanje (Sl. 3.5 b) gdje se upadni (isprekidana linija) i odbijeni vektor (puna linija) ne moraju nalaziti u istoj ćeliji.

$$f_{in}(\vec{x}) = f_{out}(\vec{x}) \tag{5}$$



Sl. 3.5 Prikaz običnog (a) i zrcalnog (b) odbijanja [9]

3.1.4 LBM jedinice

Prije ubacivanja vrijednosti u jednadžbe, trebamo ih pretvoriti iz SI jedinica u LBM jedinice. Konverzija je opisana jednadžbama (6) i (7). Varijable δ_x i δ_t predstavljaju jedan prostorni i vremenski LBM korak, a varijable *L*, *N* i *c* označavaju duljinu cijele rešetke, broj ćelija smještenih duž duljine rešetke *L* te lokalnu brzinu zvuka.

$$\delta_x = L \,/\, N \tag{6}$$

$$\delta_t = \delta_x \,/\, c \tag{7}$$

Korištenje stvarne vrijednosti brzine zvuka *c* može uzrokovati prekratke vremenske korake δ_t. Rješenje toga je povećanje Machovog broja (8) koji predstavlja omjer brzine fluida *u* i lokalne brzine zvuka *c*. Kako se ne bi poremetila vrijednost Reynoldovog broja (engl. Reynolds number) (9), povećava se i vrijednost kinematičke viskoznosti ν.

$$M = \frac{u}{c} \tag{8}$$

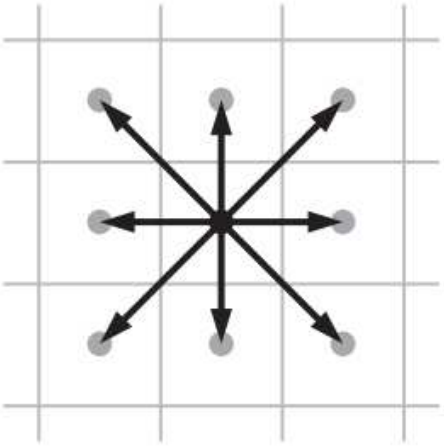
$$Re = \frac{uL}{\nu} \tag{9}$$

4 Implementacija

Za implementaciju je korišten programski jezik Java. Značajke sustava inspirirane su Weber State University projektom [10] i Dan Schroederovom implementacijom simulacije fluida [11].

4.1 Opis sustava

Sustav koristi D2Q9 diskretizaciju – ima dvije dimenzije i devet mogućih smjerova. Prikaz i vrijednosti smjerova su na slici (Sl. 4.1). Vektori $\vec{e_i}$ su izraženi LBM jedinicama.



$$\vec{e}_0 = 0$$
$$\vec{e}_1 = (1,0) \qquad \vec{e}_5 = (1,1)$$
$$\vec{e}_2 = (0,1) \qquad \vec{e}_6 = (-1,1)$$
$$\vec{e}_3 = (-1,0) \qquad \vec{e}_7 = (-1,-1)$$
$$\vec{e}_4 = (0,-1) \qquad \vec{e}_8 = (1,-1)$$

Sl. 4.1 Diskretizirani smjerovi [10]

Svakom je smjeru dodana određena vjerojatnost – težina w_i. Zbroj težina mora biti jedan. Izabrane težine prikazane su jednadžbama (10).

$$w_0 = \frac{4}{9}, \quad w_1 = w_2 = w_3 = w_4 = \frac{1}{9}, \quad w_5 = w_6 = w_7 = w_8 = \frac{1}{36} \qquad (10)$$

Za svaku komponentu (smjer) trebamo moći izračunati iznos gustoće u ravnotežnom stanju. Fiksiranjem brzine zvuka *c* na 1 dolazimo do jednadžbe (11). Varijable ρ, w_i, $\vec{u}$ i $\vec{e_i}$ predstavljaju sveukupnu gustoću rešetke (zbroj gustoća po komponentama), težinu, brzinu fluida na makroskopskoj razini te smjer.

$$n_i^{eq} = \rho w_i \left[1 + 3\vec{e_i} \cdot \vec{u} + \frac{9}{2}(\vec{e_i} \cdot \vec{u})^2 - \frac{3}{2}|\vec{u}|^2\right] \tag{11}$$

Izračun nove vrijednosti gustoće dan je jednadžbom (12) gdje novu gustoću n_i^{new} računamo pomoću gustoće iz prethodnog koraka n_i^{old} i pomoću faktora ω koji određuje brzinu približavanja fluida ravnotežnom stanju. Ovisnost kinematičke viskoznosti ν o faktoru ω dana je jednadžbom (13).

$$n_i^{new} = n_i^{old} + \omega\left(n_i^{eq} - n_i^{old}\right) \tag{12}$$

$$\nu = \frac{1}{3}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{13}$$

4.2 Strukture podataka

Podaci o ćelijama pamte se u matricama gdje indeksi matrice predstavljaju x i y koordinatu ćelije unutar rešetke. Vrijednosti gustoće po komponentama i sveukupna gustoća (zbroj po svim komponentama) pamte se u matricama prikazanim na slici (Sl. 4.2). Varijable `rows` i `columns` predstavljaju broj redaka i stupaca rešetke.

```
double[][] densityNORTH = new double[columns][rows];
double[][] densitySOUTH = new double[columns][rows];
double[][] densityEAST = new double[columns][rows];
double[][] densityWEST = new double[columns][rows];
double[][] densityNORTHEAST = new double[columns][rows];
double[][] densityNORTHWEST = new double[columns][rows];
double[][] densitySOUTHEAST = new double[columns][rows];
double[][] densitySOUTHWEST = new double[columns][rows];
double[][] densityMIDDLE = new double[columns][rows];
double[][] density = new double[columns][rows];
```

Sl. 4.2 Strukture podataka za pamćenje gustoće

Za ćelije se pamti i vrijednost brzine fluida na toj poziciji po x i y komponentama. Matrica `obstacles` za pojedinu ćeliju poprima vrijednost `true` ako je na zadanoj poziciji prepreka, a `false` ako je na toj poziciji fluid (Sl. 4.3).

```
double[][] x_comp = new double[columns][rows];
double[][] y_comp = new double[columns][rows];
boolean[][] obstacles = new boolean[columns][rows];
```

Sl. 4.3 Strukture podataka za pamćenje brzine i prepreka

4.3 Inicijalizacija

Rubovi rešetke pri simulaciji imaju ulogu izvora – lijevi i desni rub se ponašaju kao jedna cjelina iz koje teče fluid konstantnom brzinom v koja se zadaje prije pokretanja simulacije. Brzina v se zadaje preko komponenata v_x i v_y. Primjerice, za $v_x > 0$ i $v_y = 0$ fluid će prividno izvirati s lijeve strane i teći prema desno. Gornji i donji rub, ovisno o postavkama (Poglavlje 4.8), može također biti izvor s brzinom v ili izvor s brzinom v_x (zanemaruje se v_y komponenta).

U konstruktoru klase LBM pozivaju se pomoćne metode za inicijalizaciju vrijednosti fluida. Dio njih prikazan je na slici (Sl. 4.4).

```
calculate_init_densities();
initFluidZero();
make_obstacle_tunnel();
```

Sl. 4.4 Isječak iz konstruktora klase LBM

Metoda `calculate_init_densities` računa vrijednosti gustoće po komponentama pomoću jednadžbe (11) za rubove rešetke uz $\rho = 1$ i uz zadanu brzinu v. Te se vrijednosti pohranjuju u pomoćne varijable koje onda koristi metoda `initFluidZero`.

Metoda `initFluidZero` iterira po svim ćelijama rešetke te postavlja inicijalnu vrijednost gustoće po komponenata na nulu (osim za komponentu pridruženoj vektoru smjera (0, 0), koju postavlja na 1). Vrijednosti rubnih ćelija inicijalizira na prethodno izračunate vrijednosti (metoda `calculate_init_densities`).

Metoda `make_obstacle_tunnel` u rešetku umeće prepreku zadanog oblika. U ovom slučaju postavlja tunel. Ostale opcije razmotrene su u kasnijim poglavljima (Poglavlje 4.8).

U konstruktoru se također inicijaliziraju početne pozicije pomoćnih čestica koje služe za lakšu vizualizaciju toka fluida.

4.4 Metoda *collision*

Metoda `collision` predstavlja korak sudaranja čestica. Metoda iterira po svim ćelijama (osim rubnih koje su izvori) te za svaku ćeliju koja nije prepreka (vrijednost matrice `obstacles` za tu poziciju je `false`) računa ukupnu gustoću i iznos brzine po komponentama (Sl. 4.5). Varijable `x` i `y` predstavljaju poziciju ćelije unutar rešetke.

```
density_total = densityMIDDLE[x][y] + densityNORTH[x][y] + densitySOUTH[x][y] + densityEAST[x][y]
        + densityWEST[x][y] + densityNORTHEAST[x][y] + densityNORTHWEST[x][y]
        + densitySOUTHEAST[x][y] + densitySOUTHWEST[x][y];

density[x][y] = density_total;

//velocity components
if (density_total > 0) {
    vx = (densityEAST[x][y] + densityNORTHEAST[x][y] + densitySOUTHEAST[x][y] - densityWEST[x][y]
            - densityNORTHWEST[x][y] - densitySOUTHWEST[x][y]) / density_total;
    vy = (densityNORTH[x][y] + densityNORTHEAST[x][y] + densityNORTHWEST[x][y] - densitySOUTH[x][y]
            - densitySOUTHEAST[x][y] - densitySOUTHWEST[x][y]) / density_total;
} else {
    vx = 0;
    vy = 0;
}

x_comp[x][y] = vx;
y_comp[x][y] = vy;
```

Sl. 4.5 Izračun gustoće i brzine za zadanu ćeliju

Uz pomoć tih vrijednosti računa novu raspodjelu gustoće po komponentama nakon sudara (Sl. 4.6). Varijabla `omega` predstavlja faktor ω (zadaje se prije početka simulacije), a varijable `vx_squared` i `vy_squared` su kvadratne vrijednosti komponenata brzine v_x i v_y.

```
//new density redistribution after collision
densityMIDDLE[x][y] += omega
        * (4.0 / 9 * density_total * (1 - 1.5 * (vx_squared + vy_squared)) - densityMIDDLE[x][y]);
densityNORTH[x][y] += omega * (1.0 / 9 * density_total
        * (1 + 3 * vy + 4.5 * vy_squared - 1.5 * (vx_squared + vy_squared)) - densityNORTH[x][y]);
densitySOUTH[x][y] += omega * (1.0 / 9 * density_total
        * (1 - 3 * vy + 4.5 * vy_squared - 1.5 * (vx_squared + vy_squared)) - densitySOUTH[x][y]);
densityEAST[x][y] += omega * (1.0 / 9 * density_total
        * (1 + 3 * vx + 4.5 * vx_squared - 1.5 * (vx_squared + vy_squared)) - densityEAST[x][y]);
densityWEST[x][y] += omega * (1.0 / 9 * density_total
        * (1 - 3 * vx + 4.5 * vx_squared - 1.5 * (vx_squared + vy_squared)) - densityWEST[x][y]);
densityNORTHEAST[x][y] += omega * (1.0 / 36 * density_total * (1 + 3 * vx + 3 * vy
        + 4.5 * (vx_squared + vy_squared + 2 * vx * vy) - 1.5 * (vx_squared + vy_squared))
        - densityNORTHEAST[x][y]);
densityNORTHWEST[x][y] += omega * (1.0 / 36 * density_total * (1 - 3 * vx + 3 * vy
        + 4.5 * (vx_squared + vy_squared - 2 * vx * vy) - 1.5 * (vx_squared + vy_squared))
        - densityNORTHWEST[x][y]);
densitySOUTHEAST[x][y] += omega * (1.0 / 36 * density_total * (1 + 3 * vx - 3 * vy
        + 4.5 * (vx_squared + vy_squared - 2 * vx * vy) - 1.5 * (vx_squared + vy_squared))
        - densitySOUTHEAST[x][y]);
densitySOUTHWEST[x][y] += omega * (1.0 / 36 * density_total * (1 - 3 * vx - 3 * vy
        + 4.5 * (vx_squared + vy_squared + 2 * vx * vy) - 1.5 * (vx_squared + vy_squared))
        - densitySOUTHWEST[x][y]);
```

Sl. 4.6 Redistribucija gustoće unutar ćelije

4.5 Metoda *stream*

Metoda `stream` predstavlja korak propagacije čestica. Ideja metode je da se vrijednosti iz matrica koje pamte gustoće translatiraju u smjeru komponente koju predstavljaju. Primjerice, vrijednosti iz matrice `densityNORTH`, koja je predstavljena vektorom smjera (0,1), će se pomaknuti za jedno mjesto prema gore. Vrijednosti matrice `densityMIDDLE` se ne translatiraju jer je njen vektor smjera (0,0). Dio metode koji provodi opisani postupak prikazan je na slici (Sl. 4.7).

```
for (int x=1; x<columns-1; x++) {
    for (int y=1; y<rows-1; y++) {
        densitySOUTHWEST[x][y] = densitySOUTHWEST[x+1][y+1];
        densitySOUTH[x][y] = densitySOUTH[x][y+1];
        densityWEST[x][y] = densityWEST[x+1][y];

    }
}

for (int x=columns-2; x>0; x--) {
    for (int y=1; y<rows-1; y++) {
        densitySOUTHEAST[x][y] = densitySOUTHEAST[x-1][y+1];

    }
}

for (int x=columns-2; x>0; x--) {
    for (int y=rows-2; y>0; y--) {
        densityNORTHEAST[x][y] = densityNORTHEAST[x-1][y-1];
        densityNORTH[x][y] = densityNORTH[x][y-1];
        densityEAST[x][y] = densityEAST[x-1][y];

    }
}

for (int x=1; x<columns-1; x++) {
    for (int y=rows-2; y>0; y--) {
        densityNORTHWEST[x][y] = densityNORTHWEST[x+1][y-1];

    }
}
```

Sl. 4.7 Propagacija gustoće u prikladne ćelije

4.6 Metoda *bounce*

Metoda `bounce` predstavlja korak odbijanja fluida od prepreke. Dio gustoće se u prethodnom koraku propagirao na područje prepreke. Ovaj korak to rješava propagiranjem gustoće u ćeliju iz koje je došla, ali u suprotnom smjeru. Nakon toga postavlja svoju gustoću na nula. Metoda `bounce` prikazana je na slici (Sl. 4.8).

```
private void bounce() {
    for (int x=1; x<columns-1; x++) {
        for (int y=1; y<rows-1; y++) {
            if(obstacles[x][y]) {
                densityNORTH[x][y+1] += densitySOUTH[x][y];
                densitySOUTH[x][y-1] += densityNORTH[x][y];
                densityEAST[x+1][y] += densityWEST[x][y];
                densityWEST[x-1][y] += densityEAST[x][y];
                densityNORTHEAST[x+1][y+1] += densitySOUTHWEST[x][y];
                densityNORTHWEST[x-1][y+1] += densitySOUTHEAST[x][y];
                densitySOUTHEAST[x+1][y-1] += densityNORTHWEST[x][y];
                densitySOUTHWEST[x-1][y-1] += densityNORTHEAST[x][y];

                densitySOUTHWEST[x][y] = 0;
                densitySOUTHEAST[x][y] = 0;
                densityNORTHEAST[x][y] = 0;
                densityNORTHWEST[x][y] = 0;
                densityNORTH[x][y] = 0;
                densityEAST[x][y] = 0;
                densitySOUTH[x][y] = 0;
                densityWEST[x][y] = 0;
            }
        }
    }
}
```

Sl. 4.8 Odbijanje čestica od prepreke

4.7 Metoda *animate*

Metoda `animate` iterativno poziva funkcije `collision`, `stream` i `bounce` do kraja simulacije (prekid rada programa). Svakih nekoliko koraka osvježava prikaz grafičkog sučelja na kojem se prikazuje simulacija. Pozivom ove metode pokreće se simulacija.

4.8 Podesivi parametri

Simulacija će se različito ponašati ovisno o postavljenim inicijalnim parametrima. Implementacija nudi nekoliko podesivih varijabli čiji utjecaj je prikazan pri testiranju (Poglavlje 5).

Korisnik može podesiti brzinu fluida koja dolazi iz izvora (rubovi) podešavanjem varijabli `vx_init` i `vy_init`. Lijevi i desni rub uvijek u obzir uzimaju obje komponente, a `boolean` varijabla `horizontal_borders_y` utječe na to hoće li gornji i donji rub uzimati u obzir samo v_x komponentu brzine (`horizontal_borders_y = false`) ili obje komponente brzine (`horizontal_borders_y = true`).

Dimenzija rešetke se može podesiti mijenjanjem varijabli `columns` i `rows`.

Također se može podesiti vrijednost varijable `omega` koja utječe na viskoznost fluida (omega i viskoznost su obrnuto proporcionalni).

Varijable `switch_x` i `switch_y`, ako su postavljene na `true`, periodično mijenjanju smjer brzine izvora (npr. za x komponentu se periodično izvršava naredba `vx_init = -1 * vx_init`). Koliko je česta ta promjena se također može podesiti.

Varijabla `display_particles` nudi opciju prikazivanja pomoćnih čestica u fluidu za lakše praćenje kretanja fluida.

Varijabla `display` utječe na način prikazivanja fluida – ovisno o vrijednosti varijable, na grafičkom sučelju se prikazuje gustoća, iznos brzine (korijen zbroja kvadrata komponenata v_x *i* v_y), v_x komponenta brzine ili v_y komponenta brzine fluida.

Primjer početnih uvjeta simulacije podešavanjem navedenih varijabli prikazan je na slici (Sl. 4.9).

```
//values to change in order to experiment:
double vx_init = 0.05;
double vy_init = 0.04;
int columns = 600; int rows = 200;
double omega = 1.5;
boolean switch_x = true; boolean switch_y = false;
boolean horizontal_borders_y = true;
boolean display_particles=true;
Display display = Display.SPEED;
```

Sl. 4.9 Primjer postavljanja podesivih parametara

U konstruktoru se za inicijalizaciju fluida može izabrati metoda `initFluid` ili `initFluidZero`. Metoda `initFluidZero` već je opisana (Poglavlje 4.3), a `initFluid` inicijalizira gustoće unutar svake rešetke na iste vrijednosti (jednake vrijednostima rubova, tj. izvora). Osnovna razlika je da `initFluidZero` inicijalizira statički, a `initFluid` dinamički fluid.

U konstruktoru se također može izabrati vrsta prepreke (ili prepreka) koje se nalaze u rešetci. To se postiže pozivanjem jednom ili više od navedenih metoda – `make_obstacle_tunnel`, `make_obstacle_tunnel_upper`, `make_obstacle_bowl`, `make_obstacle_arrow_left`, `make_obstacle_arrow_right`, `make_obstacle_x`, `make_obstacle_rectangles`. Poziv ovih metoda nije nužan, ali simulacija bez prepreka nema zanimljiva svojstva.

5 Testiranje

Podešavanjem parametara može se modelirati mnogo slučajeva za simulaciju fluida. Ovdje su prikazani utjecaji nekih parametara na ponašanje fluida. Broj redaka je u svim primjerima 200, a broj stupaca 600. Osim ako je navedeno suprotno, koristi se metoda `initFluidZero`, varijabla `omega` je postavljena na 1, a varijable `switch_x`, `switch_y` i `horizontal_borders_y` na `false`. Varijabla `display_particles` je, zbog lakšeg snalaženja na slikama, uvijek postavljena na `true`.

Prepreke su prikazane crnom, a pomoćne čestice bijelom bojom. Za prikaze vrijednosti v_x i v_y komponenti tirkizna boja označava vrijednost nula. Ona se interpolira preko plave, ljubičaste do crvene boje za pozitivne vrijednosti (crvena je najveća vrijednost). Za negativne vrijednosti se od tirkizne interpolira preko zelene, žute, narančaste do crvene boje (crvena je najveća vrijednost u suprotnom smjeru).

Brzina i gustoća, čije su vrijednosti uvijek veće ili jednake nuli, se interpoliraju od žute preko zelene, tirkizne, plave i ljubičaste do crvene. Žuta je za najmanje, a crvena za najveće vrijednosti.

Pri zadavanju vrijednosti v_y treba voditi računa o tome da koordinatni sustav u Javi ima ishodište (0,0) u gornjem lijevom kutu te y vrijednosti rastu prema dolje, tj. pozitivna vrijednost varijable `vy_init` predstavlja tok od vrha prema dnu rešetke.

5.1 Utjecaj viskoznosti

Preporučeni raspon varijable `omega` je između 0 i 2 [10]. Ovdje su razmatrani slučajevi za 0.5 (veća viskoznost) i 1.5 (manja viskoznost).

Varijabla `vx_init` je u oba slučaja postavljena na 0.1, a varijabla `vy_init` na 0. Prepreka je umetnuta u rešetku metodom `make_obstacle_bowl`.

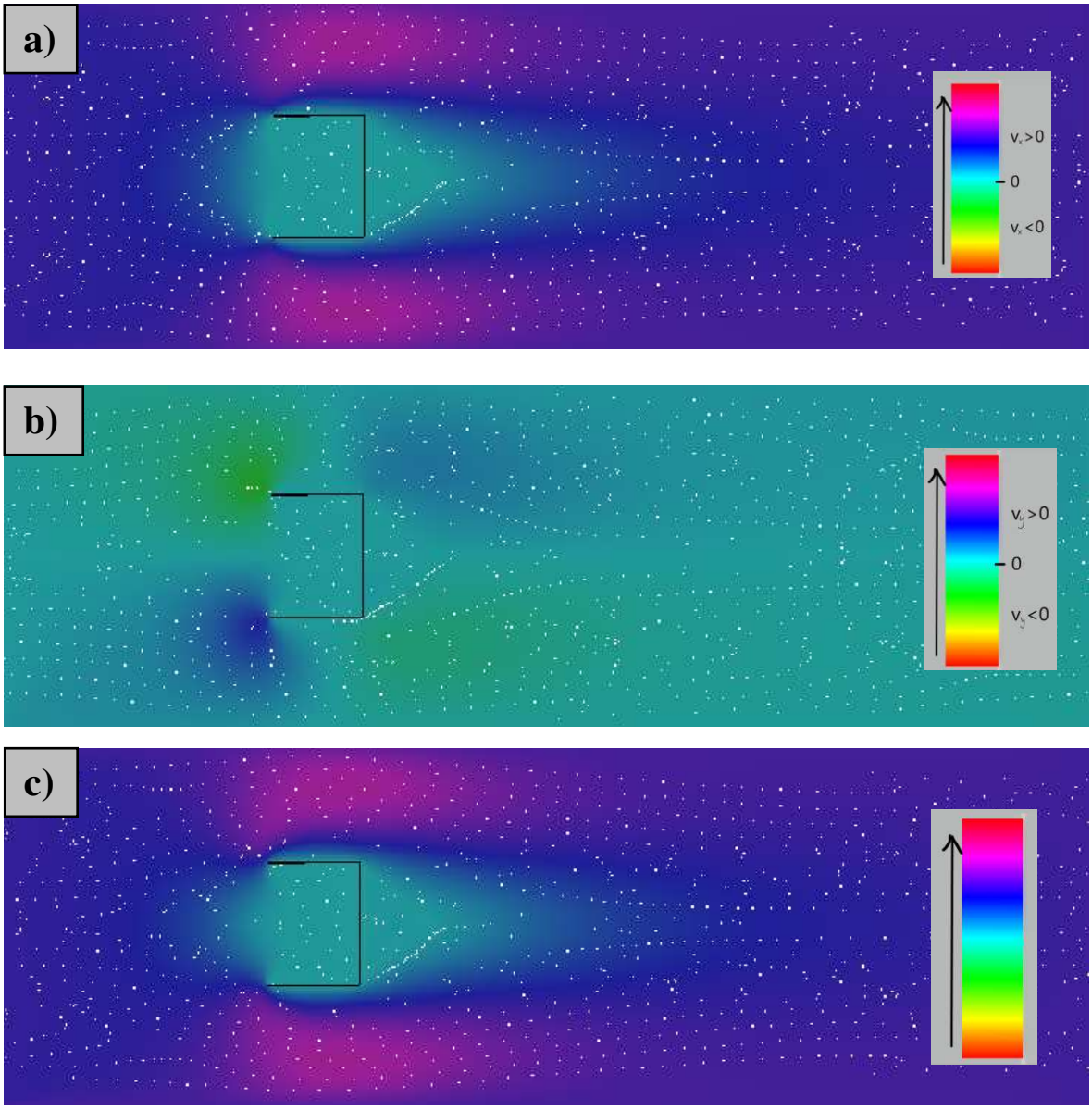
5.1.1 Veća viskoznost

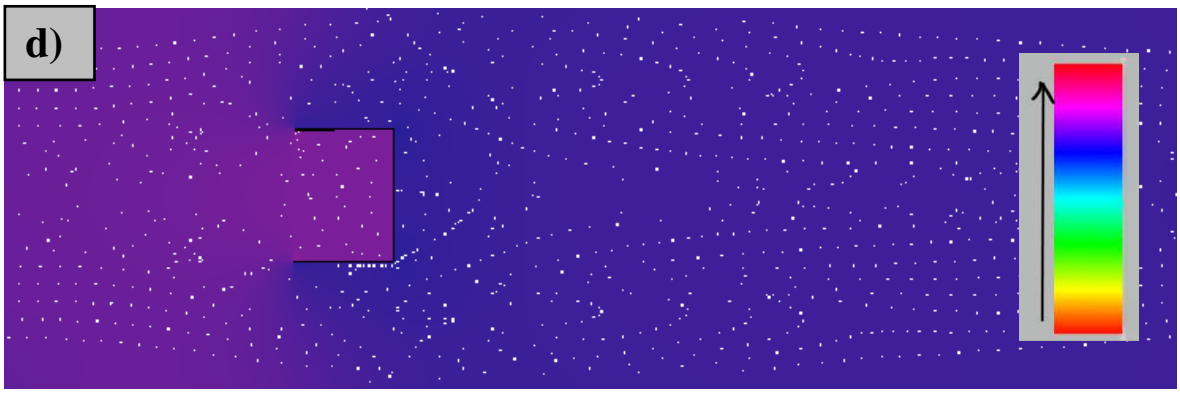
Varijabla `omega` postavljena je na 0.5 te je prikaz takvog fluida na slici (Sl. 5.1). Primjećujemo da su vrijednosti v_x i v_y umjerene, a posljedično tome i brzina (Sl. 5.1 c).

Vrijednost v_x je malo veća od prosjeka neposredno ispod i iznad prepreke gdje fluid ubrzava, a unutar i uz vanjski rub prepreke pada na nulu (Sl. 5.1 a). Vrijednost v_y skoro je svugdje nula osim na izbočinama prepreke gdje mijenja smjer kako bi zaobišla prepreku (Sl. 5.1 b).

Gustoća je najveća unutar prepreke, gdje se fluid nakuplja (Sl. 5.1 d).

U desnom kutu slika prikazane su interpolacije boja u smjeru rasta pripadnih vrijednosti.



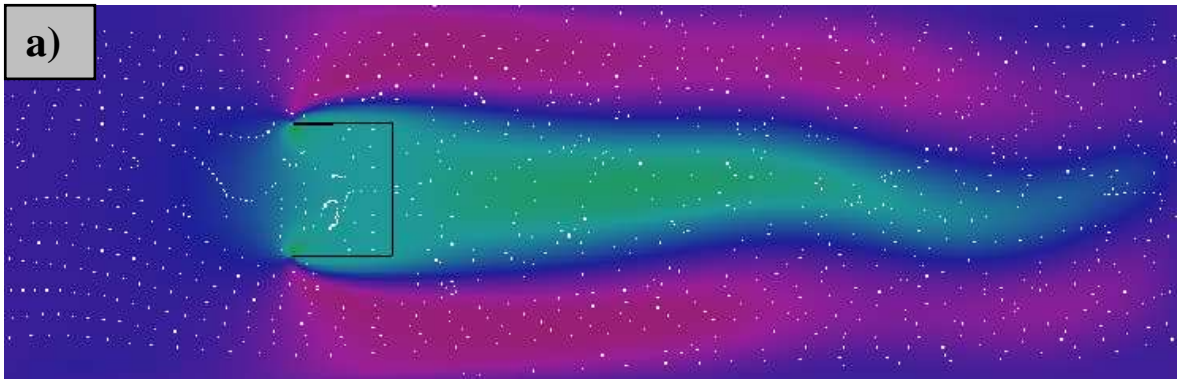


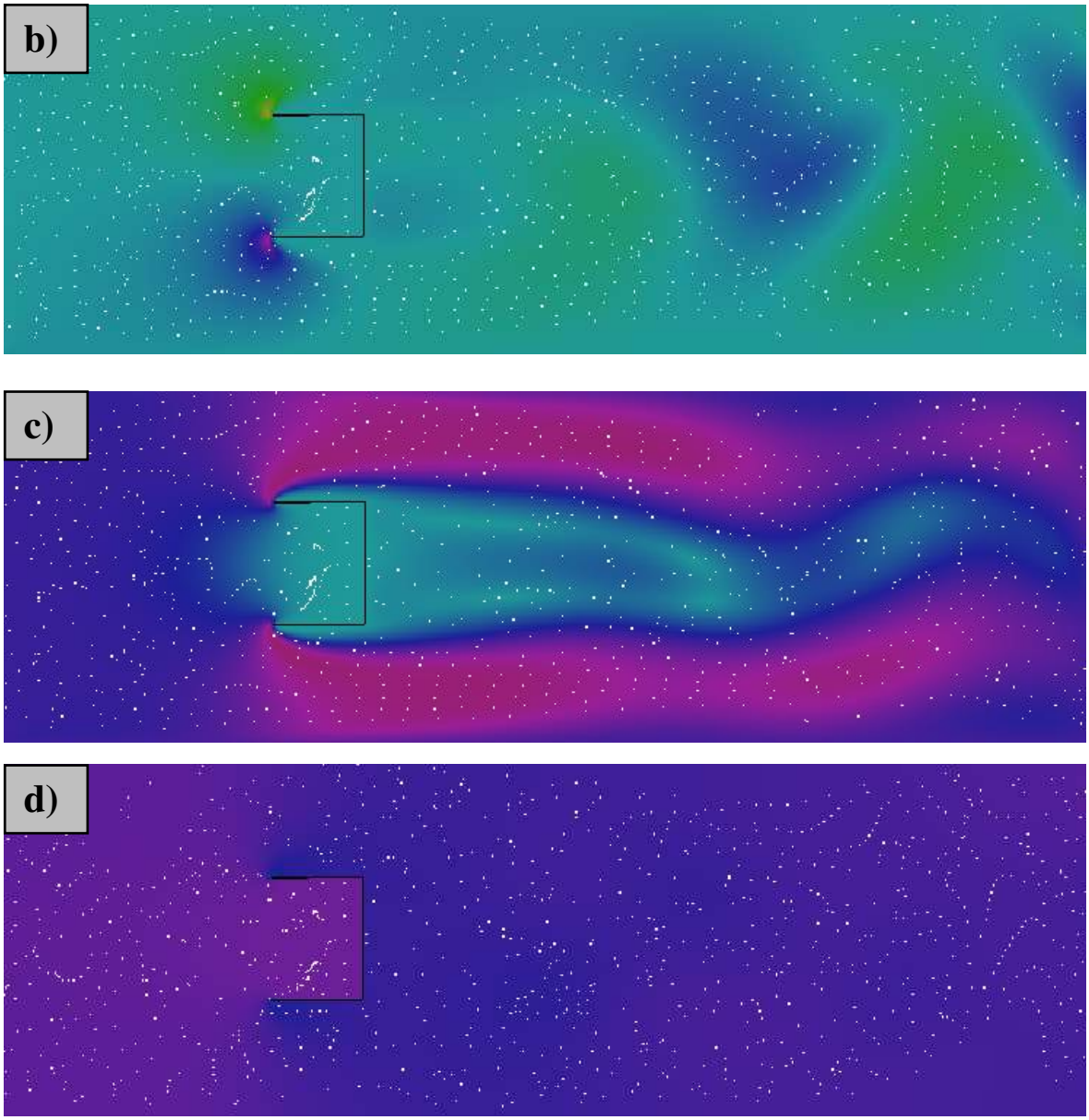
Sl. 5.1 Prikaz vrijednosti v_x (a), v_y (b), brzine (c) i gustoće (d) za $\omega = 0.5$

5.1.2 Manja viskoznost

Varijabla `omega` postavljena je na 1.5. Prikaz takvog fluida vidljiv je na slici (Sl. 5.2).

Može se uočiti značajna razlika naspram prethodnog primjera (Poglavlje 5.1.1). Utjecaj prepreke na brzinu raste smanjenjem viskoznosti (Sl. 5.2 c). Također dolazi do uvijanja v_y komponente te primjećujemo titranje gore-dolje iza prepreke (Sl. 5.2 b). Dio fluida se okrene u suprotnu stranu – zelena područja pokazuju da je vrijednost v_x komponente negativa iza prepreke (Sl. 5.2 a). Možemo primijetiti pojavu zelene boje i na rubovima prepreke s unutarnje strane (Sl. 5.2 a), što znači da dio fluida koji je unutar prepreke teče van. To kao posljedicu ima manju gustoću fluida unutar prepreke (Sl. 5.2 d) nego u prethodnom primjeru (Poglavlje 5.1.1).



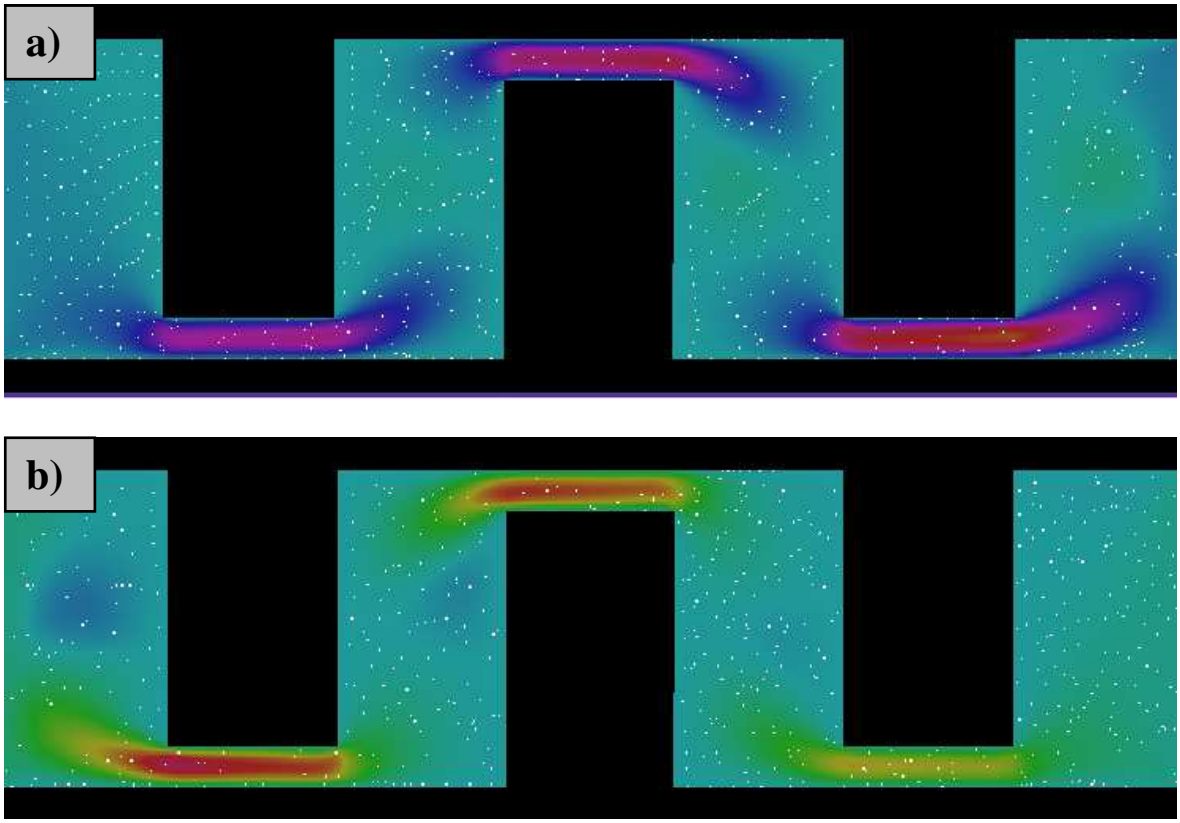


Sl. 5.2 Prikaz vrijednosti v_x (a), v_y (b), brzine (c) i gustoće (d) za ω=1.5

5.2 Utjecaj smjera v_x komponente

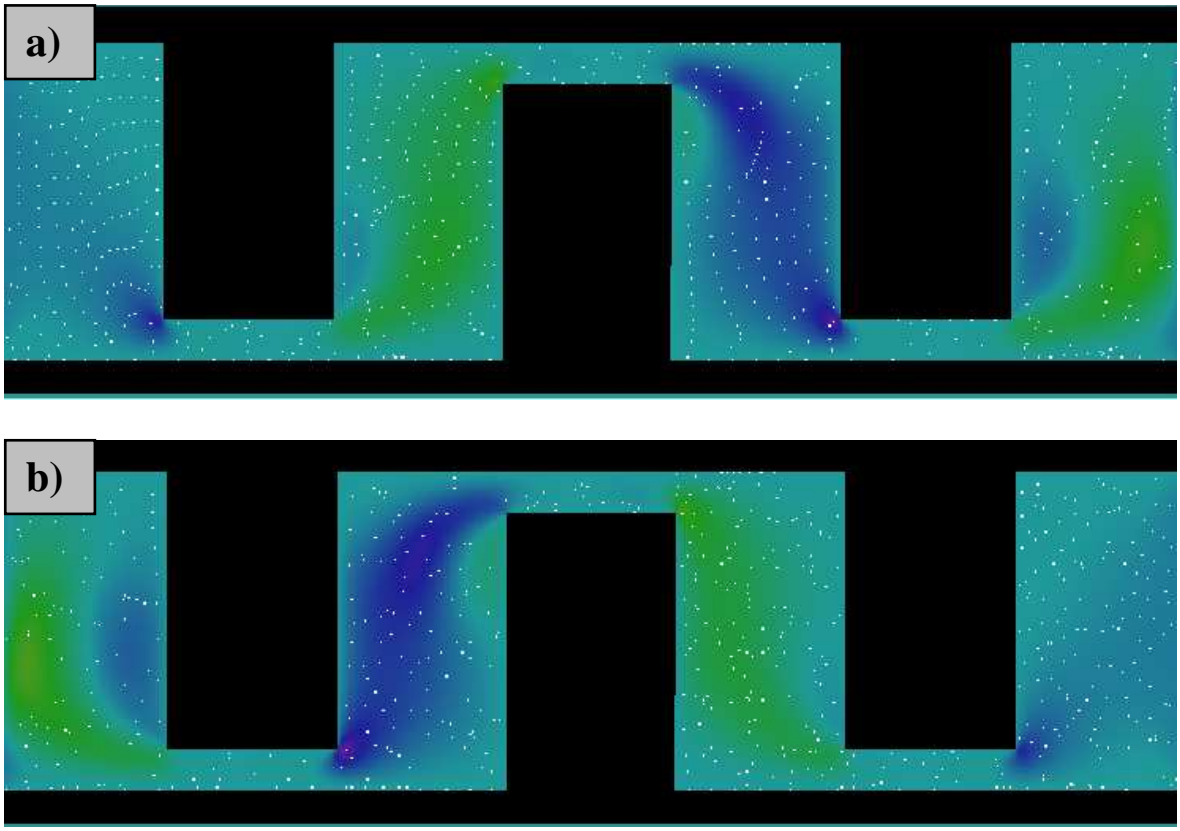
Varijabla `switch_x` postavljena je na `true` te se stoga varijabli `vx_init` periodično mijenja smjer (predznak). Varijabla `vx_init` je inicijalno postavljena na 0.13, a varijabla `vy_init` na 0.02. Apsolutna vrijednost varijable `vx_init` ostaje ista tijekom cijele simulacije (samo joj se predznak mijenja). Prepreka je umetnuta u rešetku metodom `make_obstacle_bowl`.

Apsolutna vrijednost v_x komponente je u oba slučaja (za pozitivnu i negativnu vrijednost varijable) najveća u uskim područjima (Sl. 5.3). Zanimljivo je primijetiti da dio fluida između susjednih uskih prolaza ima v_x vrijednost različitog predznaka od izvora, tj. dolazi do zakretanja – zelena područja za $v_x > 0$ (Sl. 5.3 a), odnosno plava područja za $v_x < 0$ (Sl. 5.3 b).



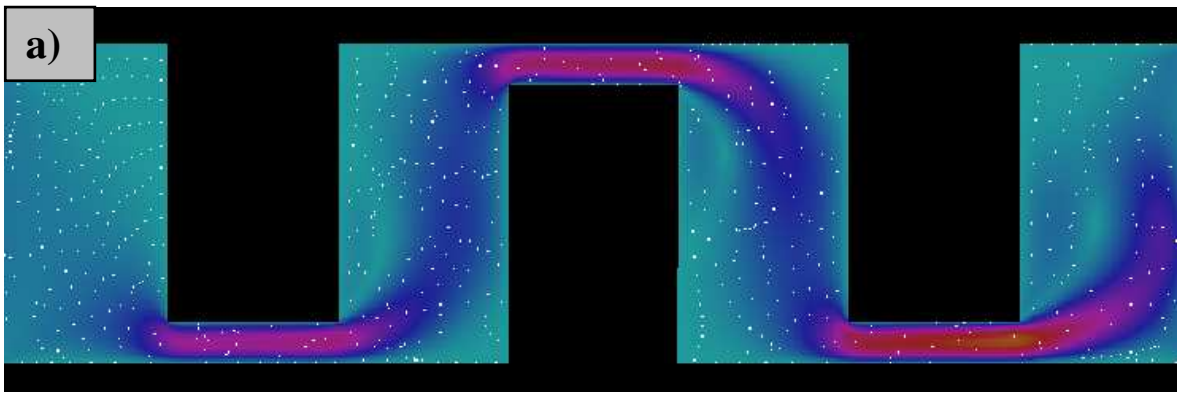
Sl. 5.3 Prikaz vrijednosti v_x za $v_x > 0$ (a) i $v_x < 0$ (b)

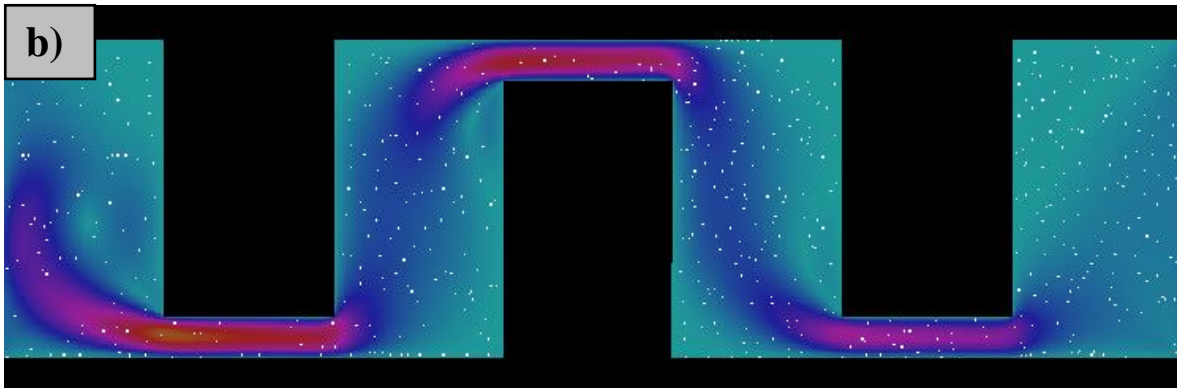
Kod v_y komponente se maksimum apsolutne vrijednosti dostiže u širim područjima, a minimum u užim (Sl. 5.4).



Sl. 5.4 Prikaz vrijednosti v_y za $v_x > 0$ (a) i $v_x < 0$ (b)

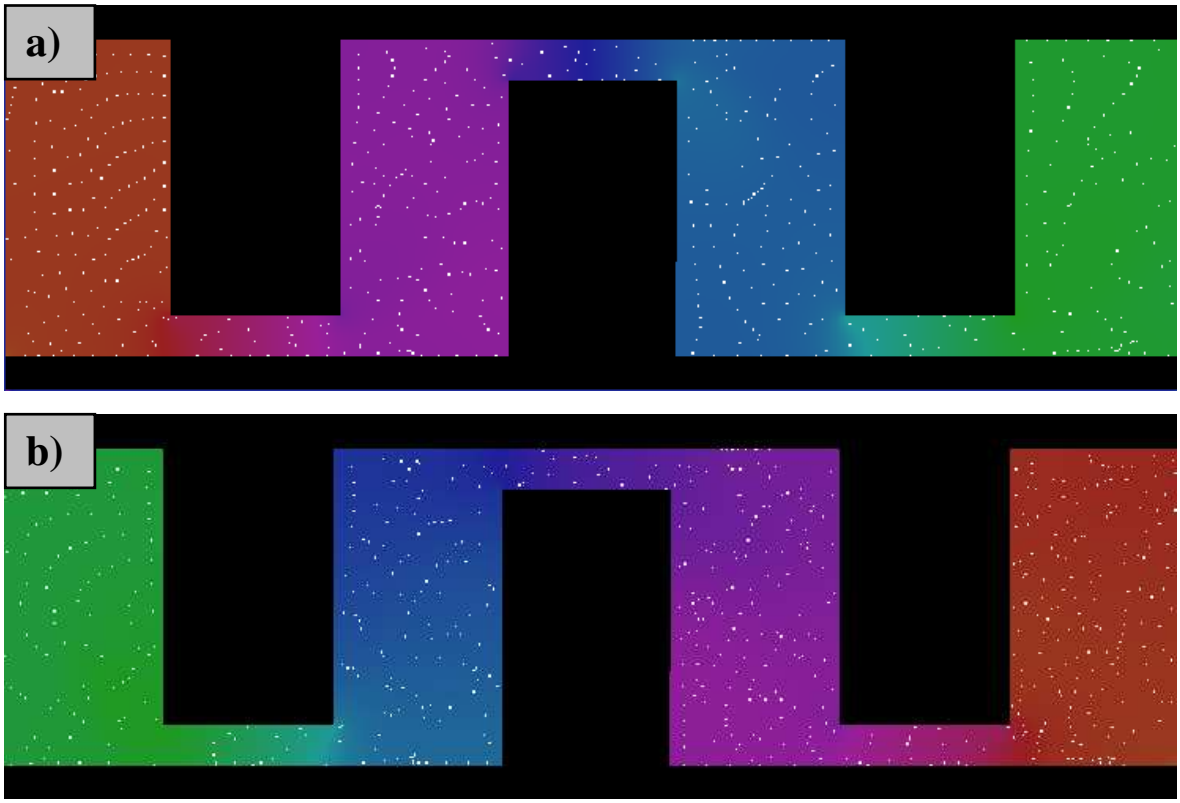
Iznos brzine je u oba slučaja skoro identičan (Sl. 5.5) jer pri izračunu brzine gledamo kvadrat komponente v_x koji je jednak u oba slučaja.





Sl. 5.5 Prikaz vrijednosti brzine za $v_x > 0$ (a) i $v_x < 0$ (b)

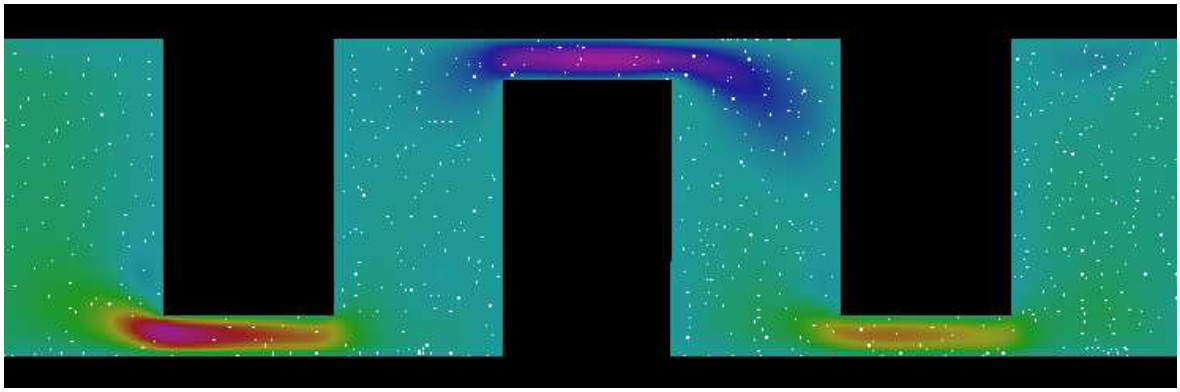
Za $v_x > 0$ se gustoća jednoliko smanjuje slijeva nadesno, dok za $v_x < 0$ jednoliko raste slijeva nadesno (Sl. 5.6). Slike su zrcalne te je gustoća u oba slučaja najveća na izvoru.



Sl. 5.6 Prikaz vrijednosti gustoće za $v_x > 0$ (a) i $v_x < 0$ (b)

Može se zaključiti da zrcaljenjem vrijednosti v_x komponente također dolazi do zrcaljenja dobivenog fluida.

Trenutak promjene smjera v_x iz pozitivnog u negativni, prije uravnoteženja fluida, prikazan je na slici (Sl. 5.7).



Sl. 5.7 Prikaz vrijednosti v_x neposredno nakon promjene smjera v_x iz pozitivnog u negativni

5.3 Podešavanje uloge rubova

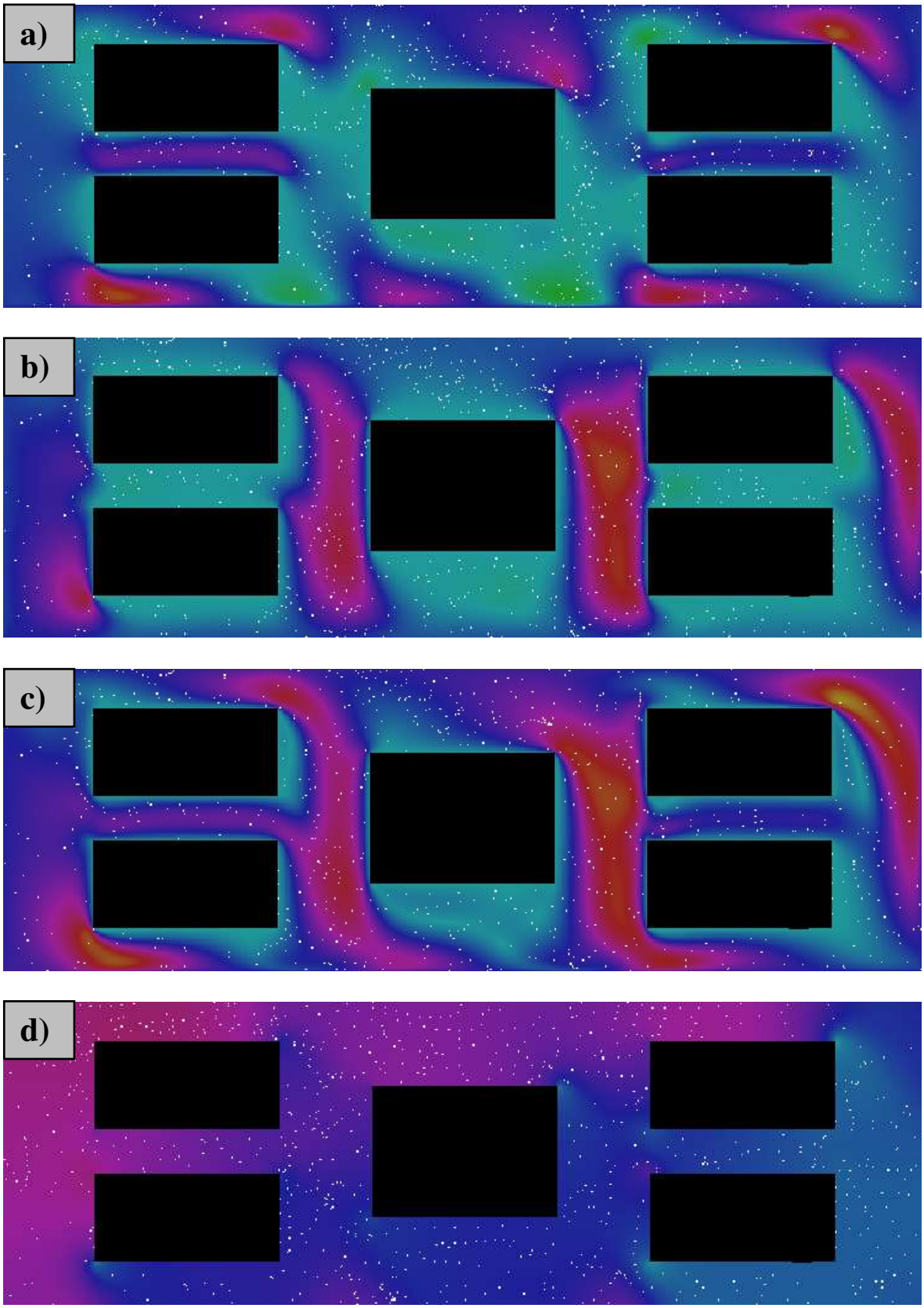
Promatran je utjecaj varijable `horizontal_borders_y` na ponašanje fluida.

Varijabla `vx_init` je postavljena na 0.08, a varijabla `vy_init` na 0.06. Prepreka je umetnuta u rešetku metodom `make_obstacle_rectangles`.

5.3.1 Opcija uključena

Varijabla `horizontal_borders_y` postavljena na `true`, odnosno svi rubovi rešetke su izvori sa stalnom brzinom (v_x, v_y). U ovom slučaju, za pozitivne `vx_init` i `vy_init`, fluid teče iz lijevog i gornjeg ruba prema donjem i desnom rubu.

Prikaz takvog fluida je na slici (Sl. 5.8).

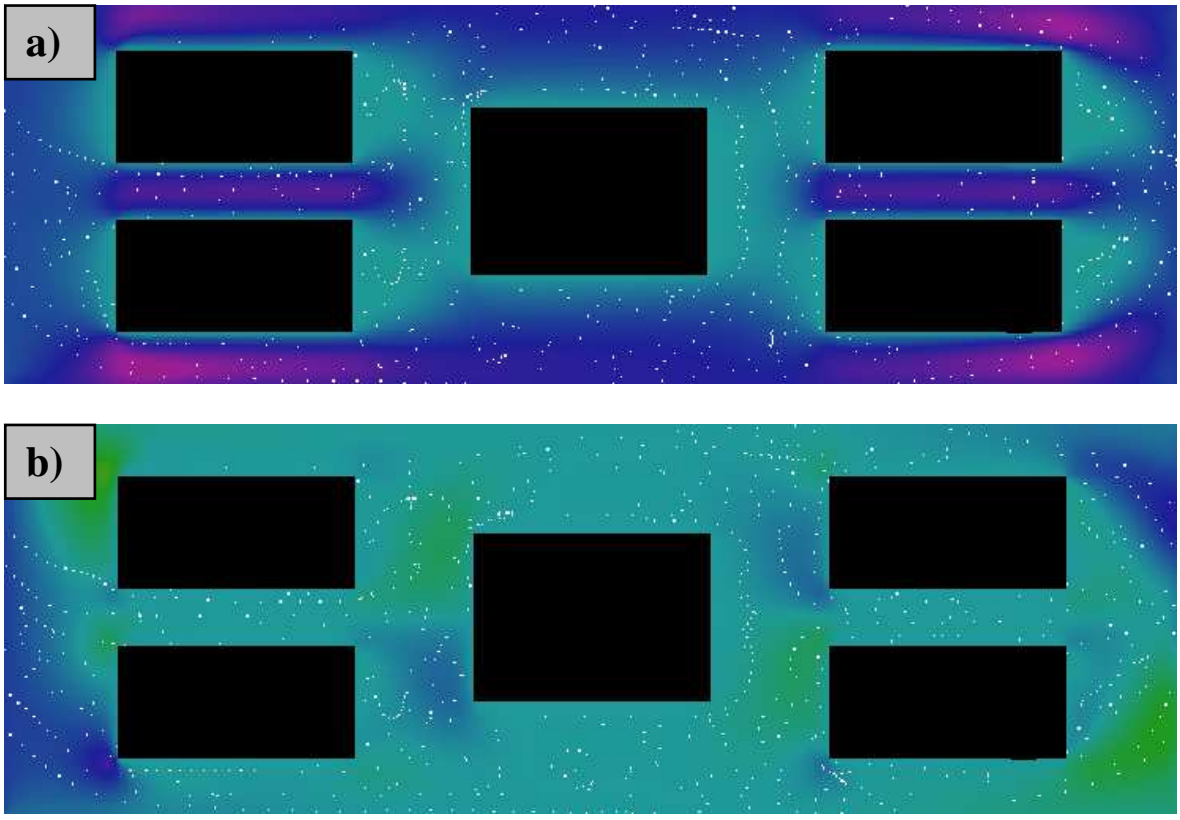


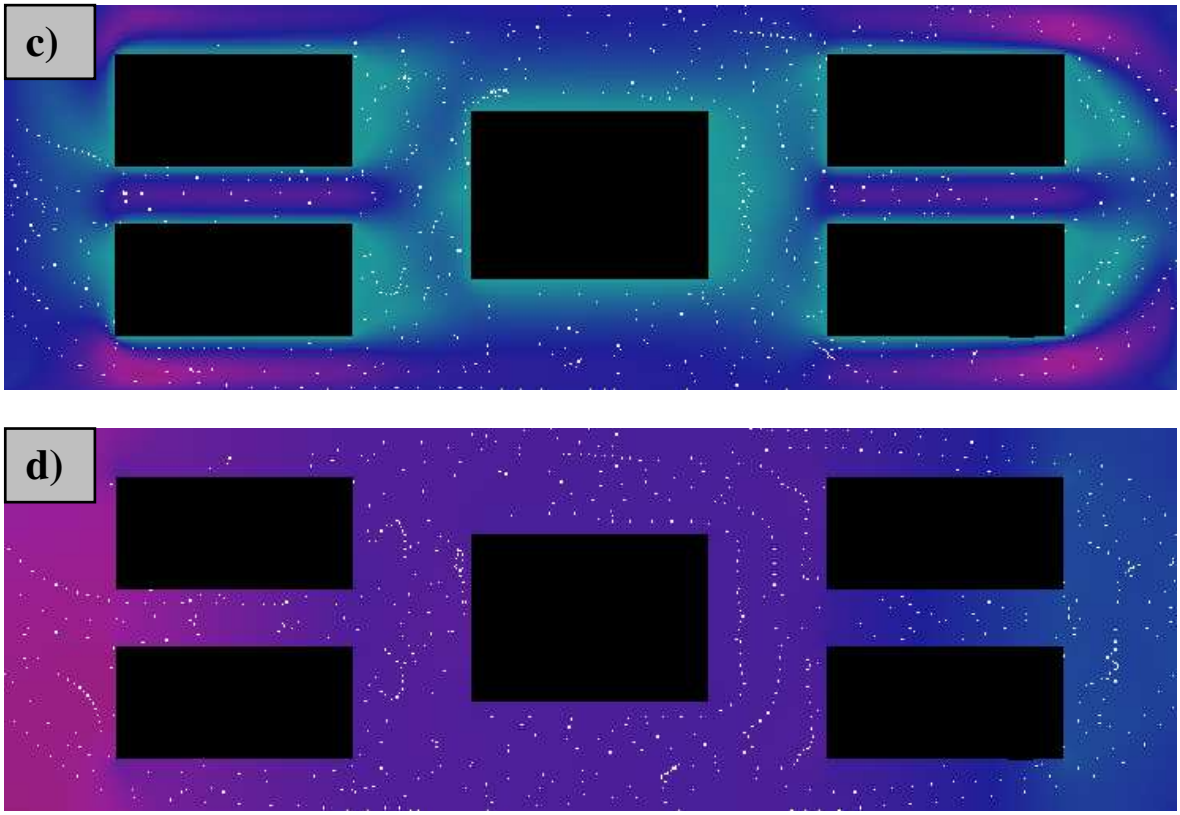
Sl. 5.8 Prikaz vrijednosti v_x (a), v_y (b), brzine (c) i gustoće (d) za `horizontal_borders_y = true`

5.3.2 Opcija isključena

Varijabla `horizontal_borders_y` postavljena na `false`. To znači da su lijevi i desni rub rešetke izvori sa stalnom brzinom (v_x, v_y), a gornji i donji rub sa stalnom brzinom (v_x, 0). Ideja ovdje je da je cijela rešetka tunel – fluid teče slijeva nadesno (ili obrnuto), a gornji i donji rub vode fluid u zadanom smjeru po x osi (fluid ne može izaći van rešetke preko gornjeg i donjeg ruba kao u prošlom primjeru, već ga ti rubovi guraju natrag u sredinu rešetke).

Prikaz takvog fluida je na slici (Sl. 5.9).





Sl. 5.9 Prikaz vrijednosti v_x (a), v_y (b), brzine (c) i gustoće (d) za `horizontal_borders_y = false`

5.4 Rušenje simulacije

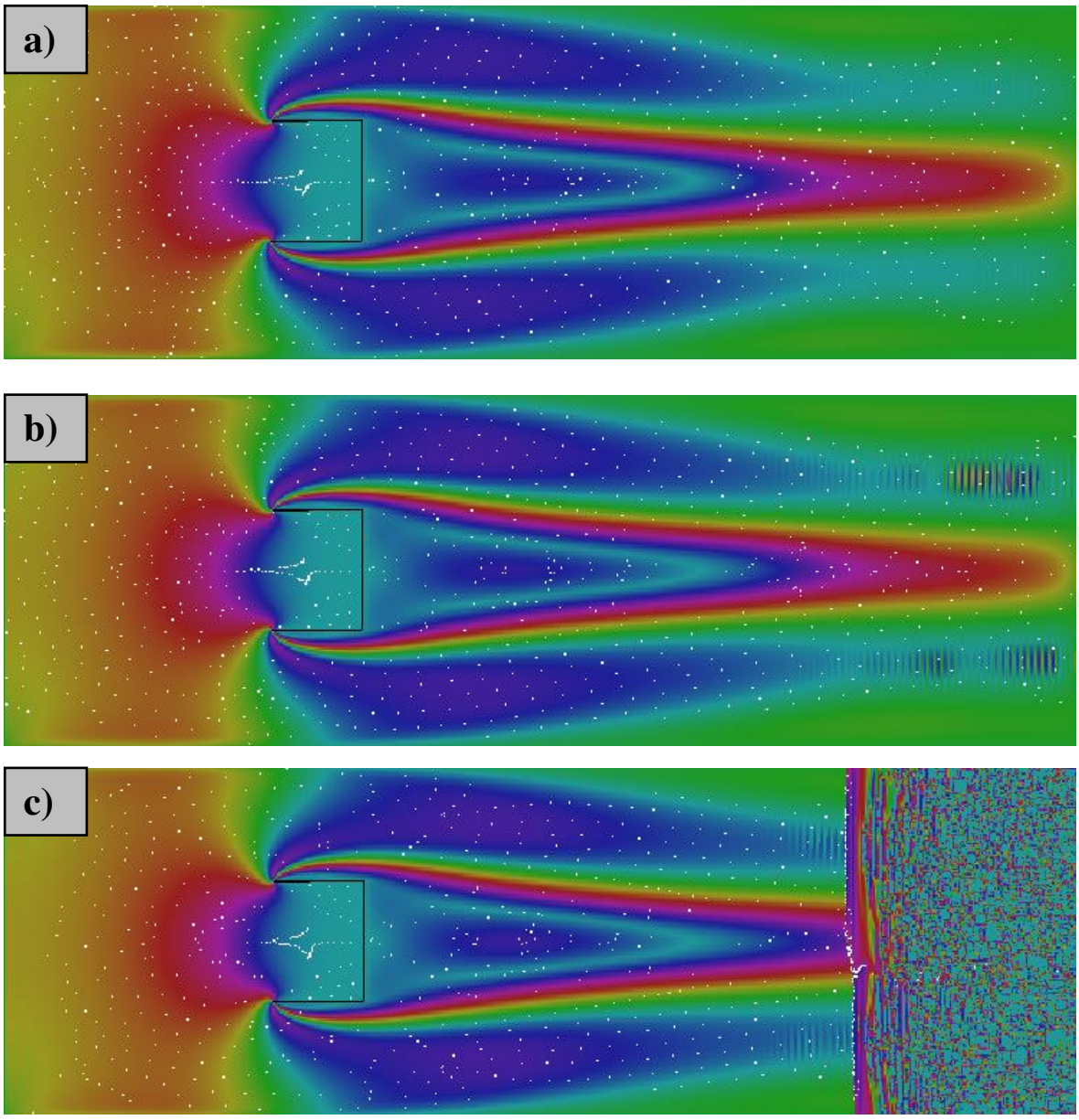
Simulacija pod određenim uvjetima ima neočekivana ponašanja. Dva slučaja kod kojih dolazi do rušenja simulacije prikazana su u nastavku.

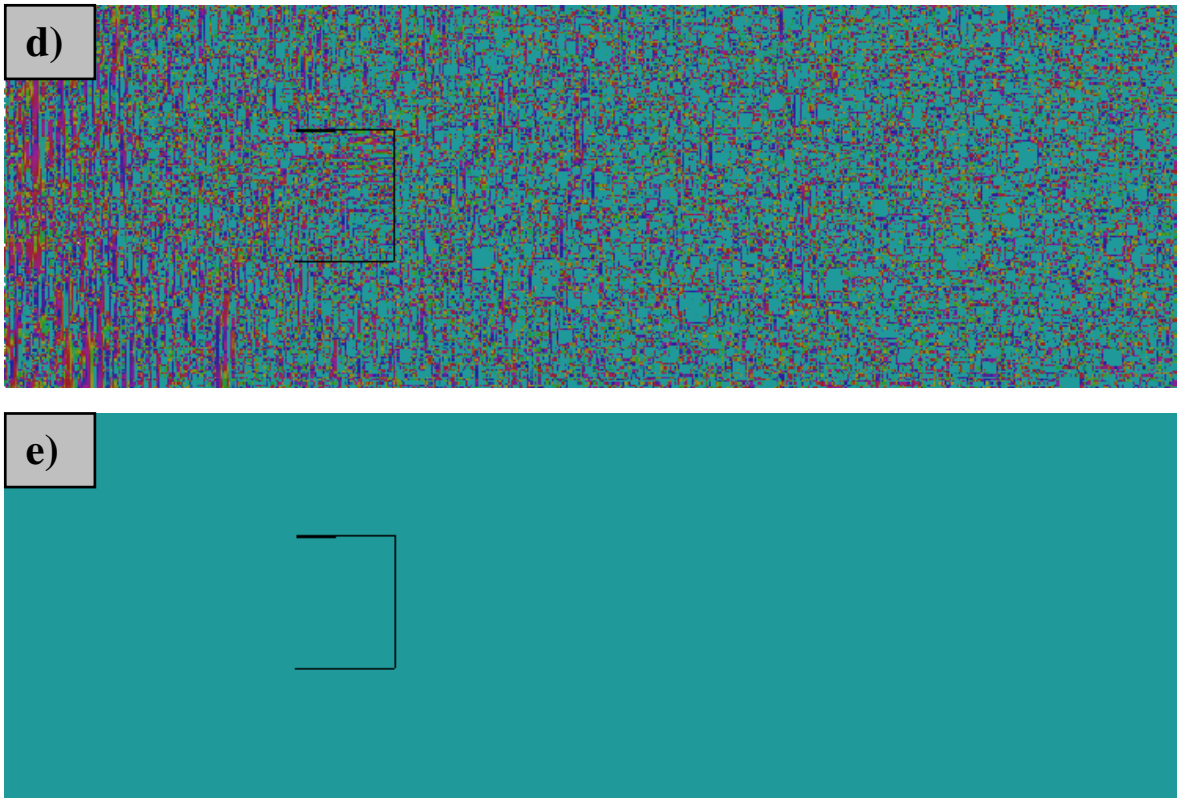
5.4.1 Prevelika brzina

Kod jednostavnijih prepreka simulacija ima bolju toleranciju na veće brzine. Uz prepreku `make_obstacle_bowl` i uz `vy_init` = 0, simulacija postaje nestabilna za vrijednosti `vx_init` = 0.4 i naviše (za usporedbu, lokalna brzina zvuka *c* fiksirana je na 1).

Za kompliciranije prepreke se simulacija ruši za puno manje brzine od ovih (tj. za manji Machov broj) jer fluid više ubrzava zbog većeg broja odbijanja od prepreka te time lakše dostiže prevelike brzine u užim prolazima.

Prikaz postupnog rušenja simulacije prikazan je na slici (Sl. 5.10). Pri ovim brzinama boje gube svoje izvorno značenje – brzina izlazi van očekivanih granica te time remeti očekivani prikaz.



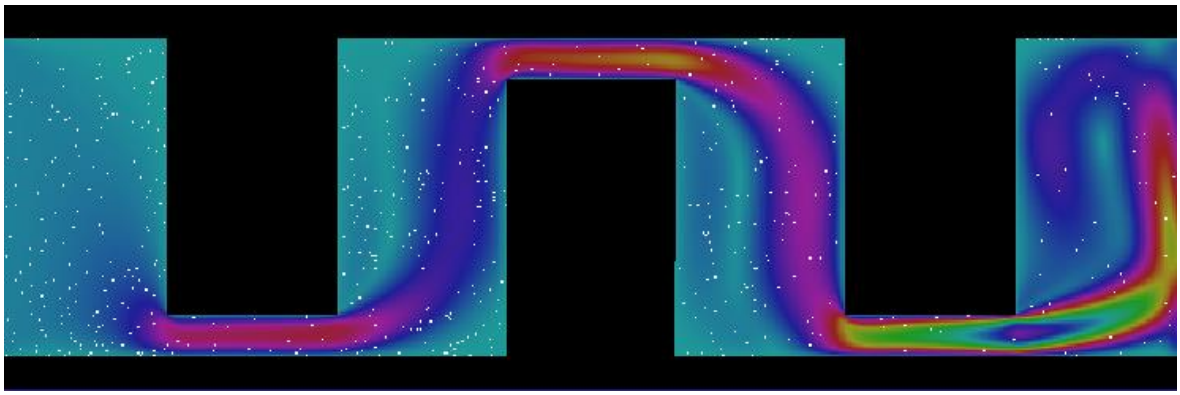


Sl. 5.10 Prikaz vrijednosti brzine neposredno prije (a), tokom (b, c, d) te nakon (e) rušenja simulacije

5.4.2 Inicijalizacija fluida

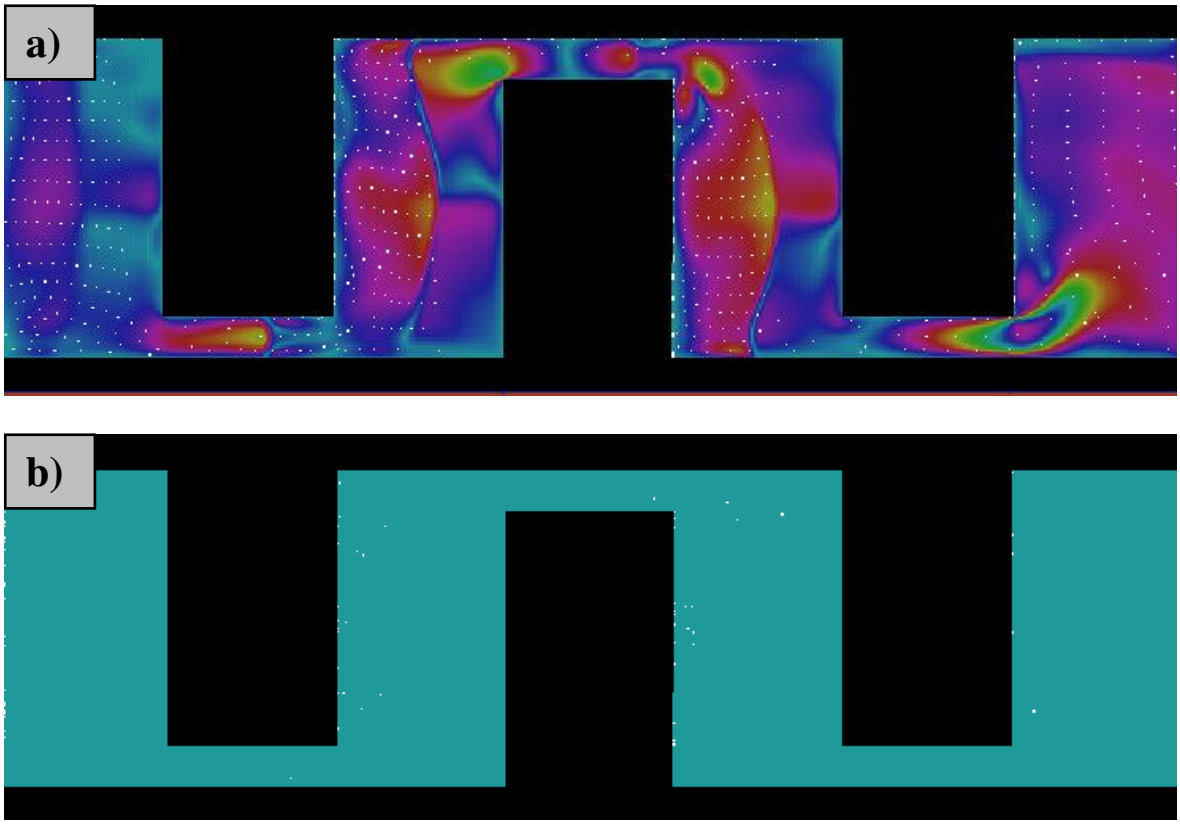
Inicijalizacija statičkog fluida metodom `initFluidZero`, koja je dosad korištena, ima veću toleranciju od inicijalizacije dinamičkog fluida metodom `initFluid`.

Za `vx_init` = 0.25, `vy_init` = 0, prepreku `make_obstacle_tunnel` i inicijalizaciju metodom `initFluidZero` simulacija se ne ruši, ali brzina na određenim mjestima dostiže previsoke vrijednosti te time boje gube predefinirano značenje (Sl. 5.11).



Sl. 5.11 Prikaz vrijednosti brzine za inicijalizaciju fluida metodom `initFluidZero`

Za iste vrijednosti varijabli i uz zamjenu `initFluidZero` s `initFluid`, simulacija se ruši (Sl. 5.12).



Sl. 5.12 Prikaz vrijednosti brzine neposredno prije (a) i nakon (b) rušenja simulacije za inicijalizaciju fluida metodom `initFluid`

Zaključak

Cilj rada bio je ukazati na prednosti i nedostatke rešetkastih Boltzmannovih metoda simulacije fluida naspram klasičnih metoda simulacije fluida temeljenih na rješavanju Navier-Stokes jednadžbi.

Nakon izlaganja osnova Navier-Stokes jednadžbi, pobliže je opisan algoritam i konkretna implementacija dinamičke simulacije fluida rešetkastom Boltzmannovom metodom. Implementacija je testirana uz različite početne parametre.

Testiranje je pokazalo da je ovaj pristup pogodan za male Machove brojeve, dok za veće dolazi do neočekivanog ponašanja i rušenja simulacije.

Najveća prednost korištenja LBM nad drugim metodama simulacije fluida je prilagođenost LBM paralelnom izvođenju. U ovoj implementaciji se koristi samo jedna dretva za izračun svih vrijednosti. Zbog toga simulacija postaje prespora za dimenzije rešetke većih od korištenih 200 x 600. Uvođenjem višedretvenog načina rada bi se brzina izvođenja uvelike popravila.

Literatura

[1] Stam, J. *Real Time Fluid Dynamics for Games*, 2003.

[2] *Navier-Stokes Equations*, (2015, siječanj). Poveznica: https://www.comsol.com/multiphysics/navier-stokes-equations; pristupljeno 7. lipnja 2022.

[3] Tschirschnitz, G., Sabrowski, P. *Computational Fluid Dynamics Methods Explained,* (2020, studeni). Poveznica: https://www.dive-solutions.de/articles/cfd-methods#LBM; pristupljeno 7. lipnja 2022.

[4] *Lattice-Boltzmann Method,* (2008). Poveznica: https://www.sciencedirect.com/topics/materials-science/lattice-boltzmann-method; pristupljeno 8. lipnja 2022.

[5] Fitzgerald, Barry & Zarghami, Ahad & Mahajan, Vinay & K. P. Sanjeevi, Sathish & Mema, Ivan & Verma, Vikrant & El Hasadi, Yousef & Padding, Johan. (2019). *Multiscale Simulation of Elongated Particles in Fluidised Beds*. Chemical Engineering Science: X. 2. 100019. 10.1016/j.cesx.2019.100019.

[6] Sauro Succi et al. (2010) *Lattice Boltzmann Method*. Scholarpedia, 5(5):9507. Poveznica: http://www.scholarpedia.org/article/Lattice_Boltzmann_Method#The_Lattice_Boltzmann_equation; pristupljeno 7. lipnja 2022.

[7] Neumann, Philipp & Bungartz, Hans-Joachim & Mehl, Miriam & Neckel, Tobias & Weinzierl, Tobias. (2012). *A Coupled Approach for Fluid Dynamic Problems Using the PDE Framework Peano*. Communications in Computational Physics. 12. 10.4208/cicp.210910.200611a.

[8] Martin Barrios, Raidel & Martínez-Mesa, Aliezer & Uranga-Piña, Llinersy. (2018). *Lattice-Boltzmann study of the wind-driven dynamics of shallow sea water*. Revista Cubana de Fisica. 35. E4-E7.

[9] Bülling, A. (2012). *Modelling of electrokinetic flow using the lattice-Boltzmann method.*

[10] *Lattice-Boltzmann Fluid Dynamics*, Physics 3300, Weber State University, Spring Semester, 2012

[11] Schroeder, D. *Fluid Dynamics Simulation*, Weber State University. Poveznica: https://physics.weber.edu/schroeder/fluids/; pristupljeno 24.5.2022.

Simulacija fluida metodom LBM

Sažetak

U ovom završnom radu obrađena je rešetkasta Boltzmannova metoda (LBM). Opisane su Navier-Stokes jednadžbe te osnovna podjela metoda korištenih pri simuliranju fluida. Opisana je razlika klasičnih metoda simuliranja fluida, temeljenih na Navier-Stokes jednadžbama, i rešetkastih Boltzmannovih metoda. Detaljnije je opisana metoda LBM. Implementirana je simulacija dinamičkog ponašanja fluida u dvodimenzionalnom prostoru koristeći LBM metodu. Implementacija je testirana s različitim početnim parametrima te su dobiveni rezultati uspoređivani i komentirani.

Ključne riječi: simulacija fluida, rešetkasta Boltzmannova metoda (LBM), Navier-Stokes jednadžbe (NSE)

Fluid Simulation Based on LBM Method

Summary

In this final paper, the lattice Boltzmann method (LBM) is explained. The Navier-Stokes equations and the basic division of the methods used in fluid simulations are described. The difference between traditional fluid simulation methods, based on the Navier-Stokes equations, and lattice Boltzmann methods is described. The LBM method is described in more detail. A simulation of the dynamic behavior of a fluid in a two-dimensional space is implemented using the LBM method. The implementation is tested with different initial parameters and the obtained results are compared and reviewed.

Keywords: fluid simulation, Lattice Boltzmann method (LBM), Navier-Stokes equations (NSE)

Privitak

Link na git repozitorij s izvornim kodom implementacije:
https://gitlab.com/KimStanicic/lbm-implementation