SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1623

VIZUALIZACIJA SIMULACIJE DINAMIKE PLINOVITIH FLUIDA

Kristina Štargel

Zagreb, listopad 2006.

SADRŽAJ:

1. UVOD

Modeliranje prirodnih fenomena (kao što su dim, vatra, oblaci itd.) je jedan od najzahtjevnijih i najpopularnijih izazova u računalnoj grafici. Prirodni fenomeni su popularni zbog svojeg zanimljivog izgleda i unatoč tome što nas svakodnevno okružuju i dalje privlače pozornost i divljenje. Zahtjevnost njihovog modeliranja proizlazi iz njihove kompleksnosti i složene dinamike koju je teško matematički opisati. Ovaj rad se fokusira na jedan od tih prirodnih fenomena a to je dim, ili općenitije na plinovite fluide.

Želja ili potreba za modeliranjem plinovitih fluida na računalu svodi se na dva osnovna područja primjene: zabava i istraživanja. S područjem zabave misli se na specijalne efekte u filmovima i interaktivnim igrama (Slika 1.1). Danas je teško zamisliti film koji ne koristi barem jedan simulator fluida. Danas korišteni simulatori su došli do tolike razine realnosti da je običnom laiku teško primijetiti da se radi o računalnoj simulaciji. Samo desetak godina unazad se bilježi velik napredak područjima animacije fluida, što je vidljivo po filmovima iz tog područja. Isto tako se bilježi procvat animiranog filma, koji se u današnje vrijeme potpuno svodi na računalnu animaciju.

Postoji i druga "korisna" primjena simulatora fluida. U industriji su potrebne razne simulacije strojeva koji su u interakciji s fluidom. Za primjer se može navesti industrija automobila, gdje je naravno potrebno simulirati rad motora. U to ulaze razni fluidi, od vode, plinova, do goriva. Svaki od tih simulatora mora biti izrazito točan.

U idealnom slučaju, dobar model dima na računalu trebao bi biti jednostavan za upotrebu i davati visoko realistične prikaze u stvarnom vremenu. Međutim, takve zahtjeve je teško postići današnjim računalnim resursima. Najčešće je potrebno raditi kompromise između točnosti i brzine simulacije. Što je za korišteni simulator važnije proizlazi iz područja primjene. Ako je potreban visoko realistični prikaz kao na primjer u filmskoj industriji, koristiti će se sporiji simulatori. Ako se radi o interaktivnim igrama koje se moraju odvijati u stvarnom vremenu (engl. *real-time*) , točnost prikaza više neće biti bitna. Bitno će biti samo da je simulator što brži tj. da se vrši što manje izračuna po jedinici vremena. Zadnji primjer je simulacija plinova u industriji. U tom slučaju najvažnija je točnost. Međutim bitno je i da taj proces ne traje predugo te će za to područje biti potrebna što bolja verzija simulatora.

Zbog sve većeg zanimanja za područje računalne simulacije fluida postoji područje koje se bavi samo tim problemima - računalna dinamika fluida (engl. *computational fluid dynamics* - CFD). To novo područje posvećeno je isključivo razvijanjem numeričkih metoda

za računanje dinamike fluida pomoću računala. Razvijanje tog područja je tek počelo. Zasad su postignuti razni modeli, međutim mnogi modeli pate ili od presporih algoritama ili od prevelike numeričke disipacije što je izazov za daljnje razvijanje novih, poboljšanih modela.

U ovom radu opisana je dinamika plinovitih fluida zajedno s pozadinskim matematičkim modelima, te osnovne numeričke metode korištene pri rješavanju tih matematičkih modela. Poseban naglasak je na model temeljen na radovima Stama [2] i Stama, Fedkiwa i Jensena [1] koji predstavlja visoko realističan model plinovitih fluida koji je stabilan za svaki vremenski korak i može se odvijati u stvarnom vremenu. Za taj model postavljen je matematički model i priložena metoda rješavanja i implementacija algoritma u programskom jeziku C++ sa grafičkim standardom OpenGL.

a.


b.


c.


Slika 1.1. Primjena simulatora fluida na filmu Dan poslije sutra (a. i b.) i animiranom filmu Ledeno doba 2 (c.)

1.1. POVIJEST MODELIRANJA FLUIDA

Modeliranje dima i ostalih plinovitih fenomena poprima veliku pažnju u računalnoj grafici tijekom proteklih 20 godina. Raniji su modeli se fokusirali na pojedinačne fenomene i animaciju gustoće dima direktno bez modeliranja njegove brzine. [6, 8, 4, 9]. Dodatni detalji su pridodani koristeći teksture krutih tijela čije bi parametre mijenjali tijekom vremena. Naknadno su korištena nasumična polja brzina temeljena na Kolmogoroff-ovom spektru, za modeliranje kompleksnih gibanja karakterističnih za dim [10]. Zajednička činjenica za sve rane modele je bila da se nisu temeljili na dinamici fluida. Kreiranje uvjerljive simulacije dinamike fluida je vremenski prezahtjevan zadatak koji je bio ostavljen animatoru.

Mnogo prirodniji način za modeliranje kretanja dima je direktno simuliranje jednadžbi dinamike fluida. Kajiya i Von Herzen su prvi napravili taj korak u računalnoj grafici [7]. Nažalost, računalna snaga dostupna u to vrijeme (1984) je dopuštala simulaciju jedino na poprilično grubim mrežama. Izuzev nekoliko modela specificiranih u dvije dimenzije [13,5], nikakav napredak nije učinjen na tom području do rada Fostera i Metaxasa [6] 1996. godine. Njihova simulacija koristi relativno grube mreže, ali daju lijepe vrtložne prikaze gibanja dima u tri dimenzije. Zbog toga što njihov model koristi eksplicitnu integracijsku shemu, simulacija je stabilna samo ako je vremenski korak dovoljno malen. To čini simulaciju relativno sporom, pogotovo kada je brzina fluida velika bilo gdje u domeni interesa. Za izbjegavanje ovog problema Stam je uveo model koji je bezuvjetno stabilan i prema tome simulacija može biti izvedena pri bilo kojim brzinama [2]. To je postignuto koristeći kombinaciju semi-Lagrangian sheme za rješavanje problema horizontalnog strujanja u jednadžbama dinamike fluida i implicitnih solvera. Zbog korištenja integracijskih shema prvog reda, simulacija pati zbog prevelike numeričke disipacije. Iako sveukupno gibanje izgleda kao gibanje fluida, vrtlozi malih razmjera tipični za gibanje dima nestaju prebrzo.

Prije par godina, Yngve predlaže rješavanje stlačivih verzija jednadžbi toka fluida prema modelu eksplozija [14]. Stlačive inačice jednadžbi toka fluida su korisne za kreiranje potresnih valova i ostalih stlačivih fenomena, međutim takav model uvodi vrlo strogo ograničenje vremenskog koraka povezanog s akustičnim valovima. U većini CFD modela izbjegava se korištenje tih strogih uvjeta koristeći nestlačive inačice jednadžbi za tok fluida gdje god je to moguće. Iz tog razloga se ovaj rad ne temelji na stlačivim inačicama jednadžbi.

Spajanjem Stamove metode stabilnih fluida i uvođenjem novih metoda za izbjegavanje numeričke disipacije nastaje rad Fedkiwa, Stama i Jensena. Taj model koristi simulator temeljen na Stamovom radu koji koristi semi-Lagranigan integracijsku shemu te solver

Poissonove jednadžbe. To garantira stabilnost kroz bilo kakav vremenski korak. Za izbjegavanje numeričke disipacije koje taj model nosi sa sobom, koristi se metoda poznata kao "ograničavanje vrtložnosti" (engl. *voticity confinement*) [12].Glavna ideja metode je vraćanje energije izgubljene zbog numeričke disipacije natrag u fluid koristeći izraz za silu. Sila je dizajnirana specijalno za povećanje vrtložnosti toka. Vizualno ta metoda održava dim živim kroz vrijeme simulacije. Korišteni izraz za silu je potpuno konzistentan sa Eulerov-im jednadžbama (koje simulator koristi) u smislu da nestaje kad se broj mrežnih ćelija povećava. U CFD-u se navedena tehnika uvodi za numerički izračun kompleksnih turbulentnih tokova fluida okolo helikoptera gdje nije moguće dodati dovoljno mrežnih ćelija da bi polje toka bilo točno riješeno. Izračunavanje sile dodaje samo mali računski *overhead* na već navedenu Stam-ovu verziju simulatora bez metode ograničavanja vrtložnosti. Ovakav model ostaje stabilan dokle god je magnituda izraza za silu ispod određene granice.

Semi-Lagrangian sheme su vrlo popularne u znanstvenim ustanovama koje se bave atmosferskim promjenama (pojavama) za modeliranje tokova velikih razmjera kojima dominira konstantno strujanje gdje su poželjni veliki vremenski koraci [11]. Takav pristup je specijalno efektivan za modele gdje se gustoća i temperatura pomiču kroz polje brzine.

Uz sva navedena obilježja ovaj model može rješavati granične uvjete unutar računalne domene. Zbog toga je moguće simulirati dim kako obilazi oko raznih objekata.

1.2. FLUID OPĆENITO

Fluidi su dio stvarnosti koja nas okružuje, u formi vode koju pijemo, zraka kojeg udišemo, u formi jutarnje kave, dima, sve do velikih prirodnih fenomena kao što su rijeke, mora, oblaci itd. Svaki od tih fenomena pokazuju fascinantne oblike ponašanja. Svi smo izgradili određeno intuitivno razumijevanje ponašanja fluida, međutim, još uvijek postoje posebni oblici ponašanja koje većina ljudi ne može pojmiti, kao na primjer let aviona. Svi ljudi su svjesni toga da jedan veliki, teški objekt, kao što je avion, može letjeti, međutim samo mali dio ljudi razumije kako.

Poznavanje dinamike fluida je postao jedan od najvećih izazova fizike, primijenjene matematike i inženjerstva kroz zadnjih 100 godina. Dinamika fluida je postala ključ razumijevanja nekih od najvažnijih fenomena u fizikalnom svijetu: oceanskih struja i atmosferskih promjena, promjena na vanjskim slojevima Sunca, eksplozija supernova, vrtloženja plinova u galaksijama i mnogih drugih fenomena.

Kao i kod mnogih fascinantnih pojava, razumijevanje ponašanja fluida nije lako. Pobliže, u dinamici fluida postoje mnogi izrazi i matematičke metode koje su većini ljudi nepoznate. Fluidi su također fascinantni zbog toga jer je ponekad njihovo ponašanje nepredvidivo, što znači velik trud za njihovo matematičko predstavljanje.

Fluidi su supstance čije molekularne strukture ne pružaju otpor vanjskim pomičnim silama: čak i najmanja sila uzrokuje deformaciju čestica fluida. Krute stvari mijenjaju svoj oblik i deformiraju se dok se ne postigne ravnoteža između primijenjene vanjske sile i unutrašnjih sila (ili ako dođe do pucanja materijala). S druge strane, fluidi će se deformirati kontinuirano i mijenjati oblik pod djelovanjem sile. To je fundamentalna definicija fluida.

Iako postoje bitne razlike između tekućina i plinova, oba tipa fluida se ponašaju po istim zakonima gibanja. Tok fluida je uzrokovan djelovanjem vanjskih sila. Uobičajene pokretačke sile uključuju razlike u tlaku, gravitaciju, smicanje, rotaciju i napetost površine. One mogu biti klasificirane kao površinske sile (smicanje, tlak) i sile koje djeluju na cijelo tijelo (gravitacija, sile rotacije).

Dok se svi fluidi ponašaju slično pod djelovanjem sile, njihova makroskopska obilježja su uvelike različita. Najvažnija svojstva fluida su gustoća i viskoznost. Ostala svojstva kao Prandtl-ov broj (dodatak A.3), specifična toplina i površinska napetost utječu na tok fluida samo pod određenim uvjetima tj. kada dolazi do velikih temperaturnih razlika.

Obilježja fluida su funkcije ostalih termodinamičkih varijabli (npr. temperatura i tlaka) koje se dobivaju laboratorijskim mjerenjima.

Brzina toka fluida utječe na njegova svojstva na razne načine. Pri dovoljno malim brzinama, inercija fluida može biti ignorirana i tada imamo "puzajući tok". Kako se brzina povećava, iznos inercije postaje relevantan i putanja toka je glatka trajektorija za svaku česticu fluida. Za tok se tada kaže da je laminaran. Daljnje povećanje brzine može voditi do nestabilnosti, koja dovodi do sve više nepredvidivog oblika toka koji se naziva turbulentnim. Proces prelaženja fluida između ta dva oblika je posebno važno područje samo za sebe.

Postoji još jedan parametar koji utječe na ponašanje fluida. Omjer brzine toka i brzine zvuka u sredstvu (Mach broj-dodatak A.2) određuje da li izmjena kinetičke energije gibanja i unutarnji stupnjevi slobode dolaze do izražaja. Za male vrijednosti Mach broja, $Ma < 0.3$, tok se može smatrati nestlačivim, inače stlačivim. Ako je vrijednost $Ma < 1$, brzina toka fluida je ispod razine zvuka (engl. *subsonics*), ako je vrijednost $Ma > 1$, tada je brzina fluida iznad zvuka (engl. *supersonic*) i mogući su udarni valovi. I konačno kada je $Ma > 5$ tok može proizvesti dovoljno velike temperature da promjeni kemijsku prirodu fluida (engl. *hypersonic fluids*). Ove razlike utječu na matematičku prirodu problema pa tako i na metodu rješavanja problema.

Jedno od bitnih obilježja fluida je viskoznost. Viskoznost je pojava unutarnjeg trenja u fluidima. Najbolji način da se opiše to svojstvo je primjerom. Silu predstavlja uski kruti predmet (npr. nož) koji se pomiče kroz fluid u istoj ravnini. Za razliku od krutih tijela, fluid "propušta" predmet kroz sebe, ali uz određenu količinu otpora. Svaki fluid daje drugačiji iznos otpora. Taj otpor fluida nazivamo viskoznost. Za mnoge modele viskoznost može biti zanemarena, tj. pretpostavlja se da joj je vrijednost jednaka nuli. U tom slučaju fluid se opisuje kao ne viskozan. Pretpostavka o ne viskoznim fluidima u računalnoj dinamici fluida je česta. Treba naglasiti da su granični uvjeti različiti za viskozne i ne viskozne fluide (jer je utjecaj viskoznosti bitan samo u blizini prepreka).

Proučavanje dinamike fluida nije tako jednostavno jer je fluid raširen po prostoru, i kada se giba (npr. zbog pomicanja granica oko fluida), sile su povezane s unutrašnjosti fluida utjecajem fluida na samog sebe. Unatoč tome, temelj svih ponašanja fluida su empirički verificirani fizički zakoni: zakon o očuvanju mase, zakon o očuvanju energija i linearnog momenta, kombinirani za zakonima termodinamike.

2. MATEMATIČKI MODELI

S matematičkog gledišta, proučavanje dinamike fluida se svodi na izraze koji uključuju skalarna (masa, gustoća i tlak fluida) i vektorska polja (brzina toka fluida) definirana na Euklidskom prostoru $\Re^3$, i njihove međusobne veze. S obzirom da su veze među njima pretežno diferencijalne prirode, zadire se u polje matematike koje se naziva vektorski račun (engl. *vector calculus*). Za bolje razumijevanje jednadžbi toka fluida prvo će se razjasniti neke osnovne matematičke operacije i termini korišteni za izračun.

Prvo će se navesti vektorski analitički diferencijalni operatori u desno orijentiranom Kartezijskom koordinatnom sustavu sa bazom u $\Re^3$, (*i, j, k*) sa koordinatama (x, y, z), gdje vrijedi

$$1 = |i| = |j| = |k|. \tag{2.1}$$

Ako na domeni $D \subset \Re^3$ definiramo Φ kao diferencijabilnu skalarnu funkciju pozicije $\mathrm{r} = (x, y, z)^{\mathrm{T}}$, i $\mathbf{A}=(A_x, A_y, A_z)^{\mathrm{T}}$ kao diferencijablinu vektorsku funkciju od r, tada definiramo vektorsko analitički diferencijalni operator i njegovo djelovanje na skalarna i vektorska polja:

HAMILTONOV OPERATOR

$$\nabla = \frac{\partial}{\partial x} i + \frac{\partial}{\partial y} j + \frac{\partial}{\partial z} k \tag{2.2}$$

Hamiltonov operator se ponaša kao vektor i kao operator parcijalnog deriviranja u smjeru baznih vektora. Taj operator djeluje na funkcije koje se nalaze iza (desno) njega i to na skalarna polja, te na vektorska polja na dva načina.

GRADIJENT

$$\nabla \phi = \frac{\partial \phi}{\partial x} i + \frac{\partial \phi}{\partial y} j + \frac{\partial \phi}{\partial z} k \tag{2.3}$$

Gradijent je primjena Hamiltonovog operatora na skalar i kao rezultat se dobije vektor.

DIVERGENCIJA

$$\nabla \cdot \mathbf{A} = \frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z} \tag{2.4}$$

Divergencija je skalarni umnožak Hamiltonovog operatora i vektorskog polja i kao rezultat se dobije skalarna vrijednost.

ROTOR

$$\nabla \times \mathbf{A} = \left[\frac{\partial A_z}{\partial y} - \frac{\partial A_y}{\partial z} \right] i + \left[\frac{\partial A_x}{\partial z} - \frac{\partial A_z}{\partial x} \right] j + \left[\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y} \right] k \tag{2.5}$$

Rotor je vektorski umnožak Hamiltonovog operatora i vektorskog polja i njime se dobiva vektorska vrijednost.

LAPLACEOV OPERATOR

$$(\nabla \cdot \nabla)\phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \tag{2.6}$$

Laplaceov operator je u biti Hamiltonov operator skalarno pomnožen sa samim sobom.

Sada su opisani svi operatori koji se koriste u matematičkim modelima toka fluida i može se krenuti na postavljanje matematičkog modela.

2.1. POJEDNOSTAVLJENI MATEMATIČKI MODELI

Navier-Stokesove jednadžbe, nazvane po Claude-Luis Navieru i George Gabriel Stokesu, su sustav jednadžbi koji opisuju gibanje čestica fluida kao što su tekućine i plinovi. Te jednadžbe se temelje na pretpostavci da promjene u momentu (akceleraciji) čestica fluida su jednostavno rezultati promjena u tlaku i disipativnim viskoznim silama (unutrašnje trenje fluida) koje se odvijaju unutar fluida.

Jednadžbe očuvanja mase i momenta su mnogo kompleksnije nego se čine. Jednadžbe su nelinearne, međusobno povezane i teško rješive. Teško je dokazati postojećim matematičkim alatima da postoji jedinstveno rješenje za partikularne granične uvijete. Iskustvo govori da Navier-Stokesove jednadžbe opisuju tok Newtonovog fluida (tok fluida koji se ponaša po Newtonovim zakonima očuvanja mase i momenta) ispravno. Samo za mali broj slučajeva, pretežno potpuno razvijenih tokova u jednostavnim geometrijama (npr. u cijevima, između paralelnih prepreka), postoji jedinstveno analitičko rješenje Navier-Stokesovih jednadžbi. Ti specijalni slučajevi tokova su bitni za proučavanje osnova dinamike fluida, ali njihova praktična upotreba (relevantnost) je ograničena.

U svim slučajevima kada je rješenje moguće, mnogi izrazi u jednadžbi su jednaki nuli. Za ostale tokove neki od izraza su nebitni i možemo ih zanemariti i tako pojednostaviti navedene jednadžbe. Treba imati na umu da takvo pojednostavljenje donosi i pogrešku. Osim toga, pojednostavljenje donosi i mnogo manju cijenu izračuna nego što je slučaj za izračunavanje potpunih jednadžbi, te ta činjenica opravdava potrebu za pojednostavljenjem. U većini slučajeva, čak i pojednostavljene jednadžbe se ne mogu riješiti analitički, te se koriste numeričke metode.

Kada se govori o Navier-Stokesovim jednadžbama, obično se misli na spomenute pojednostavljene matematičke verzije koje se koriste za specijalne slučajeve ponašanja fluida. Tako i u ovom tekstu se neće ulaziti podrobnije u njihov izvod i analizu, već će se koristiti već pojednostavljenje verzije.

U nastavku će biti navedena dva bitna pojednostavljenja na kojima se temelje numeričke metode za izračun toka fluida: nestlačivi tok i neviskozni Eulerov tok.

2.1.1. NESTLAČIVI TOK

U mnogim slučajevima gustoća fluida se može pretpostaviti konstantnom. To je ispravan izraz ne samo za tokove tekućina, čija stlačivost zbilja može biti zanemariva, nego i za plinove čiji Mach broj ima vrijednost manju od 0.3. Za takve tokove kažemo da su nestlačivi. Ako je tok uz to i izotermičan, viskoznost fluida je također konstantna. Za takav oblik toka fluida možemo koristiti pojednostavljene Navier-Stokesove jednadžbe.

Fluid čija gustoća i temperatura približno su konstantne opisuje se vektorskim poljem brzine **u** i poljem tlakova *p*. Navedene vrijednosti općenito variraju u vremenu i prostoru i ovise o granicama koje okružuju fluid. Ako se pretpostavi da su brzina i tlak poznati za svaku točku fluida u nekom inicijalnom vremenu t=0, tada možemo tok predstaviti jednadžbama:

$$\nabla \cdot \mathbf{u} = 0 \qquad (2.7)$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \quad , \qquad (2.8)$$

gdje su:

ν - kinematička viskoznost fluida,

ρ - gustoća fluida i

f - vanjske sile koje djeluju na fluid.

Kao što se može uočiti izraz za tok fluida se sastoji od dvije jednadžbe. Jednadžba (2.7) je izvedenica zakona o očuvanju mase za nestlačive fluide. Iz jednadžbe je vidljivo da je divergencija vektora brzine toka jednaka nuli tj. količina fluida (jedinica mase) koji uđe u neko određeno područje je jednaka količini fluida koja izađe iz tog područja. Druga jednadžba (2.8) je malo kompleksnijeg oblika, ali je isto tako samo izvedenica zakona o očuvanju momenta ili energije. Iz te jednadžbe se može očitati da je promjena brzine gibanja je jednaka sumi četiriju izraza (redom s lijeva na desno): advekciji fluida, promjeni tlaka, difuziji i vanjskim silama koji djeluju na promatrani segment fluida.

1. Izraz za advekciju $(\mathbf{u} \cdot \nabla)\mathbf{u}$ predstavlja prijenos nekog svojstva fluida (u ovom slučaju brzine) samo gibanjem mase. Pobliže objašnjeno, taj izraz osigurava da će se svaki mali djelić toka ,"ubačen" u glavni tok, gibati u smjeru gradijenta vektora brzine.

2. Idući izraz je izraz za razliku tlaka koji predstavlja činjenicu da se fluid uvijek giba iz područja višeg tlaka u područje nižeg.
3. Izraz za difuziju $v\nabla^2\mathbf{u}$, uz konstantni iznos viskoznosti, opisuje koliko se brzo poništi razlika u brzinama unutar fluida koji okružuje zadanu točku. Uz veći iznos viskoznosti, brzine se brže ponište.
4. Zadnji izraz je izraz za vanjsku silu. Ta sila može biti bilo koji vanjski utjecaj. U nju ubrajamo gravitacijsku silu, silu uzgona te bilo koje sile definirane od strane korisnika. U ovom slučaju simulatora fluida, uz te dvije vrste sila, koristit će se i treći izraz koji će predstavljati silu koja obnavlja vrtloženje dima.

Osim pretpostavke o nestlačivosti fluida, moguće je učiniti još jedno pojednostavljenje: pretpostaviti neviskoznost fluida i tako definirati jednadžbe za neviskozan ili Eulerov tok.

2.1.2. NEVISKOZAN (EULER-OV) TOK

U tokovima koji su daleko od krutog stanja, efekt viskoznosti je obično vrlo malen. Ako su viskozni efekti sveukupno zanemarivi, Navier-Stokesove jednadžbe se svode na Eulerove. Jednadžba kontinuiteta je identična jednadžbi (2.7), dok se jednadžba očuvanja momenta svodi na:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f} \qquad (2.9)$$

S obzirom da se fluid smatra neviskoznim, on se ne može priljubiti uz prepreke i moguće će je da fluid sklizne niz krute granice. Eulerove jednadžbe se obično koriste za proučavanje stlačivih tokova pri velikom iznosu Mach broja. Na velikim brzinama, iznos Reynoldov broja je vrlo visok te efekti viskoznosti i turbulencije su vrlo bitni samo u malim područjima u blizini prepreka. Takvi tokovi su najčešće jako dobro predviđeni Eulerovim jednadžbama.

Unatoč tome što Eulerove jednadžbe nisu jednostavne za računanje, činjenica da nije potrebno računati niti jedan granični sloj uz prepreke omogućava korištenje grubljih mreža. To rezultira manjom količinom računanja po vremenskom koraku.

3.METODE RJEŠAVANJA JEDNADŽBI

3.1. ŠTO JE CFD?

Tok i slične pojave se mogu opisati parcijalno diferencijalnim jednadžbama, koja se u većini slučajeva ne mogu riješiti analitički. Kao što je već navedeno, CFD je polje znanosti koje se bavi proučavanjem rješavanja tih jednadžbi numeričkim metodama na računalu.
Da bi koristili numeričke metode rješavanja, potrebno je koristiti diskretizacijske metode koje aproksimiraju diferencijalne jednadžbe kao sustav algebarskih jednadžbi, koje se kao takve mogu riješiti na računalu. Aproksimacije su primjenjive na maloj domeni u prostoru i vremenu tako da numerička rješenja daju rezultate u diskretnim lokacijama u vremenu i prostoru. Unatoč tome što razina točnosti eksperimentalnih podataka ovisi o kvaliteti korištenih alata, točnost numeričkih rješenja ovise o kvaliteti korištene diskretizacije.

Unutar velikog opsega računalne dinamike fluida su aktivnosti koje pokrivaju od automatizacije dobro postavljenih inženjerskih dizajnerskih metoda do korištenja detaljnih rješenja Navier-Stokesovih jednadžbi kao zamjenu za eksperimentalno proučavanje prirode kompleksnih tokova. Zbog tako velikog opsega zanimanja CFD pronalazi mjesto u mnogim područjima znanosti i rada. Međutim interes ovog rada leži u malom dijelu CFD-a, a to su numeričke metode rješavanja postavljenih jednadžbi. U daljnjem će tekstu prvo biti navedene osnovne komponente i svojstva numeričkih metoda, zatim će biti objašnjena metoda koja je korištena za rješavanje postavljenog modela.

3.2. NUMERIČKE METODE RJEŠAVANJA

Jednadžbe koje predstavljaju dinamiku toka fluida su poznate više od stoljeća, ali su rješive samo za ograničen broj specifičnih slučajeva. Ti specijalni poznati slučajevi su od velike pomoći pri razumijevanju problematike toka ali su rijetko korisne za direktnu upotrebu u inženjerskoj analizi i dizajnu. Zbog toga su inženjeri tradicionalno bili prisiljeni koristiti neke druge metode pri rješavanju takvih jednadžbi.

Kao što je već spomenuto, najčešće se koriste pojednostavljene jednadžbe, koje su obično temeljene na kombinaciji aproksimacijske i dimenzijske analize, međutim empirički podaci, kao ulazi u sustav, su svejedno bili prijeko potrebni za dobivanje rješenja.

Zbog složenosti jednadžbi i nemogućnosti dobivanja empiričkih podataka eksperimentima za većinu slučajeva, počele su se razvijati numeričke metode rješavanja. Trebalo je malo mašte da bi se uvidjelo da korištenjem računala moguće je proučavati tok fluida lakše i mnogo učinkovitije. Jednom kada je otkrivena moć računala, interes za numeričke metode naglo je porastao. Rješavanje jednadžbi dinamike fluida na računalu je postalo toliko bitno da trenutno je time okupirana trećina istraživača koji rade na tom području, i proporcija još uvijek raste. To područje je poznato kao **računalna dinamika fluida** (engl. *computational fluid dynamics* - CFD).

3.3. KOMPONENTE NUMERIČKIH METODA

MATEMATIČKI MODEL

Matematički model je postavljanje jednadžbi uz granične uvjete koji dobro opisuju zadani problem. Taj korak je već učinjen u poglavlju 2.1.2.

METODA DISKRETIZACIJE

Nakon određivanja matematičkog modela potrebno je odabrati odgovarajuću metodu diskretizacije tj. metodu aproksimiranja jednadžbi kao sustav algebarskih jednadžbi za varijable na području diskretnih vrijednosti u vremenu i prostoru. Postoje mnogi pristupi diskretizaciji, najvažniji od njih su: metoda konačnih razlika (engl. *finite difference* -FD), metoda konačnih volumena (engl. *finite volume* -FV) i metoda konačnih elemenata (engl. *finite elemen*t -FE). Sve metode daju jednaka rješenja za dovoljno "fine" mreže. Međutim, pojedine metode su prikladnije za pojedine klase problema od ostalih.

KOORDINATE I SUSTAV BAZIČNIH VEKTORA

Ovisno o korištenom koordinatnom sustavu i korištenim bazičnim vektorima, proizlazi kako će se postaviti jednadžbe. Koordinatni sustav može biti Kartezijski, cilindričan, sferni itd. Odabir ovisi o cilju i obliku toka i utječe na izbor metode diskretizacije i odabira tipa mreže. Najčešće korišten sustav je Kartezijski sustav.

NUMERIČKA MREŽA

Diskretne lokacije na kojima se računaju vrijednosti varijabli su definirane numeričkom mrežom koja je u biti diskretna reprezentacija geometrijske domene na kojoj se rješava problem.

OGRANIČENE (KONAČNE) APROKSIMACIJE

Prateći odabir tipa numeričke mreže, potrebno je i odabrati aproksimaciju koja će se koristiti u diskretizacijskom procesu. U metodama konačnih razlika moraju biti odabrane metode aproksimacije derivacija u točkama mreže. Za metode konačnih volumena potrebno je odabrati metodu aproksimacije površinskih i volumnih integrala. I na kraju, za metodu konačnih elemenata potrebno je odabrati oblik funkcija (elemenata) i njihove težinske funkcije. Postoje mnoge mogućnosti odabira tih metoda. Pri odabiru se mora imati na umu kompromis između jednostavnosti, lakoj implementaciji, točnosti i računarskoj efikasnosti.

METODA RJEŠAVANJA

Diskretizacija povlači velik sustav nelinearnih algebarskih jednadžbi. Metoda rješavanja ovisi o postavljenom problemu. Za nestabilne sustave korištene su metode temeljene na onima koje se koriste za probleme inicijalne vrijednosti za uobičajene diferencijalne jednadžbe - metoda napredovanja kroz vrijeme (engl. marching in time). Stabilni tokovi se obično rješavaju pseudo-kretanjem kroz vrijeme (engl. pseudo-time marching) ili ekvivalentnom iteracijskom shemom.

KRITERIJI KONVERGENCIJE

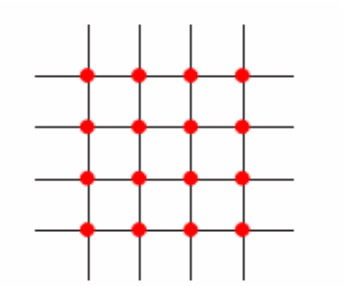
Na kraju, potrebno je postaviti uvjete konvergencije za iteracijske metode. Određivanje zaustavljanja iteracijskog procesa je vrlo važna komponenta, i za točnost postupka i za efikasnost točke pogleda.

3.4. DISKRETIZACIJA PROSTORA

Matematički model za rješavanje toka fluida predstavlja sustav diferencijalnih jednadžbi. Nakon što je model postavljen, potrebno ga je prilagoditi rješavanju na računalu, tj. potrebno je izabrati prikladnu metodu za diskretiziranje vremena i prostora. Najvažniji pristupi diskretizacije prostora su metoda konačnih razlika (engl. *finite diference* -FD), metoda konačnih volumena (engl. *finite volume* -FV) i metoda konačnih elemenata (engl. *finite element* -FE). Bitna obilježja navedenih metoda biti će opisana u daljnjem tekstu. Osim ovih metoda postoje i druge metode koje se rijetko koriste i najčešće su ograničene na specijalne slučajeve problema. Neke od tih metoda su sheme spektra, metode graničnih elemenata, celularni automati i druge.

Svaka od metoda daje isto rješenje ako je mreža dovoljno fino podijeljena. Unatoč toj činjenici, neke metode su prikladnije za neke klase problema od ostalih i odabir najčešće ovisi o preferiranju inženjera koji rješava problem.

3.4.1. METODA KONAČNIH RAZLIKA



Slika 3.1. Primjer podjele prostora za metodu konačnih razlika, čvorovi se nalaze na sjecištima

Metoda konačnih razlika najstarija je metoda za numeričko rješavanje diferencijalnih jednadžbi koju je predstavio Euler u 18. stoljeću. To je ujedno i najjednostavnija metoda ako se koristi jednostavna geometrija. U principu, metoda konačnih razlika se može primijeniti na bilo koji tip mreža. Međutim, najčešće se koriste strukturalne mreže kojima su linije poravnate sa linijama koordinatnog sustava. Najčešće se kreće od jednadžbi očuvanja u diferencijalnim oblicima koje je potrebno diskretizirati. Domena rješenja se sastoji od mreže (Slika 3.1.). Svaki presjek predstavlja jedan čvor kojeg unikatno predstavlja set indikatora, za

2D prostor (*i* ,*j*), za 3D prostor (*i, j, k*). Na svakom čvoru mreže aproksimira se polazna diferencijalna jednadžba tako da se parcijalne derivacije u jednadžbi zamjene sa aproksimacijskim izrazima u tekućem čvornom mjestu. To rezultira jednom algebarskom jednadžbom po čvoru mreže, u kojoj su nepoznanice vrijednosti varijabli tog i određenog broja susjednih čvorova. Naravno, broj jednadžbi i broj nepoznanica moraju biti jednaki. Na graničnim čvorovima gdje su ponuđene vrijednosti varijabli (Drichletovi uvjeti) jednadžbe nisu potrebne. Kada se koriste granični uvjeti koji sadrže derivacije (Neumannovi uvjeti), oni se moraju diskretizirati i pridružiti ostalim jednadžbama u sustavu.

Za računanje jednadžbi u čvorovima potrebne su nam aproksimacije derivacija. Najčešće korištene metode za aproksimacije derivacija su aproksimacija pomoću Taylorovog razvoja u red i aproksimacija polinomom (engl. *polinomical fitting*) . Kada je potrebno, te metode se također koriste za određivanje vrijednosti varijabli na lokacijama koje su izvan mrežnih čvorova (interpolacija).

Najčešća aproksimacija derivacije u metodi konačnih razlika temelji se na Taylorovom razvoju u red. Sve aproksimacije biti će izvedene za komponentu brzine u smjeru *x* osi (*u*). Za komponentu u *y* smjeru (*v*) izvod je analogan.

Na primjer, ako je $u_{i,j}$ označava *x* komponentu brzine u točki (i, j), tada brzina $u_{i+1,j}$ u točki (i+1, j) se može predstaviti Taylorovim razvojem u red oko točke (i, j):

$$u_{i+1,j} = u_{i,j} + \left(\frac{\partial u}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j} \frac{(\Delta x)^3}{6} + \ldots \qquad (3.1)$$

Navedena jednadžba je matematički egzaktan izraz za $u_{i+1,j}$, ako je broj članova beskonačan i konvergira, te $\Delta t \rightarrow 0$. Iz toga slijedi:

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \underbrace{\frac{u_{i+1,j} - u_{i,j}}{\Delta x}}_{(1)} - \underbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j} \frac{(\Delta x)^3}{6} + \ldots}_{(2)} \qquad (3.2)$$

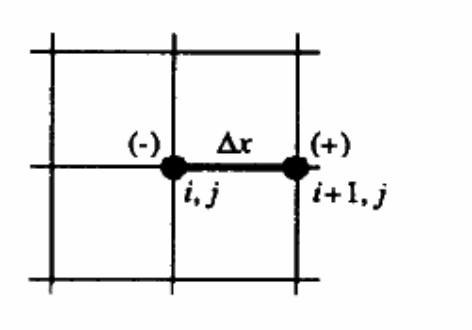
U tom izrazu (1) je aproksimacija prve derivacije konačnim razlikama:

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \qquad (3.3)$$

Ostatak izraza (2), (engl. *truncation error*), predstavlja izraz koji se zanemaruje pri aproksimaciji. U jednadžbi (3.2), izraz najnižeg reda u jednadžbi za pogrešku uključuje prvu potenciju Δx. Iz toga slijedi da je izraz za konačne razlike (3.3) prvog reda točnosti i točniji zapis je:

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x) \qquad (3.4)$$

U gore navedenoj jednadžbi $O(\Delta x)$ je formalna matematička notacija koja predstavlja izraz reda Δx. Zbog računanja derivacije pomoću idućeg člana u nizu $u_{i+1,j}$, tj. uzimanja podataka koji su postavljeni desno od točke (i, j), ova metoda se naziva prva derivacija prvog reda s korakom unaprijed (engl. *first-order forward difference*). Način dobivanja te aproksimacije se može opisati i grafički (Slika 3.2).

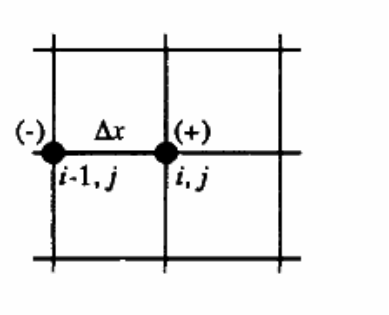


Slika 3.2. Dobivanje aproksimacije derivacije prvog reda s korakom unaprijed

Analogno tome, iz razvoja Taylorovog reda za element $u_{i-1,j}$ (jednadžba br. 3.5) može se izvesti i prva derivacija prvog reda s korakom unazad (engl. *first-order backward (rearward) difference*):

$$u_{i-1,j} = u_{i,j} + \left(\frac{\partial u}{\partial x}\right)_{i,j}(-\Delta x) + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j}\frac{(-\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j}\frac{-\Delta x)^3}{6} + \ldots \qquad (3.5)$$

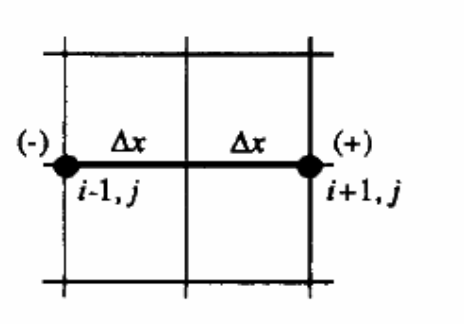
$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + O(\Delta x) \qquad (3.6)$$



Slika 3.3. Dobivanje aproksimacije derivacije prvog reda sa korakom unazad

U mnogim aplikacijama u području računalne dinamike fluida, točnost aproksimacija prvog reda nije dovoljna. Za konstruiranje kvocijenta konačnih razlika drugog reda točnosti potrebno je samo oduzeti jednadžbe (3.5) od (3.1) :

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + O(\Delta x)^2 \qquad (3.7)$$



Slika 3.4. Dobivanje aproksimacije prve derivacije drugog reda

Informacija korištena za dobivanje kvocijenta u jednadžbi (3.7) dolazi sa obje strane od promatrane mrežne točke (i, j) kao što je vidljivo na slici, tj. koriste se članovi $u_{i+1,j}$ i $u_{i-1,j}$. Tako se dobiva greška aproksimacije drugog reda točnosti koja je funkcija od $(\Delta x)^2$. Zbog toga se jednadžba (3.7.) naziva prva centralna derivacija drugog reda (engl. *second- order central difference*).

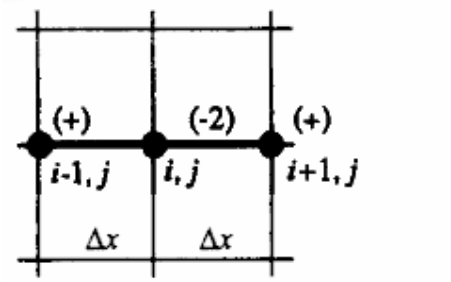
Sve do sad navedene jednadžbe u ovom poglavlju predstavljaju kvocijente konačnih razlika prvog reda koje predstavljaju prve parcijalne derivacije. Promatrajući Eulerove jednadžbe, prva parcijalna derivacija je sve što je potrebno za izračun. Međutim, za viskozne tokove koji se ponašaju po Navier-Stokesovim jednadžbama (viskozni tokovi) potrebno je

aproksimirati i drugu parcijalnu derivaciju. Sumiranjem Taylorovih razvoja (3.1) i (3.5) dobiva se:

$$u_{i+1,j} + u_{i-1,j} = 2u_{i,j} + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} (\Delta x)^2 + \left(\frac{\partial^4 u}{\partial x^4}\right)_{i,j} \frac{(\Delta x)^4}{12} + \ldots \qquad (3.8)$$

Iz toga se može izvesti izraz za drugu parcijalnu derivaciju:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + O(\Delta x)^2 \qquad (3.9)$$

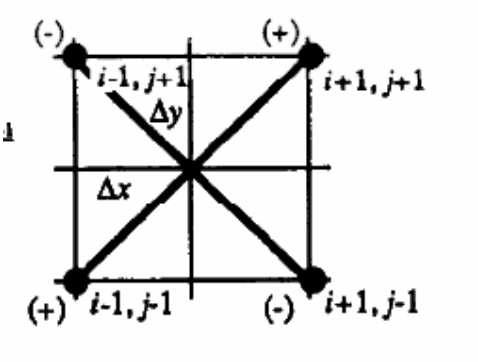


Slika 3.5. Dobivanje aproksimacije druge derivacije drugog reda

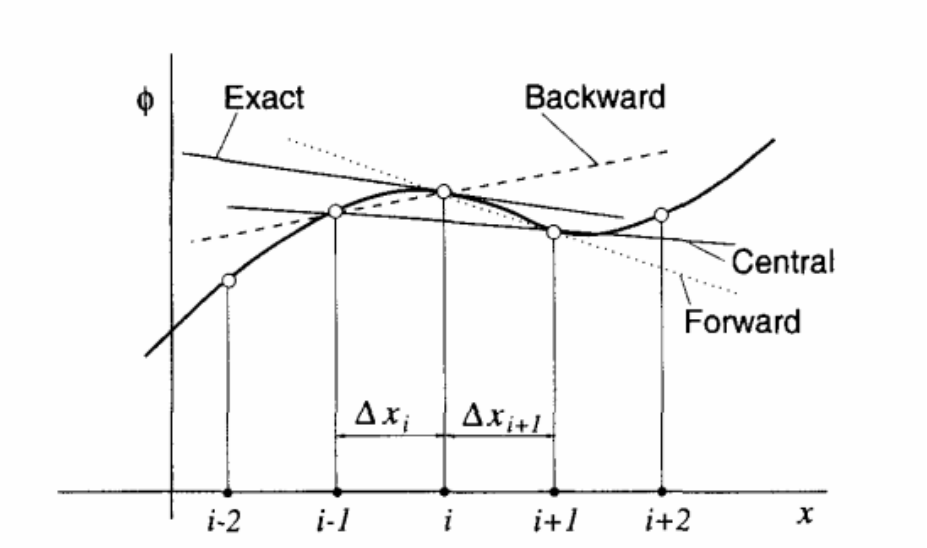
Iz formule je vidljivo da je ovo centralni diferencijal drugog reda točnosti. Osim derivacija po zasebnim komponentama potrebna nam je i mješovita druga derivacija. Ona se jednostavno dobiva deriviranjem Taylorovog reda (3.1) i (3.5) te njihovim oduzimanjem. Iz dobivenog izraza, daljnjim sređivanjem lako je dobiti izraz koji predstavlja derivaciju po x i y komponenti:

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} + O\left[(\Delta x)^2, (\Delta y)^2\right] \qquad (3.10)$$

To je izraz za drugu mješovitu centralnu derivaciju drugog reda točnosti.



Slika 3.6. Dobivanje aproksimacije mješovite derivacije



Slika 3.7. Geometrijska interpretacija svih aproksimacija parcijalnih derivacija, usporedba

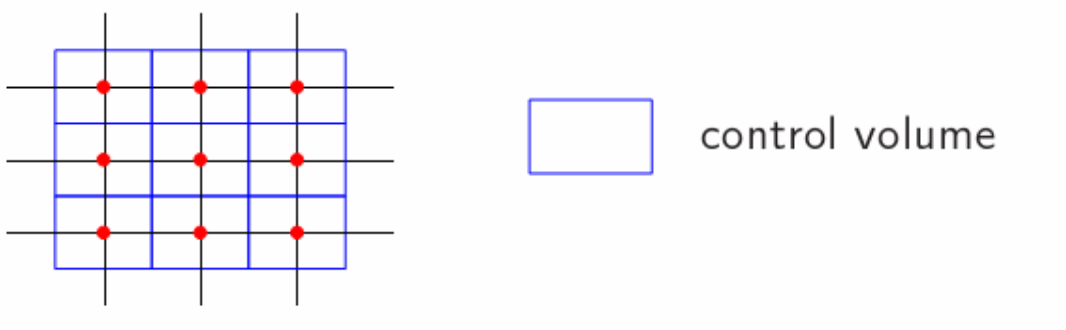
Usporedbu svih triju aproksimacija prvih parcijalnih derivacija moguće je vidjeti na slici 3.7. Iz takvog geometrijskog prikaza moguće je zaključiti koje su aproksimacije točnije. Kao što je poznato, derivacija u točki predstavlja nagib tangente na krivulju u toj točki (Na slici puni pravac označen sa *Exact*). Kao što je vidljivo aproksimacije kroz samo lijevu ili desnu susjednu točku nisu dovoljno točne, tj. njihova točnost uvelike ovisi o finoći podjele mreže (što je raspored točaka gušći, to će nagib pravca aproksimacije biti bliži nagibu točnog pravca). Centralna aproksimacija derivacije ima nagib najbliži točnoj derivaciji. Zbog toga je moguće zaključiti da je za točniju aproksimaciju bolje koristiti metode drugog reda. Međutim, zbog zahtijevanja većeg broja točaka za izračunavanje aproksimacije, metode višeg reda

točnosti zahtijevaju i više vremena za izračun po vremenskom koraku tj. takav algoritam je sporiji.

Kao što je spomenuto, nakon diskretizacije prostora i operatora, dobiva se algebarska jednadžba za svaki čvor mreže. Time dobivamo potpuni sustav jednadžbi. Ako je polazna jednadžba nelinearna (kao što su Navier-Stokesove i Eulerove jednadžbe), aproksimacije će sadržati neke nelinearne elemente. Numerička metoda rješavanja će tada zahtijevati dodatnu linearizaciju.

Prednosti metode konačnih razlika su jednostavnost i efektivnost na strukturnim mrežama. Uz to je posebno lako postići sheme višeg reda integracije. Nedostatak metode je da se ne poštuju zakoni očuvanja u slučaju da se ne vodi o tome računa posebno. Osim toga, bitni nedostatak je ograničenje na jednostavnu geometriju ako se radi o kompleksnim tokovima.

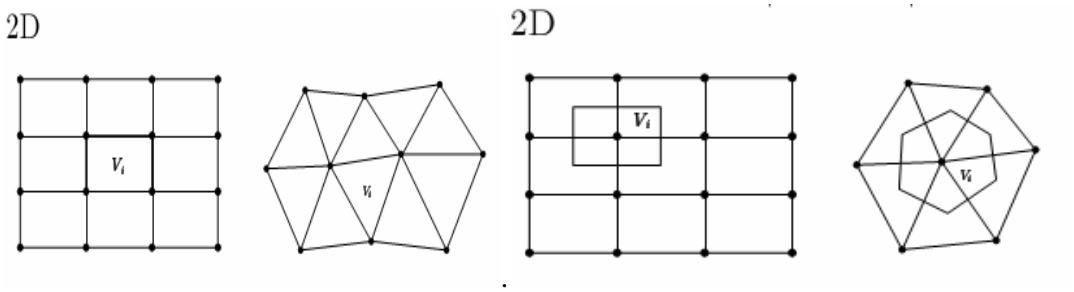
3.4.2. METODA KONAČNIH VOLUMENA



Slika 3.8. Metoda konačnih volumena dijeli prostor na kontrolne volumene u čijim središtima se nalaze čvorovi u kojima računamo vrijednosti varijabli

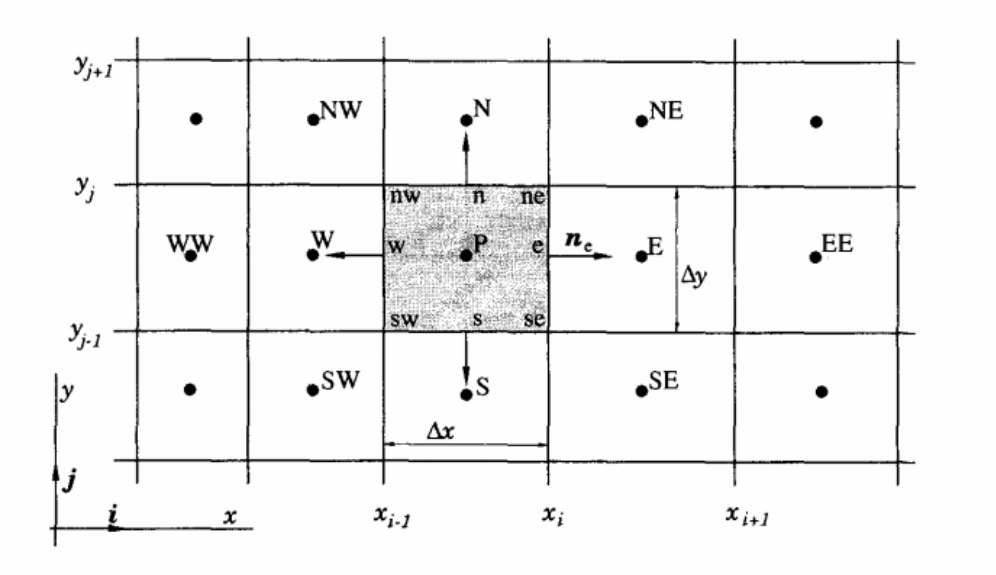
Metoda konačnih volumena najčešće kreće od integralnog oblika zakona očuvanja. Domena rješenja je podijeljena u konačan broj kontrolnih volumena (engl. *control volumes* - CV) ili ćelija (engl. *cells* ili *voxels*). Jednadžbe očuvanja se primjenjuju na svaku od ćelija. U centru svake ćelije se nalazi čvor u kojemu se izračunavaju vrijednosti varijabli (u nekim slučajevima neke od vrijednosti se postavljaju na stranice ćelija). Ostale vrijednosti se dobivaju interpolacijom. Površinski i volumni integrali se aproksimiraju prikladnim kvadratnim formulama. Kao rezultat dobiva se algebarska jednadžba za svaku ćeliju u kojima se pojavljuju neke vrijednosti susjednih ćelija. Ova metoda podržava bilo kakav tip mreže te je prikladna i za kompleksne geometrije. Za razliku od prošle metode, podržava i zakone očuvanja.

Najčešći pristup je definiranje ćelija odgovarajućom mrežom i određivanjem računskog čvora u centru ćelije. Kakogod, moguće je definirati (za strukturne mreže) čvorne lokacije prvo i onda konstruirati ćelije oko njih, tako da stranice ćelija leže na jednakoj udaljenosti od susjednih čvorova. Prednost prvog pristupa je ta da čvorna vrijednost predstavlja srednju vrijednost ćelije s većom točnošću (drugog reda) od drugog pristupa, s obzirom da je čvor postavljen na centru ćelije. Prednost drugog pristupa je u tome da aproksimacije derivacija na stranicama ćelija su točnije kad su stranice postavljene na polovini između dva čvora. Prva varijanta je češće korištena u praksi.



Slika 3.9. Dva različita pristupa u podjeli prostora na kontrolne volumene

Za rješavanje sustava ovom metodom potrebne su aproksimacije površinskih i volumnih integrala. U ovom radu će biti ukratko opisani načini aproksimacija samo u 2D prostoru. Na slici 3.10. je opisana korištena notacija.



Slika 3.10. Notacija korištena pri izvođenju aproksimacija površinskih integrala

Površina ćelije sastoji se od 4 (u 2D prostoru) ravne stranice, označene sa malim slovima koja odgovaraju njihovim smjerovima (engleski nazivi, e, w, n, s) u odnosu na čvor P koji se nalazi u centru ćelije. Tok kroz granice ćelija se tada može zapisati kao suma integrala kroz sve četiri stranice:

$$\int_S f dS = \sum_k \int_{S_k} f dS \,, \qquad (3.11)$$

gdje je f komponenta konvektivnog ili difuzivnog vektora toka u smjeru normale na stranicu ćelije. Zbog održavanja zakona o očuvanju, bitno je da se ćelije ne preklapaju, tj. da svaka stranica ćelije je unikatna za svaku od dviju ćelija koje leže sa svake njezine strane.

Za računanje površinskog integrala potrebno je poznavati integrant f duž cijele površine S. Ta informacija nije dostupna, osim u čvorovima (koji se nalaze u centrima ćelija) te je potrebna aproksimacija koja se odvija u dva koraka:

- integral se aproksimira izrazima sastavljenim od vrijednosti varijabli na jednom ili više lokacija na straniciama ćelija
- vrijednosti na stranicama ćelija se aproksimiraju od vrijednosti u čvorovima (centrima ćelija)

Najjednostavnija aproksimacija integrala je pravilo središnje točke (engl. *midpoint rule*) : integral je aproksimiran kao produkt integranta u središtu stranice ćelije (što je samo po sebi aproksimacija srednje vrijednosti površine) i područja stranice ćelije:

$$F_e = \int_{S_e} f dS = \bar{f}_e S_e \approx f_e S_e \qquad (3.12)$$

Ovakva aproksimacija integrala je drugog reda točnosti i zahtjeva vrijednost integranta f na lokaciji "e". S obzirom da vrijednost od f nije poznata na lokaciji "e", potrebno ju je dobiti interpolacijom. Da bi se sačuvala točnost drugog reda koje donosi pravilo središnje točke, vrijednost f_e treba se računati postupkom koji je isto tako barem drugog reda točnosti.

Drugi način aproksimacije površinskog integrala drugog reda točnosti u 2D prostoru je pravilo trapezoida, koje vodi do izraza:

$$F_e = \int_{Ss_s} f dS \approx \frac{S_e}{2}(f_{ne} + f_{se}) \qquad (3.13)$$

U tom slučaju potrebno je izraziti tok u kutovima ćelija.

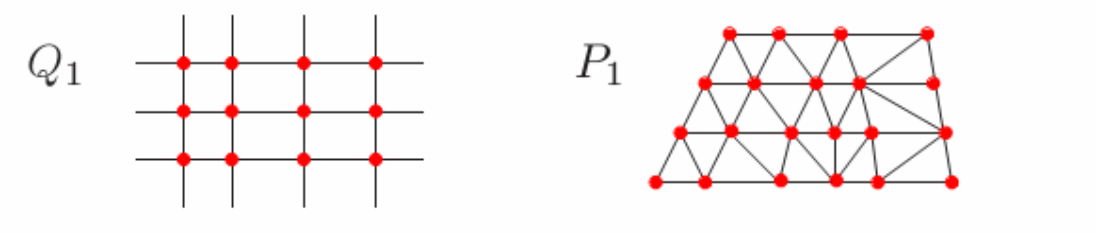
Za aproksimacije višeg reda, tok mora biti računat na više od dvije lokacije. Aproksimacija četvrtog reda je Simpsonovo pravilo, koje aproksimira integral kao:

$$F_e = \int_{Ss_s} f dS \approx \frac{S_e}{6}(f_{ne} + 4f_e + f_{se}) \qquad (3.14)$$

Kao što je uočljivo iz jednadžbe, za izračun su potrebne tri lokacije. Za dobivanje vrijednosti na tim lokacijama je potrebna interpolacija višeg reda, barem kubična.

Ovakav pristup je možda najjednostavniji za shvaćanje i programiranje. Svi izrazi koje je potrebno aproksimirati imaju fizikalno značenje, zbog čega je ta metoda popularna među inženjerima. Nedostatak metode je taj što metode integracije većeg reda je teže razviti u 3D prostoru.

3.4.3. METODA KONAČNIH ELEMENATA



Slika 3.11. Dva načina prikaza metode konačnih elemenata, kao što je vidljivo na slikama elementi ne moraju biti uniformni

Ova metoda je slična protekloj po mnogim značajkama. Domena je razbijena u set diskretnih volumena ili konačnih elemenata koji su generalno nestrukturirani (u 2D prostoru najčešće su to trokuti ili četverokuti, dok u 3D su to tetraedri ili heksaedri). Najvažnije obilježje ove metode je da su jednadžbe množene sa težinskim funkcijama prije integracije po cijeloj domeni. U najjednostavnijim oblicima, rješenje je predstavljeno linearnim oblikom funkcija za svaki element koji garantira kontinuitet rješenja po granicama elementa. Takva funkcija može se konstruirati od vrijednosti na kutovima elemenata. Težinske funkcije su najčešće iste forme.

Ovakva metoda diskretizacije prostora se vrlo rijetko koristi te neće biti pobliže razrađena u ovom radu.

3.5. RJEŠAVANJE SUSTAVA LINEARNIH JEDNADŽBI

U prethodnom poglavlju je pokazano kako je moguće diskretizirati jednadžbe koristeći različite metode. U svakom slučaju rezultat diskretizacije je sustav algebarskih jednadžbi koji je linearan ili nelinearan, ovisno o prirodi pripadajuće parcijalno diferencijalne jednadžbe od koje je sustav nastao. Za nelinearne slučajeve, jednadžbe nastale diskretizacijom rješavaju se iterativnim tehnikama koje uključuju pogađanje rješenja, linearizaciju jednadžbi oko tog rješenja i poboljšanja rješenja sve dok se ne postigne konvergencija rezultata. Stoga, bile jednadžbe linearne ili nelinearne, potrebne su metode za rješavanje linearnih algebarskih jednadžbi.

Matrice izvedene iz parcijalnih diferencijalnih jednadžbi su uvijek slabo popunjene, tj. većina elemenata matrice je jednaka nuli. Svi ne-nul elementi obično leže na malom broju dijagonala što je moguće iskoristiti za brže rješavanje zadanog sustava.

Sustav se može zapisati u matričnoj notaciji kao:

$$\mathbf{A\Phi = Q} \tag{3.15}$$

gdje je **A** kvadratna slabo popunjena matrica koeficijenata uz nepoznanice zapisane u vektoru **Φ**. Vektor **Q** sadrži izraze koji nisu uz varijable sustava. Kada se radi o velikim sustavima, matricu **A,** zbog malog broja ne-nul elemenata, nema smisla spremati kao dvodimenzionalno polje. Svaka dijagonala sa ne-nul elementima se sprema u poseban red, te se tako štedi memorijski prostor.

U nastavku biti će opisane neke od metoda za rješavanje linearnih algebarskih sustava koji nastaju diskretizacijom parcijalnih diferencijalih jednadžbi.

3.5.1. DIREKTNE METODE

Prvo će biti dan kratak pregled metoda za rješavanje općenitih matrica, s obzirom da su slabo popunjene matrice usko vezane uz njih. Za rješavanje takvih sustava koristi se potpuna matrična notacija (nasuprot dijagonalnoj koja se može koristiti za slabo popunjene matrice).

3.5.1.1. GAUSSOVA ELIMINACIJA

Temeljna metoda za rješavanje linearnih sustava algebarskih jednadžbi je Gaussova eliminacija. Bit te metode je sistematično reduciranje velikih sustava jednadžbi na manje. U tom postupku, elementi matrice se modificiraju, ali imena ovisnih varijabli se ne mijenjaju, prikladno je opisati metodu samim matričnim izrazima:

$$\mathbf{A} = \begin{pmatrix} A_{11}A_{12}A_{13}...A_{1n} \\ A_{21}A_{22}A_{23}...A_{2n} \\ \vdots \\ A_{n1}A_{n2}A_{n3}...A_{nn} \end{pmatrix} \qquad (3.16)$$

Bit algoritma je tehnika eliminiranja A_{21}, tj. zamjena vrijednosti tog elementa sa nulom. To se dobiva množenjem prve jednadžbe (prvi red matrice) sa A_{21}/A_{11} i oduzimanjem od druge jednadžbe (drugi red matrice). Općenito, za eliminiranje A_{i1}, prvi red matrice se množi sa A_{i1}/A_{11} i oduzima od i-tog retka. Nakon što se eliminiraju svi elementi ispod elementa A_{11}, niti jedna jednadžba od 2, 3, ...,n ne sadrži varijablu Φ_1 i tada te jednadžbe čine sustav od n-1 jednadžbi za varijable Φ_2, $\Phi_{3,....,}$ Φ_n. Ista procedura se primjenjuje na tom manjem sustavu jednadžbi za sve elemente ispod A_{22}.

Nakon što je proces gotov, izvorna matrica se svodi na gornju trokutastu:

$$\mathbf{A} = \begin{pmatrix} A_{11}A_{12}A_{13}...A_{1n} \\ 0, A_{22}A_{23}...A_{2n} \\ \vdots \\ 0,0,0...A_{nn} \end{pmatrix} \qquad (3.17)$$

S obzirom da se izvorne vrijednosti matrice neće više koristiti, efikasno je spremiti novo dobivenu matricu na mjesto izvorne matrice. Ovaj dio algoritma se naziva eliminacija unaprijed (engl. *forward elimination*) . Elementi desne strane jednadžbe Q_i također se modificiraju tijekom postupka.

Gornji trokutasti sustav jednadžbi se jednostavno rješava. Zadnja jednadžba sadrži samo jednu varijablu Φ_n i rješenje je intuitivno:

$$\phi_n = \frac{Q_n}{A_{nn}}. \qquad (3.18)$$

Jednadžba prije zadnje, sadrži samo nepoznanice Φ_n i Φ_{n-1} , te s obzirom da je Φ_n sada poznat moguće ju je također intuitivno riješiti. Kako navedeni algoritam napreduje prema gornjim jednadžbama, svaka jednadžba ima samo po jednu nepoznanicu, i rješenje se općenito može zapisati kao:

$$\phi_i = \frac{Q_i - \sum_{k=i+1}^{n} A_{ik}\phi_k}{A_{ii}}. \qquad (3.19)$$

Desna strana jednadžbe (3.19) je rješiva jer sve nepoznanice koje se pojavljuju u sumi su već poznate u koracima ispred. Ovaj dio algoritma koji počinje trokutastom matricom i računa nepoznanice se zove supstitucija unazad (engl. *back substitution*).

Nije teško pokazati da za velike brojeve nepoznanica *n*, broj operacija potreban za rješavanje sustava Gaussovom eliminacijom je proporcionalna sa $n^3/3$. Zbog toga je Gaussova eliminacija skupa za izračun, ali za potpune matrice je dobra kao i bilo koje druge metode. Međutim Gaussova eliminacija se ne vektorizira ili paralelizira efikasno i zato se rijetko koristi bez modifikacija u CFD problemima.

3.5.1.2. LU DEKOMPOZICIJA

Postoje brojne varijacije metoda Gaussove eliminacije. Većina nije interesantna u ovim razmatranjima. Jedna od varijacija koja ima vrijednost u CFD metodama je LU dekompozicija.

Kao što je moguće vidjeti u prethodnom poglavlju, u Gaussovoj eliminaciji, eliminacija unaprijed reducira potpunu matricu u gornju trokutastu. Taj proces može biti izveden puno efikasnije množenjem matrice **A** sa donjom trokutastom matricom. Samo po sebi to nije bitno, međutim s obzirom da je inverzna matrica donje trokutaste matrice također donja trokutasta, rezultat toga je da bilo koja matrica **A**, sa nekim ograničenjima koja su ovdje nebitna, može se rastaviti na donju (**L**) i gornju (**U**) trokutastu matricu:

$$\mathbf{A} = \mathbf{LU} \qquad (3.20)$$

Da bi rastavljanje bilo unikatno, zahtjeva se da svi dijagonalni elementi od **L**, L_{ii}, budu jedinstveni (alternativno može se uvjet primijeniti na matricu **U**).

Ono što čini to rastavljanje korisno je to što se radi o jednostavnom postupku. Gornja trokutasta matrica U je jednaka onoj sto se dobiva u prvoj fazi Gaussove eliminacije. Elementi matrice **L** su faktori kojima se množi u eliminacijskom procesu (A_{ji}/A_{ii}). Zbog toga je moguće da se rastavljanje matrice postigne manjom modifikacijom Gaussove eliminacije.

Prednost LU dekompozicije pred Gaussovom eliminacijom je taj da rastavljanje matrice se može izvesti bez znanja vrijednosti vektora **Q**.

3.5.2. ITERATIVNE METODE

3.5.2.1. TEMELJNI KONCEPT

Bilo koji kvadratni sustav linearnih nezavisnih jednadžbi se može riješiti pomoću Gaussove eliminacije ili LU dekompozicije. Nažalost, trokutasti faktori slabo popunjenih matrica nisu slabo popunjeni, pa stoga su te metode neučinkovite i skupe za takve sustave. Nadalje, pogreška diskretizacije je obično mnogo veća od točnosti računalnih aritmetičkih operacija, tako da nema potrebe rješavati sustave do te točnosti. Zadovoljavajuće rješenje je ono koje je nešto točnije od diskretizacijske sheme.

To ostavlja otvorena vrata iterativnim metodama. One se koriste za nelinearne probleme, ali su također korisne i za rješavanje slabo popunjenih sustava. U iterativnim metodama prvo se pogađa prikladno rješenje, i tada se koriste jednadžbe da bi ga sistematski poboljšali. Ako je svaka iteracija "jefitina" za izračun, i broj iteracija je mali, iterativni solver je brži od direktne metode. To je najčešći slučaj u CFD problemima.

Kreće se od matričnog zapisa problema navedenog u jednadžbi (3.15). Nakon *n* iteracija dobiva se aproksimacijsko rješenje Φ^n koje ne zadovoljava zadani sustav jednadžbi potpuno. Umjesto toga, postoji ostatak ρ^n:

$$\mathbf{A}\phi^n = \mathbf{Q} - \rho^n \qquad (3.21)$$

Oduzimajući jednadžbu gore (3.21) od jednadžbe (3.15), dobiva se relacija između iteracijskih pogrešaka definirana kao:

$$\varepsilon^n = \phi - \phi^n \tag{3.22}$$

Gdje je Φ rješenje kojem se konvergira, a ostatak:

$$A\varepsilon^n = \rho^n \tag{3.23}$$

Bit iteracijske metode je svođenje ostatka na nulu; u procesu ε također se svede na nulu. Da bi se pokazalo kako se to radi, uzima se u obzir iterativni postupak za linearni sustav. Tada se može zapisati:

$$\mathbf{M}\phi^{n+1} = \mathbf{N}\phi^n + \mathbf{B} \tag{3.24}$$

Očito pravilo koje se mora poštivati je da rezultat kojem se konvergira zadovoljava jednadžbu (3.15). S obzirom da po definiciji pri konvergenciji $\phi^{n+1} = \phi^n = \phi$, mora vrijediti:

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \qquad \mathbf{B} = \mathbf{Q} \tag{3.25}$$

ili generalno:

$$\mathbf{PA} = \mathbf{M} - \mathbf{N} \qquad \mathbf{B} = \mathbf{PQ} \tag{3.26}$$

Gdje je P nesingularna preduvjetna (engl. *pre-conditioning*) matrica.

Alternativna verzija se dobiva oduzimanjem $\mathbf{M}\phi^n$ sa svake strane jednadžbe (3.24). Time se dobiva:

$$\mathbf{M}(\phi^{n+1} - \phi^n) = \mathbf{B} - (\mathbf{M} - \mathbf{N})\phi^n \text{ ili } \mathbf{M}\delta^n = \rho^n \tag{3.27}$$

gdje $\delta^n = \phi^{n+1} - \phi^n$ se naziva ispravak i alternativa je iteracijskoj pogrešci.

Da bi iterativna metoda bila efektivna, rješavanje sustava (3.24) ne smije biti računski zahtjevno i metoda mora ubrzano konvergirati. Jedna iteracija mora sadržavati jednostavan izračun izraza $\mathbf{N}\phi^n$ i rješenja sustava. Drugi uvjet koji se mora ispuniti u svakoj iteraciji je jednostavnost izračuna inverzne matrice **M**.

3.5.2.2. KONVERGENCIJA

Kako je bilo navedeno, brza konvergencija iterativne metode je ključ njezine efektivnosti. U ovom odlomku će biti dana kratka analiza koja je korisna za razumijevanje što je mjerilo konvergencije i kako je poboljšati.

Za početak potrebno je izvesti jednadžbu koja opisuje ponašanje iteracijske pogreške. Kreće se od početnog uvjeta za konvergenciju $\phi^{n+1} = \phi^n = \phi$, tako da rješenje kojem se konvergira zadovoljava jednadžbu:

$$\mathbf{M}\phi = \mathbf{N}\phi + \mathbf{B} \tag{3.28}$$

Oduzimanjem te jednadžbe od jednadžbe (3.24) i koristeći definiciju iteracijske pogreške (3.22) dobiva se:

$$\mathbf{M}\varepsilon^{n+1} = \mathbf{N}\varepsilon^n \quad \text{ili} \quad \varepsilon^{n+1} = \mathbf{M}^{-1}\mathbf{N}\varepsilon^n \tag{3.29}$$

Iterativna metoda konvergira ako

$$\lim_{n\to\infty} \varepsilon^n = 0 \tag{3.30}$$

Kao jednostavni primjer uzet će se sustav koji koristi samo jednu jednadžbu. Pretpostavi se da se želi riješiti:

$$ax = b \tag{3.31}$$

i da se koristi iterativna metoda (primijetiti da je $m = a + n$ i p je brojač iteracija):

$$mx^{p+1} = nx^p + b \tag{3.32}$$

Tada pogreška mora zadovoljavati skalarni ekvivalent jednadžbe (3.29)

$$\varepsilon^{p+1} = \frac{n}{m}\varepsilon^{p} \tag{3.33}$$

Iz toga je vidljivo da se greška reducira brzo ako *n/m* je malo tj. ako je *n* mali, što znači da $m \approx a$. Pri konstruiranju iterativnih metoda za sustave, vrijedi analogno pravilo: što je matrica **M** bolje aproksimira matricu **A**, sustav konvergira brže.

U iterativnim metodama je bitno moći utvrditi iteracijsku grešku da bi se odlučilo kada je potrebno stati.

3.5.2.3. NEKE OSNOVNE METODE

U najjednostavnijoj Jacobijevoj metodi, **M** je dijagonalna matrica čiji elementi su dijagonalni elementi od matrice **A**. Za diskretizaciju Laplaceove jednadžbe u pet točaka, svaka iteracija je započeta u donjem lijevom (jugozapadnom) kutu domene i koristeći geogradsku notaciju (kao u poglavlju o metodi konačnih volumena) metoda glasi:

$$\phi_P^{n+1} = \frac{Q_P - A_S\phi_S^n - A_W\phi_W^n - A_N\phi_N^n - A_E\phi_E^n}{A_P} \tag{3.34}$$

Moguće je pokazati da za konvergenciju ova metoda zahtjeva broj iteracija proporcionalan kvadratu broja mrežnih čvorova u jednom smjeru. To znači da je ova metoda računski skuplja od direktne metode pa postoji malo razloga da bi se je koristilo.

U Gauss-Seidelovoj metodi, M je donji trokutasti dio matrice A. Ta metoda je specijalni slučaj SOR metode koja je opisana ispod. Ta metoda konvergira dva puta brže od Jacobijeve metode, međutim postoje i mnogo efikasnije metode.

Poboljšana verzija Gauss-Seidelove metode su sukcesivne pre-relaksacijske metode (engl. *successive over-relaxation-SOR*). Svaka iteracija započinje u donjem lijevom (jugozapadnom) kutu domene i ponovno koristeći geografsku notaciju, SOR metoda se može zapisati:

$$\phi_P^{n+1} = \frac{Q_P - A_S\phi_S^{n+1} - A_W\phi_W^{n+1} - A_N\phi_N^n - A_E\phi_E^n}{A_P} + (1-\omega)\phi_P^n \tag{3.35}$$

gdje ω predstavlja relaksacijski faktor, koji mora biti veći od 1 za ubrzanje, i *n* je broj iteracija. Postoje teorije koje se bave odabirom relaksacijskog faktora za jednostavne

probleme poput korištene Laplaceove jednadžbe na kvadratnoj domeni, međutim, to je teško primijeniti na kompleksnije probleme. Općenito, što je veća mreža, to je veći optimum relaksacijskog faktora. Za vrijednosti ω manje od optimuma, konvergencija je montona i što je ω veći to je razina konvergencije veća. Kada se premaši optimum, razina konvergencije počinje oscilirati. Pomoću tih činjenica moguće je naći optimum relaksacijskog faktora. Kada se koristi optimum relaksacijskog faktora, broj iteracija je proporcionalan broju mrežnih točaka u jednom smjeru i za iznos faktora jednak 1, ova metoda se svodi na Gauss-Seidelovu metodu.

Osim ovih postoje mnoge numeričke metode koje se bave rješavanjem linearnih i nelinearnih sustava algebarskih jednadžbi. Neke od njih su metoda konjugiranih gradijenata (engl. *conjugate gradient method*) , više mrežne metode (engl. *multigrid methods*) i mnoge druge. Za pobliže upoznavanje tih metoda čitatelj se upućuje na poglavlja o rješavanju linearnih sustava u knjigama [15] i [16]

3.6. METODE ZA NESTABILNE PROBLEME - DISKRETIZACIJA VREMENA

Pri računanju nestabilnih tokova, moramo uzimati u obzir i 4 dimenziju: vrijeme. Vrijeme je potrebno diskretizirati kao i prostor. Možemo smatrati da je mreža po kojoj je podijeljeno vrijeme, ili tipa konačnih razlika, kao diskretne točke u vremenu, ili tipa konačnih volumena kada se promatraju "vremenski volumeni". Glavna razlika između vremenskih i prostornih koordinata je smjer utjecaja: budući da sila u bilo kojoj prostornoj lokaciji može utjecati na tok bilo gdje drugdje, primjena sile u danom trenutku će utjecati na tok tek u budućnosti, ne postoji utjecaj unazad. To povlači da niti jedan uvjet ne može biti nametnut na rješenje (osim na granice) poslije inicijalizacije računa, što ima jak utjecaj na odabir strategije rješavanja.

Nakon diskretizacije prostornih derivacija u glavnim jednadžbama (kao što su za primjer toka Navier-Stokesove ili Eulerove jednadžbe), postiže se povezani sustav nelinearnih diferencijalnih jednadžbi oblika:

$$\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}, t) \tag{3.36}$$

Takva jednadžba se može integrirati po vremenu koristeći neku od metoda za rješavanje problema nestabilnih tokova. Da bi se ostalo vjerno prirodi vremena, sve metode računanja napreduju kroz vrijeme korak po korak tj. "marširanjem" (engl. *time- marching methods*). Za rješavanje stabilnih tokova, prostorna diskretizacija vodi do povezanog sustava nelinearnih algebarskih jednadžbi oblika:

$$\vec{F}(\vec{u}) = 0 \tag{3.37}$$

Zbog nelinearnosti navedenih jednadžbi potrebna je neka vrsta iterativne metode za dobivanje rješenja. Na primjer, moguće je odabrati put do stabilnog stanja i koristiti metodu marširanja kroz vrijeme za integriranje nestabilnog oblika jednadžbi dok rješenje ne bude dovoljno blizu stabilnog rješenja. Zbog toga, metode marširanja kroz vrijeme su relevantne i za stabilne i za nestabilne tokove. Kada se navedene metode koriste za rješavanje stabilnih

tokova, cilj je jednostavno uklanjanje prijelaznog udjela rješenja što je brže moguće, vremenska točnost nije bitna.

3.6.1. METODE MARŠIRANJA KROZ VRIJEME

3.6.1.1. DVORAZINSKE METODE

Za probleme inicijalnih vrijednosti, dovoljno je promatrati običnu diferencijalnu jednadžbu prvog reda sa inicijalnim uvjetom:

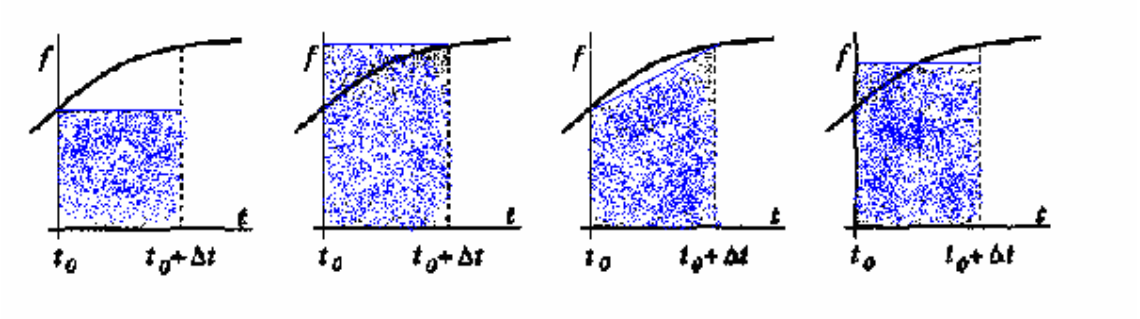
$$\frac{d\phi(t)}{dt} = f(t,\phi(t)) \quad ; \qquad \phi(t_0) = \phi^0 \ . \tag{3.38}$$

Glavni problem je naći rješenje Φ za neko kratko vrijeme Δt nakon inicijalne točke. To rješenje u vremenskom trenutku $t_1 = t_0 + \Delta t$, je inicijalno stanje za novi korak i rješenje se dalje nastavlja za trenutke $t_2 = t_1 + \Delta t$, $t_3 = t_2 + \Delta t$, ...

Najjednostavnije metode se mogu konstruirati integrirajući formulu (3.38) od vremenskog trenutka t_n do $t_{n+1} = t_n + \Delta t$:

$$\int_{t_n}^{t_{n+1}} \frac{d\phi}{dt} dt = \phi^{n+1} - \phi^n = \int_{t_n}^{t_{n+1}} f(t,\phi(t))dt \tag{3.39}$$

gdje se koristi skraćena notacija $\phi^{n+1} = \phi(t_{n+1})$. Ta jednadžba je egzaktna. Kakogod, desna strana jednadžbe se ne može razviti bez poznavanja rješenja te je potrebna neki oblik aproksimacije. Teorem srednje vrijednosti tvrdi ako je integrant razvijen u ispravnoj točki $t=r$ između t_n i t_{n+1}, integral je tada jednak izrazu $f(r,\phi(r))\Delta t$, ali to je od male koristi ako je r nepoznat. Zbog toga se koriste neke od aproksimativnih numeričkih metoda za razvijanje integrala.



Slika 3.12. Aproksimacija vremenskog integrala f(t) na intervalu Δt, s lijeva na desno: eksplicitna Eulerova metoda, implicitna Eulerova metoda, trapezno pravilo i pravilo srednje točke

Ako se integral na desnoj strani jednadžbe (3.39) se rješava koristeći vrijednost integranta u inicijalnoj točki, dobiva se:

$$\phi^{n+1} = \phi^n + f(t_n, \phi^n)\Delta t \tag{3.40}$$

što je poznato kao **eksplicitna ili unaprijedna Eulerova metoda**.

Ako se umjesto toga koristi konačnu točku u rješavanju integrala, dobiva se **implicitna ili unazadna Eulerova metoda**:

$$\phi^{n+1} = \phi^n + f(t_{n+1}, \phi^{n+1})\Delta t \tag{3.41}$$

Moguće je razviti još jednu metodu koristeći središnju točku intervala:

$$\phi^{n+1} = \phi^n + f(t_{n+\frac{1}{2}}, \phi^{n+\frac{1}{2}})\Delta t \tag{3.42}$$

što je poznato kao **pravilo središnje točke** i može se smatrati osnovom bitne metode za rješavanje parcijalnih diferencijalnih jednadžbi -**Leapfrogove metode**.

I kao zadnja solucija, moguće je koristiti linearnu interpolaciju po pravcu između inicijalne i krajnje točke konstruirajući aproksimaciju:

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f(t_n, \phi^n) + f(t_{n+1}, \phi^{n+1})\right]\Delta t \tag{3.43}$$

koja se naziva trapezoidno pravilo i također je osnova bitne metode za rješavanje parcijalnih diferencijalnih jednadžbi - **Crank-Nicolsonove metode**.

Kolektivno, ove metode se nazivaju dvorazinske metode jer uključuju vrijednosti nepoznanica iz dva vremenska trenutka (metode koje se temelje na pravilu središnje točke se mogu, ali ne moraju svrstavati u dvorazinske metode).

Iz navedenih jednadžbi može se primijetiti da sve metode, osim prve, zahtijevaju vrijednost od $\Phi(t)$ u nekom vremenskom trenutku različitom od $t=t_n$ (što je inicijalna točka integracijskog intervala u kojem je rješenje poznato). Za takve metode nije moguće izračunati desnu stranu izraza bez sljedeće iteracije. Zbog toga, prva metoda pripada klasi eksplicitnih metoda, dok ostale pripadaju implicitnim metodama.

Sve metode daju dobra rješenja za mali Δt. Međutim, ponašanje metoda za veće vremenske korake je bitno zbog toga što za probleme s varijabilnim vremenskim skalama često je cilj izračunati, sporo, dugoročno ponašanje rješenja i kratki vremenski koraci su samo smetnja. Problemi sa velikim rasponom vremenskih koraka se nazivaju engl. *stiff* i oni su najveća teškoća na koju se nailazi pri rješavanju regularnih diferencijalnih jednadžbi. Zbog toga je od velike važnosti ispitati ponašanje određene metode pri velikim vremenskim koracima. To prerasta u pitanje stabilnosti metode.

Postoje mnoge definicije stabilnosti u literaturi. Ovdje će se koristiti gruba definicija koja naziva metodu stabilnom ako ona dolazi do ograničenog rješenja kada je rješenje temeljnih diferencijalnih jednadžbi također ograničeno. Za eksplicitnu Eulerovu metodu stabilnost zahtjeva zadovoljenje izraza:

$$\left|1+\Delta t\frac{\partial f(t,\phi)}{\partial \phi}\right|<1 \qquad (3.44)$$

za koji, ako $f(t,\phi)$ dozvoljava kompleksne vrijednosti, zahtjeva se da $\Delta t\partial f(t,\phi)/\partial\phi$ bude ograničeno na područje unutar kružnice sa centrom u -1. Metoda koja zadovoljava navedeni uvjet naziva se uvjetno stabilnom. Za realne vrijednosti od f, uvjet (3.44) se reducira na:

$$\left|\Delta t\frac{\partial f(t,\phi)}{\partial \phi}\right|<2 \qquad (3.45)$$

Sve ostale metode definirane do sada su bezuvjetno stabilne, tj. daju ograničeno rješenje za bilo koji vremenski korak ako $\partial f(t,\phi)/\partial\phi < 0$. Unatoč toj činjenici, implicitna Eulerova metoda teži glatkim rješenjima čak i kada je Δt jako velik dok trapezoidno pravilo daje rješenje koje oscilira sa laganim prigušenjem.

Konačno, potrebno je razmotriti i pitanje točnosti pojedinih metoda. Globalno gledajući, teško je reći nešto o tom problemu zbog velike razlike jednadžbi kojima se može pristupiti navedenim metodama. Za jedan mali korak moguće je koristi Taylorov razvoj da bi se pokazalo da eksplicitna Eulerova metoda, krećući od poznatog rješenja za t_n, daje rješenje za $t_n + \Delta t$ sa greškom proporcionalnom sa $(\Delta t)^2$.Međutim, s obzirom da broj koraka potreban za računanje nekog konačnog vremena $t = t_0 + T$ je obrnuto proporcionalan Δt , i greška se gomila pri svakom vremenskom koraku, greška će na kraju biti proporcionalna sa samim Δt. Zbog toga je eksplicitna Eulerova metoda prvog reda točnosti. Implicitna Eulerova metoda je također prvog reda točnosti, dok trapezoidno pravilo i pravilo središnje točke imaju pogreške proporcionalne sa $(\Delta t)^2$ i zbog toga su drugog reda točnosti. Moguće je dokazati da je drugi red točnosti najveći red koji se može doseći dvorazinskim metodama.

3.6.1.2. PREDIKTOR-KOREKTOR METODE

Svojstva navedena za dvorazinske metode su prilično općenita. Eksplicitne metode su lagane za programiranje te koriste jako malo računalne memorije i vrijeme za računanje. Njihov glavni nedostatak je nestabilnost za velike korake. U jednu ruku, implicitne metode zahtijevaju iterativno rješenje za dobivanje vrijednosti u novom vremenskom koraku. To ih čini težima za programiranje i zahtijevaju mnogo više računalne memorije i vremena po koraku, ali su zato stabilnije (Ovdje navedene implicitne metode su bezuvjetno stabilne, međutim to ne vrijedi generalno za sve implicitne metode, ali su one u svakom slučaju stabilnije od eksplicitnih metoda). Postavlja se pitanje da li se mogu kombinirati dobra svojstva iz obje vrste metoda. Prediktor-korektor metode (engl. *Predictor-Corrector Methods*) su pokušaj takvog pristupa.

U ovoj metodi, rješenje u novom vremenskom koraku se predviđa koristeći eksplicitnu Eulerovu metodu:

$$\phi_{n+1}^{*} = \phi^{n} + f(t_n, \phi^{n})\Delta t \qquad (3.46)$$

gdje * indicira da to nije finalna vrijednost rješenja za t_{n+1}. Umjesto toga, rješenje se ispravlja trapezoidnim pravilom koristeći ϕ^*_{n+1} za računanje derivacije:

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f(t_n, \phi^n) + f(t_{n+1}, \phi^*_{n+1})\right]\Delta t \qquad (3.47)$$

Može se dokazati da je ova metoda drugog reda točnosti (red točnosti trapezoidnog pravila) ali sadrži, ugrubo gledajući, stabilnost eksplicitne Eulerove metode .

Ova metoda također pripada familiji dvorazinskih metoda, što znači da je drugi red točnosti najviši koji se može postići.

3.6.1.3. METODE KOJE KORISTE VIŠE TOČAKA

Za bolju razinu točnosti potrebno je koristiti podatke sa više točaka. Te točke mogu biti one u kojima su podaci već izračunati ili točke između t_n i t_{n+1} koje se koriste izričito samo za računalnu vjerodostojnost. Prve takve metode su bile metode koje koriste više točaka (engl. *multipoint methods*).

Najpoznatije takve metode, Adamsove metode, su izvedene pomoću metode izjednačavanja polinoma derivacijama u određenim točkama u vremenu. Ako se Lagrangeov polinom namjesti na $f(t_{n-m}, \phi^{n-m}), f(t_{n-m+1}, \phi^{n-m+1}), \ldots, f(t_n, \phi^n)$ i rezultat se iskoristi za računanje integrala (3.38), dobiva se metoda reda m+1. Metode ovog tipa se nazivaju Adams-Bashforthove metode. Za rješavanje parcijalnih diferencijalnih jednadžbi koriste se jedino metode nižeg reda. Metoda prvog reda je eksplicitna Eulerova metoda, dok metode drugog i trećeg reda su:

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{2}\left[3f(t_n, \phi^n) - f(t_{n-1}, \phi^{n-1})\right] \qquad (3.48)$$

i

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{12}\left[23f(t_n, \phi^n) - 16f(t_{n-1}, \phi^{n-1}) + 5f(t_{n-2}, \phi^{n-2})\right] \qquad (3.49)$$

Ako se u interpolaciju polinoma uključuje podatak iz vremenskog trenutka t_{n+1}, radi se o implicitnim metodama poznatim kao Adams-Moultonove metode. Metoda prvog reda koja se dobiva tim načinom je Eulerova implicitna metoda, drugog reda je trapezoidno pravilo, dok je metoda trećeg reda:

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{12}\left[5f(t_{n+1},\phi^{n+1}) + 8f(t_n,\phi^n) - f(t_{n-1},\phi^{n-1})\right] \qquad (3.50)$$

Obično metode koriste Adams-Bashfourthove metode (m-1)-og reda kao prediktor, i Adams-Moultonovu metodu m-tog reda kao korektor. Tim načinom se može postići prediktor-korektor metoda bilo kojeg reda točnosti.

Pomoću pristupa metoda koje koriste više točaka moguće je jednostavno konstruirati metode bilo kojeg reda točnosti. Osim toga, te metode je relativno lako programirati i zahtijevaju samo jednu evaluaciju derivacije po vremenskom koraku, što ih čini relativno "jeftinim". Glavni nedostatak metode je da zahtjeva podatke iz većeg broja točaka i nije moguće pokrenuti program s podacima samo iz inicijalne točke. Jedan od mogućih rješenja tog problema je početi s malim vremenskim korakom i metodom nižeg reda, te onda polako sa dobivanjem većeg broja točaka povećavati red metode.

Navedene metode su temelj za mnoge metode rješavanja diferencijalnih jednadžbi korištenim u raznim programima. U tim programima, koriste se procjenjivači pogrešaka za određivanje točnosti rješenja u svakom koraku. Ako rješenje nije dovoljno blizu točnoj vrijednosti, povećava se red metode sve do maksimalnog reda koji dozvoljava korišteni program. U suprotnom slučaju, kada je rješenje točnije nego što je potrebno, može se smanjiti red metode i pri tome uštedjeti vrijeme potrebno za računanje. Zbog otežanog mijenjanja veličine vremenskog koraka u takvim metodama, red metode se smanjuje samo kada se dosegne maksimalni red metode.

Zbog toga što navedene metode koriste podatke iz više vremenskih koraka, mogu rezultirati ne-fizičkim rješenjima što vodi do nestabilnosti. Taj problem je moguće djelomično zaobići koristeći pomno odabranu početnu metodu.

3.6.1.4. RUNGE-KUTTA METODE

Poteškoće pri startanju metoda koje koriste više točaka mogu se nadvladati koristeći točke između t_n i t_{n+1} umjesto ranijih točaka. Metode s ovakvim načinom pristupa se nazivaju Runge-Kutta metode. Slijede dva primjera.

Runge-Kutta metoda drugog reda se sastoji od dva koraka. Prvi korak se može promatrati kao polu-korak temeljen na eksplicitnoj Eulerovoj metodi; nakon toga slijedi korektor temeljen na pravilu središnje točke zbog čega je metoda drugog reda:

$$\phi^*_{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} f(t_{n,}\phi^n) \tag{3.51}$$

$$\phi^{n+1} = \phi^n + \Delta t(t_{n+\frac{1}{2}}, \phi^*_{n+\frac{1}{2}}) \tag{3.52}$$

Ova metoda je laka za upotrebu i nije potrebno imati dodatne podatke za startanje, samo inicijalni uvjeti koje zahtijevaju same diferencijalne jednadžbe. U mnogim pogledima, slična je već opisanim prediktor-korektor metodama.

Runge-Kutta metode većeg reda točnosti su također jako razvijene. Najpopularnije su metode četvrtog reda. Prva dva koraka te metode koriste eksplicitnu Eulerovu metodu kao prediktor i implicitnu Eulerovu metodu kao korektor u trenutku tn+1/2. Nakon toga slijedi prediktor temeljen na pravilu središnje točke za cijeli korak i Simpsoovo pravilo kao finalni korektor zbog kojeg je metoda četvrtog reda:

$$\phi^*_{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} f(t_n, \phi^n) \tag{3.53}$$

$$\phi^{**}_{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{2} f\left(t_{n+\frac{1}{2}}, \phi^*_{n+\frac{1}{2}}\right) \tag{3.54}$$

$$\phi^*_{n+1} = \phi^n + \Delta t f\left(t_{n+\frac{1}{2}}, \phi^{**}_{n+\frac{1}{2}}\right) \tag{3.55}$$

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{6}\left[f\left(t_n, \phi^n\right) + 2f\left(t_{n+\frac{1}{2}}, \phi^*_{n+\frac{1}{2}}\right) + 2f\left(t_{n+\frac{1}{2}}, \phi^{**}_{n+\frac{1}{2}}\right) + f\left(t_{n+1}, \phi^*_{n+1}\right)\right] \tag{3.56}$$

Postoje mnoge varijacije ove metode. Najveći problem je to što je prilično teško razviti metode većeg reda i, kao što je već viđeno u gore navedenim metodama, *n*-ti red Runge-Kutta metoda zahtjeva razvoj derivacije *n* puta po vremenskom koraku, što čini ovu metodu skupljom za izračun u usporedbi sa ostalim metodama. Međutim, Runge-Kutta metode su točnije (koeficijenti uz izraz pogreške su manji) i stabilnije od ostalih metoda istog reda.

3.7. SPECIJALNA METODA SEMI-LAGRANGE

3.7.1. SEMI-LAGRANGEOVA METODA

Odabir integracijske metode, tj. modela aproksimacije kretanja kroz vrijeme, je od velike važnosti za modele postavljene pomoću diferencijalnih jednadžbi. Najveći problem integracijskih shema je da odabirom maksimalnog vremenskog koraka upravlja kompromis između stabilnosti metode i točnosti izračuna. Za stabilnu integraciju potreban je mali vremenski korak, tako da je pogreška ograničavanja vremenskog koraka mnogo manja od pogreške ograničavanja prostora, te je tada potrebno izvoditi mnogo vremenskih koraka, što inače ne bi bio slučaj. Zbog takvih ograničenja, do sada navedene metode ne rješavaju jednadžbe u stvarnom vremenu te je potreban drugačiji pristup problemu. Semi-Lagrangeova metoda je izazvala velik interes zbog pružanja mogućnosti većih vremenskih koraka (bez velikih gubitka točnosti) za razliku od Eulerovih metoda.

Semi-Lagrangeova metoda, u upotrebi je u računalnoj grafici od 1999. godine kada ju je prestavio Stam u svom radu [2], koristi se za rješavanje izraza advekcije u jednadžbi očuvanja momenta gibanja fluida. Taj izraz uvodi u jednadžbu nelinearnost, stoga nije moguće dobiti rješenje uobičajenim analitičkim metodama. Semi-Lagrangeova metoda rješava jednadžbe advekcije koristeći metodu karakteristika za svaku točku unutar regularne mreže. Za razliku od cjelovite Lagrangeove metode, koja uzima inicijalnu mrežu vrijednosti i promjene vrši na njoj, semi-Lagrangeova metoda je efektivno praćenje čestica između dvije mreže od vremenskog trenutka do idućeg vremenskog trenutka.

U Eulerovim metodama za rješavanje jednadžbe advekcije, promatrač promatra svijet oko sebe iz fiksne točke u prostoru. Takve sheme djeluju dobro na regularnim Kartezijevim mrežama, ali često vode do prestrogih ograničenja vremenskog koraka zbog izbjegavanja nestabilnosti. U Lagrangeovim metodama, promatrač promatra svijet, ali i prati putanju čestica fluida. Takve metode daju mnogo veće vremenske korake, međutim imaju i jedan nedostatak: inicijalno uobičajeni razmaci u prostoru između seta čestica (karakteristično za dim) s vremenom uglavnom postanu nepravilni, te se može dogoditi da bitna svojstva toka budu zanemarena, loše prezentirana. Ideja semi-Lagangeove metode je obuhvatiti dobra svojstva obiju metoda, ispravnu rezoluciju Eulerove metode i istaknute stabilnosti Lagrangeove metode. To je postignuto korištenjem drugačijeg seta čestica za svaki vremenski

korak. Te čestice su odabrane tako da njihova putanja završava točno u točkama regularne Kartezijeve mreže na kraju svakog vremenskog koraka.

Za rješavanje postavljene advekcijske jednadžbe koristi se metoda karakteristika, koja je osnova semi-Lagrangeove metode. Ta metoda se koristi za rješavanje jednadžbi oblika:

$$\frac{\partial \phi}{\partial t} + \mathbf{u}(\mathbf{x},t) \cdot \nabla \phi = 0 \qquad (3.57)$$

gdje je Φ skalarno polje i $\mathbf{u}(\mathbf{x}, t)$ je funkcija brzine. Jednadžba (3.57) pasivno propagira svojstvo Φ kroz vektorsko polje brzine **u**. Semi-Lagrangeova metoda se temelji na opažanju da jednadžba (3.57) propagira Φ po karakterističnoj krivulji $\mathbf{x}=\mathbf{p}(t)$ definiranom sa:

$$\frac{d}{dt}\mathbf{p}(t) = \mathbf{u}(\mathbf{p}(t),t) \ , \quad \text{uz} \qquad \mathbf{p}(0) = \mathbf{x}_0 \, . \qquad (3.58)$$

Iz toga slijedi mogućnost pronalaženja vrijednosti Φ, za svaki trenutak t, tako da se pronađe karakteristična krivulja koja prolazi kroz $(\mathbf{x}, t)$, prateći je unatrag do neke prijašnje točke $(\mathbf{x_0},t_0)$, gdje je vrijednost od Φ poznata, i postavljajući $\Phi(\mathbf{x},t)= \Phi (\mathbf{x_0},t_0)$. Za dane vrijednosti Φ za vrijeme t_n, ovakav način rješavanja aproksimira vrijednost $\Phi(\mathbf{x},t_{n+1})$ za bilo koju točku **x** u vremenskom trenutku $t_{n+1}=t_n + \Delta t$ razvijajući prijašnju brzinu $\mathbf{u}(\mathbf{x}, t_n)$, aproksimirajući karakteristiku kroz **x** unatrag po ravnoj liniji

$$\mathbf{p}(t) \approx \mathbf{x} - (t_{n+1} - t)\mathbf{u}(\mathbf{x},t_n) \, , \qquad (3.59)$$

i interpolirajući Φ za vremenski trenutak t_n do točke

$$\mathbf{p}(t_n) \approx \mathbf{x} - (\Delta t)\mathbf{u}(\mathbf{x},t_n) \, . \qquad (3.60)$$

Tada se izraz $\phi(x,t_{n+1})$ postavlja na interpoliranu vrijednost $\phi(\mathbf{p}(t_n),t_n)$.

Uz pomoć navedene metode rješava se advekcijski izraz u jednadžbi očuvanja momenta kroz vremenski interval $[t, t+\Delta t]$ za svaki element fluida. Osim tog izraza, ova metoda se koristi i za izračun izraza za promjenu tlaka i temperature fluida.

Ukratko, ova metoda donosi novosti u rješavanje nelinearnih diferencijalnih jednadžbi drugačijim pristupom. Proces je stabilan, ukidaju se svi uvjeti na veličinu vremenskog koraka u ovisnosti o finoći podjele prostora. Druga novost je dijeljenje vodeće jednadžbe na više dijelova i rješavanje po koracima, te se time omogućuje korištenje jednostavnijih solvera. I treća novost je uvođenje polja bez divergencije koje se računa u zasebnom koraku. Te tri značajke obilježavaju ovu metodu i omogućavaju realan prikaz fluida u stvarnom vremenu uz stabilnost za bilo koju veličinu vremenskog koraka. U nastavku će biti pobliže opisano rješavanje diferencijalnih jednadžbi toka plinovitih fluida ovom metodom.

4. POSTAVLJANJE PROBLEMA

Svrha ovog rada je postavljanje algoritma za izradu simulatora dima u 2D ili 3D području koji se zasniva na radu Fedkiwa, Stama i Jensena [1], te rada Stama [2] i njegovoj teoriji stabilnih fluida (engl. *stable fluid*). U daljnjem tekstu je opisana problematika, metode koje se koriste pri rješavanju, te način rješavanja postavljenog problema.

4.1. OSNOVNE JEDNADŽBE

U uvodu su navedene jednadžbe koje vjerno opisuju tok fluida. Interes ovog rada se svodi na uski dio područja fluida - na simulaciju dima i sličnih plinovitih fluida. Pretpostavka je da navedeni fluid je neviskozan, nestlačiv i ima gustoću konstantne vrijednosti. Efekt viskoznosti u plinovitom stanju fluida je zanemariv, pogotovo na grubljim mrežama gdje numerička disipacija dominira nad fizikalnom viskoznošću i molekularnoj difuziji. Kada je brzina gibanja plinova daleko ispod brzine zvuka efekt stlačivosti je također zanemariv (kao što je i navedeno u uvodu). Te dvije pretpostavke poprilično pojednostavljuju način izračuna jednadžbi. Zbog navedenih činjenica, za opis gibanja opisanog fluida koriste se nestlačive Eulerove jednadžbe (iste su navedene u uvodnom dijelu):

$$\nabla \cdot \mathbf{u} = 0 \tag{4.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}\,, \tag{4.2},$$

gdje je brzina plina **u** predstavljena kao vektor brzina kroz sve tri dimenzije :

$$\mathbf{u} = (u, v, w)\,. \tag{4.3}$$

Postavljene jednadžbe rješavaju se u osnovna dva koraka (ovo je samo način rješavanja navedenih jednadžbi, kasnije će biti navedeni koraci algoritma) :

- Prvo se rješava "posredničko polje brzina" (engl. *intermediate velocity field*) $\mathbf{u}^*$ rješavajući jednadžbu (4.2) kroz vremenski korak Δt bez izraza za tlak:

$$\frac{\mathbf{u}^* - \mathbf{u}}{\Delta t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \mathbf{f} \tag{4.4}$$

- Nakon tog koraka prisiljavamo polje $\mathbf{u}^*$ da bude nestlačivo koristeći **projekcijsku metodu** koja će biti objašnjena kasnije u tekstu**.** To je ekvivalentno računanju tlaka iz Poissonove jednadžbe (Poissonova jednadžba se rješava iterativnim solverom):

$$\nabla^2 p = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^* \tag{4.5}$$

sa čistim **Neummannovim graničnim uvjetom**:

$$\frac{\partial p}{\partial \mathbf{n}} = 0\,, \tag{4.6}$$

na graničnoj točki s normalom **n**.

Nakon toga se računa nestlačivo polje brzina tako da se iz posredničkog polja brzina oduzme gradijent tlaka:

$$\mathbf{u} = \mathbf{u}^* - \Delta t \nabla p\,. \tag{4.7}$$

Osim jednadžbi kojima se dobiva brzina čestica fluida, trebamo jednadžbe po kojima se mijenjaju temperatura i gustoća:

$$\frac{\partial T}{\partial t} = -(\mathbf{u} \cdot \nabla)T \tag{4.8}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho\,. \tag{4.9}$$

Obje vrijednosti, gustoća i temperatura utječu na brzinu fluida. Teški plinovi imaju tendenciju pada prema dolje zbog utjecaja gravitacije, dok topliji plinovi imaju tendenciju rasta zbog utjecaja uzgona. Za primjer ovog simulatora se koristi jednostavan model

uključivanja ovih efekata tako da se definira vanjska sila koja je direktno proporcionalna gustoći i temperaturi plina:

$$\mathbf{f}_{\mathbf{buoy}} = -\alpha\rho\mathbf{z} + \beta(T - T_{amb}-)\mathbf{z} \qquad (4.10)$$

gdje je z=(0, 0, 1) vektor s orijentacijom "prema gore", T_{amb} ambijentalna temperatura, a α i β dvije pozitivne konstante s prikladnom jedinicom tako da gore navedena jednadžba (4.10) ima fizikalno značenje. Dobro je primijetiti da kada je ρ=0 i T= T_{amb} , definirana sila je jednaka nuli. Kako se uzima utjecaj gravitacije?

Uz navedenu silu uzgona, koristit ćemo i dodatni izraz za silu koji će definirati pojave malih vrloga te tako činiti izgled dima življim. Više o tome u poglavlju o metodi ograničavanja vrtložnosti.

Jednadžbe (4.8) i (4.9), kao i jednadžba (4.4), sadrže operator divergencije $-(\mathbf{u}\cdot\nabla)$. Divergencija je transportiranje očuvane skalarne vrijednosti u vektorskom polju. U meteorologiji i fizikalnom proučavanju fluida, divergencija predstavlja strujanje (obično horizontalno) nekih atmosferskih svojstava fluida, kao što su u ovom slučaju toplina i gustoća, u smjeru brzine fluida. Taj navedeni izraz se rješava semi-Lagrangeovom metodom, koja je već opisana poglavlju metoda vremenske diskretizacije.

Prije rješavanja postavljenog problema potrebno je opisati još jednu bitnu metodu koja se koristi u ovom solveru. Ta metoda je ograničavanje vrtložnosti koja služi za izbjegavanje numeričke disipacije i čini prikaz dima realističnijim.

4.2. OGRANIČAVANJE VRTLOŽNOSTI

Uobičajeno je da mješavine zraka i dima sadrže veće prostorne devijacije udružene sa značajnim količinama rotacionih i turbulentnih struktura različitih razmjera (npr. dim cigarete). Nefizikalna numerička disipacija, koju mnoge numeričke metode imaju za posljedicu, guši ta interesantna svojstva plinova. Cilj ovog pristupa je "vraćanje" tih fenomena nazad u mrežu prikaza. Jedan od načina vraćanja takvih pojava nazad u tok bi bilo kreiranje slučajnih ili pseuodo-slučajnih preturbacija toka koristeći heuristički ili fizikalno temeljen model. Ovakav način stvara rotacione i turbulentne strukture međutim ne postavlja ih na fizikalno ispravno mjesto u toku gdje bi se one inače pojavljivale. Umjesto toga, detalji se dodaju slučajnim odabirom pozicije u tok, te se time osigurava "živost" simuliranog plina. Međutim ključ realistične animacije dima je da izgleda kao pasivno realistična prirodna tvorevina nasuprot navedenoj metodi koja stvara "živo" biće od dima. Zato se uvode fizikalno korektne metode, kao što je metoda ograničavanja vrtložnosti (engl. *vorticity confinement*-VC) koja je opisana dalje u tekstu.

Vrtložne pojave u tokovima sa velikim iznosom Reynoldsovog broja (turbulentna polja-dodatak A.1) su učestale. Numeričko rješenje Navier-Stokesovih jednadžbi za vrtložne pojave malih razmjera u toku je računalno prezahtjevno. Stoga bilo kakav izračun ponašanja takvih tokova mora sadržavati izdvojenu metodu koja će implementirati takvo ponašanje fluida. Postoje razne metode koje rješavaju navedeni problem. Međutim korištenje nekih metoda kao što je tradicionalna metoda simuliranja velikih vrtloga (engl. *Large Eddy Simulation* - LES) će uzrokovati preveliko zagušenje za najmanje vrtložne pojave zbog numeričke disipacije koja nastaje zbog diskretizacije vodećih jednadžbi i bilo kojeg korištenog modela za simulaciju viskoznosti vrtloga. Te male pojave vrtložnosti mogu biti bitne za prikaz u mnogim tokovima. Zbog njihove važnosti za realističnost prikaza razvijaju se nove metode kao što je metoda ograničavanja vrtložnosti.

Ograničavanje vrtložnosti (dalje u tekstu VC) je metoda koja rješava pojedine pojave malih razmjera do razine otprilike dvije mrežne ćelije. Na primjer, VC dozvoljava izoliranim, uskim vrtlozima da propagiraju kroz proizvoljno daleke udaljenosti bez numeričke disipacije, zadržavajući kompaktan oblik (do razine otprilike dvije ćelije). Objektivno načelo VC-a je pronalaženje tih bitnih vrtloga malih razmjera.

U smislu efekata propagirajućih vrtloga na ostatak polja, najvažnije svojstvo je da je cirkulacija očuvana na uskom području i da je položaj centroida navedenog vrtloga točno izračunat. VC metodom je moguće uspješno izračunati takve zahtjeve bez korištenja specijalne logike ili Lagrangeovih markera.

Metoda ograničavanja vrtložnosti uključuje sustav diferencijalnih jednadžbi koje mogu biti zapisane kao diskretizacija jednadžbi kontinuiteta i momenta uz jedan dodatak: "ograničavajući" izraz. Diskretizacija može biti niskog stupnja, ali svejedno imati uske stabilne strukture koje mogu neodređeno propagirati kroz tok fluida. Može se dokazati da pridodani izraz čuva cirkulaciju i može očuvati moment ekspilicitno. Zbog ograničavajućeg izraza djeluje samo na strukture malih razmjera, dok strukture većih razmjera pokrivaju dogovorene CFD metode.

Glavni cilj metode ograničavanja vrtložnosti je identičan metodama koje koriste tehnike pronalaženja sukoba i kontakata (engl. *shock and contact methods*): tretiranje bitnih gledišta na male pojave na djelotvoran način, bez neophodnog rješavanja detalja oko unutrašnje strukture. Kao u slučaju sukoba, naglasak je na tome da su te male uske strukture na vanjskom toku točno izračunate i da ostaju istih razmjera. Ovaj način modeliranja omogućuje točan izračun kompleksnih tokova i izbjegava računarski prezahtjevne detalje viskoznih rješenja unutar pojava malih razmjera s velikim gradijentom.

2D Kevin-Helmholtzova nestabilnost (Dodatak A.4) može se iskoristiti za pojašnjenje glavne ideje []. U takvom toku, metoda ograničavanja vrtloženja predstavlja jednostavan način vraćanja energije (i njezinih popratnih pojava) nazad u filtrirano polje toka fluida: u 2D diskontinuitetima kontakta filtriranog polja nedostajati će energija koja je prisutna u nefiltriranim poljima. Ta energija koja nedostaje, koja se naziva latentna energija, je potrebna za simuliranje nestabilnosti prisutnih u pravim, nefiltriranim diskontinuitetima kontakta. Čak kada bi se izbjegle sve numeričke disipacije, i riješile precizno Eulerove jednadžbe, rješenje svejedno ne bi bilo dobar pokazatelj nestablinosti i nasljeđenog zasićenja fizičkog dijela sloja do puno kasnijeg trenutka, ako su inicijalna polja vrtloga neprirodno raspoređena**,** ili filtrirana**,** čime su nestablinosti prigušene**.** Ograničavanje vrtložnosti dopušta mnogo realističnije rješenje toka s mnogo točnijom prezentacijom nestablinosti. Nadalje, stvarna tanka ploha, nakon što postane nestabilna, počinje se omotavati umjesto da divergira. VC modelira ovaj fenomen s automatskim zasićenjem koje je popraćeno inicijalnom nestabilnošću. Pri tome, inicijalne nestabilnosti se pretvaraju u male vrtložne tvorevine (engl.*"blobs"*), raširene kroz nekoliko mrežnih ćelija. Time se predstavljaju "omotavajuće spirale" dima u pravim

tokovima, koji bi trebali imati precizne detalje i zahtijevaju mnogo finiju mrežu, stoga moraju biti prezentirani kao izdvojeni (filtrirani) objekti.

4.2.1. FORMULACIJA METODE

Glavna ideja ograničavanja vrtložnosti je izračun stabilnih vrtložnih struktura malih razmjera koje (u izolaciji) mogu propagirati nedefinirano dugo. Unatoč tome što VC jednadžbe mogu biti napisane kao diskretizacija sustava parcijalno diferencijalnih jednadžbi, dobiveno rješenje na pojavama malih razmjera nije točno (čak ni aproksimacijski) rješenje orginalnih jednadžbi. Zbog toga VC metoda pronalazi, ili modelira, pojave malih razmjera kroz samo par mrežnih ćelija, te je diskretizacijska pogreška reda samo O(1). Kao takva, prateća pojava je efektivno nelinearan izdvojeni val koji "živi" na mrežnoj rešetki. Za veće razmjere, VC se reducira u odabranu CFD metodu, u kojoj je diskretizacija navedenih parcijalno diferencijalnih jednadžbi točna.

Dvije različite verzije metode ograničavanja vrtložnosti su razvijene, ali su male razlike između njihovih formulacija i numeričkih rezultata. VC1 metoda je osmišljena prva i sadrži efektivno vođenje vrtložnosti unutar toka duž njegovog gradijenta. Ta verzija vrlo dobro pokriva očuvanje momenta, ali neće biti opisana u ovom radu (za detalje [2]). VC2, koja eksplicitno čuva moment, će biti pobliže objašnjena, za neke dodatne detalje pogledati [4] i [5].

VC jednadžbe su diskretizirane jednadžbe kontinuiteta i momenta s dodatnim izrazom:

$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \left[\mu \nabla^2 \mathbf{u} - \varepsilon \mathbf{s}\right], \qquad (4.11),$$

Tumačenje jednadžbi je jednako kao i kod N-S jednadžbi, samo što:

- μ-koeficijent difuzije koji kombinira numeričku disipaciju i fizičku viskoznost (koja je mnogo manja pa se može zanemariti)
- ε-numerički koeficijent koji kontrolira "uvjet kontrakcije" - $\boldsymbol{s}$ (engl. *contraction term*)

Kombinacija μ i ε kontrolira pojave najmanjih razmjera. Uvjet kontrakcije korišten u VC2 je harmonička sredina dana sa:

$$\mathbf{s} = \nabla \times \mathbf{w}^n \tag{4.12}$$

gdje je:

$$\mathbf{w}^n = \frac{\boldsymbol{\omega}^n}{\tilde{\boldsymbol{\omega}}^n} \left[\frac{\sum_l (\tilde{\boldsymbol{\omega}}_l^n)^{-1}}{N} \right]^{-1} \tag{4.13}$$

i

$$\tilde{\boldsymbol{\omega}}_l^n = \left| \boldsymbol{\omega}_l^n \right| + \delta \tag{4.14}$$

U ovom izrazu $\boldsymbol{\omega}$ je vektor vrtložnosti, l je korišten za definiranje (N točaka) matrice na kojima je izračunata srednja harmonička vrijednost i δ je mala pozitivna konstanta (10^{-8}) koja se koristi za prevenciju konačnih računskih grešaka u preciznosti. Harmonička srednja vrijednost je odabrana tako da najmanje vrijednosti u matrici imaju veće težine. Može se koristiti ograničavajući izraz baziran na minimumu funkcije, međutim dokazano je da je bolje koristiti glatke funkcije kao što je navedena harmonička sredina.

Ova metoda će biti korištena u solveru da bi poništila numeričku disipaciju koja prati Lagrangeovu metodu. Za razliku od Stamovog solvera opisanog u radu [2], dim će izgledati življi i realističniji. S obzirom da se u jednadžbu dodaje kao iznos sile, uz samo par operacija više, neće značajno usporiti cijeli algoritam.

5. RJEŠAVANJE ZADANOG SUSTAVA

5.1. DISKRETIZACIJA PROSTORA

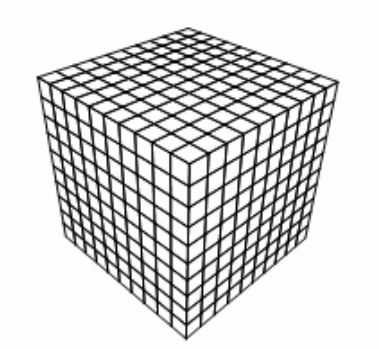
Unatoč kompleksnosti sustava diferencijalnih jednadžbi, moguće je riješiti jednadžbe na intuitivan način. Prvi korak pri rješavanju je diskretizacija jednadžbi i prostora u kojem se razvija model. Postoje brojni načini kako to učiniti, ali bitno je imati na umu nekoliko stvari:

- U tipičnim grafičkim aplikacijama koje uključuju fluide, postoje mnoge granice između fluida i ostalih objekata i između fluida i medija koji ga okružuje (najčešće zraka). Cijena izračuna može biti minimizirana ako se takva sučelja homogeno pridruže modelu umjesto da budu tretirana zasebno kao specijalni slučajevi.

- Općenitost je sve. Korisnici sustava moraju biti u mogućnosti da specificiraju geometriju okružja brzo, bez referenciranja na jednadžbe na kojima se temelje točni granični uvjeti.

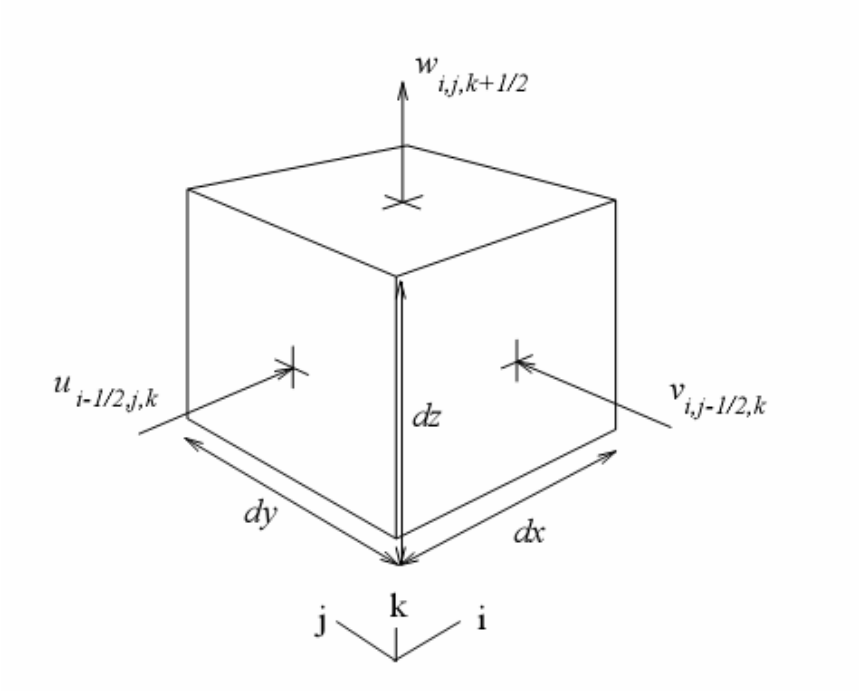
- Mora biti moguće uvesti vanjsku kontrolu sustava, tako da animator može točno definirati kako će se fluid ponašati.

- Opseg gibanja koji može biti animiran koristeći tehnike trebao bi uključivati set efekata dostupnih u postojećim računalno-grafičkim metodama, i proširen dodavajući nove, interesantne i korisne metode.



Slika 5.1. Primjer volumnog prostora podijeljenog u ćelije

Da bi se navedene jednadžbe (2.7) i (2.8), ili (2.7) i (2.9) za neviskozne fluide, bile rješive na računalu, potrebna je diskretizacija cijelog područja na kojem rješavamo navedene jednadžbe. Pri toj diskretizaciji, prepreke i sredstvo koje okružuje fluid se tretiraju isto kao i fluid, samo sa nekim specijalnim svojstvima koja ostaju konstantna tijekom cijelog izračuna. Za rješavanje postavljenih jednadžbi koristi se već navedena metoda konačnih volumena (FV). Ta metoda dijeli domenu rješenja u konačan broj susjednih kontrolnih volumena - ćelija (engl. *voxel* ili *control volume* - CV) raspoređene u fiksnu pravokutnu mrežu poravnate sa Kartezijskim koordinatnim sustavom (slika 5.1). Komponente vektora brzine, *u* u smjeru osi x, *v* u smjeru osi y i *w* u smjeru osi z, definirane su u centrima svake stranice ćelije i vezane su lokalno, dok tlak *p*, i ostale skalarne vrijednosti definiramo u centru svake od ćelija kao što je prikazano na slici 5.2.



Slika 5.2. 3D ćelija sa polovičnom notacijom

Koristeći polovičnu indeks notaciju komponente zapisujemo kao:

$$\begin{aligned} u_{i+1/2,j,k} \quad & i = 0,...,N,\ j,k = 1,...,N \\ v_{i,j+1/2,k} \quad & j = 0,...,N,\ i,k = 1,...,N \\ w_{i,j,k+1/2} \quad & k = 0,...,N,\ i,j = 1,...,N \end{aligned} \tag{5.1}$$

Iz slike je vidljivo da je:

$$v_{i,j-1/2,k} \equiv v_{i,(j-1)+1/2,k} \tag{5.2}$$

Koristeći takvu notaciju možemo definirati neke diskretne operatore i izraze koji će se koristiti u metodi rješavanja. Divergencija se tada definira kao:

$$(\nabla \cdot \mathbf{u})_{i,j,k} = \frac{(u_{i+1/2,j,k} - u_{i-1/2,j,k} + v_{i,j+1/2,k} - v_{i,j-1/2,k} + w_{i,j,k+1/2} - w_{i,j,k-1/2})}{h}\ , \tag{5.3}$$

dok se diskretni gradijenti (primijetiti da je $\nabla p = (p_x, p_y, p_z)$) definiraju kao:

$$\begin{aligned}
(p_x)_{i+1/2,j,k} &= \frac{(p_{i+1,j,k} - p_{i,j,k})}{h} \\
(p_x)_{i,j+1/2,k} &= \frac{(p_{i,j+1,k} - p_{i,j,k})}{h} \\
(p_x)_{i,j,k+1/2} &= \frac{(p_{i,j,k+1} - p_{i,j,k})}{h}
\end{aligned} \tag{5.4}$$

Diskretan Laplaceov operator je kombinacija operatora divergencije i gradijenta.

Isto tako potrebno je definirati diskretnu verziju vrtložnosti $\boldsymbol{\omega}=(\omega^1, \omega^2, \omega^3)$. Prvo se računaju brzine u sredini ćelija pomoću aritmetičke sredine:

$$\begin{aligned}
\overline{u}_{i,j,k} &= \frac{(u_{i-1/2,j,k} + u_{i+1/2,j,k})}{2} \\
\overline{v}_{i,j,k} &= \frac{(v_{i,j-1/2,k} + v_{i,j+1/2,k})}{2} \\
\overline{w}_{i,j,k} &= \frac{(w_{i,j,k-1/2} + w_{i,j,k+1/2})}{2}\ .
\end{aligned} \tag{5.5}$$

Tada vrijedi:

$$\omega^1_{i,j,k} = \frac{\overline{w}_{i,j+1,k} - \overline{w}_{i,j-1,k} - \overline{v}_{i,j,k+1} + \overline{v}_{i,j,k-1}}{2h}$$

$$\omega^2_{i,j,k} = \frac{\overline{u}_{i,j,k+1} - \overline{u}_{i,j,k-1} - \overline{w}_{i+1,j,k} + \overline{w}_{i-1,j,k}}{2h}$$

$$\omega^3_{i,j,k} = \frac{\overline{v}_{i+1,j,k} - \overline{v}_{i-1,j,k} - \overline{u}_{i,j+1,k} + \overline{u}_{i,j-1,k}}{2h} \qquad (5.6)$$

Sve sile su definirane u centrima ćelija. Da bi se dobile vrijednosti na stranicama ćelija potrebno je napraviti aritmetičku sredinu susjedne dvije ćelije. Ako je polje sila **f**=(f^1, f^2, f^3), tada u svakom koraku algoritma brzine se ažuriraju na način:

$$u_{i+1/2,j.k} += \frac{\Delta t(f^1_{i,j,k} + f^1_{i+1,j,k})}{2}$$

$$v_{i,j+1/2.k} += \frac{\Delta t(f^2_{i,j,k} + f^2_{i,j+1,k})}{2}$$

$$w_{i+1/2,j.k} += \frac{\Delta t(f^3_{i,j,k} + f^3_{i,j,k+1})}{2} \qquad (5.7)$$

Sada su definirane sve potrebne diskretizacije, opis rješavanja jednadžbi je naveden u idućem poglavlju.

5.2. GRANIČNI UVJETI

Na početku simulacije sadržaj svake ćelije je određen (poznat). S obzirom na to što ćelija sadrži dijelimo ih na:

- **Okupirana ćelija** - ćelija koja sadrži krutu prepreku
- **Ispunjena ćelija** - ćelija koja sadrži čestice fluida
- **Površinska ćelija** - ćelija koja je na granici između fluida i okolnog medija (bitno za viskozne tokove)
- **Prazna ćelija** - koja ne sadrži niti fluid niti prepreku.

U svim slučajevima komponente brzine i tlak su definirani za svaku od ćelija.

Za manipuliranje granicama objekata uronjenih u fluid obilježavamo svaku ćeliju koja siječe objekt kao okupiranu. Sve okupirane ćelije imaju postavljene komponente brzina na

brzine objekta. Slično, temperatura u centru ćelije se postavlja na temperaturu objekta. Zbog toga je moguće kreirati mnogo interesantnih efekata samo pomicanjem ili zagrijavanjem objekta. Gustoća plina unutar objekta se postavlja na nulu. Ali, za izbjegavanje iznenadnog pada gustoće na granici s objektom, na graničnim ćelijama postavlja se gustoća plina jednaka gustoći najbliže susjedne neokupirane (ispunjene) ćelije.

Postoje dva tipa graničnih uvjeta koji su korisni u praktičnim aplikacijama:

- periodični granični uvjeti, i
- fiksni granični uvjeti.

U slučaju periodičnih granica, fluid je definiran na n-dimenzionalnom torusu (n=2,3). U tom slučaju ne postoje zidovi, samo fluid koji se rasprostire po prostoru. Unatoč tome što takvi fluidi nisu upotrebljivi u praksi, korisni su za kreiranje mapa tekstura. Isto tako, takvi granični uvjeti omogućavaju vrlo elegantnu implementaciju koja koristi brzu Furierovu transformaciju [2]. Drugi tip graničnih uvjeta se uzima u obzir kada se fluid nalazi u nekoj ograničenoj domeni D. U tom slučaju, granični uvjeti su zadani funkcijom $\mathbf{u_D}$ definiranoj na granici ∂D domene.

Za bolju diskusiju navedenih graničnih uvjeta pogledati rad Fostera i Metaxasa [18]. U svakom slučaju, granični uvjeti bi trebali biti takvi da komponenta normale na polje brzine je nula na graničnim područjima.

5.3. METODA RJEŠAVANJA SUSTAVA

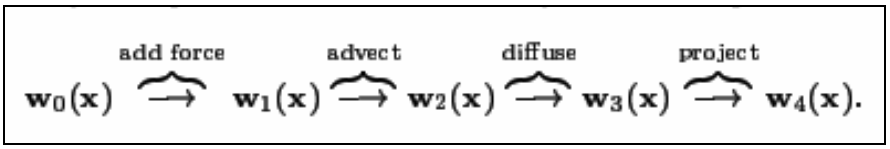
Pri postavljanju jednadžbi u odlomku 4.1., navedene su osnovne jednadžbe dinamike fluida. Nakon toga ponuđeno je pojednostavljenje rješavanja sustava jednadžbi rastavljanjem na dva koraka. To je osnovna ideja ove numeričke metode. U ovom poglavlju će se te jednadžbe raščlaniti na više koraka, pri čemu je prvi korak (4.4) rastavljen na još dva (tri u slučaju viskoznih fluida) koraka, a drugi korak (4.5) je u biti četvrti, projekcijski korak.

Rješavanje postavljenih jednadžbi se koristi metodom koju je uveo Stam u svom radu [2]. Kreće se od inicijalnog stanja za t=0:

$$\mathbf{u}_0 = \mathbf{u}(\mathbf{x},0) \tag{5.8}$$

iz kojeg se dobiva rješenje koje je inicijalno stanje za idući korak. I tako za svaki Δt do kraja simulacije.

Pretpostavlja se da je vektor brzine riješen za vremenski trenutak t, potrebno je pronaći iznos polja za idući vremenski trenutak $t+\Delta t$. Postupak kreće od rješenja $\mathbf{w_0}(\mathbf{x})=u(\mathbf{x},t)$ prijašnjeg koraka i tada se odvojeno rješava svaki izraz na desnoj strani jednadžbe (4.4). Na slici je naveden tok postupka kada bi se rješavale Navier-Stokesove jednadžbe. Za jednadžbe ovog problema (Eulerove jednadžbe) izbacujemo treći korak zbog zanemarivanja efekta viskoznosti. Zbog mogućnosti proširenja algoritma i na Navier-Stokesove jednadžbe, biti će opisan i taj korak.

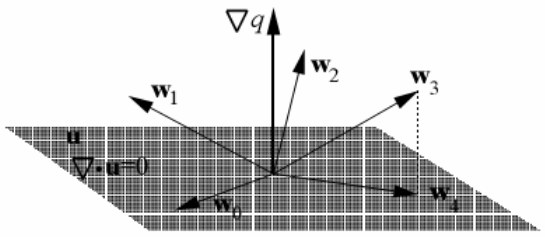


Slika 5.3. Koraci algoritma

Kao što se na slici vidi, algoritam rješavanja jednadžbi se rastavlja na 4 koraka. Konačno rješenje za vremenski trenutak $t+\Delta t$ je jednak rezultatu zadnjeg koraka:

$$\mathbf{u}(\mathbf{x},t+\Delta t)=\mathbf{w}_4(\mathbf{x})\,. \qquad (5.9)$$

Simulacija se odvija prolazeći ove korake u svakoj iteraciji. Slijedi detaljniji opis svakog od koraka.



Slika 5.4. Svaki korak simulacije je kompozicija koraka. Prva tri koraka mogu "odvesti" polje brzina izvan područja divergencije. Zadnji korak osigurava divergenciju polja nakon svakog koraka simulacije.

Postupak počine rješavanjem najlakše rješivog izraza iz jednadžbe očuvanja momenta (4.4), a to je korak dodavanja vanjske sile **f**. Uz pretpostavku da sila ne varira tijekom vremenskog koraka simulacije, vrijedi:

$$\mathbf{w_1}(\mathbf{x}) = \mathbf{w_0}(\mathbf{x}) + \Delta t\mathbf{f}(\mathbf{x}, t) \qquad (5.10)$$

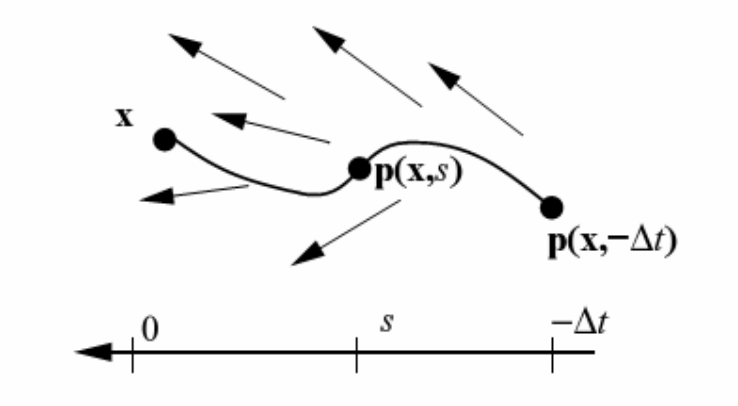
Navedeni izraz (5.10) je dobra aproksimacija sile za interaktivne sustave kroz vremenski korak Δt, jer se sile postavljaju samo na početku svakog vremenskog koraka. U tom koraku uzima se u obzir silu uzgona opisana jednadžbom (4.10), sila definirana metodom ograničavanja vrtložnosti (4.16), te bilo koja druga korisnički definirana sila. S obzirom da su navedene sile definirane u centrima ćelija, brzine se u ovom koraku ažuriraju kao što je navedeno u prijašnjem odlomku u jednadžbama (5.7).

Idući korak rješava efekt advekcije fluida na samog sebe. Ta pojava se opisuje izrazom $-(\mathbf{u}\cdot\nabla)\mathbf{u}$. Taj izraz čini Navier-Stokesove jednadžbe nelinearnima. Foster i Metaxas u svom radu za rješavanje tog izraza koriste algoritam konačnih razlika. Međutim takav pristup čini njihovu metodu nestabilnom i potrebno je dodatno ograničenje nad veličinom vremenskog koraka. Zbog toga, za male udaljenosti i velike brzine je potreban vrlo mali vremenski korak da bi postupak bio stabilan. Da bi izbjegli takav efekt, u ovoj metodi se koristi potpuno drugačiji pristup koji je uveo Stam [2]. Ta metoda se naziva semi-Lagrangeovom metodom, a temelji se na rješavanju parcijalno diferencijalnih jednadžbi metodom karakteristika. Detaljniji opis metode je već naveden u poglavlju 3.7. Ovdje će se navesti samo bitne karakteristike koje vode do rješenja.

Za svaki vremenski korak čestice fluida se kreću kroz prostor pod utjecajem brzine samog fluida. Zbog toga, da bi se došlo do vrijednosti brzine u točki $\mathbf{x}$ u novom vremenskom trenutku $t+\Delta t$, potrebno je pratiti unatrag točku $\mathbf{x}$ kroz polje brzina $\mathbf{w_1}$ kroz vrijeme Δt. To definira putanju $\mathbf{p}(\mathbf{x},s)$ koja odgovara djelomičnoj strujnici polja brzine fluida. Nova brzina u točki $\mathbf{x}$ je tada postavljena na brzinu koju je čestica, trenutno na poziciji $\mathbf{x}$, imala na svojoj prijašnjoj poziciji prije Δt vremena:

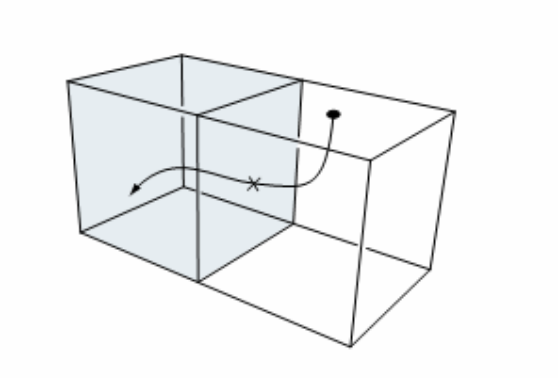
$$\mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \qquad (5.11)$$

Slika 5.5. ilustrira gore navedenu jednadžbu (5.11).



Slika 5.5. Propagiranje unazad po krivulji po kojoj se kreće tok fluida

Iz svega navedenog zaključuje se da semi-Lagrangeova metoda gradi novu mrežu praćenjem središnjih točka svake od stranica svih ćelija kroz polje toka brzine. Nove komponente brzina su tada interpolirane u tim točkama i njihove vrijednosti se upisuju na stranice ćelije odakle je krenuo postupak praćenja. Moguće je da točka praćenja završi u jednom od okupiranih ćelija, tada se jednostavno odbacuje ostatak krivulje koji je izašao izvan domene fluida (kao što je prikazano na slici 5.6).



Slika 5.6. Način rješavanja graničnih uvjeta: ako točka izlazi izvan mreže uzima se u obzir točka na granici

Ova metoda rješavanja nelinearnog izraza za horizontalno strujanje $-(\mathbf{u}\cdot\nabla)\mathbf{u}$, ima par prednosti. Najvažnija prednost je bezuvjetna stabilnost. Osim toga, iz gore navedene jednadžbe (5.11) vidljivo je da maksimalna vrijednost novog polja nikad nije veća od najveće vrijednosti prijašnjeg polja. Osim toga, ova metoda je vrlo jednostavna za implementiranje. Sve što zahtjeva u praksi je algoritam slijeđenja čestica i linearni interpolator. Jednostavna linearna interpolacija koja je potrebna za ovaj postupak lako se implementira, i u kombinaciji s metodom ograničavanja vrtložnosti daje zadovoljavajuće rezultate. Umjesto jednostavne linearne interpolacije moguće je koristiti i interpolacije viših razina, kao npr. kubična

interpolacija. Takve interpolacije daju bolji rezultat, međutim sa sobom donose i određene nedostatke što dovodi do nestabilnosti algoritma. Jednostavnost i stabilnost su dva važna svojstva su poželjna za svaki simulator fluida u računalnoj grafici. Na sličan način simuliramo i strujanje gustoće i temperature kroz fluid.

Treći korak rješava efekt viskoznosti. Taj efekt je zanemariv za plinovite fluide, ali će svejedno zbog potpunosti biti opisan. Efekt viskoznosti možemo predstaviti difuzijskom jednadžbom:

$$\frac{\partial \mathbf{w}_2}{\partial t} = \nu \nabla^2 \mathbf{w}_2 \tag{5.12}$$

Navedena jednadžba je standardna jednadžba za koju su razvijene brojne numeričke metode. Najjednostavniji način rješavanja ove jednadžbe je diskretizirati Laplaceov operator ∇^2 i tada primijeniti eksplicitni vremenski korak kao u radu Metaxasa i Fostera [3]. Međutim ta metoda je nestabilna za vrlo velike viskoznosti. Zbog toga se preferira uporaba implicitne metode:

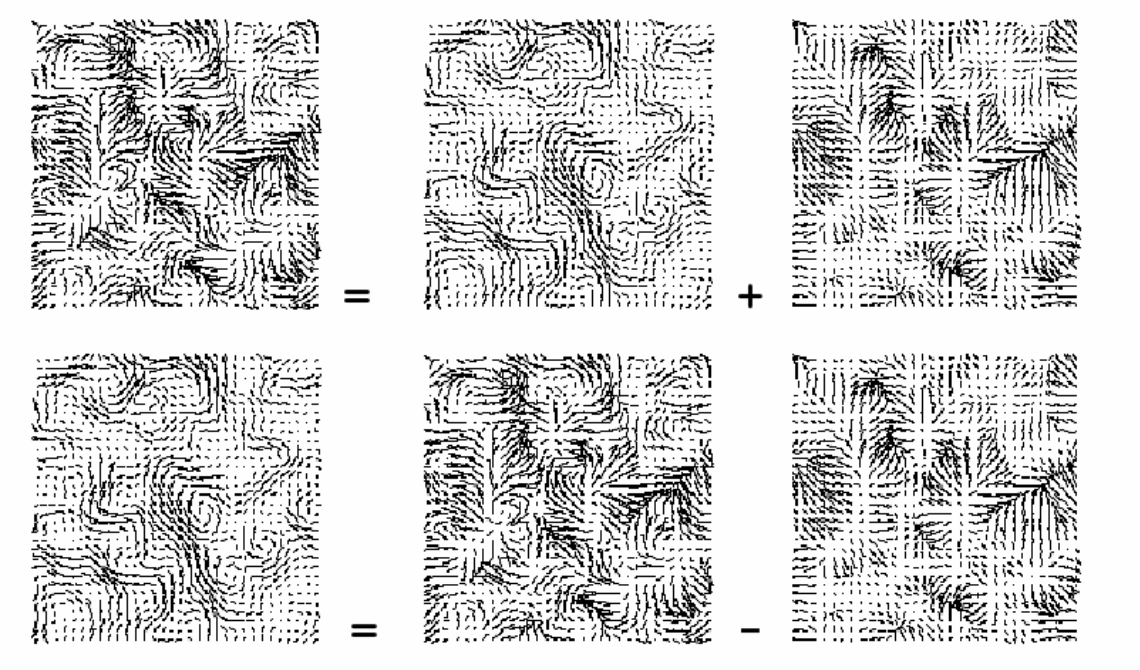
$$(\mathrm{I} - \nu \Delta t \nabla^2)\mathbf{w_3}(\mathbf{x}) = \mathbf{w_2}(\mathbf{x}) \tag{5.13}$$

gdje je I jedinična matrica ili operator identiteta. Kada se operator difuzije diskretizira, ova jednadžba se svodi na slabo popunjeni linearni sustav (engl. *sparse linear sistem*) za nepoznato polje $\mathbf{w_3}$. Rješavanje ovog sustava je jednostavno.

Četvrti korak uključuje korak projekcije, koji služi za dobivanje rezultantnog polja bez divergencije. Kao što je bilo navedeno u poglavlju gdje su postavljene jednadžbe toka, ovaj korak uključuje rješavanje Poissonove jednadžbe koja glasi:

$$\nabla^2 q = \nabla \cdot \mathbf{w}_3 \tag{5.14}$$

$$\mathbf{w}_4 = \mathbf{w}_3 - \nabla q \tag{5.15}$$



Slika 5.7. Svako polje se sastoji od polja bez divergencije i divergencije, da bi se dobilo polje bez divergencije jednostavno je potrebno oduzeti divergenciju

Dakle, korak projekcije zahtjeva dobar Poissonov solver. Zbog utjecaja na preciznost metode karakteristika kojom rješavamo izraz za advekciju, bitno je da polje bude što bliže polju bez divergencije. Još važnije, iz vizualnog gledišta, projekcijski korak prisiljava polje da sadrži vrtložne pojave što rezultira realnijem prikazu plinovitih fluida. Zbog tih uvjeta koristi se što točniji solver za ovaj korak.

Poissonova jednadžba, kada se diskretizira, postaje također slabo popunjeni linearni sustav jednadžbi koji se može riješiti bilo kojom metodom spomenutom u poglavlju 3.5. U ovom primjeru se koristi Gauss-Seidelova relaksacijska shema koja je također spomenuta u tom poglavlju. Ta metoda se koristi za rješavanje sustava od *N* dobivenih algebarskih jednadžbi koje možemo zapisati matrično:

$$\mathbf{Ax} = \mathbf{b} \tag{5.16}$$

Taj sustav se rješava jedanput po iteraciji algoritma, koristeći rezultate prethodnih koraka po formuli:

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{i \geq i} a_{ij} x_j^{(k-1)}}{a_{ii}} \tag{5.17}$$

Opisano matričnim računom, Gauss-Seidelova metoda se može izraziti kao:

$$x^{(k)} = (\mathbf{D} - \mathbf{L})^{-1}(\mathbf{U}x^{(k-1)} + \mathbf{b}), \tag{5.18}$$

gdje **D**, **L** i **U** su dijagonalna, donja trokutasta i gornja trokutasta matrica matrice **A**.

6. ALGORITAM

Do sada je postavljen problem i opisan je način rješavanja sa svim dodatnim korištenim metodama. Sada se može postaviti algoritam rješavanja problema toka plinovitih fluida.

Za rješavanje svih fizikalnih veličina, potrebne su dvije mreže ćelija. Simulacija se odvija ažuriranjem jedne mreže iz druge kroz fiksan vremenski korak Δt. Pri svakom koraku sadržaj dviju mreža se zamjenjuje. Na početku mreža sadrži inicijalno stanje zadano od korisnika.

6.1.KORACI ALGORITMA

Za svaki Δt od početka do kraja simulacije:

Za svaku ćeliju:

1.korak: **ažuriranje brzina**

- ažuriraj vrijednost vektora brzine pribrajanjem komponente sile pomnožene sa Δt, korak ponovi za svaku od definiranih sila (korisnička, uzgon, ograničavanje vrtložnosti)
- ažuriraj vrijednost vektora brzine rješavanjem izraza advekcije koristeći semi-Lagrangeovu metodu
- ažuriraj vrijednost vektora brzine projekcijskim korakom: rješavanjem Poissonove jednadžbe (32) i oduzimanjem rezultata od postojeće vrijednosti brzine

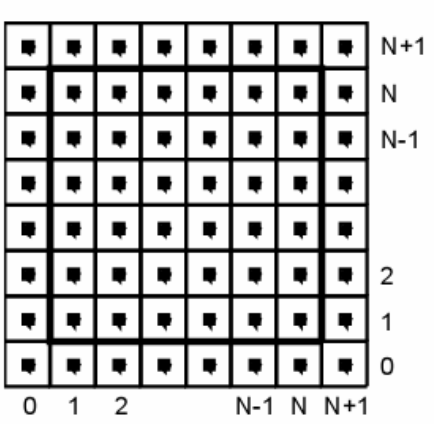
2.korak: **ažuriranje temperature i gustoće**

- ažuriraj vrijednost temperature fluida rješavajući izraz (11) semi-Lagrangeovom meotodom
- ažuriraj vrijednost gustoće fluida rješavajući izraz (12) semi-Lagrangeovom meotodom

6.2. OPIS PROGRAMA ZA NAVEDENI ALGORITAM

Program se temelji na ideji algoritma opisanoj u radu Stama, Jensena i Fedkiwa [1], u kojem se koriste dijelovi simulatora opisanog u radu Stam [2]. U programu se koriste neke od funkcija preuzete djelomično ili u potpunosti iz tog rada.

Simulacija je rađena u 2D prostoru na uniformnoj mreži. Proširenje na 3D područje se dobiva dodavanjem treće dimenzije i proširenjem prostornih derivacija tom komponentom. Fluid se modelira na kvadratnoj mreži kako je prikazano na slici 6.1. Vanjski sloj ćelija je granični sloj i ne sadrži fluid kao što je vidljivo na slici. Fluid se nalazi u ćelijama od indeksa 1 do N uključujući, dok se ćelije s indeksima 0 i N+1 tretiraju kao granice, te fluid izgleda kao zatvoren između četiri zida.



Slika 6.1. Fluid unutar graničnog sloja

Svaka ćelija se tretira kao da sadrži konstantni iznos gustoće, temperature (u centru ćelije) i brzine (na rubovima ćelije). Sve te veličine se predstavljaju poljima veličine

```
vel = (N+2)*(N+2):
```

```
static float * u, * v, * u_prev, * v_prev;
static float * dens, * Temp,  * dens_prev, * Temp_prev;
```

Uz alokaciju memorije:

```
u = (float *) malloc ( vel*sizeof(float) );
```

za ostala polja vrijedi analogno. U tim definicijama pokazivači `u` i `v` predstavljaju polja komponenti brzina za svaku ćeliju, a `dens` i `Temp`, polja gustoće i temperature. Pokazivači sa sufiksom `_prev` predstavljaju pomoćna polja, koja služe za prijenos definiranih vrijednosti iz glavnog programa u simulator dima. U simulatoru se ta polja tretiraju kao pomoćna mreža, pomoću koje se računa novo inicijalno stanje koje će biti vraćeno u glavni program pomoću glavnih pokazivača. Time je osigurano da u poljima na koje pokazuju `u, v, dens, Temp,` uvijek stoji zapisano trenutno "korisno" stanje. Koriste se jednostruki pokazivači zbog veće učinkovitosti, dok se elementi polja tj. pojedine ćelije dohvaćaju pomoću sljedeće definirane makro funkcije:

```
#define IX(i,j) ((i)+(N+2)*(j)).
```

Na primjer, vrijednost komponente brzine *u* za ćeliju s indeksima (*i , j*) dohvaćamo pomoću izraza `u[IX(i,j)]`. Isto tako se pretpostavlja da fizikalna duljina svake strane mreže je jedan tako da je onda veličina svake stranice ćelije zadana izrazom:

```
h = 1.0f/N;
```

Temeljna struktura simulatora dima je sljedeća. Kreće se od inicijalnog stanja polja brzina, gustoće i temperature. Pri svakom koraku algoritma se ažuriraju vrijednosti u mreži kao rezultat događaja u zadanom okruženju. U ovom konkretnom primjeru, početno stanje ima definiran jedan stalni izvor. To znači da su neke ćelije markirane stalnim iznosom sile i gustoće i to se kroz vrijeme ne mijenja. Osim toga definiran je i ponor. U tom slučaju određen broj ćelija ima nulti iznos gustoće i negativni iznos brzine. Pri negativnom iznosu brzine misli se na ili konstantnu brzinu usmjerenu "prema van", što bi u tom slučaju značilo da je to aktivni ponor (isisava dim), ili brzina mora biti definirana sa suprotnim predznakom brzine koju bi inače ta ćelija imala u tom koraku, u tom slučaju radi se o pasivnom ponoru. Osim navedenog izvora i ponora, moguće je tijekom simulacije pomoću tipki miša dodavati izvore gustoće i brzine u okruženje.

Algoritam se sastoji od dva osnovna koraka. Koraka ažuriranja gustoće i temperature, te koraka ažuriranja brzine. Prvo se obrađuje korak ažuriranja brzine, međutim zbog jednostavnijeg shvaćanja biti će prvo objašnjen korak ažuriranja gustoće i temperature.

Kao što je spomenuto, postoji jedan fiksni izvor i moguće je u svakom trenutku dodati još koji. To se u glavnom programu sprema u pomoćno polje koje se predaje potprogramu kao `dens0`. U glavnom programu se poziva funkcija koja je definirana u solveru:

```
void dens_step ( int N, float * dens, float * dens0, float * Temp, float *
Temp0, float * u, float * v, float dt ).
```

Ta funkcija predstavlja korak ažuriranja gustoće. Uz gustoću ovdje se ažurira i temperatura, međutim ona je manje bitan faktor u razvijanju osnovnog simulatora. Temperatura služi samo za dodatne efekte kao što je sila uzgona, dok je gustoća ono što se iscrtava na zaslon.

Vrijednosti gustoće izvora, koje su predane iz glavnog programa, nalaze se u polju `dens0` . Te vrijednosti se dodaju vrijednostima inicijalnog stanja (tj. stanja koje je simulator imao u prošlom koraku) koja se nalaze u polju `dens` tako da se pomnože sa jedinicom vremena.

```
for ( i=0 ; i<vel ; i++ ) dens[i] += dt*dens0[i];
```

Sada inicijalna mreža sadrži i inicijalno stanje iz prošlog koraka i iznose gustoća dodanih s izvorima u tom koraku. Nakon toga slijedi korak advekcije.

Prije samog koraka zamjenjuju se vrijednosti poljima `dens` i `dens0` jer se u tom koraku koristi inicijalno stanje za izračunavanje novog stanja (kao što je opisano u prijašnjim poglavljima, advekcija je širenje nekog svojstva u smjeru vektora brzine, u ovom slučaju gustoće):

```
zamjena (dens, dens0, N);
```

Sada je dosadašnje stanje spremljeno u polju i moguće je pomoću toga računati novo stanje za svaku ćeliju.

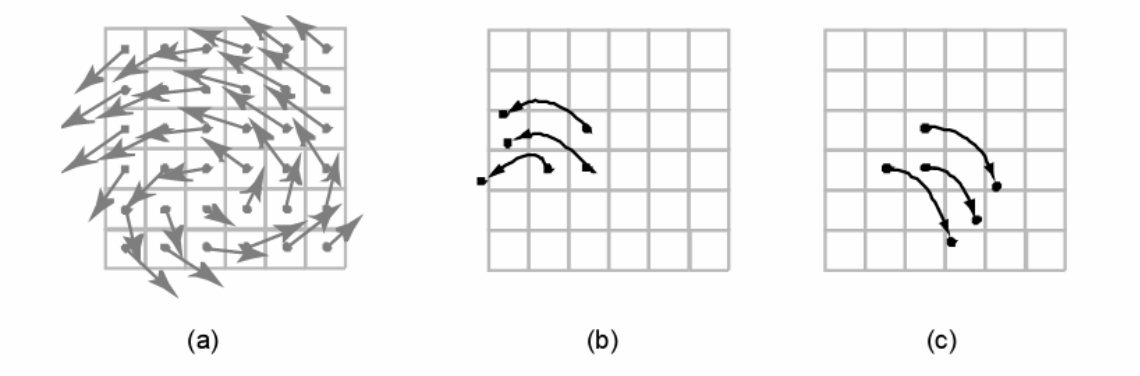
```
for ( i=1 ; i<=N ; i++ )
      {
            for ( j=1 ; j<=N ; j++ )
            {
```

Gustoća se nalazi u sredini ćelije, dok su brzine na stranicama. Zbog toga potrebno je prvo za svaku ćeliju izračunati iznos brzine u središtu:

```
usred=(u[IX(i,j)]+u[IX(i-1,j)])*0.5f;
vsred=(v[IX(i,j)]+v[IX(i,j-1)])*0.5f;
```

Nakon računanja srednje brzine, traži se pozicija središnje točke ćelije u prijašnjem trenutku, pomicanjem trenutne pozicije (*i, j*) po suprotnom smjeru brzine (u nazad):

```
x = i-dt*N*usred;
y = j-dt*N*vsred;
```



Slika 6.2. Korak advekcije: (a) tok fluida se može opisati vektorima koji prate strujnice, (b) korak algoritma u naprijed, (c) korak algoritma u nazad - Lagrangeova metoda

Time se dobiva točka (x, y) koja označava točnu poziciju središnje točke tekuće ćelije u prijašnjem trenutku. Pri ovom koraku bitno je paziti da vrijednosti x i y ne bi izašli izvan domene mreže, te se u tom slučaju moraju postaviti na sam rub:

```
if (x<0.5f) x=0.5f; if (x>N+0.5f) x=N+0.5f;
if (y<0.5f) y=0.5f; if (y>N+0.5f) y=N+0.5f;
```

Kad je sigurno da točka (x, y) leži unutar domene računanja, određuju se četiri susjedne ćelije koje će se koristiti za interpolaciju gustoće na traženoj poziciji:

```
i0=(int)x; i1=i0+1;
j0=(int)y; j1=j0+1;
```

Kada su određeni indeksi ćelija između kojih će se vršiti interpolacija gustoće, potrebno je definirati parametre za linearnu interpolaciju:

```
s1 = x-i0; s0 = 1-s1; t1 = y-j0; t0 = 1-t1;
```

Nakon toga dobiva se nova vrijednost tako da joj se pridruži vrijednost na poziciji (x, y) iz prijašnjeg koraka koja se dobiva interpolacijom četiriju susjednih vrijednosti iz prošlog trenutka koje su najbliže dobivenoj poziciji:

```
dens[IX(i,j)] = s0*(t0*dens0[IX(i0,j0)]+t1*dens0[IX(i0,j1)])+
                    s1*(t0*dens0[IX(i1,j0)]+t1*dens0[IX(i1,j1)]);
Temp[IX(i,j)] = s0*(t0*Temp0[IX(i0,j0)]+t1*Temp0[IX(i0,j1)])+
                s1*(t0*Temp0[IX(i1,j0)]+t1*Temp0[IX(i1,j1)]);
```

Time završava korak advekcije gustoće i temperature nakon kojega novo stanje se nalazi u poljima `dens` i `Temp` koja se šalju nazad u glavni program, gdje se iscrtavaju na zaslon, te služe kao inicijalno stanje za idući korak.

Druga funkcija koja obilježava simulator je funkcija koraka ažuriranja brzine:

```
void vel_step ( int N, float * u, float * v, float * u0, float * v0,
                       float * dens, float * Temp, float Tamb, float alfa,
                       float beta, float dt ).
```

Isto kao i kod funkcije za ažuriranje gustoće, polja `u` i `v` sadrže stanje iz prošlog koraka koje se koristi kao inicijalno stanje u tekućem koraku. Polja `u0` i `v0` sadrže iznose sila izvora koja se prenose iz glavnog programa. Ostale varijable su konstante i polja koja sadrže podatke potrebne za izračun pojedinih koraka.

Funkcija za ažuriranje brzine se sastoji od 3 glavna koraka, kao što je navedeno u algoritmu u poglavlju prije. Prvi korak je dodavanje sila. U ovom primjeru koriste se tri različite vrste sila: sile definirane od korisnika u glavnom programu prenesene u poljima `u0` i `v0`, sila uzgona i sila ograničavanja vrtložnosti. Za silu ograničavanja vrtložnosti potrebno je prvo izračunati njezin iznos i položaj na mreži što je izvedeno u funkciji:

```
void vorticity_confinement(int N, float * u, float * v, float *fx, float
*fy, float epsilon, float dt);
```

Ta funkcija vraća iznose sila u poljima `fx` i `fy`. Funkcija traži pozicije u mreži gdje je potrebno "umetnuti" vrtloge malih razmjera kao što je opisano u poglavlju X.

Nakon što su poznate sve vrijednosti svih definiranih sila, one se dodaju u polja brzina na isti način kao što su se dodavali izvori u polje gustoće, uz jednu malu razliku: sile se definiraju u središtima ćelija, te je zbog toga potrebno brzine ažurirati s njihovom srednjom aritmetičkom vrijednosti.

Prije drugog koraka vrši se jedan međukorak. Naime, korak advekcije brzine je točniji ako je polje brzine bez divergencije. Zbog toga se korak projekcije koji uklanja divergenciju primjenjuje jedanput prije advekcije. Korak projekcije se sastoji od računanja divergencije za svaku ćeliju pozivom funkcije:

```
void divergencija (int N, float * div, float * p, float * u, float * v);
```

U toj funkciji se računa polje divergencija `div` po formuli:

```
div[IX(i,j)] = -0.5f*(u[IX(i+1,j)]-u[IX(i-1,j)]+
                      v[IX(i,j+1)]-v[IX(i,j-1)])/N;
```

Kada su poznate sve divergencije poziva se funkcija za rješavanje sustava linearnih jednadžbi, koja je u potpunosti preuzeta iz rada Stam [X]:

```
void lin_solve ( int N, int b, float * x, float * x0, float a, float c )
```

U toj funkciji se koristi Gauss-Seidel relaksacijska shema za rješavanje sustava linearnih jednadžbi kao što je pojašnjeno u postavljanju problema. Implementacija tog algoritma je jednostavna i sastoji se od 3 petlje. Varijabla *k* je broj iteracija algoritma.

```
for ( k=0 ; k<20 ; k++ ) {
            for ( i=1 ; i<=N ; i++ )
            {
                  for ( j=1 ; j<=N ; j++ )
                  {

                        x[IX(i,j)]=(x0[IX(i,j)] +
                                    a*(x[IX(i1,j)]+x[IX(i+1,j)]+
                                    x[IX(i,j-1)]+x[IX(i,j+1)]))/c;
                  }
            }
```

Nakon rješavanja sustava linearnog sustava pozivanjem gornje funkcije, u polju `x` se nalaze spremljena rješenja sustava koja se u ovom slučaju predaju nazad u program kao polja tlakova. Tada se pomoću funkcije:

```
void oduzmi_gradijent_tlaka(int N, float * u, float * v, float * p)
```

od dotadašnjeg stanja u poljima brzina `u` i `v` za svaku ćeliju oduzima gradijent tlaka:

```
u[IX(i,j)] -= N*(p[IX(i+1,j)]-p[IX(i,j)]);
v[IX(i,j)] -= N*(p[IX(i,j+1)]-p[IX(i,j)]);
```

S time je međukorak projekcije gotov i polja brzina su bez divergencije.

Nakon toga slijedi korak advekcije brzine. On je u principu isti kao korak advekcije gustoće sa malom preinakom: s obzirom da se brzine nalaze na stranicama ćelija, potrebno je računati nove pozicije dviju različitih točaka (poziciju vektora brzine *u* koji se nalazi na polovini desne stranice, i poziciju vektora brzine *v* koji se nalazi na polovini gornje stranice ćelije):

```
x = i-dt*N*u0[IX(i,j)];
y = j+1/2-dt*N*v0[IX(i,j)];

x1 = i+1/2-dt*N*u0[IX(i,j)];
y1 = j-dt*N*v0[IX(i,j)];
```

Time su dobivene dvije nove pozicije: (x, y) kao pozicija točke vektora brzine *u* u prijašnjem vremenskom trenutku, te točka (x1, y1) kao pozicija točke vektora brzine *v* u prijašnjem vremenskom trenutku. Daljnji postupak je analogan advekciji gustoće. Pazi se da novo dobivene točke ne izlaze izvan mreže i za obje točke se ponavlja postupak nalaženja se četiri susjedne ćelije između kojih se radi interpolacija po vrijednostima brzina. Na kraju postupka, u poljima `u` i `v` se nalaze trenutna stanja brzina. Dobivena polja opet sadrže divergenciju te je potrebno ponoviti korak projekcije.

Novi korak projekcije, koji je zadnji korak ažuriranja brzine je identičan predkoraku za advekciju:

```
divergencija( N, div, p, u, v );
lin_solve ( N, 0, p, div, 1, 4 );
oduzmi_gradijent_tlaka( N, u, v, p);
```

Na kraju ovog postupka u poljima `u` i `v` se nalaze brzine koje se vraćaju u glavni program. U glavnom programu se pomoću tih polja iscrtavaju vrijednosti na zaslon te se nakon toga koriste kao novo inicijalno stanje za idući vremenski korak.

U glavnom programu poziva se simulator dima i vrši iscrtavanje na isti način kao u Stamovom simulatoru. Simulator se poziva u posebnoj GLUT funkciji *idle* koja se izvršava dok nema drugih aktivnosti:

```
static void idle_func ( void )
{
      get_from_UI ( dens_prev, u_prev, v_prev );
      vel_step ( N, u, v, u_prev, v_prev, dens, Temp, 24, alfa, beta, dt );
      dens_step ( N, dens, dens_prev, Temp, Temp_prev, u, v, dt );

      glutSetWindow ( win_id );
      glutPostRedisplay ();
}
```

U toj funkciji se prvo poziva rutina za interakciju sa korisnikom gdje se prikupljaju novi podaci o trenutnim izvorima i promjenama u sustavu. Nakon toga se pozivaju opisane funkcije i to prvo funkcija za ažuriranje brzine, te onda funkcija ažuriranja gustoće. Nakon svakog takvog koraka vrši se iscrtavanje slike.

Iscrtavanje se vrši funkcijom:

```
static void display_func ( void )
{
      pre_display ();

            if ( dvel ) draw_velocity ();
            else        draw_density ();

      post_display ();
}
```

Iz funkcije je vidljivo da je moguće vršiti dva načina iscrtavanja koje korisnik može mijenjati pomoću varijable `dvel` koju je moguće promijeniti tijekom izvršavanja programa tipkom "v". S obzirom na vrijednost varijable poziva se funkcija za iscrtavanje gustoće ili funkcija za iscrtavanje vektora brzina. Pri iscrtavanju gustoće koristi se funkcija:

```
static void draw_density ( void ),
```

u kojoj se pomoću indeksa ćelija određuje pozicija centra za svaku ćeliju (gdje je definirana gustoća):

```
      x = (i-0.5f)*h;
      y = (j-0.5f)*h;
```

Pomoću GLUT rutine:

```
      glBegin ( GL_QUADS );
```

na zaslon se iscrtavaju pravokutnici između središta četiriju susjednih ćelija s međusobnom interpolacijom po boji, tj. za svaku središnju točku definira se ekvivalent boje prema vrijednosti gustoće u toj točki:

```
d00 = dens[IX(i,j)];
d01 = dens[IX(i,j+1)];
d10 = dens[IX(i+1,j)];
d11 = dens[IX(i+1,j+1)];
```

Nakon čega se definiraju pripadajuće točke između kojih se vrši interpolacija

```
glColor3f ( d00, 0, 0 ); glVertex2f ( x, y );
glColor3f ( d10, 0, 0 ); glVertex2f ( x+h, y );
glColor3f ( d11, 0, 0 ); glVertex2f ( x+h, y+h );
glColor3f ( d01, 0, 0 ); glVertex2f ( x, y+h );
```

Veličina koja predstavlja gustoću je ograničena razlučivosti boje. S obzirom da je definicija GLUT funkcije:

```
void glColor3f (GLfloat red,GLfloat green, GLfloat blue );
```

najveća moguća vrijednost je ograničena sa tipom `GLfloat`, što je ekvivalentno float tipu u C-u (32-bitni realni broj). Vrijednost se mapira na interval vrijednosti [0, 1] prije interpolacije ili upisivanja u spremnik boje (engl. *color buffer*).

Funkcija za iscrtavanje vektora brzina:

```
static void draw_velocity ( void )
```

iscrtava vektore brzina kao linije uz pomoć GLUT rutine:

```
glBegin ( GL_LINES );
```

Brzinu crtamo iz donjeg desnog kuta ćelije kao liniju koja predstavlja zbroj vektora brzine:

```
glVertex2f ( x, y );
glVertex2f ( x+u[IX(i,j)], y+v[IX(i,j)] );
```

Program se izvršava u beskonačnoj `glutMainLoop ()` petlji sve dok ga korisnik ne prekine pomoću tipke "q".

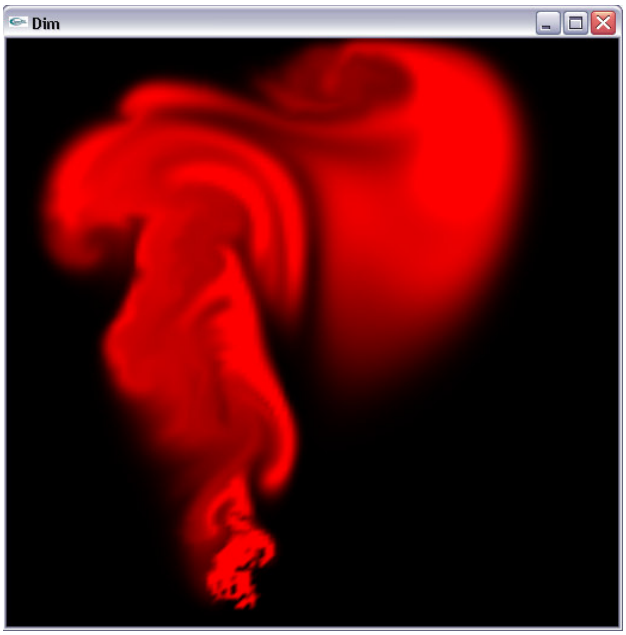
7. REZULTATI

Implementirani algoritam daje lijepe i realistične prikaze dima u realnom vremenu. Moguće je mijenjati razne parametre te tako utjecati na izgled i fizikalne osobine dima. Pri pokretanju programa moguće je zadati neke od parametara ili se koriste predviđene vrijednosti. Korišteni parametri i njihove inicijalne vrijednosti navedene su u tablici 1, gdje su N broj ćelija u jednom smjeru, dt vremenski korak, alfa i beta konstante korištene za silu uzgona, a force i source konstante korištene za skaliranje ulaznih podataka pri interakciji s mišem. Osim navedenih parametara moguće je dodavati i izvore i ponore te različite preturbacije. U nastavku poglavlja dani su prikazi nekih od rezultata uz različite parametre.

PARAMETAR	VRIJEDNOST
N	128
dt	0,1
alfa	0
beta	0
force	10
source	100

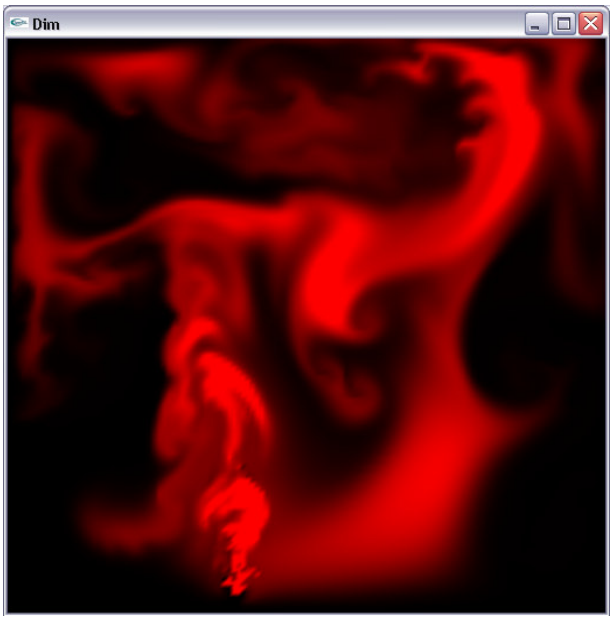
Tablica 1. Inicijalne vrijednosti korištene u programu

Slika 7.1 prikazuje rezultat simulacije za jedan izvor uz inicijalne vrijednosti svih parametar



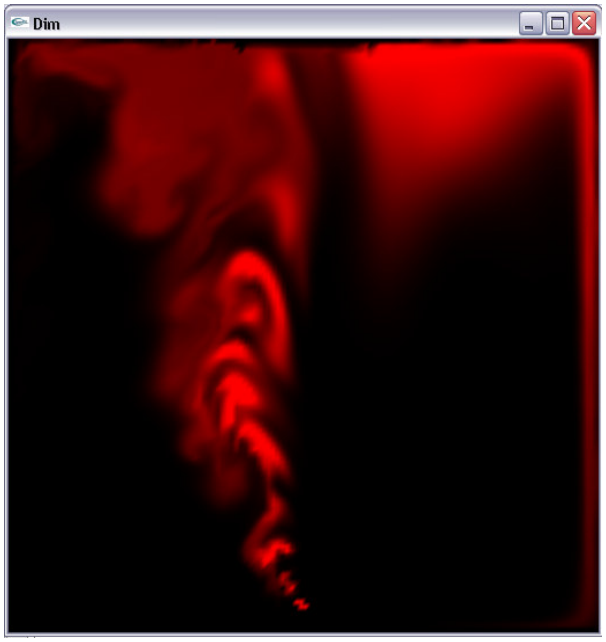
Slika 7.1. Inicijalne vrijednosti uz jedan izvor

Dodavanjem sila desnom tipkom miša, te dodavanjem dodatnih izvora gustoće lijevom tipkom miša, postižu se razne preturbacije dima kao na slici 7.2.



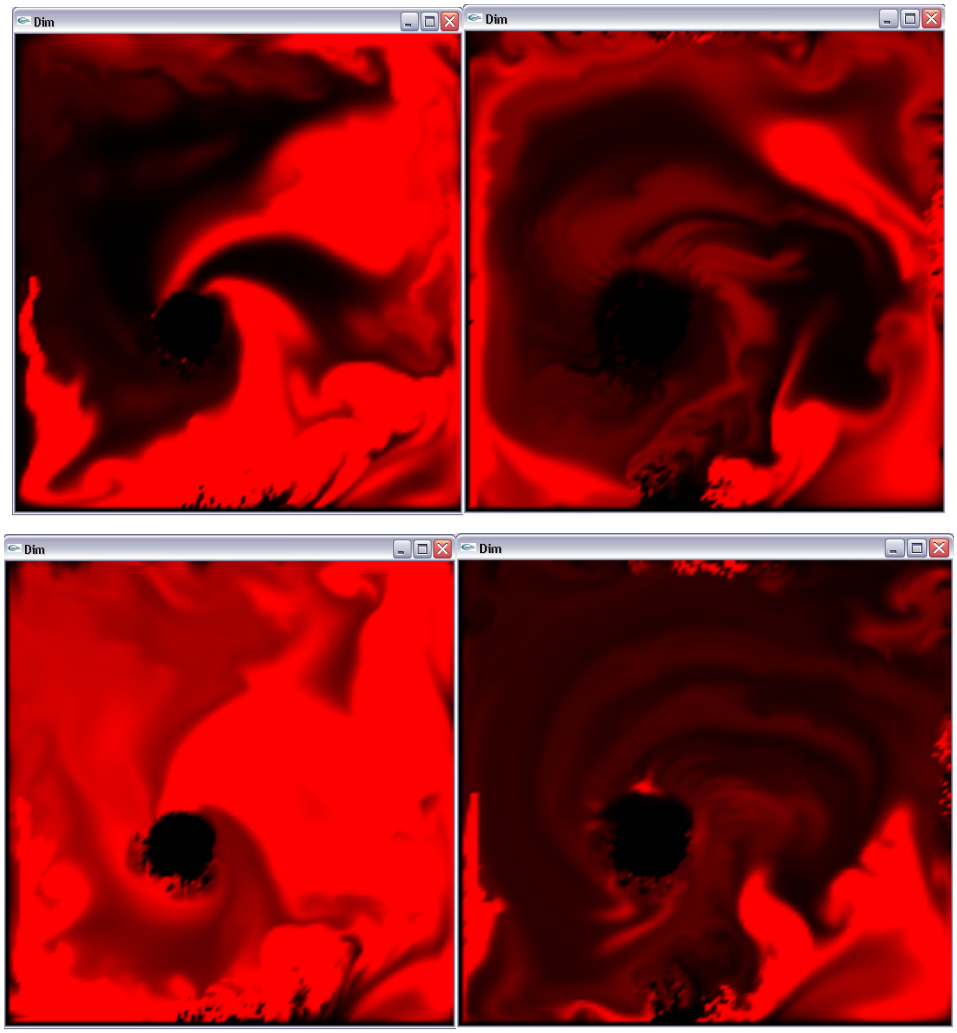
Slika 7.2. Jedan izvor uz inicijalne vrijednosti parametra uz dodavanje sila i izvora interakcijom korisnika pomoću miša

Osim izvora u sceni se mogu definirati i ponori. Na slici 7.3 definiran je tako jedan aktivni ponor na sredini gornjeg ruba, na slici je vidljivo kako ponor "izvlači" dim iz zatvorenog prostora.



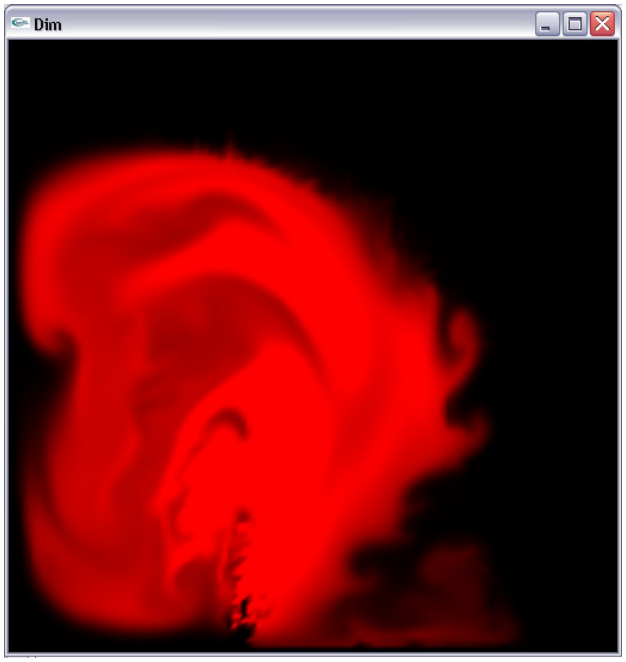
Slika 7.3. Jedan izvor i jedan ponor uz inicijalne vrijednosti parametara

Osim tako definiranog ponora, moguće je definirati ponor kao "crnu rupu" koja uvlači dim oko sebe kao na slici 7.4.



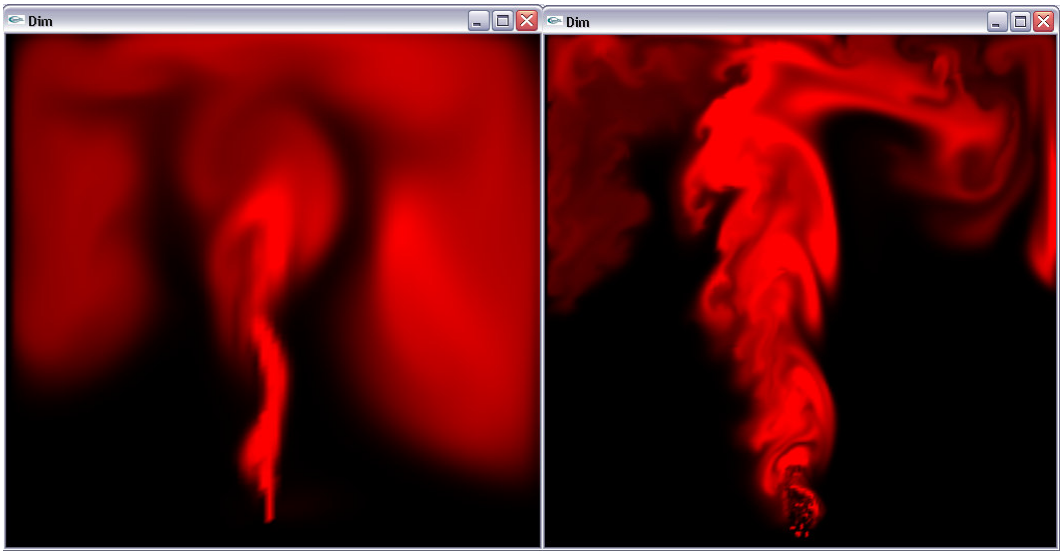
Slika 7.4. Ponor je definiran kao crna rupa koja uvlaci okolni dim u sebe

Ako se želi simulirati silu uzgona moguće je mijenjati parametre *alfa* i *beta*. Tako za dobivamo utjecaj da je gušći dim teži kao na slici 7.5.



Slika 7.5. Vrijednost parametra *alfa* 0.1

Implemenitrani algoritam radi u realnom vremenu zbog korištenja malog broja ćelija. Međutim unatoč grubljoj mreži dim i dalje izgleda realan. Na slici 7.6 može se vidjeti kako izgleda dim na grubljim i finijim mrežama.



a) b)

7.6. Prikaz dima na grubljim mrežama uz N=80 (a) i na finijim mrežama uz N=200 (b)