Symulacja rozprzestrzeniania się dymu w przestrzeni 3D

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej

Kacper Tonia Sławomir Kalandyk Mikołaj Sieńko

I sprawię dziwy na górze - na niebie,
i znaki na dole - na ziemi.
Krew i ogień, i kłęby dymu

Dzieje Apostolskie 2, 19

1 Wstęp

Przepływ powietrza wokół samolotu ponaddźwiękowego. Fala uderzeniowa wywołana eksplozją. Cyrkulacja krwi w układzie krążenia. Wszystkie te zjawiska łączy obecność płynów: cieczy i gazów. Niniejsza praca jest skromną próbą poznania, choćby powierzchownego, nauki poświęconej symulowaniu wyżej wspomnianych – i wielu innych – zjawisk: obliczeniowej mechaniki płynów (CFD – *Computational Fluid Dynamics*).

Głównym zadaniem projektu było stworzenie trójwymiarowego modelu rozprzestrzeniania się dymu w pomieszczeniu. Model uwzględnia takie czynniki jak temperaturę czy kierunek ruchu powietrza. W tej pracy omówiliśmy zastosowany model oraz przedstawiliśmy wyniki symulacji 3D dymu.

Na początku prac nad projektem rozważane były dwa podejścia do symulacji CFD:

- Równania Naviera-Stokesa
- Metody siatkowe Boltzmanna (*Lattice Boltzmann Methods,* LBM)

Równania Naviera-Stokesa są równaniami różniczkowymi cząstkowymi drugiego rzędu. Operują na polach prędkości i ciśnień, badając ich ewolucję w czasie. Z kolei LBM wykorzystuje automaty komórkowe, a istotną rolę odgrywają rozkłady prawdopodobieństwa występowania cząstek. LBM bezpośrednio symuluje przepływ płynu w skali mikro, który w skali makro dobrze odwzorowuje wyniki uzyskane z równań Naviera-Stokesa. Możliwe jest także efektywne zrównoleglanie obliczeń LBM, co czyni ową metodę poważnym kandydatem w przypadku np. złożonych naukowych symulacji [1].

Obie metody mają swoich zwolenników. Równania Naviera-Stokesa stanowią bardziej dojrzały, dokładniej zgłębiony dział mechaniki płynów, co w połączeniu z ich bardziej intuicyjną interpretacją sprawiło, że zdecydowaliśmy się zrezygnować z metody LBM w tej pracy.

2 Teoria

2.1 Równania Naviera-Stokesa

Równania Naviera-Stokesa pozwalają, w ogólności, zamodelować szeroki wachlarz zjawisk fizycznych. Poczynimy kilka założeń w celu uproszczenia badanego zagadnienia. Potraktujemy powietrze jako nieściśliwy płyn niutonowski. Nieściśliwość płynu oznacza, iż jego gęstość pozostaje stała w całej przestrzeni,

co jest słusznym przybliżeniem przy niewielkich prędkościach przepływu, tj. gdy liczba Macha płynu jest mniejsza niż 0.3 ([2]).

Wyprowadzenie równań Naviera-Stokesa wykracza poza materiał tego sprawozdania; znacznie bardziej wyczerpujące opracowanie tematu można znaleźć m.in. w [3]. Równania Naviera-Stokesa dla nieściśliwego płynu niutonowskiego mają następującą postać:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\,\mathbf{u} - \frac{1}{d}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{1a}$$

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

gdzie $\mathbf{u}$ – prędkość płynu, d – gęstość płynu, p – ciśnienie, ν – lepkość kinematyczna, $\mathbf{f}$ – wypadkowa sił zewnętrznych działających na płyn.

Równanie 1a jest w zasadzie przekształconą drugą zasadą dynamiki Newtona dla płynów i stanowi równanie pędu. Pozwoli nam na obliczanie prędkości płynu w kolejnych krokach symulacji. Człony równania 1a można opisać następująco:

$\frac{\partial \mathbf{u}}{\partial t}$ **pochodna prędkości po czasie.**

$-(\mathbf{u} \cdot \nabla)\,\mathbf{u}$ **człon adwekcyjny** (konwekcyjny). Jest odpowiedzialny za przemieszczanie się prędkości płynu w przestrzeni z upływem czasu.

$-\frac{1}{d}\nabla p$ **człon ciśnieniowy.** Reprezentuje siły występujące wewnątrz płynu na skutek różnicy ciśnień między różnymi obszarami płynu. Można powiedzieć, że ów człon odpowiada za „spychanie" materii z obszarów wysokiego ciśnienia ku obszarom o niższym ciśnieniu.

$\nu \nabla^2 \mathbf{u}$ **Człon lepkościowy.** Reprezentuje występujące wewnątrz płynu siły tarcia, zachodzące między obszarami o różnych prędkościach. Znaczenie tych sił determinuje lepkość kinematyczna płynu, oznaczona przez ν. W praktyce gazy cechuje na tyle niska lepkość, że niektóre prace nt. symulacji dymu (na przykład [4]) pomijają ten człon.

$\mathbf{f}$ **Siły zewnętrzne.** Wypadkowa sił działających na płyn, ale nie mających źródła w samym płynie. Siły zewnętrzne uwzględnione w naszym modelu omówimy szerzej w podrozdziale 3.2.

Równanie 1b stanowi warunek nieściśliwości – wymusza, by ilość płynu wpływającego do danego obszaru przestrzeni była równa ilości płynu wypływającego. Równanie to będzie potrzebne przy ustalaniu zmian ciśnienia tak, by dywergencja płynu w całej przestrzeni była (ze względu na niedoskonały charakter metod numerycznych) możliwie bliska zeru.

Jak łatwo zauważyć, p nie występuje bezpośrednio w równaniu 1b. Jest to jedna z podstawowych trudności pojawiających się przy rozwiązywaniu równań Naviera-Stokesa dla płynu nieściśliwego: brak bezpośredniej metody na wyznaczenie p w kolejnym kroku czasowym. Uzyskanie ograniczenia na p otrzymamy poprzez przekształcenie wzoru 1b do równania Poissona, o postaci $\nabla^2 p = b$. Należy zaznaczyć, że rozwiązanie równania Poissona jest nietrywialne i zajmuje większą część mocy obliczeniowej potrzebnej na przeprowadzenie symulacji CFD. Wrócimy do tego problemu w podrozdziale 3.5.

2.2 Dyskretyzacja równań Naviera-Stokesa

Niezdolność komputerów do przechowywania informacji o płynie w każdym możliwym punkcie przestrzeni pociąga za sobą konieczność dyskretyzacji równań Naviera-Stokesa. W naszym modelu będziemy reprezentować badaną przestrzeń za pomocą regularnej siatki sześciennej, w której określonych punktach będą przechowywane parametry płynu. Zachodzi zatem $\Delta x = \Delta y = \Delta z$. Od tej pory będziemy zapisywać wszystkie te trzy parametry jako Δx (tudzież `dx` w kodzie programu).

Należy wybrać metodę rozwiązywania równań różniczkowych cząstkowych, którymi są bez wątpienia równania Naviera-Stokesa. Istnieje wiele sposobów rozwiązywania, przy czym warto wymienić trzy naj-

popularniejsze metody: różnic skończonych (*finite difference method*, FDM), objętości skończonych (*finite volume method*, FVM) i elementów skończonych (*finite element method*, FEM).

Metoda różnic skończonych polega na zastąpieniu pochodnych cząstkowych różnicami skończonymi. Jej zaletą jest prostota zrozumienia oraz implementacji, o ile geometria badanej przestrzeni jest stosunkowo prosta, a tak jest w naszym modelu.

Podczas gdy FDM przeprowadza obliczenia w węzłach siatki, metoda objętości skończonych operuje na poszczególnych objętościach, tudzież komórkach siatki. Metoda ta ma pewną przewagę nad FDM pod względem stabilności.

Metoda elementów skończonych ma wiele zalet, które ujawniają się dopiero przy złożonych geometriach, a ponadto jest podejściem czysto matematycznym, któremu ciężko przyporządkować interpretację fizyczną. [5] Ostatecznie więc zdecydowaliśmy się zastosować metodę różnic skończonych.

Będziemy prowadzić rozważania w trzech wymiarach. Niech wektor prędkości $\mathbf{u} = (u, v, w)$. Równanie pędu 1a po rozwinięciu ma następującą postać:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - w\frac{\partial u}{\partial z} - \frac{1}{d}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + f_x \tag{2a}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - w\frac{\partial v}{\partial z} - \frac{1}{d}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + f_y \tag{2b}$$

$$\frac{\partial w}{\partial t} = -u\frac{\partial w}{\partial x} - v\frac{\partial w}{\partial y} - w\frac{\partial w}{\partial z} - \frac{1}{d}\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + f_z \tag{2c}$$

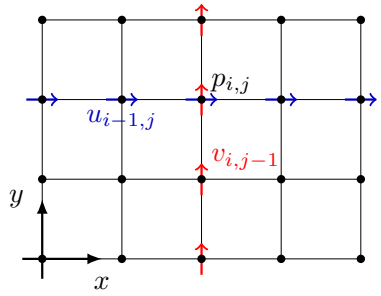
Z kolei równanie 1b:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{3}$$

2.3 Siatka obliczeniowa

> There is more to this than one might think.
>
> Ferziger, Perić. *Computational Methods for Fluid Dynamics*

Utworzenie poprawnej siatki obliczeniowej jest znacznie bardziej złożone niż może się na pierwszy rzut oka wydawać, co trafnie podsumowuje powyższy cytat. Wykorzystujemy siatkę typu *colocated*, czyli taką, w której wszystkie parametry płynu są przechowywane w tych samych punktach przestrzeni. W istocie, wielu autorów preferuje rozkład typu *staggered*, w którym informacje o prędkościach płynu przechowywane są na ściankach komórek siatki, zaś takie wielkości jak ciśnienie czy gęstość – w centrach komórek. Rozdzielenie tych wielkości w przestrzeni poprawia stabilność symulacji, jednak komplikuje równania oraz inicjalizację samej siatki.



Rysunek 1: Siatka 2D typu *colocated*

Wprowadzenie siatki typu *staggered* do naszego modelu wydaje się być rozsądnym krokiem na drodze ulepszania jakości symulacji, jednak obecne wyniki uważamy za zadowalające.

Prócz rozkładu parametrów dymu na siatce, możemy rozważyć zastosowanie tzw. siatki dynamicznej, która polega na przechowywaniu tylko tych węzłów przestrzeni, które zawierają dym bądź leżą w bliskim otoczeniu węzłów zawierających dym. Siatka dynamiczna pozwala zignorować nieistotne z punktu widzenia symulacji dymu komórki i wyraźnie zmniejszyć zużycie pamięci [6]. W celu uproszczenia implementacji decydujemy się jednak na siatkę statyczną, pokrywającą całą badaną przestrzeń, inicjalizowaną w całości tuż po uruchomieniu programu.

3 Algorytmy symulacji

3.1 Symulacja dymu

Nasz model uwzględnia następujące parametry fizyczne: prędkość przepływu płynu $\mathbf{u}$, ciśnienie płynu p, gęstość dymu d_{smoke} (nie mylić z gęstością płynu) oraz temperaturę płynu T. Prócz równań 1a i 1b, które rządzą dynamiką płynu (powietrza), wprowadzamy dodatkowo dwa równania adwekcji

$$\frac{\partial d}{\partial t} = -\left(\mathbf{u} \cdot \nabla\right) d \tag{4}$$

$$\frac{\partial T}{\partial t} = -\left(\mathbf{u} \cdot \nabla\right) T \tag{5}$$

odpowiedzialne za ruch dymu (i jego temperatury) zgodnie z polem prędkości. Gdyby nie równanie 4, dym nie opuściłby swojego źródła; z kolei bez równania 5 każdy węzeł siatki utrzymywałby cały czas tę samą temperaturę.

Wiedząc, jakie równania rządzą naszym modelem, możemy przedstawić ogólny algorytm symulacji:

Listing 1: Pętla główna symulacji

```
dla każdego kroku symulacji:
    dodaj źródła
    oblicz nową prędkość
    oblicz nową gęstość
    wyświetl
```

„Dodaj źródła” polega na wprowadzeniu do obszaru symulacji nowej porcji dymu. W praktyce oznacza to, że w określonych węzłach siatki ustawiamy „na sztywno” gęstość dymu, prędkość i temperaturę.

„Oblicz nową prędkość” jest realizacją równań 1a i 1b. Zgodnie z tzw. *Chorin projection method* ([7]), obliczane są prędkości i ciśnienia płynu w kolejnym kroku czasowym.

Listing 2: Algorytmiczna realizacja kroku prędkości

```
oblicz nową prędkość:
    dodaj siły zewnętrzne
    zamień miejscami pola prędkości i tymczasowe pola prędkości
    dyfuzja
    rzutowanie
    zamień miejscami pola prędkości i tymczasowe pola prędkości
    adwekcja
    rzutowanie
```

„Oblicz nową gęstość” realizuje równania 4 oraz 5 – przemieszcza dym zgodnie z rozkładem prędkości przepływu w płynie. Dodatkowo, we wszystkich węzłach obliczeniowych zmniejszamy gęstość dymu o pewną małą wartość, określoną parametrem `density_decay`.

Listing 3: Algorytmiczna realizacja kroku gęstości

```
oblicz nową gęstość:
    dyfuzja gęstości
    dyfuzja temperatury
    dla każdego węzła obliczeniowego:
```

```
gęstość = gęstość − density_decay
jeśli gęstość < 0:
    gęstość = 0
```

3.2 Siły zewnętrzne

3.2.1 Siła wyporu

Podstawowymi siłami działającymi na dym jest ciężar oraz siła wyporu. Przyjmujemy, że ciężki (gęsty) dym opada, zaś gorący dym unosi się. Opisuje to następujący prosty wzór (na podstawie [4]):

$$f_{i,j,k} = \Delta t(k_{\text{rise}} T_{i,j,k} - k_{\text{fall}} d_{\text{smoke}}); \tag{6}$$

W ogólnym przypadku człon $k_{\text{rise}} T_{i,j,k}$ powinien zawierać różnicę temperatur między dymem a otoczeniem. Dla uproszczenia przyjęliśmy arbitralnie temperaturę otoczenia jako zero.

Należy zaznaczyć, że wymienione siły działają wyłącznie w pionie. $f_{i,j,k}$ wpłynie bezpośrednio wyłącznie na składową pionową prędkości płynu $v_{i,j,k}$.

3.2.2 Wiatr

$$\mathbf{u}^{(n+1)} = (1 - k_{\text{wind}} \Delta t)\mathbf{u}^{(n)} + k_{\text{wind}} \Delta t \, \mathbf{u}_{\text{wind}} \tag{7}$$

Nasz symulator oferuje prosty model wiatru zapożyczony z [6], który można uwzględnić podczas obliczania zmian prędkości dymu. Kluczowy jest tutaj parametr k_{wind}, charakteryzujący, jak szybko prędkość płynu dąży do prędkości wiatru.

3.2.3 *Vorticity confinement*

Problemem dobrze znanym obliczeniowej mechanice płynów jest fakt szybkiego zanikania wirowości płynu. Na siatce dyskretnej, szczególnie przy stosunkowo dużych krokach Δx i Δt, można zaobserwować, że generowany dym pozbawiony jest zawirowań. [4] oferuje rozwiązanie: dodanie siły odpowiedzialnej za „wstrzykiwanie" energii, a przez to wzmacnianie ruchu wirowego płynu.

Choć podana metoda zwiększa turbulentność, a przez to wizualną atrakcyjność dymu, nie ma charakteru naukowego, przez co jest pomijana w symulacjach naukowych. Umożliwiamy sterowanie siłą oddziaływania *vorticity confinement* poprzez parametr `vort_conf_coef`. Gdy `vort_conf_coef` jest równy zero, efekt jest wyłączony.

3.3 Adwekcja

Węzły siatki obliczeniowej są nieruchome. Aby wielkości fizyczne ulegały przemieszczeniu zgodnie z danym polem prędkości, należy zastosować algorytm adwekcji.

Niech φ oznacza pole skalarne „unoszone" przez pole prędkości $\mathbf{u}$. Opisuje to równanie

$$\frac{\partial \varphi}{\partial t} = -\left(\mathbf{u} \cdot \nabla\right) \varphi \tag{8}$$

$$\frac{\partial \varphi}{\partial t} = -u\frac{\partial \varphi}{\partial x} - v\frac{\partial \varphi}{\partial y} - w\frac{\partial \varphi}{\partial z} \tag{9}$$

Jednym możliwym podejściem jest zastosowanie jawnej metody Eulera:

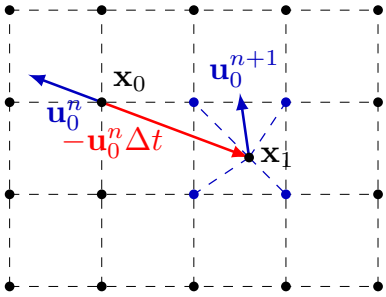
$$\frac{\varphi^{n+1}_{i,j,k} - \varphi^{n}_{i,j,k}}{\Delta t} = -u^{n}_{i,j,k}\frac{\varphi^{n}_{i+1,j,k} - \varphi^{n}_{i-1,j,k}}{2\Delta x} - v^{n}_{i,j,k}\frac{\varphi^{n}_{i,j+1,k} - \varphi^{n}_{i,j-1,k}}{2\Delta x} - w^{n}_{i,j,k}\frac{\varphi^{n}_{i,j,k+1} - \varphi^{n}_{i,j,k-1}}{2\Delta x}$$

$$\varphi^{n+1}_{i,j,k} = \varphi^{n}_{i,j,k} - \Delta t \left(u^{n}_{i,j,k}\frac{\varphi^{n}_{i+1,j,k} - \varphi^{n}_{i-1,j,k}}{2\Delta x} + v^{n}_{i,j,k}\frac{\varphi^{n}_{i,j+1,k} - \varphi^{n}_{i,j-1,k}}{2\Delta x} + w^{n}_{i,j,k}\frac{\varphi^{n}_{i,j,k+1} - \varphi^{n}_{i,j,k-1}}{2\Delta x} \right)$$

Niestety, taki algorytm jest niestabilny: poza wąskim zakresem parametrów, prędkości i ciśnienia płynu rosną w krótkim czasie do nieskończoności. Nie wszystkie zgodne, narzucające się schematy są użyteczne. Okazuje się, że zastosowanie tzw. metody Laxa pozwala uzyskać schemat stabilny [8]. Istnieje też inne podejście, zaproponowane przez Stama [9], wykorzystujące tzw. *Semi-Lagrangian scheme*.

Idea polega na tym, by wyobrazić sobie cząstkę płynu powiązaną z węzłem obliczeniowym, na pozycji $\mathbf{x}_0$. Znając prędkość $\mathbf{u}_0^n$ w pewnym kroku czasowym n, znajdujemy pozycję cząstki $\mathbf{x}_1$ w chwili $n-1$. Następnie, metodą interpolacji można wyliczyć prędkość płynu na podstawie prędkości z okolicznych węzłów. Tak obliczona prędkość zostaje przypisana węzłowi $\mathbf{x}_0$.

Metoda jest stabilna, bowiem prędkości w kolejnym kroku symulacji są uśrednionymi wartościami prędkości z kroku poprzedniego.



Rysunek 2: Idea *Semi-Lagrangian scheme*

3.4 Dyfuzja

Rozważmy równanie dyfuzji dla pewnej zmiennej skalarnej φ; w praktyce może to być którakolwiek ze współrzędnych prędkości płynu: u, v bądź w.

$$\frac{\partial \varphi}{\partial t} = \nu \nabla^2 \varphi = \nu \left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} \right) \tag{10}$$

Dokonując dyskretyzacji równania 10 zgodnie z metodą różnic skończonych [10], otrzymamy:

$$\varphi_{i,j,k}^{n+1} = \frac{1}{1+6\beta} \left[\varphi_{i,j,k}^{n} + \beta \left(\varphi_{i+1,j,k}^{n+1} + \varphi_{i-1,j,k}^{n+1} + \varphi_{i,j+1,k}^{n+1} + \varphi_{i,j-1,k}^{n+1} + \varphi_{i,j,k+1}^{n+1} + \varphi_{i,j,k-1}^{n+1} \right) \right]$$

przy czym

$$\beta = \frac{\nu \Delta t}{\Delta x^2}$$

Jest to metoda niejawna, którą rozwiążemy iteracyjnie. Zastosowanie metody jawnej byłoby mniej kosztowne obliczeniowo, ale groziłoby zmniejszeniem stabilności symulatora.

3.5 Rzutowanie

Wyżej wymienione w tym rozdziale kroki są kluczowe, by uwzględnić różne oddziaływania wpływające na ruch dymu. Samo obliczenie adwekcji czy dyfuzji prowadzi jednak do tego, że pole prędkości przepływu zyskuje niezerową dywergencję, co stoi w sprzeczności z warunkiem nieściśliwości (równanie 1b).

Aby spełnić warunek nieściśliwości, znajdziemy takie pole ciśnień płynu, by po uwzględnieniu członu $1/d \cdot \nabla p$ (który występuje w równaniu pędu 1a, a którego sposobu na obliczenie dotąd nie podaliśmy) pole prędkości posiadało zerową dywergencję.

Załóżmy, że uwzględniliśmy siły zewnętrzne oraz zjawiska adwekcji i dyfuzji. Uzyskujemy w ten sposób pewne pośrednie pole prędkości $\mathbf{u}^*$. Jak wspomnieliśmy w podrozdziale 2.1, warunek 1b można przekształcić, uzyskując równanie Poissona.

$$\nabla^2 p^{n+1} = \frac{d}{\Delta t} \nabla \cdot \mathbf{u}^* \tag{11}$$

Chcąc obliczyć ciśnienie w pewnym węźle (i, j, k) okaże się, że wartość $p_{i,j,k}$ jest bezpośrednio powiązana z ciśnieniami sąsiadujących węzłów. Powstaje układ liniowy o tylu równaniach, ile mamy węzłów obliczeniowych, i tyluż niewiadomych. Przykładowo dla pomieszczenia opisanego siatką $12 \times 12 \times 12$ węzłów, mamy $10 \cdot 10 \cdot 10 = 1000$ węzłów obliczeniowych. Możemy przedstawić taki układ równań poprzez rzadką macierz o wymiarach 1000×1000 i wykorzystać np. algorytm Gaussa-Seidla. W ten sposób problem rozwiązania równania Poissona zostaje sprowadzony do rozwiązania równania macierzowego $\mathbf{Ap} = \mathbf{b}$.

Metoda jest dość szeroko stosowana, jednak ze względu na kilka niedogodności (konieczność utworzenia dużej macierzy oraz zainicjalizowanie jej, z uwzględnieniem warunków brzegowych), postanowiliśmy zastosować nieco inne podejście. Rozwińmy równanie 11:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = \frac{d}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y} + \frac{\partial w^*}{\partial z}\right)$$

Po dyskretyzacji (dla przejrzystości pominęliśmy indeksy górne $n+1$ przy wszystkich p):

$$\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{\Delta x^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{\Delta x^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{\Delta x^2} = \frac{d}{\Delta t}\left(\nabla \cdot \mathbf{u}^*\right)_{i,j,k}$$

przy czym $\left(\nabla \cdot \mathbf{u}^*\right)_{i,j,k}$ oznacza dywergencję pola $\mathbf{u}^*$ w węźle (i, j, k). Ostatecznie, po przekształceniu ze względu na $p_{i,j,k}$:

$$p_{i,j,k} = \frac{1}{6}\left(p_{i+1,j,k} + p_{i-1,j,k} + p_{i,j+1,k} + p_{i,j-1,k} + p_{i,j,k+1} + p_{i,j,k-1} - \frac{d\Delta x^2}{\Delta t}\left(\nabla \cdot \mathbf{u}^*\right)_{i,j,k}\right)$$

W formie pseudokodu, algorytm rzutowania wygląda następująco:

Listing 4: Rzutowanie – pseudokod

```
rzutowanie:
    // Oblicz dywergencję pola prędkości pośrednich.
    dla każdego węzła obliczeniowego:
        diverg(i,j,k) = (
            u(i+1,j,k) - u(i-1,j,k) +
            v(i,j+1,k) - v(i,j-1,k) +
            w(i,j,k+1) - w(i,j,k-1)
        ) / (2 * dx)
    ustaw warunki brzegowe dla diverg

    // Wyznacz pole ciśnień w nowej chwili czasowej.
    dla q = {0, ..., 19}:
        dla każdego węzła obliczeniowego:
            p(i,j,k) = (
                -d * dx * dx / dt * diverg(i,j,k) +
                p(i+1,j,k) + p(i-1,j,k) +
                p(i,j+1,k) + p(i,j-1,k) +
                p(i,j,k+1) + p(i,j,k-1)
            ) / 6
        ustaw warunki brzegowe dla p

    // Oblicz prędkości płynu w nowej chwili czasowej.
    vc = dt / d / (2 * dx)
    dla każdego węzła obliczeniowego:
        u(i,j,k) = u(i,j,k) - vc*(p(i+1,j,k) - p(i-1,j,k))
        v(i,j,k) = v(i,j,k) - vc*(p(i,j+1,k) - p(i,j-1,k))
        w(i,j,k) = w(i,j,k) - vc*(p(i,j,k+1) - p(i,j,k-1))
    ustaw warunki brzegowe dla u, v, w
```

3.6 Warunki brzegowe

Poprawne zdefiniowanie warunków brzegowych jest kluczowe do poprawnego przeprowadzenia symulacji numerycznej. Głównym scenariuszem rozważanym w naszym projekcie był zamknięty prostopadłościenny pokój. Każda ściana stanowi nieprzepuszczalną barierę dla dymu, co z punktu widzenia symulacji oznacza, że komponent prędkości prostopadły do bariery jest równy zeru. Ponadto, różnica ciśnień na ścianach powinna równać się zeru, aby niezerowy gradient ciśnienia nie wypychał płynu poza przestrzeń pomieszczenia.

4 Realizacja

4.1 Technologie

Kod symulacji został napisany w języku C++. Pisane w nim oprogramowanie jest stosunkowo szybkie, a ponadto istnieje wiele innych symulatorów CFD napisanych w C++, co pozwala na porównanie zastosowanych rozwiązań.

Do realizacji graficznej strony projektu wykorzystaliśmy OpenFrameworks. Jest to zestaw bibliotek o szerokich możliwościach – przede wszystkim jednak może służyć jako wysokopoziomowa nakładka na OpenGL, który uznaliśmy za nieco zbyt obszerny i miejscami nieintuicyjny.

Podczas poszukiwania optymalnych metod obliczania kolejnych kroków symulacji przydatna okazała się biblioteka Eigen, specjalizująca się w zagadnieniach algebraicznych. Za użyciem Eigen przemawiała względna łatwość użytkownia i obszerna dokumentacja. W późniejszym etapie rozwoju kodu zrezygnowaliśmy z metody bazującej na bibliotece Eigen; nie jest ona konieczna do uruchomienia symulacji.

4.2 Implementacja

Do przechowywania wartości danej wielkości fizycznej w obrębie całej siatki służy klasa `Field3D`, którą można traktować jak tablicę trójwymiarową o rozszerzonej funkcjonalności.

$$T_{i,j,k} \equiv \texttt{T(i, j, k)}$$

Uwagi odnośnie indeksowania i użycia klasy `Field3D`: dany jest obiekt

```
Field3D f(10, 20, 30);
```

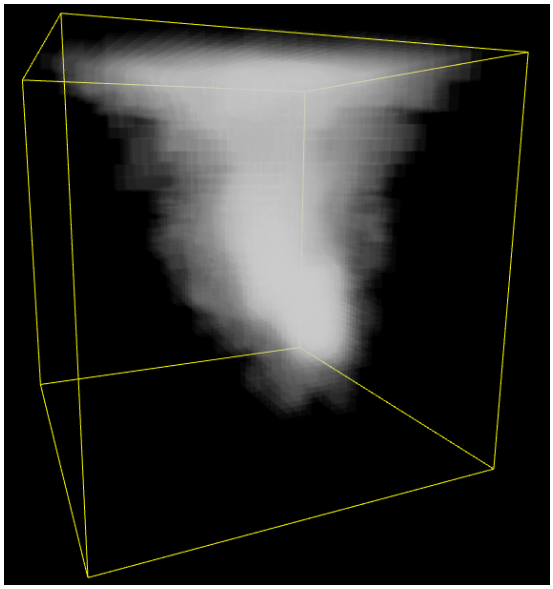
Obiekt `f` opisuje siatkę wartości o wymiarach $10 \times 20 \times 30$. Dopuszczalne wartości indeksów to: $0 \leqslant i \leqslant 9, 0 \leqslant j \leqslant 19, 0 \leqslant k \leqslant 29$. Węzłami obliczeniowymi nazywamy węzły, które nie leżą przy krawędziach siatki, o indeksach: $1 \leqslant i \leqslant 8, 1 \leqslant j \leqslant 18, 1 \leqslant k \leqslant 28$. Generalnie, węzły obliczeniowe są aktualizowane przy wszelkich obliczeniach obejmujących faktyczną matematykę symulacji, podczas gdy węzły krawędziowe są modyfikowane przy ustalaniu warunków brzegowych.

Trzonem symulacji dymu jest klasa `SmokeSolver`. Dziedziczy ona po klasie `BaseSmokeSolver`, która służy jako „opakowanie" na gęstości dymu i fundamentalne stałe symulacji. Każda klasa dziedzicząca po `BaseSmokeSolver` powinna współgrać z graficzną częścią programu, która wymaga przede wszystkim dostępu do gęstości dymu i Δx. `SmokeSolver` wprowadza resztę wielkości fizycznych i parametrów omawianych w poprzednich rozdziałach. Parametry symulacji zawarte w naszym modelu przedstawia tabela 1.

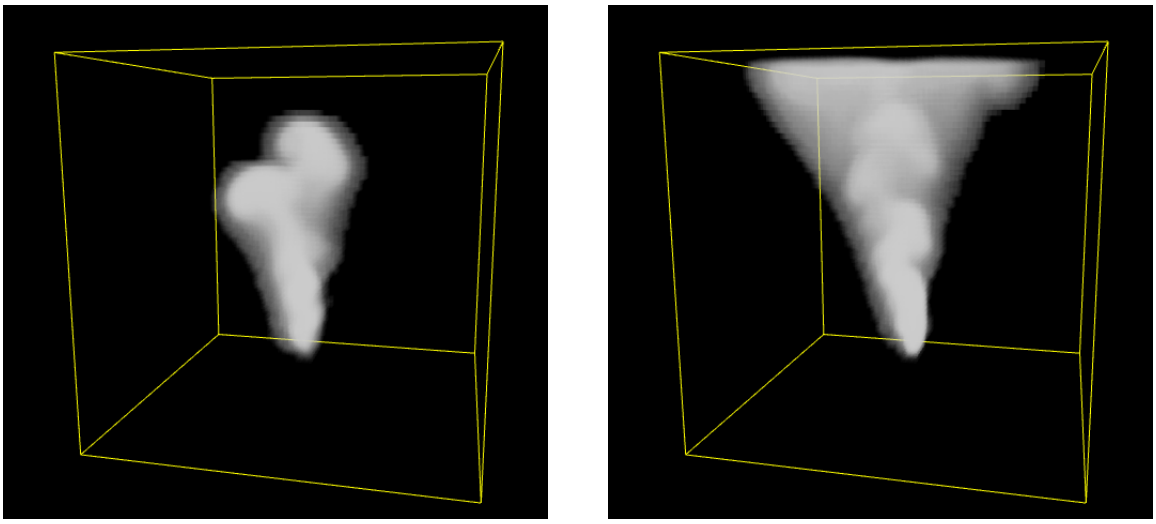
Tabela 1: Parametry symulacji

Nazwa parametru	Symbol	Opis
Parametry zdefiniowane w `BaseSmokeSolver`		
`dt`	Δt	długość kroku czasowego
`dx`	Δx	odległość między węzłami siatki obliczeniowej
`density_decay`	-	ilość, o jaką zmniejszana jest wartość gęstości dymu w każdym węźle siatki w kroku czasowym
Parametry zdefiniowane w `SmokeSolver`		
`vort_conf_coef`	-	współczynnik określający wpływ *vorticity confinement* na zachowanie płynu
`kinematic_viscosity`	ν	lepkość kinematyczna płynu
`fluid_density`	d	gęstość płynu (powietrza, nie dymu)
`k_fall`	k_{fall}	współczynnik opadania charakteryzujący siłę grawitacji działającą na dym
`k_rise`	k_{rise}	współczynnik unoszenia charakteryzujący siłę wyporu działającą na dym
`wind_velocity`	$\mathbf{u}_{\text{wind}}$	wektor prędkości wiatru
`k_wind`	k_{wind}	współczynnik określający, jak szybko prędkość płynu dąży do prędkości wiatru

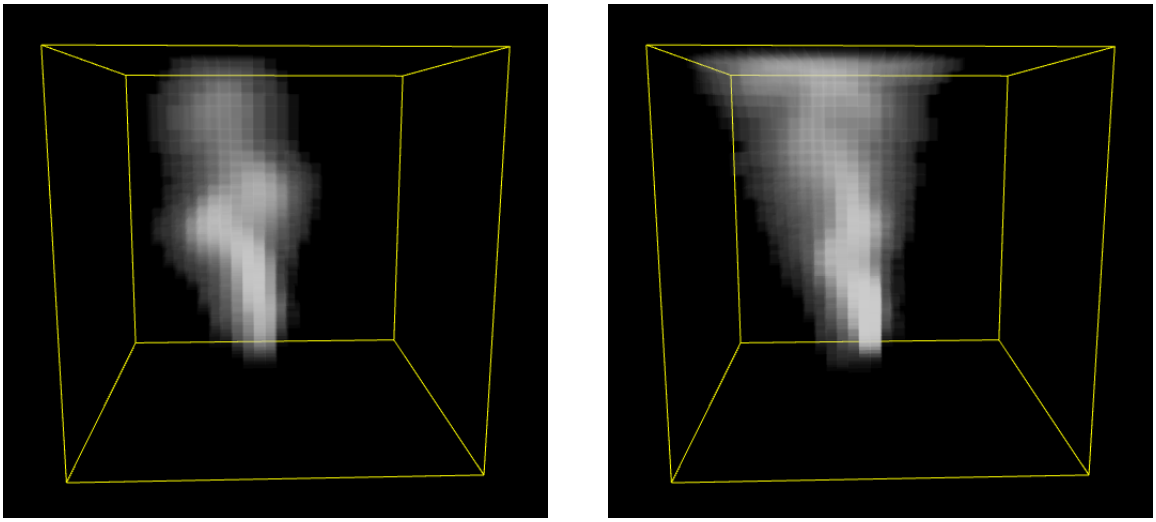
4.3 Rezultaty



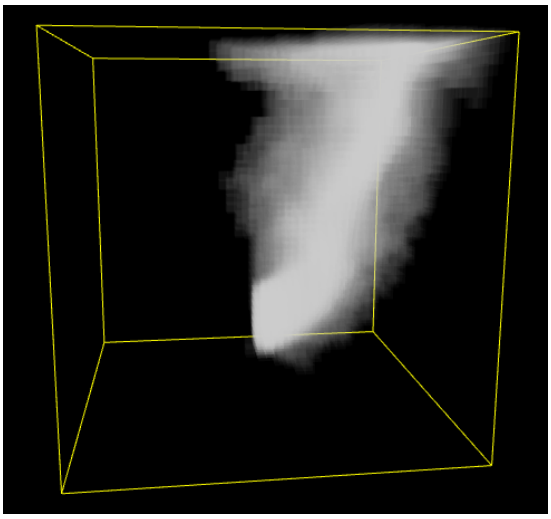
Rysunek 3: Przykładowy wynik działania symulacji.



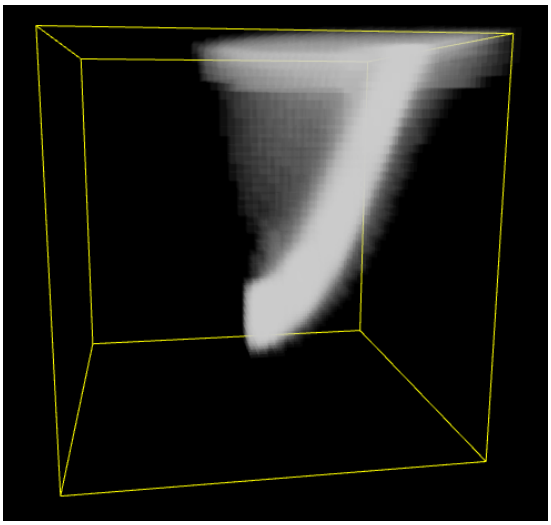
Rysunek 4: Dwa kroki symulacji dla siatki $60 \times 60 \times 60$.



Rysunek 5: Dwa kroki symulacji dla siatki $30 \times 30 \times 30$.

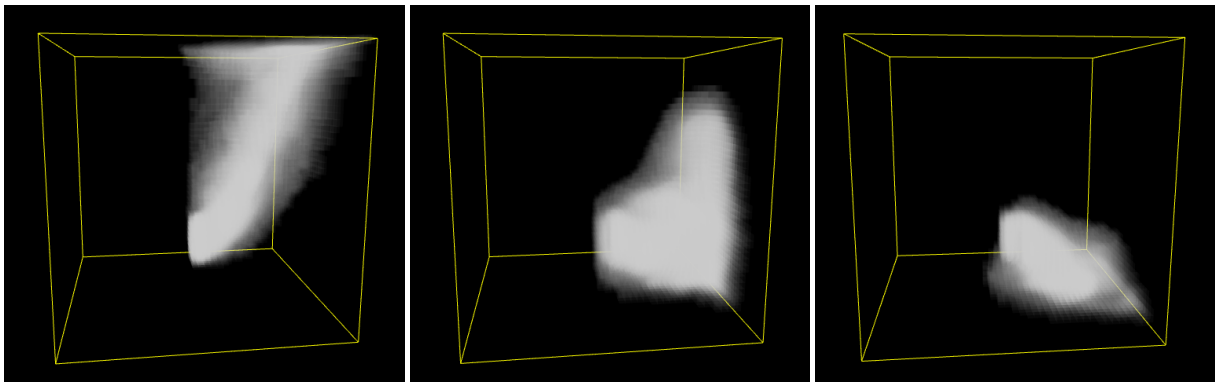


Rysunek 6: `vort_conf_coef = 2.5`



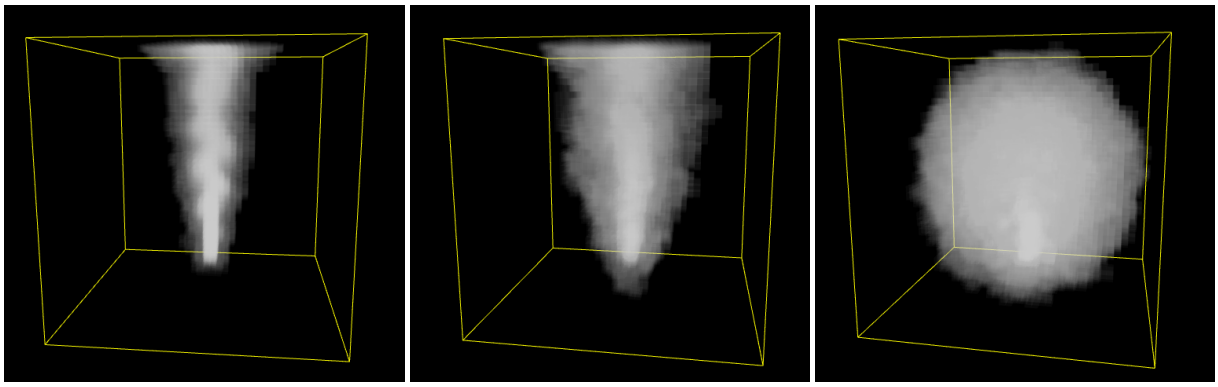
Rysunek 7: `vort_conf_coef = 0`

Rysunek 8: Wpływ *vorticity confinement*. Siatka $40 \times 40 \times 40$; niezerowa początkowa prędkość pozioma dymu.



`k_fall = 1` `k_fall = 100` `k_fall = 150`

Rysunek 9: Wpływ współczynnika opadania k_{fall}. Siatka $40 \times 40 \times 40$; niezerowa początkowa prędkość pozioma dymu.



`dt = 0.05` `dt = 0.25` `dt = 0.5`

Rysunek 10: Utrata wiarygodności symulacji na skutek zbyt dużego kroku czasowego. Siatka $40 \times 40 \times 40$.

5 Podsumowanie

Przedstawiony symulator dynamiki dymu nie jest w żadnym wypadku projektem kompletnym i zamkniętym. Istotą projektu było poznanie i implementacja matematyki stojącej za mechaniką płynów, co, jak sugerują renderowane obrazy, udało nam się zrealizować.

Sądzimy, że najbardziej obiecującymi kierunkami rozwoju byłoby dostosowanie symulacji do siatki typu *staggered*, a także połączenie jej z metodą objętości skończonych. Duży nacisk należałoby położyć na optymalizację stworzonego programu – obecna wersja symulatora działa w trybie jednowątkowym, choć żyjemy w świecie wielordzeniowych procesorów i bibliotek pozwalających wykorzystać tysiące rdzeni procesorów graficznych. Zastosowanie dynamicznej siatki obliczeniowej pozwoliłoby z kolei zaoszczędzić pamięć i potencjalnie czas, o ile narzut obliczeniowy związany z aktualizowaniem takiej siatki nie byłby znaczny.

Przede wszystkim należy jednak rozważyć zadania, jakie miałby realizować program. Wymagania wobec symulatora CFD będą różnić się w zależności, czy badamy rozwój pożaru w budynku, czy tworzymy animację dymu na potrzeby grafiki komputerowej. W jednym przypadku priorytetem może być wierność fizyce, a w drugim wizualna atrakcyjność.

Literatura

[1] D. Arumuga Perumal, Anoop K. Dass *A Review on the development of lattice Boltzmann computation of macro fluid flows and heat transfer*

[2] J. H. Ferziger, M. Perić, *Computational Methods for Fluid Dynamics*

[3] Ryszard Gryboś, *Podstawy mechaniki płynów. Część 1*

[4] Ronald Fedkiw, Jos Stam, Henrik Wann Jensen, *Visual Simulation of Smoke*

[5] *Finite Element vs Finite Volume*,
źródło: https://knowledge.autodesk.com/support/cfd/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/SimCFD/files/GUID-12A9AED8-2047-4D3A-BC80-82BE9CF47517-htm.html

[6] David Cline, David Cardon, Parris K. Egbert, *Fluid Flow for the Rest of Us: Tutorial of the Marker and Cell Method in Computer Graphics*

[7] https://en.wikipedia.org/wiki/Projection_method_(fluid_dynamics)

[8] David Potter, *Metody obliczeniowe fizyki*

[9] Jos Stam, *Stable Fluids*

[10] Eren Algan, *Real-Time Smoke Simulation*

[11] Quanbing Luo, Ting Ren, *Discretized pressure Poisson algorithm for the steady incompressible flow on a nonstaggered grid*

[12] Lorena A. Barba, Gilbert F. Forsyth, *12 steps to Navier-Stokes*, Step 11: Cavity Flow with Navier–Stokes