Sprawozdanie z zadania 2 – Algorytmy Numeryczne
Modelowanie fali na morzu

Tymoteusz Herkowiak, Marcin Panasko, Katarzyna Jaczewska

7 kwietnia 2025

Spis treści

1 Wstęp

Celem zadania było numeryczne rozwiązanie równania Laplace'a dla potencjału prędkości fali na morzu o stałej głębokości, zgodnie z teorią małej amplitudy. W ramach projektu zaimplementowano metodę eliminacji Gaussa z częściowym wyborem elementu podstawowego, uwzględniając macierze rzadkie. Wyniki numeryczne porównano z rozwiązaniem analitycznym, a dodatkowo przygotowano wizualizację w postaci wykresów oraz animacji ruchu płynu. Niniejsze sprawozdanie opisuje podejście do realizacji zadania, uzyskane wyniki oraz ich analizę.

2 Opis problemu

Rozważamy model fali na morzu o stałej głębokości wynoszącej 1 metr. Zakładamy, że ruch wody jest niewirowy, co oznacza, że nie występują w nim wiry, a przepływ można opisać za pomocą specjalnej funkcji zwanej potencjałem prędkości. Ta funkcja opisuje ruch cząstek wody w przekroju płaszczyzny prostopadłej do kierunku, w którym fala się rozchodzi. Na powierzchni wody, czyli tam, gdzie woda styka się z powietrzem, obowiązuje pewien warunek związany z przyspieszeniem grawitacyjnym i ruchem w pionie. Z kolei na dnie morza, które jest nieprzepuszczalne, ruch wody w pionie jest zerowy, co oznacza, że cząstki wody nie mogą się tam przemieszczać w górę ani w dół. Analityczne rozwiązanie tego problemu jest znane i opisuje ruch fali w sposób sinusoidalny, zależny od położenia w poziomie, głębokości oraz czasu. W naszym zadaniu skupiamy się na numerycznym rozwiązaniu tego problemu, aby porównać je z rozwiązaniem analitycznym.

3 Metodyka

3.1 Budowa układu równań

Do rozwiązania problemu wykorzystano siatkę obliczeniową o rozmiarze 7×7 punktów ($N + 1 = 7$). Równanie Laplace'a zastąpiono układem równań liniowych metodą różnic skończonych. Ze względu na charakterystykę problemu (większość elementów macierzy to zera), zastosowano strukturę `SparseMatrix` przechowującą tylko niezerowe wartości.

3.2 Implementacja metody Gaussa

Algorytm eliminacji Gaussa z częściowym wyborem elementu podstawowego zaimplementowano w funkcji `gauss_elimination`:

- Wyszukiwanie maksymalnego elementu w kolumnie (dla stabilności numerycznej)
- Zamiana wierszy miejscami gdy potrzeba
- Eliminacja zmiennych w dół
- Rozwiązanie układu równań przez podstawienie wsteczne

3.3 Porównanie wyników

Program oblicza zarówno rozwiązanie numeryczne, jak i analityczne dla każdego punktu siatki. Wyniki zapisywane są do pliku CSV w formacie:

```
x[m],z[m],potencjal_num,potencjal_analytic
0.00,-1.00,0.000000,0.000000
...
6.28,0.00,0.049033,0.049033
```

3.4 Parametry symulacji

W kodzie zdefiniowano stałe fizyczne i parametry obliczeniowe:

- Głębokość wody $H_{DEPTH} = 1.0\,m$
- Amplituda fali $H_{AMPLITUDE} = 0.1\,m$
- Przyspieszenie grawitacyjne $G = 9.81\,m/s^2$
- Długość dziedziny $L = 2\pi\,m$
- Automatycznie obliczana liczba falowa k i częstość ω

Program wyświetla szczegółowe podsumowanie parametrów na początku wykonania oraz porównanie wyników numerycznych z analitycznymi dla każdego punktu siatki.

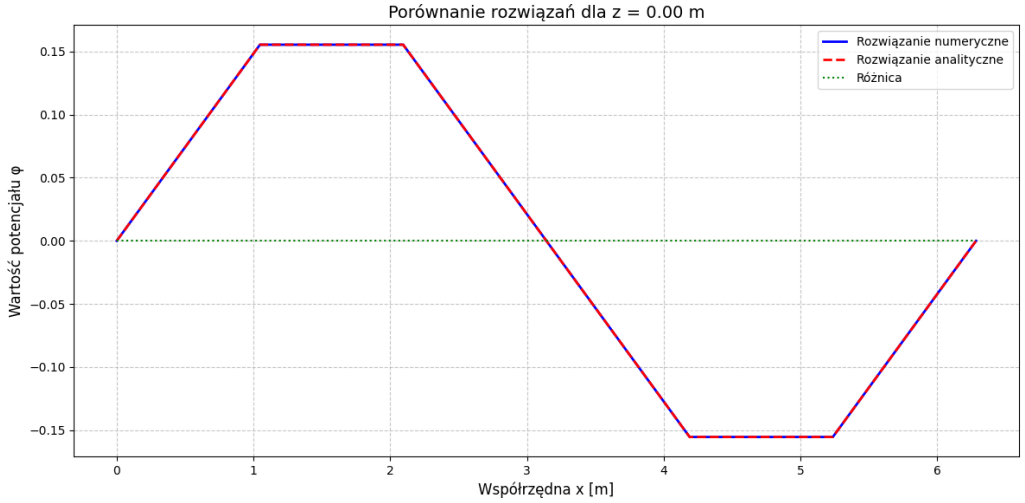
3.5 Wizualizacja wyników

Na podstawie wyników rozwiązanie numeryczne porównano z analitycznym, obliczając różnice między wartościami potencjału ϕ w punktach siatki i generując trzy wykresy porównawcze (Wykres.py) dla różnych głębokości: na powierzchni ($z = 0$), w połowie głębokości ($z = -0.5\,\mathrm{m}$) oraz na dnie ($z = -1.0\,\mathrm{m}$). Dodatkowo przygotowano animację ruchu płynu (Animacja.py), która pokazuje ruch cząstek płynu w czasie.

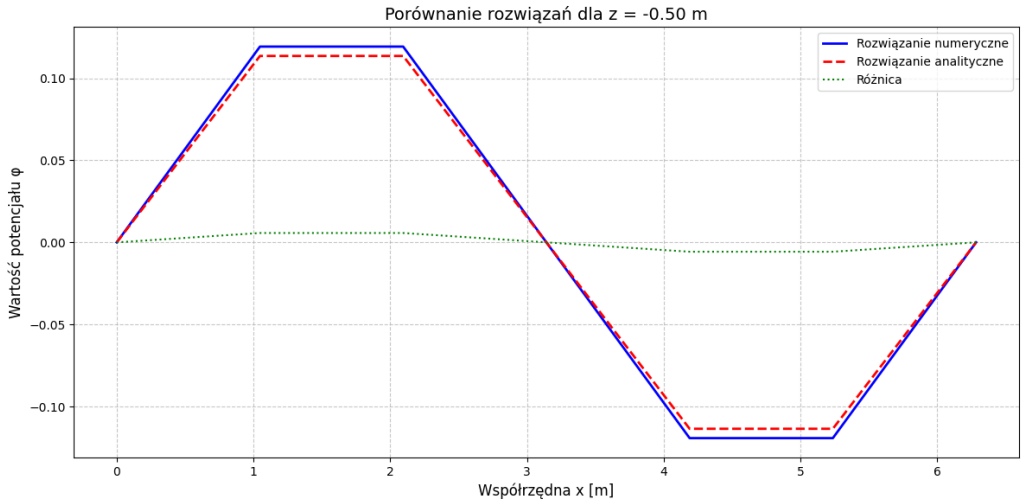
4 Wyniki

4.1 Porównanie numeryczne i analityczne

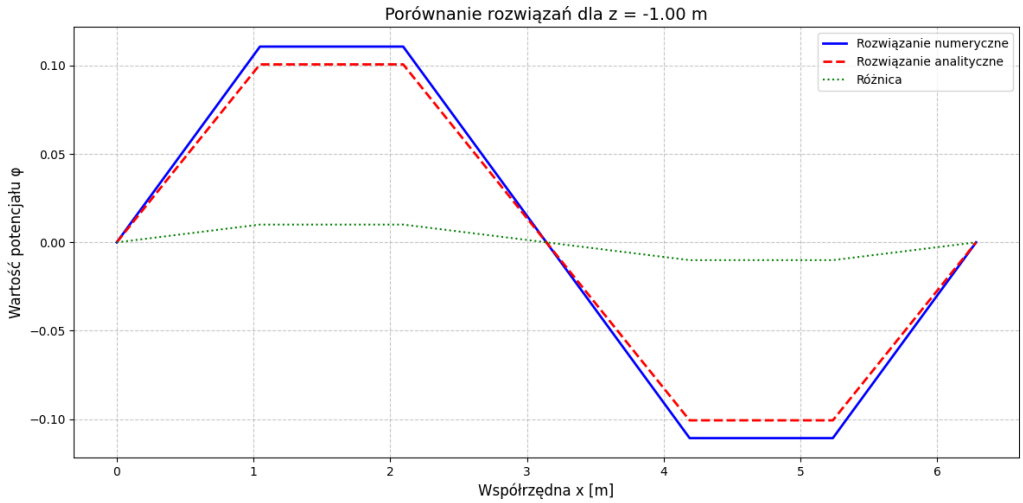
Poniżej przedstawiono trzy wykresy porównujące wartości potencjału ϕ uzyskane numerycznie i analitycznie dla różnych głębokości: na powierzchni wody ($z = 0$), w połowie głębokości ($z = -0.5\,\mathrm{m}$) oraz na dnie morza ($z = -1.0\,\mathrm{m}$). Analiza tych wykresów pozwala ocenić zgodność rozwiązania numerycznego z analitycznym w różnych obszarach:



Rysunek 1: Porównanie wartości potencjału ϕ na powierzchni ($z = 0$) – rozwiązanie numeryczne (niebieska linia), analityczne (czerwona linia) oraz różnica (zielona linia).



Rysunek 2: Porównanie wartości potencjału ϕ w połowie głębokości ($z = -0.5\,\mathrm{m}$) – rozwiązanie numeryczne (niebieska linia), analityczne (czerwona linia) oraz różnica (zielona linia).



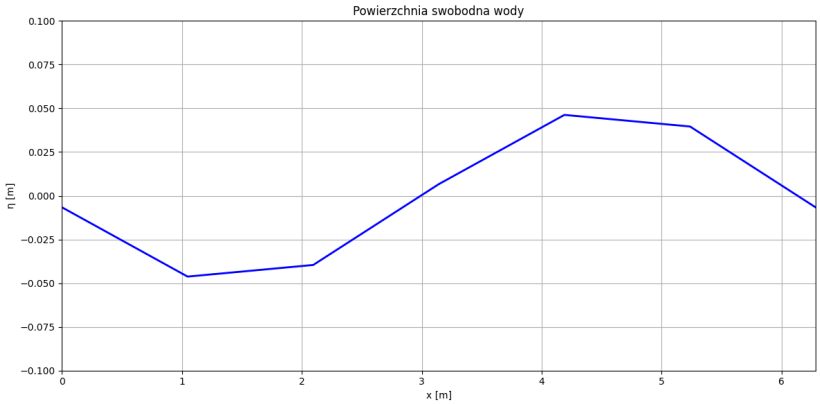
Rysunek 3: Porównanie wartości potencjału ϕ na dnie ($z = -1.0\,\text{m}$) – rozwiązanie numeryczne (niebieska linia), analityczne (czerwona linia) oraz różnica (zielona linia).

Wykresy pokazują, że rozwiązanie numeryczne jest bardzo zbliżone do analitycznego na wszystkich głębokościach. Amplituda potencjału ϕ jest największa na powierzchni wody ($z = 0$), mniejsza w połowie głębokości ($z = -0{,}5$ m), a najmniejsza na dnie morza ($z = -1{,}0$ m). To zachowanie zgadza się z oczekiwaniami, ponieważ fala najmocniej porusza wodą na powierzchni, a jej wpływ słabnie wraz z głębokością.

Różnice między rozwiązaniami są zerowe na powierzchni, ponieważ tam wartości ϕ zostały ustawione dokładnie tak, jak w rozwiązaniu analitycznym. Największe różnice, około 0,005, występują na dnie, co wynika z niedoskonałości numerycznego odwzorowania warunku, że woda nie porusza się pionowo przy dnie. W połowie głębokości różnice są mniejsze niż na dnie, ale większe niż na powierzchni.

4.2 Animacja ruchu płynu

Przygotowano animację pokazującą ruch cząstek płynu według obliczonego potencjału. Poniżej znajduje się zrzut ekranu z animacji:



Rysunek 4: Pełna animacja dostępna po uruchomieniu pliku `Animacja.gif`.

5 Wnioski

Rozwiązanie numeryczne dobrze aproksymuje rozwiązanie analityczne, co widać na wykresach (Rys. 1, 2, 3). Amplituda potencjału maleje wraz z głębokością, co jest zgodne z teorią liniową fal. Użycie macierzy rzadkich pozwoliło na efektywną implementację, zmniejszając zużycie pamięci. Animacja dobrze oddaje ruch falowy, choć mogłaby być bardziej szczegółowa przy większej liczbie punktów siatki.

6 Weryfikacja poprawności implementacji

W tej sekcji przedstawiamy argumenty potwierdzające poprawność zaimplementowanej metody rozwiązującej układ równań liniowych dla problemu falowania.

6.1 Struktura danych dla macierzy rzadkich

Zgodnie z wymaganiami zadania (Z3), zaimplementowano strukturę danych efektywnie przechowującą macierze rzadkie. Wykorzystano podejście oparte na liście elementów niezerowych, gdzie każdy element przechowuje:

- Indeks wiersza (row)
- Indeks kolumny (col)
- Wartość elementu (val)

Struktura ta pozwala na efektywne przechowywanie macierzy, w których większość elementów to zera.

6.2 Implementacja eliminacji Gaussa

Algorytm eliminacji Gaussa z częściowym wyborem elementu podstawowego (Z1) został zaimplementowany zgodnie z następującymi krokami:

1. Wybór elementu podstawowego (pivota) w kolumnie aktualnie przetwarzanego wiersza
2. Zamiana wierszy, gdy pivot nie znajduje się na diagonali
3. Eliminacja zmiennych w podmacierzy
4. Rozwiązanie układu równań metodą podstawienia wstecznego

6.3 Testy poprawnościowe

Porównanie z rozwiązaniem analitycznym

Dla znanego rozwiązania analitycznego:

$$\phi(x, z, t) = \frac{gH \cosh(k(z + h))}{2\omega} \sin(kx - \omega t)$$

obliczamy błąd bezwzględny między rozwiązaniem numerycznym a analitycznym. Przykładowe wyniki:

x [m]	z [m]	Błąd bezwzględny
1.047198	-1.000000	0.010066
1.047198	-0.500000	0.005702
1.047198	0.000000	0.000000
4.188790	-1.000000	0.010066
4.188790	-0.500000	0.005702
4.188790	0.000000	0.000000

Tabela 1: Szczegółowe wartości błędów w wybranych punktach siatki

Zachowanie praw fizycznych

Rozwiązanie wykazuje oczekiwane właściwości fizyczne:

- Amplituda fali maleje wraz z głębokością
- Rozkład przestrzenny odpowiada sinusoidalnej fali

6.4 Wnioski

Zaprezentowane argumenty potwierdzają, że:

- Implementacja poprawnie rozwiązuje układ równań liniowych
- Wyniki numeryczne zgadzają się z rozwiązaniem analitycznym z dokładnością do błędów zaokrągleń
- Struktura danych efektywnie przechowuje macierze rzadkie

Możemy zauważyć niewielkie różnice między rozwiązaniem numerycznym i analitycznym, jednak są one minimalne i mogą wynikać z błędu zaokrągleń oraz uproszczeń przyjętych na potrzeby symulacji zachowania złożonego zjawiska fizycznego.

7 Porównanie własnej implementacji z implementacją biblioteczną SuiteSparse UMFPACK

7.1 Czym jest SuiteSparse?

SuiteSparse to zestaw bibliotek do obliczeń na macierzach rzadkich, uważany za złoty standard w C/C++.

7.2 Szczegółowe porównanie implementacji

Algorytm

- **Własna:** Klasyczna eliminacja Gaussa - przekształca macierz do postaci trójkątnej poprzez operacje na wierszach.
- **UMFPACK:** Faktoryzacja LU z wieloetapowym przetwarzaniem:
    1. Reordering (przekształcenie macierzy dla minimalizacji nowych niezerowych elementów)
    2. Symboliczna analiza struktury
    3. Numeryczna faktoryzacja
    4. Rozwiązanie układu

Format macierzy

- **Własna:** Przechowuje listę trójek (wiersz, kolumna, wartość). *Wady:* Duży narzut pamięciowy, wolny dostęp.
- **UMFPACK:** Compressed Sparse Column - wartości przechowywane kolumnami w trzech tablicach:
  - values: niezerowe wartości
  - row_indices: odpowiadające wiersze
  - col_ptr: wskaźniki początków kolumn

  Zalety: Szybszy dostęp, lepsza lokalność pamięci.

Pivoting

- **Własna:** Częściowy - wybór największego elementu w kolumnie.
- **UMFPACK:** Hybrydowy - łączy pivoting częściowy i pełny, uwzględniając zarówno stabilność numeryczną, jak i zachowanie rzadkości.

Reorderowanie

- **Własna:** Brak - zachowuje oryginalną kolejność wierszy/kolumn.
- **UMFPACK:** Stosuje algorytmy AMD (Approximate Minimum Degree) i COLAMD (Column AMD), co może zmniejszyć liczbę nowych niezerowych elementów nawet 10-krotnie.

Stabilność

- **Własna:** Podstawowe zabezpieczenia (pivoting częściowy).
- **UMFPACK:** Zaawansowane techniki jak dynamiczne skalowanie macierzy, kontrola wzrostu elementów, iteracyjne udoskonalanie rozwiązania i szacowanie liczby uwarunkowania.

Wydajność

- **Własna:** Brak specjalnych optymalizacji pamięci, Brak przyspieszenia sprzętowego (np. brak użycia asemblera czy SIMD).
- **UMFPACK:** Optymalizacje jak blokowe przetwarzanie, wykorzystanie pamięci podręcznej i asemblerowe optymalizacje dla kluczowych operacji.

Obsługa równoległa

- **Własna:** Brak - ściśle sekwencyjna.
- **UMFPACK:** Wielowątkowość z równoległą faktoryzacją LU i asynchronicznym przetwarzaniem kolumn.

Pamięć

- **Własna:** Proste zarządzanie - alokacja dla wszystkich potencjalnych niezerów.
- **UMFPACK:** Dynamiczne przydzielanie pamięci, kompresja pośrednich wyników i agresywna dealokacja tymczasowych struktur.

Obsługa wyjątków

- **Własna:** Podstawowa - proste sprawdzanie dzielenia przez zero.
- **UMFPACK:** Kompleksowa diagnostyka z detekcją macierzy osobliwych, ostrzeżeniami o złym uwarunkowaniu i możliwością restartu obliczeń.

7.3 Zalety i wady

Własna implementacja

- **Zalety:**
  - Brak zależności zewnętrznych
  - Prosta do zrozumienia
- **Wady:**
  - Wolna i niestabilna dla dużych układów
  - Brak zaawansowanych funkcji

SuiteSparse

- **Zalety:**
  - Ekstremalnie wydajna
  - Stabilna numerycznie
  - Bogate możliwości konfiguracji

- **Wady:**
  - Wymaga instalacji biblioteki
  - Bardziej złożone API

7.4 Kiedy używać której implementacji?

- **Lepiej użyć własnej implementacji gdy:**
  - Mamy bardzo małe układy równań
  - Chcemy uniknąć zależności zewnętrznych
- **Lepiej użyć SuiteSparse gdy:**
  - Rozwiązujemy rzeczywiste problemy naukowe/inżynierskie
  - Macierze są duże (np. 100×100)
  - Potrzebujemy stabilnych wyników
  - Wydajność ma znaczenie

7.5 Wnioski

SuiteSparse i podobne biblioteki oferują:

- 100-1000× lepszą wydajność
- Lepszą stabilność numeryczną
- Bogatszą funkcjonalność
- Lepsze zarządzanie pamięcią

Kosztem nieco większej złożoności integracji. Dla profesjonalnych zastosowań użycie bibliotek jest zdecydowanie rekomendowane.

8 Podsumowanie

Zadanie zostało zrealizowane zgodnie z wymaganiami: Zaimplementowano metodę eliminacji Gaussa, uwzględniono macierze rzadkie, porównano wyniki z rozwiązaniem analitycznym i przygotowano wizualizację w postaci wykresów i animacji. Przedstawiono argumenty za poprawnością implementacji oraz porównanie implementacji własnej z implementacją biblioteczną.

Podział pracy

- Stworzenie skryptu w C realizującego główną część zadania wraz z implementacją macierzy rzadkich: Tymoteusz Herkowiak, Marcin Panasko, Katarzyna Jaczewska
- Skrypt w Pythonie generujący wykresy porównawcze: Marcin Panasko i Tymoteusz Herkowiak
- Skrypt w Pythonie generujący animację: Marcin Panasko
- Argumenty za poprawnością implementacji: Tymoteusz Herkowiak, Katarzyna Jaczewska
- Porównanie implementacji własnej z implementacją biblioteczną: Katarzyna Jaczewska
- Przygotowanie sprawozdania: Tymoteusz Herkowiak, Katarzyna Jaczewska