Raport Zadanie NUM3

Tomasz Dziób

17.11.2023

1 Wstęp techniczny

Poniższy raport dotyczy zadania numerycznego NUM3 z listy 2. Załączony program o nazwie *num3.cpp* został napisany w języku *C++*, do sprawdzenia obliczeń został użyty plik *check.cpp* również napisany w języku *C++* z wykorzystaniem bibioteki *GSL*. Aby uzyskać wykres prezentujący zależność wielkości danych wejściowych od czasu dla algorytmu z zadania *num3.cpp* służy plik *plot.cpp* napisany języku *C++* korzystający z bibioteki *GNU Plot*. Przed skompilowaniem programu należy zainstalować powyższe biblioteki.

1.1 Jak uruchomić program?

Razem z załączonymi plikami znajdziemy *Makefile* który służy do uruchomienia programu *num3.cpp* oraz *check.cpp* komendą: *make run*.
Aby uruchomić program *plot.cpp* korzystamy z komendy: *make runplot* $DATA\_AMOUNT = rozmiar$,
gdzie *rozmiar* to maksymalny rozmiar macierzy, $rozmiar > 50$

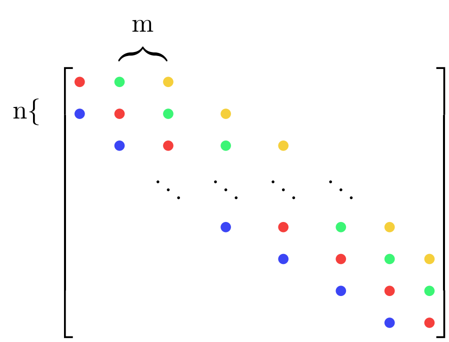
2 Nakreślenie problemu

Najwiekszą przeszkodą w rozwiązywaniu równań macierzowych jest przechowywanie danych w pamięci. Macierze dla nawet stosunkowo nie dużych zestawów danych (patrząc z perspektywy tego ile danych jest procesowanych w obecnych czasach) potrafią zajmować ogromną ilość miejsca na dysku.

Dla $N = milion$:

$$8\text{bajtów} \cdot N^2 = 8 \cdot (10^6)^2 = 8 \cdot 10^{12} \text{ bajtów} = 8 \cdot 10^3 \text{ GB}$$

Jednak w przypadku zadania *NUM3* mamy doczynienia z macierzą tak zwaną wstęgową. Charakteryzuje się ona tym, że posiada wartości jedynie na diagonali oraz na pasmach, pod oraz nad diagonalą (liczba pasm pod oraz nad mogą być różne! - tak jak jest w tym przypadku). Pozostałe elementy są wypełnione zerami.



Przypadek ten pozwala nam zaoszczędzić pamięć pomijając zapisywanie zer. Można to osiągnąć poprzez trzymanie macierzy w tablicy zawierającej tylko wypełnione pasma. Rozważając ponownie poprzedni przykład:

Dla $N = milion$:

$$8\text{bajtów} \cdot 4N = 32N = 32 \cdot 10^6 \text{ bajtów} = 32 \text{ MB}$$

Jak zauważamy jest to diametralna różnica jeśli chodzi o zużycie pamięci.
Jednakże jako, że liczba naszych danych wejściowych jest mniejsza, zmniejsza się również złożoność obliczeniowa. Jest to ogromną zaletą ponieważ standardowo dla rozwiązania układów równań liniowych jest to rząd wielkości O(N^3), tymczasem dla macierzy wstęgowych tą metodą jesteśmy wstanie to rozwiązać w czasie O(N).

3 Użyta metoda

Rozwiązanie sprowadza się do obliczenia rokładu LU macierzy A oraz obliczenia dwóch równań $Lt = x$ (metodą *forward substitution*) oraz $Uy = t$ (metodą *back substitution*) które wynikają z prostych przekształceń zadanego równania oraz podstawienia.
Do osiągnięcia rozkładu LU macierzy użyte zostały uproszczone wzory na uzyskanie poszczególnych elementów LU (uproszczone ponieważ niektóre fragmenty uległy skróceniu przez wartości zerowe w macierzy).

$$u_{i,i} = a_{i,i} - l_{i,i-1} \cdot u_{i-1,i} \qquad u_{i,i+1} = a_{i,i+1} - l_{i,i-1} \cdot u_{i-1,i+1}$$
$$u_{i,i+2} = a_{i,i+2} \qquad l_{i+1,i} = \frac{a_{i+1,i}}{u_{i,i}}$$

4 Uzyskany wynik

Po wykonaniu komendy *make run* wykonają się dwa programy jeden po drugim. Najpierw otrzymamy wynik uzyskany omówioną wcześniej metodą a później obliczony z pomocą bibioteki *GSL*.

• Plik *num3.cpp*

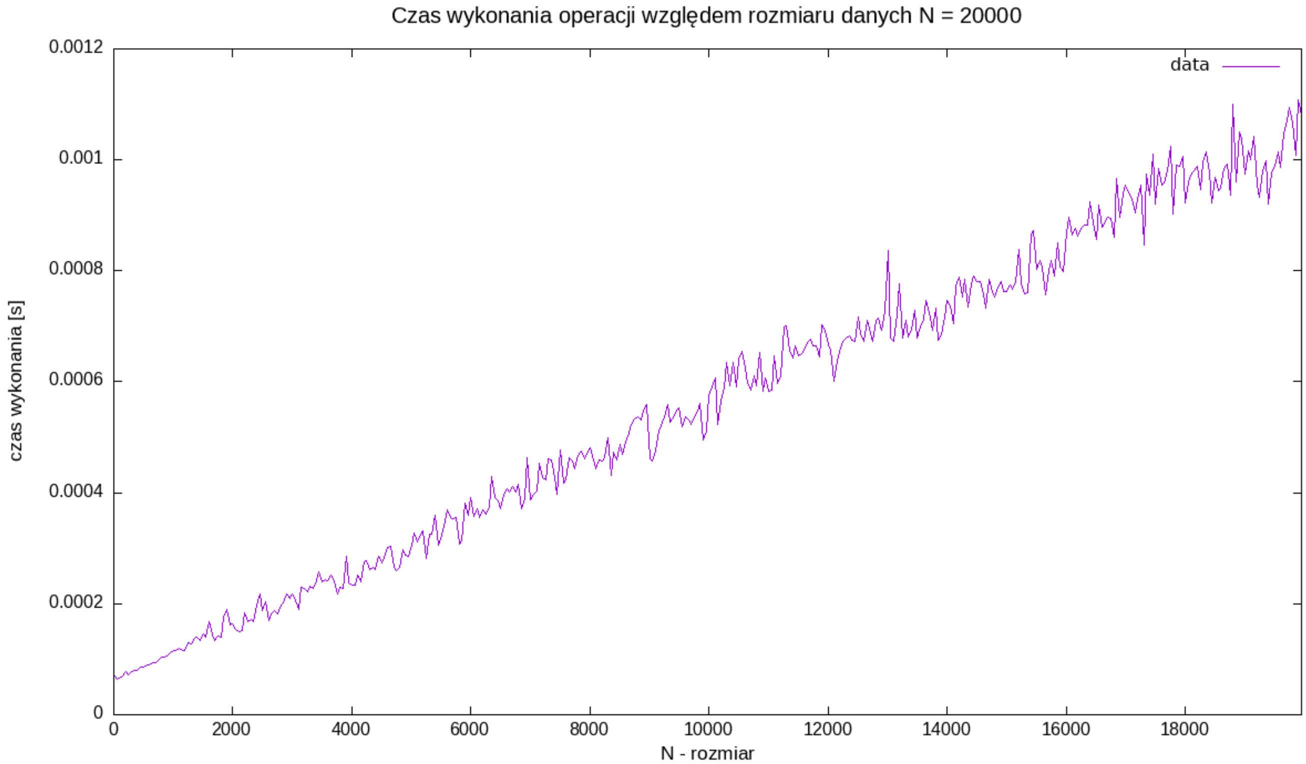
```
Wektor y:
0.448701 1.41327 2.13488 2.86901 3.59149 4.3116 5.02983 5.74701 6.4635 7.17953 7.89521 8.61065 9.3259
10.041 10.756 11.4709 12.1857 12.9005 13.6152 14.3299 15.0445 15.7591 16.4736 17.1882 17.9027 18.6172
19.3317 20.0462 20.7606 21.4751 22.1895 22.9039 23.6184 24.3328 25.0472 25.7616 26.476 27.1903 27.9047
28.6191 29.3335 30.0478 30.7622 31.4765 32.1909 32.9053 33.6196 34.3339 35.0483 35.7626 36.477 37.1913
37.9056 38.62 39.3343 40.0486 40.763 41.4773 42.1916 42.9059 43.6203 44.3346 45.0489 45.7632 46.4775
47.1919 47.9062 48.6205 49.3348 50.0491 50.7634 51.4777 52.1921 52.9064 53.6207 54.335 55.0493 55.7636
56.4779 57.1922 57.9065 58.6208 59.3351 60.0494 60.7637 61.478 62.1924 62.9067 63.621 64.3353 65.0496
65.7639 66.4782 67.1925 67.9068 68.6211 69.3354 70.0497 70.764 71.4783 72.1926 72.9069 73.6212 74.3355
75.0498 75.7641 76.4784 77.1927 77.9069 78.6212 79.3355 80.0498 80.7641 81.4784 82.1927 82.907 83.6213
84.3356 85.0499 85.7642 86.4785 87.1928 87.9078 88.682
Wyznacznik macierzy A:
6.14197e+09
```

• Plik *check.cpp*

```
Wektor y:
0.448701 1.41327 2.13488 2.86901 3.59149 4.3116 5.02983 5.74701 6.4635 7.17953 7.89521 8.61065 9.3259
10.041 10.756 11.4709 12.1857 12.9005 13.6152 14.3299 15.0445 15.7591 16.4736 17.1882 17.9027 18.6172
19.3317 20.0462 20.7606 21.4751 22.1895 22.9039 23.6184 24.3328 25.0472 25.7616 26.476 27.1903 27.9047
28.6191 29.3335 30.0478 30.7622 31.4765 32.1909 32.9053 33.6196 34.3339 35.0483 35.7626 36.477 37.1913
37.9056 38.62 39.3343 40.0486 40.763 41.4773 42.1916 42.9059 43.6203 44.3346 45.0489 45.7632 46.4775
47.1919 47.9062 48.6205 49.3348 50.0491 50.7634 51.4777 52.1921 52.9064 53.6207 54.335 55.0493 55.7636
56.4779 57.1922 57.9065 58.6208 59.3351 60.0494 60.7637 61.478 62.1924 62.9067 63.621 64.3353 65.0496
65.7639 66.4782 67.1925 67.9068 68.6211 69.3354 70.0497 70.764 71.4783 72.1926 72.9069 73.6212 74.3355
75.0498 75.7641 76.4784 77.1927 77.9069 78.6212 79.3355 80.0498 80.7641 81.4784 82.1927 82.907 83.6213
84.3356 85.0499 85.7642 86.4785 87.1928 87.9078 88.682
Wyznacznik macierzy A:
6.14197e+09
```

Porównując oba wyniki jesteśmy wstanie identyczne wyniki w obu przypadkach.

Aby sprawdzić czy algorytm jest optymalny powstał plik *plot.cpp*. Uruchamia się go komendą *make runplot DATA_AMOUNT = rozmiar*. Wykonuje on algorytm do zadanej liczby N, zaczynając od $i = 50$ inkrementując również o 50. Dla uzyskania wyników odpornych na inne procesy działające w tle program wykonuje operacje dla każdego N sto razy oraz uśrednia wynik. Po wykonaniu obliczeń otrzymujemy wyniki naniesione na wykres.



Rysunek 1: Wynik uzyskany po uruchomienu programu *plot.cpp*

Razem ze wzrostem rozmiaru N jesteśmy wstanie zauważyć wzrost szumów wystepujących mimo obliczania średniej z stu uzyskanych wyników.

5 Podsumowanie

Znajomość struktury macierzy jest kluczowa przy wybieraniu odpowiedniego sposobu rozwiązania równań liniowych. Może on pozwolić na znaczne odciążenie naszego procesora ze zbędnych obliczeń poprzez zmniejszenie jego złożoności czasowej. Co więcej, właściwe dobranie sposobu przechowywania macierzy w pamięci pozwoli na rozwiązanie problemów z danymi o kilka rzędów wiekszych od dostępnego.