Raport Zadanie NUM4

Tomasz Dziób

25.11.2023

1 Wstęp techniczny

Poniższy raport dotyczy zadania numerycznego NUM4 z listy 3. Załączony program o nazwie *num4.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 *num4.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 *num4.cpp* oraz *check.cpp* komendą: *make run*
Aby uruchomić program *plot.cpp* korzystamy z komendy: *make runplot*

2 Nakreślenie problemu

Podczas rozwiązywania równań typu $\mathbf{Ay} = \mathbf{b}$ dla niewiadomej b napotykamy problem w postaci potrzeby odnalezienia wartości A^{-1} której obliczenie jest skomplikowane obliczeniowo jak i pamięciowo. Najlepiej wykorzystać jest to co się posiada a mianowicie macierz A.

3 Użyta metoda

Spoglądając na macierz A jesteśmy wstanie odrazu w zauważyć jej chrakterytyczną budową, na diagonali 12, wstęga nad diagonalą same 7, reszta jedynki. Pozwala to nam na użycie wzoru Shermana-Morrisona:

$$A^{-1}b = B^{-1} - \frac{B^{-1}uv^TB^{-1}}{1+v^TB^{-1}u}, \qquad \text{gdzie } A = uv^T + B$$

Z pomocą tego wzoru jesteśmy wstanie rozwiązać cały problem w czasie złożoności *O(N)*. Najpierw **macierz A** rozpisujemy na wartości **u**, $\mathbf{v^T}$ oraz **macierz B**.

$$\begin{bmatrix} 12 & 8 & 1 & 1 & 1 \\ 1 & 12 & 8 & 1 & 1 \\ 1 & 1 & \ddots & \ddots & 1 \\ 1 & 1 & 1 & 12 & 8 \\ 1 & 1 & 1 & 1 & 12 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \end{bmatrix} + \begin{bmatrix} 11 & 7 & 0 & 0 & 0 \\ 0 & 11 & 7 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 11 & 7 \\ 0 & 0 & 0 & 0 & 11 \end{bmatrix}$$

Następnie możemy zastosować podstawienie we wzorze aby uzyskać dwa równania.

$$A^{-1}b = B^{-1} - \frac{B^{-1}uv^TB^{-1}}{1+v^TB^{-1}u} \qquad / \cdot b$$

$$A^{-1}b = B^{-1}b - \frac{B^{-1}uv^TB^{-1}b}{1+v^TB^{-1}u}$$

$$A^{-1}b = {\color{blue}B^{-1}b} - \frac{{\color{red}B^{-1}u}v^T{\color{blue}B^{-1}b}}{1+v^T{\color{red}B^{-1}u}}$$

$$A^{-1}b = {\color{blue}z} - \frac{{\color{red}q}v^T{\color{blue}z}}{1+v^T{\color{red}q}}$$

$$\begin{cases} {\color{blue}z} = B^{-1}b \\ {\color{red}q} = B^{-1}u \end{cases}$$

Pozostaje nam rozwiazać dwa równania umieszczone poniżej używając rozkładu LU macierzy wstęgowej poprzez *forward substitution* oraz *back substitution*, da się to osiągnąć w czasie *O(N)*. Pozostałe operacje to oblicznie iloczynu skalarnego, mnożenie wektorów przez skalar czy odejmowanie wektorów. Wszystkie do obliczania w tej samej złożoności czasowej.

$$\begin{cases} Bz = b \\ Bq = u \end{cases}$$

4 Uzyskany wynik

Obliczenia wykonane z użyciem algorytmu *Shermana-Morrisona* pokrywa się z wynikiem uzyskanym z pojedynczego wywołania komendy *solve* w bibiotece *GSL*.

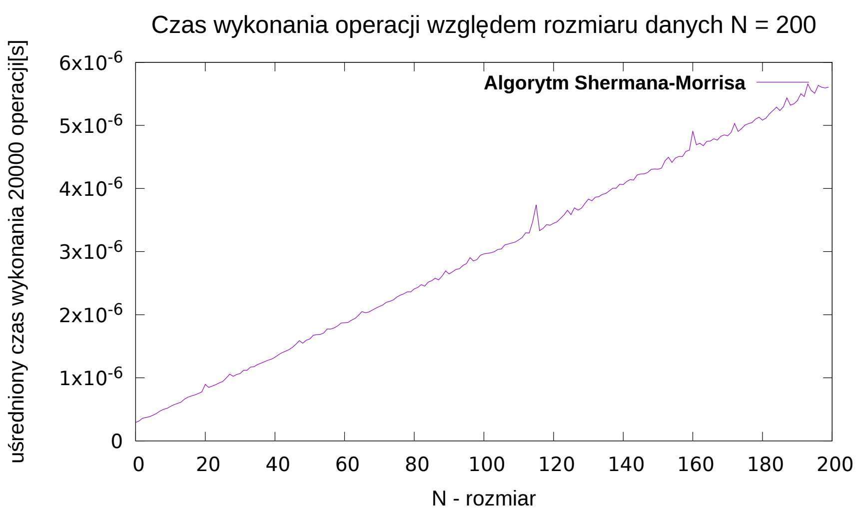
• Wynik obliczony przez program *num4.cpp*

Wektor y:
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508188 0.0508187 0.0508188 0.0508187 0.0508188
0.0508187 0.0508188 0.0508186 0.050819 0.0508183 0.0508194 0.0508178 0.0508203 0.0508163 0.0508226
0.0508127 0.0508282 0.0508039 0.0508421 0.050782 0.0508765 0.050728 0.0509614 0.0505946 0.0511709
0.0502653 0.0516885 0.0494521 0.0529664 0.0474439 0.0561221 0.0424849 0.0639148 0.0302393 0.0831579

• Wynik obliczony przez program *check.cpp*

Wektor y:
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508187
0.0508187 0.0508187 0.0508187 0.0508187 0.0508187 0.0508188 0.0508187 0.0508188 0.0508187 0.0508188
0.0508187 0.0508188 0.0508186 0.050819 0.0508183 0.0508194 0.0508178 0.0508203 0.0508163 0.0508226
0.0508127 0.0508282 0.0508039 0.0508421 0.050782 0.0508765 0.050728 0.0509614 0.0505946 0.0511709
0.0502653 0.0516885 0.0494521 0.0529664 0.0474439 0.0561221 0.0424849 0.0639148 0.0302393 0.0831579

Patrząc na wykres uzyskany z pliku *plot.cpp* widzimy zachowanie wyników dla algorytmu *Shermana-Morrisona*. Każda operacja została wykonana *20000 razy* aby uśrednić wynik i zmniejszyć wpływ czynników zewnętrznych takich jak inne procesy działające w tle. Wzraz ze wzrostem danych czas wykonania omawianego algorytmu obeswujemy liniową złożoność czasową. Jesteśmy wstanie zaobserowować, mimo dużego uśrednienia, wciąż pojawiający się szum wzrastający równo z ilością danych wejściowych.



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

5 Podsumowanie

Wzór Shermana-Morrisona sprawuje się idealne w przypadku potrzeby obliczenia odwrotności macierzy jeśli jesteśmy tylko wstanie wyrazić ją jako wynik $uv^T + B$. Jest to operacja o wiele prostsza od znajdywania odwrotności macierzy. Wykonanie jej wiąże się również z polepszoną złożonością czasową.