Konrad Gądek

Grupa laboratoryjna 2

Sprawozdanie z zadania 2

# Wykonanie ćwiczenia

Jako zadanie dostaliśmy napisanie programów w MatLabie rozwiązujących równanie konwekcji-dyfuzji oraz równanie przewodnictwa ciepła. Wykonując to ćwiczenie starałem się pisać programy zwięzłe i w miarę możliwości ogólne. Gdzie było to możliwe, wprowadzałem parametry zamiast konkretnych danych liczbowych.

Dla obu równań stworzyłem osobne funkcje je rozwiązujące (znajdują się one w plikach rownanieKonwekcjiDyfuzjiFDM.m oraz rownaniePrzewodnictwaCieplaFDM.m), a osobno (w pliku main.m) utworzyłem program, który generuje testy, wywołuje funkcje z odpowiednimi parametrami, wypisuje podsuwania oraz tworzy wykresy.

Ponieważ w sprawozdaniu, porównując algorytmy, podaję ich czas działania, warto zamieścić przynajmniej częściowy opis platformy testowej

|  |  |
| --- | --- |
| Komponent | Opis |
| Oprogramowanie | MatLab 7.11.0 (R2010) |
| System operacyjny | Windows 7 Professional 64-bit |
| Procesor | Intel Pentium Dual Core E2180 2,0GHz |
| Zainstalowana pamięć operacyjna | 2 GB |

# Zastosowanie MRS do rozwiązania równania konwekcji-dyfuzji

## Błędy aproksymacji

By obliczyć błędy aproksymacji, napisałem mały program, którego zadaniem było – dla każdej wybranej wartości n (5, 10, 50, 100, 500) – policzyć wynik dokładny, a także wynik przybliżony. Obliczenia przeprowadzałem na 4 zestawach parametrów wejściowych podanych w treści zadania.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | b |  |  |  |
| 2 | 5 | 2 | 0 | 0 |
| 2 | 5 | 2 | 0 | 1 |
| 2 | 5 | 2 | 1 | 1 |
| 5 | 9 | 5 | 10 | 20 |

Wyniki podsumowuje przedstawia poniższa tabela:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Błąd maksymalny | Bł. średni | Bł. minimalny |
| 5 | 0,6987 | 0,05 | 0,0020 |
| 10 | 0,7065 | 0,0112 | 0,00035087 |
| 50 | 0,7089 | 0,00041198 | 0,0000027533 |
| 100 | 0.7091 | 0,00010196 | 0,00000034334 |
| 500 | 0,7091 | 0,0000040459 | 0,0000000027415 |

Łatwo zauważyć, że wraz ze wzrostem wartości liczby n maleje błąd minimalny i średni błąd aproksymacji. Wydaje się, że ów błędy maleją proporcjonalnie do kwadratu liczby n. Błąd maksymalny natomiast się nie zmienia. Gdy przedstawi się błąd aproksymacji na wykresie, przyczyna staje się jasna – błąd jest minimalny na granicy przedziału natomiast pośrodku osiąga swoje maksimum o wartości ok. 0,7091. Wykres zależności błędu aproksymacji dla wartości n=500 zamieściłem obok.

## Szybkość działania

Podobnie jak wcześniej, do porównania szybkości działania stworzonych funkcji wykorzystałem mały program wywołujący funkcje wiele razy i zbierający statystyki wywołań. Porównywałem 3 metody rozwiązywania układu równań liniowych: metody polegającej na wyznaczeniu macierzy odwrotnej, metody wbudowanej w program MatLab (metoda eliminacji Gaussa) oraz własnej implementacji eliminacji Gaussa. Każdy test wywołałem 10-krotnie by zmniejszyć zakłócenia mierzonych czasów.

|  |  |  |  |
| --- | --- | --- | --- |
| n | inv(A)\*B | A\B | Własna impl. Gaussa |
| 5 | 0,0004975 | 0,0002937 | 0,0003495 |
| 10 | 0,0008685 | 0,0003153 | 0,0005075 |
| 50 | 0,0015 | 0,0008 | 0,0025 |
| 100 | 0,0032 | 0,0014 | 0,0085 |
| 500 | 0,0977 | 0,03 | 1,7598 |

Zgodnie z oczekiwaniami, wbudowana implementacja eliminacji Gaussa okazała się najszybsza; najwolniejsza okazała się własna implementacja co nie powinno dziwić - została napisana bez żadnych modyfikacji mających przyspieszyć jej działanie. Stosunkowo dobrze wypadła metoda polegająca na odwróceniu macierzy – działała średnio 2-3x dłużej niż najszybsza metoda.

## Wnioski

Mimo, że zarówno badanie błędów aproksymacji może – dla niektórych danych wejściowych – wprowadzać w błąd podając sprzeczne ze zdrowym rozsądkiem wyniki, oraz mimo, że porównywanie czasów działania różnych algorytmów jest rzeczą trudną o ile w ogóle możliwą, to udało się uzyskać rozsądne wyniki, które potwierdzają teorię jak i przypuszczenia.

Wraz z dwukrotnym wzrostem liczby n, czterokrotnie maleje błąd minimalny. Odbywa się to oczywiście kosztem czasu działania, jednak – jak zobaczyliśmy – dla n=500 wbudowana metoda eliminacji Gaussa radzi sobie doskonale i generuje wynik średnio w 0,03 sekundy. Okazało się też niestety, że pisanie własnych implementacji niektórych algorytmów – oprócz wątpliwej sensowności duplikowania kodu – może powodować zmniejszenie wydajności programu o kilka rzędów wielkości nawet dla stosunkowo niewielkich danych.

# Zastosowanie MRS do rozwiązania równania przewodnictwa ciepła

## Omówienie

W czasie rozwiązywania tej części zadania boleśnie dała mi się we znaki wrażliwość przedstawionej metody jawnej na dane wejściowe – mimo, że dane *wydawały się* odpowiednie, program generował całkowicie bezsensowne wyniki. Po bardzo długich poszukiwaniach błędu okazało się, że wszystkiemu winne są złe parametry przekazane do programu.

## Wyniki

Po wprowadzeniu następujących parametrów:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a |  |  |  |  |  | n |  |  |
| 1 | 0 | 1 | 0 | 0 | 1 | 50 | 0,0001 | 100 |

otrzymałem 3 następujące wykresy przedstawiające – odpowiednio – obliczenia przeprowadzone metodą jawną, metodą niejawną i wzorem analitycznym.



Porównując dwie pierwsze metody z metodą referencyjną, otrzymujemy następujące wykresy błędów bezwzględnych:



Zgodnie z przewidywaniami okazało się, że metoda niejawna jest nieco dokładniejsza. Warto zauważyć, że wraz o ile na początku *symulacji* błędy są dość znaczące, to wraz z biegiem czasu stają się coraz mniejsze. Daje to nadzieje na możliwość wykorzystania tych technik w praktyce.