Algorytmy numeryczne

Zadanie 3
Tomasz Adamczyk | Aleksander Kosma
243217 | 238193
grupa 1 tester-programista

8 Listopad 2017

Dane wejściowe

Większość testów została przeprowadzona dla:

rozmiar planszy	ilość grzybów	wartości kostki
7	4	-4 -3 -2 -1 0 1 2 3 4

Implementacja metod iteracyjnych

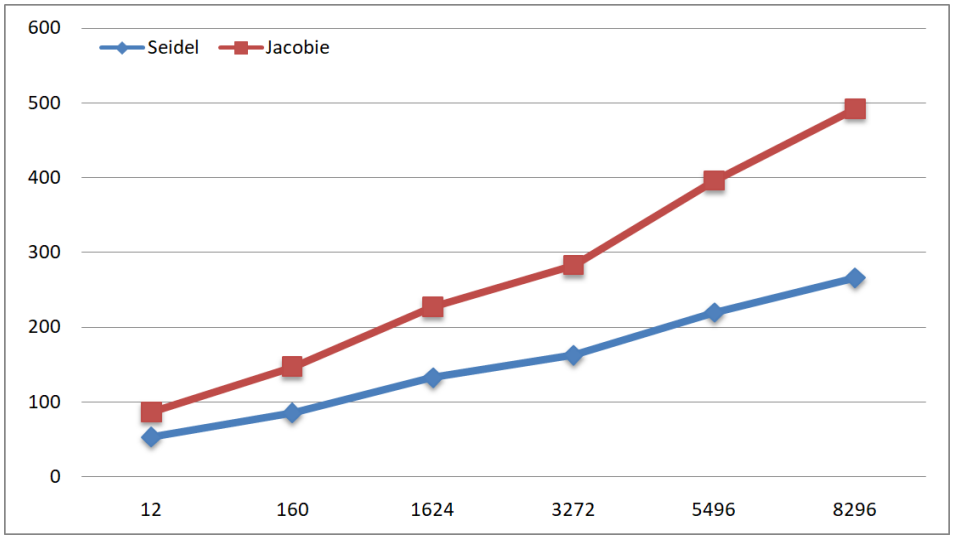
W pierwszej części sprawozdania zbadamy poprawność metod iteracyjnych do rozwiązywania układów liniowych.

Rozmiar macierzy	Iteracje	Monte Carlo	Gauss Siedl	Jacobie
1598	20	0,58998	0,57727760827	0,57468277056
1598	50	0,58756	0,58965872158	0,58952519470

.
W przypadku kiedy ilość iteracji jest równa 20, różnica wyniku mieści się w granicy nawet ponad 1% szansy na wygraną. W momencie kiedy iteracji jest już 50, różnica ta spada do zaledwie około 0.2%. Można więc stwierdzić, iż metody iteracyjne można wykorzystać do rozwiązania zagadnienia Grzybobranie.

Optymalna ilość iteracji w przypadku metod iteracyjnych

Aby zoptymalizować czas potrzebny od wyliczenia wyniku przez metody iteracyjne, warto wiedzieć ile danych iteracji potrzebujemy. Stratą czasu jest liczenie kolejnego przybliżenia, które nic więcej nam nie powie. Za duża ilość iteracji może przeważyć na końcowych rezultatach i wnioskach.
Zastosowany został przez nas prosty algorytm, liczący średnią różnice sum wektora poprzedniego i aktualnego. Jeśli dana różnica jest mniejsza niż podany epsylon, kończymy kolejne iteracje. Dodatkowym zabezpieczeniem jest kontynuacja dopóki podana różnica jest mniejsza od epsylon określoną ilość z rzędu.
Wykres pokazuje mniejszą potrzebę iteracji dla Gaussa-Seidela. Jacobie potrzebuje więcej iteracji, jak i szybciej przyspiesza z kolejnymi wymaganymi iteracjami.

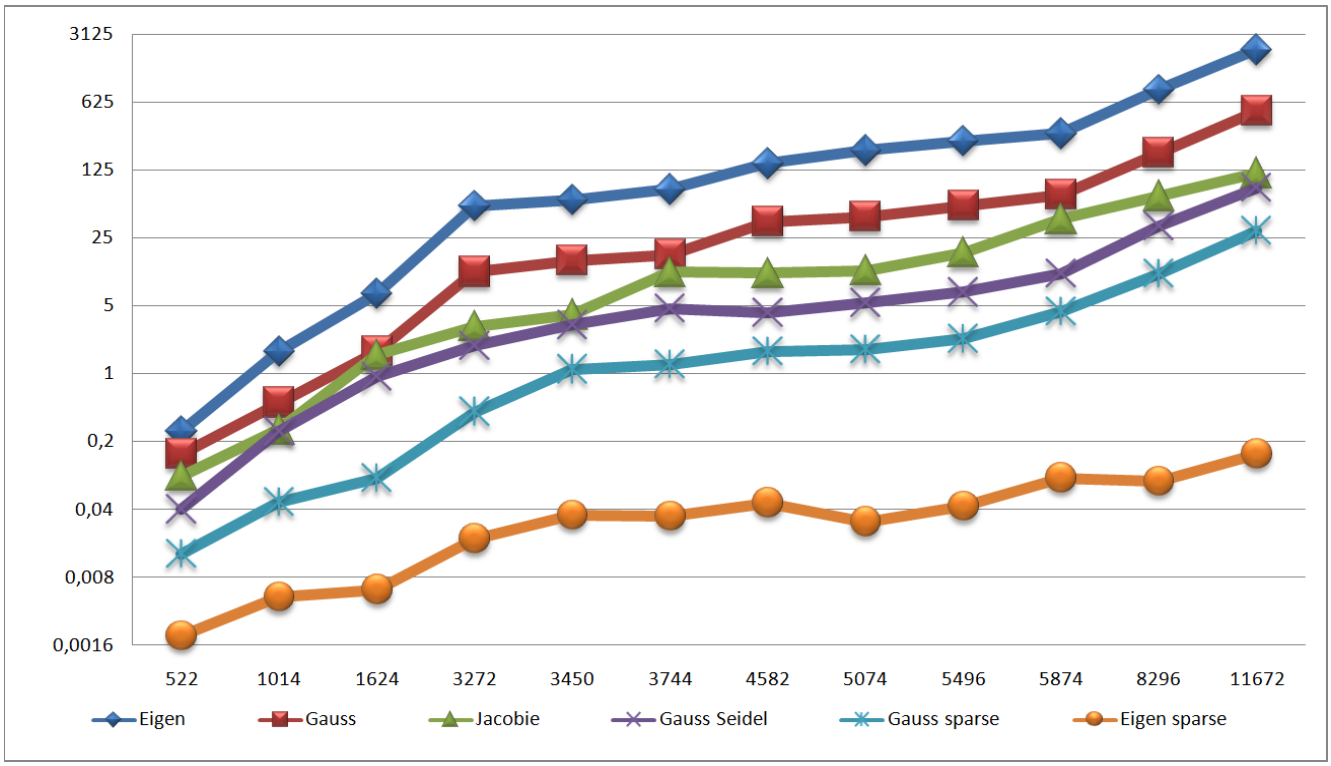


Optymalny dobór metody w grze Grzybobranie

Druga część opisuje wnioski, oparte o porównania wyników i czasu działania podanych metod:

-metoda Gaussa z częściowym wyborem elementu podstawowego
-metoda Gaussa z częściowym wyborem elementu podstawowego z optymalizacją dla macierzy rzadkich
-metoda iteracyjna Gaussa-Seidela
-metoda iteracyjna Jacobiego
-metoda z biblioteki Eigen partialPivLu, z częściowym wyborem elemntu podstawowego
-metoda z biblioteki Eigen SparseLU, z częściowym wyborem przy użyciu macierzy rzadkich

Poniższy wykres prezentuje czas potrzebny do rozwiązania równania. Eigen sprase ma miażdżącą przewagę nad resztą. Zdecydowanie najlepiej korzystać z tej metody. Jedynym jej mankamentem jest potrzeba dostarczenia wektora z informacją o ilości wartości miejsc niezerowych w kolumnach.



Precyzja wyników obliczona przez każdą z powyższych metod była praktycznie ta sama. W lwiej części sytuacji wyniki pokrywały się idealnie. Ma to związek z bardzo małą ilością wartości niezerowych. W konsekwencji komputer nie ma szansy zgubić gdzieś precyzji obliczeń. Prawdopodobnie wraz ze wzrostem macierzy, mogłyby się pojawić pewne różnice w wynikach.

Obliczenia wraz ze wzrostem rozmiaru macierzy były liczone od 10 do 2 próbek. Zazwyczaj tyle wystarczało by zauważyć tendencje wzrostowe.

Podział obowiązków	
Aleksander Kosma	**Tomasz Adamczyk**
-Implementacja Monte Carlo	-Implementacja metod iteracyjnych
-Iteracyjne generowanie układu równań	-Ustalenie i implementacja warunków wygranej
-XXOptymalizacja na HashTreeXX	-Wprowadzanie układu równań do macierzy
-Optymalizacja metod iteracyjnych i Gaussa	-Generowanie wyniku poprzez bibliotekę Eigen w C++
-Testy i generowanie wyników	-Napisanie skryptów do generowania wyników
-Obróbka wyników, wykresy i opracowanie	-