Aproksymacja średniokwadratowa dyskretna

Grams, Stanisław
Jezierski, Maciej
Korczakowski, Juliusz
MFI UG
Algorytmy Numeryczne

14 stycznia 2019

1 Implementacja

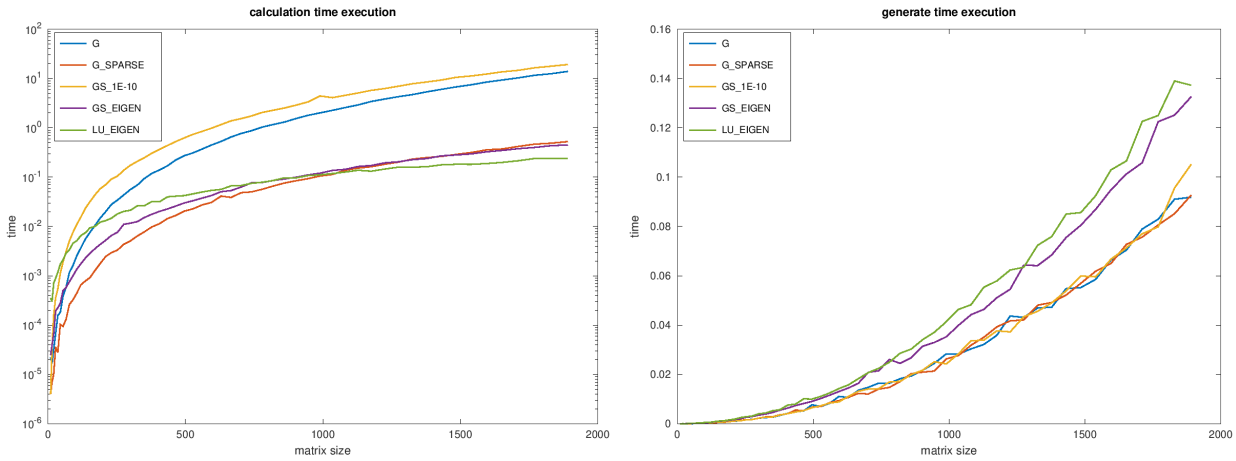
Program *„approximations”* został napisany w języku C++ a wyniki działania programu zapisywane są do poszczególnych plików **.csv*.

1.1 Zaimplementowane oraz użyte algorytmy

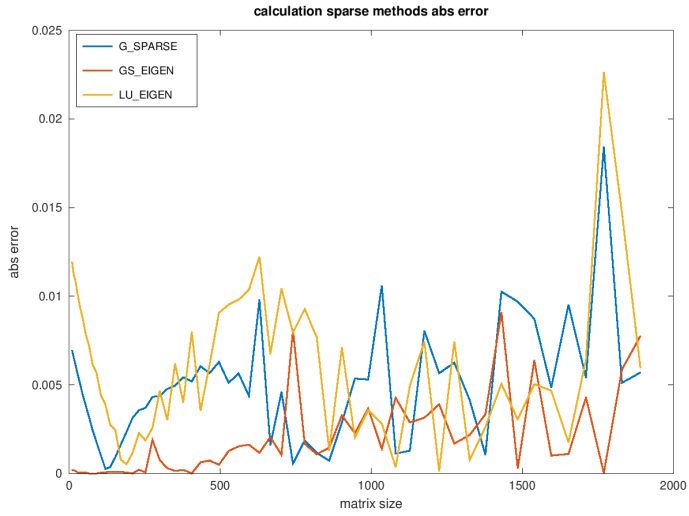
- (**G**): Algorytm Gaussa z częściowym wyborem elementu
- (**G_SPARSE**): Algorytm Gaussa z częściowym wyborem elementu i optymalizacją dla macierzy rzadkich
- (**GS_1E10**): Algorytm Gaussa-Seidela (precyzja 1×10^{-10}, struktura macierzy tablicowa)
- (**GS_EIGEN**): Algorytm Gaussa-Seidela z implementacją dla biblioteki *Eigen*3 (precyzja 1×10^{-10}) [1]
- (**LU_EIGEN**): Algorytm SparseLU pochodzący z biblioteki *Eigen*3 [2]

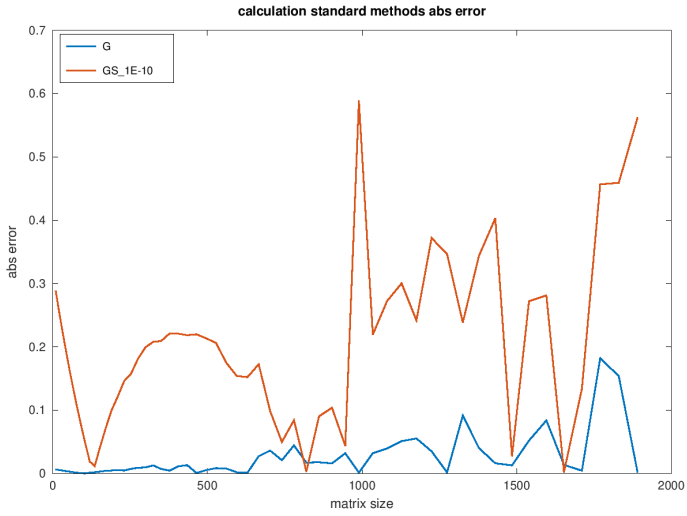
W celu obsługi metod **GS_EIGEN** oraz **LU_EIGEN** opartych o bibliotekę *Eigen*3 należało w dodatku do zadania nr 3 doimplementować klasy *SparseMatrix* oraz *SparseGenerator* pozwalające na wydajne operacje na nowych typach.
Testy zostały wykonane dla ilości agentów równej $N = 3..60$.
Przypomnienie: rząd macierzy można wyznaczyć ze wzoru $A = \frac{(N+1)*(N+2)}{2}$

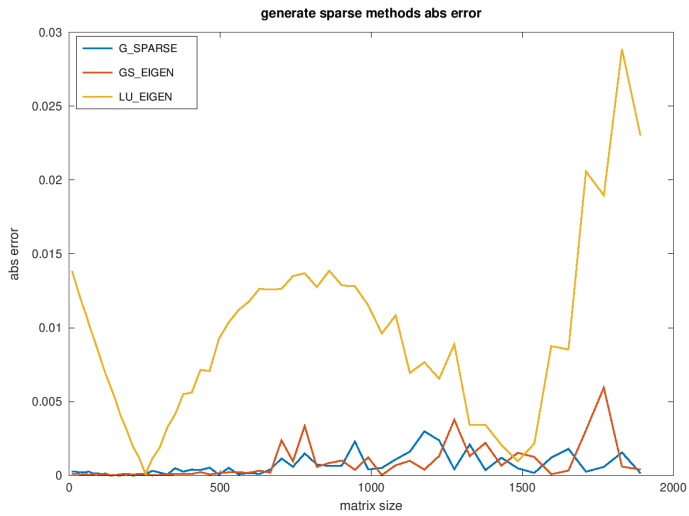
2 Analiza wyników

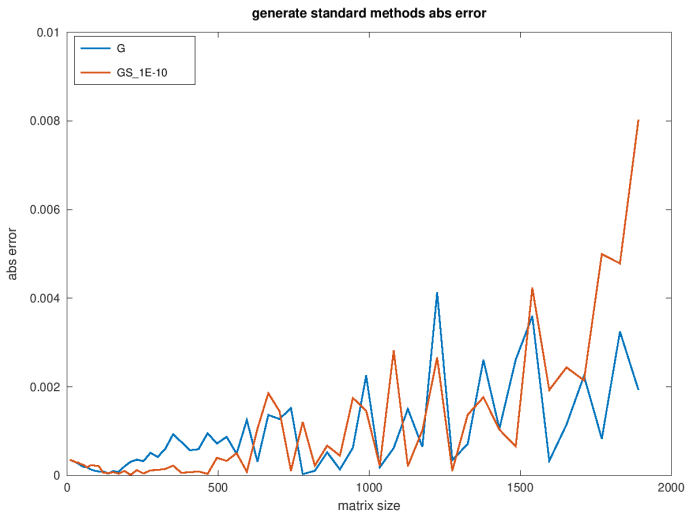


[1] http://komi.web.elte.hu/elektronikus/src/p184-koester.pdf
[2] https://eigen.tuxfamily.org/dox/classEigen_1_1SparseLU.html









3 Aproksymacja

3.1 Wyliczone współczynniki wielomianów

- Rozwiązywanie układu równań:
  1. (**G**): $f(x) = 2.08794\text{e}{-9}x^3 + (-8.95133\text{e}{-8})x^2 + 8.29139\text{e}{-5}x^1 - 0.00728348$
  2. (**G_SPARSE**): $f(x) = 1.85075\text{e}{-7}x^2 + (-7.88855\text{e}{-5})x^1 + 0.00772861$
  3. (**GS_1E10**): $f(x) = 6.81554\text{e}{-6}x^2 + (-0.00319507)x^1 + 0.319738$
  4. (**GS_EIGEN**): $f(x) = 1.29824\text{e}{-7}x^2 + (-3.45283\text{e}{-6})x^1 + 0.000242044$
  5. (**LU_EIGEN**): $f(x) = 0.000129468x^1 - 0.0128829$
- Generowanie macierzy:
  1. (**G**): $f(x) = 1.04135\text{e}{-12}x^3 + (2.25873\text{e}{-8})x^2 + 3.36946\text{e}{-6}x^1 - 0.000383727$
  2. (**G_SPARSE**): $f(x) = 2.51315\text{e}{-8}x^2 + 1.55354\text{e}{-6}x^1 - 0.00028272$
  3. (**GS_1E10**): $f(x) = 2.83629\text{e}{-8}x^2 + (-2.47976\text{e}{-6})x^1 + 0.000380382$
  4. (**GS_EIGEN**): $f(x) = 3.71718\text{e}{-8}x^2 + (-2.87381\text{e}{-8})x^1 + 0.000124703$
  5. (**LU_EIGEN**): $f(x) = 6.80852\text{e}{-5}x^1 - 0.014481$

3.2 Poprawność wyników aproksymacji – błędy bezwzględne

	Obliczanie	Generowanie
G	0.307364774043810629	0.009470184227150496
G_SPARSE	0.043499922329411848	0.006761815415625999
GS_1E10	1.792431959293490085	0.013518592473739061
GS_EIGEN	0.020786725071974720	0.009817717627604215
LU_EIGEN	0.055710669049853154	0.081653183626553646

Wniosek 1. Uzyskane wyniki znajdują się w granicy tolerancji błędu dla funkcji aproksymującej opartej o aproksymacje średniokwadratową dyskretną.

4 Ekstrapolacja

4.1 Ekstrapolacja czasu obliczeń dla układu o rozmiarze rzędu 100000

	Wyliczony czas [s]
G	2087048.540863713948056102
G_SPARSE	1842.864744169787172723
GS_1E10	67836.170996347194886766
GS_EIGEN	1297.890374618339365043
LU_EIGEN	12.933910601128554063

5 Próba obliczenia układu o rozmiarze rzędu 100000 i klasa SparseMatrix

Jako najszybszą metodę uznaliśmy **LU_EIGEN** i tą metodą wykonaliśmy test dla macierzy rzędu 100 128 (446 agentów). Uzyskany czas wynosi 63.2919210000000002095 i jest około 4.9x gorszy od aproksymowanego. Metody oparte o bibliotekę *Eigen*3 odznaczają się również znacznie niższym zużyciem pamięci operacyjnej.

6 Podział pracy

Stanisław Grams	Juliusz Korczakowski	Maciej Jezierski
Implementacja algorytmu Gaussa-Seidela	Implementacja klasy Approximation	Przygotowanie sprawozdania
Implementacja klasy SparseMatrix oraz SparseGenerator w oparciu o *Eigen*3	Przygotowanie wykresów oraz tabel	Agregacja uzyskanych wyników
Implementacja wywołania poszczególnych prób (*main.cc*)	Uruchomienie prób i testowanie	Praca nad strukturą projektu