Aproksymacja średniokwadratowa dyskretna

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

14 stycznia 2019

1 Implementacja

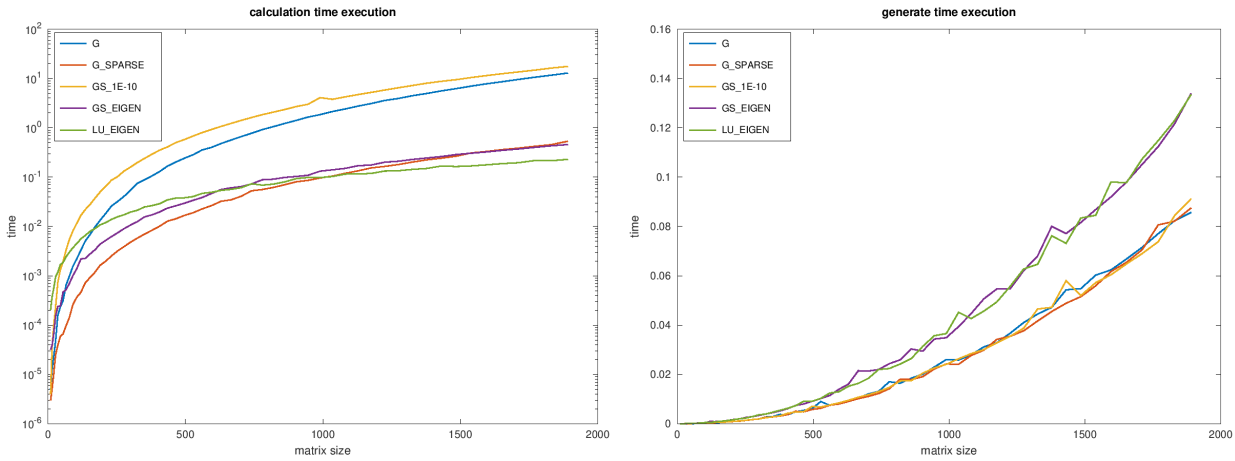
Program *„approximations”* został napisany w języku C++ a wyniki działania programu zapisywane są do poszczególnych plików **.csv*.

1.1 Zaimplementowane oraz użyte algorytmy

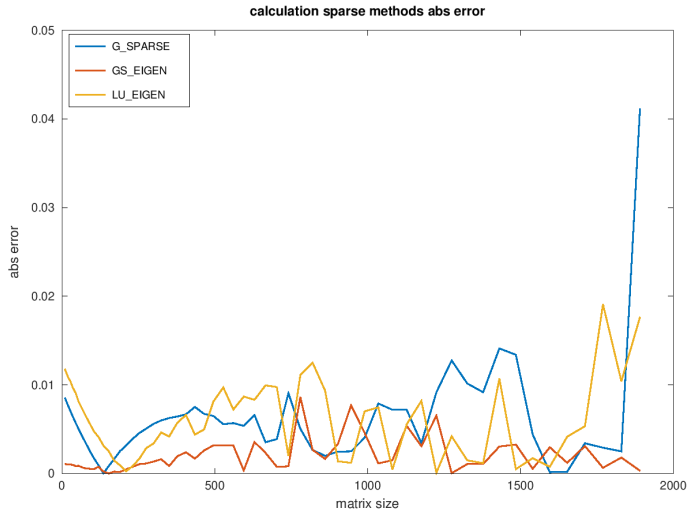
- (**G**): Algorytm Gaussa z częściowym wyborem elementu
- (**G_SPARSE**): Algorytm Gaussa z częściowym wyborem elementu i optymalizacją dla macierzy rzadkich
- (**GS_1E10**): Algorytm Gaussa-Seidela (precyzja 1×10^{-10}, struktura macierzy tablicowa)
- (**GS_EIGEN**): Algorytm Gaussa-Seidela z implementacją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

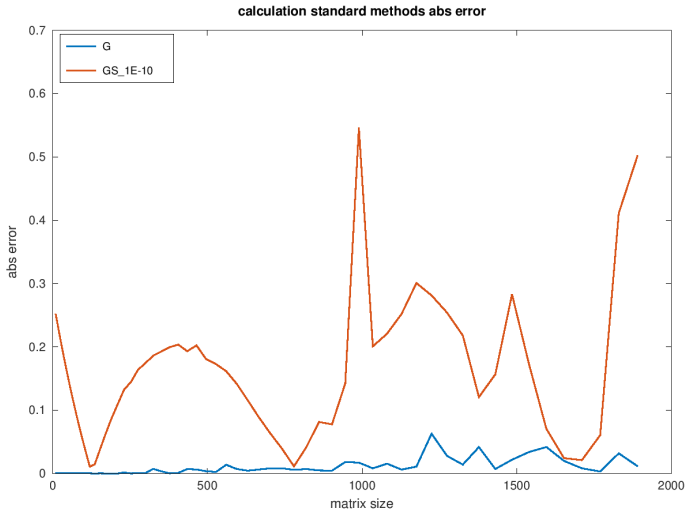
W celu obsługi metod **GS_EIGEN** oraz **LU_EIGEN** opartych o bibliotekę *Eigen*3 należało w dodatku do zadania nr 3 doimplementować klasy *SparseMatrix* oraz *SparseGenerator* pozwalające na wydajne operacje na nowych typach.
Testy zostały wykonane dla ilości agentów równej $N = 3..60$.
Przypomnienie: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

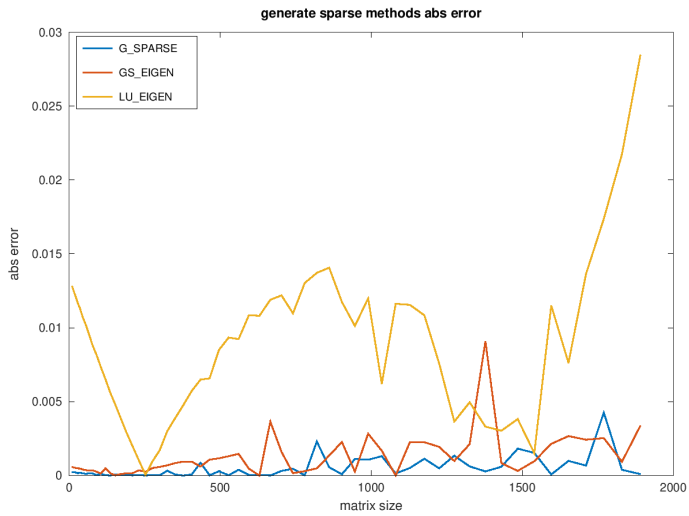
2 Analiza wyników

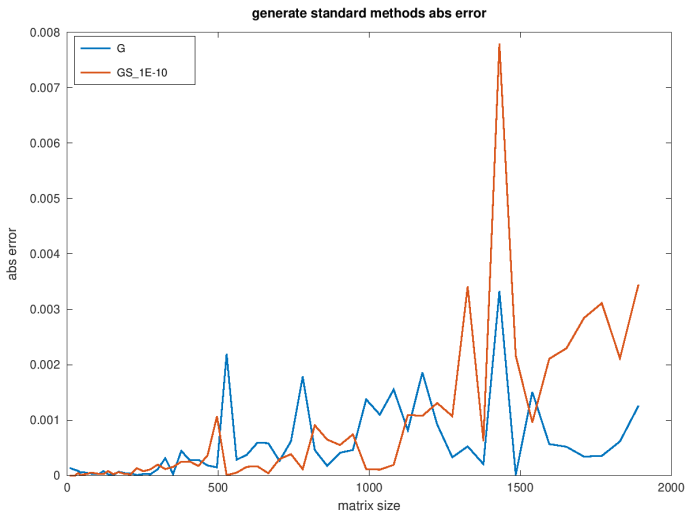


[1] `http://komi.web.elte.hu/elektronikus/src/p184-koester.pdf`
[2] `https://eigen.tuxfamily.org/dox/classEigen_1_1SparseLU.html`





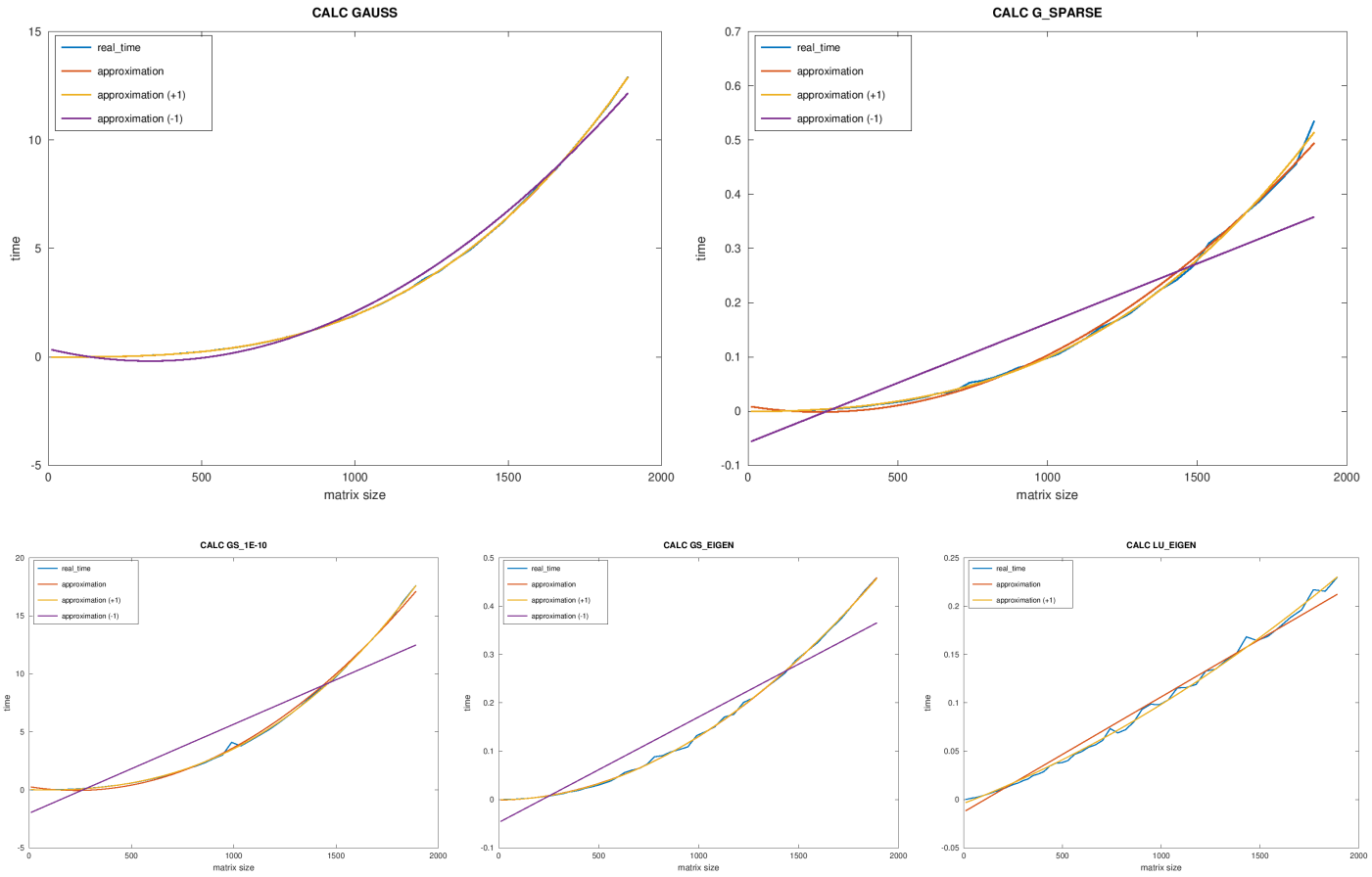




3 Aproksymacja

3.1 Wyliczone współczynniki wielomianów

- Rozwiązywanie układu równań:
  1. (**G**): $f(x) = 1.88345\text{e}{-9}x^3 + 5.25579\text{e}{-8}x^2 + (-1.30456\text{e}{-5})x^1 + 0.00124971$
  2. (**G_SPARSE**): $f(x) = 1.82546\text{e}{-7}x^2 + (-8.8859\text{e}{-5})x^1 + 0.00941105$
  3. (**GS_1E10**): $f(x) = 6.21178\text{e}{-6}x^2 + (-0.00284503)x^1 + 0.279666$
  4. (**GS_EIGEN**): $f(x) = 1.25348\text{e}{-7}x^2 + 6.14431\text{e}{-6}x^1 - 0.00112571$
  5. (**LU_EIGEN**): $f(x) = 0.000119035x^1 - 0.0127681$
- Generowanie macierzy:
  1. (**G**): $f(x) = -2.06199\text{e}{-12}x^3 + 2.9385\text{e}{-8}x^2 + -2.30313\text{e}{-6}x^1 + 0.000161398$
  2. (**G_SPARSE**): $f(x) = 2.54238\text{e}{-8}x^2 + (-1.99597\text{e}{-6})x^1 - 0.000262153$
  3. (**GS_1E10**): $f(x) = 2.4649\text{e}{-8}x^2 + (-1.68417\text{e}{-7})x^1 + 5.40007\text{e}{-6}$
  4. (**GS_EIGEN**): $f(x) = 3.39916\text{e}{-8}x^2 + 5.10909\text{e}{-8}x^1 - 0.000618658$
  5. (**LU_EIGEN**): $f(x) = 6.26423\text{e}{-5}x^1 - 0.0134335$



3.2 Poprawność wyników aproksymacji – średnie błędy bezwzględne

	Obliczanie	Generowanie
G	0.116641068509220541	0.006139134153799684
G_SPARSE	0.061993311882769679	0.006427472787361275
GS_1E10	1.467795323159247101	0.011466237689022329
GS_EIGEN	0.019821774223782174	0.013693956132461119
LU_EIGEN	0.056598547270937556	0.076729386139618563

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	1297.8903746183393650 43
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 58.611972999999998990 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