

# Aproksymacja średniokwadratowa dyskretna

Grams, Stanisław  
Jeziński, 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

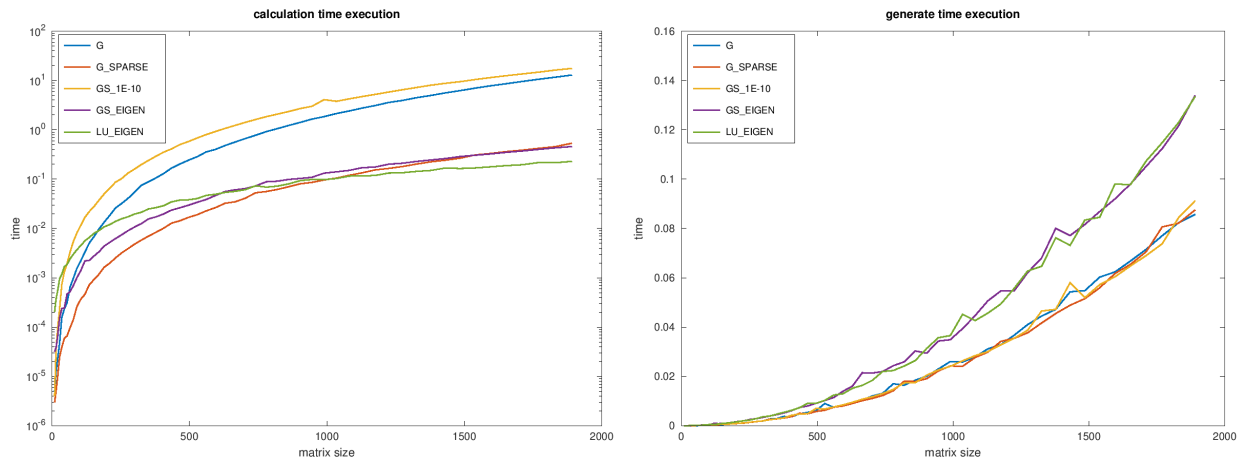
- (**G**): Algorytm Gaussa z częściowym wyborem elementu
- (**G\_SPARSE**): Algorytm Gaussa z częściowym wyborem elementu i optymalizacją dla macierzy rzadkich
- (**GS\_1E10**): Algorytm Gaussa-Seidela (precyzja  $1 \times 10^{-10}$ , struktura macierzy tablicowa)
- (**GS\_EIGEN**): Algorytm Gaussa-Seidela z implementacją dla biblioteki *Eigen3* (precyzja  $1 \times 10^{-10}$ ) <sup>1</sup>
- (**LU\_EIGEN**): Algorytm SparseLU pochodzący z biblioteki *Eigen3* <sup>2</sup>

W celu obsługi metod **GS\_EIGEN** oraz **LU\_EIGEN** opartych o bibliotekę *Eigen3* 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  $N = 3..60$ .

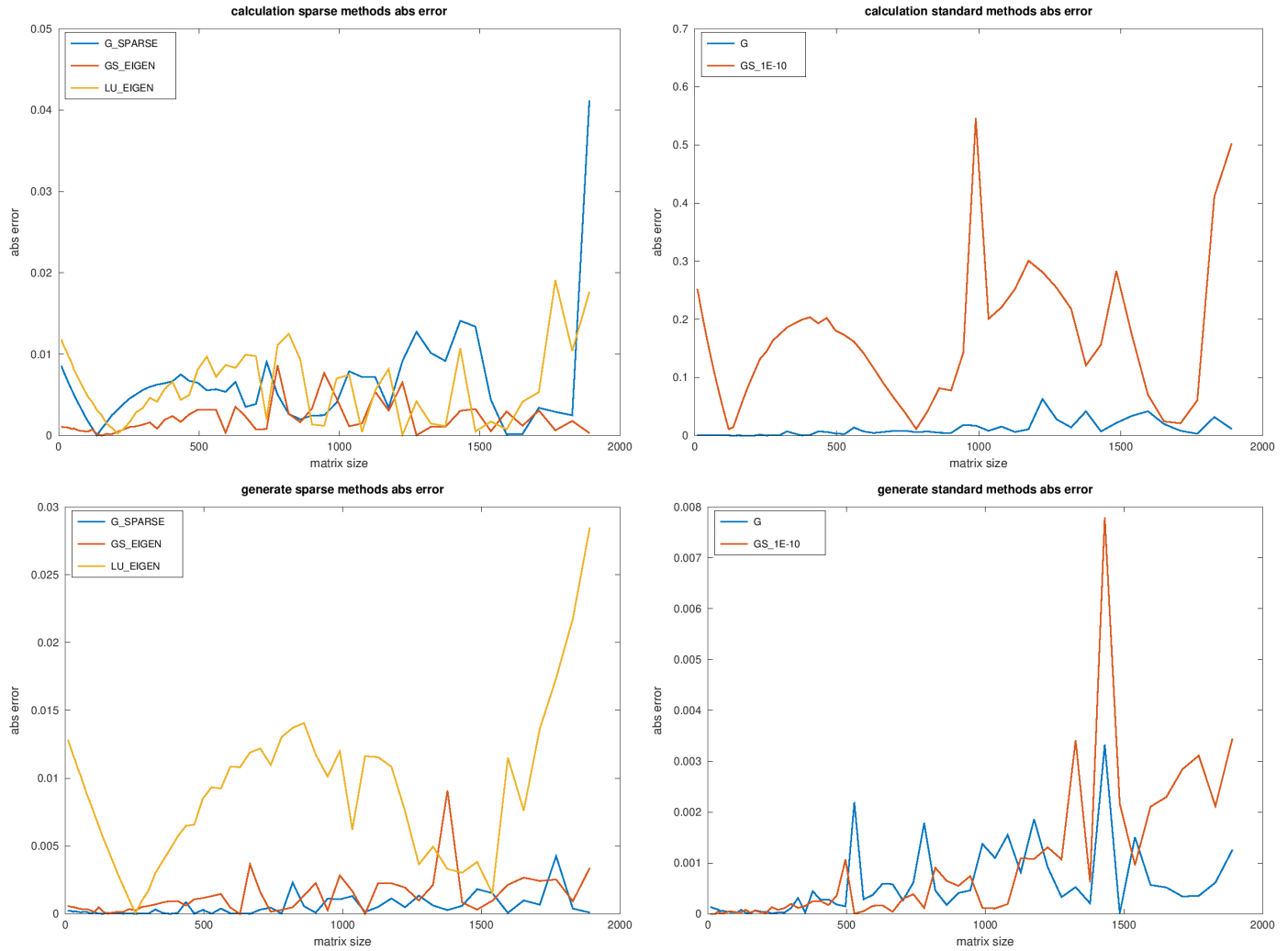
**Przypomnienie:** rząd macierzy można wyznaczyć ze wzoru  $A = \frac{(N+1)*(N+2)}{2}$

## 2 Analiza wyników



<sup>1</sup><http://komi.web.elte.hu/elektronikus/src/p184-koester.pdf>

<sup>2</sup>[https://eigen.tuxfamily.org/dox/classEigen\\_1\\_1SparseLU.html](https://eigen.tuxfamily.org/dox/classEigen_1_1SparseLU.html)



### 3 Aproksymacja

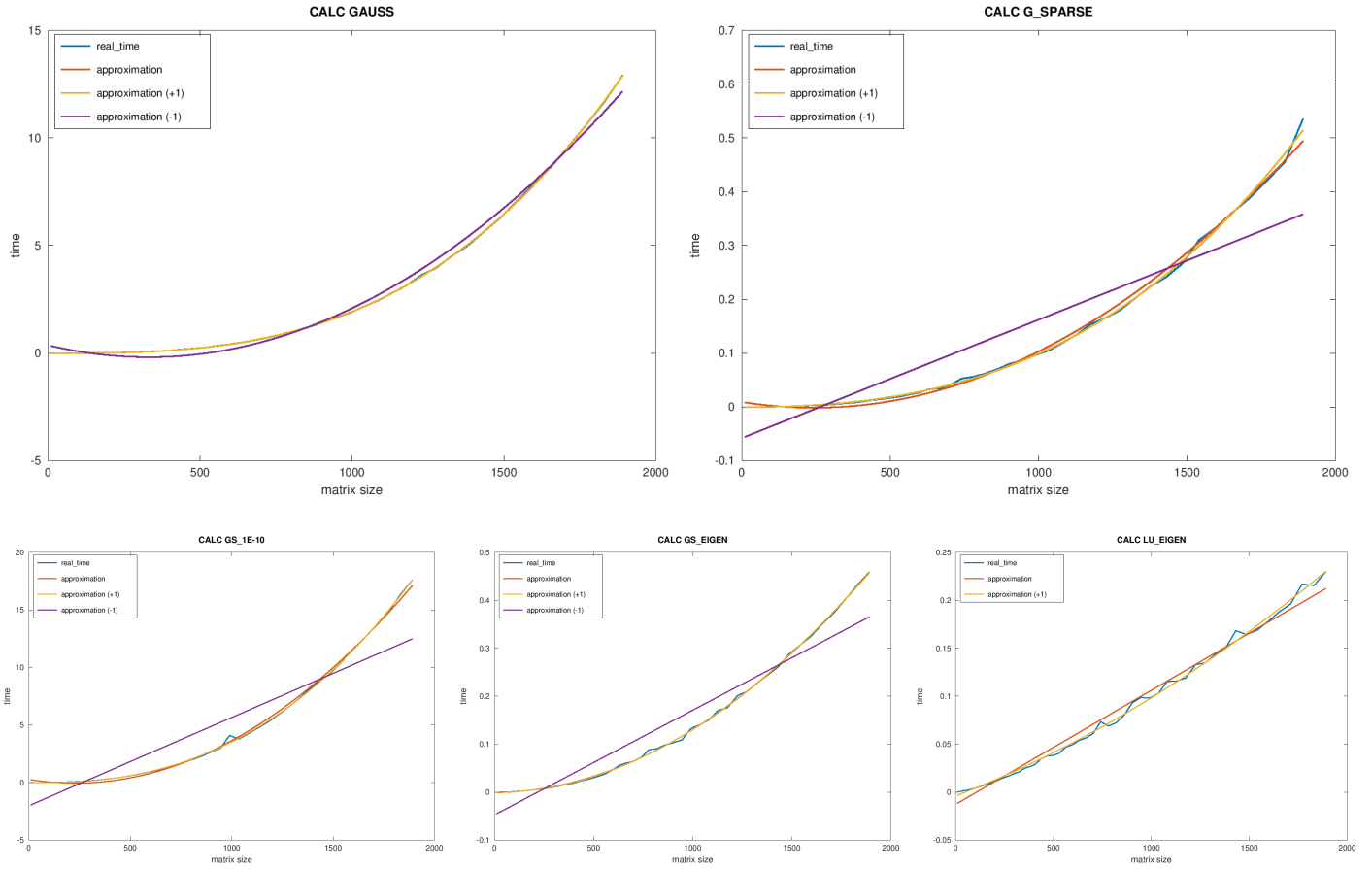
#### 3.1 Wyliczone współczynniki wielomianów

- Rozwiązanie układu równań:

1. **(G)**:  $f(x) = 1.88345e-9x^3 + 5.25579e-8x^2 + (-1.30456e-5)x^1 + 0.00124971$
2. **(G\_SPARSE)**:  $f(x) = 1.82546e-7x^2 + (-8.8859e-5)x^1 + 0.00941105$
3. **(GS\_1E10)**:  $f(x) = 6.21178e-6x^2 + (-0.00284503)x^1 + 0.279666$
4. **(GS\_EIGEN)**:  $f(x) = 1.25348e-7x^2 + 6.14431e-6x^1 - 0.00112571$
5. **(LU\_EIGEN)**:  $f(x) = 0.000119035x^1 - 0.0127681$

- Generowanie macierzy:

1. **(G)**:  $f(x) = -2.06199e-12x^3 + 2.9385e-8x^2 + -2.30313e-6x^1 + 0.000161398$
2. **(G\_SPARSE)**:  $f(x) = 2.54238e-8x^2 + (-1.99597e-6)x^1 - 0.000262153$
3. **(GS\_1E10)**:  $f(x) = 2.4649e-8x^2 + (-1.68417e-7)x^1 + \exp 5.40007-6$
4. **(GS\_EIGEN)**:  $f(x) = 3.39916e-8x^2 + 5.10909e-8x^1 - 0.000618658$
5. **(LU\_EIGEN)**:  $f(x) = 6.26423e-5x^1 - 0.0134335$



### 3.2 Poprawność wyników aproksymacji – średnie 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 aproksymację 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 63.291921000000002095 i jest około 4.9x gorszy od aproksymowanego. Metody oparte o bibliotekę *Eigen3* 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 <i>Eigen3</i>	Przygotowanie wykresów oraz tabel	Agregacja uzyskanych wyników
Implementacja wywołania poszczególnych prób ( <i>main.cc</i> )	Uruchomienie prób i testowanie	Praca nad strukturą projektu