Wstęp

Z punktu widzenia technik rozwiązywania układów równań liniowych istnieją dwa zasadniczo różne typy macierzy układu:

◦ Macierze gęste (ang. dense matrix) – z małą liczbą elementów zerowych

◦ Macierze rzadkie (ang. sparse matrix) – z dużą liczbą elementów zerowych

Rozróżnienie nie jest ścisłe i w praktyce sprowadza się do rozstrzygnięcia czy dla danej macierzy bardziej optymalne będzie przechowywanie w standardowym formacie (wiersz po wierszu lub kolumna po kolumnie), czy w jednym ze specjalnych formatów, gdzie przechowywane są głównie lub wyłącznie wyrazy niezerowe.

Typowymi formatami przechowywania macierzy rzadkich są: CRS (Compressed Row Storage), BCRS (Block Compressed Row Storage), CCS (Compressed Column Storage) i wiele innych.

W programach MES głównie występują macierze rzadkie. Zadaniem Państwa podczas dzisiejszych ćwiczeń będzie badanie postaci macierzy tworzonych w programach MES oraz analizowanie algorytmu i wydajności solwera liniowego (programu do rozwiązywania układów równań liniowych) Pardiso ( https://www.pardiso-project.org/ ) , w jego wersji opracowanej przez firmę Intel i dostarczanej w ramach biblioteki MKL (math kernel library , https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html#gs.yyybdr ). W ramach ćwiczeń utworzycie Państwo wizualizacje macierzy sztywności dla różnych przykładowych problemów. W trakcie zajęć wykonane zostanie również przenumerowanie macierzy (and. renumbering) – technika ta pozwala na ułożenie niezerowych wyrazów macierzy w taki sposób, aby zoptymalizować działanie solwera liniowego. Jako przykładowa metoda przenumerowania stosowany będzie algorytm Reverse Cuthill–McKee, grupujący wyrazy macierzy jak najbliżej diagonali.

Zadanie 1.

Utworzenie katalogu lab_08

Skopiowanie plików z poprzednich laboratoriów, **A2.jk, problem_heat.dat, bc_heat.dat**.

W przypadku skopiowania z innych laboratoriów należy upewnić się czy w pliku jk jest zwiększona grubość i ilość warstw.

Zmiana nazwy problemu w **problem_heat.dat**.

Sprawdzenie działania programu ModFEM, trzykrotna adaptacja siatki (m), w przypadku wyskoczenia błędu edycja pierwszej linijki w pliku jk poprzez dodanie zer

np. `100000 400000 500000 200000.`

Rozwiązanie zadania (s), generacja pliku PARAVIEW(v), wyjście z programu (q).

Modyfikacja pliku `modfem/src/sid_mkb/sis_mkb_intf.c` poprzez komentowanie i odkomentowanie linijek

```
71   #define PRINT_MATRIX // lab 08
72   #undef RENUMBERING // lab 08
73   //#define RENUMBERING // lab 08
```

Możliwe jest także sprawdzenie poprawności poprzez dodanie wypisania

```
268   #ifdef PRINT_MATRIX
269   printf("PRINT_MATRIX symbol defined\n");
270   #endif
271   #ifdef RENUMBERING
272   printf("RENUMBERING symbol defined\n");
273   #endif
```

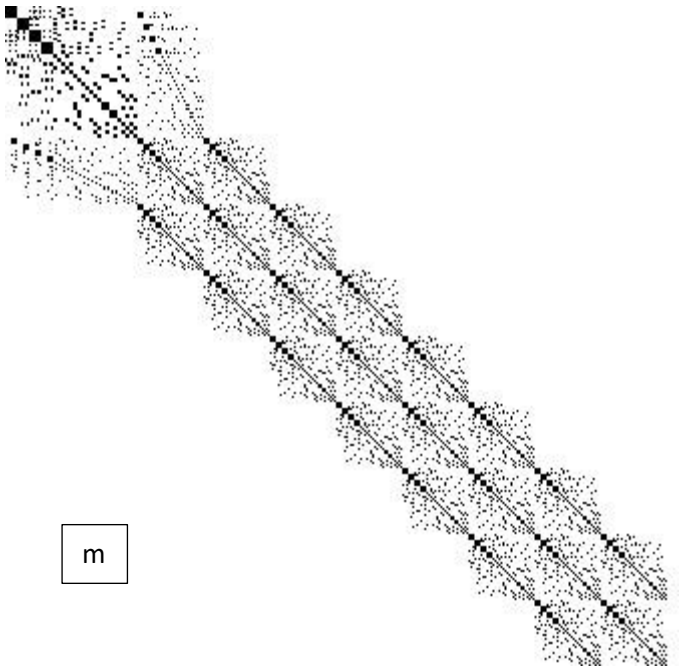
Po edycji pliku należy dokonać rekompilacji kodu poprzez komendę `make` (można to zrobić bez opcji –j4 w celu znalezienia ewentualnych błędów.

## Uruchomienie ModFem'a

`~/ModFEM/bin_cmake/imie_nazwisko_nompi_none_gcc_g++/MOD_FEM_heat_prism_std .`

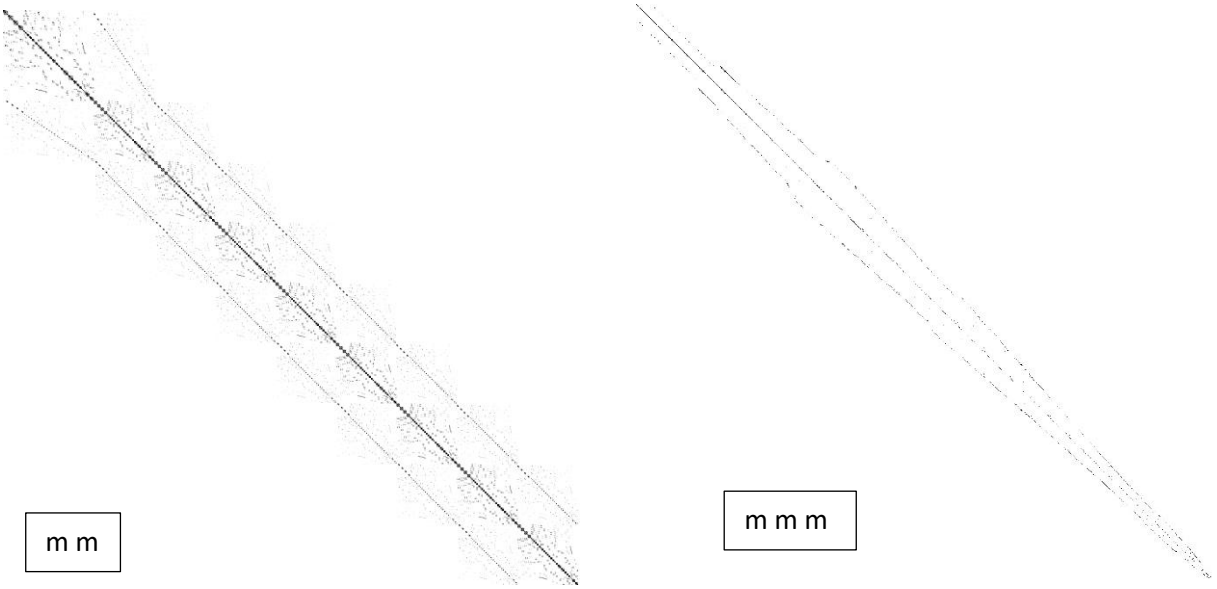
Wybranie opcji 's'.

W efekcie powstaje plik `sm_matrix.pbm` który przedstawia strukturę macierzy układu równań liniowych – jeden czarny piksel odpowiada niezerowemu wyrazowi macierzy sztywności.



m

Macierz należy wygenerować jeszcze dwukrotnie poprzez użycie opcji ‘m’, ‘s’ oraz ‘m’, ‘m’, ‘s’.




Zadanie 2.

**(Po modyfikacji pliku pozostałe kroki pozostają bez zmian)**

Modyfikacja pliku `modfem/src/sid_mkb/sis_mkb_intf.c` poprzez komentowanie i odkomentowanie linijek

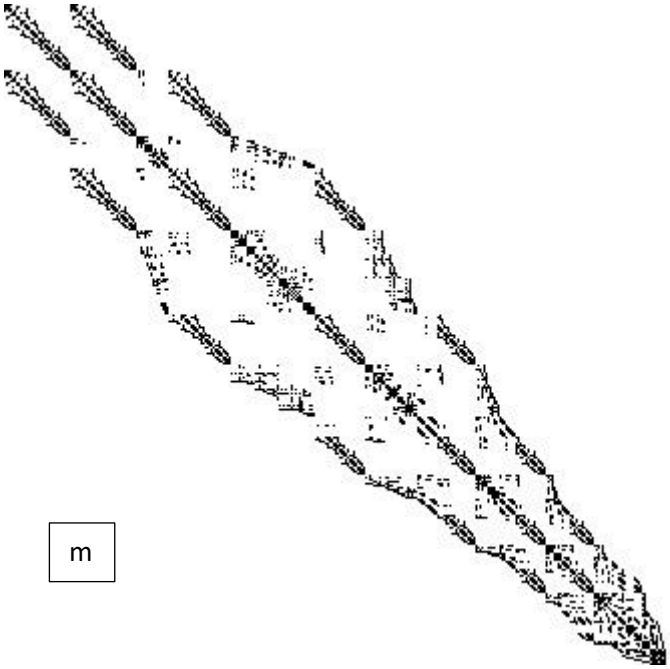
```
71    #define PRINT_MATRIX // lab 08
72    //#undef RENUMBERING // lab 08
73    #define RENUMBERING // lab 08
```

Po edycji pliku należy dokonać rekompilacji kodu poprzez komendę `make` (można to zrobić bez opcji –j4 w celu znalezienia ewentualnych błędów.

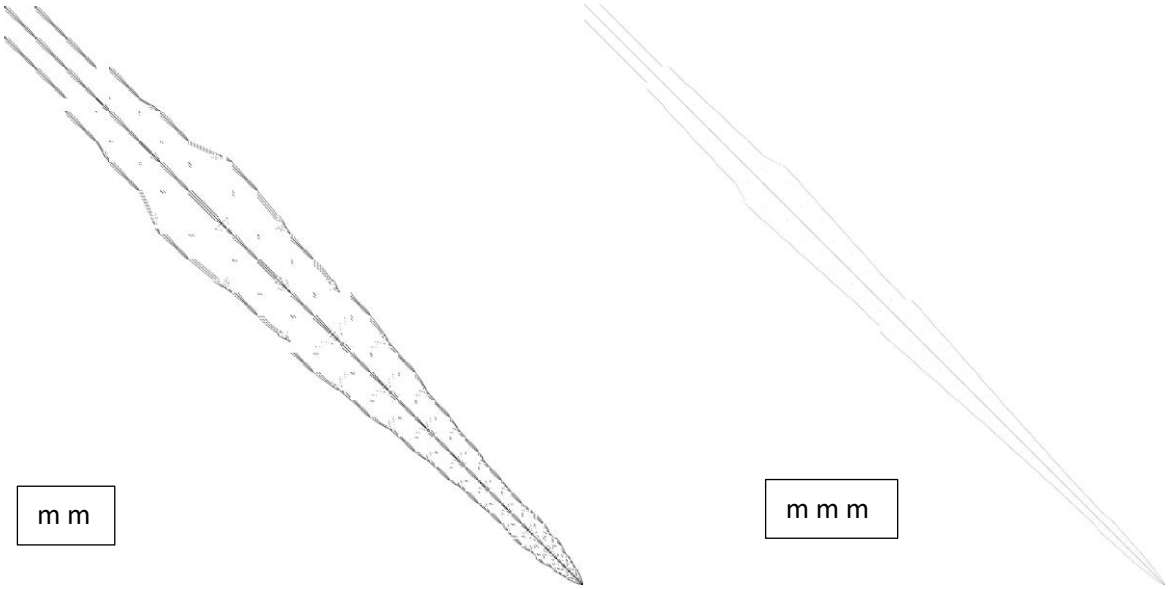
Uruchomienie ModFem’a

`~/ModFEM/bin_cmake/imie_nazwisko_nompi_none_gcc_g++/MOD_FEM_heat_prism_std .`

Wybranie opcji ‘s’.




Macierz należy wygenerować jeszcze dwukrotnie poprzez użycie opcji ‘m’, ‘s’ oraz ‘m’, ‘m’, ‘s’.




Zadanie 3.

Modyfikacja pliku `modfem/src/sid_mkb/sis_mkb_intf.c` poprzez zakomentowanie linijki

```
71    //#define PRINT_MATRIX // lab 08
72    //#undef RENUMBERING // lab 08
73    #define RENUMBERING // lab 08
```

Po edycji pliku należy dokonać rekompilacji kodu poprzez komendę `make` (można to zrobić bez opcji –j4 w celu znalezienia ewentualnych błędów.

Uruchomienie ModFem’a

`~/ModFEM/bin_cmake/imie_nazwisko_nompi_none_gcc_g++/MOD_FEM_heat_prism_std .`

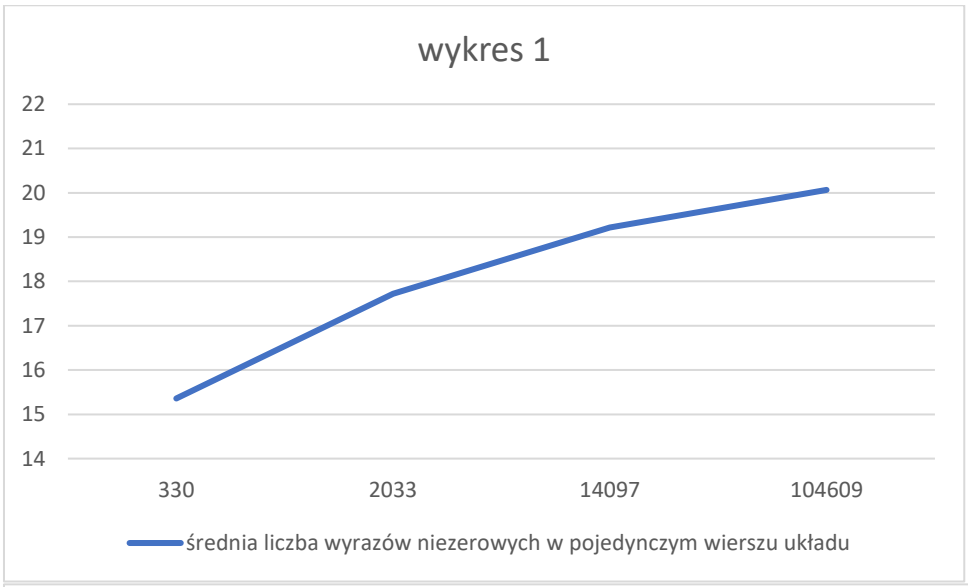
Wybranie opcji ‘s’

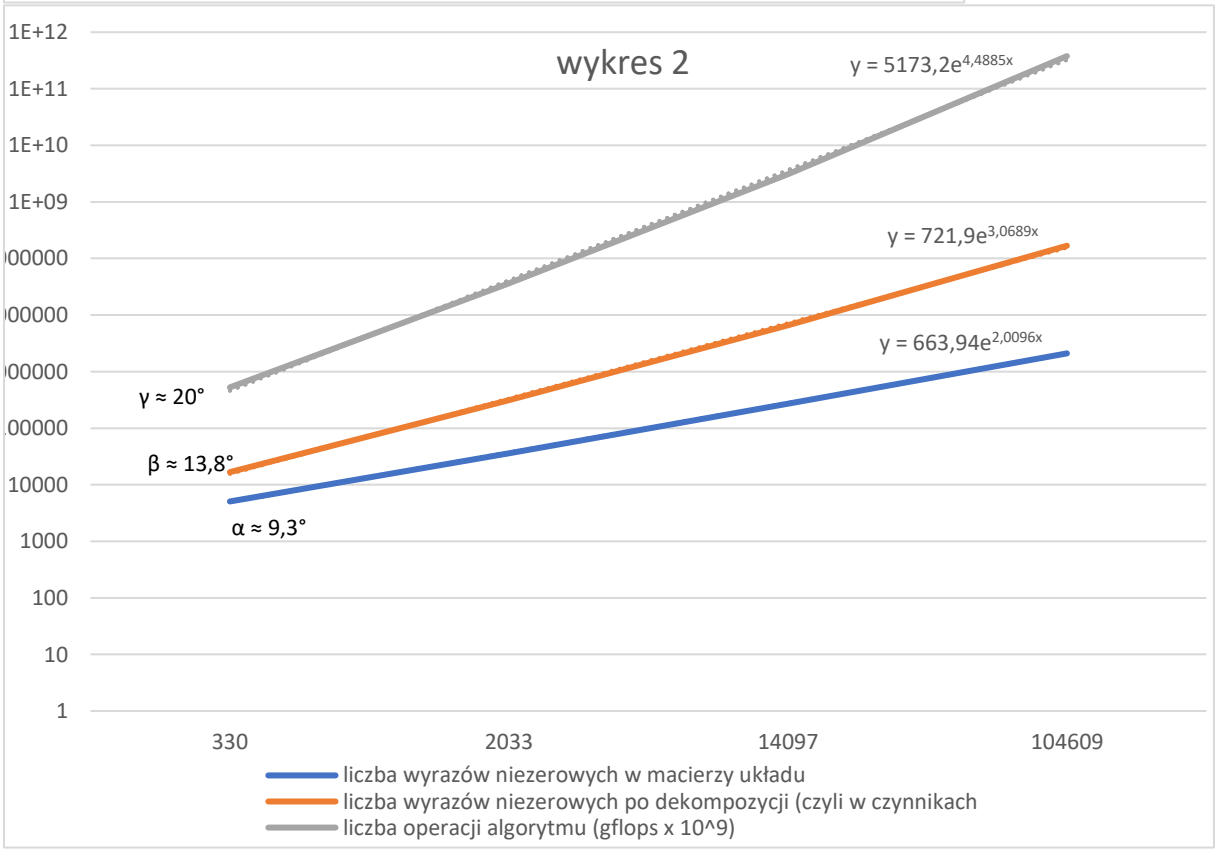
Skopiowanie wyników do arkusza Excel.

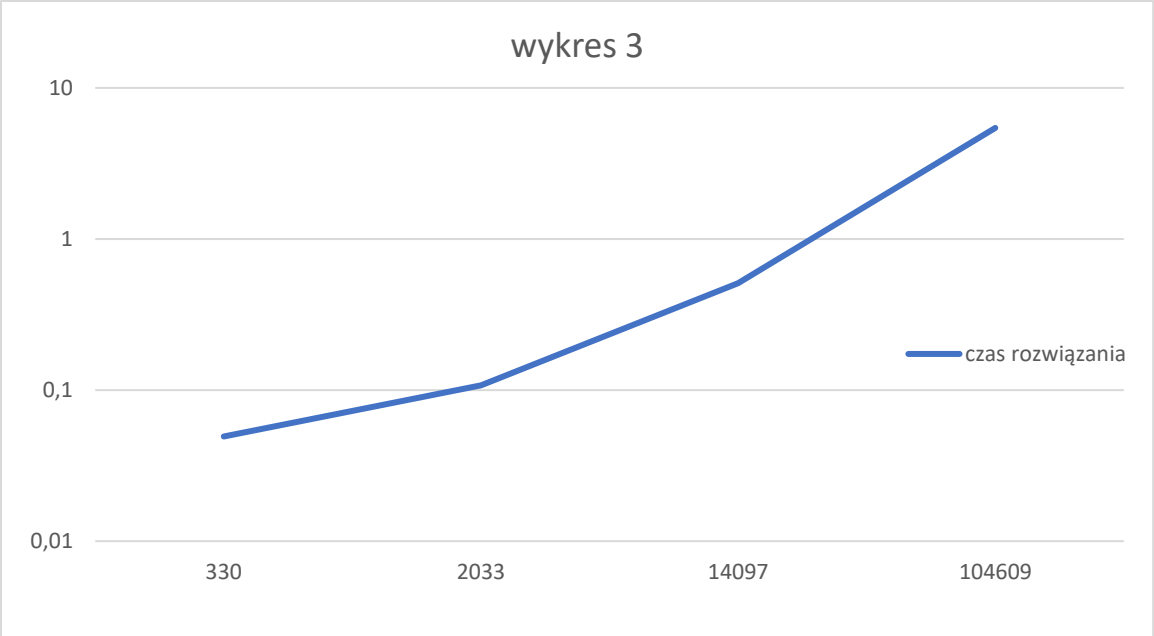
```
Statistics:
===========
Parallel Direct Factorization is running on 24 OpenMP

< Linear system Ax = b >
             number of equations:           330
             number of non-zeros in A:      5068
             number of non-zeros in A (%): 4.653811

             number of right-hand sides:    1

< Factors L and U >
< Preprocessing with multiple minimum degree, tree height >
< Reduction for efficient parallel factorization >
             number of columns for each panel: 128
             number of independent subgraphs:  0
             number of supernodes:                    146
             size of largest supernode:               50
             number of non-zeros in L:                10321
             number of non-zeros in U:                6393
             number of non-zeros in L+U:              16714
             gflop   for the numerical factorization: 0.000527

             gflop/s for the numerical factorization: 0.008899

Running PARDISO finished with result: 0 (0 - means no error)
Number of threads used by PARDISO: 1
Solution in struct 0, block 17, dof_ent 7, posglob=17
      300.0000002082
Solution in struct 6, block 16, dof_ent 1, posglob=16
      300.0000003956
Solution in struct 19, block 22, dof_ent 8, posglob=22
      300.0000003964
Solution in struct 38, block 5, dof_ent 2, posglob=5
      300.0000001794
Solution in struct 40, block 28, dof_ent 9, posglob=28
      300.0000001794
Solution in struct 42, block 9, dof_ent 3, posglob=9
      300.0000002087
Solution in struct 48, block 7, dof_ent 5, posglob=7
      300.0000001074
Solution in struct 52, block 25, dof_ent 6, posglob=25
      300.0000001074
Solution in struct 62, block 0, dof_ent 4, posglob=0
      300.0000001111

Entering lsr_mkb_free_matrix: Solver_type 0, Solver id 0

Entering lar_free_SM_and_LV: storage_type: 2
matrix id: in interface 0, internally in implementation 0

Entering lsr_mkb_destroy: Solver_type 0, Solver id 0

Leaving linear solver 0 (sir_destroy)

Time total: 0.171892
```

Ponowne odpalenie programu z opcjami ‘m’,’s’; ‘m’,’m’,’s’; ’m’,’m’,’m’,’s’.

| m | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| number of equations | 330 | 2033 | 14097 | 104609 |
| number of non-zeros in A | 5068 | 36025 | 270865 | 2099041 |
| Average | 15,35757576 | 17,72011805 | 19,21437185 | 20,06558709 |
| number of non-zeros in L+U | 16714 | 313643 | 6561205 | 168129053 |
| gflop | 527000 | 36386000 | 3079660000 | 377607014000 |
| Time total | 0,049319 | 0,107463 | 0,508605 | 5,435543 |






Wnioski 3.7

Liczba operacji oraz wyrazów niezerowych w macierzy zwiększa się wraz z ilością adaptacji siatki, ponieważ powoduje ona rozdrobnienie na mniejsze elementy.

Wraz z każdą adaptacją liczba wyrazów niezerowych w każdym wierszu wzrasta, a w całej macierzy jest to 7-8 krotny wzrost. Jednocześnie procentowa ilość wyrazów niezerowych spada 5-7 krotnie. Ilość wyrazów niezerowych w macierzy po dekompozycji LU znacząco wzrasta.

Z wykresu 2 można wywnioskować to, że rząd zależności wielkości zadania od liczby stopni swobody może mieć istotny wpływ na możliwości obliczeniowe jest to spowodowane tym że wzrost liczby niezerowych elementów po dekompozycji LU jest bardzo szybki. Efektem tego może być znaczne przekroczenie wymaganej mocy obliczeniowej, czego efektem może być zapchanie pamięci maszyny i konieczność jej resetu.

Wykorzystywana macierz jest kwadratowa, dlatego jej wymiary to n x n.
W skutek dekompozycji otrzymywane są macierze trójkątne, górna U i dolna L.
Każda z powstałych macierzy ma n(n+1)/2 wyrazów niezerowych.
Macierz po dekompozycji LU ma n(n+1) wyrazów niezerowych.

Liczba stopni swobody dla której przekroczone zostanie 1000GB
1000GB = $10^{12}$B = 1 000 000 000 000B
$10^{12}$/8 = 125*10^9 = 125 000 000 000

1000GiB (błędnie mylone z GB) = $1024^3$*1000B = 1 073 741 824 000B
1 073 741 824 000/8 = 134 217 728 000

a = 125 000 000 000 v 134 217 728 000

n(n+1) = $n^2$+n => $n^2$+n-a = 0

$n_1$ = 366 356,88 $n_2$ = 353 552,89

Dla liczby swobody większej od 366 357 lub 353 553, zależnie od nomenklatury, pojemność pamięci wymaganej do przechowywania macierzy przekroczy 1000GB.

Złożoność obliczeniowa metody eliminacji Gaussa wynosi O(n^3) gdzie n to liczba równań.

Przy założeniu że wydajność wynosi 600 Gflop/s a czas który ma rpzekroczyć rozwiązanie układu równań to 1 godzina, można wyliczyć jakie ma być jego rozmiar poprzez pomnożenie ilości maksymalnego czasu przez wydajność.

1h = 60min = 3600s

600Gflops/s = 600 000 000 000flops/s

$2{,}16 * 10^{17}$ flops.

$n^3 <= 2{,}16 * 10^{17}$

n ≈ 129 266,08

Maszyna o wydajności 600Gflops/s w ciągu godziny może policzyć zadanie o maksymalnym rozmiarze wynoszącym 129 266

Zadanie 4.

Powtórzenie zadania 3 z wybraniem opcji z na końcu, np. ‘m’, ‘s’, ‘z’.

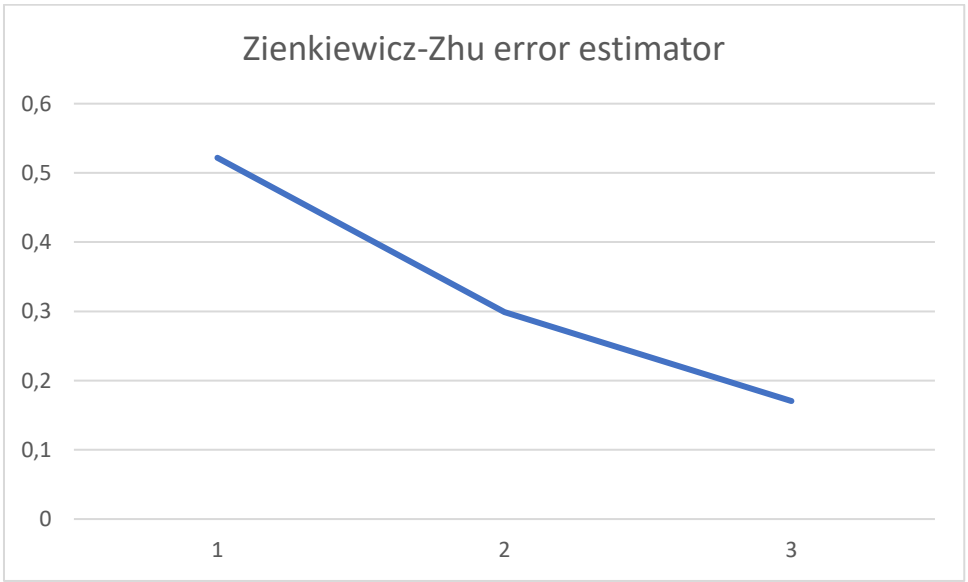
W efekcie otrzymuje się estymator błędu.

```
Patch creation - time 0.003586
Derivative recovery - time 0.047217

Zienkiewicz-Zhu error estimator = 0.521855685313
```

Po odpaleniu programu i zapisaniu wyników dla 1, 2 i 3 ‘m’ należy zrobić wykres.

Wartości mogą się różnić, zakres może obejmować liczby 1 – 0,1, jak i 200 – 50.

| m | 1 | 2 | 3 |
|---|---|---|---|
| Zienkiewicz-Zhu | 0,521855685 | 0,29871882 | 0,17048414 |




Wartość błędu spada wraz z ilością adaptacji siatki

Zadanie 5.

Wnioski

Poprzez adaptację siatki, rozmiar trójkątów na które jest podzielona się zmniejsza, zwiększając ich liczbę oraz liczbę węzłów.

Układ można rozwiązać, za pomocą metody eliminacji Gaussa.

Czas obliczania większych macierzy znacząco rośnie poprzez złożoność obliczeniową O(n^3)). Przez to nie wszystkie układy da się rozwiązać w rozsądnym czasie (przy bardzo wielkich macierzach).

| Zadanie – lab 08 | OCENA własna w % (0-100) | OCENA prowadzącego w % (0-100) |
|---|---|---|
| Zad. 1 Modyfikacja pliku źródłowego i rekompilacja kodu | 100 | |
| Zad. 1 Trzykrotne uruchomienie programu i pobranie informacji o strukturze macierzy | 100 | |
| Zad. 1 Wizualizacja struktury trzech macierzy układu równań liniowych | 100 | |
| Zad. 2 Modyfikacja pliku źródłowego (włączenie przenumerowania) i rekompilacja kodu | 100 | |
| Zad. 2 Trzykrotne uruchomienie programu i pobranie informacji o strukturze macierzy | 100 | |
| Zad. 2 Wizualizacja struktury trzech macierzy układu równań liniowych | 100 | |
| Zad. 3 Modyfikacja pliku źródłowego (wyłączenie drukowania macierzy) i rekompilacja kodu | 100 | |
| Zad. 3 Uruchomienie programu z trzykrotną adaptacją i pobranie informacji o strukturze macierzy | 100 | |
| Zad. 3 Analiza danych solwera bezpośredniego dla czterech rozwiązywanych układów równań liniowych | 100 | |
| Zad. 4 Obserwacja zmian oszacowania błędu metodą ZZ wraz z adaptacją siatki | 100 | |
| ŁĄCZNIE (1000): | 1000 | |
| OCENA KOŃCOWA: | ---------- ---------- | |