Advanced Computational Techniques

LAB 05

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 (ang. *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.

1 Zadanie 1 (obowiązkowe) – badanie wpływu przenumerowania na postać (strukturę wyrazów niezerowych) macierzy układów równań liniowych MES

1.1 Utworzenie katalogu roboczego *lab_08*

1.2 W katalogu roboczym utworzenie podkatalogu o nadanej przez siebie nazwie

1.3 Skopiowanie do podkatalogu plików konfiguracyjnych z dowolnego działającego zadania 3D

1.3.1 przed przystąpieniem do właściwego ćwiczenia należy sprawdzić działanie kodu dla przyjętego pliku siatki

- w katalogu problemowym uruchomić program ModFEM (istotne jest aby pracować na sprawdzonym zadaniu, które wykonuje się poprawnie)
  - **należy nadać nową nazwę zadania, np. *zadanie_test_solwerow*, tak, żeby nie włączały się warianty dokonanych wcześniej modyfikacji kodu (jest to istotne dla porównania z solwerami iteracyjnymi w ramach następnego laboratorium)**
- dokonać trzykrotnej jednorodnej adaptacji siatki
  [W przypadku informacji ze strony programu o braku miejsca w pamięci należy zmodyfikować pierwszą linijkę w pliku siatki

...jk – np. zwiększając dziesięciokrotnie każdą z wartości, przykładowa linijka:
100000 400000 500000 200000
(Uwaga: **nie należy przekraczać wartości 100 000 dla pierwszego parametru – liczby wierzchołków siatki**)]

- sprawdzić, że liczba węzłów dla ostatniej czwartej siatki mieści się w granicach 50 000-100 000 (siatka początkowa powinna mieć mniej więcej pomiędzy 200 a 300 węzłów – **tę wartość można uzyskać przez zadanie odpowiedniej liczby warstw w pliku ...jk**)
- rozwiązać zadanie (opcja '**s**') - należy sprawdzić, że do rozwiązywania układu równań liniowych został użyty solwer PARDISO (pomiędzy wydrukami powinny znaleźć się informacje solwera:

=== PARDISO: solving a real structurally symmetric system ===

itd. itp.)

- *dodatkowo w celu jakościowego sprawdzenia poprawności obliczeń można także wygenerować pliki Paraview (opcja '**v**'), a następnie w sposób standardowy przeglądnąć wyniki.*

1.4 Modyfikacja interfejsu z solwerem liniowym. W katalogu **modfem2015/src/sid_mkb/sis_mkb_intf.c** należy otworzyć w edytorze plik **sis_mkb_intf.c** i znaleźć miejsce definiowania symbolu **PRINT_MATRIX** (linia ok. 70, domyślnie definiowanie jest wykomentowane). Należy zdefiniować ten symbol oraz spowodować niezdefiniowanie symbolu **RENUMBERING**:

```
#define PRINT_MATRIX – ok linii 71
...
#undef RENUMBERING – ok. linii 72
```

[dla zapewnienia poprawnego działania kodu można umieścić po definiowaniu symboli sprawdzenie w postaci np.

```
#ifdef PRINT_MATRIX
  printf("PRINT MATRIX defined\n");
#else
  printf("PRINT MATRIX NOT defined\n");
#endif
```

i podobnie dla **RENUMBERING**
]

1.5 Następnie należy skompilować kod wykorzystując program **make**:

```
make
```

(**make** bez opcji -j korzysta z jednego wątku, co często ułatwia znalezienie błędów w kompilowanym kodzie)

Przypomnienie:

- *katalog **~/ModFEM/** powinien być katalogiem, w którym przez wszystkie zajęcia przechowujecie Państwo kod źródłowy programu, w którym dokonujecie modyfikacji kodu oraz jego kompilacji i rekompilacji*
- *standardowo uruchomienie kodu (np. **MOD_FEM_heat_prism_std**) powinno odbywać się zawsze poprzez wykonanie polecenia:*

```
~/ModFEM/modfem2015/bin_cmake/ACT_nompi_none_gcc_g++/MOD_FEM_heat_prism_std .
```

w katalogu problemowym (zawierającym pliki konfiguracyjne rozwiązywanego zadania)

- *w przypadku aktualnego lab 08 można kopiować uzyskany pliku binarny (np.* ***MOD_FEM_heat_prism_std)*** *do katalogu z przykładem, dokonując odpowiedniej zmiany nazwy:*

```
cp ~/ModFEM/bin_cmake/ACT_nompi_none_gcc_g++/MOD_FEM_heat_prism_std
./MOD_FEM_heat_prism_std_print_matrix
```

a następnie uruchamiać utworzony program:

```
./MOD_FEM_heat_prism_std_print_matrix .
```

- *argumentem wywołania konkretnego programu w ModFEM jest katalog roboczy z plikami problemowymi, argument można pominąć kiedy uruchamia się plik binarny z katalogu problemowego*

1.6 Uruchomienie programu ModFEM

- Po dokonaniu modyfikacji pliku źródłowego i rekompilacji programu należy uruchomić program do symulacji zagadnienia rozchodzenia się ciepła w 3D
- Po uruchomieniu programu należy wybrać opcję rozwiązania pojedynczego zadania stacjonarnego '**s**' (bez wcześniejszej adaptacji siatki!)
- proszę na wydruku odnaleźć informację o kolejnych krokach realizowanej procedury rozwiązywania pojedynczego stacjonarnego zadanie MES:

```
Beginning solution of a single heat problem
...
Entering linear solver module
...
Allocated CRS matrix 0: nglob=..., nnz=..., half bandwidth = ...
...
```

- bezpośrednio przed rozpoczęciem wykonywania procedur solwera liniowego wyświetlane są informacje o formacie przechowywania siatki oraz o globalnych parametrach macierzy układu (liczba niewiadomych, liczba niezerowych elementów macierzy, największa odległość wyrazu poza przekątną od przekątnej głównej), w postaci:

```
Initialized CRS matrix 0: n = ..., nnz = ..., half bandwidth = ...
The average number of non-zeros in a single row: ...
```

[powyższa informacja powinna znaleźć się w sprawozdaniu z laboratorium dla każdej z badanych macierzy (także po przenumerowaniu)]

1.7 Po zdefiniowaniu symbolu **PRINT_MATRIX** program po dokonaniu całkowania numerycznego i agregacji globalnej macierzy układu (macierzy sztywności) produkuje plik **sm_matrix.pbm** ilustrujący strukturę macierzy układu równań liniowych – jeden czarny piksel odpowiada niezerowemu wyrazowi macierzy sztywności

1.8 Otrzymany plik należy skopiować pod jednoznacznie identyfikującą problem nazwą (oraz pobrać na swój lokalny komputer i oglądnąć dowolnym programem graficznym) – obrazek z pliku powinien także znaleźć się w sprawozdaniu.

1.8.1 dla dużych plików więcej szczegółów uzyskuje się po odpowiednim powiększeniu obrazka

1.9 Powyższe kroki (uruchomienie, rozwiązanie, znalezienie informacji o parametrach macierzy układu, skopiowanie pliku **sm_matrix.pbm**) należy powtórzyć, tym razem dla siatki otrzymanej przez jednorodną adaptacje siatki początkowej (w menu głównym opcja '**m**' z wartością -1 przed uruchomieniem solwera opcją '**s**').

1.10 Ostatnim krokiem jest powtórzenie wszystkich powyższych kroków dla siatki dwukrotnie zaadaptowanej.

2 Zadanie 2 (obowiązkowe): Badanie wpływu przenumerowania na własności macierzy.

2.1 W katalogu **modfem2015/src/sid_mkb/sis_mkb_intf.c** należy otworzyć w edytorze plik **sis_mkb_intf.c** i tym razem spowodować zdefiniowanie symbolu **RENUMBERING** (zostawiając zdefinowany symbol **PRINT_MATRIX):**
#define PRINT_MATRIX – ok. linii 71

2.1.1 zdefiniowanie symbolu **RENUMBERING** może odbyć się na 2 sposoby:

- jeśli w pliku konfiguracyjnym ***ACT.cmake*** ustawiona jest opcja `set(ENABLE_RENUMBERING TRUE)` **(tak powinno być zgodnie z instrukcją lab 1)** wystarczy wykomentować linię powodującą niezdefiniowanie symbolu **RENUMBERING**

// #undef RENUMBERING – ok. linii 73

- jeśli w pliku konfiguracyjnym ***ACT.cmake*** ustawiona jest opcja `set(ENABLE_RENUMBERING FALSE)` należy zdefiować symbol **RENUMBERING**

#define RENUMBERING

[ponownie dobrze jest sprawdzić poprawność zdefiniowania symboli dzięki wprowadzonym wydrukom z kompilacją warunkową]

2.2 Następnie należy skompilować kod – jak w p. 1.5

2.3 Proszę kolejno powtórzyć kroki opisane w p. 1.6-1.10 dla nowo utworzonej wersji kodu. W sprawozdaniu należy dokonać porównania obrazków i parametrów macierzy układu równań liniowych w przypadku niestosowania przenumerowania wierzchołków (w tym wypadku węzłów MES) i użycia algorytmu RCM.

3 Zadanie 3 (obowiązkowe). Badanie własności algorytmu rozwiązywania układów równań liniowych za pomocą dekompozycji LU.

3.1 W katalogu **modfem2015/src/sid_mkb/sis_mkb_intf.c** należy otworzyć w edytorze plik **sis_mkb_intf.c** i spowodować niezdefiniowanie symbolu **PRINT_MATRIX** (symbol **RENUMBERING** może pozostać zdefiniowany):
//#define PRINT_MATRIX – ok. linii 71

[ponownie dobrze jest sprawdzić poprawność zdefiniowania symboli dzięki wprowadzonym wydrukom z kompilacją warunkową - **pozostawienie zdefiniowanego symbolu PRINT_MATRIX powoduje błąd naruszenia ochrony pamięci dla dużych zadań**]

3.2 Po rekompilacji kodu proszę uruchomić zadanie i odnaleźć na wydruku informacje o działaniu algorytmu PARDISO dokonującego dekompozycji LU. Interesujące nas dane znajdują się wśród wielu wydruków PARDISO w sekcji STATISTICS.

```
Statistics:
===========
```

```
Parallel Direct Factorization is running on 20 OpenMP
< Linear system Ax = b >
             number of equations:           ...
             number of non-zeros in A:      ...
             number of non-zeros in A (%):  ...
             number of right-hand sides:    ...
< Factors L and U >
< Preprocessing with multiple minimum degree, tree height >
....
             number of non-zeros in L+U:              ...
             gflop   for the numerical factorization: ...
             gflop/s for the numerical factorization: ...
Running PARDISO finished with result: 0 (0 - means no error)
```

3.3 Po zapisaniu informacji o statystykach wykonania PARDISO proszę dokonać jednokrotnej adaptacji siatki (**m -1**) i ponownie rozwiązać zadanie (**s**) oraz odnaleźć statystyki dla rozwiązania na nowej siatce.

3.4 Powyższe czynności należy powtórzyć jeszcze dwukrotnie, tak aby ostatecznie uzyskać statystyki dla czterech siatek. Ostatnia siatka powinna mieć liczbę wierzchołków w zakresie 50 000 – 100 000 (nie należy przekraczać wartości 100 000 węzłów)

---------- 3.0 ----------

3.5 Uzyskane wyniki proszę zebrać w tabeli, a następnie na jej podstawie utworzyć wykresy zależności **od** liczby stopni swobody (liczby funkcji bazowych = liczby wierzchołków siatki MES = liczby węzłów MES = liczby równań w układzie równań liniowych) dla:

- wykres 1:
  - średniej liczby wyrazów niezerowych w pojedynczym wierszu układu
- wykres 2 (trzy krzywe):
  - liczby wyrazów niezerowych w macierzy układu
  - liczby wyrazów niezerowych po dekompozycji (czyli w czynnikach L i U)
  - liczby operacji algorytmu (dane z wydruku należy pomnożyć x 10^9)
- wykres 3:
  - czasu rozwiązania (na wydruku linijka `Time total` bezpośrednio przed menu głównym)

3.6 Uwaga: dla wykresów 2 i 3 należy zastosować skalę logarytmiczną na obu osiach

- **jeśli wykres zależności jest linią prostą to odpowiada to zależności potęgowej $y=a*x^n$**
- **kąt nachylenia linii określa wartość potęgi *n* czyli rzędu zależności (*log y = n log x + log a*)**
- **proszę odczytać z wykresu 2 jakiego rzędu są zależności poszczególnych badanych wielkości od liczby stopni swobody (zależność liniowa, kwadratowa, trzeciego rzędu itd.)**

---------- 4.0 ----------

3.7 Wyniki w tabeli oraz na wykresach należy zanalizować, a wynikające z tego wnioski umieścić w sprawozdaniu

- jaki wpływ na liczbę wyrazów niezerowych w macierzy, przed i po dekompozycji, oraz liczbę wykonywanych operacji mają adaptacje siatki?
- jak zmienia się po adaptacjach średnia liczba wyrazów niezerowych w pojedynczym wierszu?, jaki ma to wpływ na procent niezerowych wyrazów w pojedynczym wierszu i w całej macierzy?
  - porównaj średnią liczbę niezerowych wyrazów w wierszu przed i po dekompozycji LU dla rozważanych siatek – z jakim zjawiskiem podczas rozwiązywania rzadkich układów równań liniowych metodami bezpośrednimi (wariantami eliminacji Gaussa) powiązana jest zmiana procentu liczby wyrazów niezerowych w macierzy przed i po dekompozycji?
- czy rząd zależności wielkości z wykresu 2 od liczby stopni swobody może znacząco ograniczyć rozmiar zadania możliwego do rozwiązania na danej maszynie?
  - proszę spróbować obliczyć dla jakiego rozmiaru zadania (liczby stopni swobody) pojemność pamięci wymaganej do przechowania wyrazów niezerowych macierzy po dekompozycji przekroczy 1000GB (zakładając 8 bajtów na pojedynczy wyraz)
  - proszę spróbować obliczyć dla jakiego rozmiaru zadania (liczby stopni swobody) czas wymagany do rozwiązania układu równań liniowych przekroczy 1 godzinę (zakładając wykonanie 600 000 000 000 operacji zmiennoprzecinkowych (flop) na sekundę (wydajność 600 Gflop/s))
- analizując czas wykonania algorytmu (powiązany z liczbą wykonanych operacji) należy uwzględnić wzrost wydajności obliczeń dla rosnącego rozmiaru zadania (pozycja `gflop/s` – liczba wykonywanych operacji zmiennoprzecinkowych na sekundę / 10^9)

---------- 4.5 ----------

4 Zadanie 4. Badanie błędu aproksymacji MES przy jednorodnej adaptacji sitaki
- Powtórzenie rozwiązania zadania dla trzech kolejnych siatek otrzymanych przez jednorodną adaptację (podział wszystkich elementów siatki)
- Obserwacja oszacowania błędu dla każdego z przypadków (opcja '**z**' menu – szacowanie błędu metodą Zienkiewicza-Zhu)
- Jak zmienia się oszacowanie normy błędu metodą ZZ dla kolejnych siatek (dla których rozmiar elementu maleje dwukrotnie po każdej adaptacji siatki)?
  - zachowanie oszacowania błędu może silnie zależeć od rozwiązywanego zadania (np. czy istnieje silne skondensowane źródło ciepła, czy występują nieliniowości itp.)
  - dlatego najlepiej badać błąd w przypadku standardowym – bez źródeł ciepła, nieliniowych materiałów i warunków brzegowych itp.

5 Podsumowanie zadań

Zadanie (skrócony opis)	Realizacja w % (0-100)
Zad. 1 Modyfikacja pliku źródłowego i rekompilacja kodu	
Zad. 1 Trzykrotne uruchomienie programu i pobranie informacji o strukturze macierzy	
Zad. 1 Wizualizacja struktury trzech macierzy układu równań	

liniowych	
Zad. 2 Modyfikacja pliku źródłowego (włączenie przenumerowania) i rekompilacja kodu	
Zad. 2 Trzykrotne uruchomienie programu i pobranie informacji o strukturze macierzy	
Zad. 2 Wizualizacja struktury trzech macierzy układu równań liniowych	
Zad. 3 Modyfikacja pliku źródłowego (wyłączenie drukowania macierzy) i rekompilacja kodu	
Zad. 3 Uruchomienie programu z trzykrotną adaptacją i pobranie informacji o strukturze macierzy	
Zad. 3 Analiza danych solwera bezpośredniego dla czterech rozwiązywanych układów równań liniowych	
Zad. 4 Obserwacja zmian oszacowania błędu metodą ZZ wraz z adaptacją siatki	

Sprawozdanie powinno zawierać opis realizacji wszystkich zadań zawartych w temacie, wraz z omówieniem podstaw teoretycznych, odpowiedziami na pytania, wydrukami kodu i plików konfiguracyjnych oraz zamieszczonymi zrzutami ekranu. Opis realizacji każdego zadania może kończyć się wnioskami wynikającymi z przebiegu realizacji, całe sprawozdanie powinno kończyć się wnioskami dotyczącymi całości tematu.