Temat pracy:

Optymalizacja drzew eliminacji dla solwerów wielofrontalnych

1 WYMAGANIA

- Python 3.4 https://www.python.org/
- NumPy 1.9.1 http://www.numpy.org/
- Matplotlib 1.4.3 http://matplotlib.org/
- bintrees 2.0.2 https://pypi.python.org/pypi/bintrees/2.0.2

2 POBIERANIE

2.1 ANACONDA

Najprostszym sposobem instalacji Python'a z niezbędnymi bibliotekami jest pobranie dystrybucji Anaconda ze strony: http://continuum.io/downloads#py34. Należy pamiętać, aby pobrać wersję z Python'em 3.4. Pakiet nie zawiera biblioteki bintrees, **którą należy pobrać po instalacji**.

2.2 RĘCZNA INSTALACJA

2.2.1 Python 3.4

- *Windows*: https://www.python.org/
- *Linux Debian/Ubuntu*: `sudo apt-get install python3` (w nowszych wersjach jest zainstalowany domyślnie)

2.2.2 NumPy/SciPy/Matplotlib

- *Linux*: `sudo yum install numpy scipy python-matplotlib`
- *Windows*: Ze strony http://www.lfd.uci.edu/~gohlke/pythonlib nalezy pobrać pliki .whl powyższych pakietów i w folderze je zawierającym uruchomić polecenie `pip install pakiet.whl` dla każdego z pakietów.

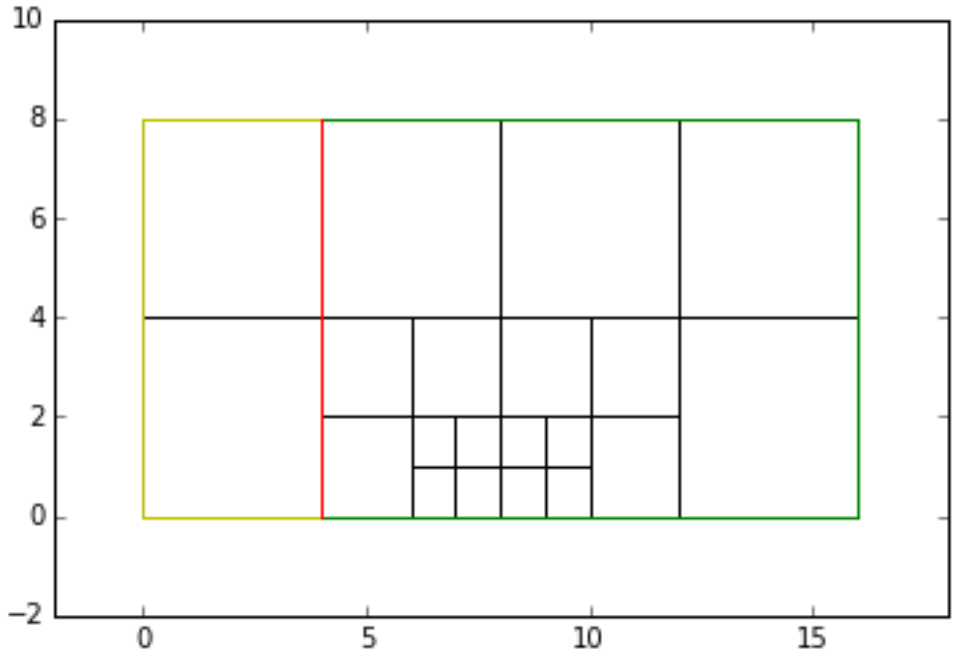
2.2.3 bintrees

- *Linux*: `pip install bintrees`
- *Windows*: Ze strony: https://pypi.python.org/simple/bintrees/ pobrać i zainstalować plik: *bintrees-2.0.2.win32-py3.4.exe*

3 URUCHAMIANIE

3.1 SPYDER IDE

Po instalacji Anaconda'y mamy dostęp do IDE Spyder (w razie ręcznej instalacji można go pobrać z https://github.com/spyder-ide/spyder). Należy otworzyć w nim plik *test.py* z głównego folderu projektu. Pogram uruchamia się przyciskiem F5. W prawym dolnym rogu aplikacji powinien się pojawić się rysunek podstawowej siatki:



3.2 PLIK WEJŚCIOWY

W *test.py* do zmiennej *fileName* przypisujemy ścieżkę pliku wejściowego, do dyspozycji mamy *"mesh_tests/test1.txt"* czyli prostą siatkę (20 elementów) wyświetloną na początku oraz *"mesh_tests/duzy_test.txt"* bardzo dużą siatkę zawierającą >20 K elementów.

Format pliku wejściowego (taki jak w artykule "Quasi-Optimal Elimination Trees for 2D Grids with singularities", tylko z pominięciem części opisującej drzewo rozkładu):

2 <- polynomial order of approximation
2 <- number of elements
1 1 0 0 1 1 <- first element id (1,1) level 1, element 1, and its coordinates (0,0), (1,1)
1 2 1 0 2 1 <- first element id (1,2) level 1, element 2, and its coordinates (1,0), (2,1)

3.3 GŁĘBOKOŚĆ SIATKI

Zmienna *depth_level* określa maksymalny poziom do którego będzie ona wyświetlona (powyżej 10 poziomu nie ma to sensu, gdyż nie zobaczymy widocznych różnic, a program będzie pracował znacznie dłużej). Wartość 0 oznacza wszystkie poziomy.

3.4 PRZYKŁADOWE CIĘCIE

Ustawienie zmiennej logicznej *test_slice* na „True" umożliwia podzielenie struktury danych przechowującej siatkę na dwie oddzielne. Na rysunku krawędzie tnące zaznaczone są kolorem czerwonym. Dwie nowe struktury przechowujące siatkę zaznaczone są kolorem żółtym oraz zielonym, każda z nich zawiera krawędź czerwoną (tnącą).

4 STRUKTURA DANYCH

Struktura danych przechowująca siatkę składa się z trzech rodzajów elementów: **wierzchołki (Vertex)**, **boki (Edge)** i **wnętrza (Face)**. Każdy element przechowuje informacje o tym, jakie elementy znajdują się w jego sąsiedztwie np. wierzchołek przechowuje listę przyległych do niego krawędzi oraz listę sąsiadujących wnętrz, wnętrze ma listę przyległych krawędzi i wierzchołków. Takie podejście umożliwia szybkie i wygodne przemieszczanie się po siatce. Wszystkie listy zaimplementowane są na niskim poziomie poprzez drzewa czerwono czarne.