Sprawozdanie

Metody Obliczeniowe w Nauce i Technice
Laboratorium 2
Układy równań liniowych

W ramach projektu zrealizowany został w sposób całkowity wariant na ocenę 3.5 oraz częściowo wariant na ocenę 4.0.

Konfiguracja

Projekt został zeralizowany w języku Julia, korzystając z środowiska Jupyter. W ramach zadania został przesłany plik Jupytera. Aby otworzyć plik lokalnie należy najpierw pobrać oraz zainstalować Julię lokalnie na komputerze (https://julialang.org/downloads/),a następnie wykonać poniższe komendy:

```
>using Ijulia
>notebook()
```

Może wystąpić potrzeba doinstalowania odpowiednich pakietów:

```
>using Pkg
>Pkg.add("PackageName")
>using PackageName
```

Istnieje możliwość odpalanie Jupytera korzystającego z Julii w sposób nielokalny, przykładowo korzystając z JuliaBox (https://juliabox.com/). W sprawozdaniu umieszczam także link do konta github, gdzie można w łatwy sposób obejrzeć zawartość pliku Jupytera (https://github.com/kjano10/mownit/tree/master/proj1). Github jednak czasami nie radzi sobie wystarczająco dobrze z wykresami w Julii i niektóre z skonstruowanych wykresów wyświetla, a niekótre nie. Jednak w przypadku uruchomieniu Jupytera samemu nie ma ww problemów. Powyższy opis uruchomiania jest spowodowany tym, że język z którego skorzystałem jest językiem bardzo młodym i mało popularnym w porównaniu do innych odpowiedników.

Parsowanie danych

Dane wejściowe parsowane są z pliku tekstowego. W pierwszej linii znajduje się liczba wierzchołków. Z kolejnych linii czytane są szczegółowe informacje o wierzchołkach w konwencji:

```
>NazwaWierzchołka Input
```

Input, gdzie Input oznacza ilość samochodów wjeżdżających/wyjeżdżających z miasta. Dodatnia liczba definiuje samochody wjeżdżające, a ujemna wyjeżdżające. Parsowane jest w ten sposób dokładnie tyle linii, ile wynosi liczba wierzchołków. Następnie z kolejnych linii w pliku tekstowym czytane są informacje o krawędziach w konwencji:

```
>PierwszyWierzchołek DrugiWierzchołek Max
```

PierwszyWierzchołek oznacza nazwę wierzchołka z którego krawędź wychodzi. Analogicznie DrugiWierzchołek oznacza nazwę wierzchołka do którego krawędź zmierza. Max oznacza maksymalną liczbę samochodów które mogą poruszać się tą krawędzią (przepustowość). Przykładowe poprawne dane wejściowe umieściłem w 4 plikach tekstowych dołączonych w pliku zip.

Podczas wczytywanie danych początkowych sprawdzana jest ich poprawność. Wszystkie samochody wjeżdżające oraz wyjeżdżające powinny sumować się do 0.

Obliczanie przepływu

Pierwszym z kroków jest zbudowanie tablicy z danymi na której bazuję cały czas w moim programie. Dla każdego wierzchołka spisuję równania krawędzi wchodzących oraz wychodzących z niego. Następnie korzystam z algorytmu Gaussa-Jordana do wyznaczenia macierzy trójkątnej. Korzystając z faktu, iż krawędzi będzie zazwyczaj więcej niż wierzchołków, a dodatkowo zazwyczaj jedno z równań nie jest liniowo niezależne, realizuję następujący pomysł rozwiązania problemu. Biorę E-(N-1) krawędzi, które traktuję jako parametry tj wszystkie inne krawędzi obliczam z stworzonej funkcji która jako parametry przyjmuje wartości E-(N-1) wierzchołków. (gdzie E oznacza liczbę krawędzi, a N liczbę wierzchołków). A więc uzależniam jak najwięcaj danych od jak najmniejszej jej ilości.

Przykład:

Dzięki algorytmowi Gaussa-Jordana sprowadzam macierz do następującej postaci:

-1 -1 0 1 0 80
0 -1 -1 1 0 30
0 0 0 1 1 -100
0 0 0 0 0 0

Mam więc 4 wierzchołki oraz 5 krawędzi. Jedno z równań okazało się liniowo zależne, a więc finalnie buduję czy funkcje które obliczą mi wartość trzech krawędzi w zależności od wartości dwóch pozostałych. Konkretnie w tym przypdku uzależniłem pokolei x4(x5), następnie x2(x3,x4), a finalnie x1(x2,x4). Z powyższych danych można wyliczyć przykładowy przepływ samochodów w wirtualnym mieście.

Przepustowość

Kolejnym krokiem jest przepustowość. Zakładam, że każda krawędź ma przepustowść większą bądź równą 0, a mniejszą bądź równą tej, którą wczytujemy jako dane wejściowe. Mój pomysł opierał się na stopniowym uzależnieniom zakresu możliwych wartości parametrów funkcji idąc od funkcji najmniej złożynych do tych najbardziej. Ograniczam najpierw moje parametry, potem tylko te funkcje, które korzystają jedynie z parametrów, następnie te, które korzystają już z przeanalizowanych funkcji i tak dalej. Ograniczanie przedziałów było podobnie przeprowadzane, jak budowanie funkcji na poprzednim przykładzie.

Pomysł ten, choć początkowo zachęcający nie okazał się do końca poprawny. W każdym kolejnym kroku zmieniałem zakresy poszczególncy wartości modyfikując zakresy wolnych parametrów. Jednak możemy mieć przypadki w których dane równanie będzie korzystało jedynie z innych funkcji, już nigdzie explicite nie wykorzystując parametru. Taki problem pokazałem na wykresie grafu z pliku "test4.txt", gdzie czasami w zależności od randomizowanej wartości pojawaiają się wartości ujemne.

Wykresy

Do wygenerowania wykresów posłużyłem się bibliotek LightGraphs (do budowania grafu) oraz GraphPlot (rysowanie). Aby na wykresach pojawiły się konkretne wartości generowałem randomowe wartości z wcześniej wyliczonych przedziałów. W nawiasie umieściłem maksymalną dopuszczalną przepustowość dla danej krawędzi.

Podsumowanie

Podsumowując projekt pozwolił pokazać nam, że układy równań liniowych mogą doskonale sprawdzać się w analizie nie tylko tak prostych przypadków, ale także dużych zbiorów danych. Są doskonałym narzędziem symulacji sytuacji z życia codziennego takich jak symulacja ruchu drogowego w mieście.