Metody Obliczeniowe w Nauce i Technice
Laboratorium 1
Arytmetyka komputerowa

Patryk Wojtyczek

Zadanie 1

Napisz i sprawdź funkcje rozwiazujaca układ równań liniowych n n metoda Gaussa-Jordana. Dla rozmiarów macierzy współczynników wiekszych niż 500 500 porównaj czasy działania zaimplementowanej funkcji z czasami uzyskanymi dla wybranych funkcji bibliotecznych.

Metoda Gaussa-Jordana służy do znajdowania rozwiazań dla liniowych układów równań. Działa podobnie do metody Eliminacji Gaussa, która za pomoca operacji elementarnych sprowadza macierz A do postaci schodkowej (macierzy trójkatnej górnej). Uzyskanie rozwiazania do naszego równania:

$$Ax = b$$

Uzyskujemy poprzez podstawienie wsteczne zaczynajac od ostatniego wiersza. Metoda Gaussa-Jordana eliminuje konieczność podstawiania wstecznego od razu sprowadzajac macierz A do postaci macierzy trójkatnej górnej i dolnej (diagonalnej), z której możemy odczytać rozwiazanie bezpośrednio.

Implementacja w czystym Pythonie pozostawia wiele do życzenia jeśli chodzi o wydajność. Implementacje biblioteczne sa do kilkuset razy szybsze.

	Rozmiar	Numpy [s]	Implementacja własna [s]
x	100	0.000195	0.067569
x	200	0.000684	0.261762
x	300	0.002007	0.743834
x	400	0.002409	1.215598
x	500	0.002121	1.982436
x	600	0.005212	3.238851
x	700	0.061224	4.692816
x	800	0.010994	6.840780
x	900	0.016667	8.256598

Przy samej implementacji musimy pamietać, że ‘pivot‘ nie może być zerem wiec musimy wykonywać zamiany wierszy, gdy element na przekatnej jest zerem. W celu zminimializowania błedu propagowanego przez reprezentacje zmiennoprzecinkowa możemy skorzystać z metody ‘partial pivoting‘, w której przy każdej redukcji bedziemy szukali najwiekszego elementu w danej kolumnie. Aby uzyskać jeszcze lepsza skuteczność możemy przeskalować każdy wiersz tak aby najwiekszy element w wierszu miał wartość jeden. Wtedy za pivota bedziemy wybierać element który ma najwieksza wartość wzgledem elementów w wierszu co znacznie poprawi dokładność naszego algorytmu.

Zadanie 2

Napisz i sprawdź funkcje dokonujaca faktoryzacji $A = LU$ macierzy A. Zastosuj cześciowe poszukiwanie elementu wiodacego oraz skalowanie.

Celem faktoryzacji LU jest uzyskanie macierzy trójkatnej dolnej (L) i macierzy trójkatnej górnej (U) takich, że $A = LU$. Taka dekompozycja sprowadza rozwiazanie układu równań

$$Ax = b,$$

do

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

Który możemy rozwiazać w czasie $O(n^2)$. Sama dekompozycja zajmuje $O(n^3)$ wiec jeżeli chcemy znaleźć rozwiazanie pojedyńczego układu równań zysk jest znikomy, natomiast gdy wielokrotnie rozwiazujemy układ równań z ta sama macierza współczynników możemy wykonać faktoryzacje LU tylko raz i nastepnie rozwiazywać układ w czasie kwadratowym. Gdy liczba układów do rozwiazania jest proporcjonalana do liczby równań takie postepowanie zajmie $O(n^3) + O(n) \cdot O(n^2) = O(n^3)$ podczas gdy używajac eliminacji gaussa bedziemy mieli $O(n \cdot n^3) = O(n^4)$. Techniki opisane w poprzednim zadaniu ('partial pivoting', 'scaling') również sie tutaj aplikuja. Jednak tutaj używajac techniki 'partial pivoting' przy rozwiazywaniu równań trzeba wziaść pod uwage które wiersze zostały zamienione - mianowicie:

$$PA = LU, \text{ gdzie P to macierz zamian wierszy (permutation matrix)}$$

Ostatecznie nasze rozwiazanie bedzie postaci

$$\begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

Przy używaniu full pivotingu - nie ograniczamy sie do zamian wierszy ale zamieniamy też kolumny.

$$PAQ = LU, \text{ gdzie P to macierz zamian wierszy (permutation matrix)}$$

Wtedy nasze rozwiazanie bedzie postaci

$$\begin{cases} Lz = Pb \\ Uy = z \\ x = Qy \end{cases}$$

	N	Scipy time [s]	My time [s]
x	100.0	0.000244	0.345174
x	200.0	0.012697	2.615853
x	300.0	0.001271	8.152312
x	400.0	0.001881	17.481591

Znowu biblioteczne implementacje sa szybsze o kilka rzedów wielkości.

Zadanie 3

Analiza obwodu elektrycznego

Rozwiazanie zaimplementowałem używajac I i II prawa Kirchoffa generujac odpowiednia ilość równań. Mianowicie dobierałem V - 1 równań z pierwszego I Kirchoffa, nastepnie brałem pewne specjalne pokrycie cyklowe grafu - Cycle basis (każdy cykl w grafie może zostać utworzony jako suma cykli z pokrycia) i korzystajac z II prawa zapisywałem równania dla oczek. Aby wziać pod uwage SEM szukałem prostych ścieżek od s do t. Postepujac w taki sposób zawsze (jeśli nie ma błedów z preprocessingiem losowego grafu) kończe z E + 1 równaniami. Otrzymany w ten sposób układ równań jest nadokreślony wiec korzystam z przybliżonych metod rozwiazania problemu - Ordinary Least Squares:

$$I = (A^T \cdot A)^{-1} \cdot A^T b$$

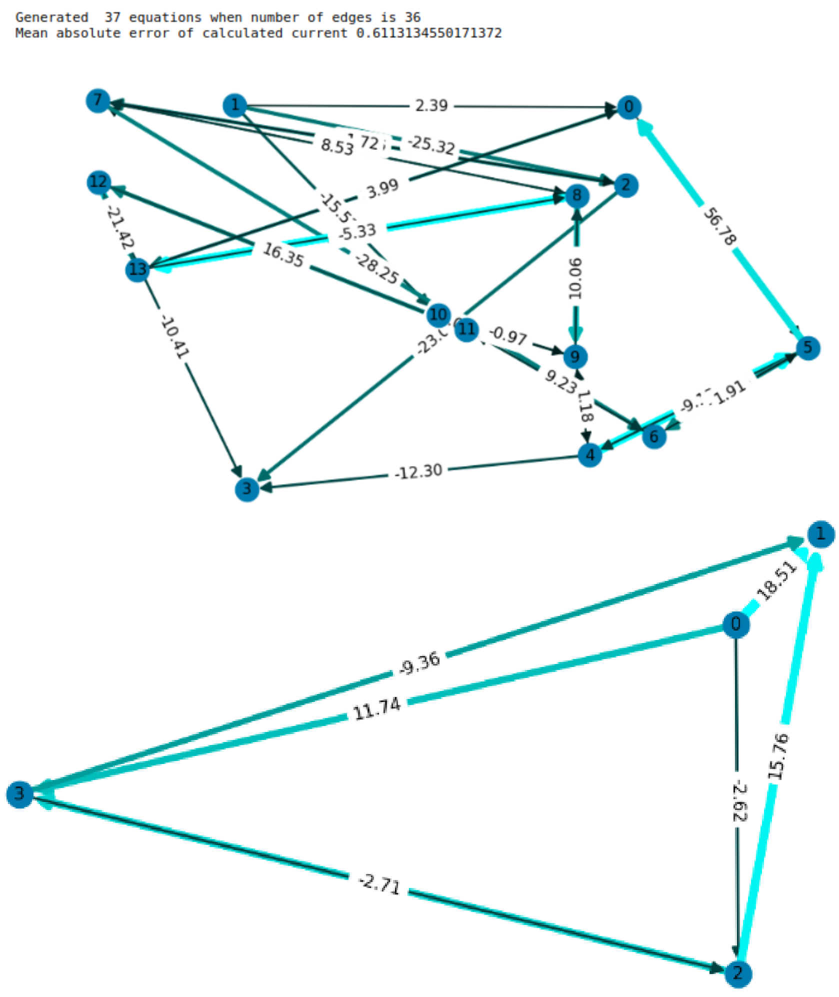

Figure 1: Przykładowe rozwiazania

Dla niewielkich grafów rozwiazanie zwraca niewielki blad natomiast dla dużych grafów bład jest tak duży, że rozwiazanie wydaje sie być dalekie od poprawnego. Błedy moga brać sie ze sposobu wyboru równań, akumulacji błedu przez wykonywanie dużej ilości operacji na liczbach zmiennoprzecinkowych lub być może błedu implementacji.