Celem laboratorium nr 6 jest pokazanie jak uwarunkowanie układu równań (w naszym przypadku układu równań, w którym występuje macierz Hilberta) wpływa na numeryczną poprawność metody eliminacji Gaussa. Autor instrukcji w zadaniu nr 1 podaje nam do obliczenia układ równań: Hx=b, gdzie H oznacza właśnie macierz Hilberta (otrzymujemy wzór na każdy element tej macierzy), x to wektor z niewiadomymi, które mamy na celu obliczyć, oraz b to wektor z prawymi stronami równań, które ponownie są obliczane z podanego wzoru. Przed wykonaniem laboratorium załączam odpowiednie pliki.

Pierwsze polecenie wykonane jest w części głównej, gdzie za pomocą funkcji ‘malloc’ dynamicznie alokuję dwuwymiarową tablicę będącą macierzą ‘H’, a następnie alokuję wektory szukanych ‘x’ oraz prawych stron ‘b’.

Kolejnym etapem w rozwiązaniu zadania jest stworzenie odpowiednich funkcji, które obliczą oraz pokażą macierz Hilberta, a następnie obliczą i wyświetlą prawe strony równań.

Funkcję ‘HilbertMatrix’ realizuję za pomocą podanego nagłówka oraz dwóch pętli znajdujących się w środku funkcji. Za każdym przejściem pętli każdy poszczególny wyraz jest liczony za pomocą podanego wzoru. W funkcji ‘displayMatrix’ zamiast wzoru zapisuję komendę ‘printf’, która wyświetla na ekranie każdy element macierzy.

Tworzę funkcję ‘computeVtec’, która oblicza wektor z prawymi stronami równań. W każdym kroku wartość ‘b[i]’ jest zerowana tak, żeby przy dodawaniu uzyskać jedynie sumę elementów macierzy ‘H’. W środku wewnętrznej pętli zapisuję odpowiedni wzór na sumę i zamykam funkcję.

‘Plotvtec’ jest złożony jedynie z jednej pętli ze względu na to, iż ‘b’ jest tablicą jednowymiarową. Ponownie zapisuję komendę ‘printf’ oraz zamykam funkcję.

W ‘main-nie’ oprócz zapisania odpowiednich alokacji umieściłem jedynie wywołanie funkcji, aby przygotować wektory i macierz do użycia w funkcji ‘gauss’, która została zaimplementowana wraz z odpowiednimi plikami. Po wykonaniu funkcji ‘gauss’ wyświetlam rozwiązania na ekran. Zamykam cały program.

Po przetestowaniu działania programu dla różnych wartości ‘n’ doszedłem do wniosku, iż informacja w poleceniu się potwierdza, mianowicie dla n mniejszego lub równego 8 rozwiązania są dokładne, a powyżej 8 zaczynają się delikatne błędy w wartościach ‘x-ów’.

Po zmianie typów danych na ‘float’ wyniki stają się bardzo niedokładne (różnią się mniej więcej o jeden rząd wielkości względem wyników, które otrzymałem po użyciu typu ‘double’).