Zad. 3: Układ równań liniowych

1 Cel ćwiczenia

Wykształcenie umiejętności modelowania kluczowych dla danego problemu pojęć. Definiowanie właściwego interfejsu klasy. Zwrócenie uwagi na dobór odpowiednich struktur danych w zależności od metody rozwiązywania problemu. Praktyczne zapoznanie się z problemem skończonej binarnej reprezentacji liczb oraz wynikającego stąd problemu niedokładności obliczeń.

2 Program zajęć

- *Demonstracja przykładu problemu skończonej reprezentacji binarnej liczb*
- *Ocena realizacji zadania z poprzedniego laboratorium* – ocenie podlega poprawność realizacji zadania, styl pisania programu oraz opisy.
- *Ocena przygotowania do zajęć* – ocenie podlega schemat blokowy działania programu (szczegóły wymagań patrz podrozdział 4)
- *Modyfikacja programu wg wskazań osoby prowadzącej* – ocenie będzie podlegała poprawność realizacji modyfikacji. Pracę nad modyfikacją programu (wszystkie operacje należy wykonywać na kopii) należy rozpocząć już w trakcie pierwszej fazy laboratorium, gdyż prowadzący nie będzie w stanie ocenić wcześniejszego programu wszystkim jednocześnie.

3 Opis zadania programowego

Należy napisać program, który umożliwia rozwiązanie układu równań liniowych z trzema niewiadomymi postaci:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

Przedstawiony powyżej układ równań wygodnie jest reprezentować w postaci macierzowej jako

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Tak więc można zapisać sformułowany problem w bardziej zwartej i ogólnej postaci:

$$\mathbf{Ax} = \mathbf{b}$$

gdzie $\mathbf{A}$ – to macierz współczynników równania, $\mathbf{x}$ – wektor nieznanych wartości, które należy wyliczyć, $\mathbf{b}$ – wektor wyrazów wolnych. Ponadto program dla znalezionego rozwiązania powinien wyliczyć błąd wynikający z niedokładności obliczeń. Reprezentantem błędu będzie wektor różnic przedstawiony poniżej

$$\vec{\varepsilon} = \mathbf{Ax} - \mathbf{b}.$$

Miarą błędu jest długość wektora $\vec{\varepsilon}$, a więc

$$\varepsilon = \sqrt{\vec{\varepsilon} \cdot \vec{\varepsilon}},$$

gdzie $\vec{\varepsilon} \cdot \vec{\varepsilon}$ jest iloczynem skalarnym wektorów ε. Należy zbadać, kiedy wspomniany błąd staje się znaczący. Aby móc to określić należy odwołać się do interpretacji geometrycznej wspomnianego układu równań. Nie przypadkiem został wybrany układ z trzema niewiadomymi. Dla układu z czterema (i więcej) niewiadomymi byłoby to już niemożliwe (dlaczego?).

UWAGA:

- Definiowane w programie klasy muszą odzwierciedlać kluczowe pojęcia znajdujące się w opisie problemu. Do pojęć takich należy między innymi *macierz współczynników*, *wektor wyrazów wolnych* oraz *układ równań liniowych*.

- Pomimo tego, że program piszemy dla układu z trzema niewiadomymi, w programie jako rozmiaru **nie można** używać liczby 3. Należy zdefiniować poprzez dyrektywę preprocesora odpowiedni symbol, który będzie dalej wykorzystywany, np.

  ```
  #define ROZMIAR    3
  ```

  Symbol ten powinien zostać wykorzystany później zarówno przy definiowaniu klasy `Wektor`, jak też `Macierz`. Musi on być wykorzystany we wszystkich miejscach, w których odwołujemy się do rozmiaru wektora lub macierzy (np. w pętlach typu `for`). Wspomniany wcześniej symbol powinien być zdefiniowany w osobnym pliku nagłówkowym.

3.1 Działanie programu

Program nie ma udostępniać żadnego interfejsu użytkownika. Zakłada się, że kolumny macierzy współczynników równania oraz wektor wyrazów wolnych $[\mathbf{A}\ \mathbf{b}]$ czytane są w postaci transponowanej, tzn $[\mathbf{A}\ \mathbf{b}]^T$. Dla lepszego zrozumienia rozważmy układ równań:

$$\begin{cases} 2x_1 + 2x_2 + x_3 = 9 \\ x_1 + 2x_2 + 1{,}5x_3 = 8{,}5 \\ x_1 + 3x_2 + x_3 = 8 \end{cases}$$

Połączony zapis macierzy współczynników i wektora wyrazów wolnych ma postać:

$$[\mathbf{A}\ \mathbf{b}] = \begin{bmatrix} 2 & 2 & 1 & 9 \\ 1 & 2 & 1{,}5 & 8{,}5 \\ 1 & 3 & 1 & 8 \end{bmatrix}$$

Postać transponowana to:

$$[\mathbf{A}\ \mathbf{b}]^T = \begin{bmatrix} 2 & 1 & 1 \\ 2 & 2 & 3 \\ 1 & 1{,}5 & 1 \\ 9 & 8{,}5 & 8 \end{bmatrix}$$

Dla ułatwienia sobie pracy warto zapisać dane w pliku i później czytać dane z pliku poprzez przekierowanie go na wejście standardowe programu. Taki sposób postępowania przedstawiony jest poniżej.

```
jkowalsk@noxon: rozwiazanie> cat rownanie_liniowe.dat
       2  1    1
       2  2    3
       1  1.5  1
       9  8.5  8

jkowalsk@noxon: rozwiazanie> ./uklad_rownan < rownanie_liniowe.dat
Macierz A^T:

       2  1    1
       2  2    3
       1  1.5  1

Wektor wyrazow wolnych b:

       9  8.5  8

Rozwiazanie x = (x1, x2, x3):

     2  1  3

          Wektor bledu:    Ax-b  = ( 4.768e-07, 2.33e-07, 0 )
 Dlugosc wektora bledu: ||Ax-b|| = 5.3068563e-07

jkowalsk@noxon: rozwiazanie>_
```

Uwaga: Dane o błędzie mają jedynie charakter poglądowy. Nie są to faktycznie wyliczone wartości.

Ułożenie znaków w linii w przypadku wyświetlania macierzy, nie musi być idealnie równe, aby jeden wiersz długością pasował do drugiego.

3.2 Materiały pomocnicze

Zalążek programu wraz z plikiem zawierającym przykładowe dane znajduje się w katalogu `~bk/edu/kpo/zad/z3`.

3.3 Wymagania co do konstrukcji programu

Oprócz wymagań sformułowanych w opisie zadania należy uwzględnić uwarunkowania przedstawione poniżej.

- Należy zdefiniować klasy `Wektor`, `Macierz` oraz `UkladRownanLiniowych`. Muszą one mieć **tylko i wyłącznie niezbędne pola reprezentujące atrybuty danego pojęcia**.
- Należy odpowiednio przeciążyć operator indeksujący dla klasy `Wektor`, operator funkcyjny dla klasy `Macierz` i odpowiednio posługiwać się nimi w programie. Możliwe są również również inne kombinacje tych operatorów.

- Należy przeciążyć operator mnożenia, tak aby była możliwość przemnożenia macierzy przez wektor. Ponadto konieczne jest przeciążenie operatora dodawania i odejmowania wektorów oraz mnożenia i dzielenia przez liczbę.
- Program musi zachować strukturę modułową i odpowiednią strukturę kartotek. O ile będzie to konieczne, należy zmodyfikować plik `Makefile` (np. gdy dodany zostanie nowy moduł).
- Każda z klas powinna zostać zdefiniowana w oddzielnym pliku nagłówkowym. Metody tej klasy powinny być natomiast definiowane w osobnym module związanym z daną klasą, np. definicja klasy `UkladRownanLiniowych` powinna znaleźć się w pliku nagłówkowym `UkladRownanLiniowych.hh`, zaś metody w pliku `UkladRownanLiniowych.cpp`. Proste metody można definiować bezpośrednio w ciele klasy.
- Wszystkie metody, które nie zmieniają stanu obiektu, na którym działają, powinny być metodami typu **const**.

Oprócz tego pozostają w mocy wszystkie wcześniejsze wymagania dotyczące struktury katalogów, pliku Makefile, modułowej struktury programu, jak też opisów.

3.4 Wersja uproszczona – oceniana nie więcej niż na 4,0

W wersji uproszczonej można ograniczyć się do układu z dwoma zmiennymi.

3.5 Rozszerzenia nieobowiązkowe

Proponuje się, aby równanie było wyświetlane przez program w postaci:

```
|  2  1  1 ||x_1|   |  1 |
|  1  2  1 ||x_2| = |  2 |
|  1  1  1 ||x_3|   | -1 |
```

Zamiast oddzielnie macierz `A` i wektor wyrazów wolnych `b`. **Niniejszy sposób wyświetlenia musi być zrealizowany poprzez przeciążenie operatora $<<$ działającego na strumieniu wyjściowym dla klasy modelującej pojęcie układu równań liniowych.**

4 Przygotowanie do zajęć

4.1 Tydzień 0

Przed zajęciami należy napisać diagram czynności reprezentujący ideę działania programu bez wchodzenia nadmiernie w detale, np. operacja wyświetlenia układu równań i jego rozwiązania, operacja czytania parametrów równania. Tym operacjom powinny odpowiadać pojedyncze czynności reprezentowane przez pojedyncze bloczki. Zaleca się wybranie metody rozwiązywania układu równań z wykorzystaniem wzorów Cramera. W takim przypadku operację liczenia wyznacznika należy również przedstawić w postaci pojedynczej czynności, jak też wstawianie odpowiedniej kolumny do macierzy.

4.2 Tydzień 1

Przed zajęciami muszą zostać przygotowane następujące elementy zadania. Wszystko co będzie ponad to będzie oceniane *in plus*.

- Należy przygotować możliwie szczegółowy schemat blokowy liczenia wyznacznika metodą eliminacji Gaussa (diagram musi mieć postać elektroniczną).
- Powinna być zdefiniowana klasa `Wektor` wraz z przeciążeniem operatorów zapisu do strumienia wyjściowego (`cout`, operator `<<`). czytania ze strumienia wejściowego (`cin`, operator `>>`). Należy również przeciążyć operację dodawania i odejmowania wektorów, mnożenie wektora przez wektor (iloczyn skalarny) oraz wektora przez liczbę, jak też dzielenia wektora przez liczbę. Klasa i metody muszą być opisane. Definicje klasy i metod powinny być zapisane w osobnym module.
- Należy również zdefiniować klasę `Macierz`. Należy przeciążyć operację zapisu macierzy do strumienia wyjściowego (`cout`, operator `<<`). Klasa i metody muszą być opisane. Definicja klasy powinna być w osobnym module.
- Oba moduły powinny zostać uzupełnione modułem `main` i kompilować się oraz konsolidować jako program.

4.3 Tydzień 2

Rozliczenie się z gotowego programu i rozpoczęcie następnego zadania (tydzień 0 dla zadania nr 3).