Praca domowa 8 – aprox

Michał Szczygieł

Wstęp

Zadanie składa się z 2 plików:

- Main.java
- Aprox.java

CEL:

Wartości pewnej funkcji y = f(t), gdzie zmienna niezależna t reprezentuje czas (z dokładnością do 1 minuty) a y przyjmuje wartości rzeczywiste, pamiętane są w repozytorium danych (w bazie danych) w tabeli o nazwie Ttable .

Drugi z parametrów wywołania programu <date> wskazuje dzień (datę) początku interesującego nas przedziału czasu $T(t_{0,}t_k)$, przy czym t0 oznacza godzinę 08:00 w dniu określonym przez <date> a t_k oznacza godzinę 07:59 w dniu następnym (czyli przedział zawsze obejmuje 24 godziny). Należy napisać program dokonujący aproksymacji średniokwadratowej funkcji f w przedziale T wielomianem trzeciego stopnia

$g(x)=a_{3x}^3+a_{2x}^2+a_{1x}+a_0$ gdzie x określone jest w minutach oraz $x=t-t_0$.

Klasa Main

Zadaniem tej klasy jest wczytanie danych z bazy danych oraz przekazanie otrzymanych wartości do klasy Aprox. Sposób tworzenia zapytania dla bazy danych w celu uzyskania danych odbywa się z użyciem:

```
/**
 * Query to be executed on database
 */
private static final String QUERY = "SELECT y, t FROM Ttable WHERE t
BETWEEN ? AND ?";

PreparedStatement statement = conn.prepareStatement(QUERY);
```

Oprócz tego jak kolejnym parametrem jest data, która może być napisana w dowolnym stylu operająca się o Locale używane przez wirtualna maszynę Javy (przeważnie są to takie same lokale jak systemowe). Datę pobieram wykorzystując poniższą metodę:

```java
/**
 * This method changes executable param of date in string format to object
 * of {@link java.sql.Timestamp}.
 *
 * @param date
 *            The string with date value;
 * @param milliPerDay
 * @return The Object of {@link java.sql.Timestamp}.
 */
private static Timestamp getDate(Long milliPerDay, final String... date) {
    String complexDate = "";
    for (int i = 1; i < date.length; i++) {
        complexDate += date[i] + " ";
    }

    for (DateFormat format : formats) {
        try {
            return new Timestamp(format.parse(complexDate).getTime()
                    + 28800000L + milliPerDay);
        } catch (ParseException e) {
            continue;
        }
    }

    return null;
}
```

Metody:

`public static void main(final String... args)`

- Metoda główna, przyjmuje dwa parametry connection string i datę, przekazuje wczytane dane i rozmiar do klasy Aprox, wykonując cel, oraz wyświetlenie wyniku aproksymacji, z dokładnością do 5 miejsc po przecinku.

`public static Map<Timestamp, Float> dbconnection(final String connector, Timestamp t0, Timestamp tk)` – Metoda ta ma na celu obsłużenie połączenia bazodanowego, oraz wykonanie zapytania SQL dla bazy danych. Oraz zwrócenie poprawnie sformatowanych danych poprzez funkcję umieszczoną poniżej.

`private static Map<Timestamp, Float> retriveData(final ResultSet result) throws SQLException` – Metoda ta odbiera dane po wykonaniu zapytania SQL do bazy danych, oraz upakowywuje dane do mapy gdzie klucz stawowi czas reprezentowany przez timestamp, a wartość jest wynikiem funkcji y=f(t) .

`private static Timestamp getDate(Long milliPerDay, final String... date)`

- Metoda ta zwraca czas w postaci Timestampa.

Klasa Aprox

Klasa ta, otrzymawszy dane od klasy wywołującej, tworzy na początku, macierz symetryczną dla danej funkcji średnio-kwadratowej aproksymującej wielomian 3 stopnia.

$$\begin{matrix} <1,1> & <1,x> & <1,x^2> & \dots & <1,x^n> \\ <x,1> & <x,x> & <x,x^2> & \dots & <x,x^n> \\ <x^2,1> & <x^2,x> & <x^2,x^2> & \dots & <x^2,x^n> \\ & & \dots & & \\ <x^n,1> & <x^n,x> & <x^n,x^2> & \cdots & <x^n,x^n> \end{matrix} \begin{matrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{matrix} = \begin{matrix} <y,1> \\ <y,x> \\ <y,x^2> \\ \vdots \\ <y,x^n> \end{matrix}$$

W kolejnym kroku dla otrzymanej macierzy symetrycznej przeprowadzam dekompozycję metodą Banasiewicza-Choleskiego, dzięki czemu otrzymuję macierz trójkątną dolną.

Kolejnym etapem jest dekompozycja LU, polegająca na na wyznaczeniu naszego rozwiązania (parametrów a). Po wyznaczeniu tych wartości następuje obliczenie funkcji i zwrócenie wyniku.

Realizacja algorytmu Aprox

Dla wyznaczonej już macierzy symetrycznej otrzymujemy następujące równie:

$$A * X = Y$$

gdzie X to wektor poszukiwanych współczynników wielomianu. Następnie macierz A poddajemy dekompozycji metodą Choleskiego. Otrzymujemy macierz L taką że:

$$L * L^T * X = Y$$

Wyliczamy X w dwóch etapach:

Propagacja w przód:

$$L * Z = Y$$

gdzie:

$$Z[i] = \frac{Y[i] - \sum_{j=0}^{i-1} L[i][j] * Z[j]}{L[i][i]}$$

$$i = 0...3$$

Propagacja w tył:

$$L^T * X = Z$$

gdzie:

$$X[i]=\frac{Z[i]-\sum_{j=i+1}^{3} L^T[i][j]*X[j]}{L^T[i][i]}$$

$i = 3...0$

Macierz transponowaną otrzemujemy: $L^T[i][j]=L[j][i]$

Otrzymujemy wektor X t.że:

$$g(x)=X[0]+X[1]*x+X[2]*x^2+X[3]*x^3$$

Złożoność obliczeniowa dla tego programu wynosi:

- Rozwiązanie układu Ax=b – $O(\frac{1}{3}n^3)$
- Dla procesu rozpoznawania propagacją w przód i w tył - $O(2n^2)$

Dyskusja

Można postawić sobie pytanie, czy użycie BigInteger i BigDecimal jest sensownym podejściem. Jednak na zamienienie z typów Double na wymienione powyżej dostaję wyniki całkowicie innego rzędu. W pewnym momencie podczas używania podstawowych typów, napotkałem na diagonali macierzy po dekompozycji Cholskiego wartość „0”. Według definicji samo przeczy przez siebie, ponieważ podając symetryczną macierz dekompozycji Cholskiego, winniśmy otrzymać macierz trójkątną dolną gdzie jej wartości na przekątnej są różne od zera.