Programmentwurf

Problemstellung

In dem Programmentwurf sollen zwei Verfahren zur Berechnung der Lösung eines linearen Gleichungssystems, das Jacobi- und das Gauß-Seidel-Verfahren implementiert werden (siehe de.wikipedia.org/wiki/Jacobi-Verfahren und de.wikipedia.org/wiki/Gauß-Seidel-Verfahren).

Das lineare Gleichungssystem hat allgemein die Form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad\qquad\quad \vdots \quad &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nn}x_n &= b_n \end{aligned}$$

Die beiden Verfahren berechnen aus einem Startvektor $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ eine Folge von Vektoren $x^m = (x_1^m, x_2^m, \ldots, x_n^m)$, die unter bestimmten Voraussetzungen gegen die Lösung des Gleichungssystems konvergiert.

Implementierung

Für die Implementierung sollen zwei C-Strukturen `Matrix` und `Vector` und ein Aufzählungsdatentyp `Method` der Form:

```
typedef struct
{
  int n;
  double **data;
} Matrix;

typedef struct
{
  int n;
  double *data;
} Vector;

typedef enum
{
  JACOBI = 0, GAUSS_SEIDEL = 1
} Method;
```

verwendet werden. Die Struktur `Matrix` speichert die Koeffizienten a_{ij} als eine Matrix mit n Zeilen und n Spalten. Die Struktur `Vector` wird zum Speichern der Koeffizienten b_i und der Iterationsvektoren x^m verwendet.

Zur Berechung der Iterationsfolge sollen folgende C-Funktionen implementiert werden:

(a) Die Funktion `load` hat folgende Deklaration:

```
bool load (const char *filename, Matrix *A, Vector *b, Vector *x);
```

Der erste Parameter ist der Name einer csv-Datei, in der die Koeffizienten a_{ij} und b_i für das LGS, sowie die Werte des Startvektors x^0 gespeichert sind. Die Werte sind zeilenweise in der Datei gespeichert, zwei Werte werden durch ein Komma getrennt:

$$a_{i1}, a_{i2}, ..., a_{in}, b_i[, x_i^0]$$

Leerzeichen vor oder nach einem Wert sind zulässig. Der Startwert x_i^0 ist optional, fehlt der Wert, dann muss x_i^0 mit 0 initialisert werden. Leere Zeilen ohne Werte sollen ignoriert werden.

Die Funktion liest die Datei ein, überprüft ob das Format eingehalten wurde und speichert die Daten in den Strukturen, die durch die drei Pointer `A`, `b` und `x` referenziert werden. Der Rückgabewert der Funktion ist `true`, wenn die Daten fehlerfrei eingelesen und gespeichert wurden, andernfalls wird `false` zurückgegeben.

(b) Die Funktion `solve` hat folgende (unvollständige) Deklaration:

```
solve (Method method, Matrix *A, Vector *b, Vector *x, double e);
```

Der Rückgabetyp muss entsprechend der gewählten Implementierung ergänzt werden. Über den ersten Parameter wird das Iterationsverfahren ausgewählt. Die folgenden drei Parameter sind drei Pointer auf Strukturen mit den Koeffizienten des LGS und dem Startvektor x^0. Der letzte Parameter e ist eine Fehlerschranke.

Die Funktion berechnet die Folge der Vektoren x^m nach dem gewählten Iterationsverfahren und speichert die Vektoren in einem Feld oder als verkettete Liste. Die Iteration wird beendet, wenn der Abstand zweier aufeinanderfolgender Iterationsvektoren kleiner als der Fehlerwert e ist. Falls nach 100 Iterationsschritten die Fehlerschranke nicht erreicht wurde, dann wird die Iteration abgebrochen. Der Rückgabewert ist das Feld oder die Liste mit den Iterationsvektoren x^m, bei einem Fehler wird ein leeres Feld oder eine leere Liste zurückgegeben.

(c) Die `main`-Funktion fragt den Anwender nach dem Namen einer csv-Datei mit den Daten und liest anschließend die Datei mit der Funktion `load` ein. Falls die Daten in der Datei gültig sind, darf der Anwender das Iterationsverfahren auswählen und eine Fehlerschranke eingeben. Das Programm berechnet daraufhin die Iterationsvektoren mit Hilfe der Funktion `solve`. Falls die Berechung erfolgreich war, wird der Anwender gefragt, ob die gesamte Folge oder nur der letzte Vektor angezeigt werden soll und anschließend die gewünschten Werte auf der Konsole ausgegeben. Falls bei der Eingabe oder der Berechnung ein Fehler auftritt, dann soll eine Fehlermeldung erfolgen und der Anwender gefragt werden, ob er eine neue Berechnung starten oder das Programm beenden möchte.

Bedingungen

- Eine Bearbeitung in Gruppen von 3 Studierenden ist zulässig. Alle Studierenden in einer Gruppe erhalten die gleiche Bewertung.
- Das Programm muss unter Linux mit dem GCC-Compiler unter Verwendung der Optionen `-Wall` und `-pedantic-errors` fehlerfrei und ohne Warnungen kompilierbar sein.
- Die vorgegebenen Datentypen und Funktionen müssen wie oben beschrieben zur Lösung der Aufgabe eingesetzt werden. Es dürfen nach Bedarf weitere Datentypen und Funktionen ergänzt werden. Globale Variablen sind nicht erlaubt.
- Abgabe: Source-Code als c-File oder zip-File auf Moodle hochladen. Die Datei mit den Namen der Gruppenmitglieder benennen. Abgabetermin ist 22.03.2020.

Bewertungskriterien

Teilaufgabe (a)	15 Punkte
Teilaufgabe (b)	15 Punkte
Teilaufgabe (c)	10 Punkte
Kompilierbarkeit	5 Punkte
Strukturierung und Kommentierung	5 Punkte
Gesamt	50 Punkte