Programming C++

Project **Linear System of equations**

Status: 25. Mai 2021, 19:36

Supervisor: Prof.Dr. G. Haase, gundolf.haase@uni-graz.at

Gaußsche Elimination[1]:

Dass diese Aufgabe so ausführlich beschrieben ist bedeutet keinesfalls, dass sie entsprechend aufwendig auszuführen ist.

Ein lineares Gleichungssystem von m Gleichungen in n Unbekannten hat die Gestalt

$$\begin{array}{ccccccccc} a_{11}\xi_1 & + & a_{12}\xi_2 & + & \cdots & + & a_{1n}\xi_n & = & \beta_1 \\ a_{21}\xi_1 & + & a_{22}\xi_2 & + & \cdots & + & a_{2n}\xi_n & = & \beta_2 \\ \vdots & + & \vdots & + & & + & \vdots & = & \vdots \\ a_{m1}\xi_1 & + & a_{m2}\xi_2 & + & \cdots & + & a_{mn}\xi_n & = & \beta_m \end{array} \quad . \tag{1}$$

Dabei sind für $i = 1, \ldots, m$ und $j = 1, \ldots, n$ die a_{ij} und β_i gegebene Zahlen aus einem Körper $\mathbb{K}$. Gesucht sind die Zahlen $\xi_1, \ldots, \xi_n \in \mathbb{K}$.
Die zum linearen Gleichungssystem (1) zugehörige Matrix ist gegeben durch

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \quad . \tag{2}$$

Die Systemmatrix des linearen Gleichungssystems (1) ist gegeben durch

$$(A \mid \beta) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & \beta_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & \beta_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & \beta_m \end{pmatrix} \quad . \tag{3}$$

Als MathematikerIn oder TechnikerIn kommen Sie während Ihres Studiums öfter in die Verlegenheit, kleine lineare Gleichungssysteme über $\mathbb{R}$ oder $\mathbb{C}$ (händisch) lösen zu müssen. Was machen sie? Sie werfen wohl Matlab oder Mathematica an, tippen die Systemmatrix ein und hoffen auf ein Ergebnis. Manchmal ergeben sich aber Probleme:

- Das lineare Gleichungssystem ist nicht (eindeutig) lösbar. Dann können Sie sich zumindest die reduced–row–echelon form[2] berechnen und den Lösungsraum, so es einen gibt, händisch ablesen.
- Sie benötigen die einzelnen Zwischenschritte Ihrer Rechnung. Diese Funktionalität stellen Ihnen Matlab oder Mathematica nicht zur Verfügung. Diesbezüglich werden Sie im Internet auf Amateurseiten wohl fündig werden.

[1] http://de.wikipedia.org/wiki/Gau%C3%9F-Jordan-Algorithmus
[2] http://en.wikipedia.org/wiki/Row_echelon_form#Reduced_row_echelon_form

- Sie müssen ein lineares Gleichungssystem nicht über $\mathbb{R}$ oder $\mathbb{C}$ lösen, sondern über einem beliebigen Körper $\mathbb{K}$. Dies könnte problematisch werden.

Vorsicht: Bei diesem Beispiel geht es nur um kleine lineare Gleichungssysteme und die Darstellung der Zwischenschritte der Gaußschen Elimination beim Lösen von (1). Möchten Sie lineare Systeme mit mehreren tausend Unbekannten lösen, müssen Sie andere Methoden entwickeln und anders programmieren!

Eine Möglichkeit, das lineare Gleichungssystem (1) zu lösen, ist das Gaußsche Eliminationsverfahren. Wir gehen von der erweiterten Systemmatrix (3) aus und verwenden einen modifizierten Gauß–Jordan Algorithmus[3]:

1. Eliminationsphase

   1.1. Gibt es eine Zeile z_i, die noch nicht Pivotzeile war und ein Element ungleich 0 enthält?
   - Wenn nein, gehe zu Schritt 1.4.
   - Wenn ja, dann wähle das kleinste i, sodass z_i noch nicht Pivotzeile war und ein Element ungleich 0 enthält. Wähle das betragsmäßig größte Element a_{ij} (b_i wird nicht berücksichtigt!) in der Zeile z_i. (Die i–te Zeile und die j–te Spalte heißen jetzt Pivotzeile, bzw. Pivotspalte. a_{ij} heißt jetzt Pivotelement).

   1.2. Dividiere z_i durch das Pivotelement a_{ij}. Von jeder anderen Zeile z_k, die noch nicht Pivotzeile war, subtrahiere das a_{kj}/a_{ij}–fache von z_i.

   1.3. Gehe zu Schritt 1.1.

   1.4. Gibt es eine Zeile z_l, in denen alle $a_{lj} = 0$, aber $b_l \neq 0$?
   - Wenn ja, dann ist das lineare System nicht lösbar. Breche den Algorithmus ab.
   - Wenn nein, gehe zu Schritt 2.1.

2. Rücksubstitutionsphase

   2.1. Gibt es noch Zeilen, die nicht für die Rücksubstution verwendet wurden. Wenn ja, wähle aus ihnen jene Zeile z_i aus, die als letzte Pivotzeile war und gehe zu Schritt 2.2. Wenn nein, gehe zu Schritt 3.1.

   2.2. Wähle das ehemalige Pivotelement a_{ij}. (Es muss nun gelten $a_{ij} = 1$.) Von allen Zeilen z_k, die vor z_i Pivotzeilen waren, subtrahiere das a_{kj}/a_{ij}–fache der Pivotzeile z_i.

   2.3. Gehe zu Schritt 2.1.

3. Freie Parameter und Auflösung

   3.1. Gibt es Spalten, die nie Pivotspalten waren? Setze die entsprechenden Variablen ξ_j als freie Parameter μ_j an.

   3.2. Aus jeder Zeile z_i, deren Pivotelement a_{ij} war, lässt sich der Wert von ξ_j jetzt ablesen. Ende des Verfahrens.

Basisaufgabe
Schreiben Sie ein Programm, welches lineare Gleichungssysteme der Gestalt (1) über $\mathbb{R}$ löst.

- Die Systemmatrix ist in zwei Textdateien gespeichert. Lesen Sie die Koeffizienten der Systemmatrix aus diesen Dateien ein und speichern Sie diese. Die erste Datei enthält die Werte m und $n-1$, sowie die Matrix (2). Die zweite Datei enthält n, sowie die letzte Spalte der Systemmatrix (3).

- Implementieren Sie die Schritte *Eliminationsphase* und *Rücksubstitutionsphase* des obigen Algorithmus'. (6 Pkt.)
  Geben Sie dabei die Zwischenschritte aus und markieren Sie die jeweiligen Pivotelemente mit $*$.

- Arbeiten Sie mit einer *template*–Klasse, welche die Systemmatrix abstrahiert. Achten Sie drauf, dass Sie als *template*–Parameter die Datentypen `float` und `double` übergeben können. Der *template*–Parameter soll also den Typen der Zahlen a_{ij} und β_i, also den Körper $\mathbb{K} = \mathbb{R}$ repräsentieren.

Haben Sie das erledigt, soll Ihr Programm für die Dateien matrix_5_4.txt[4] und rhs_5.txt[5] zum Beispiel den folgenden Output liefern:

```
Systemmatrix:
2       1       5       1       1       |  5
1       1       -3      -4      2       |  2
3       6       -2      1       3       |  1
2       2       2       -3      4       |  5

Eliminationsphase
0.4     0.2     *1      0.2     0.2     |  1
2.2     1.6     0       -3.4    2.6     |  5
3.8     6.4     0       1.4     3.4     |  3
1.2     1.6     0       -3.4    3.6     |  3

0.4     0.2     *1      0.2     0.2     |  1
-0.65   -0.47   -0      *1      -0.76   |  -1.5
4.7     7.1     0       0       4.5     |  5.1
-1      0       0       0       1       |  -2

0.4     0.2     *1      0.2     0.2     |  1
-0.65   -0.47   -0      *1      -0.76   |  -1.5
0.67    *1      0       0       0.63    |  0.72
-1      0       0       0       1       |  -2

0.4     0.2     *1      0.2     0.2     |  1
-0.65   -0.47   -0      *1      -0.76   |  -1.5
0.67    *1      0       0       0.63    |  0.72
*1      -0      -0      -0      -1      |  2

Rücksubstitutionsphase
0       0.2     *1      0.2     0.6     |  0.2
0       -0.47   -0      *1      -1.4    |  -0.18
0       *1      0       0       1.3     |  -0.62
*1      -0      -0      -0      -1      |  2

0       0       *1      0.2     0.34    |  0.32
0       0       0       *1      -0.8    |  -0.47
0       *1      0       0       1.3     |  -0.62
*1      -0      -0      -0      -1      |  2

0       0       *1      0       0.5     |  0.42
0       0       0       *1      -0.8    |  -0.47
0       *1      0       0       1.3     |  -0.62
*1      -0      -0      -0      -1      |  2

0       0       *1      0       0.5     |  0.42
0       0       0       *1      -0.8    |  -0.47
0       *1      0       0       1.3     |  -0.62
*1      -0      -0      -0      -1      |  2

Das System ist lösbar.
```

Tipps, die Sie nicht befolgen müssen, aber nutzen können:

- Implementieren Sie zuerst eine Klasse `Row<T>`, welche die Zeilen der Systemmatrix abstrahieren soll. *(Hinweis: Dies ist für unsere Zwecke sehr nützlich, aber für große lineare Gleichungssysteme grober Unfug!)*

- Speichern Sie die Elemente der Zeile in `std::vector<<T> > m_data`.

- Schreiben Sie eine Methode `typename std::vector<T>::iterator Row<T>::getMaxElement()` welches das betragsmäßig Großte von allen Elementen der Zeile bis auf das letzte Element liefert (dieses wird ja das Pivotelement der Zeile) und eine Methode `int std::Row<T>::getMaxElementPosition()`, welches dieses Element liefert.

- Speichern Sie, ob die Zeile eine Pivotzeile ist, sowie die Position des eventuellen Pivotelementes. Dies ergibt sich natürlich erst während der Ausführung des Algorithmus'.

- Überladen sie die Operatoren `+`, `-`, `*`, `/` der Klasse `Row<T>`, um elementare Zeilenoperationen zu abstrahieren (alternativ können Sie auch `std::val_array<<T> >` nutzen, oder sogar davon ableiten).

- Deklarieren und definieren Sie
  ```
  template<typename T>
  std::ostream& operator<<(std::ostream& s, const Row<T>& rhs);
  ```
  um eine bequeme Ausgabe einer Zeile zu ermöglichen.

[3] `http://mathematik.oehunigraz.at/files/2014/07/linalg.pdf`
[4] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/matrix_5_4.txt`
[5] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/rhs_5.txt`

- Bauen Sie Ihre Systemmatrixklasse `SysMat<T>` als `std::vector<Row<T> >` von Zeilen auf.
- Implementieren Sie die Konstruktoren, die Sie benötigen.
- Implementieren Sie die Methode `bool SysMat<T>::rref()`, welche die *Eliminationsphase* und *Rücksubstitutionsphase* auf den Zeilen ausführt und zurückgibt, ob das lineare System lösbar ist, oder nicht.
- Wenn Sie überprüfen möchten, ob ein Element gleich 0 ist, müssen Sie bei `float` und `double` aufpassen. Sie können hier beispielsweise annehmen, dass ein Element `el` ungleich 0 ist, wenn `fabs(static_cast<double>(el)) > 2.0*std::numeric_limits<double>::epsilon()` gilt.

Die Zusatzaufgaben können jeweils unabhängig voneinander gelöst werden.

Zusatzaufgabe a – Darstellung des Lösungsraumes

- Ist das lineare Gleichungssystem lösbar, geben Sie noch den Lösungsraum aus, das heißt, implementieren Sie noch den Schritt *3. Freie Parameter und Auflösung.* (+2 Pkt.)

Haben Sie das erledigt, soll Ihr Programm für die Dateien matrix_5_4.txt[6] und rhs_5.txt[7] zum Beispiel den folgenden Output liefern:

```
The (affine) solution space is given by the set
{ (0.42)        - mu_4(0.5)    }
{ (-0.47)       - mu_4(-0.8)   }
{ (-0.62)       - mu_4(1.3)    }
{ (2)           - mu_4(-1)     }
```

Zusatzaufgabe b – Lineare Gleichungssysteme über $\mathbb{Q}$

Lineare Gleichungssysteme können nicht nur über $\mathbb{R}$ oder $\mathbb{C}$ formuliert werden, sondern über beliebigen Körpern $\mathbb{K}$.

- Sie haben in Beispiel 22 eine Klasse `bruch` implementiert, die eine rationale Zahl $q \in \mathbb{Q}$ abstrahiert.
- Sie haben Ihre Klasse `SysMat<T>` bereits templatisiert. Erweitern Sie Ihre Klasse `bruch` soweit, dass Sie lineare Gleichungssysteme über $\mathbb{Q}$ lösen können. (+2 Pkt.)
- Eventuell müssen Sie die Methode `bool Matrix<T>::rref()` auch anpassen. Dies wird dann der Fall sein, wenn die Methode nicht wirklich gemäß der *generic–programming*–Denkweise implementiert wurde.

Tipps, die Sie nicht befolgen müssen, aber nutzen können:

- Deklarieren und definieren Sie `std::istream& operator>>(std::istream& s, bruch& fraction)`, um elegant Bruchzahlen aus den Textdateien, die die Koeffizienten der Systemmatrix in der Form `z/n` beinhalten, einzulesen zu können:

```
std::istream& operator>>(std::istream& s, bruch& fraction) {
    std::string str; s >> str;
    std::replace( str.begin(), str.end(), '/', ' ');

    std::istringstream ss(str);
    int nom; ss >> nom;
    int den; ss >> den;
```

[6] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/matrix_5_4.txt
[7] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/rhs_5.txt

```
    fraction = bruch(nom,den);

    return s;
}
```

- Überladen Sie die Vergleichsoperatoren `<,<=,>,>=` für die Bruchklasse, um Brüche vergleichen zu können.
- Sie müssen an irgendeiner Stelle in der Eliminationsphase überprüfen, ob ein Element `el` ungleich 0 ist. Wenn Sie sich bei der Basisaufgabe dazu entschieden habe, zu überprüfen ob `fabs(static_cast<double>(el)) > 2.0*std::numeric_limits<double>::epsilon()` gilt, können Sie in der Bruchklasse den *double* Operator überladen und so eine rationale Zahl in $\mathbb{R}$ einlagern:

```
bruch::operator double() const { return nom/den; }
```

Haben Sie das erledigt, soll Ihr Programm für die Dateien matrix_5_4_frac.txt[8] und rhs_5_frac.txt[9] zum Beispiel den folgenden Output liefern:

```
Systemmatrix:
2       1       5       1       1       |  5
1       1       -3      -4      2       |  2
3       6       -2      1       3       |  1
2       2       2       -3      4       |  5

Eliminiationsphase
2/5     1/5     *1      1/5     1/5     |  1
11/5    8/5     0       -17/5   13/5    |  5
19/5    32/5    0       7/5     17/5    |  3
6/5     8/5     0       -17/5   18/5    |  3

2/5     1/5     *1      1/5     1/5     |  1
-11/17  -8/17   0       *1      -13/17  |  -25/17
80/17   120/17  0       0       76/17   |  86/17
-1      0       0       0       1       |  -2

2/5     1/5     *1      1/5     1/5     |  1
-11/17  -8/17   0       *1      -13/17  |  -25/17
2/3     *1      0       0       19/30   |  43/60
-1      0       0       0       1       |  -2

2/5     1/5     *1      1/5     1/5     |  1
-11/17  -8/17   0       *1      -13/17  |  -25/17
2/3     *1      0       0       19/30   |  43/60
*1      0       0       0       -1      |  2
```

```
Rücksubstitutionsphase
0       1/5     *1      1/5     3/5     |  1/5
0       -8/17   0       *1      -24/17  |  -3/17
0       *1      0       0       13/10   |  -37/60
*1      0       0       0       -1      |  2

0       0       *1      1/5     17/50   |  97/300
0       0       0       *1      -4/5    |  -7/15
0       *1      0       0       13/10   |  -37/60
*1      0       0       0       -1      |  2

0       0       *1      0       1/2     |  5/12
0       0       0       *1      -4/5    |  -7/15
0       *1      0       0       13/10   |  -37/60
*1      0       0       0       -1      |  2

0       0       *1      0       1/2     |  5/12
0       0       0       *1      -4/5    |  -7/15
0       *1      0       0       13/10   |  -37/60
*1      0       0       0       -1      |  2

Das System ist lösbar.
```

Zusatzaufgabe c – Lineare Gleichungssysteme über $\mathbb{Z}/p\mathbb{Z}$ (für Algebraiker)

Sei p eine Primzahl. Wir betrachten die Menge

$$\mathbb{Z}/p\mathbb{Z} = \{[0], [1], ..., [p-1]\} .$$

Für $[x], [y] \in \mathbb{Z}/p\mathbb{Z}$ definieren wir die Operatoren

$$+ \colon \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} \quad \text{und} \quad \cdot \colon \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$$

durch

$$[x] + [y] = (x + y)\ mod\ p \quad \text{und} \quad [x] \cdot [y] = (xy)\ mod\ p\ .$$

Dann ist $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$ ein (endlicher) Körper.
Für die Subtraktion

$$- \colon \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$$

[8] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/matrix_5_4_frac.txt
[9] http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/rhs_5_frac.txt

gilt dann:

$$[x] - [y] = (x - y) \; mod \; p \; .$$

Für die Division

$$/ : \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \backslash \{[0]\} \rightarrow \mathbb{Z}/p\mathbb{Z}$$

gilt: gilt weiters

$$[x]/[y] = [x] \cdot [y]^{p-2} \; .$$

- Implementieren Sie analog zur Klasse `bruch` und zu Zusatzaufgabe b eine Klasse `GF<unsigned int p>` und stellen Sie darin die Funktionalität zur Verfügung, um lineare Gleichungssysteme über $\mathbb{Z}/p\mathbb{Z}$ zu lösen. +2 Pkt.
- Der Parameterkonstruktor `GF::GF(unsigned int x)` bekommt das darzustellende Körperelement x als Parameter.

Haben Sie das erledigt, soll Ihr Programm für die Dateien matrix_3_3_p5.txt[10] und rhs_3_p5.txt[11] zum Beispiel den folgenden Output liefern:

```
System Matrix:
[4]     [0]     [3]     |  [2]
[2]     [2]     [2]     |  [3]
[2]     [3]     [1]     |  [4]

Elimination:
*[1]    [0]     [2]     |  [3]
[0]     [2]     [3]     |  [2]
[0]     [3]     [2]     |  [3]

*[1]    [0]     [2]     |  [3]
[0]     [4]     *[1]    |  [4]
[0]     [0]     [0]     |  [0]

*[1]    [0]     [2]     |  [3]
[0]     [4]     *[1]    |  [4]
[0]     [0]     [0]     |  [0]
```

Zusatzaufgabe d – Determinante für quadratische Matrizen

Ist die Matrix A quadratisch ($m = n$), so kann man ihre Determinante $det(A)$ berechnen. Falls A weniger als $m = n$ Pivotzeilen hat, gilt: $det(A) = 0$ Falls A genau $m = n$ Pivotzeilen hat, gilt: $det(A)$ ist das Produkt der Pivotelemente aus Schritt 1.2 des Algorithmus'. (Also bevor das Pivotelement durch sich selbst dividiert wird.)

- Berechnen Sie $det(A)$, falls A eine quadratische Matrix ist und geben Sie ihren Wert aus. +1 Pkt.

[10] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/matrix_3_3_p5.txt`
[11] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/LinSys_Ex/rhs_3_p5.txt`