Dünnbesetzte Matrizen

Einleitung

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt dünnbesetzt (engl.: *sparse*), wenn sie nur wenige Einträge $a_{ij} \neq 0$ für $i,j=0,\ldots,n-1$ hat. Die Anzahl dieser von Null verschiedenen Einträge nennen wir $r \in \mathbb{N}$. Das Gegenteil einer dünnbesetzen Matrix ist eine dichtbesetze Matrix (engl.: *dense*).

Dünnbesetzte Matrizen treten sehr häufig in der numerische Mathematik auf, wenn man z. B. Wärmeleitung oder Fluidströmungen berechnen möchte. Daher hat man sich ein Format zur einfacheren Speicherung überlegt: die komprimierte Zeilenspeicherung (engl.: *Compressed Row Storage, CRS*). Der Vorteil dieses Formats liegt nicht nur in dem geringeren Speicherverbrauch, sondern auch in der schnelleren Arithmetik.

Wir betrachten in unserem Fall nur reguläre Matrizen, obwohl die *CRS* auch singuläre Matrizen speichern kann.

In der CRS gibt es drei Datenfelder:

- Das erste Feld `wa` enthält die von Null verschiedenen reellen Werte von A und hat die Länge r.
- Das zweite Feld `ja` enthält die Spaltenindizes zu den Zeilen von A und hat die Länge r.
- Das dritte Feld `ia` enthält die Startindizes der Zeilen von A und als letztes Element die Dimension der Matrix n. Der Startindex `k = ia[m]` gibt an, dass die Zeile `m` mit dem Wert `wa[k]` in der Spalte `ja[k]` beginnt, also das erste von 0 verschiedene Element ist. Das Feld hat die Länge $n+1$.

Zum besseren Verständnis werden wir drei farbige Counter benutzen:

- Die Zeilenindizes sind ROT.
- Die Spaltenindizes sind GRÜN.
- Die Startindizes sind BLAU.

In `wa` sind die Werte von A von links nach rechts und von oben nach unten eingetragen:

Beispiel 1.1:

$$A = \begin{pmatrix} \pi & 0 & sin(2) \\ 0 & 0 & \sqrt{3} \\ 0 & 4! & 0 \end{pmatrix} \Rightarrow \begin{cases} \text{wa} = [\pi, sin(2), \sqrt{3}, 4!] \\ \text{Länge } r = 4 \end{cases}$$

`ia` enthält die Startindizes der Zeilen 0 bis n-1. Es gilt immer `ia[n] = r`.

Der Index `k = ia[1]` gibt an, dass die Zeile 1 mit dem Wert `wa[k]` in der Spalte `ja[k]` beginnt.

Beispiel 1.2:

$$A = \begin{pmatrix} \pi & 0 & sin(2) \\ 0 & 0 & \sqrt{3} \\ 0 & 4! & 0 \end{pmatrix} \Rightarrow \begin{cases} \text{ia} = [0,2,3,4] \\ \text{Länge } n+1 \\ r = \text{ia}[n] = 4 \end{cases}$$

Lies:

- „Zeile 0 beginnt am Index `ia[0]`= 0 mit dem Wert `wa[0]`=π."
- „Zeile 1 beginnt am Index `ia[1]`= 2 mit dem Wert `wa[2]`=$\sqrt{3}$."
- „Zeile 2 beginnt am Index `ia[2]`= 3 mit dem Wert `wa[3]`=4!."

`ja` enthält die Spaltenindizes der Werte `wa` in der Originalmatrix.

Beispiel 1.3:

$$A = \begin{pmatrix} \pi & 0 & sin(2) \\ 0 & 0 & \sqrt{3} \\ 0 & 4! & 0 \end{pmatrix} \Rightarrow \begin{cases} \text{ja} = [0,2,2,1] \\ \text{Länge} = r = 4 \end{cases}$$

Lies:

- „In Zeile 0 (ia[0] = 0) liegt der Wert wa[0]=π in Spalte ja[0]= 0."
  Also $a_{00}=\pi$.

  Wir erhöhen den blauen Counter um eins (auf 1). Zeile 1 beginnt aber erst bei ia[1]= 2, darum muss es noch einen weiteren Eintrag in Zeile 0 geben:

- „In Zeile 0 liegt der Wert wa[1]= $sin(2)$ in Spalte ja[1]= 2."
  also $a_{02}=sin(2)$ und der Wert dazwischen $a_{01}=0$.

  Wir erhöhen den blauen Counter um eins (auf 2). Hier beginnt Zeile 1.

- „In Zeile 1 (ia[1]= 2) liegt der Wert wa[2]= $\sqrt{3}$ in Spalte ja[2]= 2."
  also $a_{12}= \sqrt{3}$ und alle Werte davor $a_{10}=a_{11}=0$.

  Wir erhöhen den blauen Counter um eins (auf 3). Hier beginnt Zeile 2.

- „In Zeile 2 (ia[2] = 3) liegt der Wert wa[3]= 4! in Spalte ja[3]= 1."
  also $a_{21}=4!$ und der Wert davor $a_{20}=0$.

  Wir erhöhen den blauen Counter um eins (auf 4). Wir haben nun 4 Werte ihrer Position zugeordnet, da r=5 sind wir fertig und die restlichen Werte sind alle 0.

Beispiel 2:

$$A = \begin{pmatrix} \pi & 0 & sin(2) & 0 \\ 0 & 0 & \sqrt{3} & 0 \\ 0 & 4! & 0 & 0 \\ 0 & e^5 & 0 & \frac{1}{6} \end{pmatrix} \Rightarrow \begin{cases} \text{wa} = \left[\pi, sin(2), \sqrt{3}, 4!, e^5, \frac{1}{6}\right] \\ \text{ia} = [0,2,3,4,6] \\ \text{ja} = [0,2,2,1,1,3] \\ n = 4, r = 6 \end{cases}$$

Aufgabenstellung

Es soll eine Programmbibliothek für reguläre Matrizen erstellt werden, welche Matrizen im *CRS*-Format speichert und die folgenden Operationen bereitstellt:

- Addition von Matrizen
- Multiplikation von zwei Matrizen
- Multiplikation von Matrizen mit Skalaren

Es soll explizit im *CRS*-Format gerechnet werden, ohne eine Umwandlung in vollbesetzte Matrizen vorzunehmen. Für diese Aufgabe dürfen keine vorgefertigten Bibliotheken zum Umgang mit CRS-/CSR-Formaten genutzt werden.

Hinweise

Definition der Eingabedatei:
Eine Eingabedatei hat immer folgenden Aufbau. Sie beginnt mit drei Kommentarzeilen, welche den Namen des Beispiels enthalten können. Kommentare werden mit ‚#' eingeleitet. Leerzeilen werden ignoriert. Danach folgt entweder eine Matrix in CRS-Schreibweise oder ein Skalar. Darauf eine Zeile mit einem Rechenzeichen (‚+' bzw. ‚*'), gefolgt von einer weiteren Matrix oder einem Skalar.
Matrizen A und B werden mit den drei Feldern ia/ib, ja/jb und wa/wb gefolgt von einem Doppelpunkt und mit Kommata getrennten Werten definiert.
Bei allen reellen Zahlen wird der Punkt als Dezimaltrennzeichen verwendet.

Beispiele von Eingabedateien:

```
#
# Beispiel 1
#

ia:0,2,3,4
ja:0,2,2,1
wa:3.14,0.91,1.73,24

*

ib:0,2,3,4
jb:1,2,0,1
wb:44,33,22,11
```

```
#
# Beispiel 2
#

ia:0,2,3,4,6,7,8,9,10,11
ja:1,5,7,4,7,8,2,6,0,3,5
wa:11,111,22,33,222,44,55,66,77,88,99

*

ib:0,2,3,4,6,7,8,9,10,11
jb:1,5,7,4,7,8,2,6,0,3,5
wb:1,10,2,3,11,4,5,6,7,8,9
```

```
#
# Beispiel 3
#

ia:0,2,3,4
ja:0,2,2,1
wa:11,22,33,44

*

5.3
```

```
#
# Beispiel 4
#

4

*

ib:0,2,3,4
jb:0,2,2,1
wb:44,33,22,11
```

Die Ergebnisse sollen in eine Textdatei ausgegeben werden. Dabei wird die Ergebnismatrix C im obigen *CRS*-Format dargestellt, d. h. mit den Wertefeldern `ic, jc, wc`. Die Ausgabedatei beginnt mit einem dreizeiligen Kommentarbereich, gefolgt von dem Ergebnis der Rechnung und der Größe aller in der Rechnung vorkommenden Matrizen (A, B und C) in Byte.

Die Angaben in Byte können vom Betriebssystem, der verwendeten Programmiersprache und des Kompilers/Interpreters abhängen. Daher sind die hier in den Beispielen genannten Zahlen nur Richtwerte.

Beispiel einer Ausgabedatei:

```
#
# Ergebnis Beispiel 1
#

ic:0,2,3,4
jc:1,2,1,0
wc:148.17,103.62,19.03,528.0

A CRS: 64 Bytes
B CRS: 64 Bytes
C CRS: 64 Bytes
C vollbesetzt: 72 Bytes
```

```
#
# Ergebnis Beispiel 2
#

ic:0,2,3,4,6,7,8,10,12,13
jc:6,7,3,2,3,5,4,0,1,5,7,8,6
wc:666.0,22.0,176.0,165.0,1776.0,396.0,165.0,462.0,77.0,770.0,968.0,352.0,594.0

A CRS: 172 Bytes
B CRS: 172 Bytes
C CRS: 196 Bytes
C vollbesetzt: 648 Bytes
```

```
#
# Ergebnis Beispiel 3
#

ic:0,2,3,4
jc:0,2,2,1
wc:58.3,116.6,174.9,233.2

A CRS: 64 Bytes
B CRS: 0 Bytes
C CRS: 64 Bytes
C vollbesetzt: 72 Bytes
```

```
#
# Ergebnis Beispiel 4
#

ic:0,2,3,4
jc:0,2,2,1
wc:176.0,132.0,88.0,44.0

A CRS: 0 Bytes
B CRS: 64 Bytes
C CRS: 64 Bytes
C vollbesetzt: 72 Bytes
```

bitte wenden!