Spezifikation zum Projekt Nr. 2 der Gruppe 7 (ss11-g07)

Tutor: Steffen Bauereiss

Kurzbeschreibung

Es ist eine Funktion DGBMV zu entwickeln, welche Matrix-Vektor-Operationen in Abhängigkeit von TRANS durchführt. Wenn TRANS='N' oder 'n', wird die folgende Formel verwendet:

```
Y := ALPHA * A * X + BETA * Y.
```

Wenn TRANS='T' oder 't' oder 'C' oder 'c', wird die folgende Formel verwendet:

```
Y := ALPHA * A' * X + BETA * Y.
```

ALPHA, BETA seien Skalare; X, Y seien Vektoren; A sei eine Matrix. Die Werte bzw. Pointer werden durch ein Wrapperprogramm auf den Stack gelegt. A' sei transponiertes A.

Alternative 1: Berechnung in Normalform

A, X und Y werden jeweils in A2, X2 und Y2 gesichert. Die Matrix A in Bandspeicherform wird (mit Hilfe des invertierten Subprogramms aus der Aufgabenstellung) in eine MxN-Bandmatrix in Normalform umgewandelt und in A gespeichert. Die Vektoren werden jeweils aus den inkrementierten Arrays X bzw. Y (mit Hilfe der Variablen INCX bzw. INCY) extrahiert und in X bzw. Y gespeichert.

Dann wird nach folgendem Schema vorgegangen:

- Wenn TRANS='T' oder 't' oder 'C' oder 'c', transponiere die Matrix A (transponierte Matrix[1]) und speichere das Ergebnis in A.
- Multipliziere (Skalarmultiplikation[2]) die Matrix A mit dem Skalar ALPHA und speichere das Ergebnis in AA.
- Multipliziere (Skalarmultiplikation[3]) den Vektor Y mit dem Skalar BETA und speichere das Ergebnis in YB.
- Multipliziere (Matrix-Vektor-Multiplikation[4]) die Matrix AA mit dem Vektor X und speichere das Ergebnis in AAX.
- Addiere (Vektoraddition[5]) die Vektoren AAX und YB und speichere das Ergebnis in AAXYB.
- Setze den Vektor AAXYB in Y2 entsprechend INCY ein und gebe es als inkrementiertes Array Y zurück.
- Überschreibe A mit A2 und X mit X2.

[1] transponierte Matrix:
$[A]_{ji}$ wird überschrieben mit $[A]_{ij}$

[2] Skalarmultiplikation:
AA: $aa_{ij} = (a_{ij} \cdot \text{ALPHA})_{\ i=1..M;\ j=1..N}$

[3] Skalarmultiplikation:
YB: $yb_i = (y_i \cdot \text{BETA})_{\ i=1..n}$

[4] Matrix-Vektor-Multiplikation:

$$\text{AAX: } aax_i = \sum_{k=1}^{N} (a_{ik} \cdot x_k)_{\ i=1..M}$$

[5] Vektoraddition:
AAXYB: $aaxyb_i = (aax_i + yb_i)_{\ i=1..N}$

N=M betrage jeweils mindestens 1.

Alternative 2: Berechnung in Bandspeicherform

A, X und Y werden jeweils in A2, X2 und Y2 gesichert. Die Vektoren werden jeweils aus den inkrementierten Arrays X bzw. Y (mit Hilfe der Variablen INCX bzw. INCY) extrahiert und in X bzw. Y gespeichert.

Dann wird nach folgendem Schema vorgegangen:

- Wenn TRANS='T' oder 't' oder 'C' oder 'c', transponiere die Matrix A in Bandspeicherform (transponierte Matrix in Bandspeicherform[1]) und speichere das Ergebnis in A.
- Multipliziere (Skalarmultiplikation[2]) die Matrix A mit dem Skalar ALPHA und speichere das Ergebnis in AA.
- Multipliziere (Skalarmultiplikation[3]) den Vektor Y mit dem Skalar BETA und speichere das Ergebnis in YB.
- Multipliziere (Bandmatrix-Vektor-Multiplikation[4]) die Matrix AA mit dem Vektor X und speichere das Ergebnis in AAX.
- Addiere (Vektoraddition[5]) die Vektoren AAX und YB und speichere das Ergebnis in AAXYB.
- Setze den Vektor AAXYB in Y2 entsprechend INCY ein und gebe es als inkrementiertes Array Y zurück.
- Überschreibe A mit A2 und X mit X2.

[1] transponierte Matrix in Bandspeicherform:
Die Zeile KU+1 wird in das neue Array in Zeile KL+1 eingefügt; die Zeilen 1..KU werden jeweils in das neue Array in Zeile KU+KL+1..KL+2 eingefügt, wobei sie erst einmal jeweils so lange nach links geshiftet werden, bis die erste Position der jeweiligen Zeile definiert (referenziert) ist; die Zeilen KU+KL+1..KU+2 werden jeweils in das neue Array in Zeile 1..KL eingefügt, wobei sie erst einmal so lange nach rechts geshiftet werden, bis die letzte Position der jeweiligen Zeile definiert (referenziert) ist. Abschließend wird das alte Array A mit dem neu errechneten überschrieben. *Die erste Zeile/Spalte des Arrays sei hier 1, nicht 0.*

[2] Skalarmultiplikation:
AA: $aa_{ij} = (a_{ij} \cdot ALPHA)_{\ i=1..M;\ j=1..N}$

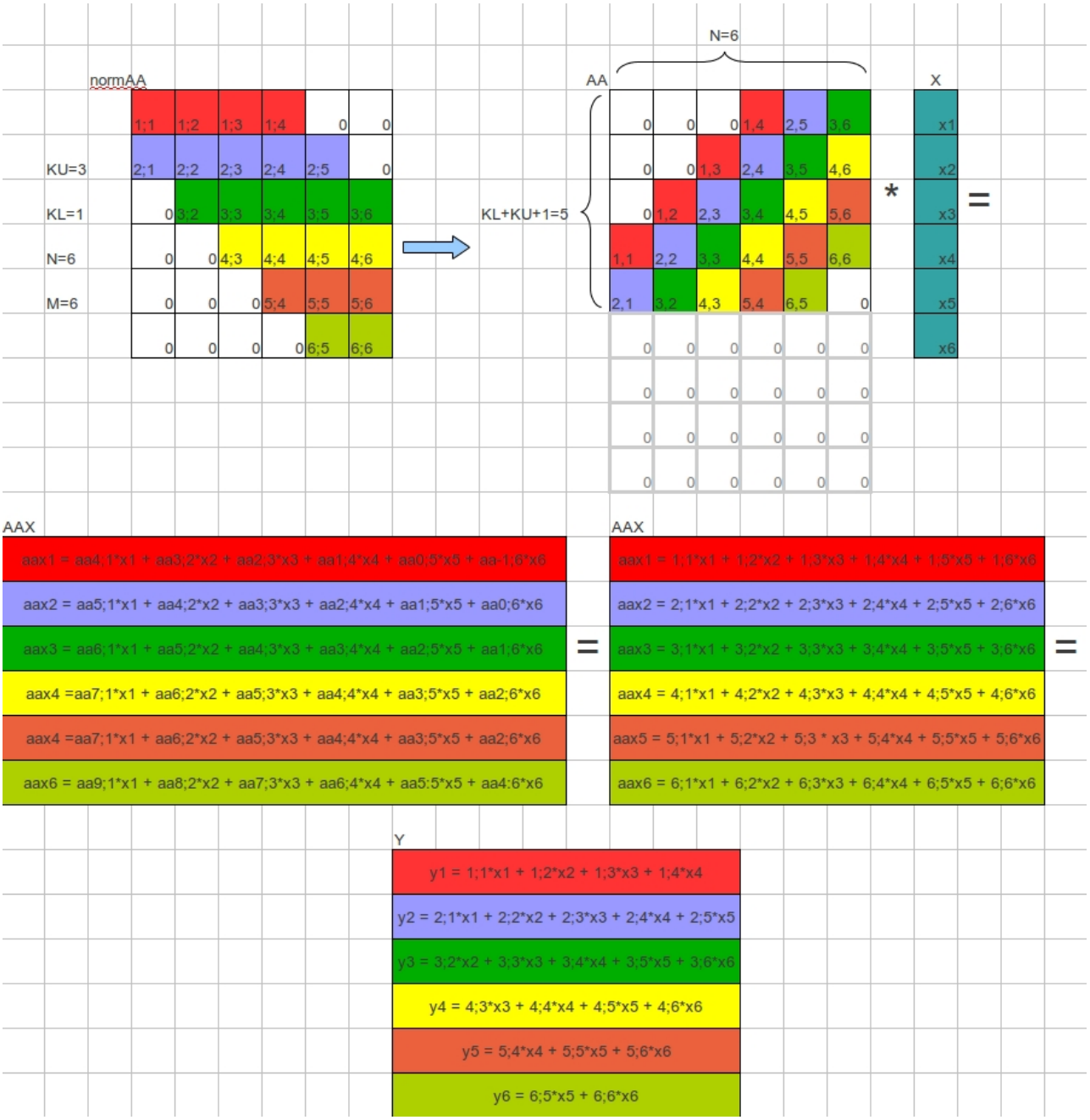
[3] Skalarmultiplikation:
YB: $yb_i = (y_i \cdot BETA)_{\ i=1..n}$

[4] Bandspeichermatrix-Vektor-Multiplikation:

$$\text{AAX: } aax_i = \sum_{k=0}^{N-1} (x_{k+1} \cdot aa_{KU+i-k;k+1})_{\ i=1..N}$$

Sollte dabei eins der Elemente, auf die zugegriffen wird, sich nicht im Array befinden, so ist es für die Berechnung als gleich 0 zu behandeln. Die erste Zeile/Spalte des Arrays sei hier 1, nicht 0.

Beispielhaftes Schema:



Beispielhaftes Python-Skript (s. bandmatrix-vector-mult.py):

```
for i in xrange(1,N+1):
    for k in xrange(0,N):
        if (KU+i-k-1) >= 0:
            if (KU+i-k-1) <= (LDA-1):
                AAXYB[i-1] = AAXYB[i-1] + (X[k+1-1] * A[KU+i-k-1][k+1-1])
```

[5] Vektoraddition:
AAXYB: $aaxyb_i = (aax_i + yb_i)_{i=1..N}$

N=M betrage jeweils mindestens 1.

Bewertung der Alternativen

Alternative 1 (A1) ist leichter zu durchschauen, da hier die üblichen Matrix-Vektor-Operationen verwendet werden. Somit kann man im beliebigem Stadium der Berechnung ihren aktuellen Zustand intuitiv nachvollziehen.

Alternative 2 (A2) ist effizienter, da mit der übergebenen Matrix in Bandspeicherform gerechnet wird. Dies erfordert allerdings eine spezielle und nicht unbedingt intuitive Multiplikations-Operation für Matrizen in Bandspeicherform. Da aber auf die Normalform-Konversion verzichtet werden kann, sinkt die Komplexität der Implementierung und somit auch die Fehleranfälligkeit.

A2 verfügt also über ein besseres Cacheverhalten, eine kürzere Laufzeit und eine höhere Speichereffizienz, ist aber weniger intuitiv und eventuell schwieriger zu diagnostizieren. A1 ist im Vergleich zu A2 etwas langsamer in der Ausführung und benötigt etwas mehr Speicherplatz, lässt sich dafür aber leicher verstehen und durchschauen.

Implementierungsentscheidung

Wir haben uns für die Implementierung in Bandspeicherform aus Alternative 2 entschieden, da wir ihre hohe Effizienz und relativ geringe Komplexität der Übersichtlichkeit von A1 vorziehen.