

Kapitel 10

Vektorräume

Verständnisfragen

Sachfragen

1. Was ist ein allgemeiner Vektorraum?
2. Was ist ein Unterraum?
3. Erläutern Sie den Begriff der Linearkombination!
4. Was ist ein Spann von Vektoren?
5. Wann sind Vektoren linear unabhängig bzw. abhängig?
6. Erläutern Sie den Begriff der Basis!
7. Was versteht man unter Basis-Koordinaten?
8. Was ist ein Koordinatenvektor?
9. Erläutern Sie den Begriff der Dimension!
10. Was ist eine Basistransformation?
11. Wie kann die Basistransformationsmatrix gebildet werden?
12. Was ist ein Zeilenraum einer Matrix?
13. Was ist ein Spaltenraum einer Matrix?
14. Was ist der Nullraum einer Matrix?
15. Wie kann die Lösbarkeit eines linearen Gleichungssystems mit Hilfe des Rangbegriffs beschrieben werden?
16. Wie kann die allgemeine Lösung eines linearen Gleichungssystems beschrieben werden?
17. Was ist ein Skalarprodukt?
18. Was ist eine Norm?
19. Wie kann mit Hilfe des Skalarprodukts eine Norm und ein Abstands begriff definiert werden?

20. Was ist eine Orthonormalbasis?
21. Wie können die Basiskoeffizienten bezüglich einer Orthonormalbasis bestimmt werden?
22. Hat jeder n -dimensionaler Vektorraum eine Orthonormalbasis?
23. Erläutern Sie die QR -Zerlegung einer $m \times n$ -Matrix!
24. Was ist eine Householder-Transformation?
25. Wie kann mit Hilfe von Householder-Transformationen eine QR -Zerlegung berechnet werden?

Methodenfragen

1. Für eine Menge, einer Addition und einer skalaren Multiplikation die Vektorraum-Gesetze nachrechnen können.
2. Überprüfen können, ob eine Teilmenge eines Vektorraums ein Unterraum ist.
3. Gegebene Vektoren auf lineare Unabhängigkeit überprüfen können.
4. Die Basiskoordinaten eines Elements eines gegebenen Spans von Basisvektoren berechnen können.
5. Die Dimension eines Spans bestimmen können.
6. Die Basistransformationsmatrix zwischen zwei Unterräumen aufstellen können.
7. Spalten-, Zeilen- und Nullraum einer gegebenen Matrix bestimmen können.
8. Eine Basis für einen gegebenen Spann bestimmen können.
9. Die Basis des Nullraums einer Matrix bestimmen können.
10. Für einen n -dimensionalen Vektorraum eine Orthonormalbasis mit Hilfe des Gram-Schmidt-Verfahrens bestimmen können.
11. Die QR -Zerlegung einer Matrix mit Hilfe des Gram-Schmidt-Verfahrens bestimmen können.
12. Die lineare Ausgleichsrechnung mit Hilfe der QR -Zerlegung lösen können.
13. Für einen gegebenen Vektor die Householder-Transformation bilden und anwenden können.
14. Die QR -Zerlegung mit Hilfe von Householder-Transformationen anwenden und implementieren können.

Übungsaufgaben

1. Weisen Sie die Gültigkeit der Vektorraumaxiome für $F[0; 1]$ nach!

Lösung:

Die Vektorraumgesetze folgen aus den Eigenschaften der Arithmetik der reellen Zahlen und der punktweisen Definition der Vektorraum-Operationen.

2. Ist die Menge $V = \{(x, y) \in \mathbb{R}^2\}$ mit $(x, y) + (x', y') = (x + x', y + y')$, $\lambda(x, y) = (2\lambda x, 2\lambda y)$ ein Vektorraum?

Lösung:

Nein, denn weder $(\lambda\mu)(x, y) = \lambda(\mu(x, y))$ noch $1(x, y) = (x, y)$ sind erfüllt!

3. Ist die Menge $\{f \in F[0; 1] \mid f(1) = 0\}$ mit den aus $F[0; 1]$ übernommenen Operationen ein Vektorraum?

Lösung:

Diese Menge ist ein Vektorraum. Einerseits ist sie ein Unterraum der $F[0; 1]$. Man kann auch einfach nachweisen, dass die Addition abgeschlossen ist, die skalare Multiplikation auch und die Gültigkeit der Vektorraumgesetze nachrechnen.

4. Welche der folgenden Teilmengen von $P_3[0; 1]$ sind Unterräume: $U_1 = \{p \in P_3[0; 1] \mid a_0 = 0\}$, $U_2 = \{p \in P_3[0; 1] \mid a_0 + a_1 + a_2 + a_3 = 0\}$?

Lösung:

Beides sind Unterräume. Das Nullpolynom ist in beiden enthalten; und die beiden Vektorraumoperationen sind abgeschlossen.

Für zwei Polynome mit $a_0 = b_0 = 0$ hat auch die Summe das konstante Element 0.

Sind p und q zwei Polynome mit $p(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ und $q(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ mit $a_3 + a_2 + a_1 + a_0 = 0 = b_3 + b_2 + b_1 + b_0$, dann ist auch $(p + q)(x)$ in dieser Menge, denn $(a_3 + b_3) + (a_2 + b_2) + (a_1 + b_1) + (a_0 + b_0) = (a_3 + a_2 + a_1 + a_0) + (b_3 + b_2 + b_1 + b_0) = 0$. Analog können Sie die Abgeschlossenheit der skalaren Multiplikation nachweisen.

5. Sind die Vektoren $(0, 3, 1, -1)^T$, $(6, 0, 5, 1)^T$, $(4, -7, 1, 3)^T \in \mathbb{R}^4$ linear unabhängig?

Lösung:

Nein. Wenn Sie die drei Vektoren in einer 3×4 -Matrix schreiben und Gauß-Elimination durchführen, dann erhalten Sie die eine Matrix vom Rang 2. Dann können die Vektoren aber nicht unabhängig sein!

Beispielsweise erhält man aus

$$\begin{pmatrix} 4 & -7 & 1 & 3 \\ 0 & 3 & 1 & -1 \\ 6 & 0 & 5 & 1 \end{pmatrix}$$

durch Gauß-Elimination

$$\begin{pmatrix} 4 & -7 & 1 & 3 \\ 0 & 3 & 1 & -1 \\ 0 & 0 & 0 & \end{pmatrix}.$$

6. \mathbf{x}_3 sei ein Vektor eines Vektorraums, der nicht in $[\mathbf{x}_1, \mathbf{x}_2]$ liegt. Zeigen Sie, dass $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ linear unabhängig sind!

Lösung:

Angenommen, die Vektoren wären linear abhängig. Dann könnten Sie \mathbf{x}_3 als Linearkombination darstellen, ein Widerspruch. Denn dann würde \mathbf{x}_3 ja im Spann von \mathbf{x}_1 und \mathbf{x}_2 liegen!

7. Welche der Unterräume $U_1 = \left[\begin{pmatrix} 1 \\ 1 \\ 5 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \\ 13 \end{pmatrix} \right]$, $U_2 = \left[\begin{pmatrix} 1 \\ -1 \\ -2 \end{pmatrix}, \begin{pmatrix} 3 \\ -2 \\ -3 \end{pmatrix} \right]$ und $U_3 = \left[\begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 4 \\ -3 \\ -1 \end{pmatrix}, \begin{pmatrix} 3 \\ -1 \\ 3 \end{pmatrix} \right]$ sind gleich?

Lösung:

Die Basis der drei Spalten kann durch Gauss-Elimination bestimmt werden. Schreiben Sie wie im Buch auf Seite 255 angegeben die gegebenen Vektoren als Spaltenvektoren in einer Matrix und führen elementare Zeilenumformungen durch. Dann können Sie die Basen der drei Unterräume ablesen.

Für U_1 und U_2 erhalten Sie die Basen $\{\mathbf{e}_1, \mathbf{e}_2\}$, für U_3 erhält man die Zeilenstufenform

$$\begin{pmatrix} 1 & 0 & -5 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix},$$

dann ist $\{\mathbf{e}_1, \mathbf{e}_2\}$ auch eine Basis von U_3 . Dann stimmen aber alle drei Unterräume überein!

8. Für welche Werte von $a \in \mathbb{R}$ sind die Vektoren $(0, 1, a)^T$ und $(1, a, 0)^T$ linear abhängig?

Lösung:

Wenn Sie die beiden Vektoren als Spaltenvektoren einer 2×2 -Matrix schreiben und Gauß-Elimination durchführen erhalten Sie die Matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Diese hängt nicht mehr von a ab, also sind die Vektoren für alle $a \in \mathbb{R}$ linear unabhängig!

9. Welche Dimension hat der Unterraum $[p_0, p_1, p_2, p_3] \subset P_3$ mit $p_0(x) = 1$, $p_1(x) = x$, $p_2(x) = \frac{1}{2}(3x^2 - 1)$, $p_3(x) = \frac{1}{2}(5x^3 - 3x)$?

Lösung:

Sie können die Basiskoeffizienten der vier Polynome bezüglich der Monom-Basis $\{1, x, x^2, x^3\}$ ablesen und als Spaltenvektoren in einer Matrix schreiben:

$$\begin{pmatrix} 1 & 0 & -\frac{1}{2} & 0 \\ 0 & 1 & 0 & -\frac{3}{2} \\ 0 & 0 & \frac{3}{2} & 0 \\ 0 & 0 & 0 & \frac{5}{2} \end{pmatrix}.$$

Diese Matrix wird durch elementare Zeilenumformungen auf I_4 überführt. Dann sind die vier Polynome aber linear unabhängig, der Unterraum hat die Dimension 4.

10. Bestimmen Sie die Basistransformationsmatrix P für $B = \{(1, 1)^T, (2, 1)^T\}$ und $B' = \{\mathbf{e}_1, \mathbf{e}_2\}$ und die Koordinaten des Vektors $\mathbf{z} = (-3, 5)^T$ bezüglich B !

Lösung:

Es ist $\mathbf{e}_1 = -(1, 1)^T + (2, 1)^T$ und $\mathbf{e}_2 = 2(1, 1)^T - (2, 1)^T$, also lautet die Basistransformationsmatrix

$$P = \begin{pmatrix} -1 & 2 \\ 1 & -1 \end{pmatrix}.$$

Als Koordinaten des Vektors $\mathbf{z} = (-3, 5)^T$ (angegeben bezüglich den kanonischen Einheitsvektoren) erhalten Sie durch $P\mathbf{z}$ das Ergebnis $(13, -8)^T$.

11. Die kubischen Bernsteinpolynome sind definiert durch $B_0(x) = (1-x)^3$, $B_1(x) = 3x(1-x)^2$, $B_2(x) = 3x^2(1-x)$ und $B_3(x) = x^3$. Weisen Sie nach, dass diese Polynome eine Basis von $P_3[0; 1]$ sind und bestimmen Sie die Basistransformationsmatrix für den Übergang zur Monombasis und zur Basis $\{H_0^3, H_1^3, H_2^3, H_3^3\}$ der Hermite-Polynome!

Lösung:

Wenn Sie die Definition der Bernstein-Polynome ausmultiplizieren und so ablesbaren Basiskoeffizienten bezüglich der Monom-Basis $\{1, x, x^2, x^3\}$ als Spaltenvektoren in eine Matrix

schreiben, erhalten Sie die untere Dreiecksmatrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix}.$$

Dann sind die vier Bernstein-Polynome aber linear unabhängig.

Durch Nachrechnen können Sie überprüfen, dass die folgenden Gleichungen gelten:

$$\begin{aligned} B_0(x) + B_1(x) + B_2(x) + B_3(x) &= 1, \\ \frac{1}{3}B_1(x) + \frac{2}{3}B_2(x) + B_3(x) &= x, \\ \frac{1}{3}B_2(x) + B_3(x) &= x^2. \end{aligned}$$

Dann können Sie (es ist $B_3(x) = x^3$) die Basistransformationsmatrix zwischen Bernstein- und Monombasis ablesen:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & 0 & 0 \\ 1 & \frac{2}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

Zwischen Hermite und Bernstein können Sie das analog durchführen, oder die Inverse der Basistransformation zwischen Hermite- und Monombasis verwenden und die beiden Matrizen multiplizieren. Man erhält die Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{4}{3} & \frac{1}{9} & 0 & 0 \\ 2 & \frac{4}{9} & \frac{1}{9} & 0 \\ 4 & 2 & \frac{4}{3} & 1 \end{pmatrix}$$

12. B und B' seien Basen eines Vektorraums, P und Q die beiden Basistransformationsmatrizen. Weisen Sie nach, dass $PQ = QP = I$ erfüllt ist!

Lösung:

$B = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ und $B' = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ seien die Basis des n -dimensionalen Vektorraums U . Dann gilt mit den Basistransformationsmatrizen P und Q und jeden Vektor $\mathbf{u} \in U$ für die Koordinatenvektoren

$$\mathbf{u}_B = P\mathbf{u}_{B'}, \mathbf{u}_{B'} = Q\mathbf{u}_B.$$

Angenommen, es gilt $QP \neq I_n$. Dann muss es mindestens einen Vektor $\mathbf{v} \in V$ geben mit $QP\mathbf{v} \neq \mathbf{v}$.

Sind \mathbf{v}_B und $\mathbf{v}_{B'}$ die entsprechenden Koordinatenvektoren dieses Vektors, dann gilt aber

$$\mathbf{v}_{B'} = Q\mathbf{v}_B = QP\mathbf{v}_{B'};$$

ein Widerspruch. Analog kann $PQ = I_n$ nachgewiesen werden.

13. Lösen Sie das lineare Gleichungssystem $\begin{pmatrix} 1 & 6 & 4 \\ 2 & 4 & -1 \\ -1 & 2 & 5 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ und bestimmen Sie eine Basis des Nullraums der Koeffizientenmatrix!

Lösung:

Die erweiterte Koeffizientenmatrix kann durch Gauß-Elimination auf die Form

$$\begin{pmatrix} 1 & 6 & 4 & 1 \\ 0 & -8 & -9 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

gebracht werden. Das System ist nicht eindeutig lösbar. Man kann $x_3 = \lambda$ setzen und erhält danach durch Rücksubstitution die allgemeine Lösung $x_1 = \frac{1}{4}(1 + 11\lambda)$ und $x_2 = \frac{1}{8}(1 - 9\lambda)$.

Für $\lambda = 0$ erhält man die spezielle Lösung

$$\mathbf{x}_s = \left(\frac{1}{4}, \frac{1}{8}, 0 \right)^T.$$

Eine Basis des Nullraums kann jetzt abgelesen werden, denn es gilt $\mathbf{x} = \mathbf{x}_s + \mathbf{x}_h$ mit Lösung des homogenen Gleichungssystems mit der gleichen Koeffizientenmatrix. Dann ist

$$NR(A) = \left[\left(\frac{11}{4}, -\frac{9}{8}, 1 \right)^T \right].$$

14. Welche der beiden Matrizen ist orthogonal: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 & \frac{1}{2}\sqrt{2} \\ 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2}\sqrt{2} \end{pmatrix}$?

Lösung:

Die Einheitsmatrix ist orthogonal.

Die zweite Matrix hat zwar Spaltenvektoren, die alle Länge 1 besitzen. Aber das Skalarprodukt zwischen zweiter und dritter Spalte ist nicht 0, sondern $\frac{1}{2}\sqrt{2}$. Also liegt kein Orthonormalsystem vor!

15. Zeigen Sie, dass für eine Matrix A mit linear unabhängigen Spaltenvektoren und einer rechten Seite \mathbf{b} , die orthogonal zum Spaltenraum von A ist, $\mathbf{x} = \mathbf{0}$ die Ausgleichslösung von $A\mathbf{x} = \mathbf{b}$ ist.

Lösung:

Für A existiert die QR -Zerlegung. Da \mathbf{b} orthogonal zum Spaltenraum ist, gilt $Q^T\mathbf{b} = \mathbf{0}$, denn die rechte Seite steht auf allen Spalten von Q senkrecht.

R ist invertierbar, also hat das lineare Gleichungssystem $R\mathbf{x} = Q^T\mathbf{b} = \mathbf{0}$ nur die triviale Lösung.

16. Berechnen Sie die Ausgleichslösung von $A\mathbf{x} = \mathbf{b}$ mit $A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ -1 & 2 \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} 7 \\ 0 \\ -7 \end{pmatrix}$.

Lösung:

Q und R können durch die Anwendung des Gram-Schmidt-Verfahrens auf die Spalten von A berechnet werden.

Man erhält

$$Q = \begin{pmatrix} \frac{1}{3}\sqrt{3} & \frac{5}{42}\sqrt{42} \\ -\frac{1}{3}\sqrt{3} & \frac{1}{42}\sqrt{42} \\ -\frac{1}{3}\sqrt{3} & \frac{2}{21}\sqrt{42} \end{pmatrix}, R = \begin{pmatrix} \sqrt{3} & -\frac{2}{3}\sqrt{3} \\ 0 & \frac{1}{3}\sqrt{42} \end{pmatrix}.$$

Es ist $Q^T\mathbf{b} = \left(\frac{14}{3}\sqrt{4}, \frac{5}{6}\sqrt{42} - \frac{2}{3}\sqrt{42} \right)^T$ und die Lösung von $R\mathbf{x} = Q^T\mathbf{b}$ ergibt sich als

$$\mathbf{x} = \left(5, \frac{1}{2} \right)^T.$$

17. Berechnen Sie die QR -Zerlegung der Matrix $A = \begin{pmatrix} 1 & 0 \\ 1 & 3 \\ 1 & 4 \\ 1 & 7 \end{pmatrix}$ mit Hilfe von Householder-Transformationen und lösen Sie damit das lineare Ausgleichsproblem mit der rechten Seite $\mathbf{b} = (1, 2, 6, 4)^T$.

Lösung:

Für den ersten Spaltenvektor $\mathbf{a}_1 = (1, 1, 1, 1)^T$ ist $\mathbf{u}_1 = (\frac{3}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T$. Dann ist

$$P_1 A = \begin{pmatrix} -2 & -7 \\ 0 & \frac{2}{3} \\ 0 & \frac{5}{3} \\ 0 & \frac{14}{3} \end{pmatrix}.$$

Im nächsten Schritt wird der Spaltenvektor $(\frac{2}{3}, \frac{5}{3}, \frac{14}{3})^T$ auf den ersten kanonischen Einheitsvektor im \mathbb{R}^3 gedreht, mit $\mathbf{u}_2 = (\frac{17}{15}, \frac{1}{3}, \frac{14}{15})^T$. Es ergibt sich dann

$$P_2 A = \begin{pmatrix} 2 & -7 \\ 0 & -5 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Im unteren Teil von A kann die Information über \mathbf{u}_1 und \mathbf{u}_2 abgelegt werden. Es fehlt zwar ein Speicherplatz, aber die beiden Vektoren können ja so abgelegt werden, dass durch eine Skalierung die erste Koordinate immer 1 ist. Dann ergibt sich nach Ablauf des Algorithmus

$$P_2 A = \begin{pmatrix} 2 & -7 \\ \frac{1}{3} & -5 \\ \frac{1}{3} & 5 \\ \frac{1}{3} & \frac{17}{17} \end{pmatrix}.$$

Auch $Q^T \mathbf{b}$ hätte man gleich berechnen können, durch $P_2 P_1 \mathbf{v} = (-\frac{13}{2}, \frac{5}{2}, \frac{99}{34}, -\frac{5}{34})^T$. Dann ergibt sich das lineare Gleichungssystem

$$\begin{pmatrix} -2 & -7 \\ 0 & -5 \end{pmatrix} \mathbf{x} = \begin{pmatrix} -\frac{13}{2} \\ \frac{5}{2} \end{pmatrix}$$

und die Lösung

$$\mathbf{x} = \begin{pmatrix} 3 \\ \frac{7}{2} \end{pmatrix}.$$

18. Implementieren Sie die QR -Zerlegung mit Householder-Transformationen in der Programmiersprache Ihrer Wahl und lösen Sie damit das Ausgleichsproblem aus den Aufgaben 16 und 17!

Lösung:

Auch bei der QR -Zerlegung kann eine Pivot-Suche eingesetzt werden, um die Stabilität zu erhöhen. Hier werden Spalten vertauscht, um sicherzustellen, dass die euklidische Norm der jeweils ersten Spalte der restlichen Matrix maximal wird. Dann bricht der Algorithmus erst dann zusammen, wenn die restliche Matrix wirklich (numerisch) die Nullmatrix ist.

Diese Spaltenpivotsuche entspricht einer Umnummerierung der Variablen, also hat man mit der Permutationsmatrix P eine Zerlegung der Form

$$QRP = A$$

berechnet.

Bei der Auflösung des linearen Ausgleichsproblems wird die QR -Zerlegung mit Spaltenpivotsuche durchgeführt. Anschließend wird auf die rechte Seite \mathbf{b} Q^T angewandt und das gestaffelte System $R\mathbf{x} = Q^T \mathbf{b}$ berechnet. Abschließend muss die Spaltenvertauschung rückgängig gemacht werden mit $P^{-1}\mathbf{x}$.

```
#!/
 * Lösung der Aufgabe 18, Kapitel 10.
 */

import com.ibm.math.array.*;    // Ninja Klassen importieren

public class qrtest {

    public static void main(String[] args) {
        int i, m = 4, n=3;
        double tol = 0.000001;

        doubleArray1D b = new doubleArray1D(m);
        doubleArray1D x = new doubleArray1D(n);
        doubleArray1D diag = new doubleArray1D(m);
        doubleArray2D A = new doubleArray2D(m,n);
        intArray1D p = new intArray1D(n);

        initArrays(m, n, A, b);

        // Die QR-Zerlegung durchführen
        QRzerlegung(A, diag, p);

        // Berechnung der Lösung des Ausgleichsproblems
        QRsol(A, diag, p, b, x);

        System.out.println("\nDie berechnete Lösung x");
        for (i=0; i<n; i++)
            System.out.println("x[" + i + "] = "+x.get(i));

        if (validate(x))
            System.out.println("\nDie Probe war erfolgreich!");
        else
            System.out.println("\nDie Probe hat nicht gepasst!");
    }

    /*!
     * Anwendung von  $Q^T$  auf einen gegebenen Vektor als Teil der
     * Auflösung eines linearen Ausgleichsproblems mit Hilfe
     * der QR-Zerlegung. Die Funktion QRzerlegung muss vorher aufgerufen
     * werden!
     */
    private static void QTb(doubleArray2D A, doubleArray1D diag,
                           doubleArray1D x, doubleArray1D y)
    {
        int i, j, m = A.size(0), n = A.size(1);
        double gamma, summe;

        // Kopieren von x auf y, x wird nicht veraendert!
        Blas.dcopy(x,y);

        // Householder-Transformationen anwenden
        for (j=0; j<n; j++) {
            summe = 0.0;
            for (i=j; i<m; i++)
                summe += A.get(i,j)*y.get(i);
            gamma = summe/(diag.get(j)*A.get(j,j));
            for (i=j; i<m; i++)
```

```

        y.set(i, y.get(i)+gamma*A.get(i,j));
    }
}

/*
 * Auflösungsfunktion für ein lineares überbestimmtes
 * Asungleichsproblem, nach Durchführung der QR-Zerlegung
 * mit der Funktion Qrzerlegung. Auf A steht das Ergebnis der
 * Zerlegung; die Diagonale von R ist auf dem Vektor diag
 * gespeichert. p enthält die durchgeführten Spaltenvertauschungen,
 * b die rechte Seite und x die berechnete Lösung.
 */
private static void QRsol(doubleArray2D A, doubleArray1D diag,
                          intArray1D p, doubleArray1D b,
                          doubleArray1D x)
{
    int i, j, m = A.size(0), n = A.size(1);
    double summe;
    doubleArray1D work = new doubleArray1D(m);

    // Q^T auf die rechte Seite anwenden mit der Funktion QTb
    QTb(A, diag, b, work);

    // Auflösen von Rx = Q^T b
    work.set(n, work.get(n)/diag.get(n));
    for (i=n-1; i>=0; i--) {
        summe = 0.0;
        for (j=i+1; j<n; j++)
            summe += A.get(i,j)*work.get(j);
        work.set(i, (work.get(i)-summe)/diag.get(i));
    }

    // Pivotsuche rückgängig machen durch Rücknummerierung
    // der berechneten Lösung
    for (i=0; i<n; i++)
        x.set(p.get(i), work.get(i));
}

/*
 * QR-Zerlegung der Matrix A mit Householder-Transformationen.
 * Zur besseren Stabilität ist die Pivotwahl nach Businger-
 * Golub (vgl. Wilkinson/Reinsch: Linear Algebra) eingebaut.
 *
 * Die Matrix A enthaelt nach Durchlauf der Routine oberhalb
 * der Diagonale die Matrix R mit A=QR. Die Diagonale von R
 * steht im Vektor diag. Dadurch kann auf der Diagonale von A
 * und darunter die Information ueber die orthogonale Matrix Q
 * abgelegt werden. Der Vektor p enthaelt Information über
 * die Permutation der Zeilen.
 */
private static void QRzerlegung(doubleArray2D A, doubleArray1D diag,
                                intArray1D p)
{
    int i, j, k, tausch, m = A.size(0), n = A.size(1);
    double q, s, h, sigma, summe, qrkk, alphak, beta;
    doubleArray1D work = new doubleArray1D(n);

    // Den Permutationsvektor initialisieren und die Spaltensummen

```

```

// für die Pivotsuche auf work abspeichern.
for (j=0; j<n; j++) {
    summe = 0.0;
    for (i=0; i<m; i++)
        summe += A.get(i,j)*A.get(i,j);
    work.set(j, summe);

    p.set(j,j);
}

// Householder-Transformationen
for (k=0; k<n; k++) {
    sigma = work.get(k);
    tausch = k;

    // Pivotwahl nach Businger-Golub
    for (j=k+1; j<n; j++) {
        if (sigma <= work.get(j)) {
            sigma = work.get(j);
            tausch = j;
        }
    }

    // Spaltentausch durchführen
    if (tausch != k) {
        i = p.get(k);
        p.set(k, p.get(tausch));
        p.set(tausch, i);
        work.set(tausch,work.get(k));
        work.set(k, sigma);
        for (i=0; i<m; i++) {
            sigma = A.get(i,k);
            A.set(i,k,A.get(i,tausch));
            A.set(i,tausch, sigma);
        }
    }

    // Q berechnen
    // Abbruch für work(k) < eps, nicht realisiert!
    qrkk = A.get(k,k);
    if (qrkk < 0.0)
        alphak = Math.sqrt(work.get(k));
    else
        alphak = -Math.sqrt(work.get(k));

    diag.set(k, alphak);
    sigma = work.get(k) - qrkk*alphak;
    beta = 1.0/sigma;
    A.set(k,k,qrkk-alphak);

    // Orthogonalprojektion auf den Rest der Matrix anwenden
    for (j=k+1; j<n; j++) {
        sigma = 0.0;
        for (i=k; i<m; i++)
            sigma += A.get(i,k)*A.get(i,j);
        sigma *= beta;
        for (i=k; i<m; i++)
            A.set(i,j,A.get(i,j)-sigma*A.get(i,k));
    }
}

```

```

        work.set(j, work.get(j) - A.get(k, j) * A.get(k, j));
    }
}

private static void initArrays(int m, int n, doubleArray2D A,
                               doubleArray1D b) {
    int i, j;

    b.set(0, 17.0);
    b.set(1, 57.0);
    b.set(2, 121.0);
    b.set(3, 209.0);

    for (i=0; i<m; i++) A.set(i, 0, 1.0);
    A.set(0, 1, 2.0);
    A.set(1, 1, 4.0);
    A.set(2, 1, 6.0);
    A.set(3, 1, 8.0);

    A.set(0, 2, 4.0);
    A.set(1, 2, 16.0);
    A.set(2, 2, 36.0);
    A.set(3, 2, 64.0);
}

private static boolean validate(doubleArray1D x) {
    // Probe der berechneten Lösung
    boolean ok = true;
    double eps = 1e-09, error;
    int n = x.size();

    error = Math.abs(x.get(0) - 1.0);
    if (error > eps) {
        ok = false;
    }
    error = Math.abs(x.get(1) - 2.0);
    if (error > eps) {
        ok = false;
    }
    error = Math.abs(x.get(2) - 3.0);
    if (error > eps) {
        ok = false;
    }
    return ok;
}
}

```

Die Funktion `initArrays` besetzt die rechte Seite und die 4×3 -Matrix A als

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 4 & 16 \\ 1 & 6 & 36 \\ 1 & 8 & 64 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 17 \\ 57 \\ 121 \\ 209 \end{pmatrix}.$$

Die lineare Ausgleichslösung dafür ist der Vektor $\mathbf{x} = (1, 2, 3)^T$. Die Ausgabe des *Java*-Programms ist:

Die berechnete Lösung \mathbf{x}

```
x[0] = 1.00000000000000282
x[1] = 1.9999999999999976
x[2] = 3.0000000000000004
```

Die Lösungen für die Aufgaben 16 und 17 werden mit dieser Implementierung auch reproduziert; dazu kann man die Funktion `initArrays` und `validate` entsprechend verändern:

```
private static void initArrays16(doubleArray2D A, doubleArray1D b) {
    int i, j, m = A.size(0), n = A.size(1);

    b.set(0, 7.0);
    b.set(1, 0.0);
    b.set(2, -7.0);

    A.set(0,0, 1.0); A.set(1,0, -1.0); A.set(2,0, -1.0);
    A.set(0,1, 1.0); A.set(1,1,1.0); A.set(2,1,2.0);
}

private static void initArrays17(doubleArray2D A, doubleArray1D b) {
    int i, j, m = A.size(0), n = A.size(1);
    double h1 = Math.sqrt(3.0), h2 = Math.sqrt(42.0);

    b.set(0, 1.0);
    b.set(1, 2.0);
    b.set(2, 6.0);
    b.set(3, 4.0);

    for (i=0; i<m; i++) A.set(i,0,1.0);
    A.set(0,1,0.0); A.set(1,1,3.0); A.set(2,1,4.0); A.set(3,1,7.0);
}

private static boolean validate16(doubleArray1D x) {
    // Probe der berechneten Lösung
    boolean ok = true;
    double eps = 1e-09, error;

    error = Math.abs(x.get(0)-5.0);
    if (error > eps) {
        ok = false;
    }
    error = Math.abs(x.get(1)-0.5);
    if (error > eps) {
        ok = false;
    }
    return ok;
}

private static boolean validate17(doubleArray1D x) {
    // Probe der berechneten Lösung
    boolean ok = true;
    double eps = 1e-09, error;

    error = Math.abs(x.get(0)-1.5);
    if (error > eps) {
        ok = false;
    }
    error = Math.abs(x.get(1)-0.5);
    if (error > eps) {
        ok = false;
    }
}
```

```

}
return ok;
}

```

19. Berechnen Sie die Ausgleichsgerade zur Tabelle 10.1 mit Ihrer Implementierung aus Aufgabe 18!

Tabelle 10.1: Die Messwerte zu Aufgabe 19

150	150	150	200	200	200	250	250	250	300	300	300
77,4	76,7	78,2	84,1	84,5	83,7	88,9	89,2	89,7	94,2	94,7	95,9

Lösung:

Die Lösung aus Aufgabe 18 wurde leicht verändert. Denkbar wäre auch, aus dem Quelltext eine Package zu machen und die Messdaten beispielsweise einzulesen. Mit den gegebenen Messwerten wird die Matrix in `initArrays` besetzt. Bei der Berechnung der Ausgleichsgerade benötigt man eine $m \times 2$ Matrix und eine rechte Seite $\mathbf{b} \in \mathbb{R}^m$, wenn m die Anzahl der Messwerte ist. Die erste Spalte der Matrix A ist 1.0, die zweite Spalte besteht aus den gemessenen x -Werten. Die y -Werte bilden die rechte Seite \mathbf{b} . Wendet man die QR -Zerlegung aus Aufgabe 18 auf dieses Ausgleichsproblem an, dann erhält man

```

Die berechnete Lösung x
x[0] = 60.483333333333308
x[1] = 0.11533333333333454

```

Die erste Komponente der Lösung ist der Nulldurchgang, die zweite die Steigung der Ausgleichsgerade. Also lautet die Ausgleichsgerade $f(x) = 0,115\,333x + 60,4833$.

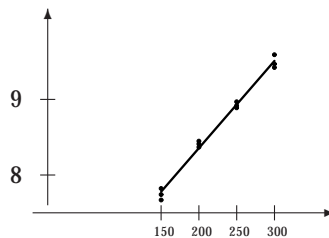


Abbildung 10.1: Die Messwerte und die berechnete Ausgleichsgerade für Aufgabe 19

Hier der Quelltext:

```

import com.ibm.math.array.*; // Ninja Klassen importieren

public class regressgerade {

    public static void main(String[] args) {
        int i, m=12, n=2;

        doubleArray1D xwerte = getXValues(m);
        doubleArray1D ywerte = getYValues(m);

        doubleArray1D x = regress(xwerte, ywerte);

        System.out.println("\nDie berechnete Lösung x");
        for (i=0; i<n; i++)
            System.out.println("x[" + i + "] = "+x.get(i));
    }
}

```

```

}

/*!
* Zusammenfassen der Besetzung der Felder und der Lösung des
* linearen Ausgleichproblems
*/
private static doubleArray1D regress(doubleArray1D xwerte,
                                     doubleArray1D ywerte) {
    int m = xwerte.size(), n=2;

    doubleArray2D A = new doubleArray2D(m,n);
    doubleArray1D b = new doubleArray1D(m);

    initArrays(xwerte, ywerte, A, b);

    return QR(A,b);
}

/*!
* Zusammenfassung der QR-Zerlegung und der Auflösung zu einem
* Funktionsaufruf.
*/
private static doubleArray1D QR(doubleArray2D A, doubleArray1D b) {
    int m = A.size(0), n = A.size(1);
    doubleArray1D x = new doubleArray1D(n);
    doubleArray1D diag = new doubleArray1D(m);
    intArray1D p = new intArray1D(n);

    // Die QR-Zerlegung durchführen
    QRzerlegung(A, diag, p);

    // Berechnung der Lösung des Ausgleichsproblems
    QRsol(A, diag, p, b, x);

    return x;
}

/*!
* Anwendung von  $Q^T$  auf einen gegebenen Vektor als Teil der
* Auflösung eines linearen Ausgleichsproblems mit Hilfe
* der QR-Zerlegung. Die Funktion QRzerlegung muss vorher aufgerufen
* werden!
*/
private static void QTb(doubleArray2D A, doubleArray1D diag,
                       doubleArray1D x, doubleArray1D y)
{
    int i, j, m = A.size(0), n = A.size(1);
    double gamma, summe;

    // Kopieren von x auf y, x wird nicht veraendert!
    Blas.dcopy(x,y);

    // Householder-Transformationen anwenden
    for (j=0; j<n; j++) {
        summe = 0.0;
        for (i=j; i<m; i++)
            summe += A.get(i,j)*y.get(i);
        gamma = summe/(diag.get(j)*A.get(j,j));
        for (i=j; i<m; i++)

```

```

        y.set(i, y.get(i)+gamma*A.get(i,j));
    }
}

/*!
 * Auflösungsfunktion für ein lineares überbestimmtes
 * Asungleichsproblem, nach Durchführung der QR-Zerlegung
 * mit der Funktion Qrzerlegung. Auf A steht das Ergebnis der
 * Zerlegung; die Diagonale von R ist auf dem Vektor diag
 * gespeichert. p enthält die durchgeführten Spaltenvertauschungen,
 * b die rechte Seite und x die berechnete Lösung.
 */
private static void QRsol(doubleArray2D A, doubleArray1D diag,
                        intArray1D p, doubleArray1D b,
                        doubleArray1D x)
{
    int i, j, m = A.size(0), n = A.size(1);
    double summe;
    doubleArray1D work = new doubleArray1D(m);

    //  $Q^T$  auf die rechte Seite anwenden mit der Funktion QTb
    QTb(A, diag, b, work);

    // Auflösen von  $Rx = Q^T b$ 
    work.set(n, work.get(n)/diag.get(n));
    for (i=n-1; i>=0; i--) {
        summe = 0.0;
        for (j=i+1; j<n; j++)
            summe += A.get(i,j)*work.get(j);
        work.set(i, (work.get(i)-summe)/diag.get(i));
    }

    // Pivotsuche rückgängig machen durch Rücknummerierung
    // der berechneten Lösung
    for (i=0; i<n; i++)
        x.set(p.get(i), work.get(i));
}

/*!
 * QR-Zerlegung der Matrix A mit Householder-Transformationen.
 * Zur besseren Stabilität ist die Pivotwahl nach Businger-
 * Golub (vgl. Wilkinson/Reinsch: Linear Algebra) eingebaut.
 *
 * Die Matrix A enthaelt nach Durchlauf der Routine oberhalb
 * der Diagonale die Matrix R mit  $A=QR$ . Die Diagonale von R
 * steht im Vektor diag. Dadurch kann auf der Diagonale von A
 * und darunter die Information ueber die orthogonale Matrix Q
 * abgelegt werden. Der Vektor p enthaelt Information über
 * die Permutation der Zeilen.
 */
private static void QRzerlegung(doubleArray2D A, doubleArray1D diag,
                                intArray1D p)
{
    int i, j, k, tausch, m = A.size(0), n = A.size(1);
    double q, s, h, sigma, summe, qrkk, alphak, beta;
    doubleArray1D work = new doubleArray1D(n);

    // Den Permutationsvektor initialisieren und die Spaltensummen

```

```

// für die Pivotsuche auf work abspeichern.

for (j=0; j<n; j++) {

    // Aus work stehen die euklidischen Normen der Spaltenvektoren
    summe = 0.0;
    for (i=0; i<m; i++)
        summe += A.get(i,j)*A.get(i,j);
    work.set(j, summe);

    p.set(j,j);
}

// Householder-Transformationen
for (k=0; k<n; k++) {
    sigma = work.get(k);
    tausch = k;

    // Pivotwahl nach Businger-Golub
    for (j=k+1; j<n; j++) {
        if (sigma <= work.get(j)) {
            sigma = work.get(j);
            tausch = j;
        }
    }

    // Spaltentausch durchführen
    if (tausch != k) {
        i = p.get(k);
        p.set(k, p.get(tausch));
        p.set(tausch, i);
        work.set(tausch,work.get(k));
        work.set(k, sigma);
        for (i=0; i<m; i++) {
            sigma = A.get(i,k);
            A.set(i,k,A.get(i,tausch));
            A.set(i,tausch, sigma);
        }
    }

    // Q berechnen
    // Abbruch für work(k) < eps, nicht realisiert!
    qrkk = A.get(k,k);
    if (qrkk < 0.0)
        alphak = Math.sqrt(work.get(k));
    else
        alphak = -Math.sqrt(work.get(k));

    diag.set(k, alphak);
    sigma = work.get(k) - qrkk*alphak;
    beta = 1.0/sigma;
    A.set(k,k,qrkk-alphak);

    // Orthogonalprojektion auf den Rest der Matrix anwenden
    for (j=k+1; j<n; j++) {
        sigma = 0.0;
        for (i=k; i<m; i++)
            sigma += A.get(i,k)*A.get(i,j);
    }
}

```

```
        sigma *= beta;
        for (i=k; i<m; i++)
            A.set(i,j,A.get(i,j)-sigma*A.get(i,k));
        work.set(j, work.get(j)-A.get(k,j)*A.get(k,j));
    }
}

private static void initArrays(doubleArray1D xwerte, doubleArray1D ywerte,
                              doubleArray2D A, doubleArray1D b) {

    // Besetzung der Matrix und der rechten Seite für die übergebenen
    // Messwerte (x_i, y_i)
    int i, j, m = A.size(0), n = A.size(1);
    Range spalten = new Range(0, m-1);

    b.assign(ywerte);

    A.section(spalten, 0).assign(1.0);
    A.section(spalten, 1).assign(xwerte);
}

private static doubleArray1D getXValues(int m) {
    doubleArray1D x = new doubleArray1D(m);

    x.set(0,150.0);
    x.set(1,150.0);
    x.set(2, 150.0);

    x.set(3, 200.0);
    x.set(4, 200.0);
    x.set(5, 200.0);

    x.set(6, 250.0);
    x.set(7, 250.0);
    x.set(8, 250.0);

    x.set(9, 300.0);
    x.set(10, 300.0);
    x.set(11, 300.0);

    return x;
}

private static doubleArray1D getYValues(int m) {
    doubleArray1D y = new doubleArray1D(m);

    y.set(0,77.4);
    y.set(1,76.7);
    y.set(2, 78.2);

    y.set(3, 84.1);
    y.set(4, 84.5);
    y.set(5, 83.7);

    y.set(6, 88.9);
    y.set(7, 89.2);
    y.set(8, 89.7);
}
```

```

y.set(9, 94.2);
y.set(10, 94.7);
y.set(11, 95.9);

return y;
}
}

```

20. Für exakte Eingangsgrößen x_i wurden die Ausgangsgrößen y_i wie in Tabelle 10.2 beobachtet. Als Ansatz für den Ausgleich wird $f(x) = a_1 \frac{x}{1+x} + a_2 (1 - e^{-x})$ verwendet. Stellen Sie die Matrix A und die rechte Seite \mathbf{b} auf und lösen Sie das lineare Ausgleichsproblem durch eine QR -Zerlegung!

Tabelle 10.2: Die Messwerte zu Aufgabe 20

x_i	0,2	0,5	1,0	1,5	2,0	3,0
y_i	0,3	0,5	0,8	1,0	1,2	1,3

Lösung:

Der Unterschied zur Berechnung der Ausgleichsgerade besteht in der Besetzung der Matrix A für das Ausgleichsproblem. Die rechte Seite \mathbf{b} enthält wieder die gemessenen y -Werte. Würden alle Messwerte auf der angegebenen Funktion mit den Koeffizienten a_1 und a_2 liegen, dann gilt für den k -ten Messwert

$$a_1 \frac{x_k}{1+x_k} + a_2 (1 - e^{-x_k}) = y_k.$$

Also ist der Vektor $\mathbf{x} = (a_1, a_2)^T$, die erste Spalte der Matrix A besteht aus den Werten $\frac{x_k}{1+x_k}$, $1 \leq k \leq m$, die zweite Spalte aus $1 - e^{-x_k}$, $1 \leq k \leq m$. Wendet man die QR -Zerlegung aus Aufgabe 18 auf dieses Ausgleichsproblem an, dann erhält man

```

Die berechnete Lösung x
x[0] = 0.38244529084477336
x[1] = 1.0391925936468365

```

$x[0]$ entspricht a_1 , die zweite Komponente a_2 . Die berechnete Ausgleichslösung ist die Funktion $f(x) = 0,382445 \frac{x}{1+x} + 1,03919(1 - e^{-x})$. In Abbildung 10.2 ist sie mit den Messwerten dargestellt.

Der Quelltext unterscheidet sich von der Lösung zu Aufgabe 19 nur durch die Besetzung der Felder, und dass die beiden Funktionen implementiert wurden. Wenn Sie darin für `basis1` die konstante Funktion $f(x) = 1$ und für `basis2` die Funktion $f(x) = x$ setzen, dann erhalten Sie die Lösung aus Aufgabe 19 zurück. Hier die Implementierung:

```

private static void initArrays(doubleArray1D xwerte, doubleArray1D ywerte,
                               doubleArray2D A, doubleArray1D b) {

    // Besetzung der Matrix und der rechten Seite für die übergebenen
    // Messwerte (x_i, y_i)
    int i, m = A.size(0);
    Range spalten = new Range(0, m-1);

    b.assign(ywerte);

    for (i=0; i<m; i++) {
        A.set(i,0,basis1(xwerte.get(i)));
    }
}

```

```
        A.set(i,1,basis2(xwerte.get(i)));
    }
}

private static double basis1(double x) {
    return x/(1.0+x);
}

private static double basis2(double x) {
    return 1.0-Math.exp(-x);
}
```

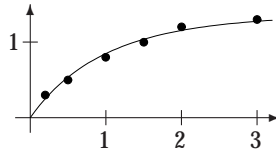


Abbildung 10.2: Die Messwerte und die berechnete Ausgleichsfunktion für Aufgabe 20