Die Q-R-Faktorisierung mit Hilfe der Householder-Transformation ist bei *wxMaxima* in dem package "lapack" als Funktion "dgeqrf" implementiert, welches die Q- bzw. die R-Matrix in einer Liste zurückgibt. Sie ist numerisch etwas stabiler als das Gram-Schmidt Verfahren oder Givens-Rotationen (2 andere Möglichkeiten).

Zur Erinnerung: die Q-Matrix ist orthogonal also $Q^t = Q^{-1}$ und R besteht aus einer oberen Dreiecksmatrix und einer Nullmatrix, d.h. alle Elemente unterhalb der Hauptdiagonale verschwinden:

$$R = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{nn} \\ 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \qquad \text{wobei } \hat{R} \text{ eine obere Dreiecksmatrix ist}$$

Nocheinmal die Ausgangssituation kurz zusammengefasst:
Gegeben ist ein (womöglich) überbestimmtes Gleichungssystem

$$A \cdot x = b \qquad \text{mit} \quad A \in \mathbb{R}^{m \times n} \wedge m \geq n,\, x \in \mathbb{R}^n,\, b \in \mathbb{R}^m,\, Rang(A) = n$$

Im allgemeinen wird es zu keiner eindeutigen Lösung x führen (überbestimmt), aber wir wären schon mit der Lösung des linearen Ausgleichsproblems zufrieden, nämlich wir finden ein x mit der Eigenschaft

$$|A\,x - b|^2 \to min$$

Weiter unten (nach Theorem 2) werden wir zeigen, dass die Q-R-Zerlegung genau so ein x liefert.

Gelingt nun eine Q-R-Zerlegung von $A = Q \cdot R$ mit $Q \in \mathbb{R}^{m \times m}$ und $R \in \mathbb{R}^{m \times n}$ so ergibt sich:

$$Q \cdot R\,x = b \,|\, Q^t \cdot \Rightarrow I \cdot R\,x = Q^t \cdot b \Rightarrow R\,x = Q^t \cdot b$$

Die letzte Gleichung ist aber leicht durch Rücksubstituierung zu lösen, also x_n aus letzter Zeile, dann x_{n-1} aus vorletzter Zeile usw. bis schließlich x_1 aus 1. Zeile.

Übrigens ist die Q-R-Zerlegung insofern eindeutig, dass sie immer diegleiche Dreiecksmatrix $\hat{R}$ liefert, also

$$A = Q \cdot R = (Q_1, Q_2) \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} = Q_1 \hat{R}$$

d.h. Q_1 ist eindeutig Q_2 i.a. nicht. Für diese Eindeutigkeit brauchen wir nur die positive Definitheit der Diagonalelemente von $\hat{R}$.

$$A = Q_1\,R_1 = Q_2\,R_2 \Rightarrow \boxed{R_1^t\,R_1} = R_1^t\left(Q_1^t\,Q_1\right)R_1 = A_1^t\,A_1 = R_2^t\left(Q_2^t\,Q_2\right)R_2 = \boxed{R_2^t\,R_2}$$

aus

$$R_1^t\,R_1 = R_2^t\,R_2 \Rightarrow \left(R_2^{-1}\right)^t\,R_1^t = R_2\,R_1^{-1}$$

auf der linken Gleichungsseite stehen untere Diagonalmatrizen, auf der rechten Seite obere Diagonalmatrizen - also müssen sie Diagonalmatrizen sein. Seien α_i die Diagonalelemente von R_1 und β_i die Diagonalelemente von R_2, dann ergibt obige Matrixmultiplikation

$$\forall i \in \{1, 2, \dots n\} \quad \frac{1}{\beta_i}\alpha_i = \beta_i \frac{1}{\alpha_i} \overset{\alpha_i, \beta_i \geq 0}{\underset{\Rightarrow}{\uparrow}} \alpha_i = \beta_i$$

Für die letzte Behauptung haben wir herangezogen, dass die Inverse einer Dreiecksmatrix wieder eine Dreiecksmatrix vom gleichen Typ ist (Untergruppe) und beim Invertieren sich der Reziprokwert in der Hauptdiagonale ergibt, dies kann man leicht einsehen, wenn man sich die Inverse als Unbekannte darstellt:

$$A\,X = I \Rightarrow x_{ii} = \frac{1}{a_{ii}}$$

Wir werden sehen, dass wir für eine erfolgreiche Zerlegung die Ausgangsmatrix A noch etwas verändern müssen (indem wir einige Zeilen vertauschen) - dies werden wir mit einer Permutationsmatrix P besorgen, sodass das Problem sich dann so stellt:

$$\underbrace{P \cdot A}_{A'}\ \underbrace{P \cdot x}_{x'} = \underbrace{P \cdot b}_{b'} \Rightarrow R\,x' = Q^t\,b'$$

aus dem Lösungsvektor x' lässt sich aber wieder leicht ($P \cdot P^t = I$) x berechnen! Jetzt bleibt nur mehr zu zeigen, dass diese Lösung x ganz gut durch das Ausgangsgleichungssystem "laviert" - das ja im strengen Sinn widersprüchlich ist, weil die Messwerte b ja mit Fehlern befaftet sind! Dazu benutzen wir folgendes

Theorem 1. *Ist Q orthogonal, $Q \in \mathbb{R}^{n\times n}$ und $u,\, v \in \mathbb{R}^n \Rightarrow (Q\,u)\cdot(Q\,v) = u \cdot v$*

Beweis:

$$(Q\,u)\cdot(Q\,v) = (Q\,u)^t(Q\,v) = u^t\,\underbrace{Q^tQ}_{I}\,v = u\cdot v$$

□

Theorem 2. *Ist Q orthogonal, $Q \in \mathbb{R}^{n\times n}$ und $u \in \mathbb{R}^n \Rightarrow |(Q\,u)| = |u|$*
wobei $|\,.\,|$ die euklidische Norm ist.

Beweis: $v := u$ in Theorem 1 □

Mit dieser Eigenschaft lässt sich jetzt das lineare Ausgleichsproblem umformen:

$$|A\,x - b|^2 = |Q\,R\,x - b|^2 \overset{\substack{Q^t \text{ ist orthogonal}\\ \uparrow}}{=} |R\,x - Q^t\,b|^2 = \left|\begin{pmatrix} \hat{R}\,x \\ 0 \end{pmatrix} - \begin{pmatrix} c \\ d \end{pmatrix}\right|^2 = |\hat{R}\,x - c|^2 + |d|^2$$

Diese Summe wird offensichtlich ein Minimum, wenn wir das Dreieckssystem $\hat{R}\,x - c = 0$ lösen und der Wert des Minimums sind die Quadrate von $(Q^t\,b)_j \quad j \in \{n+1, n+2, \ldots m\}$. Mit variieren von x lässt sich das Minimum nicht weiter verringern, weil d nicht von x abhängt.

In diesem Anhang versuchen wir die "blackbox" *dgeqrf* in eine "whitebox" zu verwandeln. Dazu werden wir uns etwas theoretisches Rüstzeug zulegen und das ganze Verfahren dann in *wxMaxima* implementieren.

Definition. *Sei $\vec{w} \in \mathbb{R}^n$ ein Einheitsspaltenvektor, d. h. es gilt $\vec{w}^{\,t}\cdot\vec{w} = 1$.*
Eine $n \times n$ Matrix P mit der Eigenschaft

$$P = I - 2\,\vec{w}\,\vec{w}^{\,t} \qquad p_{ij} = \delta_{ij} - 2\,w_i\,w_j$$

heißt Housholder Matrix.

Beachte: Während $\vec{w}^{\,t}\cdot\vec{w}$ das innere Produkt darstellt (Zahl), ist $\vec{w}\,\vec{w}^{\,t}$ das äußere Produkt ($n\times n$ Matrix).

Theorem 3. *Ist P eine Householder-Matrix $\Rightarrow$ P ist symmetrisch und orthogonal ($P = P^t = P^{-1}$)*

Beweis:

$$P^t = \left(I - 2\,\vec{w}\,\vec{w}^{\,t}\right)^t = I^t - \left(2\,\vec{w}\,\vec{w}^{\,t}\right)^t \overset{(A\cdot B)^t = B^t\cdot A^t}{\uparrow} = I - 2\,\left(\vec{w}^{\,t}\right)^t\,\vec{w}^{\,t} = P \Rightarrow P \text{ ist symmetrisch}$$

$$P\cdot P^t \overset{P^t = P}{\uparrow} = \left(I - 2\,\vec{w}\,\vec{w}^{\,t}\right)\left(I - 2\,\vec{w}\,\vec{w}^{\,t}\right) = I - 4\,\vec{w}\,\vec{w}^{\,t} + 4\,\vec{w}\,\vec{w}^{\,t}\,\vec{w}\,\vec{w}^{\,t}$$
$$= I - 4\,\vec{w}\,\vec{w}^{\,t} + 4\,\vec{w}\,\underbrace{\left(\vec{w}^{\,t}\,\vec{w}\right)}_{=1}\vec{w}^{\,t} = I \Rightarrow P \text{ ist orthogonal}$$

in Tensorschreibweise ist die Symmetrie eine Folge der Kommutativität in $\mathbb{R}$ und der Symmetrie von δ_{ij}:

$$p_{ij} = \delta_{ij} - 2\,w_i\,w_j = p_{ji}$$

auch die Orthogonalität ist relativ einfach zu beweisen:

$$p_{ik}\,p_{jk} = (\delta_{ik} - 2\,w_i\,w_k)(\delta_{jk} - 2\,w_j\,w_k) = \delta_{ik}\delta_{jk} - 2\delta_{jk}w_i\,w_k - 2\delta_{ik}w_j\,w_k + 4w_iw_j\,\underbrace{w_kw_k}_{1} = \delta_{ij}$$

□

Theorem 4. *Sind A und B orthogonal $\Rightarrow$ $A\cdot B$ ist orthogonal*

Beweis: Wird dem Leser überlassen □

Theorem 5. *$\vec{w}$ ist ein Eigenvektor der Householder-Matrix $P = I - 2\,\vec{w}\,\vec{w}^{\,t}$*

Beweis:

$$P\vec{w} = \left(I - 2\,\vec{w}\,\vec{w}^{\,t}\right)\vec{w} = \vec{w} - 2\,\vec{w}\,\underbrace{\left(\vec{w}^{\,t}\vec{w}\right)}_{=1} = -\vec{w}$$

in Tensorschreibweise:

$$p_{ij}w_j = (\delta_{ij} - 2\,w_i\,w_j)w_j = \delta_{ij}w_j - 2\,w_i\,\underbrace{w_jw_j}_{1} = -w_i$$

□

Theorem 6. *Jeder zu $\vec{w}$ orthogonale Vektor $\vec{x}$ ist ein Fixpunkt (Eigenvektor mit Eigenwert 1) der Householder-Matrix $P = I - 2\,\vec{w}\,\vec{w}^{\,t}$*

Beweis:

$$P\vec{x} = \left(I - 2\,\vec{w}\,\vec{w}^{\,t}\right)\vec{x} = \vec{x} - 2\,\vec{w}\,\underbrace{\left(\vec{w}^{\,t}\vec{x}\right)}_{=0} = \vec{x}$$

in Tensorschreibweise:

$$p_{ij}x_j = (\delta_{ij} - 2\,w_i\,w_j)x_j = \delta_{ij}x_j - 2\,w_i\,\underbrace{w_jx_j}_{0} = x_i$$

□

Im Folgenden sei $w := \vec{w}$; wir lassen die Vektorpfeile weg, weil es nicht zu Verwechslungen führen kann.

Theorem 7. *Sei* $x, y \in \mathbb{R}^n$ *und* $|x| = |y|$ *und* $w = \dfrac{x-y}{|x-y|}$ *der Householder-Matrix* $P = I - 2\,w\,w^t \Rightarrow$

$\boxed{P\,x = y}$

Beweis:

$$Px = y \Leftrightarrow x - 2\frac{x-y}{|x-y|}\frac{(x-y)^t}{|x-y|}x = y \Leftrightarrow$$

$$\Leftrightarrow (x-y) - 2\frac{(x-y)(x-y)^t\,x}{|x-y|^2} = 0 \Leftrightarrow$$

$$\Leftrightarrow \frac{(x-y)}{|x-y|^2}\left(|x-y|^2 - 2(x-y)^t\,x\right) = 0 \Leftrightarrow$$

$$\Leftrightarrow \frac{(x-y)}{|x-y|^2}\left((x-y)^t(x-y) - 2(x-y)^t\,x\right) = 0 \Leftrightarrow$$

$$\Leftrightarrow \frac{(x-y)}{|x-y|^2}\left(\cancel{x^tx} - x^ty - y^tx + \cancel{y^ty} - \cancel{2x^tx} + 2y^tx\right) = 0$$

$$\Leftrightarrow \frac{(x-y)}{|x-y|^2}\left(\cancel{-x^ty} - \cancel{y^tx} + \cancel{2y^tx}\right) = 0$$

Die letzte Zeile gilt wegen der Kommutativität des skalaren Produkts $\vec{x}\cdot\vec{y} = \vec{y}\cdot\vec{x}$, die vorletzte wegen $|x| = |y|$. Wenn die letzte Zeile also wahr ist, können wir zurückschließen auf die 1. Zeile - was die Behauptung ist!

In Tensorschreibweise:

$$\left(\delta_{ij} - 2\,\frac{(x_i - y_i)\,(x_j - y_j)}{x_kx_k - 2x_ky_k + y_ky_k}\right)x_j = y_i \Rightarrow \text{ mit } x_kx_k = y_ky_k$$

$$\cancel{2(x_i - y_i)}\,(x_kx_k - x_ky_k) = \cancel{2(x_i - y_i)}\,(x_j - y_j)\,x_j$$

Die letzte Gleichheit folgt wieder aus der Kommutativität des skalaren Produkts. □

Wir haben nun ein Verfahren, um mit einer Householder-Matrix (symmetrisch und orthogonal) alle Komponenten bis auf eine zum Verschwinden zu bringen:

$$x = (1,1,3,3,4)^t \overset{P=P^t=P^{-1}}{\longrightarrow} y = (y_1,0,0,0,0)^t$$

Nachdem die Normen der beiden Vektoren übereinstimmen müssen, muss gelten: $y_1 = 6$.

Führen wir das in wxMaxima durch:

```
(%i1)  x:[1,1,3,3,4]$
```

```
(%i2)  y:[6,0,0,0,0]$
```

```
(%i3)  norm2(x):=sqrt(x . x)$
```

```
(%i4)  w1:(x-y)/norm2(x-y);
```

$$(\%o4)\quad [-\frac{5}{2\sqrt{15}},\frac{1}{2\sqrt{15}},\frac{3}{2\sqrt{15}},\frac{3}{2\sqrt{15}},\frac{2}{\sqrt{15}}]$$

```
(%i5)  vec2Matrix(x):=block([l:length(x), m],
         m:zeromatrix(l,1),
         for i thru l do m[i,1]:x[i],
         m
       )$
```

```
(%i6)  w:vec2Matrix(w1);
```

$$(\%o6)\quad \begin{pmatrix} -\frac{5}{2\sqrt{15}} \\ \frac{1}{2\sqrt{15}} \\ \frac{3}{2\sqrt{15}} \\ \frac{3}{2\sqrt{15}} \\ \frac{2}{\sqrt{15}} \end{pmatrix}$$

```
(%i7)  w_t:transpose(w);
```

$$(\%o7)\quad \begin{pmatrix} -\frac{5}{2\sqrt{15}} & \frac{1}{2\sqrt{15}} & \frac{3}{2\sqrt{15}} & \frac{3}{2\sqrt{15}} & \frac{2}{\sqrt{15}} \end{pmatrix}$$

```
(%i8)  outerPr:w . w_t;
```

$$(\%o8)\quad \begin{pmatrix} \frac{5}{12} & -\frac{1}{12} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{3} \\ -\frac{1}{12} & \frac{1}{60} & \frac{1}{20} & \frac{1}{20} & \frac{1}{15} \\ -\frac{1}{4} & \frac{1}{20} & \frac{3}{20} & \frac{3}{20} & \frac{1}{5} \\ -\frac{1}{4} & \frac{1}{20} & \frac{3}{20} & \frac{3}{20} & \frac{1}{5} \\ -\frac{1}{3} & \frac{1}{15} & \frac{1}{5} & \frac{1}{5} & \frac{4}{15} \end{pmatrix}$$

```
(%i9)  I:diagmatrix(5,1)$
```

```
(%i10) P:I-2*outerPr;
```

(%o10)
$$\begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{2} & \frac{1}{2} & \frac{2}{3} \\ \frac{1}{6} & \frac{29}{30} & -\frac{1}{10} & -\frac{1}{10} & -\frac{2}{15} \\ \frac{1}{2} & -\frac{1}{10} & \frac{7}{10} & -\frac{3}{10} & -\frac{2}{5} \\ \frac{1}{2} & -\frac{1}{10} & -\frac{3}{10} & \frac{7}{10} & -\frac{2}{5} \\ \frac{2}{3} & -\frac{2}{15} & -\frac{2}{5} & -\frac{2}{5} & \frac{7}{15} \end{pmatrix}$$

```
(%i11) transpose(P . vec2Matrix(x));
```

(%o11) $\begin{pmatrix} 6 & 0 & 0 & 0 & 0 \end{pmatrix}$

Den letzten Vektor haben wir transponiert, um Platz zu sparen. Hat doch super geklappt - nun weiter.

Da es bei Matrizenumformungen meist darum geht, Spaltenvektoren zu erzeugen, die bis auf die 1. Komponente verschwinden, wird in der mathematischen Literatur obiges Theorem meist gleich mit

$$y = -sgn(x_1)|x|e_1 \qquad \text{wobei } e_1 := (1, 0, 0, \ldots, 0)^t$$

angegeben. Allerdings muss gelten $x_1 \neq 0$ (d.h. Hauptdiagonale besetzt). Es ist klar, dass $|x| = |y|$ erfüllt ist, die Householder-Matrix ergibt sich dann

$$P = I - \underbrace{\frac{2}{v^t\, v}}_{\beta} v\, v^t \qquad \text{mit } v := x - y = x + \underbrace{sgn(x_1)|x|}_{\alpha} e_1$$

Implementieren wir dieses Verfahren in wxMaxima und sehen wie es funktioniert:

```
(%i1)  signValue(r) := block([s:sign(r)], /*  sgn-Fkt wird erzeugt */
         if s='pos then 1 else if s='zero then 0 else -1)$
```

ematrix(n,m,x,i,j) – erzeugt eine $(n \times m)$-Matrix, die überall verschwindet, nur an der Position (i, j) steht x

```
(%i2)  unitVector(n) := ematrix(n,1,1,1,1)/* Einheitsvektor in x-Richtung */$
```

```
(%i3)  householder(A) := block([m : length(A),
                                alpha,v,beta, a:col(A,1)],
         alpha : signValue(A[1,1])*sqrt(a .  a),
         v : a + alpha*unitVector(m),
         beta : 2/(v . v),
         diagmatrix(m,1) - beta*(v . transpose(v)))$
```

$v = x + sgn(x_1)|x|e_1$

```
(%i4)  A:matrix([1,0],
                [1,0],
                [3,0],
                [3,0],
                [4,0])$
```

Der 2-te Spaltenvektor von A ist nur ein "dummy"

```
(%i5)  P:householder(A)$

(%i6)  P . A;
```

(%o6) $\begin{pmatrix} -6 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$ $P \cdot A$ liefert in der 1.-ten Spalte $y = -sgn(x_1)|x|e_1 = (-6, 0, 0, 0, 0)^t$

Die numerische Stabilität des Algorithmus ist natürlich verbesserbar - aber vernachlässigen wir das fürs erste.

Betrachten wir noch einmal die Matrixmultiplikation(mit der Einsteinschen Summationskovention: über alle mehrfach vorkommenden Indices wird summiert!)

$$c_{ij} := a_{ik}\, b_{kj}$$

umgedeutet auf Vektoren bedeutet dies:

$$\underbrace{c_{ij}}_{y} := \underbrace{q_{ik}}_{P}\,\underbrace{a_{kj}}_{x}$$

Für den j-ten Spaltenvektor der Matrix A gibt es eine Householder-Matrix P, sodass der j-te Spaltenvektor der Produktmatrix obige Eigenschaft (nur 1 Komponente ungleich 0) besitzt. Damit können wir folgendes Verfahren durchführen:

1. Wir formen mit Hilfe des Backtracking Algorithmus (Anhang C) die Ausgangsmatrix A so um, dass die Hauptdiagonale besetzt ist. Dies muss möglich sein, sonst wären in einer Spalte nur Nullen und das würde bedeuten das der Rang nicht n sein kann! Um es am Anfang nicht gleich zu verkomplizieren, gehen wir davon aus A würde diese Bedingung bereits erfüllen.
2. Wir wenden auf den 1. Spaltenvektor von A die Householder-Matrix Q_1 an, sodass nur die 1. Komponente nicht verschwindet (wie bei unserem oberen Beispiel).
3. Wir streichen von A 1. Zeile und 1. Spalte und erhalten A_2
4. Wir wenden auf den 1. Spaltenvektor von A_2 die Householder-Matrix Q_2 an, sodass nur die 1. Komponente nicht verschwindet.
5. Wir "blähen" Q_2 zur ursprünglichen Größe auf (Ergebnis Q_2'), sodass andere Spalten unbeeinflusst bleiben.
6. Wir gehen weiter bei Punkt 3) mit einem höherem Index-Zähler
7. Wir brechen ab, wenn wir bei der letzten Spalte angelangt sind

Wir haben also folgendes erreicht:

$$\underbrace{Q'_n\, Q'_{n-1}\, Q'_{n-1}\, Q'_{n-2} \dots Q'_2\, Q_1}_{Q}\; A = R = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{nn} \\ 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 \end{pmatrix}$$

wobei Q als Produkt von orthogonalen Matrizen selbst orthogonal ist (Theorem 4), daher gilt

$$A = Q^t\, R$$

Jetzt müssen wir nur mehr Punkt 5) unseres Verfahrens näher erläutern: das "Aufblähen" der Householder-Matrix. Dazu müssen wir wissen, dass die Multiplikationsformel für Matrizen auch für die Unterteilung in Blockmatrizen gilt, also:

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) \cdot \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right) \qquad \text{wobei} \quad C_{ij} := A_{ik} \cdot B_{kj}$$

wobei in der letzten Formel Einsteinsche Summationskovention gilt und das Multiplikationszeichen eine Matrixmultiplikation darstellt. Zeilen- und Spaltenanzahl der einzelnen Matrizen müssen natürlich derart sein, dass die Produkte existieren!

Zurück zu unserem Verfahren: durch Streichen der linken Spalte und oberen Zeile ergibt sich z.B.: im vierten Schritt folgende Situation:

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & Q_4 \cdot A_{22} \end{array}\right)$$

wobei A_{11} eine 3×3 (obere Dreiecks-)Matrix , A_{21} eine $(m-3) \times 3$ Null-Matrix, A_{12} eine $3 \times (n-3)$ Matrix und A_{22} eine $(m-3) \times (n-3)$ Matrix (wobei a_{11} nicht verschwinden darf!). Unser Verfahren liefert Q_4 wir benötigen aber Q'_4, welche bei Multiplikation A_{11}, A_{12} und A_{21} unverändert lässt - dass lässt sich aber leicht mit folgender Matrix erreichen:

$$\underbrace{\left(\begin{array}{c|c} I & 0 \\ \hline 0 & Q_4 \end{array}\right)}_{Q'_4} \cdot \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right)$$

Ausrechnen ergibt obiges Ergebnis!

Also zusammengefasst: Aus Q'_i wird Q_i indem man in der Hauptdiagonale soviele "1"-er hinzufügt, bis man die ursprüngliche Größe $(m \times m)$ erreicht hat - alles andere wird mit "0" aufgefüllt.

Wir berechnen mit diesem Verfahren ein überbestimmtes Gleichungssystem - wie immer mit *wxMaxima*:

Altbekanntes ....

```
(%i28) signValue(r) := block([s:sign(r)],
         if s='pos then 1 else if s='zero then 0 else -1)$


(%i29) unitVector(n) := ematrix(n,1,1,1,1)$


(%i30) householder(A) := block([m : length(A),alpha,v,beta, a:col(A,1)],
         alpha : signValue(A[1,1])*sqrt(a .  a),
         v : a + alpha*unitVector(m),
         beta : 2/(v . v),
         diagmatrix(m,1) - beta*(v . transpose(v)))$
```

Hier jetzt das Zurücksetzen auf die ursprüngliche Größe *size*, M ist die Q-Matrix und s der "Schrumpfungsgrad" der Matrix, es gilt: $0 \leq s < size$

```
(%i31) setOrigSize(M,s, size):=block(
           genmatrix(lambda([i,j], if (i>s and j>s) then M[i-s,j-s]
                       else if (i=j) then 1
                       else 0),size,size))$
```

Von der Matrix A wird $j-$mal 1-te Zeile und 1-te Spalte gelöscht

```
(%i32) getSubMatrix(A,j):=block([M:A],
         for i thru j do M:submatrix(1,M,1),
       M)$
```

Matrixelemente die kleiner als *threshold* werden, werden durch Null ersetzt!

```
(%i33) setZero(M):=block([mat:M, m:first(matrix_size(M)), n:second(matrix_size(M)), threshold:10^(-15)],
         for i thru m do
           for j thru n do
             if (abs(mat[i,j]) < threshold) then mat[i,j]:0,
       mat)$
```

Hier jetzt die Debugging-Version des rekursiven Householder-Algorithmus:
Input ist die bisherige "obere Dreiecksmatrix" R, in der $recNr$ Spalten unterhalb der Hauptdiagonalen Null gesetzt sind, Q ist das Produkt der einzelnen Householder-Matrizen, $recNrHalt$ ist ein "Breakpoint" (for debugging only), und $origSize$ ist die Dimension von Q, oben $m \times m$ genannt!

```
(%i34) getQR_debug(R,Q,recNr,recNrHalt,origSize):=block([subMat, q],
           if recNr < recNrHalt then block(
             subMat:getSubMatrix(R, recNr),
             q: householder(subMat),
             q: setOrigSize(q, recNr, origSize),
             getQR_debug(q . R, Q . q, recNr+1, recNrHalt, origSize)
           )
         else [Q,setZero(R)])$
```

Hier jetzt die eigentliche Q-R-Faktorisierung: $R := A$, $Q := I$ und
$recNrHalt$:=alle n-Spalten von A, $origSize = m$

```
(%i35) Q_R_Fact(A):=block([m:first(matrix_size(A)), n:second(matrix_size(A))],
         getQR_debug(A,diagmatrix(m,1),0, n, m))$
```

Hier eine Koeffizientenmatrix mit Rang 4, $m = 5$, $n = 4$

```
(%i36) A:matrix(
       [2,1,0,0],
       [1,1,0,0],
       [0,0,1,1],
       [0,0,3,2],
       [0,0,0,1]
       )$
```

Wir berechnen Q und R und schauen uns R an(obere Dreiecksmatrix):

```
(%i37) [q,r]:float(Q_R_Fact(A))$
```

```
(%i38) r;
```

$$\text{(\%o38)}\quad \begin{pmatrix} -2.23606797749979 & -1.341640786499874 & 0.0 & 0.0 \\ 0.0 & -.4472135954999579 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.162277660168379 & -2.213594362117866 \\ 0.0 & 0.0 & 0.0 & 1.048808848170151 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Wir setzen den Ergebnisvektor $\vec{b}$ so, dass sich ein Lösungsvektor $\vec{x} = (1, 2, 3, 4)^t$ ergibt

```
(%i39) b:matrix([4],[3],[7],[17],[4])$
```

Durch "Rückwärtseinsetzen" wird die Lösung bestimmt:

```
(%i40) backwardSubstitution(r,b):=
         block([cols:second(matrix_size(r)), x:zeromatrix(second(matrix_size(r)),1)],
           for c:cols thru 1 step -1 do  x[c]:((b[c]-row(r,c) . x)/r[c,c]),
         x)$
```

```
(%i41) solutionVec:backwardSubstitution(r, transpose(q) . b);
```

$$\text{(\%o41)}\quad \begin{pmatrix} 1.0 \\ 2.0 \\ 3.000000000000001 \\ 4.0 \end{pmatrix}$$

Jetzt wird der Ergebnisvektor abgeändert, dass er widersprüchlich wird:

```
(%i42) b:matrix([4.5],[3],[7.5],[16],[3.4])$
```

Neuer Lösungsvektor wird bestimmt

```
(%i43) solutionVec:backwardSubstitution(r, transpose(q) . b);
```

$$\text{(\%o43)}\quad \begin{pmatrix} 1.5 \\ 1.5 \\ 2.972727272727274 \\ 3.681818181818181 \end{pmatrix}$$

Erfüllt dieser Lösungsvektor die *Normalengleichung*? (Theorie siehe unten im Text!)

```
(%i44) transpose(A) . b;
```

(%o44)
$$\begin{pmatrix} 12.0 \\ 7.5 \\ 55.5 \\ 42.9 \end{pmatrix}$$

Offensichtlich

```
(%i46) transpose(A) . (A . solutionVec);
```

(%o46)
$$\begin{pmatrix} 12.0 \\ 7.5 \\ 55.50000000000001 \\ 42.9 \end{pmatrix}$$

Das lineare Ausgleichsproblem

Wir haben ein überbestimmtes lineares Gleichungssystem

$$a_{ij}\, x_j = b_i \qquad 1 \le i \le m,\, 1 \le j \le n,\, n \le m$$

Wir suchen $\hat{x}_j$, sodass die Summe der Residuenquadrate minimal wird:

$$r_i = b_i - a_{ij}\, x_j \qquad S = r_i\, r_i \to min$$

Dazu muss der Gradient von S verschwinden:

$$\frac{\partial S}{\partial x_j} = \frac{\partial r_i}{\partial x_j} \cdot r_i + r_i \cdot \frac{\partial r_i}{\partial x_j} = 2\, r_i \cdot \frac{\partial r_i}{\partial x_j} \overset{\frac{\partial r_i}{\partial x_j} = -a_{ij}}{\uparrow} = 2\,(b_i - a_{ik}\hat{x}_k)\,(-a_{ij}) = 0$$

Ausmultiplizieren führt zu den Normalengleichungen

$$\boxed{a_{ij}\, a_{ik}\, \hat{x}_k = b_i\, a_{ij}} \qquad \text{in Matrixschreibweise} \qquad \boxed{A^t\, A\, \vec{x} = A^t\, \vec{b}}$$

Dies haben wir oben in *wxMaxima* durchgeführt!