Übung 1 *QR-Zerlegung mittels Householder-Spiegelung (Praktische Übung)*
(Für diese praktische Übung gibt es zwei Wochen Bearbeitungszeit. Abgabe: 01.Juli 2011, 9:15 Uhr.)

Zur Lösung des linearen Ausgleichsproblems

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2$$

(wobei $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und Rang$(A) = n$) soll die QR-Zerlegung mit Householder-Spiegelungen eingesetzt werden. Lade die Datei `incomplete.tar` von der Vorlesungs-Webseite herunter und entpacke sie mit dem Befehl

```
tar xvf incomplete.tar
```

Das vorgegebene Hauptprogramm `curvefitting.cc` löst die Aufgabe 3 aus dem letzten Übungsblatt. Für die dort angegebenen Wertepaare erhält man für das gesuchte Polynom ersten Grades die Matrix A aus `A6_linear.dat` und die rechte Seite b aus `b6.dat`. Analog erhält man für das gesuchte Polynom zweiten Grades die Matrix A aus `A6_quadratic.dat`.

Im Hauptprogram wird die Funktion

```
template<typename NUMBER>
void solveLeastSquares( hdnum::DenseMatrix<NUMBER> &A,
                        hdnum::Vector<NUMBER> &b,
                        hdnum::Vector<NUMBER> &x )
```

aufgerufen, welche die Lösung $x \in \mathbb{R}^n$ unseres linearen Ausgleichsproblems liefern soll.

a) Implementiere diese Funktion in der Headerdatei `solveLeastSquares.hh` mit dem

Algorithmus 9.1:
1.) Berechne die QR_Zerlegung von A. Verwende dafür die Funktion aus Teil b)!

2.) Für $j = 1, ..., n$*:*

$$\begin{array}{ll} v_j = 1,\ v_l = A_{lj} & l = j+1, ..., m \\ \beta = \frac{2}{v^T v} & v = (v_j, v_{j+1}, ..., v_m)^T \\ b = (I - \beta v v^T) b & b = (b_j, b_{j+1}, ..., b_m)^T \end{array}$$

3.) Löse das obere Dreieckssystem $Ax = b$ *(d.h. benutze* `solveR.hh`*). Das liefert* x*.*

b) Die Funktion

```
template<typename NUMBER>
void householder_QR( hdnum::DenseMatrix<NUMBER> & A )
```

liefert die QR-Zerlegung von A. Dabei wird A durch die Funktion verändert und enthält am Ende die obere Dreiecksmatrix R und die normierten Householder-Vektoren. Normiert bedeutet hier, dass der Householder-Vektor in der ersten Komponente die 1 enthält und diese deshalb nicht extra abgespeichert werden muss.

Implementiere diese Funktion in der Headerdatei `householderQR.hh` mit dem

Algorithmus 9.2:
Für $j = 1, ..., n$:

1.) *Definiere den Vektor y mit den Einträgen $y_l = A_{lj}$ $(l = j, ..., m)$ und berechne dafür den normierten Householder-Vektor v mit zugehörigem β. Benutze dafür die Funktion* `householder_vector(...)` *(siehe weiter unten!).*

2.) *Wende die durch $P := Id - \beta vv^T$ definierte Householdertransformation auf die Teilmatrix $(A)_{pq}$ mit $p \in \{j, ..., m\}$ und $q \in \{j, ..., n\}$ an. Benutze dafür die Funktion* `householderP_times_matrix(...)` *(siehe weiter unten!).*

3.) *Solange $j < m$ ist, setze $A_{lj} = v_{l-j+1}$ für $l \in \{j + 1, ..., m\}$.*

Beachte, dass in diesen Algorithmen die Dimensionen der Vektoren und Matrizen mit j varriieren. Tip: Dafür dürfen Kopien von Teilvektoren oder Teilmatrizen geeigneter Dimensionen angelegt werden.

Die Funktion

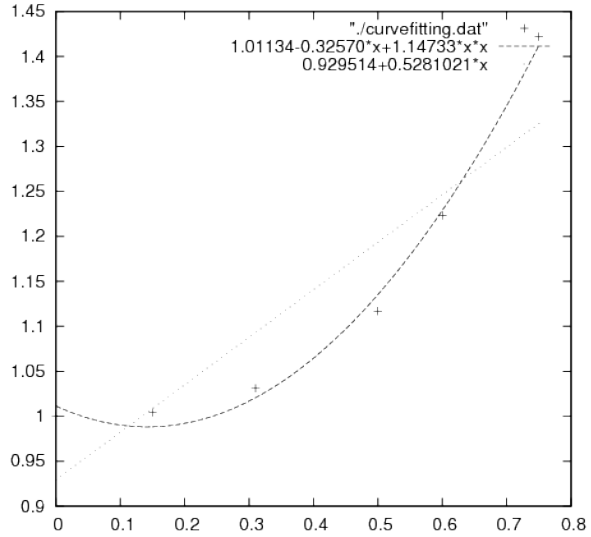
```
template<typename NUMBER>
void householder_vector( hdnum::Vector<NUMBER> &x,
                         hdnum::Vector<NUMBER> &v,
                         NUMBER & beta )
```

liefert für einen Vektor x beliebiger Dimension einen normierten Householder-Vektor v gleicher Dimension und eine Zahl β zurück, mit denen man die Householdertransformation $P := I - \beta vv^T$ definieren kann. Hier wird P gar nicht explizit berechnet, denn wir benötigen nur ihre Wirkung auf eine andere Matrix:
Die Funktion

```
template<typename NUMBER>
void householderP_times_matrix( const NUMBER & beta,
                                const hdnum::Vector<NUMBER> &v,
                                hdnum::DenseMatrix<NUMBER>& A
                                )
```

bekommt als Input die Zahl β und den Vektor v und eine beliebige Matrix A und gibt das Matrizenprodukt $P \cdot A = (I - \beta vv^T) \cdot A$ zurück.

Für weitere Einzelheiten über die Algorithmen siehe *Golub/van Loan: Matrix computations*.



Teste die Implementierung für die beiden Fälle `A6_linear.dat` und `A6_quadratic.dat`.

(5+5 Punkte)

Übung 2 *Schema von Aitken-Neville*

Sei P_k der Raum der Polynome über $\mathbb{R}$ vom Grad $\leq k \in \mathbb{N}_0$. Sei $p_{i,k} \in P_k$ das eindeutig bestimmte Interpolationspolynom zu den Wertepaaren $(x_i, y_i), \ldots, (x_{i+k}, y_{i+k})$.

a) Zeige, dass folgende Rekursionsformel gilt:

$$(i) \quad p_{i,0}(x) = y_i \qquad \text{für } i = 0, ..., n,$$

$$(ii) \quad p_{i,k}(x) = \frac{(x - x_i)p_{i+1,k-1}(x) - (x - x_{i+k})p_{i,k-1}(x)}{x_{i+k} - x_i} \qquad \text{für } i = 0, ..., n - k.$$

b) Damit lässt sich das Interpolationspolynom $p_{0,n}(x)$ zu den Wertepaaren $(x_0, y_0), \ldots, (x_n, y_n)$ an einem Punkt $x = \zeta$ auswerten, ohne die Koeffizienten des Polynoms explizit zu berechnen.

Für veschiedene Orte wurde an einem bestimmten Tag die Tageslänge gemessen:

Ort	Tageslänge	Lage
A	17h 28m	55,7 °
B	18h 00m	57,7 °
C	18h 31m	59,3 °
D	19h 56m	62,6 °
E	22h 34m	65,6 °

Man bestimme die Tageslänge am Ort F bei 61,7 ° durch Auswertung des zugehörigen Interpolationspolyoms mit Hilfe des Neville-Schemas (Es genügt 4-stellige Dezimalrechnung).

(5 Punkte)

Übung 3 *Komplexität der Interpolation*

Sei $p \in P_n$ das Interpolationspolynom zu den $n+1$ paarweise verschiedenen Stützstellen $t_0, ..., t_n$ mit den zugehörigen Werten $y_0, ..., y_n$. Bestimme die Anzahl der benötigten Operationen

- zur Berechnung der Koeffizienten von p
- und zur Auswertung von p an einer beliebigen Stelle $t = \xi$

a) bezüglich der Lagrange-Basis,

b) bezüglich der Newton-Basis und

c) bezüglich der Monom-Basis.

Bestimme zum Vergleich auch die Anzahl der benötigten Operationen zur Auswertung von $p(t)$ an einer beliebigen Stelle $t = \xi$ mit Hilfe des Neville-Aitken-Schemas.

Hinweis: Die naive Auswertung des Polynoms $p(t) = a_0 + a_1 t + ... + a_n t^n$ (in der Monom-Basis) erfordert $O(n^2)$ Mutliplikationen und n Additionen. Dagegen wird beim **Hornerschema**

$$p(t) = a_0 + (t \cdot (a_1 + t \cdot (\ldots (a_{n-1} + t \cdot a_n) \ldots)))$$

von innen nach außen ausgewertet. Wieviele Additionen und Multiplikationen sind dafür notwendig?

(4 Punkte)

Übung 4 *Äquidistante Stützstellen*

Zeige, dass man bei äquidistanten Stützstellen $x_i = x_0 + jh$, $j = 0, \ldots, n$, $h > 0$ die Newton-Darstellung des Interpolationspolynoms in

$$p(x) = \sum_{k=0}^{n} \left(\begin{array}{c} s \\ k \end{array} \right) k! \cdot h^k \cdot f[x_0, \ldots, x_k] \qquad \text{mit } s = \frac{x - x_0}{h}$$

umwandeln kann, wobei der Binomialkoeffizient durch

$$\left(\begin{array}{c} s \\ k \end{array} \right) = \frac{s \cdot (s-1) \cdots (s-k+1)}{k!}$$

auch für eine reelle Zahl $s \in \mathbb{R}$ definiert ist. **(4 Punkte)**