Prof. Dr. Marc Alexander Schweitzer
Denis Düsseldorf

universitätbonn

Übungsblatt 12.

Abgabe: 14.01.2019 vor der Vorlesung

Aufgabe 38. (Orthogonalpolynome)

Gegeben sei die Gewichtsfunktion

$$\omega(x) = 1 + x^2.$$

Berechnen Sie die ersten Orthogonalpolynome, ϕ_0 und ϕ_1 bezüglich des Skalarproduktes

$$\langle f, g \rangle_\omega := \int_{-1}^{1} f(x)g(x)\omega(x)\,dx,$$

mit $\|\varphi_i\|_\omega = 1$, $i = 0, 1$ und führenden Koeffizienten > 0.

(4 Punkte)

Aufgabe 39. (Quadraturformeln)

In dieser Aufgabe wiederholen wir die Konstruktion von interpolatorischen Quadraturformeln sowie ihre wesentliche Eigenschaften.

Es sei $f : [a, b] \to \mathbb{R}$ eine integrierbare Funktion. Die Berechnung von

$$I(f) := \int_a^b f(x)\,dx$$

kann aufwendig oder analytisch sogar nicht durchführbar sein. Jede explizite Formel zur Näherung an $I(f)$ wird Quadraturformel genannt. Eine Möglichkeit ist es, die Funktion $f(x)$ durch eine Approximation $f_n(x)$ zu ersetzen, für die man das Integral

$$I_n(f) := \int_a^b f_n(x)\,dx$$

einfacher berechnen kann. Da Polynome einfach zu interieren sind, wählen wir $n + 1$ paarweise verschiedene Stützstellen $x_0, \ldots, x_n \in [a, b]$.

a) Es sei $\ell_{i,n}(x)$ das i-te Lagrange Polynom

$$\ell_{i,n}(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \ldots, n.$$

Konstruieren Sie das Interpolationspolynom f_n zu f vom Grad n mit Hilfe der Lagrange Polynome.

b) Bringen Sie die Quadraturformel I_n in die Form

$$I_n(f) = \sum_{i=0}^{n} \omega_i f(x_i),$$

d.h. geben Sie die Gewichte ω_i, $i = 0, \ldots, n$ explizit an.

c) Konstante Funktionen $f(x) \equiv c$ sollen stets exakt integriert werden können. Welche Bedingung ergibt sich hieraus für die Gewichte?

d) Zeigen Sie, dass die so konstruierte Quadraturformel alle Polynome vom Grad n exakt integriert (man sagt, dass die Quadraturformel exakt vom Grad n ist).

e) Zeigen Sie, dass die Quadraturformel nicht exakt vom Grad $2(n+1)$ ist.
Tipp: Benutzen Sie das Polynom

$$p(x) = \prod_{i=0}^{n} (x - x_i).$$

(4 Punkte)

Programmieraufgabe 7. (Householder Spiegelungen & Givens Rotationen)

In dieser Programmieraufgabe implementieren Sie die QR Zerlegung mittels Householder Spiegelungen, sowie mittels Givens Rotationen.

a) Schreiben Sie eine Routine, die zu einem Vektor $\tilde{v} \in \mathbb{R}^k$ die Householder Matrix $\tilde{H}_{\tilde{v}} \in \mathbb{R}^{k \times k}$ berechnet, d.h.

$$\tilde{H}_{\tilde{v}} = I - 2\tilde{v}\tilde{v}^T.$$

Dabei soll es sich um die reduzierte Householder Matrix aus der Blockdarstellung

$$H_v = \begin{bmatrix} I & 0 \\ 0 & \tilde{H}_{\tilde{v}} \end{bmatrix}$$

handeln. Es handelt sich also mathematisch gesehen immer um die Spiegelung von $\tilde{v}$ auf den ersten Einheitsvektor im $\mathbb{R}^k$. Benutzen Sie die Signatur

```
def compute_householder_matrix_reduced(v):
    # Input:
    #    v          vector of length k
    # Output:
    #    H_v        Reduced Householder matrix

    # Ihr Code hier
```

b) Nun berechnen Sie die vollständige Householder Matrix. Gegeben sei ein Vektor $v \in \mathbb{R}^n$ und ein $m \in \{1, \ldots, n-1\}$. Schneiden Sie aus dem Vektor v den Teilvektor $\tilde{v} \in \mathbb{R}^{n-m}$ heraus, mit

$$\tilde{v} = \begin{bmatrix} v_{m+1} & \ldots & v_n \end{bmatrix}$$

Benutzen Sie die Routine aus a) um die reduzierte Householder Matrix für diesen Vektor zu berechnen. Betten Sie die reduzierte Matrix $\tilde{H}_{\tilde{v}} \in \mathbb{R}^{n-m \times n-m}$ anschließend in die Matrix $H_v \in \mathbb{R}^{n \times n}$ ein. Benutzen Sie die Signatur

```
def compute_householder_matrix_full(v,m):
    # Input:
    #    v          Vector of size n x n
    #    m          slicing index, size of tilde v
    # Output:
    #    H_v        Full householder matrix of size n x n

    # Ihr Code hier
```

c) Schreiben Sie eine Routine zur Berechnung der QR Zerlegung mittels Householder Transformationen,

```python
def compute_qrdecomp_householder(A):
    # Input
    #   A          real n x n matrix
    # Output
    #   Q          Orthogonal matrix from qr decomposition
    #   R          upper triangular matrix from qr decomposition

    # Ihr Code hier
```

d) Schreiben Sie eine Routine, die zu einer gegebenen Matrix A die Matrix der Givens-Rotation zurück gibt, welche den i-ten Eintrag des j-ten Spaltenvektors (für $i > j$) auf 0 dreht (siehe Blatt 11, Aufgabe 26). Die Einträge a und b des Drehkästchens in dieser Matrix sind explizit gegeben durch

$$a = \frac{A_{jj}}{\sqrt{A_{jj}^2 + A_{ij}^2}}, \quad b = -\frac{A_{ij}}{\sqrt{A_{jj}^2 + A_{ij}^2}}.$$

Benutzen Sie die Signatur

```python
def compute_givens_rotation(A,i,j):
    # Input
    #   A          real matrix, n x n
    #   i          row index
    #   j          column index, i > j
    # Output
    #   G          Matrix of givens rotation for A_ij -> 0

    # Ihr Code hier
```

e) Schreiben Sie eine Routine, die mittels Givens Rotationen die QR Zerlegung einer reellen Matrix A berechnet.

```python
def compute_qrdecomp_givens(A):
    # Input
    #   A          real n x n matrix
    # Output
    #   Q          matrix Q from qr decomp
    #   R          matrix R from qr decomp

    # Ihr code hier
```

f) Testen Sie die beiden Verfahren zur QR Zerlegung an der Matrix *matrix_qr.txt* die Sie auf der Webseite finden. Vergleichen Sie die Laufzeiten. Der folgende Code-Schnipsel zeigt Ihnen, wie Laufzeiten in Python gemessen werden können:

```python
import time
start = time.time()
# Do some stuff
end = time.time()
print('Execution took {:5.3f} seconds').format(end-start)
```

(5 Punkte)

Abgabe per Mail.