Einführung in die Grundlagen der Numerik

Winter semester 2019/2020
Prof. Dr. Marc Alexander Schweitzer
Denis Düsseldorf

universitätbonn

Übungsblatt 3.

Abgabe: 05.11.2019 vor der Vorlesung

Aufgabe 7. (Projektionsmethoden und Krylov-Unterraum Verfahren)

Die Krylov-Unterraum Verfahren sind Beispiele so genannter Projektionsmethoden zum Lösen eines linearen Gleichungssystems $Ax = b$, wobei $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix und $b \in \mathbb{R}^n$ ein Vektor ist. Projektionsmethoden versuchen, ausgehend von einer Anfangsnäherung $x_0 \in \mathbb{R}^n$, eine Approximation x_k an die echte Lösung $x^* \in \mathbb{R}^n$ in einem k dimensionalen Unterraum $K_k \subset \mathbb{R}^n$ zu finden. Dabei gilt die Beziehung

$$x_k \in x_0 + K_k \tag{P1}$$

mit der zusätzlichen Orthogonalitätsbedingung (euklidisches Skalarprodukt)

$$(b - Ax_k) \perp L_k \subset \mathbb{R}^n. \tag{P2}$$

Ein **Krylov Verfahren** erhält man nun durch spezielle Wahl des Approximationsraumes K_k als

$$K_k = K_k(A, r_0) \coloneqq \operatorname{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}, \tag{K1}$$

wobei $r_0 = b - Ax_0$ das Startresiduum ist.
In jedem Iterationsschritt wird die Dimension des Suchraumes K_k und des Orthogonalraums zum Residuum, L_k, um 1 erhöht - also terminieren Krylov Verfahren (in exakter Arithmetik!) wegen $K_n = L_n = \mathbb{R}^n$ spätestens nach n Iterationen. Rundungsfehler in der numerischen Anwendung sorgen jedoch dafür, dass die Iterationszahl nur theoretisch beschränkt ist.

Im Folgenden sollen Sie verschiedene Teile eines Beweises zur Abschätzung der Approximationsgüte nach k Iteration eines Krylov-Verfahrens beweisen. Mit dem Resultat aus e) folgt nämlich für den Fehler $e_k = A^{-1}b - x_k$ und das Residuum $r_k = Ae_k$, dass

$$\|e_k\| \leq \|A^{-1}P_k\| \min_{p\in\mathcal{P}^{k,1}} \|p(A)r_0\|$$
$$\|r_k\| \leq \|P_k\| \min_{p\in\mathcal{P}^{k,1}} \|p(A)r_0\|.$$

Die benutzten Operatoren und Größen lernen Sie in den Teilaufgaben kennen.

a) Es sei $V_k \in \mathbb{R}^{n\times k}$ eine Matrix bestehend aus Basisvektoren des Suchraumes K_k, und $W_k \in \mathbb{R}^{n\times k}$ eine Matrix bestehend aus Basisvektoren des Orthogonalraums zum Residuum, L_k. Ausserdem sei angenommen, dass

$$W_k^T A V_k \in \mathbb{R}^{k\times k}$$

regulär ist. Zeigen Sie, dass die Approximation $x_k \in K_k$ die Beziehung

$$x_k = x_0 + V_k \left[W_k^T A V_k \right]^{-1} W_k^T r_0$$

erfüllt.

b) Zeigen Sie, dass wir mit dem Resultat aus a) den Residuenvektor $r_k = b - Ax_k$ schreiben können als

$$r_k = r_0 - A V_k \left[W_k^T A V_k \right]^{-1} W_k^T r_0.$$

c) Zeigen Sie, dass der Operator

$$P_k = \mathbb{I} - A V_k \left[W_k^T A V_k \right]^{-1} W_k^T$$

ein Projektor ist, d.h. dass $P_k^2 = P_k$ gilt. Hierbei ist $\mathbb{I}$ die Identität.

d) Zeigen Sie die Beziehung

$$P_k A V_k = 0.$$

e) Zeigen Sie, dass aus b) und d) die Beziehung

$$r_k = P_k \left[r_0 + A V_k z \right], \quad \forall z \in \mathbb{R}^n$$

folgt. Im Fall von Krylov-Unterräumen erhält man so

$$r_k = P_k p(A) r_0,$$

für jedes beliebige Polynom $p \in \mathcal{P}^{k,1}$ wobei $\mathcal{P}^{k,1}$ den Raum der Polynome vom Grad k mit $p(0) = \mathbb{I}$ bezeichnet.

Notation: Wir überladen hier die Notation ein wenig: Das Polynom p aus obiger Aufgabe wird an Matrizen ausgewertet. Potenzen des Arguments, wie bspw. das Quadrat in $p(\cdot) = 2 \cdot^2 + \cdot + \mathbb{I}$ sind als Matrix-Potenzen zu verstehen.

(6 Punkte)

Aufgabe 8. (B-Splines Gram-Matrix)

In der folgenden Aufgabe werden Kenntnisse wiederholt / vermittelt, die in einigen Algorithmen aus der Vorlesung Anwendung finden werden.

Im Folgenden sei eine Menge von $n \in \mathbb{N}$ Knoten $t_i \leq t_{i+1}$, für $i = 0, \ldots, n$ gegeben. Wir definieren $n - 1$ verschiedene Funktionen, die so genannten **B-Splines** vom Grad 0, als

$$N_{i,0} \coloneqq \begin{cases} 1, & x \in [t_i, t_{i+1}) \\ 0, & \text{sonst} \end{cases}$$

Rekursiv erhält man nun die B-Splines höherer Ordnung $p > 0$ über

$$N_{i,p}(x) \coloneqq \frac{x - t_i}{t_{i+p} - t_i} N_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(x)$$

für $i = 0, \ldots, n - p - 1$. Um nicht versehentlich durch 0 zu teilen fallen alle Summanden mit $t_{i+p} = t_i$ oder $t_{i+p+1} - t_{i+1}$ weg.

a) Zeigen Sie, dass $N_{i,p}$ für $p \geq 0$ und $i = 0, \ldots, n-p-1$ ein Polynom vom Grad p ist. Machen Sie sich außerdem klar, dass der Träger von $N_{i,p}$ für steigende p größer wird.

b) Es sei $t_i = ih \in [0,1]$ für $i = 0, \ldots, n$ mit $h = \frac{1}{n}$. Zeigen Sie, dass

$$N_{i,2}(x) = f\left(\frac{x - t_i}{h}\right), \quad \text{mit} \quad f(y) = \begin{cases} \frac{1}{2}y^2, & y \in [0,1) \\ \frac{1}{2}\left(-2y^2 + 1\right), & y \in [1,2) \\ \frac{1}{2}\left(y^2 - 2y + 1\right), & y \in [2,3) \\ 0, & \text{sonst.} \end{cases}$$

c) Berechnen Sie für das Setting aus c) die Größen $\langle N_{i,2}, N_{j,2} \rangle$ für $i, j = 0, \ldots, n-3$ im L^2 Skalarprodukt

$$\langle f, g \rangle \coloneqq \int_0^1 fg \, dx.$$

d) Skizzieren Sie mit Ihren Ergebnissen aus *d*) die Struktur der Gram-Matrix für das L^2 Skalarprodukt.

(6 Punkte)

Programmieraufgabe 2. (B-Splines)

a) Implementieren Sie eine Funktion, die zu einem beliebigen gegebenen Knotenvektor die Funktion $N_{i,p}(x)$ mittels Rekursion auswertet. Die Auswertungspunkte x sollen als Liste übergeben werden. Achten Sie auch darauf, die Fälle $t_{i+p} = t_i$ bzw. $t_{i+p+1} = t_{i+1}$ richtig zu handhaben (Konvention: Terme werden weggelassen!). Benutzen Sie die Folgende Signatur für Ihre Funktion.

```
def evaluate_bspline_recursive(ts, xs, i, p):
    # Input:
    #       ts      Knotenvektor
    #       xs      Punkte an denen N_{i,p} ausgewertet werden soll
    #       i       Nummer des B-Splines, siehe Definition
    #       p       Ordnung des B-Splines, siehe Definition
    # Output:
    #       ys      Funktionswerte fuer x in xs
```

b) Erstellen Sie für jedes $p = 0, 1, 2, 3$ einen eigenen Plot mit allen B-Splines $N_{i,p}$ zum Knotenvektor

$$ts = (\underbrace{0, \ldots, 0}_{p+1 \text{ mal}}, h, 2h, 3h, 4h, 5h, \underbrace{1, \ldots, 1}_{p+1 \text{ mal}}), \quad \text{mit } h = \frac{1}{6}.$$

Benutzen Sie die Auswertungspunkte $xs = np.linspace(0, 1, 100)$.

c) In den Programmieraufgaben von Blatt 0 wurde bereits ein einfaches Schema zur numerischen Integration implementiert. Benutzen Sie die verbesserte m-Intervall Trapezregel jetzt, um die Gram Matrizen für $p = 0, 1, 2, 3$ und variables n zu erstellen.

Sie können(!) die auf der Webseite verfügbare Implementierung dieser Quadraturregel aus benutzen. Legen Sie diese hierzu im gleichen Ordner wie Ihre eigene Skriptdatei ab und binden Sie sie in Ihr Skript mit dem Befehl `import <filename>` ein.

Benutzen Sie folgende Signatur für Ihre Routine.

```
def assemble_grammatrix_bspline_recursive(ts, p):
    # Input:
    #       ts      Knotenvektor
    #       p       Ordnung der B-Splines, siehe Definition
    # Output:
    #       G       Gram Matrix als np.array
```

(5 Punkte)

Abgabe per Mail an die Tutoren