Universität zu Köln

Prof. Dr. Angela Kunoth
Samuel Leweke

Numerik für Optimierungsprobleme mit PDEs II — Sommersemester 2017

Ausgabe: 14.06.2017

Übung 6

Abgabe: 21.06.2017, 10:00 Uhr

Aufgabe 15 (20 Punkte)

Betrachten Sie die Laplace-Gleichung

$$\begin{aligned} -\Delta y &= 1 && \text{in } \Gamma := (0,1) \\ y &= 0 && \text{auf } \partial\Gamma = \{0,1\}. \end{aligned}$$

Auf jedem Level $J \in \mathbb{N}$ sind die Ansatzfunktionen $\varphi_{J,k}$ mit $k = 1, \dots, 2^J - 1$ durch Translation und Dilatation von Hutfunktionen φ gegeben:

$$\varphi_{J,k}(x) := \varphi\left(2^J x - k\right), \qquad \varphi(x) := \begin{cases} 1+x, & x \in [-1,0) \\ 1-x, & x \in [0,1) \\ 0, & \text{sonst.} \end{cases}$$

In dieser Aufgabe sollen drei Vorkonditionierer für das obige Problem in einem PCG-Verfahren miteinander verglichen werden. Die Matrizen A_J und Vektoren f_J des nach der Diskretisierung auf Level J entstehenden linearen Gleichungssystems $A_J y_J = f_J$ werden durch die Matlab-Funktion `[A, f] = LaplaceMatVek1D(J)` von der Homepage der Vorlesung erzeugt.

a) Implementieren Sie eine Matlab-Funktion `y = MGM1D(J, Jmin, Aj, fj, yj)`, welche einen V-Zyklus ($\mu = 1$) des Multigrid-Verfahrens mit $\nu_1 = \nu_2 = 1$ durchführt. Passen Sie dazu Ihre Implementierung von Aufgabe 6 aus Übung 3 an.

b) Schreiben Sie eine Matlab-Funktion `y = BPX(J, Jmin, A, x)`, welche den BPX-Vorkonditionierer

$$y = \sum_{j=\texttt{Jmin}}^{J} I_j^J \left[\operatorname{diag}(A_j)\right]^{-1} I_J^j x$$

auf einen Vektor x anwendet. Die Matrizen I_k^ℓ bezeichnen die Restriktions- und Prolongationsoperatoren welche einen Vektor von Level k auf Level ℓ transferieren (siehe Multigrid-Verfahren). Mit A_j wird die Steifigkeitsmatrix auf dem Level j bezeichnet, welche analog zum Korrektor-Problem im Multigrid-Verfahren gewonnen werden kann.

c) Implementieren Sie eine Funktion `y = fwt(x, J)`, welche auf einen gegebenen Vektor x das Pyramid-Schema anwendet um die schnelle Wavelet-Transformation (FWT) zu berechnen. Nutzen Sie dazu die auf der Homepage herunterladbaren Verfeinerungsmatrizen (`M22_3.mat`) für $j = 3$, $d = 2$ und $\tilde{d} = 2$. Stellen Sie dabei, wenn es Ihnen möglich ist, die Transformationsmatrix $T_J := T_{J,J-1} \dots T_{J,j_0}$ nicht explizit auf.

d) Implementieren Sie eine Funktion `y = fwtT(x, J)`, welche auf einen gegebenen Vektor x die zur FWT transponierte Transformation T_J^T anwendet. Stellen Sie dabei, wenn es Ihnen möglich ist, die Transformationsmatrix nicht explizit auf.

e) Implementieren Sie eine Funktion `y = diagScaling(x, J)`, welche auf einen Vektor x die diagonale Skalierungsmatrix $\left(D_J^{-1}\right)_{(j,k),(j,k)} = 2^{-j}$ anwendet.

f) Nutzen Sie die Matlab-Funktion `pcg` um das vorkonditionierte CG-Verfahren auf das obige Problem anzuwenden. Schreiben Sie dazu ein Skript `vorkond.m`, in welchem jeweils für die vier folgenden Vorkonditionierer und die Level $J = 4, 5, \ldots, 12$ die Anzahl der benötigten Iterationen, die zum Erreichen des Diskretisierungsfehlerniveaus `tol` $= 10^{-4}2^{-J}$ benötigt werden, aufgezeichnet werden:

   i) Keine Vorkonditionierung,

   ii) ein V-Zyklus des Multigrid-Verfahrens aus Teil a),

   iii) BPX-Vorkonditionierung aus Teil b) und

   iv) Wavelet-Vorkonditionierung durch Lösen des Systems $\left[D_J^{-1}T_J^T A_J T_J D_J^{-1}\right]\tilde{y}_J = D_J^{-1}T_J^T f_J$ (Teile c) – e)).

   Tragen Sie die Anzahl der benötigten Iterationen über dem Level J in einem gemeinsamen Plot auf und achten Sie dabei auf eine geeignete Skalierung der Achsen. Wählen Sie jeweils `Jmin` $= 1$ im Multigrid- und BPX-Vorkonditionierer.

   Erklären Sie das beobachtete Verhalten.

Hinweise:

- Die Funktion `pcg` bietet ein Argument zur Übergabe eines Vorkonditionierers an. Dieser kann auch ein function-handle (z. B. anonyme Funktion) sein: `@(x) BPX(J, 1, A, x)`
- Statt der Matrix A_J kann der Funktion `pcg` auch ein function-handle (z. B. anonyme Funktion) übergeben werden, welche die Matrix auf einen gegebenen Vektor anwendet: `@(x) A*x` für Ax oder `@(x) fwtT(A * fwt(x, 1), 1)` für $T_J^T A T_J x$
- Die Funktion `pcg` liefert auch die Anzahl der benötigten Iterationen zurück.
- Implementieren Sie den BPX-Vorkonditionierer so, dass die Prolongations- und Restriktionsoperatoren I_j^{j+1} und I_{j+1}^j, aus denen die Operatoren

$$I_j^J \coloneqq I_{J-1}^J \ldots I_{j+1}^{j+2} I_j^{j+1} \qquad \text{und} \qquad I_J^j \coloneqq I_{j+1}^j I_{j+2}^{j+1} \ldots I_J^{J-1}$$

  bestehen, nur einmal aufgebaut und dann wiederverwendet werden.
- Beachten Sie bei der Verfeinerungsmatrix aus `M22_3.mat` die Aufteilung in $M_{j,0}$, $M_{j,1}$ und die Anpassung auf homogene Dirichlet-Randwerte der jeweils ersten und letzten Spalte von $M_{j,0}$, $M_{j,1}$. Bei der Anpassung auf höhere Level J werden sich die jeweils innere(n) Spalte(n) von $M_{j,0}$ und $M_{j,1}$ wiederholen und nur die jeweils erste und letzte Spalte gleich bleiben.
- Beachten Sie, dass die zur Aufbau der Matrix A_J verwendeten Hut-Funktionen nicht mit $2^{J/2}$ skaliert sind.
- Verwenden Sie die (eindimensionalen) Restriktions- und Prolongationsoperatoren aus Aufgabe 6 (Übung 3) für den Multigrid- und den BPX-Vorkonditionierer.