Karlsruher Institut für Technologie

Institut für Angewandte und Numerische Mathematik
Prof. Dr. Tobias Jahnke, Dipl.-Math. techn. Patrick Krämer, M. Sc. Simon Baumstark

Einführung in das Wissenschaftliche Rechnen

Sommersemester 2016

Übungsblatt zu quadratischen finiten Elementen

Abgabe spätestens am 15.06.2016

Aufgabe 4:
Wir betrachten wieder einmal das Poisson-Problem

$$\begin{aligned} -\Delta u(x,y) &= f(x,y) && \text{für } (x,y) \in \Omega \\ u(x,y) &= g_D(x,y) && \text{für } (x,y) \in \Gamma \end{aligned}$$

auf einem beschränkten Gebiet $\Omega \subset \mathbb{R}^2$ mit Rand Γ. Die Funktionen f und g_D seien so gewählt, dass $u(x,y) = 1 - xy^3$ die exakte Lösung ist. Die Lösung soll nun durch quadratische finite Elemente approximiert werden.

(a) Schreiben Sie eine Funktion `computeAb_quad`, welche die Steifigkeitsmatrix A und den Lastvektor b des entsprechenden linearen Gleichungssystems berechnet. Ändern bzw. erweitern Sie dazu die Funktion `computeAb` aus Aufgabe 3 (bzw. der Musterlösung) in geeigneter Weise. Neumann-Randbedingungen müssen *nicht* implementiert werden. Der Aufruf soll mit

```
[A,b,Tarea] = computeAb_quad(T2,P2,EdgesDir2);
```

erfolgen. Dabei sind `T2`, `P2` und `EdgesDir2` die Daten der Triangulierung, die auch die Seitenmittelpunkte als Knoten enthält, wie das für quadratische finite Elemente nötig ist. Diese Daten erzeugen Sie durch

```
[T2,P2,EdgesDir2,EdgesNeu2] = triag_lin2quad(T,P,EdgesDir,EdgesNeu);
```

wobei die Funktion `triag_lin2quad` zusammen mit den Triangulierungen auf der Webseite der Vorlesung bereitgestellt wird. Die Triangulierungen müssen (wie in früheren Aufgaben) im Unterverzeichnis `MyGrids` abgelegt werden.
Der Vektor `Tarea` soll die Flächeninhalte der Dreiecke der Triangulierung enthalten. Die Funktionen f und g_D werden durch zwei MATLAB-Funktionen `f` und `gD` definiert.

(b) Schreiben Sie außerdem ein MATLAB-Programm `aufgabe04`, das nacheinander Triangulierungen lädt, die Lösung des Randwertproblems mit linearen und quadratischen finiten Elementen approximiert, die dafür benötigte Rechenzeit misst und jeweils den L^2-Fehler berechnet (wobei Integrale durch geeignete Quadraturformeln ersetzt werden). Zum Messen der Rechenzeit verwenden Sie `tic` und `toc`.

Ihr Programm nach Aufruf soll folgendes erstellen:

- Ordnungsplots für beide Verfahren: L^2-Fehler in Abhängigkeit von der maximalen Kantenlänge h in doppelt logarithmischen Achsen mit Referenzgeraden für Ordnung 2, 3, 4
- Work-Precision-Diagramme für beide Verfahren: L^2-Fehler der beiden Verfahren in Abhängigkeit von der Rechenzeit in doppelt logarithmischen Achsen

- Eine Tabelle mit fünf Spalten: h, L^2-Fehler für lineare Elemente, lokale Steigung, L^2-Fehler für quadratische Elemente, lokale Steigung. Mit “lokale Steigung” ist die lokale Steigung des Ordnungsplots gemeint. In den Zeilen der Tabelle sollen die Werte stehen, die Sie mit den verschiedenen Triangulierungen erhalten.

Wie man solche Tabellen erstellt wird anhand des folgenden Beispiels illustriert:

```
clc; clear all hidden;
fprintf('\n\n\n')
fprintf('x        exp(x)           1+x        1/(1-x)  \n\n')
formatSpec = '%4.2f     %e     %6.4f     %6.4f \n';
for k=0:5
    x=0.01*k;
    fprintf(formatSpec,x,exp(x),1+x,1/(1-x))
end
fprintf('\n\n\n')
```

Beachten Sie die in der Vorlesung gegebenen Hinweise zur Programmierung.