Übungen zur Vorlesung

Simulationswerkzeuge

Dr. S. Lang, D. Popović

Besprechung am 26. Mai 2009 in der Übung

Übung 10 DUNE-pdelab: Poisson-Gleichung

Nach unserem Ausflug in die *OpenCascade*-Welt wollen wir uns nun mit Simulationen befassen. Wir werden die Software *DUNE* und daraus im Wesentlichen das Modul *pdelab* kennenlernen.

Wir wollen die Poissongleichung

$$\begin{aligned} -\triangle u(\mathbf{x}) &= f(\mathbf{x}) \quad \text{in } \Omega \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \text{auf } \partial\Omega_D \\ \partial_{\mathbf{n}} u(\mathbf{x}) &= j(\mathbf{x}) \quad \text{auf } \partial\Omega_N \end{aligned}$$

in $2D$ lösen. Das Gebiet Ω sei offen, der Rand $\partial\Omega$ in einen Teil mit Dirichlet- und einen Teil mit Neumann-Randbedingungen aufgesplittet. Die „Poisson"-Gleichung beschreibt beispielsweise eine stationäre Wärmeverteilung bei bekannten Randbedingungen oder elektrostatische Potentiale.

In Ihrem Home-Verzeichnis ist ein Verzeichnis `dune-1.2` vorhanden. Sie finden dort das Modul `dune-pdelab` mit dem Unterverzeichnis `dune/pdelab/test` mit der Implementierung des Problems `testpoisson.cc`. Das `dune-pdelab`-Modul nimmt uns viel Arbeit ab, im speziellen die Implementierung der Basis und der Diskretisierung. Uns interessieren nun vor Allem die Parameterklassen `G,J` für den Rand und für die Quelle (`F`). Diese werden als Template-Parameter übergeben (*statischer Polymorphismus*!). Führen Sie zunächst in der Konsole die Befehle

```
$make testpoisson
$./testpoisson
```

aus. Dies löst die Poissongleichung und generiert Output-Files mit der Endung `vtu`. Diese können mit der Software *Paraview* visualisiert werden. Starten Sie dazu *Paraview* mit `paraview` und öffnen Sie die Ausgabe-Datei. Zu sehen ist eine Konturansicht der Lösung. Die Visualisierung kann durch verschiedene Filter gesteuert werden. Besonders interessant für uns sind die Filter *Warp to Scalar* und *Shrinking*. Der *Warp to scalar*-Filter hilft uns, den Bereich mit Neumann-Randbedingung und die Quelle zu lokalisieren.

Nachdem wir uns das Beispiel verdeutlich haben, wollen wir mit den Parameter-Klassen experimentieren. Versuchen Sie, Lösungen für folgende Probleme zu finden:

- $u = 2$ auf $\partial\Omega_D$, $\partial\Omega_N = \emptyset$, $f \equiv 0$ mit der trivialen Lösung $u \equiv 2$,
- Gemischte Randbedinungen:

  $$\begin{aligned} f(x,y) &= 50 \quad \text{für } 0.25 \le x, y \le 0.375 \\ j(x,y) &= -5 \quad \text{auf } \partial\Omega_N := \{(x,y) \in \partial\Omega;\ x = 1;\ 0.9 \le y \le 1;\} \\ g(x,y) &= 0 \quad \text{auf } \partial\Omega_D := \partial\Omega \backslash \partial\Omega_N, \end{aligned}$$

- $\partial\Omega_N = \emptyset$, $f \equiv 0$ und

  $$\begin{aligned} u(x,0) &= 0 && \text{für } 0 \le x \le 1 & \qquad u(0,y) &= \tfrac{y}{y^2+1} && \text{für } 0 \le y \le 1 \\ u(x,1) &= \tfrac{1}{(1+x)^2+1} && \text{für } 0 \le x \le 1 & \qquad u(1,y) &= \tfrac{y}{y^2+4} && \text{für } 0 \le y \le 1 \end{aligned}$$

  Die Lösung dieses Problems auf $\overline{\Omega}$ ist $u(x,y) = \frac{y}{y^2+(1+x^2)}$ und nimmt Ihr Maximum auf dem Rand an.

Hinweise:

Verwendung von DUNE auf dem eigenen Rechner: Wenn Sie über einen Rechner mit einem Unix-artigen Betriebssystem verfügen, können Sie diese Übungen auch auf dem eigenen Rechner durchführen. Zunächst sollten Sie mit Hilfe des Versionskontrollsystems *Subversion* folgende Versionen der DUNE-Module in ein Verzeichnis auf Ihren Rechner auschecken:

```
svn checkout https://svn.dune-project.org/svn/dune-common/releases/1.2 dune-common
svn checkout https://svn.dune-project.org/svn/dune-grid/releases/1.2 dune-grid
svn checkout https://svn.dune-project.org/svn/dune-istl/releases/1.2 dune-istl
svn checkout https://svn.dune-project.org/svn/dune-localfunctions/branches/1.2snapshot dune-
   localfunctions
svn checkout https://svn.dune-project.org/svn/dune-pdelab/branches/1.2snapshot dune-pdelab
```

Dann können Sie alles kompilieren mit dem Befehl

```
$./dune-common/bin/dunecontrol all
```

ausgeführt in dem Verzeichnis, in das Sie die Module kopiert haben. Informationen über vorausgesetzte Software, zusätzliche Module und weitere Installationshinweise finden Sie auf der DUNE-Homepage `http://www.dune-project.org/`.

Übung 11 DUNE-pdelab: Reentrant corner

Nun wollen wir das Beispiel mit der einspringenden Ecke aus der Vorlesung (Skript Kapitel 3, Finite Differenzen) untersuchen. Im Verzeichnis `dune-pdelab/dune/pdelab/test` finden Sie die Datei `testpoissonreentrantcorner.cc`, die das Problem implementiert. Wir lösen wiederum das Problem

$$\begin{aligned} -\triangle u(\mathbf{x}) &= f(\mathbf{x}) \quad \text{in } \Omega \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \text{auf } \partial\Omega_D \\ \partial_{\mathbf{n}} u(\mathbf{x}) &= j(\mathbf{x}) \quad \text{auf } \partial\Omega_N \end{aligned}$$

in $2D$, setzen nun allerdings $\Gamma_N = \emptyset$ und $f \equiv 0$, d.h. wir haben keine Quellen und keinen Fluss über den Rand. Dazu werden die Parameter-Klassen `G,J` für den Rand und die Quelle (`F`) entsrpechend angepasst. Sehen sie sich zunächst das Beispiel an mit

```
$make testpoissonreentrantcorner
$./testpoissonreentrantcorner
```

und verwenden Sie `paraview` zum Visualisieren der `.vtu`-Dateien.

Implementieren Sie anschließend das Testproblem aus der Vorlesung, d.h. erstellen Sie den Viertelkreis und stellen Sie die Dirichlet-Ränder wie in der Vorlesung gesehen ein. An der einspringenden Ecke sollte sich eine Singularität ausbilden. Betrachten Sie diese auch bei zunehmender Verfeinerung (Aufruf der Methode `globalRefine(...)`).