ÜBUNG 6 DIE PARALLELE RICHARDSON-ITERATION

Wir möchten das Gleichungssystem $Ax = b$ mit der Richardson-Iteration

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)})$$

lösen. Die Matrix A sei die Steifigkeitsmatrix einer P^1-Diskretisierung der Poissongleichung auf dem Einheitsquadrat. Dabei verwenden wir ein strukturiertes Dreiecksgitter mit $N = n^2$ Freiheitsgraden (Dabei bezeichnet n die Anzahl der Freiheitsgrade auf einer Horizontalen oder Vertikalen des Gitters).

Um die Berechnung zu beschleunigen, möchten wir die Iteration parallel durchführen. Dafür unterteilen wir das Einheitsquadrat in p kleinere Quadrate und verteilen die Freiheitsgrade in diesen Untergebieten auf p Prozessoren. (Dabei nehmen wir an, das p eine Quadratzahl ist; bei vier Prozessoren erhält beispielsweise jeder Prozessor $(n/2)^2$ Freiheitsgrade.) Die Indexmenge aller Freiheitsgrade bezeichnen wir mit I, die des Prozesses k mit I_k. Jeder Prozessor speichert die zu seinen Freiheitsgraden gehörigen Einträge von $x^{(k)}$ und die relevanten Zeilen von A.

Eine Iteration des parallelen Verfahrens besteht nun aus folgenden Schritten:

- Kommunikation der von den Nachbarprozessoren benötigten Einträge von $x^{(k)}$
- Berechnung von $x^{(k+1)}$

1. Beschreiben Sie die Indexmengen I_k und geben Sie an, mit welchen Prozessoren der Prozessor k welche Einträge von $x^{(k)}$ kommunizieren muss.

2. Die Rechenzeit für eine beliebige arithmetische Operation (Addition, Subtraktion oder Multiplikation) betrage t_{op}, die Zeit zur Übertragung eines Bytes an einen anderen Prozessor t_{byte}, und die Zeit, um eine Nachricht aufzusetzen, sei t_{msg}. Geben Sie eine Formel für die Gesamtrechenzeit für eine Iteration auf p Knoten an. Die Einträge von $x^{(k)}$ seien mit doppelter Genauigkeit gespeichert, so dass jeder Eintrag 8 Byte belegt. (Die Formel soll nur asymptotisch korrekt sein; beispielsweise muss nicht gesondert berücksichtigt werden, dass die Matrixzeilen zu Randknoten weniger Einträge haben.)

3. Geben Sie tabellarisch den Speedup des parallelen Verfahrens bei folgenden Parametern an:

$$\begin{aligned} t_{\text{op}} &= 2 \text{ ns} \\ t_{\text{byte}} &= 20 \text{ ns} \\ t_{\text{msg}} &= 5000 \text{ ns} \\ n &\in \{1024, 4096\} \\ p &\in \{1, 4, 16, 256, 4096\} \end{aligned}$$

12 Punkte

ÜBUNG 7 GEBIETZERLEGUNG

Sie $\Omega \subset \mathbb{R}^d$ ein Lipschitz-Gebiet (offen, zusammenhängend und beschränkt) und sei $f \in H^{-1}(\Omega)$ gegeben. Das homogene Poisson-Problem lautet

$$\begin{aligned} -\Delta u(x) &= f(x) \qquad \forall x \in \Omega \\ u(x) &= 0 \qquad \forall x \in \partial\Omega \end{aligned} \tag{1}$$

Sei Ω_1, Ω_2 eine Gebietzerlegung des Gebietes Ω:

$$\overline{\Omega} = \overline{\Omega_1 \cup \Omega_2}, \quad \Omega_1 \cap \Omega_2 = \varnothing, \quad \Gamma = \partial\Omega_1 \cap \partial\Omega_2, \quad \mu(\partial\Omega_i) > 0,$$

und $\partial\Omega_i$ sind Lipschitz-stetig. Dann ist das Problem (1) dem folgenden Problem (im schwachem Sinne) equivalent, falls f regulär genug ist:

$$\begin{aligned} -\Delta u_1(x) &= f(x) && \forall x \in \Omega_1 \\ u_1(x) &= 0 && \forall x \in \partial\Omega_1 \backslash \Gamma \\ u_1(x) &= u_2(x) && \forall x \in \Gamma \\ \frac{\partial u_1(x)}{\partial n_1} &= -\frac{\partial u_2(x)}{\partial n_2} && \forall x \in \Gamma \\ -\Delta u_2(x) &= f(x) && \forall x \in \Omega_2 \\ u_2(x) &= 0 && \forall x \in \partial\Omega_2 \backslash \Gamma \end{aligned} \tag{2}$$

Reicht die Regularität der Funktion f, d.h. $f \in H^{-1}(\Omega)$?
Hinweis:
Betrachten Sie das Poisson-Problem auf dem Gebiet $\Omega = (-1, 1)$

$$\begin{aligned} -\frac{d^2 u}{dx^2} &= -2\delta \\ u(-1) = u(1) &= 0 \end{aligned}$$

wo δ die Dirac-Delta-Funktion ist. Finden Sie die eindeutige schwache Lösung und untersuchen Sie die Bedingungen auf dem Übergang Γ in (2).

4 Punkte

ÜBUNG 8 PARALLELE BERECHNUNG DER L^2-NORM MIT DUNE

In dieser Aufgabe soll die Berechnung der L^2-Norm aus Aufgabe 1 parallelisiert werden. Dazu sollen Sie mehrere Prozesse über einen Teil des Gebietes integrieren und die Ergebnisse nachher aufsummieren. Die nötigen Teile des Interfaces wurden in der Übung vorgestellt. Zusätzlich ist die Klasse `CollectiveCommunication` nützlich – schauen Sie sich am besten die Online-Dokumentation dieser Klasse an. Den im Konstruktoraufruf benötigten MPI-Communicator erhalten Sie mit der `getCommunicator()`-Methode der `MPIHelper`-Instanz.

Um DUNE auf dem eigenen Rechner mit Unterstützung paralleler Programmierung zu kompilieren, müssen Sie eine MPI-Bibliothek installiert haben, beispielseweise OpenMPI. In allen Linux-Distributionen gibt es fertige Pakete dafür. Außerdem müssen alle Pakete mittels

```
dune-common/bin/dunecontrol --configure-opts="--enable-parallel" all
```

neu kompiliert werden. Auf den Rechnern im CIP-Pool ist DUNE bereits „parallel“ kompiliert.

4 Punkte