Universität Hamburg
Fachbereich Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, H. Lenhart, F. Hoffmann

Übungsblatt 4 zum Praktikum
Paralleles Programmieren für
Geowissenschaftler im SS 2015
Abgabe: 18.05.2015

Aufgabe 4: MPI-Matrix Operationen und sequentielle Lösung der Poissongleichung

In den folgenden Aufgaben sollen MPI-Matrix-Operationen geübt und die Poisson-Gleichung sequientiell gelöst werden.

Aufgabe 4A: MPI-Matrix Operationen mit Scatter und Gather (90 Punkte)

In der Aufgabe ist eine Integer-Matrix der Größe $24x24$ zu erstellen. Diese Integer-Matrix $M(24, 24)$ wird in absteigender Reihenfolge, d.h. von $M(1, 1) = 576$ bis $M(24, 24) = 1$, initialisiert.

Danach wird diese Matrix je nach Prozessoranzahl mittels dem Befahel scatter in gleich große Teilmatrizen aufgeteilt. Pro Teilmatrix wird die Operation $M(i, j) = M(i, j) * (ProzessorID + 1)$ ausgeführt. Nach dieser Berechnung wir von jeder Teilmatrix die Summe über alle Elemente jeder Teilmatrix errechent.

Anschließend werden mittels des MPI Befehls Gather die Teilmatrizen zu einer neuen Gesamtmatrix $M(24, 24)$ zusammengeführt. Auch für die Gesamtmatrix wir die Summe über alle Elemente errechent und diese Gesamtsumme mit der Summe verglichen, welche mittels dem MPI Befehl Reduce mit der Operation SUM als Gesamtsumme der Teilmatrizen ermittelt wird.

Dieses Programm soll für jeweils 4, 6 und 8 Prozessoren ausgeführt werden.

Aufgabe 4B + C: Lösung der Poissongleichung

Im Folgenden soll die Poisson-Gleichung sequientiell gelöst werden, indem zwei unterschiedliche Iterationsverfahren angewendet werden:

- Jacobi-Verfahren
- Gauß-Seidel-Verfahren.

Für beide Verfahren soll es möglich sein, entweder die Anzahl an Interationen vorzugeben die zu durchlaufen sind, oder eine Genauigkeit vorzugeben, nach deren Erreichen das Programm terminiert.

Die Poisson-Gleichung ist eine partielle Differentialgleichung der Form:

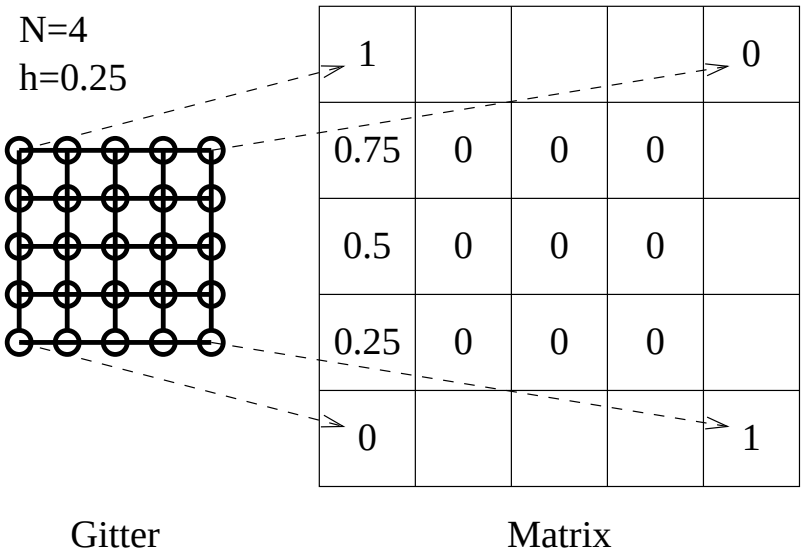
$$-u_{xx}(x, y) - u_{yy}(x, y) = f(x, y) \text{ mit } 0 < x, y < 1 \quad (1)$$

In diesen Aufgaben soll ohne die Berücksichtigung einer Störfunktion gerechnet werden, d.h. wir verwenden $f(x, y) = 0$.

Desweiteren werden die Randwerte vorgegeben mit:

- Randbelegung: $v(0,0) = v(N,N) = 1$ und $v(0,N) = v(N,0) = 0$.
  Alle Randwerte zwischen den Ecken werden linear interpoliert.
  Die inneren Werte der Lösungsmatrix werden auf Null gesetzt.

Die Initialisierung der Rändern ist hier für das Beispiel einer (N+1)x(N+1) Matrix mit N=4 illustriert, die Gitterweite $h := 1/N$ ist also 0.25:



Ein Überblick über die mathematischen Grundlagen ist im Anhang "Mathematische Grundlagen" gegeben.

Aufgabe 4B: Lösung der Poissongleichung nach Jakobi (180 Punkte)

Im Jakobi Verfahren werden zwei Matrizen verwendet, ähnlich der Vorgehensweise im Game of Life. Die Berechnung der neuen Werte erfolgt nur mit Hilfe der alten Werte.
Die Ausgabe erfolgt mit Hilfe von sogenannten Interlines mittels einer 9 x 9 Matrix. Die Berechnungs-Matrix *data* kann damit auf die Ausgabe-Matrix $outputMatrix$ abgebildet werden.

```
do i = 0,8
   do j =0,8
      outputMatrix(i,j) = data(i * (interlines+1),j * (interlines +1))
   enddo
enddo
```

Das heißt: N ist durch die Anzahl der Interlines definiert zu $N := 8 * (interlines + 1)$

Die Interlines bilden ausgewählte Gitterpunkte innerhalb der Matrix ab, mit deren Hilfe das Gesamtverhalten einfach analysiert werden kann. Daher soll zu jedem Lauf die Anfangsbelegung und der Endzustand der Matrix abgebildet werden.

Für beide Verfahren sind die Berechnungen in REAL*8 bzw. Double Precission unter Verwendung einer 185 X 185 Matrix (Interlines = 22) durchzuführen. Für die Berechnung mit vorgegebener Anzahl an Iterationen sollen 40 0000 Iterationen durchlaufen werden, während der Abbruch nach Genauigkeit bei einem Unterschreiten der Genauigkeit von 10^{-6} erfolgen soll.

Aufgabe 4C: Lösung der Poissongleichung mittels Gauß-Seidel-Verfahren (180 Punkte)

Das Gauß-Seidel Verfahren verwendet nur eine Matrix. Der aktuell berechnete Wert überschreibt den alten Wert. Für den nächsten Gitterpunkt wird dann sogleich dieser neue Werte als Nachbarpunkt mit verwendet. Für beide Berechnungen, nach Iterationen und Genauigkeit, sollen die gleichen Vorgaben wie beim obigen Jakobi Verfahren (Aufgabe 4B) verwendet werden.

Abgabe

Zu den Programmen zur Lösung der Poisson Gleichung soll ein Makefile erstellt werden, das bei Eingabe von "make" beide Programme kompiliert. Bei Eingabe von "make runjakobi" soll das Jakobi Vefahren und bei Eingabe von "make rungauss" das Gauß-Seidel Verfahren ausgeführt werden. Die Eingabe von "make clean" löscht alle kompilierten Teile, so dass nur der Quellcode übrig bleibt.

Die auf dem Cluster lauffähigen FORTRAN Programme sollen als Quellcode mit der Angabe der Gruppe (Personen in der Gruppe) bis zum **Montag (!) den 18.5.2015** geschickt werden an:

ppg-abgabe@wr.informatik.uni-hamburg.de

Bitte dabei folgende Form wählen:

1. bitte **NUR den Quellcode und das Makefile** schicken,
2. bitte für **jede Aufgabe ein separates Verzeichnis anlegen** und
3. alles **als komprimiertes Archiv .tgz oder zip** schicken! D.h. es soll wirklich nur **ein einzelnes Archiv** geschickt werden!

Als Subject im Kopf der Mail bitte die Angabe: PPG-15 Blatt4 und die Liste der Familiennamen der Personen in der Übungsgruppe.

Anhang: Mathematischer Hintergrund

Viele natürliche und technische Vorgänge lassen sich durch partielle Differentialgleichungen beschreiben. Ein Beispiel hierfür ist die Poisson-Gleichung. Mangels vorhandener analytischer Lösungsformeln muss man sich oft Methoden der numerischen Mathematik bedienen.

Hier gelangt man zunächst durch

(i) Diskretisierung (Festlegung der zu berechnenden Punkte im gewünschten Lösungsgebiet) und

(ii) Ersetzen der Differentialquotienten durch Differenzenquotienten

zu einem System von linearen Gleichungen. Für die Berechnung des Lösungsvektors dieses Systems existieren direkte und indirekte Verfahren. Wegen der Nachteile der direkten Verfahren (Eliminationsverfahren wie z. B. die Gauß-Elimination), nämlich hohe algorithmische Komplexität einerseits und numerische Instabilität andererseits, bevorzugt man heute indirekte Verfahren, zum Beispiel Iterationsverfahren, bei denen man sich iterativ bis zu einer gewünschten Genauigkeit der exakten Lösung annähern kann (sofern das Iterationsverfahren konvergiert). Zwei dieser Iterationsverfahren werden hier kurz vorgestellt.

A: Problemstellung

Gegeben ist eine partielle Differentialgleichung der Form

$$-u_{xx}(x,y) - u_{yy}(x,y) = f(x,y) \text{ mit } 0 < x, y < 1 \tag{2}$$

Diese Darstellung wird als **Poisson-Problem** bezeichnet. Dabei bezeichnet u_{ii} die zweite Ableitung der Funktion u nach i. Die Funktion $f(x,y)$ bezeichnet man als **Störfunktion**.

Die Eckpunkte der Randwerte $u(1,1)$, $u(1,N+1)$, $u(N+1,1)$ und $u(N+1,N+1)$ sind gegeben (siehe obige Abbildung). Die Werte auf dem Rand sollen dazuwischen interpoliert werden. Gesucht wird $u(x,y)$ für $0 < x, y < 1$.

B: Übergang von Differential- zu Differenzenquotienten

Wir definieren die inneren Gitterpunkte

$$u_{i,j} := u(i * h, j * h) \text{ mit } i, j = 2, ..., N \tag{3}$$

Die Ersetzung der partiellen Ableitungen aus (1) durch finite Differenzen zweiter Ordnung:

$$u_{xx;i,j} := 1/h^2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$

$$u_{yy;i,j} := 1/h^2(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})$$

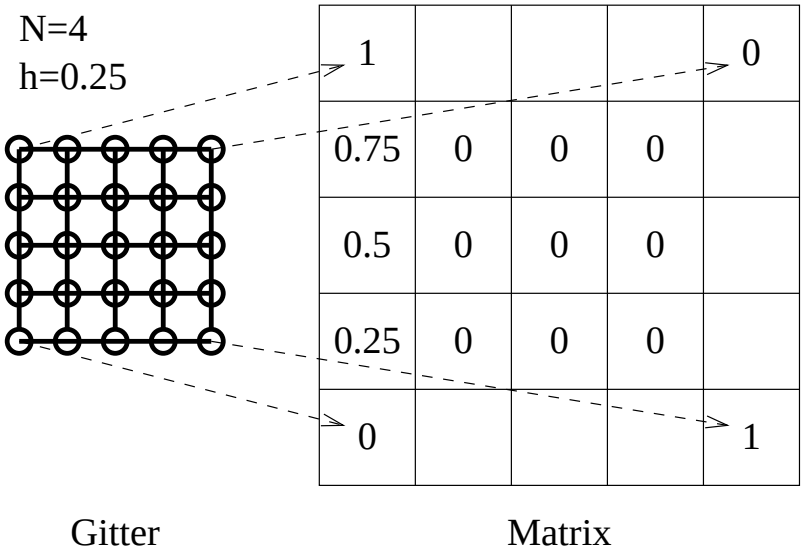
liefert ein **System von linearen Gleichungen** der Form:

$$1/h^2(4v(i,j) - v(i-1,j) - v(i+1,j) - v(i,j-1) - v(i,j+1)) = f(i,j) \tag{4}$$

C: Diskretisierung

Die Lösung der Poisson-Gleichung soll auf dem Gebiet $[0,1] \times [0,1]$ berechnet werden. Einfachste Diskretisierung ist ein **äquidistantes quadratisches Gitter**.

Anzahl Intervalle in jeder Richtung: N
Anzahl Punkte auf Gesamtgebiet (mit Rand): $(N+1)^2$
Anzahl innerer Punkte: $(N-1)^2$
Gitterweite h: $1/N$



Dieses Gitter kann in einer $(N+1) \times (N+1)$-Matrix gespeichert werden. Jeder Eintrag in der Matrix repräsentiert einen Punkt des Gitters. Die Randpunkte des Gitters werden vor Beginn der Berechnung vorbelegt.

D: Schema zum Lösen der Differentialgleichung

Das zentrale Element der Berechnung ist der sogenannte "Abtaststern". Der Stern wird wie folgt berechnet:

$$Star(i,j) = -v(i,j+1) - v(x-1,j) + 4 * v(i,j) - v(i+1,j) - v(i,j-1)$$

Danach wird der Korrekturwert berechnet. Die allgemeine Beschreibung hierzu für eine Variable $Corr$ lautet:

$$Corr(i,j) = (f(i,j) * h^2 - Star(i,j))/4$$

Mittels des berechneten Korrekturterms wird die neue Iteration t der Matrix $v_t(i,j)$, aus dem alten Wert $v_{t-1}(i,j)$ der vorherigen Iteration t-1 berechnet.

$$v_t(i,j) = v_{t-1}(i,j) + Corr(i,j)$$

Für die beiden Iterationsverfahren ist folgendes zu beachten:

Jacobi-Verfahren:
Verwendet zwei Matrizen für v: Die Berechnung der neuen Werte erfolgt nur mit Hilfe der alten Werte.

Gauß-Seidel-Verfahren:
Verwendet nur eine Matrix für v, d.h. neue Werte werden verwendet, sobald sie berechnet wurden und gehen damit gleich wieder in die Berechnung des folgenden Sterns ein.

E: Betrachtung zur Genauigkeit der Lösung

Darstellung von (4) in Matrixform: $Au = h^2 f$ (Herleitung und Aufbau der Matrix A soll für die Anwendung des Iterationsverfahrens hier nicht weiter betrachtet werden.)

Exakte Lösung: u
Näherung: v (soll iterativ berechnet werden, bis v nur noch minimal von u abweicht)
Fehler: $e = u - v$
Leider: u ist unbekannt, d.h. e ist nicht feststellbar.
Aber: berechenbar sind

- **Residuum** $r := h^2 f - Av$ (r ist der Betrag, um den die Näherung von v vom Originalproblem $Au = h^2 f$ entfernt ist)
- **Norm des Residuums** $\| r \|_\infty := \max | r(i,j) |$
- Zusammenhang: $\| r \|_\infty = 0 \Longleftrightarrow e = 0$

Aus $Au = h^2 f$ und $Av = h^2 f - r$ erhält man durch Subtraktion $Ae = r$, bzw. $e = A^{-1} r$, genannt **Residuumsgleichung**.

F: Pragmatische Vorgehensweise zum Abbruch nach Genauigkeit

An jedem Gitterpunkt wird die Differenz aus alten und neuen Werten berechnet.

$Diff(i,j) = v_t(i,j) - v_{t-1}(i,j)$

Unterschreitet diese Differenz $Diff$ an allen Punkten die geforderte Genauigkeit(z.B. 10^{-6}), wir das Programm abgebrochen.

Literatur (Begleitend und weiterführend)

- Stoer, Bulirsch: Einführung in die numerische Mathematik II, Heidelberger-Taschenbücher, Band 114, Springer
- William H. Press: Numerical Recipes in Pascal/C/Fortran, Cambridge, USA 1990