Parallelisierungsschema

1. Beschreibung der Datenaufteilung der Matrix auf die einzelnen Tasks

• Welche Daten der Matrix werden von welchem Task verwaltet?

Wir teilen die Matrix in n Submatritzen auf (n = Anzahl Prozesse). Jeder Prozess besitzt seinen eigenen Abschnitt. Eine Submatrix ist dann (Anzahl Zeilen/n) * (Anzahl Spalten) groß. Dabei muss ein Task mit anderen Tasks kommunizieren, da sie Teile des Sterns, die der andere Prozess benötigt, besitzen und weitergeben müssen. Ausnahmen sind dabei der erste und der letzte rechnende Prozess, da diese keinen Nachbarprozess an manchen Stellen besitzen.

• Visualisieren Sie die Datenaufteilung mit geeigneten Grafiken.

Thread 0					
Thread 1					
.					
.					
.					
Thread n					

2. Parallelisierungsschema für das Jacobi-Verfahren

• Beschreiben Sie aus Sicht eines Tasks, wann die Berechnung und wann die Kommunikation mit seinen Nachbarn erfolgt. Unterscheiden Sie nach gemeinsamem und verteiltem Speicher.
• Welche Daten benötigt der Task von seinen Nachbarn und wann tauscht er die Daten aus?
• Auf welche Variablen bzw. Daten muss welcher Task zugreifen?

Zunächst berechnet ein Task die Zeilen, für die er die nötigen Zeilen besitzt. Die Kommunikation mit seinen Nachbarn erfolgt, wenn er seine Randzeilen berechnen möchte.

Wenn Task n sich in der i-ten Iteration befindet, braucht er die Werte der “obersten” und “untersten” Zeile, die von Task n-1 und Task n+1 verwaltet werden, aus der i-1-ten Iteration. Hat er deren Werte erhalten kann er mit der Berechnung fortfahren. Dieses Prinzip gilt für jeden Task und nachdem diese Berechnungen abgeschlossen sind, kann die nächste Iteration gestartet werden.

Unterschiede zwischen shared und distributed memory:

Bei gemeinsamen genutzten Speicher stellt es kein Problem für einen Thread da, auf die Daten eines anderen Threads zu zugreifen, da sie sich einen Adressraum teilen.
Für die Kommunikation bei geteilten Speicher müssen sich die Threads über ein Netzwerk austauschen. Jeder Thread bekommt eine eigene Matrix auf der er arbeitet. Er berechnet die Zeilen für die er zuständig ist und übergibt seine Werte an das Netzwerk, welches die Matrix aktualisiert und die Ergebnisse an alle Threads kommuniziert (sobald es alle Aktualisierungen erhalten hat). Jeder Thread muss also warten bis das Netzwerk die Daten jedes Threads erhalten hat.

3. Parallelisierungsschema für das Gauß-Seidel-Verfahren (siehe Jacobi)
Damit man parallelisieren kann müssen die einzelnen Prozesse Zeitversetzt beginnen, da der 2. Prozess Werte vom 1. benötigt, der 3. Werte vom 2. usw.. Die Kommunikation erfolgt nur in eine Richtung. Die einzelnen Prozesse arbeiten solange bis alle die Abbruchbedingung erfüllen.

4. Diskussion der Abbruchproblematik
• Es sind vier Fälle zu betrachten: Abbruch nach Iterationszahl und Genauigkeit für jeweils Jacobi und Gauß-Seidel.
• Wann wird ein Task feststellen, dass das Abbruchkriterium erreicht wurde und er seine Arbeit beenden kann?
• In welcher Iteration bendet sich ein Task im Vergleich zu seinen Nachbarn, wenn er das Abbruchkriterium erreicht?

	Jacobi	Gauß-Seidel
Abbruch nach Iterationszahl	Jeder Prozess hat einen Iterator, wenn der Grenzwert erreicht ist (was bei allen Prozessen ungefähr gleichzeitig erfolgen sollte), geben alle Prozesse ihre Daten an den Master-Prozess.	Jeder Prozess hat einen Iterator, wenn der Grenzwert erreicht ist, dann ist die Iteration beendet. Wenn der Prozess durch ist, ist sein rechter Nachbar bereits seit einer Iteration beendet und sein linker Nachbar hat noch eine Iteration. Somit können die Daten an den linken Nachbarn weitergereicht werden.
Genauigkeit	Benötigt mehr Iterationen für Gleiche Genauigkeit.	Gauß-Seidel ist schneller (und genauer) als das Jacobi-Verfahren