1 Datenaufteilung der Matrix auf die einzelnen Prozesse

Nach der Initialisierung der Prozesse berechnet jeder einzelne Prozess wie viele Zeilen dieser aufnimmt. Dieses wird durch $N/size$ oder durch $N/size + 1$ (abhängig des Ranks) bestimmt. Anschließend erhält jeder Prozess seine Teilmatrix.
Am Ende der Mainfunktion wird die Gesamtmatrix ausgegeben, indem jeder Prozess außer 0 seine Teilmatrix an Prozess 0 sendet. Dabei printet Prozess 0 seine Teilmatrix vorher, empfängt einzeln die Teilmatrix von 1 bis N-1 und printet dieses in richtiger Reihenfolge.
Zusätzlich bekommt jeder Prozess zwei Zeilen z1 und z2 dazu. Dabei ist z1 die Zeile über der Teilmatrix und z2 die unter der Teilmatrix. Entsprechend wird z1 bei Prozess 0 und z2 bei Prozess size-1 nicht initialisert.

2 Jacobi-Verfahren

In calculate hat zunächst jede Matrix seine erforderlichen Zeilen. Am Ende der der Berechnungen werden alle Prozesse synchronisiert. Jeder Prozess sendet seine obere Zeile an den nächsten Prozess und der nächste Prozess empfängt dies in z1. Da es kein Sendezyklus gibt (Prozess 0 und Prozess size-1 sind nicht benachbart), kann kein Deadlock entstehen. Danach werden die Prozesse wieder synchronisiert. Anschließend führen wir entsprechend diese Prozedur mit z2 umgekehrt mit der ersten Zeile aus. Nach dem Empfang von z2 kann die nächste Iteration beginnen.

3 Gauß-Seidel-Verfahren

Da im Gauß-Seidel-Verfahren die aktuelle berechnete Zeile benötigt wird, setzen wir zunächst alle Prozesse 1 bis size-1 wartend auf den Prozess rank-1 mit den Empfangbuffer z1. Nur Prozess 0 darf hier starten. Nachdem Prozess 0 mit der Iteration fertig ist, schickt er seine letzte Zeile an den Prozess rank+1. Nun kann der Prozess 1 hier starten und sendet nach der Berechnung seine erste Zeile an Prozess 0. Prozess 0 erhält sein z2 und kann schon die nächste Iteration berechnen. Für alle andere Prozesse erfolgt das selbe.

4 Abbruchproblematik

Wir nehmen an, dass die Abbruchbedingung von der Anzahl X der Iteration abhängt. Bei Jacobi-Verfahren haben alle die gleiche Variable "term-iteration" und beenden ebenfalls Gleichzeit in der Iteration X.
Beim Gauß-Seidel-Verfahren ist das ähnlich, nur das Prozesse $[0, B]$ und $B < size - 1$ früher als die anderen Prozesse $[B + 1, size - 1]$ beenden. Damit beim printen nichts schief läuft, kann man eine Barriere nach calculate setzen.

Angenommen die Abbruchbedingung hängt von der Genauigkeit ab, so können wir beim Jacobi-Verfahren allen Prozesse die gleiche Variable "term-precision" übergeben. Nach jeder Iteration muss aber die Variable aktualisiert werden. Ist die Genauigkeit erreicht, terminieren alle Prozesse in der gleichen Iteration.
Im Gauß-Seidel-Verfahren erfolgt das Gleiche. Hat ein Prozess die Genauigkeit erreicht, so müssen

aber alle Prozesse in der Iteration beenden, wo sich die Prozesse $[B+1, size-1]$ befinden. Dies kann dazu führen, dass eine Iteration mehr berechnet wird. Glückerweise wäre die Genauigkeit gleich oder besser.