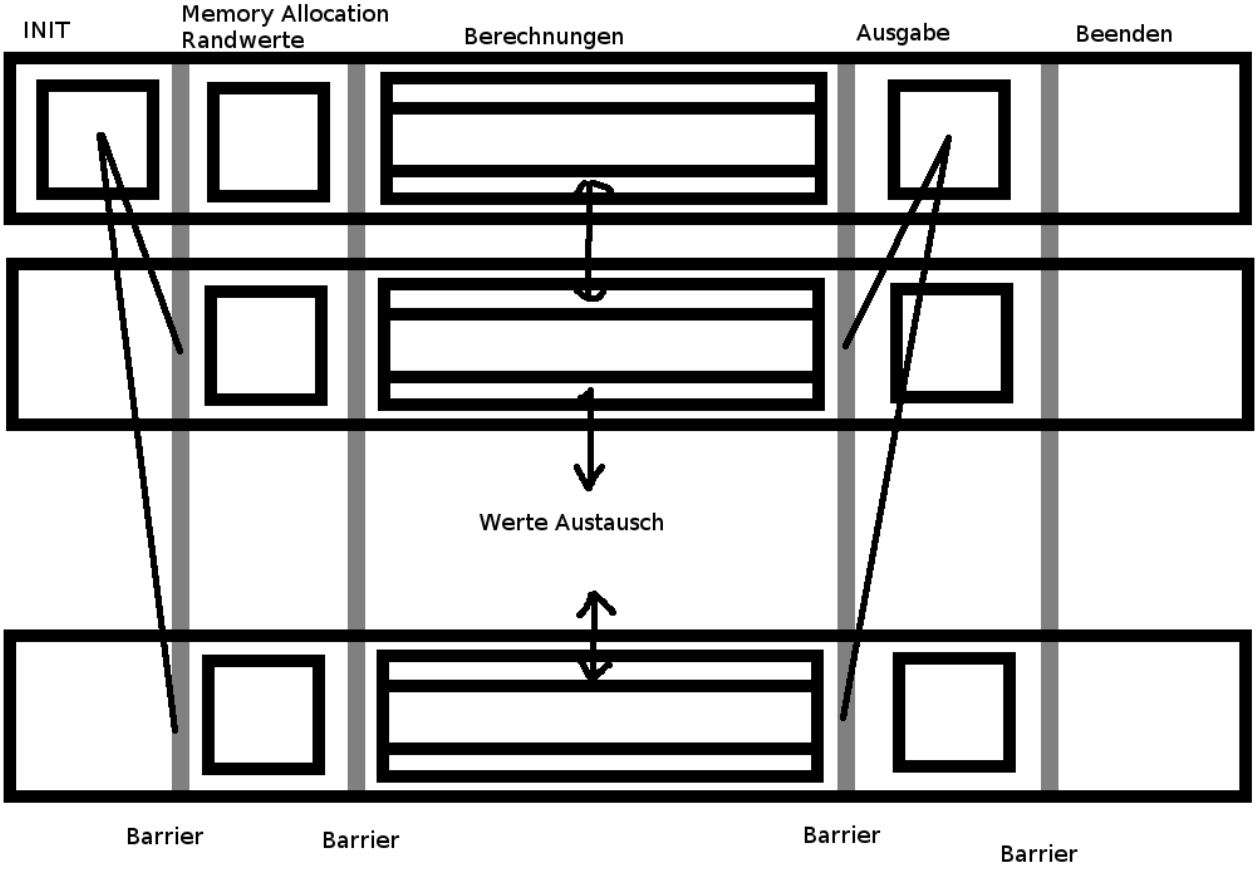
Datenverwaltung)

Der Prozess mit dem Rank 0 wird im folgenden auch Masterprozess bezeichnet. Dieser kümmert sich in der Init-Phase um die Datenaufteilung der anderen Prozesse, d.h. der Masterprozess teilt die Anzahl der Zeilen ein pro Prozess ein. Nach der Init-Phase folgt die Memory-Phase, bei der jeder Prozess seinen nötigen Speicher allokiert und die Randwerte zuweist. Bei der Speicher allokation ist auch zu beachten, dass jeder Prozess zwei weitere Zeilen allokiert, in der die Werte der anderen Prozesse verwaltet werden, eine Ausnahme hierbei ist der erste und der letzte Prozess, da diese nur von unten bzw. oben Werte empfangen können. Das gleiche Schema kann sowohl beim Jacobi- als auch beim Gauß-Seidel-Verfahren angewendet werden.

Jacobi-Verfahren)



Datenverwaltung Jacobi)

Siehe Datenverwaltung allgemein.

Iteration)

Innerhalb der ersten Iteration kann ein Prozess noch die Berechnungen durchführen ohne mit einem anderen Prozess (außer dem Masterprozess kommuniziert zuhaben). Dies hat den Grund, dass beim Jacobi-Verfahren eine Matrix mit den in der vorherigen Iteration berechneten Werten vorliegt und eine neue, in die die Werte der aktuellen Iteration geschrieben werden. Somit hat jeder Prozess zwei Matrizen, da die erste Matrix, aus der die erste Lösungsmatrix berechnet wird die Randwertmatrix ist, ist im ersten Iterationsschritt keine Komunikation mit anderen Prozessen notwendig, da die Randwertmatrix vom Prozess selbst erzeugt wurde. Diese hat jeder Prozess nach der Datenaufteilung für sich gesetzt und initialisiert.
Da nach jeder Iteration jeder Prozess auf die anderen Prozesse warten muss, befindet sich jeder Prozess in der gleichen Iteration. Somit muss dieser nach dem Warten die neu berechneten Werte am oberen und unteren Rand mit dem Prozess mit dem Rank kleiner eins und größer eins kommunizieren. Nach diesem Werteaustausch müssen alle Prozesse wieder warten, biss jeder Prozess diesen Werteaustausch vollzogen hat. Somit hat jeder Prozess wieder die berechnete Lösung der letzten Iteration.
Dies kann solange fortgesetzt werden, bis die Abbruchbedingung erreicht wurde (siehe Abbruchbedingung)). Danach sendet jeder Prozess seine Teilmatrix an den Masterprozess. Dieser gibt die empfangene Matrix aus oder schreibt sie in eine Datei, um sie danach wieder zu löschen bzw. mit einer neuen Teilmatrix zu überschreiben. So ist garantiert, dass maximal zwei Teilmatrizen im Speicher gehalten werden müssen und nicht die Matrix die die Lösung enthält.

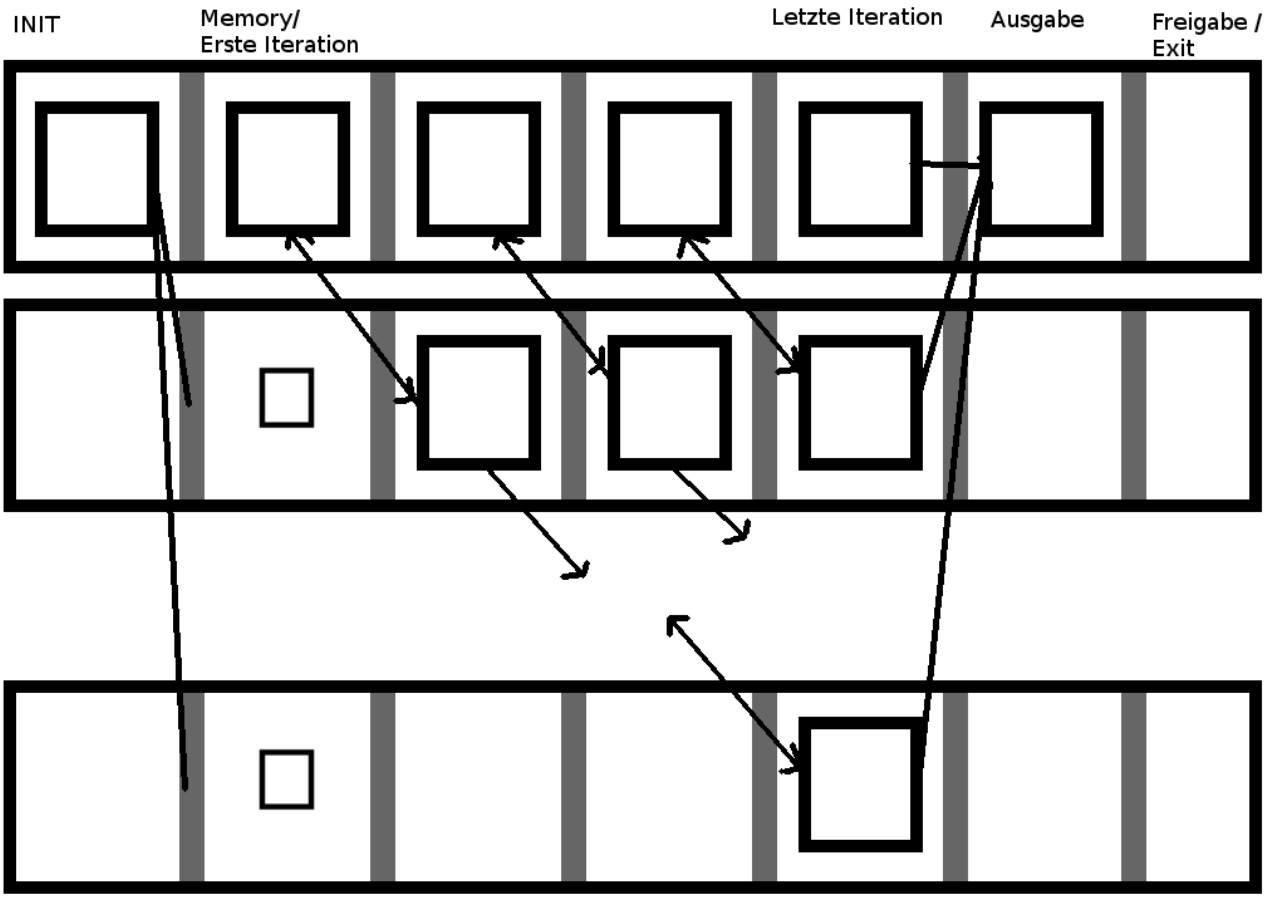
Abbruchbedingung)

Iteration)

Bei den Iterationen ist es nicht notwendig zu dass die Prozesse diese Abbruchbedingung untereinander kommunizieren, da jeder Prozess die gleiche Anzahl an Iterationen durchführen muss und somit alle Prozesse gleichzeitig beenden. Dannach wird die Teilmatrix wider an den Masterprozess zur Ausgabe gesandt.

Genauigkeit)

Beim Abbruch nach Genauigkeit, ist es im Gegensatz zum andern Abbruchkriterium zwingen notwendig, dass die Prozesse diese Untereinander kommunizieren. Dies ist aber nicht weiter Problematisch, da auch beim Genauigkeitskriterium sich alle Prozesse in der gleichen Iteration befinden. Somit muss nur der Prozess, der beenden möchte allen anderen Prozessen dies mitteilen, dies ließe sich mittels MPI_Allreduce realisieren, wobei hier nur der Abbruch an alle gesendet wird oder mittels MPI_Gather, welches aus dem Masterprozess aufgerufen wird und das maxresiduum vergleicht und darauf mit MPI_Bcast die Abbruchbedingung mitteilt. Hierbei beenden alle Prozesse erst ihre Berechnungen, nach dem sie sie für die aktuelle Matrix / Iteration fertig gestellt haben und senden diese dann zu Ausgabe an den Masterprozess.



Gauß-Seidel-Verfahren

Datenverwaltung Gauß-Seidel)

Siehe Datenverwaltung allgemein.

Iteration)

Da es beim Gauß-Seidel-Verfahren nicht möglich ist, dass sich jeder Prozess in der selben Iteration befindet, wie die anderen, muss ein Pipeline Verfahren verwendet werden, bei dem der erste Prozess dem n Prozess n - 1 Iterationen voraus läuft. Dies hat den einfachen Grund, dass beim Gauß-Seidel-Verfahren keine zwei Matrizen existieren, sondern die Lösung direkt auf der gegebenen Matrix berechnet wird. Aus dem Grund, dass durch das Pipeline-Verfahren nicht alle Prozesse gleichzeitig ab dem Start der Anwendung los rechnen können dauert dies bei m Prozessen mindesten m Iterationen, bis alle Prozesse effektiv rechnen können, dies tritt ebenfalls auf wenn das Abbruchkriterium erfüllt wurde, weil hier einige Prozesse noch ihre Berechnungen

beenden müssen. Somit ist je nach Matrix Größe und Anzahl der Prozesse mit einem Zeitlichen Verlust zu rechnen, bei dem der Großteil der Prozesse auf eine Eingabe vom vorherigen Prozess wartet. Beim beenden des Programms werden dann alle Daten zur Archivierung bzw. Ausgabe an den Masterprozess gesandt.

Abbruchbedingung)

Iteration)

Dies ist ähnlich zum Jacobi-Verfahren, nach der Berechnung der Iterationen wartet der Prozess einfach auf die Anderen Prozesse und gibt noch ein letztes mal seine Matrix an den nächsten Prozess weiter, damit dieser dann in der nächsten Iteration seine Berechnungen beenden kann. Dies läuft solange, bis alle Prozesse ihre Iterationen beendet haben, erst danach wird die Teilmatrix an den Masterprozess zur Ausgabe übergeben.

Genauigkeit)

Die Abbruch nach Genauigkeit ist nur schwer machbar, aus dem Grund, dass wenn ein Prozess feststellt, dass die Genauigkeit entsprechend gut ist und dieser Prozess nicht der Masterprozess ist, so befinden sich alle anderen Prozesse in höheren Iterationen als der Prozess, der das Abbruchkriterium festgestellt hat. Damit ist die ist die Ausgabe nicht möglich, da sich die Matrizen unterscheiden und somit falsche Ergebnisse liefern würden.
Dies Problem lässt sich zwar durch merken der Vorherigen Matrizen der Vorherigen Iterationen merken somit liegt aber im ersten Prozess die Matrix so oft vor, wie es Prozesse gibt. Sollte ein Prozess nun auf das Abbruch Kriterium treffen so würde dieser seine aktuelle Iteration an die anderen Prozesse senden und diese dann entweder die Berechnungen bis zu der Anzahl der Iterationen zu Ende führen oder die bereits berechnete Matrix heraussuchen. Dieses Verfahren hat aber dass Problem, dass es sehr Speicherlastgig werden kann und Berechnungen vorkommen, die im Nachhinein verworfen werden und somit das Verfahren nicht sehr Energieeffizient ist.
Als Alternative kann man nach dem ein Prozess die Abbruchbedingung festgestellt hat die Pipeline auslaufen lassen, hierbei sind allerdings die Ergebnisse verfälscht und entsprechen nicht der aktuellen Iteration des Prozesses, der die Abbruchbedingung festgestellt hat.