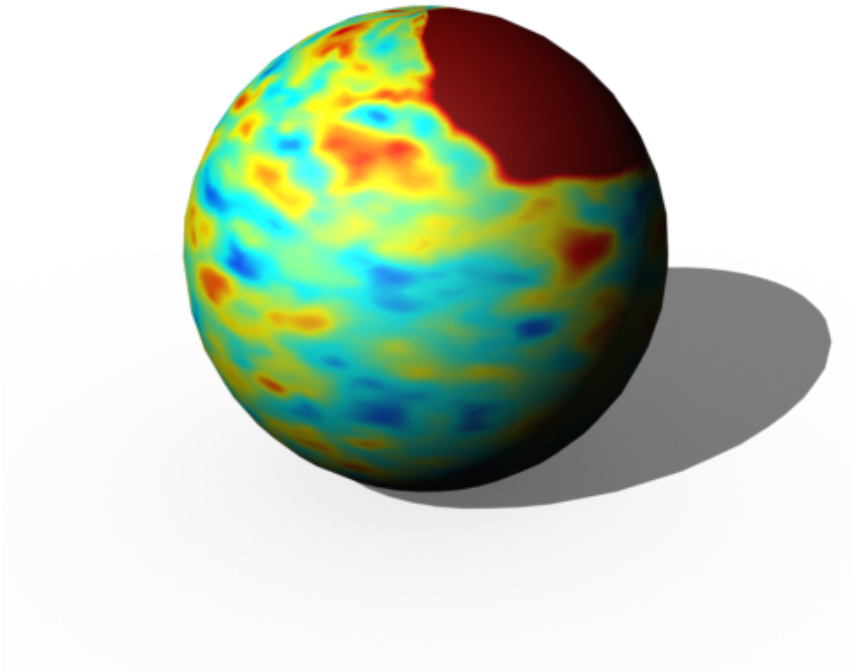
Implementierung und Paralellisierung von Daisyworld

Parallele Programmierung (DKRZ)

von

Max Hänze, Christoph Bockhahn

15. Oktober 2012



Zusammenfassung

Es wurde eine zweidimensionale Variante des "Daisyworld" Modells im Rahmen des Praktikums "Parallele Programmierung" umgesetzt. "Daisyworld" beschreibt einen fiktiven Planet, welcher von Gänseblümchen bevölkert wird. Die Bevölkerung der Gänseblümchen reagiert auf eine Änderung der Sonneneinstrahlung mit einer Anpassung der Albedo. Die mittlere Temperatur auf dem Planten bleibt durch diese Anpassung über einen großen Bereich der Sonneneinstrahlung konstant. Die Dynamik wird durch die Lösung einer Differentialgleichung und eines stochastischen Prozesses über die Zeit generiert.
Das Modell wurde sowohl sequentiell, wie auch parallel, mit Hilfe von MPI implementiert. Die verwendete Programmiersprache ist C.

Inhaltsverzeichnis

1 Einleitung

1.1 Beschreibung des Problems in 0D

Bei "Daisyworld" handelt es sich um einen fiktiven Planeten, welcher nur von schwarzen und weißen Gänseblümchen bevölkert wird. Die schwarzen und weißen Gänseblümchen unterscheiden sich in ihrer Albedo (Rückstrahlvermögen). Der Planet erfährt eine örtlich konstante und mit der Zeit zunehmende Einstrahlrate.
Das Modell veranschaulicht eine Theorie, nach der sich ein biologisches System in gewissem Maße selbst regulieren kann. In "Daisyworld" bleibt die auf dem Planeten herrschende Temperatur bis zu einem kritischen Bereich der Sonneneinstrahlung konstant. Das Verhältnis der schwarzen und weißen Gänseblümchen passt sich so an, dass sich eine für das Wachstum günstige Temperatur einstellt.

2 Erweiterung auf 2D

2.1 Mathematische Grundlagen

Das 0D-Modell wird erweitert, indem eine lokale Abhängigkeit der Temperatur eingeführt wird. Im Vergleich zum 0D-Modell gibt es anstatt der drei Temperaturen (weiße/schwarze Daisies und Boden) nun eine lokale Temperatur $T = T(x, y)$. Zur Beschreibung des zweidimensionalen Raums wird ein Gitter mit X- und Y-Koordinaten verwendet. Nicht nur die Temperatur, sondern auch die Albedo wird damit von den Gitter-Koordinaten abhängig: $A = A(x, y)$.
Der fiktive Planet besteht aus einem zweidimensionalen Gitter aus Ortspunkten. Jeder Gitterpunkt enthält eine Information über die Temperatur, die Albedo und eine Variable die festlegt ob der Gitterpunkt mit Daisys bevölkert ist.

```
1 \\ Gitterzelle
2 typedef struct
3 {
4     double T; // Temperatur
5     double A; // Albedo
6     int alive; // 0 = dead, 1 = alive
7 } WorldCell;
```

Listing 1: Struct für eine Gitter-Zelle

Die Temperatur zwischen den Gitterpunkten ist durch eine Diffusions-Gleichung gekoppelt, welche mit einem numerischen Verfahren gelöst werden muss:

$$C\frac{\partial T(x,y,t)}{\partial t} = D_T\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)T(x,y,t) - \sigma_B T(x,y,t)^4 + S(1 - A(x,y,t)) \qquad (2.1)$$

Die Lösung der Differentialgleichung wurde zunächst mit Hilfe des Operator-Splitting-Verfahrens implementiert. Diese Methode wurde aufgrund einer Inkompatibilität bei der Parallelisierung verworfen. Es erwies sich als günstig die sogenannte FTCS (Forward-Time Central-Space) Methode zu verwenden.

Die Albedos der Gitterpunkte sind durch ein stochastisches Verfahren miteinander gekoppelt. Dieses Verfahren ist in Abbildung 1 dargestellt. Es ist in zwei Abläufe unterteilt.

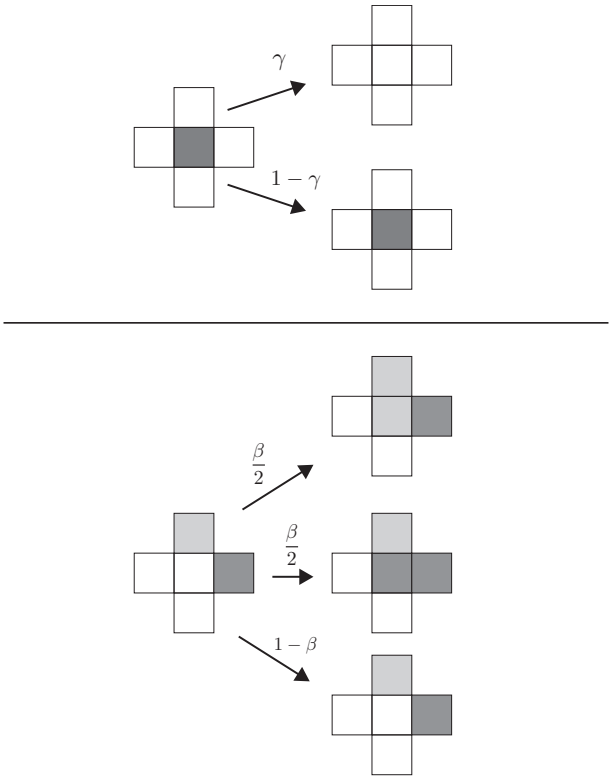


Abbildung 1: Albedo Wachstum

Gänseblümchen einer bevölkerten Zelle sterben mit einer gewissen Wahrscheinlichkeit γ, der Sterblichkeitsrate. Gänseblümchen können sich zudem von einer bevölkerten, auf eine unbevölkerte Zelle vermehren. Die neu bewachsene Zelle besitzt dann, bis auf einen geringen zufälligen Mutationswert, die Albedo der Nachbarzelle. Die Wahrscheinlichkeit, mit welcher diese Vermehrung stattfindet, hängt von der Wachstumsrate ab. Die Wachstumsrate β ist parabolisch von der Temperatur abhängig. Sie besitzt ihr Maximum bei

22,5°C. Für Temperaturen unter 5°C bzw. über 40°C ist kein Wachstum mehr möglich. Im Modell werden die Temperaturen in Kelvin berechnet.

$$\beta(T_i) = \begin{cases} \frac{4}{(40-5)^2}(T_i - 5)(40 - T), & \text{falls } 5 < T_i < 40 \\ 0, & \text{sonst} \end{cases} \tag{2.2}$$

Die Implementation der Wachstums-Funktion ist in Listing 2 gezeigt.

```
1  \\ Wachstumsrate
2  double beta (double T) {
3          static const double c = 4.0/35.0/35.0;
4          double result = c * (T-278)*(313-T);
5          if (result > 0) return result;
6          else return 0;
7  }
```

Listing 2: Funktion der Wachstumsrate

Die Funktion $\beta(T)$, sowie die Werte für γ und die Mutation entstammen [1].

Das hier verwendete Modell wurde von W. von Bloh [1] vorgeschlagen. Die numerische Lösung dieses Modells wurde von uns selbstständig erarbeitet und implementiert.

[1][1]: Von Bloh,W., Block, A. and Schellnhuber, H. J. Self stabilization of the biosphere under global change: a tutorial geophysiological approach. Tellus 49B, 249–262 (1997)

3 Design und Implementierung

3.1 Operator Splitting

3.1.1 Probleme bei der Parallelisierung

Das Operator Splitting ist ein einfaches Verfahren zum Lösen von partiellen Differentialgleichungen. Der größte Nachteil dieses Verfahrens ist die Tatsache, dass bei dem hier vorliegenden Fall, für die Lösung einer Zelle, die Werte der gesamten Reihe und Zeile benötigt werden. Zur Parallelisierung können die Zellen nun zeilen- oder spaltenweise auf die Prozessoren verteilt werden. In beiden Fällen ist es jedoch nötig, dass zur Lösung einer kompletten Zeile oder Spalte alle Zellen der gesamten Welt zur Verfügung stehen.

Bei dem Operator-Splitting müssen zwischen den Prozessoren also eine ganze Reihe von Informationen ausgetauscht werden. Dies ist für eine Parallelisierung unvorteilhaft. Aus diesem Grund wurde das Operator-Splitting verworfen und das FTCS-Verfahren verwendet. Dieses hat den Vorteil, dass für die Lösung einer Zelle nur die umgebenden Zellen (vier Stück) benötigt werden.

3.2 FTCS

Die Berechnung der Temperatur für jeden Zeitschritt setzt die Lösung der bereits beschriebenen Differentialgleichung voraus. Diese wird zunächst diskretisiert. Aus den infinitesimalen Änderungen werden diskrete Änderungen. Die Änderung im Ort wird durch die aktuellen Werte der nächsten Nachbarn einer Zelle bestimmt. Außerdem fließt der aktuelle Wert der Albedo mit ein. Abbildung 2 zeigt die diskretisierte Form der Differentialgleichung. Zur Berechnung von T^{n+1} muss die gezeigte Gleichung noch entsprechend umgestellt werden.

$$C\frac{T^{n+1}_{mid} - T^{n}_{mid}}{\Delta t} = D_T\left(\frac{T^{n}_{bot} + T^{n}_{top} - 2T^{n}_{mid}}{\Delta y^2} + \frac{T^{n}_{right} + T^{n}_{left} - 2T^{n}_{mid}}{\Delta x^2}\right) - \sigma_B {T^{n}_{mid}}^4 + S(1 - A^{n}_{mid})$$

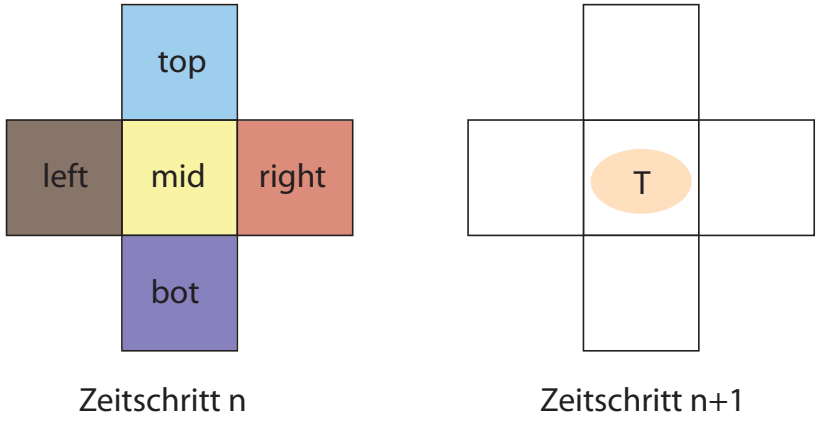


Abbildung 2: Albedo Wachstum

3.2.1 Implementierung

Um das FTCS-Verfahren zu parallelisieren, werden die vorhandenen Zellen zeilenweise auf die verschiedenen Prozessoren aufgeteilt. Da für die Berechnung nach dem FTCS-Schema nicht nur die zu berechnenden Zellen, sondern auch die nächsten Nachbarn benötigt werden, müssen diese ebenfalls zur Verfügung stehen (siehe Abbildung 3). Ebenso werden für die Berechnung der Albedo die nächsten Nachbarn benötigt. Die Kommunikation sieht also für beide Prozessschritte gleich aus.

Ebenfalls in Abbildung 3 dargestellt ist die Kommunikation, die nach jedem Zeitschritt stattfinden muss. Ein großer Vorteil ist die Tatsache, dass jeder Prozess nur mit den angrenzenden Prozessen kommunizieren muss. Der erste Prozess P0 kommunziert in jedem Fall also nur mit dem zweiten Prozess P1 und dem letzten Prozess (in diesem Beispiel P3). Diese Kommunikation ist ausreichend, um die gesamte Berechnung des Systems auszuführen.

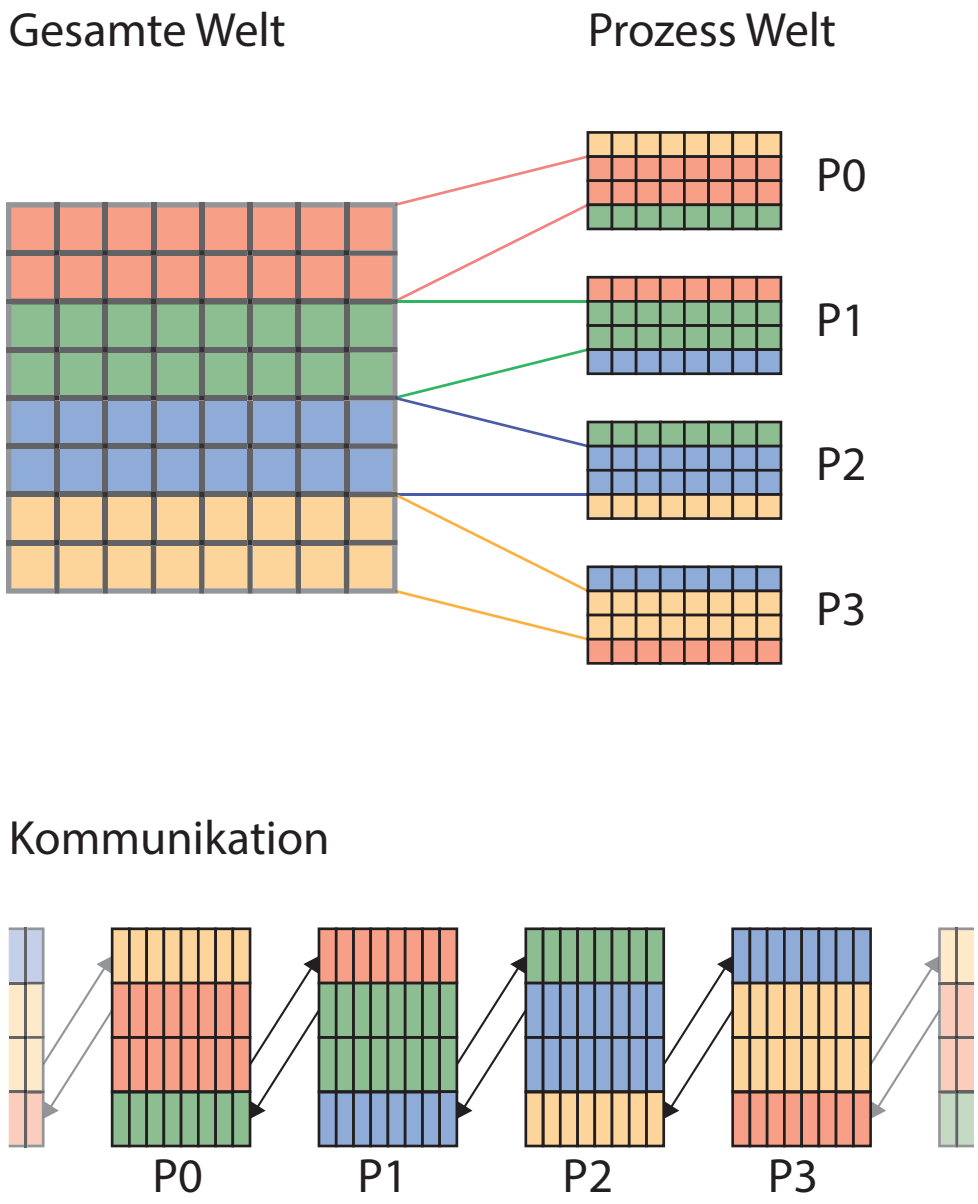


Abbildung 3: FTCS Kommunikation

Lediglich zum Abspeichern von Daten, ist eine weitergehende Kommunikation nötig. Dabei sammelt ein Master-Prozess die Daten von allen Subprozessen ein und schreibt diese zur späteren Auswertung in eine Binär-Datei.

Der gesamte Ablauf ist in Abbildung 4 dargestellt.

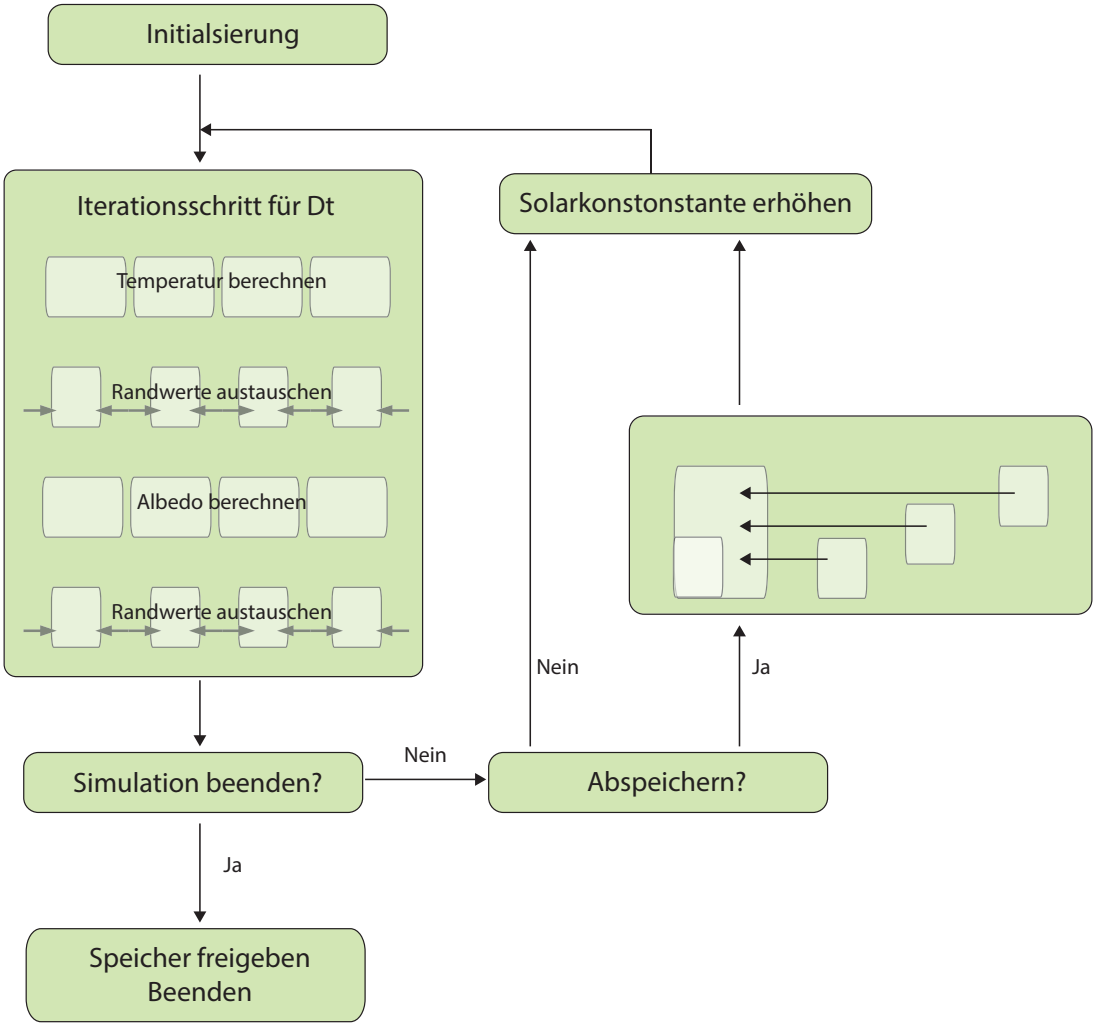


Abbildung 4: Flowchart Daisyworld

3.2.2 Leistungsanalyse

Abbildung 5 zeigt die Laufzeit der FTCS-Variante bei verschiedener Anzahl verfügbarer Prozessoren.

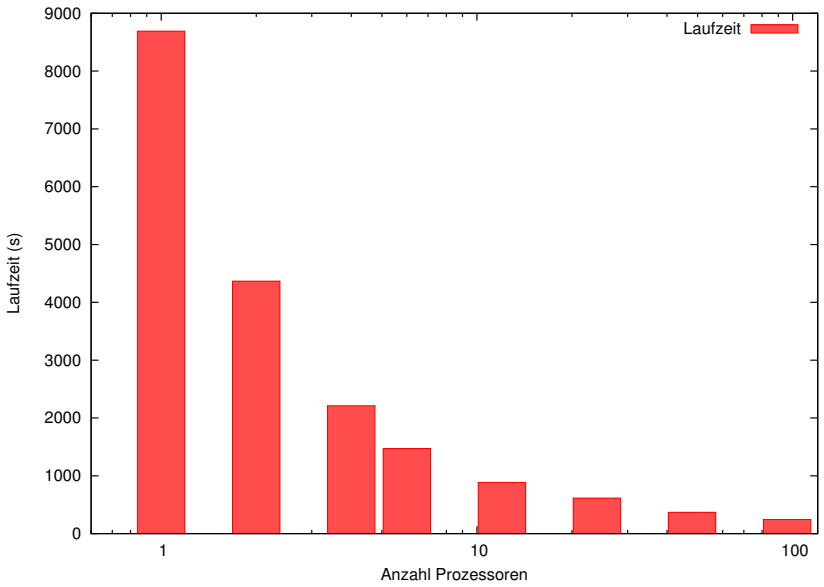


Abbildung 5: Laufzeit der FTCS Version

Abbildung 6 zeigt den Speedup der parallelen Version im Vergleich zum superlinearen Speedup.

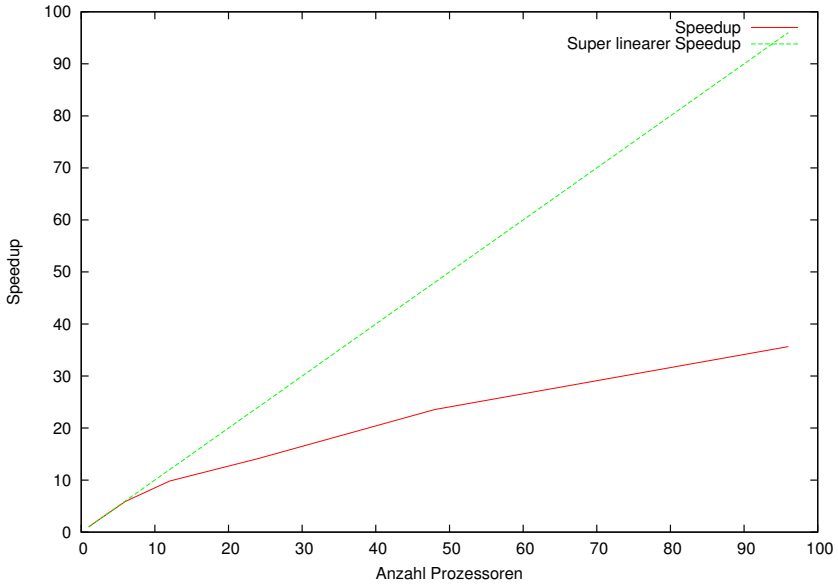


Abbildung 6: Speedup der FTCS Version

Die Messung zeigt, dass durch die Parallelisierung ein deutlicher Geschwindigkeitszuwachs erreicht werden konnte. Bis zu einer Prozesseranzahl von sechs Prozessoren verläuft der Speedup noch annäherend linear, danach beginnt der Geschwindigkeitszuwachs pro Prozessor geringer zu werden.

Anzahl Prozessoren	Laufzeit (s)	Geschwindigkeits-Zuwachs (Faktor)
1	8691.375	1
2	4367.516	1.990
4	2212.045	3.929
6	1470.236	5.911
12	884.963	9.821
24	615.477	14.121
48	368.940	23.557
96	243.781	35.652

Tabelle 1: Laufzeit FTCS

3.3 Ergebnisse der Berechnung

3.3.1 Berechnete Größen

Das Programm berechnet die zwei Kenngrößen Temperatur und Albedo für jede einzelne Zelle. Der Output wird dabei nicht für jeden Zeitschritt gespeichert, sondern wurde für die darsgestellten Ergebnisse alle 223 Schritte als Binärdatei abgespeichert.

Die Auswertung der Binärdateien wird von einem zweiten Programm, welches nicht parallelisiert wurde, übernommen. Mittels der C++ Bibliothek Magick++ werden die Binärdaten der Welt in eine Grafik überführt. Mittels ImageJ und Adobe After Effects werden aus den Bildern Videos erzeugt.

3.3.2 Selbstregulierung des Systems

Eine bemerkenswerte Eigenschaft des System ist die Selbstregulierung, die das System erfährt. Obwohl die Sonneneinstrahlung konstant erhöht wird, ist zunächst keine Temperatur-Erhöhung zu erkennen. Ohne die Daisies wäre ein linearer Anstieg zu erwarten. Dadurch, dass die einzelnen Zellen ihre Albedo anpassen, kann sich das System selbst regulieren und auf eine ideale Temperatur einstellen.

Diese Verhalten ist in Abbbildung 7 dargestellt. Es ist sehr gut zu erkennen, dass die Temperatur konstant bleibt, während sich die normierte Solarkonstante S verdoppelt. Das System kann sich jedoch nur in gewissen Maße selber regulieren und so beginnt das System ab einer gewissen Sonneneinstrahlung zu kollabieren. Die Daisies sterben ab und die Temperatur steigt sprungartig an. Im weiteren Verlauf kommt es zu einem linearen Anstieg der Temperatur, wie er ohne Daisies erwartet wird.

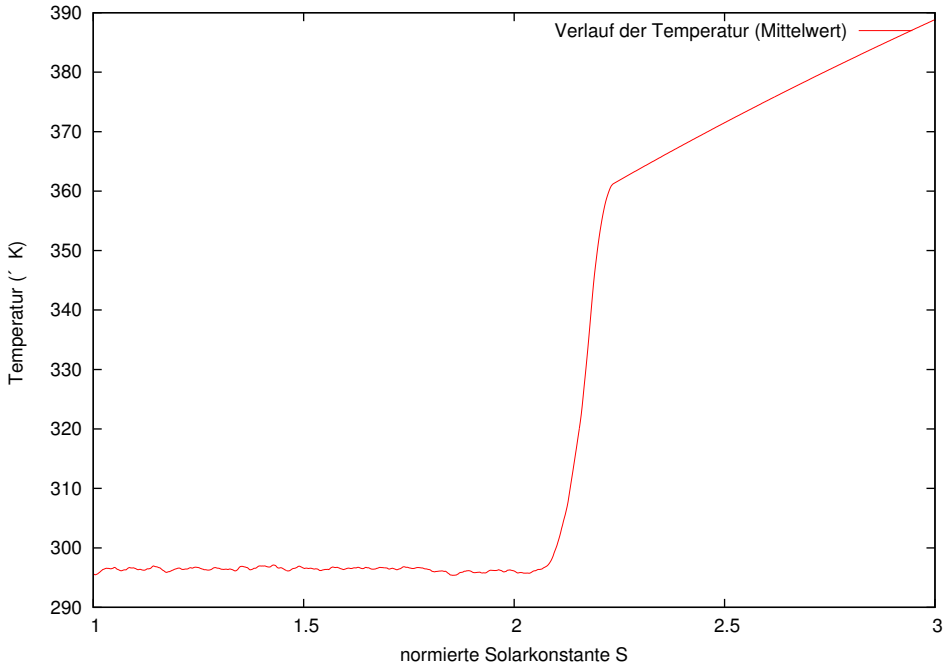


Abbildung 7: Verlauf der Temperatur

Die Albedo (siehe Abbildung 8) steigt im Zeitverlauf immer weiter an, um die erhöhte Sonneneinstrahlung zu kompensieren. Kollabiert das System und sterben die Daisies ab, bleibt die Albedo konstant bei 0.5, da dies der Wert für die Abstrahlung des Bodens, ohne Daisies, ist.

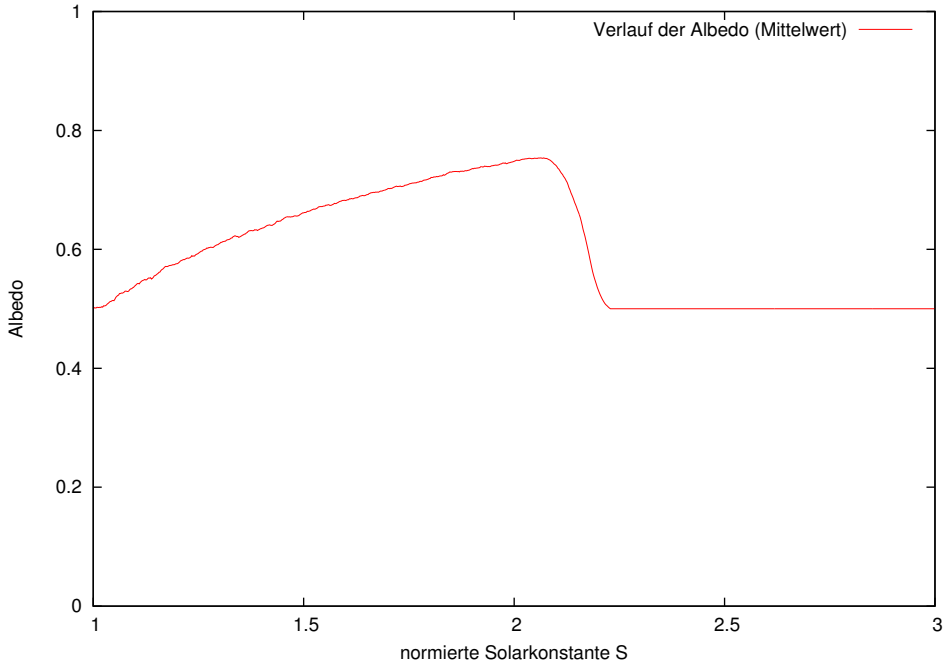


Abbildung 8: Verlauf der Albedo

0 Der Verlauf der Temperatur und der Albedo ist in Abbildung 9 grafisch dargestellt. Das jeweils dritte Bild der Reihe stellt das kollabierende System dar. Es entspricht einem Wert von $S = 2.11$.

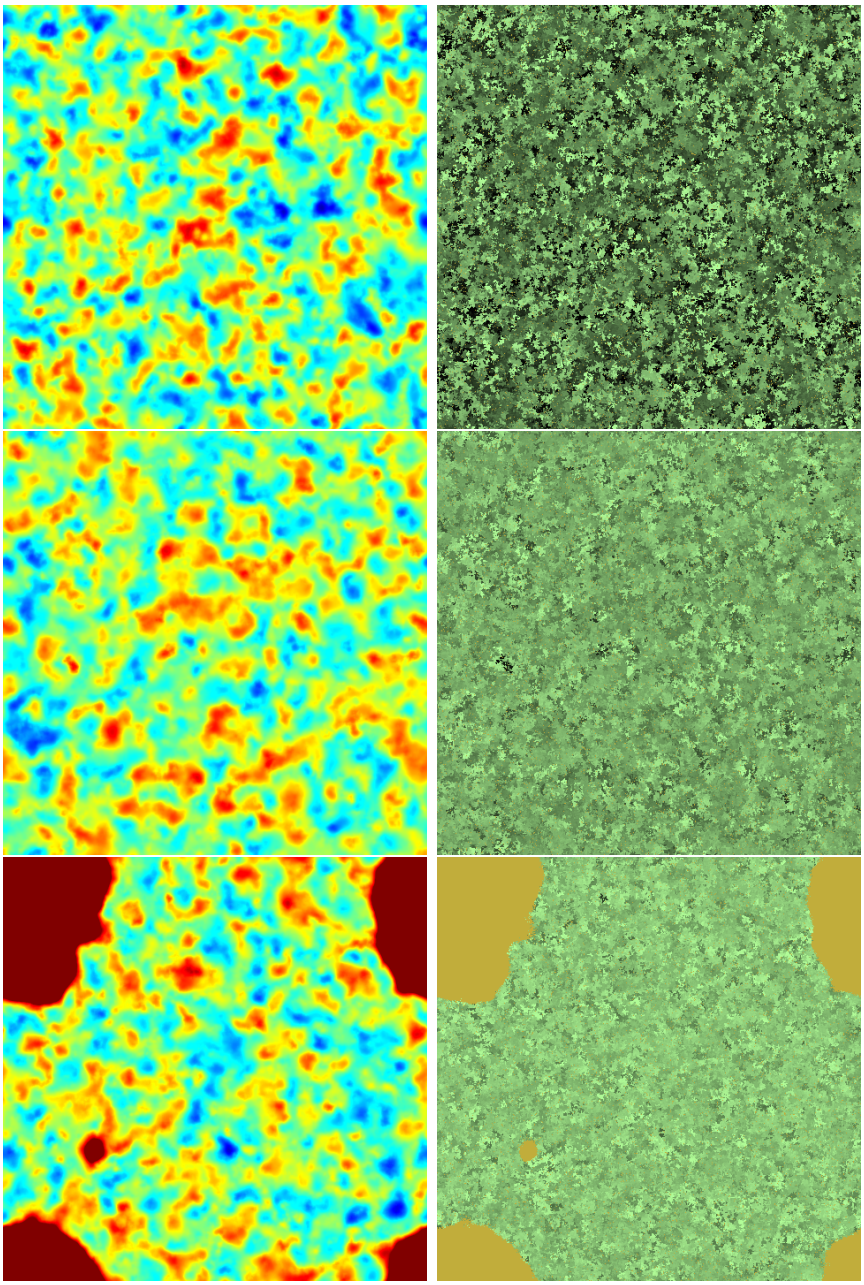


Abbildung 9: Temperatur (Heatmap) und Albedo

4 Zusammenfassung

Die sequentielle Version des Daisyworld Programms konnte in diesem Projekt erfolgreich parallelisiert werden. Zur Lösung der Differentialgleichung wurde zunächst das Operator-Splitting verfahren implementiert. Da dieses jedoch für eine Parallelisierung nicht besonders geeignet ist, wurde dieses verworfen und das FTCS (Forward Time Central Space) Verfahren verwendet.

Dieses Verfahren benötigt zur Lösung der Differentialgleichung für eine Zelle lediglich die vier umliegenden Nachbarn. So konnte eine Aufteilung der gesamten Welt auf die verschiedenen Prozessoren zeilenweise erfolgen. Kommunikation ist nur in angrenzenden Prozessen nötig, wobei jeder Prozess eine Zeile senden und eine Zeile empfangen muss.

Durch die Parallelisierung ergibt sich ein Speedup, der bis zu einer Prozessoranzahl von sechs Prozessoren annähernd linear verläuft. Ab zwölf Prozessoren fängt der Speedup an, abzuflachen. Der Zeitgewinn pro zusätzlichem Prozessor wird geringer.

5 Appendix A

Tabelle 2 listet die Werte auf, welche in dem Modell verwendet werden.

Bezeichnung	Wert
Anzahl Zellen	480x480
γ	0.02
DT	500.0
C	2500.0
A_0	0.5
Δ X	1.0
Δ Y	1.0
Δ Z	1.0
Mutattionsrate	0.01

Tabelle 2: Werte