

141.241 Projektarbeit Grundlagen der Supraleitung

Untersuchung von Flussliniengittern im Supraleiter mittels numerischer Simulation

Bachelorarbeit für das Studium Technische Physik

von Thomas Hausmaninger
betreut von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Franz M. Sauerzopf

Wien, Juni 2010

Inhaltsverzeichnis

1 Einleitung **3**

2 Physikalische Grundlagen **4**
2.1 Typ-II Supraleiter 4
2.2 Fluss Linien und ihr Gitter 5
2.3 Flusslinien Verankerung 6

3 Das Programm **7**
3.1 Übersicht, Funktion und Ziel des Programms 7
3.2 Aufbau des Programms 7
3.2.1 Die vortex_field Klasse 7
3.2.2 Die Parameter für das Feld 8
3.2.3 Berechnung von Energie und Wechselwirkung 9
3.2.4 Die Relax Methode 10
3.3 Maßnahmen zur Performance Optimierung 10
3.4 Tabellarisierung und Genauigkeit 11

4 Ergebnisse **13**
4.1 Übersicht der Simulationen 13
4.1.1 Laufzeit 13
4.2 Ergebnisse und Beobachtungen 14
4.2.1 Störung eines geordneten Feldes 14
4.2.2 Störung eines ungeordneten Feldes 15
4.3 Interpretation 15

5 Zusammenfassung **26**

1 Einleitung

Ziel der Arbeit war es ein bereits vorhandenes C++ Programm zur Simulation eines Flusslinien Gitters zu optimieren, um einige Funktionen zu erweitern und anschließend einige Testläufe laufen zu lassen.

Die Simulation von Flussliniengittern ist interessant, da diese nur sehr schwierig sichtbar gemacht werden können und daher ihr Aufbau noch nicht im Detail bekannt ist. Im Speziellen geht es dabei um Bildung von Defekten im Flussliniengitter.

Die Ergebnisse der Simulationen dieses Programms können dabei nicht als quantitative Aussagen betrachtet werden, da die Zahl der simulierten Flusslinien zu klein ist und die Randbedingungen einen zu starken Einfluss haben.

2 Physikalische Grundlagen[1]

2.1 Typ-II Supraleiter

Wenn ein Supraleiter (SL) (genauer Typ-I Supraleiter) auf eine Temperatur kleiner der kritischen Temperatur T_c abgekühlt wird, so wird er supraleitend. Dieser reine supraleitende Zustand, auch *Meissner-Phase* genannt, zeichnet sich dadurch aus, dass der elektrische Widerstand $\rho(T)$ verschwindet und der magnetische Fluss eines externen Feldes, aufgrund von ausgleichenden Kreisströmen an der Oberfläche des Supraleiters, vollkommen aus dem SL verdrängt wird. Dabei ist die supraleitende Phase nicht nur durch eine kritische Temperatur, sondern auch durch ein (maximales) kritisches externes am Supraleiter angelegtes Feld $B_c(T)$ und auch eine kritischen Stromdichte $J_c(T)$ begrenzt.

Im Gegensatz zu Typ-I Supraleiter beobachtet man bei Typ-II Supraleitern, dass zwischen normalleitender und supraleitender Phase noch ein gemischter Zustand, *Shubnikov-Phase* genannt, besteht. Dabei dringt das Feld in einzelnen *Flusslinien* (auch Flussschläuche oder kurz FL genannt) in den Supraleiter ein. Diese Phase wird erst bei Anlegen eines äußeren Magnetfeldes bemerkbar und liegt zwischen den Feldern $B_{c1}(T)$ und $B_{c2}(T)$. Dabei befindet sich der SL für ein angelegtes Feld $B_a < B_{c1}(T)$in der Meissner-Phase und für $B_{c1}(T) < B_a < B_{c2}(T)$ in der Shubnikov-Phase (sonst in der normalleitenden Phase). Für $T \to T_c$ verschwinden beide kritischen Felder.

Ein wichtiger Parameter für die Charakterisierung eines Supraleiters ist der *Ginzburg-Landau Parameter* κ (GL Parameter). Dieser setzt sich zusammen aus dem FL-Radius λ (bzw. der Eindringtiefe des B-Feldes am Rand eines Typ-I SL) und der Kohärenzlänge ξ die den Bereich um die Flusslinie (bzw. bei Typ-I SL Bereich am Rand des SL) angibt, innerhalb dessen keine supraleitende Phase ist. Genauer gesagt ist ξ ungefähr der Radius des Bereiches in dem der Ordnungsparameter Ψ für Supraleitung $\Psi = 0$ und nicht $\Psi = 1$ ist. Diese Übergangslängen entstehen dadurch, dass das B-Feld bzw. die Dichte der für die SL verantwortlichen Cooperpaare am Übergang nicht abrupt auf 0 springen kann. Der GL Parameter ist definiert als:

[1] Ernst Helmut Brandt, The flux-line lattice in superconductors, Review June 1995, Rep. Prog. Phys. 58 (1995) 1465-1594

$$\kappa = \frac{\lambda}{\xi}$$

Es gilt, dass es nur für Supraleiter mit $\kappa > 1/\sqrt{2}$ eine *Shubnikov-Phase* gibt bzw. der Supraleiter ein Typ-II SL ist. Für Typ-I SL gilt dann $\kappa < 1/\sqrt{2}$.

2.2 Flusslinien und ihr Gitter

Der Ordnungsparameter $\|\Psi(r)\|^2$ und das mikroskopische Feld $B(r)$ für eine isolierte Flusslinie sind wie folgt gegeben:

$$\|\Psi(r)\|^2 \approx \frac{1}{1 + 2\xi^2/r^2} \tag{2.1}$$

$$B(r) \approx \frac{\Phi_0}{2\pi\lambda^2} K_0[(r^2 + 2\xi^2)^{1/2}/\lambda] \tag{2.2}$$

Wobei K_0 eine modifizierte Bessel Funktion ist und $\Phi_0 = h/2e$ die Quantisierung der Flussfeldlinien aufgrund der periodischen Randbedingung (supraleitender Ring um FL) ist.

In der Shubnikov-Phase bilden die Flusslinien ein Dreiecksgitter aus, da sich die Flusslinien gegenseitig abstoßen. Dabei kann die Wechselwirkungsenergie des Feldes F_{int} (Summe über alle Wechselwirkungen zwischen den einzelnen FL) bei parallelen FL gut wie folgt approximiert werden:

$$F_{int} = \frac{\Phi_0}{2\pi\lambda^n\mu_0} \sum_i \sum_{j>i} \left[K_0\left(\frac{\|r_i - r_j\|}{\lambda'}\right) - K_0\left(\frac{\|r_i - r_j\|}{\xi'}\right) \right] \tag{2.3}$$

Wobei $\lambda' \approx \lambda/\sqrt{1-b}$ und $\xi' \approx \xi/\sqrt{1-b}$ mit $b = B/B_{c2}$ die effektive Eindringtiefe und effektive Koheränzlänge sind und B das mittlere Feld im SL ist. Der erste Term in (2.3) ist die abstoßende magnetische Wechselwirkung (überlappende FL Felder) und der zweite Term ist die Anziehung aufgrund der Kondensationsenergie (Überlappung der Flusslinienkerne). Für die Simulation wurde die London-Näherung verwendet, welche den zweiten Term vernachlässigt, da der Abstand zwischen den FL in den meisten Fällen zu groß ist und praktisch keine Überlappung der Kerne vorhanden ist.

Wenn das außen angelegte Feld B_a größer wird, dringen immer mehr Flusslinien in den SL ein und das durchschnittliche B Feld $B = \Phi_0 \cdot n$ (n ist die FL Dichte) wächst.

Bei $B_a = B_{c2}(= B)$ überlappen die Flusslinienkerne und die Supraleitung verschwindet zusammen mit Ψ.

Das Flusslinien Gitter kann für kleine Verformungen durch lineare Elastizitätstheorie beschrieben werden. Es kommt dabei heraus, dass das Schermodul verschwindet, wenn $k = 1/\sqrt{2}$ (alle Flusslinien Anordnungen haben die selbe Energie), wenn $\lambda \to \infty$ (Feld der Flussschläuche überlappt stark) oder wenn $B \to B_{c2}$ (Die FL Dichte ist so hoch, dass ihre Kerne stark überlappen).

2.3 Flusslinien Verankerung

In realen Supraleitern gibt es Inhomogenitäten z.B Fehlstellen im Material. Flusslinien werden von diesen Inhomogenitäten angezogen und bei genügend schwachen Stromdichten $J < J_c$ können diese Flusslinien als verankert betrachtet werden.

Die meisten Verankerungskräfte wirken dabei auf den FL-Kern. Verursacht werden kann die Verankerung durch räumliche Schwankungen von $T_c(r)$ oder von der mittleren freien Weglänge der Elektronen welche mit Fluktuationen des Ordnungsparameters $\|\Psi(r)\|^2$ zusammenhängen.

Diese Verankerung ist notwendig, damit ein Supraleiter einen Null-Widerstand hat. Wenn keine FL verankert sind, dann können sich die FL bei einem Strom aufgrund der Lorentzkraft bewegen. Diese Bewegung führt zu einem Energieverbrauch und somit zu einem Widerstand.

Weitere Eigenschaften die durch das Verankern von Flusslinien erklärt werden können sind zum Beispiel:

- Die Strom-Spannungs-Kennlinie des Supraleiters ist nicht linear sondern hat einen sehr komplexen Verlauf.
- Die Magnetisierungs Kurve $M(H_a)$ ist irreversibel und bildet eine Hystereseschleife.
- Verankerte Flusslinen üben auch eine Kraft auf das atomare Gitter aus, was zu magnetomechanischen Effekten und einer Veränderung der Geschwindigkeit und der Dämpfung von Ultraschall führt.

3 Das Programm

3.1 Übersicht, Funktion und Ziel des Programms

Ziel des Programms ist es das Verhalten eines Flussliniengitters mit parallelen Flusslinien (kurz FL) als zweidimensionales Feld zu simulieren. Dabei sollen die Auswirkung von Störungen und das Bilden oder Auflösen von Versetzungen untersucht werden können.

Kern des Programms ist eine "Relax Methode", welche schrittweise jede Flusslinie in Richtung Energieminimum bewegt und so die Energie des gesamten Feldes minimiert. Dabei wird die Wechselwirkung zwischen den Flusslinien durch die in Abschnitt 2 beschriebene Bessel K_0 Funktion festgelegt.

Außerdem sind Methoden implementiert, die das Feld mit Flusslinien füllen, Störungen einbauen, verankerte Flusslinien hinzufügen sowie andere grundlegende Ein/Ausgabe-Methoden.

3.2 Aufbau des Programms

Ein Feld mit Flusslinien wird durch ein Objekt der Klasse `vortex_field` repräsentiert.

3.2.1 Die vortex_field Klasse

Wichtige verwendete `Public` Methoden sind:

`vortex_field(...)`: Der Konstruktor der Klasse legt ein Gitter einer bestimmten Größe und Flusslinienanordnung an. Details siehe Abschnitt 3.2.2

`int xy_save(string filnam, int prec)`: speichert unter dem Dateinamen `filnam` mit der Genauigkeit `prec` für jede Flusslinie folgende Daten:

- x-Koordinate
- y-Koordinate
- Kraft auf die FL in x-Richtung
- Kraft auf die FL in y-Richtung
- Gesamtkraft

`int save(string filnam)`: speichert den aktuellen Inhalt der Arrays (Positionen,Wechselwirkungen etc.) der Klasse in eine Binärdatei.

`int read(string filnam)`: liest Zustand der Klasse aus einer mit `save()` gespeicherten Binärdatei aus. ACHTUNG: Objekt der Klasse muss mit denselben Parametern im Konstruktor erstellt worden sein, mit denen auch das gespeicherte Objekt erstellt wurde. Es werden keine Größenanpassungen der Felder vorgenommen.

`pinn_rand(const int pinn_num)`: wählt eine Anzahl (`pinn_num`) von FL zufällig aus und markiert sie als verankert. Diese FL werden beim Relaxieren nicht mehr bewegt (Theorie siehe Abschnitt 2.3).

`pullout(const int mode=0)`: Bei `mode=0` wird aus dem mittleren Bereich des Feldes mit x-Koordinaten zwischen $\pm$`a0` die Flusslinie mit der maximalen Energie ausgewählt und an den Punkt am Rand mit der minimalen Energie gesetzt. Bei `mode=1` wird genauso vorgegangen, jedoch wird die FL aus dem selben mittleren Bereich zufällig ausgewählt.

`int relax(double force)`: Mit dieser Methode wird ein Relaxationsvorgang gestartet. Der Parameter `force` gibt die minimale Kraft an, bis zu der relaxiert wird. Details siehe Abschnitt 3.2.4.

`int relax_step(double force)`: Führt einen Relaxationsschritt aus, bei dem jede Flusslinie in Richtung Energieminimum bewegt wird. Gibt `-1` zurück, wenn keine Feldlinien bewegt wurden. Ansonsten die Zahl der gescheiterten Versuche (von 0 bis maximal 9). Details siehe Abschnitt 3.2.4.

3.2.2 Die Parameter für das Feld

Die Parameter für das `vortex_field` Objekt werden dem Konstruktor übergeben, der so aussieht: `vortex_field(int number, double xlen, double ylen, double a0, double h0, double xfrac, double yfrac);`

Der erste Parameter `number` gibt die Art des Feldes an:

- < 0 erzeugt ein Feld mit zufälliger Positionierung der Flusslinien, wobei die Zahl und Dichte der Flusslinien ist abhängig von Parametern `a0`, `h0`, `xlen`, `ylen` ist.
- $= 0$ erzeugt ein regelmäßiges Feld ansonsten wie `number<0`.
- > 0 erzeugt ein regelmäßiges Feld, die Zahl der Flusslinien ist gleich dem Parameter `number`, der Abstand ist festgelegt durch `xlen` und `ylen` (`a0=0`).

`xlen, ylen`: Geben für `number<=0` die Länge und Breite des Feldes an. Die Zahl `num` der Feldlinien wird über `a0` wie folgt berechnet (wobei $r_h00 \approx 0.866$):

$$num = \frac{(xlen \cdot ylen)}{(a_0^2 \cdot r_h00)} + 10^{-5}$$

`a0`: Entspricht beim regelmäßigen Gitter dem doppelten Feldlininenabstand in y-Richtung. Wenn `a0==0` wird es im Konstruktor wie folgt berechnet:

$$a_0 = \sqrt{\frac{xlen \cdot ylen}{number \cdot r_h00}}$$

`h0`: Entspricht beim regelmäßigen Gitter dem doppelten Feldlinienabstand in x-Richtung. Wenn `h0==0` ist, dann wird es im Konstruktor wie folgt berechnet:

$$h_0 = r_h00 \cdot a_0$$

Es wird davon ausgegangen, dass die Größen als reduzierte Längen in Einheiten von λ übergeben werden. Da jedoch die Simulation vermutlich ohnehin keine quantitative Aussage bringen kann, wird im Weiteren nicht darauf eingegangen.

3.2.3 Berechnung von Energie und Wechselwirkung

Zum Berechnen der Energie einer FL wird die `n_analyze()` Methode verwendet. `n_analyze()` berechnet die Energie der FL in ihrer aktuellen Position und die Energie, wenn sie um die Parameter `xstep` bzw. `ystep` nach links/rechts bzw. unten/oben verschoben wird.

Dabei wird die für alle fünf Fälle die Wechselwirkung mit allen anderen Flusslinien aufsummiert.

Das Berechnen der Wechselwirkung zwischen zwei Flusslinien übernimmt `calcWWdirect()` welche die modifizierte Besselfunktion K_0 verwendet um die Wechselwirkung aus dem auf λ normierten Abstand zu berechnen. (Details siehe Abschnitt 2.2)

3.2.4 Die Relax Methode

Zum Relaxieren des Feldes enthält die `vortex_field` Klasse die `relax()` und die `relax_step()` Methode. Die `relax()` Methode ruft die `relax_step()` Methode auf, bis keine Feldlinien mehr bewegt werden oder bis das Relaxieren keine energetische Verbesserung mehr bringt (`relax_step()` gibt dann `9` zurück).

Im ersten Schritt geht `relax_step` alle Flusslinien einzeln durch und berechnet mittels `n_analyze()` die Gesamtenergie und die Richtung und Stärke der Kraft, die auf die FL wirkt, und speichert sie, aber nur wenn die Kraft größer als der Parameter `force` ist. Die Kraft wird noch mit der `privat` Variable `fac` multipliziert.

Wenn keine Feldlinie mehr bewegt wird, gibt die Methode `-1` zurück. Ansonsten werden jetzt die Punkte verschoben.

Nun wird noch überprüft ob es sinnvoll ist, den Faktor `fac` zu erhöhen oder zu verkleinern: Wenn die Gesamtenergie des Feldes durch Verschieben der FL größer wurde, wird `fac` halbiert und auch die Feldlinien um den halben bewegten Weg zurückgenommen. `fac` wird so lange halbiert, bis das Bewegen der Feldlinien eine kleiner Gesamtenergie für das Feld bringt maximal jedoch 9 mal. Die Zahl dieser „Fehlversuche“ wird dann von `relax_step` zurückgegeben. Wenn die Gesamtenergie des Feldes durch das Bewegen kleiner wurde wird überprüft, ob eine Verdopplung von `fac` vorteilhaft ist.

3.3 Maßnahmen zur Performance Optimierung

Das Programm wendet den Großteil der Rechenzeit dafür auf, die Wechselwirkung zwischen zwei Flusslinien zu berechnen (bei 4680 FL etwas mehr als die Hälfte der Rechenzeit, bei größeren Feldern verhältnismäßig deutlich mehr). Um auch Mulitcore Prozessoren ausnützen zu können, wird mittels der OpenMP Library in `n_analyze()` das Berechnen der Wechselwirkungen parallelisiert.

Die Ergebnisse der Wechselwirkung zwischen zwei Flusslinien mittels `n_analyze()` wird in einem Array gespeichert. Dadurch muss die Wechselwirkung nur dann neu berechnet werden, wenn sich die Position einer der beiden Flusslinien geändert hat.

Die Bessel K_0 Funktion wurde tabellarisiert. Genauer genommen wurde nicht nur die Funktion $BesselK_0(x)$ tabellarisiert, sondern die Funktion $BesselK_0(\sqrt{x})$, da immer die Besselfunktion bezüglich der Abstände in der Ebene ($\sqrt{x^2+y^2}$) gesucht ist und das berechnen von Wurzeln zeitaufwändig ist. Dabei wurde für das Runden die Funktion `long lrint(double x)` aus der Bibliothek `math.h` verwendet, da sie etwas schneller läuft als

die Standardfunktion `round()`, die auch angewendet wird, wenn man einem integer oder long einen double zuweist[1]. In diesem Fall benötigt das Berechnen aller Wechselwirkungen (mit Aufsummieren usw.) bei einer minimalen Schrittweite der Tabellarisierung von 10^{-6} unter Verwendung der `lrint()` Funktion um ca. 8% weniger CPU Rechenzeit als mit der `round()` Methode. Die für das Runden benötigte Zeit hängt natürlich stark von der Genauigkeit der Tabellarisierung bzw. der zu rundenden Zahlen ab. Das Umstellen des „Rounding Modes“ auf „round torward zero“ (abrunden für positive und aufrunden für negative Zahlen) mittels `fesetround()` Funktion, um die Überprüfung zu sparen, ob ab- oder aufgerundet wurde, brachte keinen messbaren Geschwindigkeitsvorteil.

Um den Speicherbedarf für größere Felder nicht zu groß werden zu lassen (bei ca. 4860 FL benötigt das Programm ca. 215MB RAM, bei 9198 FL ca. 833MB RAM) wurden die Wechselwirkungen in `float` und nicht in `double` Genauigkeit gespeichert. Das Umwandeln zwischen `float` und `double` Variablen führt zu keinem messbaren Performance Verlust.

3.4 Tabellarisierung und Genauigkeit

Die Tabellarisierung der Besselfunktion bzw. der Funktion $BesselK_0(\sqrt{x})$ wurde in drei Abschnitte mit unterschiedlicher Schrittweite eingeteilt. Zwischen den Stützpunkten wird linear approximiert. Da der Fehler wie die Steigung der Besselfunktion für große Abstände stark abnimmt, ist es vor allem wichtig für kleine x kleine Schrittweiten zu nehmen.

Die verwendete Schrittweite ist in Tabelle 3.1 dargestellt. Der Verlauf des maximalen Fehlers der linearen Approximation $g(x)$ zwischen zwei Stützpunkten für verschiedene Schrittweiten ist in Abbildung 3.1 zu sehen. Dabei ist der Fehler $error(x)$ definiert als $error(x) = \max(g(xh) - f(xh))$, wobei $xh \in [x, x + h]$ ist.

[1] Mehr Informationen: Erik de Castro Lopo, Faster Floating Point to Integer Conversions., Version 1.1 02.11.2001, http://www.mega-nerd.com/FPcast/, Zugriff am 30.05.2010

von x	bis x	Schrittweite h	Max Fehler
0,00	0,03	1E-6	∞
0,03	0,05	1E-6	$6 \cdot 10^{-11}$
0,05	0,50	1E-5	$3 \cdot 10^{-09}$
0,50	20,00	1E-4	$3 \cdot 10^{-09}$
20,00	max	1E-3	$2 \cdot 10^{-11}$

Tabelle 3.1: Tabellarisierung der Funktion $BesselK_0(\sqrt{x})$. Bei $x = 0$ wird der Fehler unendlich groß. Der minimale quadratische FL Abstand für die durchgeführten Simulationen ist im Bereich von 0,03.

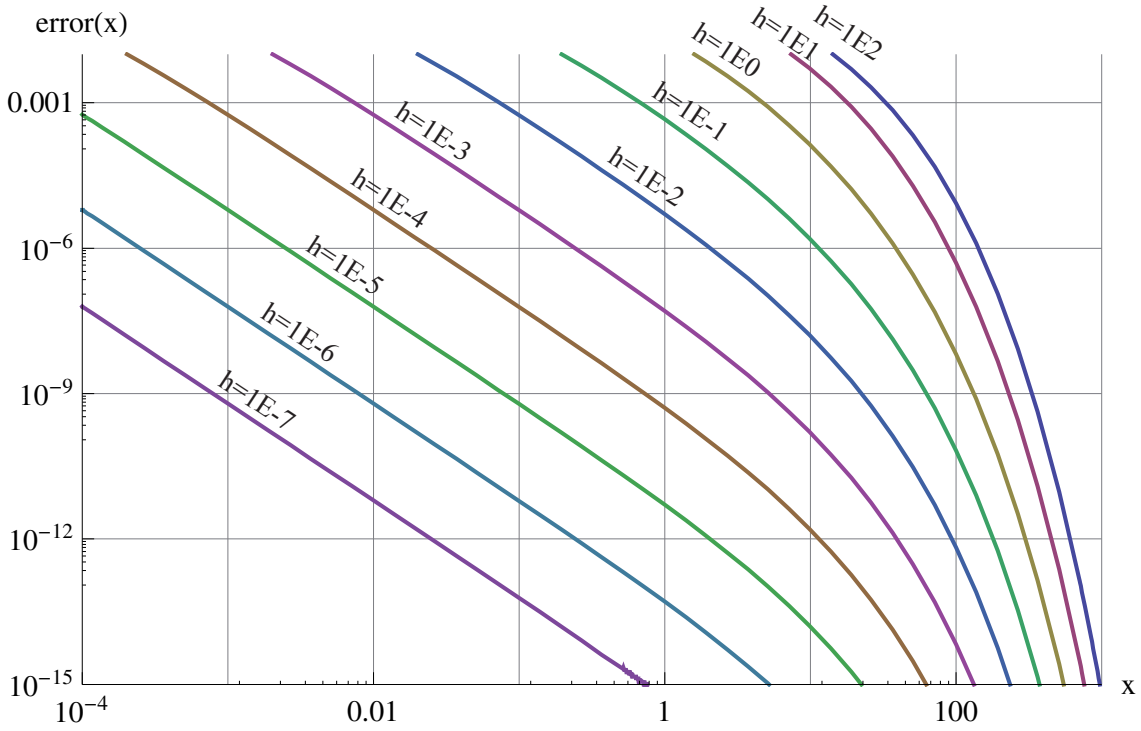


Abbildung 3.1: Maximaler Fehler $error(x)$ zwischen zwei Stützpunkten der tabellarisierten Funktion über x Werten für verschiedene Schrittweiten h.

4 Ergebnisse

4.1 Übersicht der Simulationen

Es wurden zwei Simulationsreihen gestartet.

In der ersten Simulationsreihe wurde ausgehend von einem geordnetem Dreiecksgitter die Reaktion auf kleine Störungen beobachtet. Eine Übersicht über die Simulationen ist in Tabelle 4.1 zu sehen.

In der zweiten Simulationsreihe wurden die Flusslinien zufällig angeordnet und dann ebenfalls die Reaktion auf Störungen beobachtet. Eine Übersicht über die Simulationen ist in Tabelle 4.2 zu sehen.

Die Störungen wurden durch das Herausziehen von Flusslinien aus der Mitte an den Rand simuliert. Dabei wurden jeweils zwei Flusslinien an den Rand gezogen und dann das Feld relaxiert (siehe Abbildung 4.1). Als Flusslinien zum Herausziehen wurde entweder die mit der maximalen Energie oder Zufällige ausgewählt. Genaueres zur Methode siehe die Pullout-Methode in Abschnitt 3.2.1. Die Entwicklung des Feldes nach mehreren solchen „Pullout-Relax“ Zyklen konnte dann beobachtet werden.

4.1.1 Laufzeit

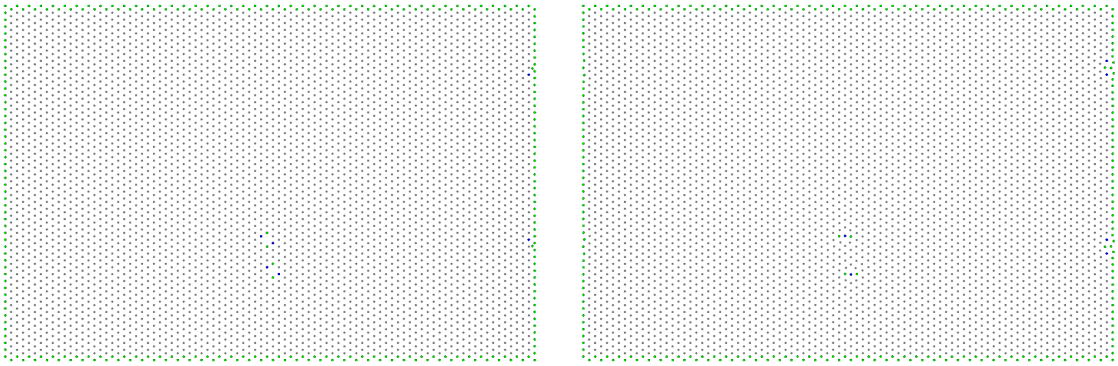
Für einen Zyklus mit drei Relaxionschritten und 7040 Flusslinien bis zu einer minimalen Kraft von 10^{-4} benötigt ein Rechner mit 2,9 GHz Prozessor im Singlecoremodus des

Versuch	B-Feld [T]	FL Zahl	Min Kraft	Zyklen
G01	1.0	4680	1E-3	53

Tabelle 4.1: Simulationen, ausgehend von einem geordneten Gitter, führten immer zu dem hier vorgestellten Resultat. Daher werden weitere Simulationen nicht angeführt. Details zu den Parametern siehe Abschnitt 3.2.2.

Simulation	B-Feld [T]	FL Zahl	Min Kraft	Zyklen
U01 (17)	1.0	4680	1E-3	586
U02 (19)	1.5	7040	1E-4	700
U03 (18)	1.5	7040	1E-3	550

Tabelle 4.2: Übersicht über die durchgeführten Simulationen ausgehend von einer zufälligen FL Anordnung. Details zu den Parametern siehe Abschnitt 3.2.2.



(a) Pullout (b) Relaxed

Abbildung 4.1: Der erste Pullout-Relax Zyklus an einem geordneten Feld.

Programms ca. 35 Minuten. Dabei wurden für die Relaxationsschritte folgende minimale Kräften: 0,005, 0,001 und 0,0001. Jedoch variiert die Laufzeit stark von Zyklus zu Zyklus.

4.2 Ergebnisse und Beobachtungen

4.2.1 Störung eines geordneten Feldes

Das Herausziehen von Flusslinien aus einem geordneten Feld hinterlässt nur einzelne Fehlstellen, die jedoch nach weiteren Pullout-Relax Zyklen nach und nach wieder verschwinden. Eine größere Tendenz zur Umordnung ist nicht zu erkennen (siehe Abbildung 4.2).

4.2.2 Störung eines ungeordneten Feldes

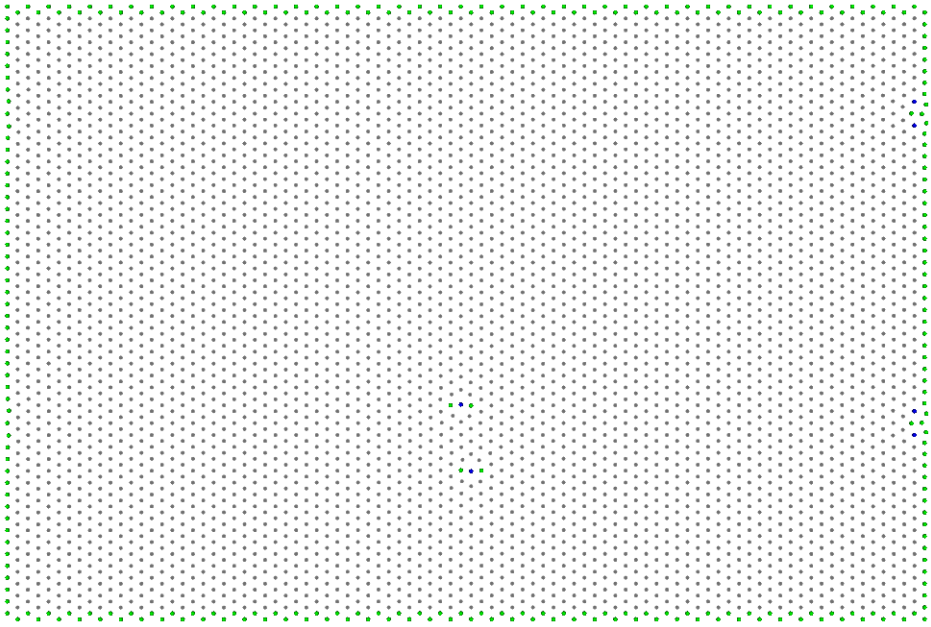
Die FL ordnen sich hier nach dem ersten Relaxieren in einem Dreiecksgitter an, welches jedoch durch Versetzungen gestört ist. Es entsteht ein polykristallines Flusslinien Gitter, in dem deutlich die durch Versetzungen gebildeten Korngrenzen zu sehen sind.

Hier ist deutlich zu beobachten, dass sich das Gitter immer mehr ordnet, je mehr Pullout-Relax Zyklen es ausgesetzt wird (siehe Abbildung 4.4). Dies geschieht, indem die Korngrenzen infolge von wanderenden Versetzungen relaxieren.

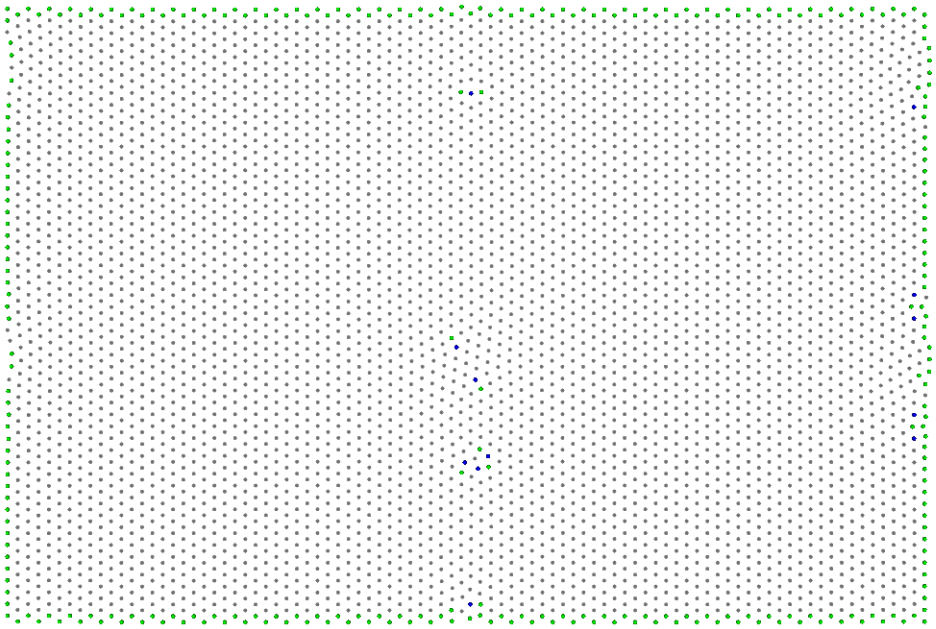
Der Unterschied zwischen kleineren Feldern (Simulation U01 Abbildung 4.3) und größeren Feldern (Simulation U02 (Abbildung 4.4), U03 (Abbildung 4.5)) hält sich in Grenzen. Bei Simulation U02 (größeres B-Feld) hält sich jedoch ein Bezirk (links im Feld) relativ stabil 4.4) und verändert sich nur langsam. Bei kleineren Feldern sind solche Bezirke nach vielen Zyklen nicht mehr zu beobachten.

4.3 Interpretation

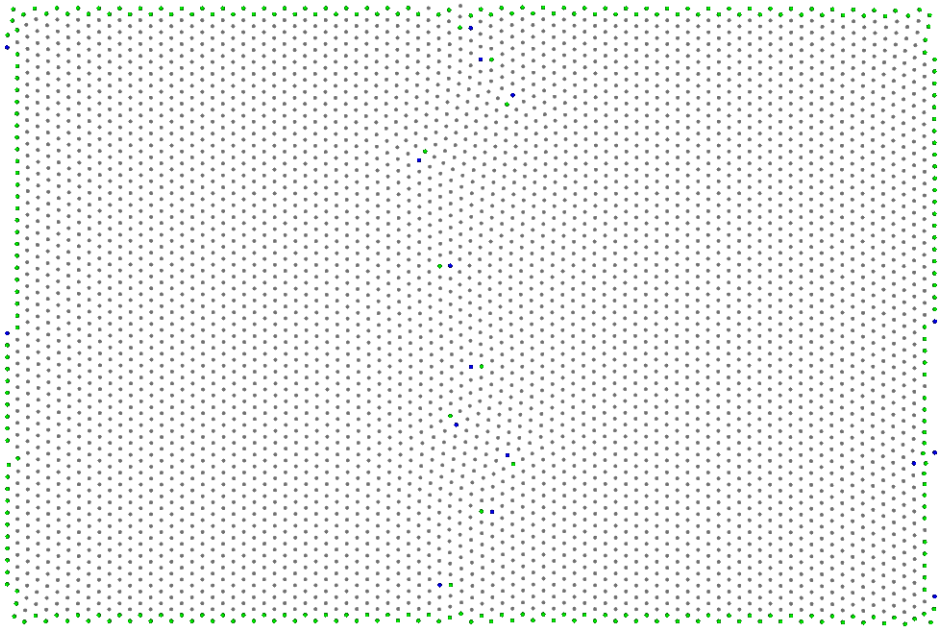
Das Flussliniengitter unter diesen periodischen Randbedingungen zeigt eine starke Tendenz sich in einem regelmäßigen Dreiecksgitter zu ordnen. Die verbleibende Körnigkeit des Gitters ist durch die restliche Kraft (1e-4 oder 1e-3 und die vergleichsweise hohe Wirkung der periodischen Randbedingungen gegeben. Die Wie sehr sich dieses Ergebnis jedoch auf Flussliniengitter im Typ-II Supraleiter übertragen lässt, bleibt eine Frage, die noch zu klären ist.



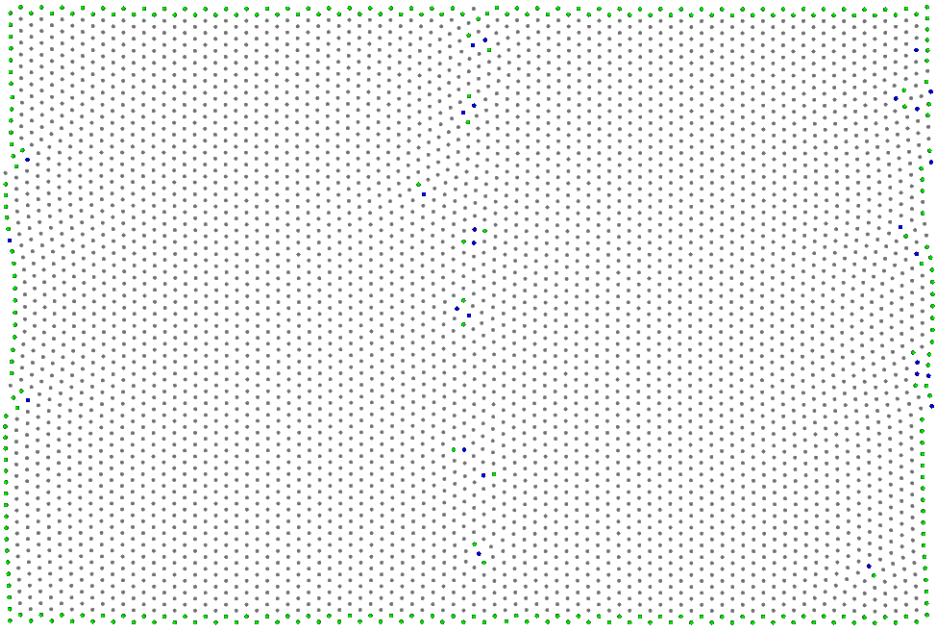
(4.2.a) Start



(4.2.b) 5 Zyklen

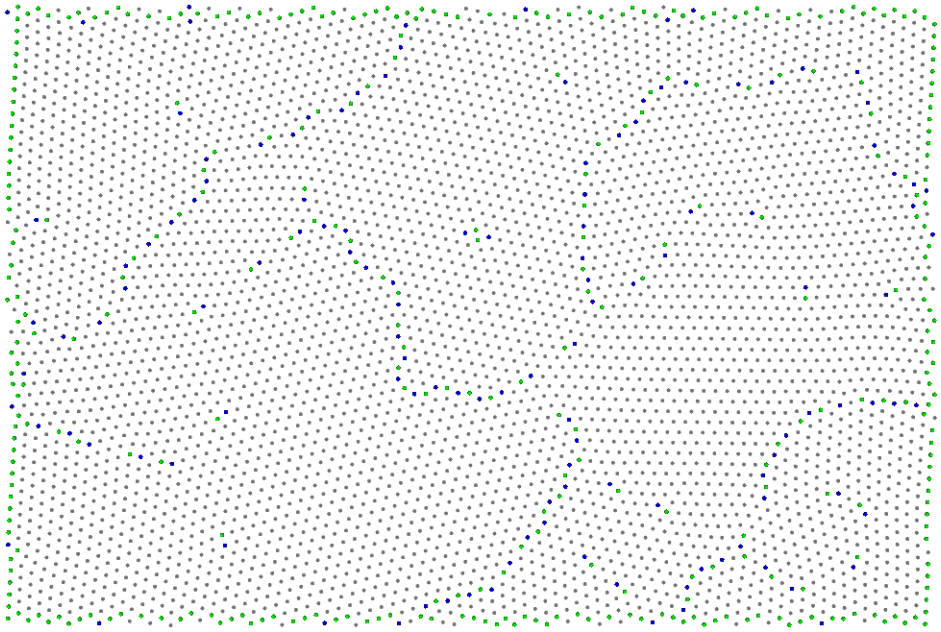


(4.2.c) 25 Zyklen

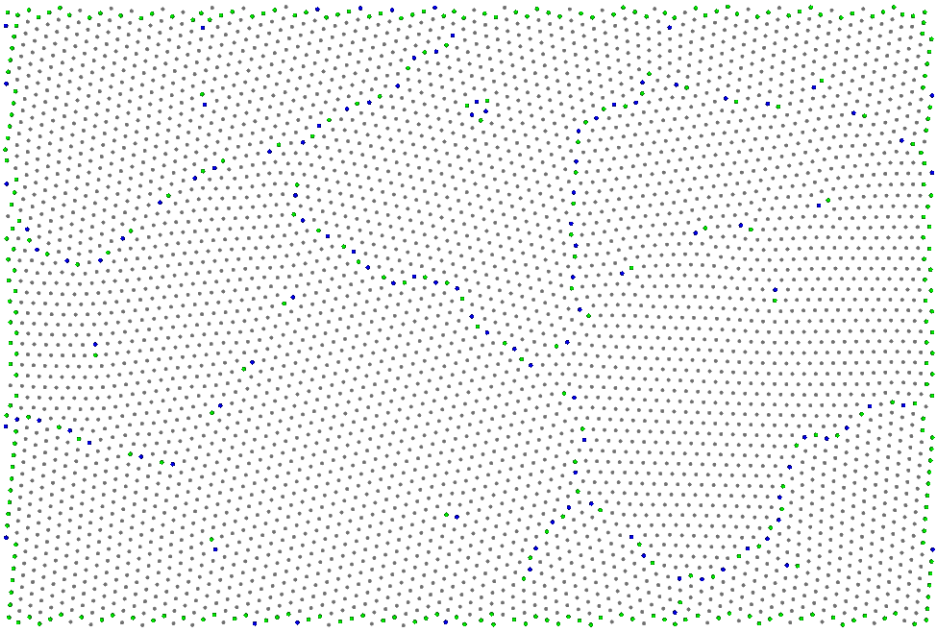


(4.2.d) 50 Zyklen

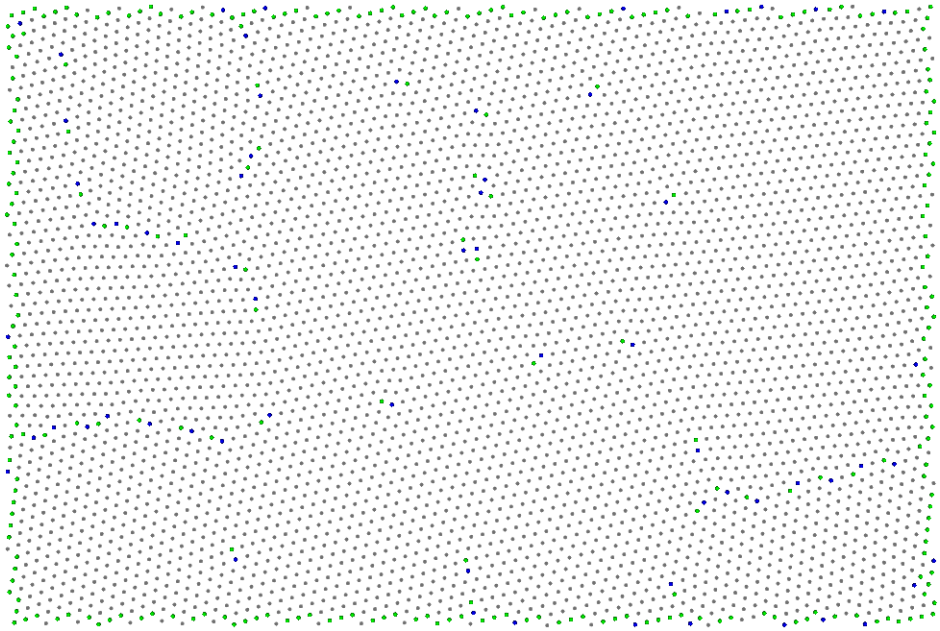
Abbildung 4.2: Die Veränderung des Feldes nach mehreren Pullout-Relax Zyklen. Versetzungen sind blau und grün makiert. (Auszug aus Simulation G01)



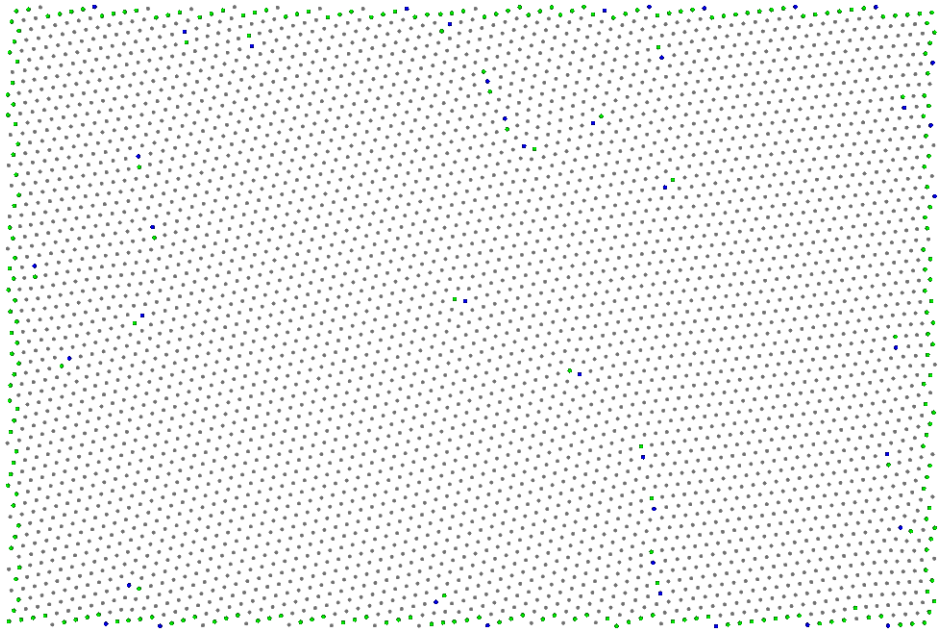
(4.3.a) Start



(4.3.b) 26 Zyklen

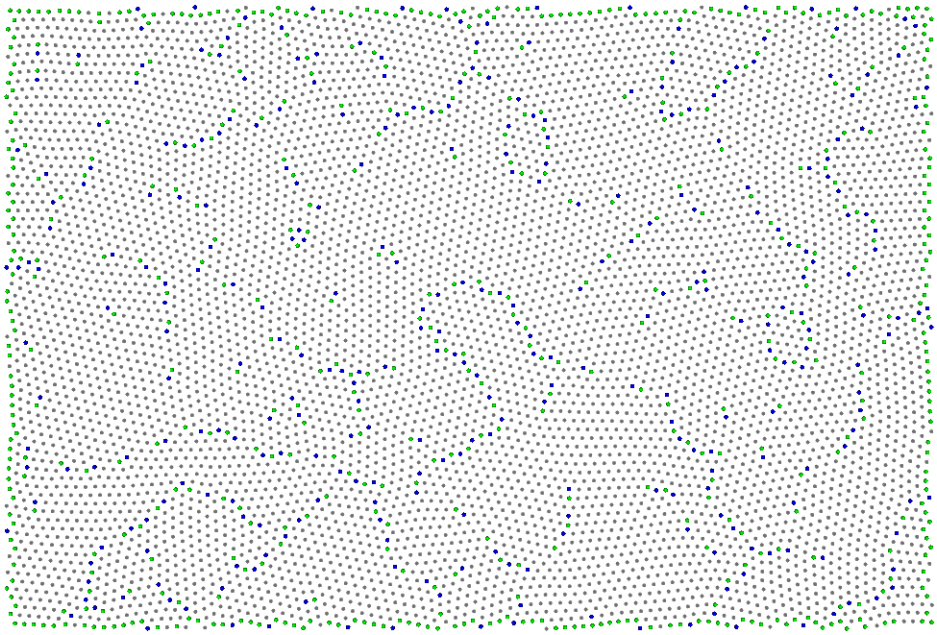


(4.3.c) 250 Zyklen

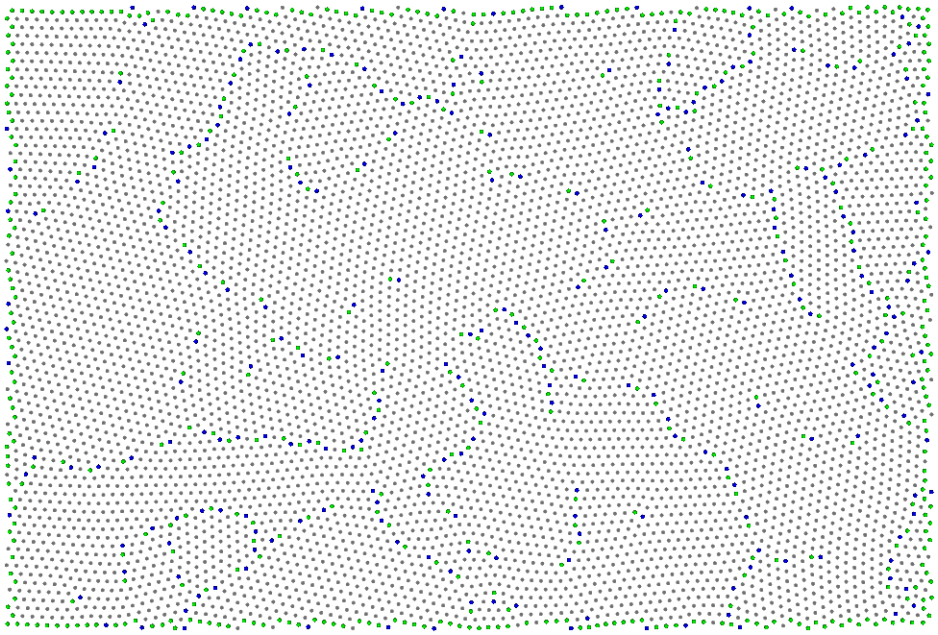


(4.3.d) 400 Zyklen

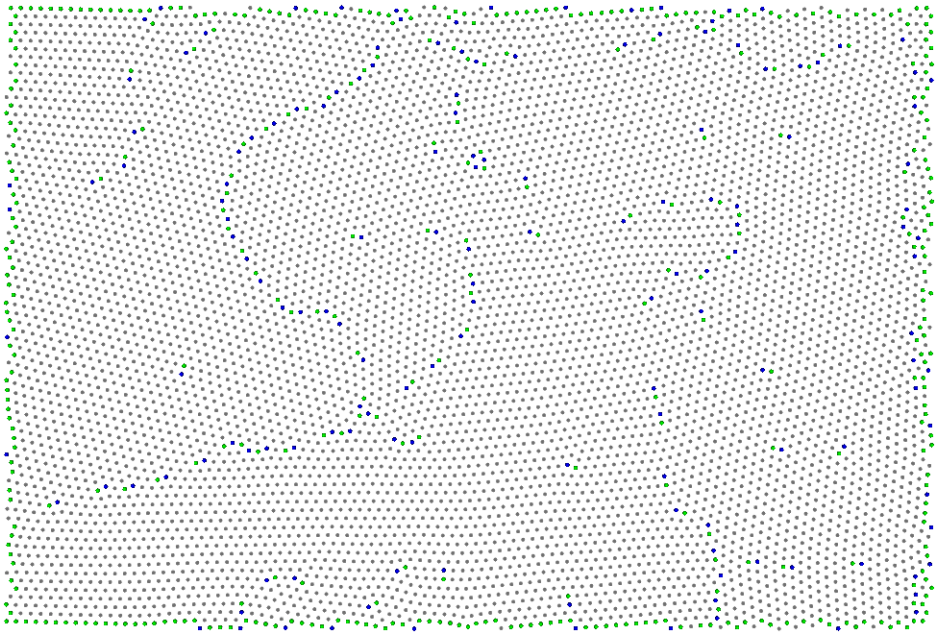
Abbildung 4.3: Die Veränderung des Feldes nach mehreren Pullout-Relax Zyklen. Versetzungen sind blau und grün markiert. (Auszug aus Simulation U01)



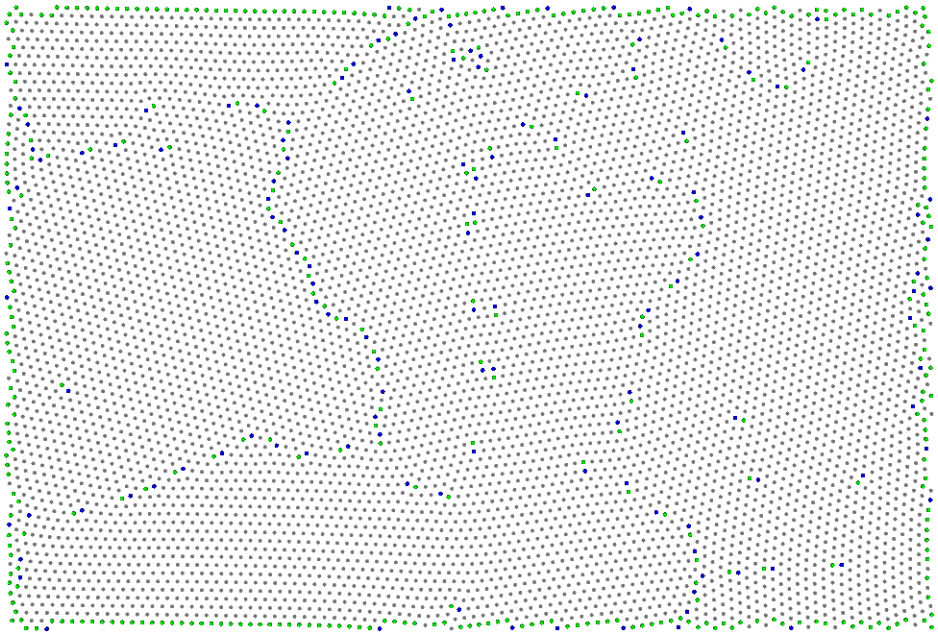
(4.4.a) Start



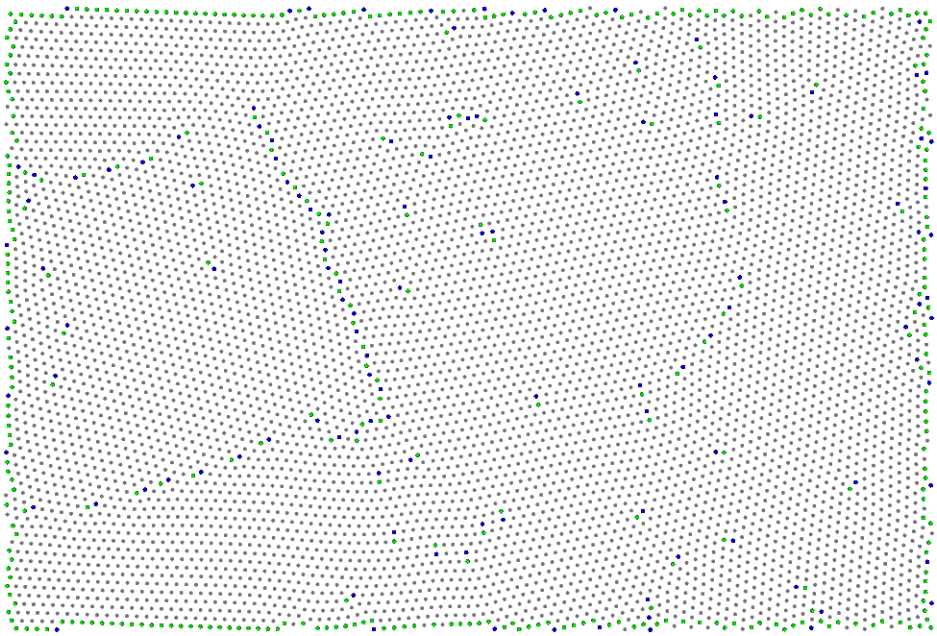
(4.4.b) 26 Zyklen



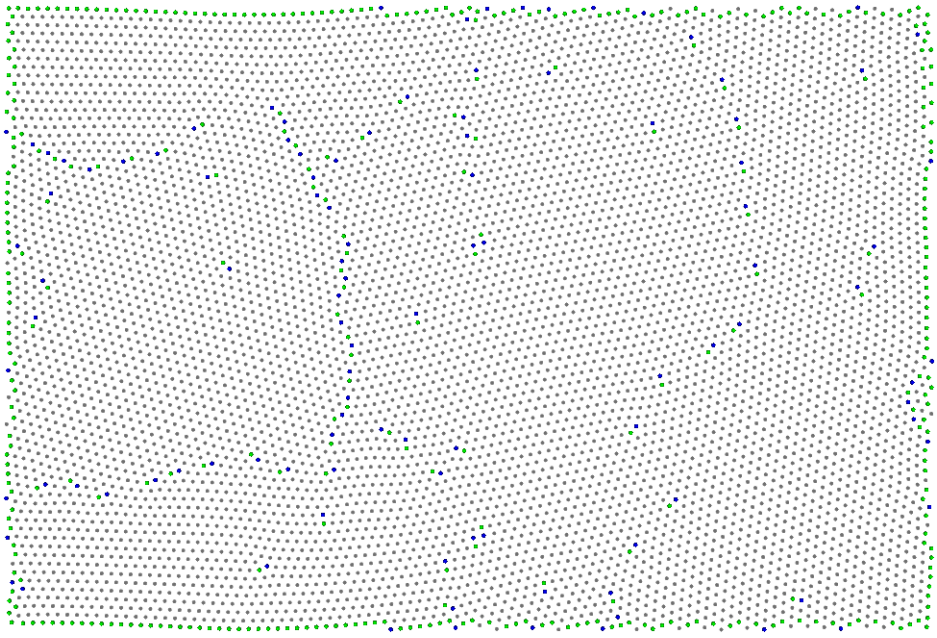
(4.4.c) 250 Zyklen



(4.4.d) 400 Zyklen

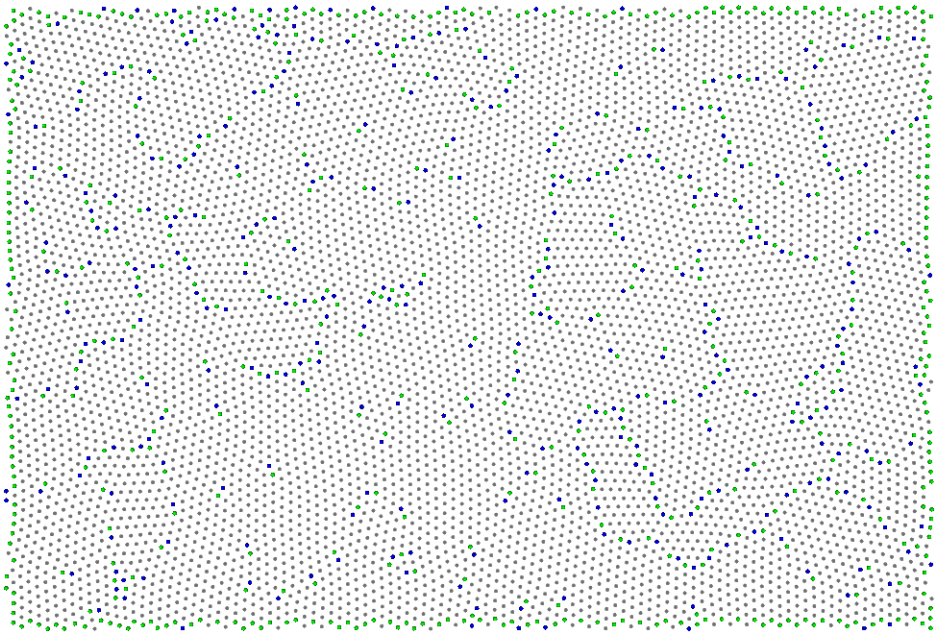


(4.4.e) 550 Zyklen

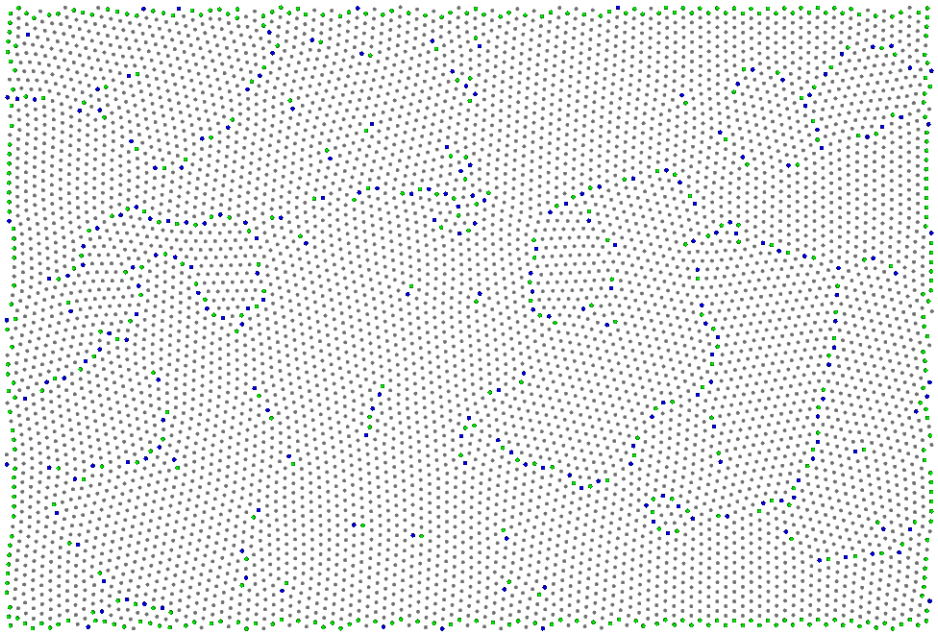


(4.4.f) 700 Zyklen

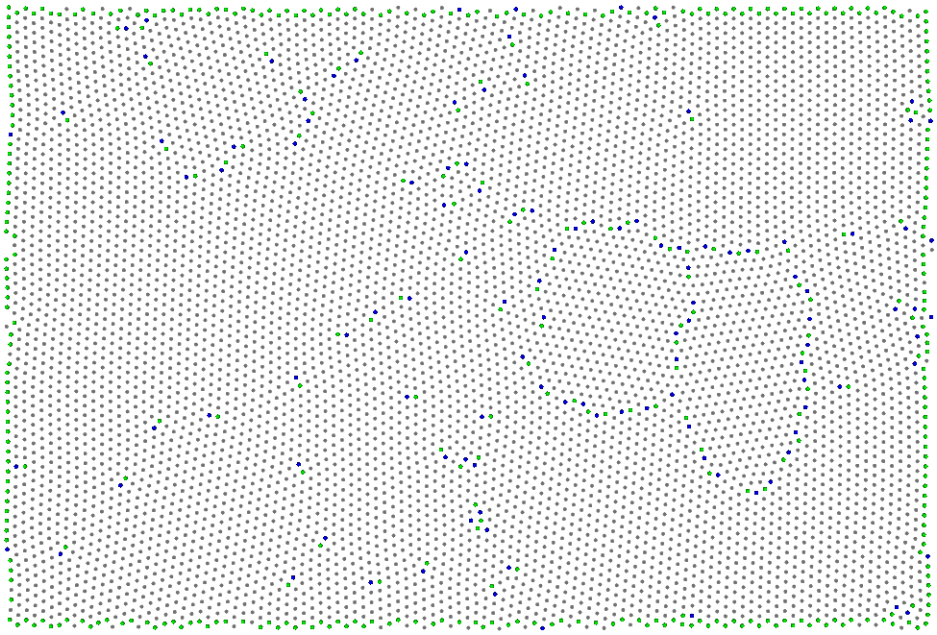
Abbildung 4.4: Die Veränderung des Feldes nach mehreren Pullout-Relax Zyklen. Versetzungen sind blau und grün markiert. (Auszug aus Simulation U02)



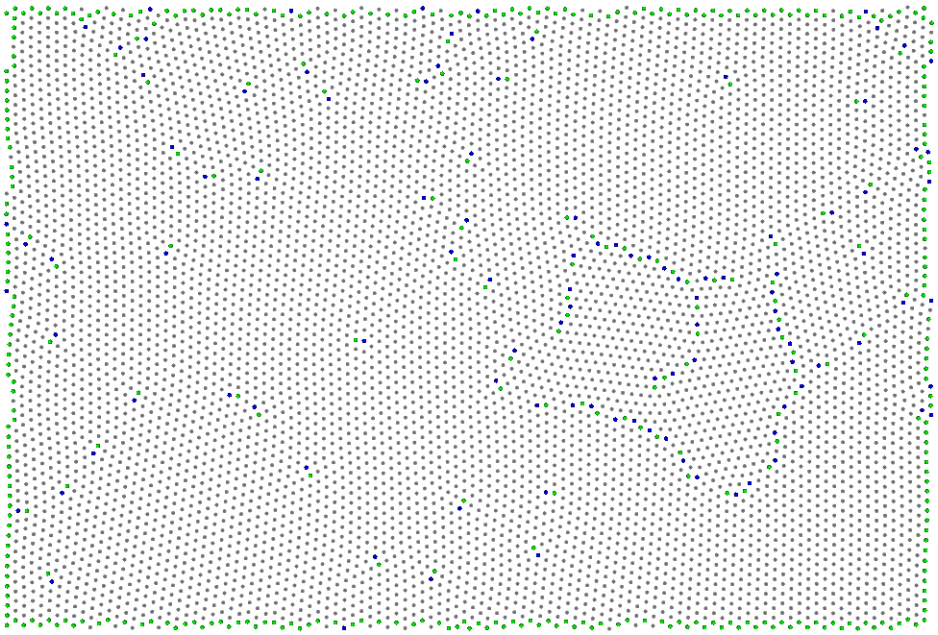
(4.5.a) Start



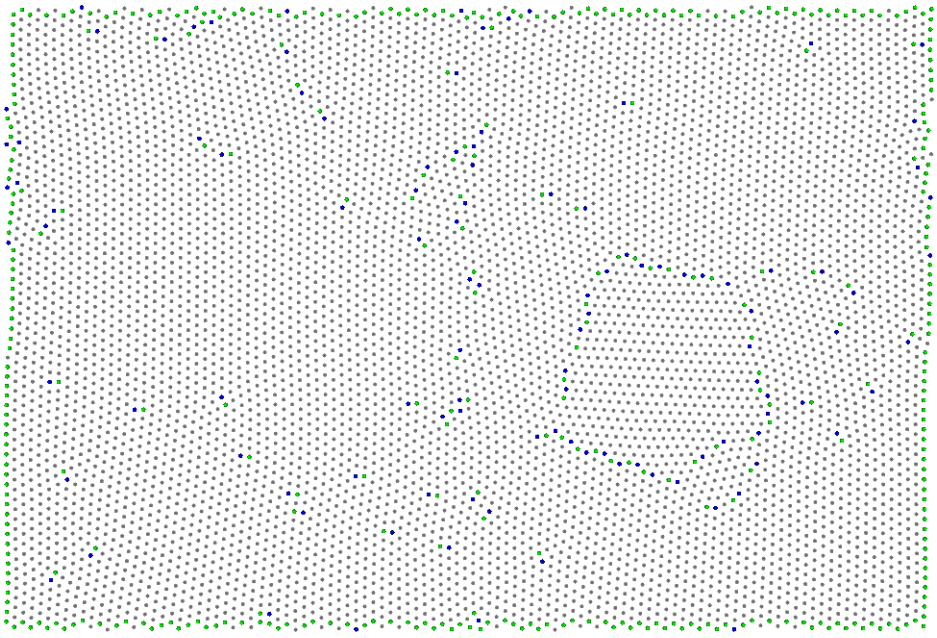
(4.5.b) 26 Zyklen



(4.5.c) 250 Zyklen



(4.5.d) 400 Zyklen



(4.5.e) 550 Zyklen

Abbildung 4.5: Die Veränderung des Feldes nach mehreren Pullout-Relax Zyklen. Versetzungen sind blau und grün markiert. (Auszug aus Simulation U03)

5 Zusammenfassung

Zusammenfassend wurden im Rahmen der Projektarbeit folgende wichtige Schritte vorgenommen:

- Optimierung des Codes durch Beseitigung redundanter Berechnungen.
- Beschleunigung durch Tabellarisieren der Bessel K_0 Funktion und Abschätzung des dadurch entstehenden Fehlers.
- Programmieren der Pullout Methode zum Einbauen von Störungen.
- Testen des Programms mit unterschiedlichen Simulationen.
- Untersuchung der Daten der Testläufe mit dem Ergebnis, dass die Felder sehr starke Tendenzen zeigen sich vollständig zu ordnen.