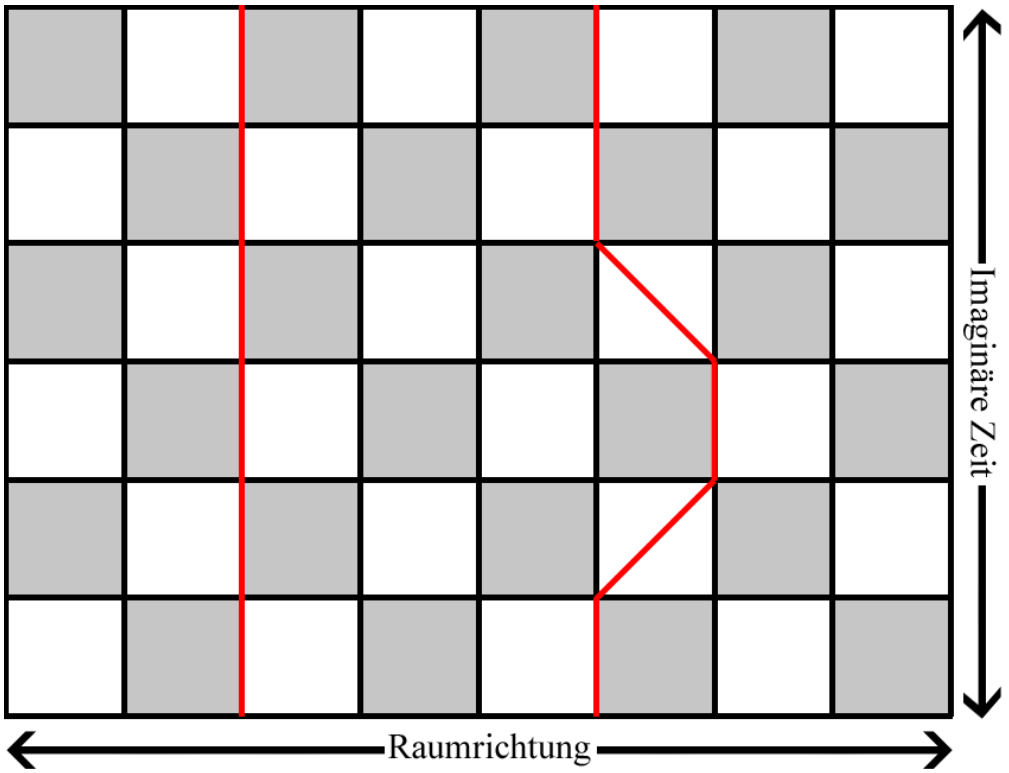
QMC Loop-Algorithmus

Im Rahmen der Veranstaltung
„Moderne numerische Methoden der Festkörperphysik“
Prof. Dr. Nils Blümer

Von Benjamin Niepelt

Weltlinien

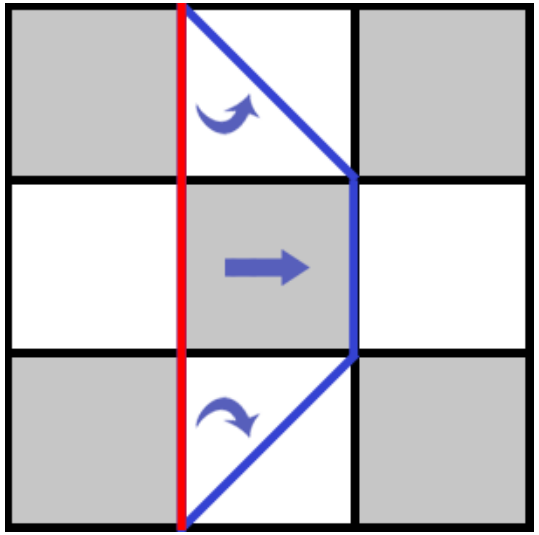
Algorithmus

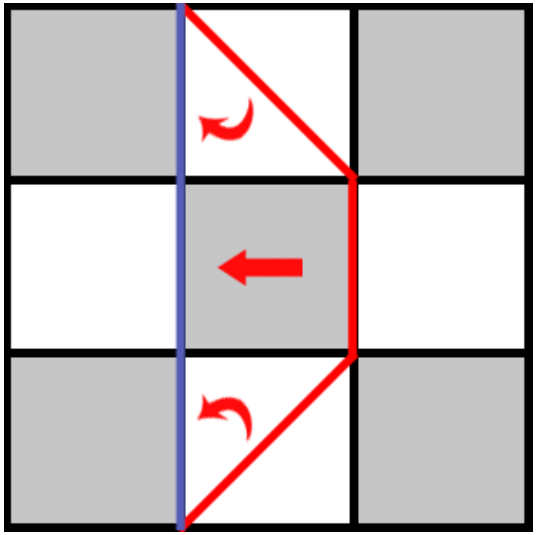


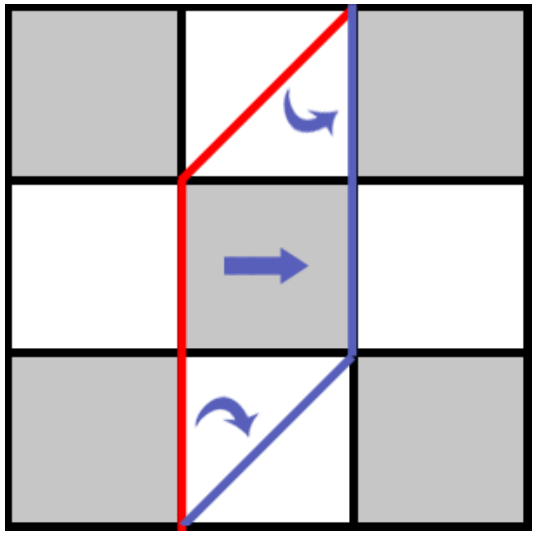
- **Trotter-Diskretisierung** ($\Delta\tau = \beta / M$) und Zerlegung des Hamilton-Operators in 2 Summanden (Reduktion auf 2-Platz-Problem)
- **d+1**-Dimensionales System (d Raum- eine Imaginäre Zeit Dimension)
- Gitter in allen Dimensionen **Gradzahlig**
- Periodische Randbedingungen
- Beschreibt Evolution von Spin-Up-Zuständen

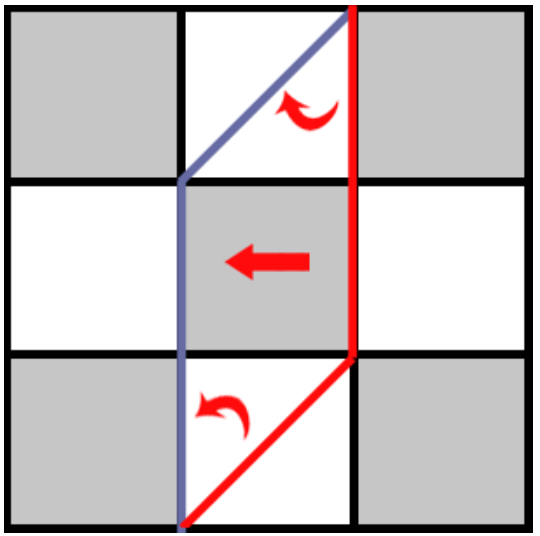
Weltlinien

Algorithmus





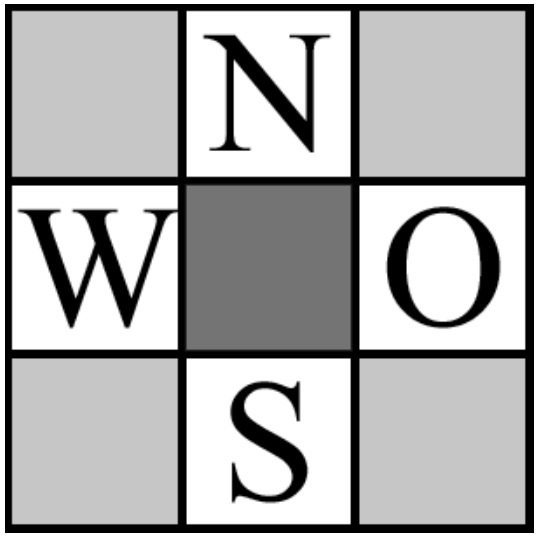




- Wechsel finden nur in grauen Feldern statt
- Nur dann, wenn im jeweiligen Feld nur **ein** Up-Spin ist
- Nachbarfelder müssen den Wechsel zulassen (dürfen also z.B. keine Diagonalen haben)
- Entscheidung ob gewechselt wird hängt von den Gewichten ab

Weltlinien

Algorithmus



- Das Gewicht der Konfiguration des hier dunkelgrauen Feldes ergibt sich aus dem Produkt der Matrixelemente der umliegenden weißen Felder

- Das relative Gewicht r erhält man aus Division der neuen, vorgeschlagenen und der alten Konfiguration

$$r = \frac{\Omega(w_{neu})}{\Omega(w_{alt})} = \frac{N_{neu} S_{neu} O_{neu} W_{neu}}{N_{alt} S_{alt} O_{alt} W_{alt}}$$

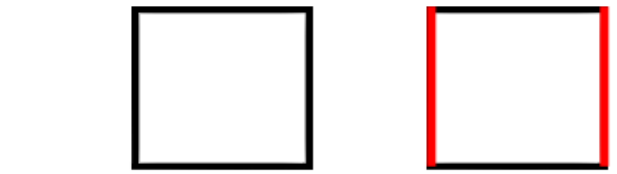
- Die Wahrscheinlichkeit, dass eine neue Konfiguration akzeptiert wird, (nach Heath-Bath) lautet dann:

$$p = \frac{r}{r+1}$$

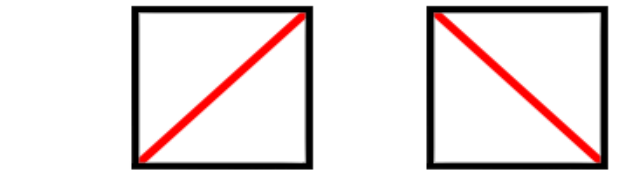
Weltlinien

Algorithmus

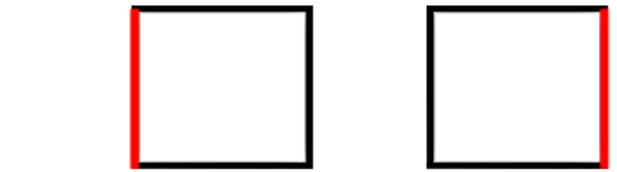
Die Gewichte ergeben sich aus der Faktorisierung der Matrixelemente in der Eigenbasis, was auf ein 2-Platz-Problem führt - welches wir komplett lösen können:



$$e^{-\Delta\tau J/4}$$



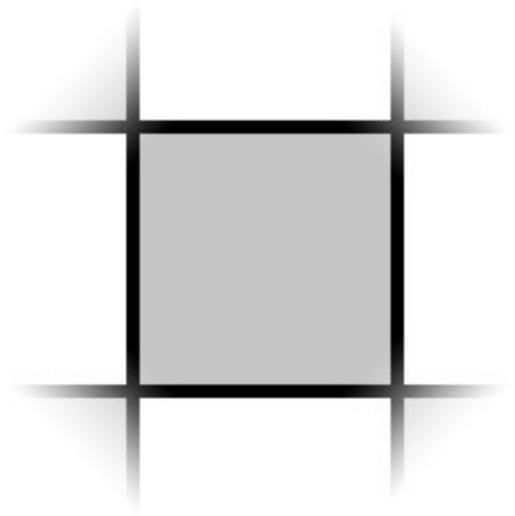
$$-e^{\Delta\tau J/4}\sinh(\Delta\tau J/2)$$



$$e^{\Delta\tau J/4}\cosh(\Delta\tau J/2)$$

Weltlinien

Implementierung



- Gitter aus Plaketten
- Jede Plakette wird durch die **2 Flanken und 2 mögliche Diagonale** charakterisiert
- Jede Flanke kann entweder 'wahr' oder 'falsch' sein
- 'wahr' entspricht hierbei unseren Weltlinien, also Up-Spins
- 'falsch' entsprechend Down-Spins
- Diagonalen sind entsprechend nur möglich, wenn keine Flanken gesetzt sind

Weltlinien

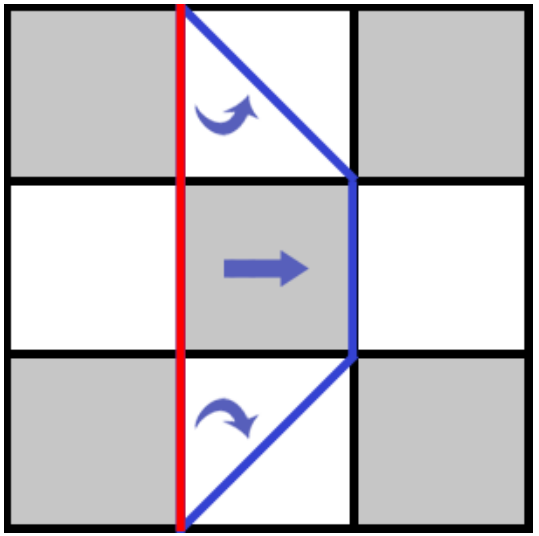
Implementierung

```
fuer alle grauen Plaketten {
        Schlage Spinflip vor
        Berechne Gewicht w_alt
        Flippe Spin
        Berechne Gewicht w_neu
        w = w_neu/w_alt
        p = w / (1+w)
        wenn random > p {
                Flippe zurueck
        }
}

Messe Observablen
```

- In der konkreten Implementierung werden alte und neue Gewichte von Flipfunktion zurückgegeben
- Der Flip wird direkt ausgeführt
- Die „wenn“-Bedingung entspricht also nicht der Akzeptanz sondern der **Nicht-Akzeptanz**
- Wird nun nicht akzeptiert, wird zurück geklappt

Weltlinien

Implementierung

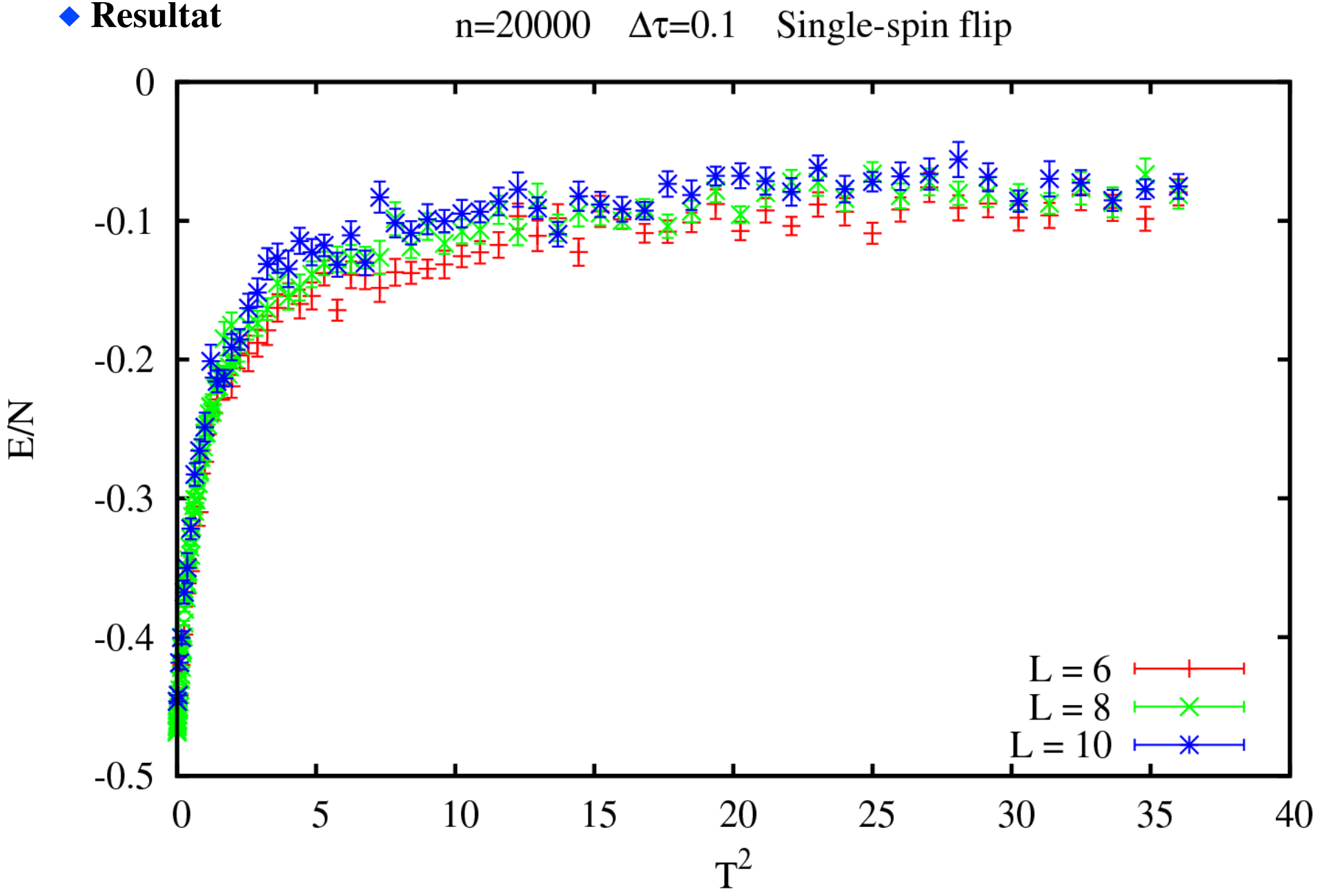


- Implementierung eines Switches

```
wenn Plakette antiferromagnetisch {
    Klappen der Plakette

    wenn obere Plakette=links {
        setze obere Plakette=Diagonal Links
    } sonst {
        setze obere Plakette=Rechts
    }
    wenn untere Plakette=Links {
        setze untere Plakette=Diagonal Rechts
    } sonst {
        setze untere Plakette=Rechts
    }
}
```

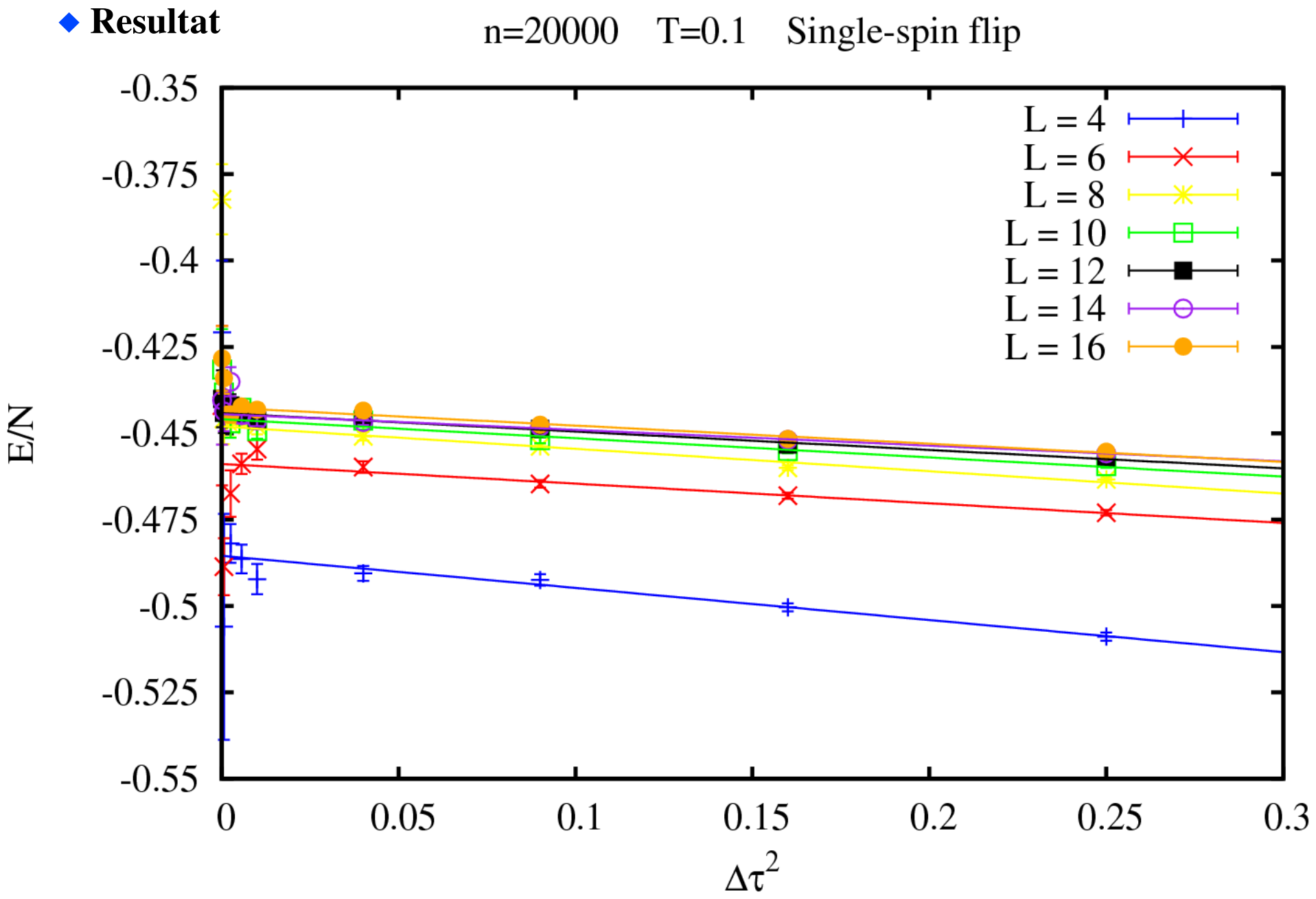
Weltlinien

- Resultat



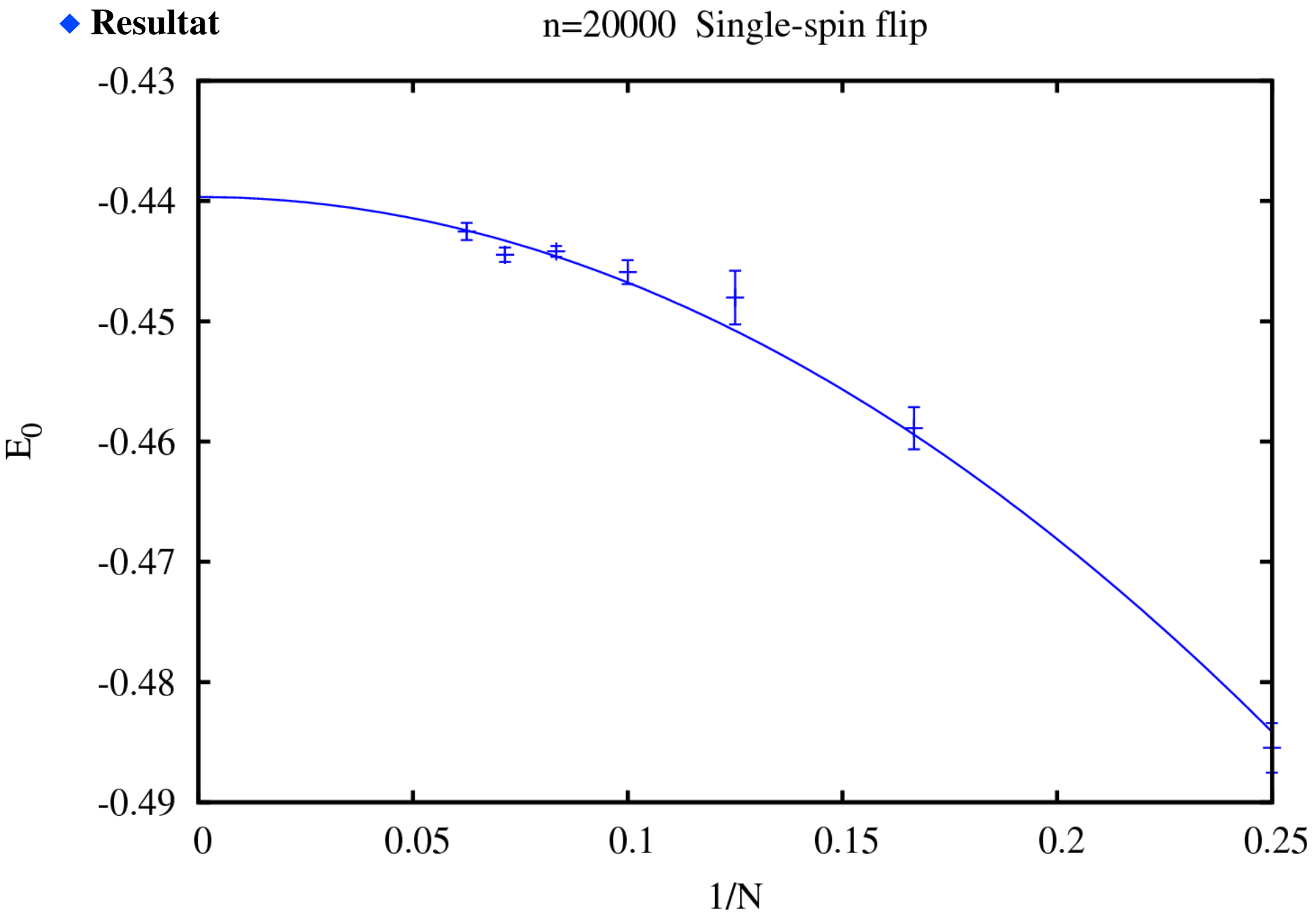
Weltlinien

- Resultat



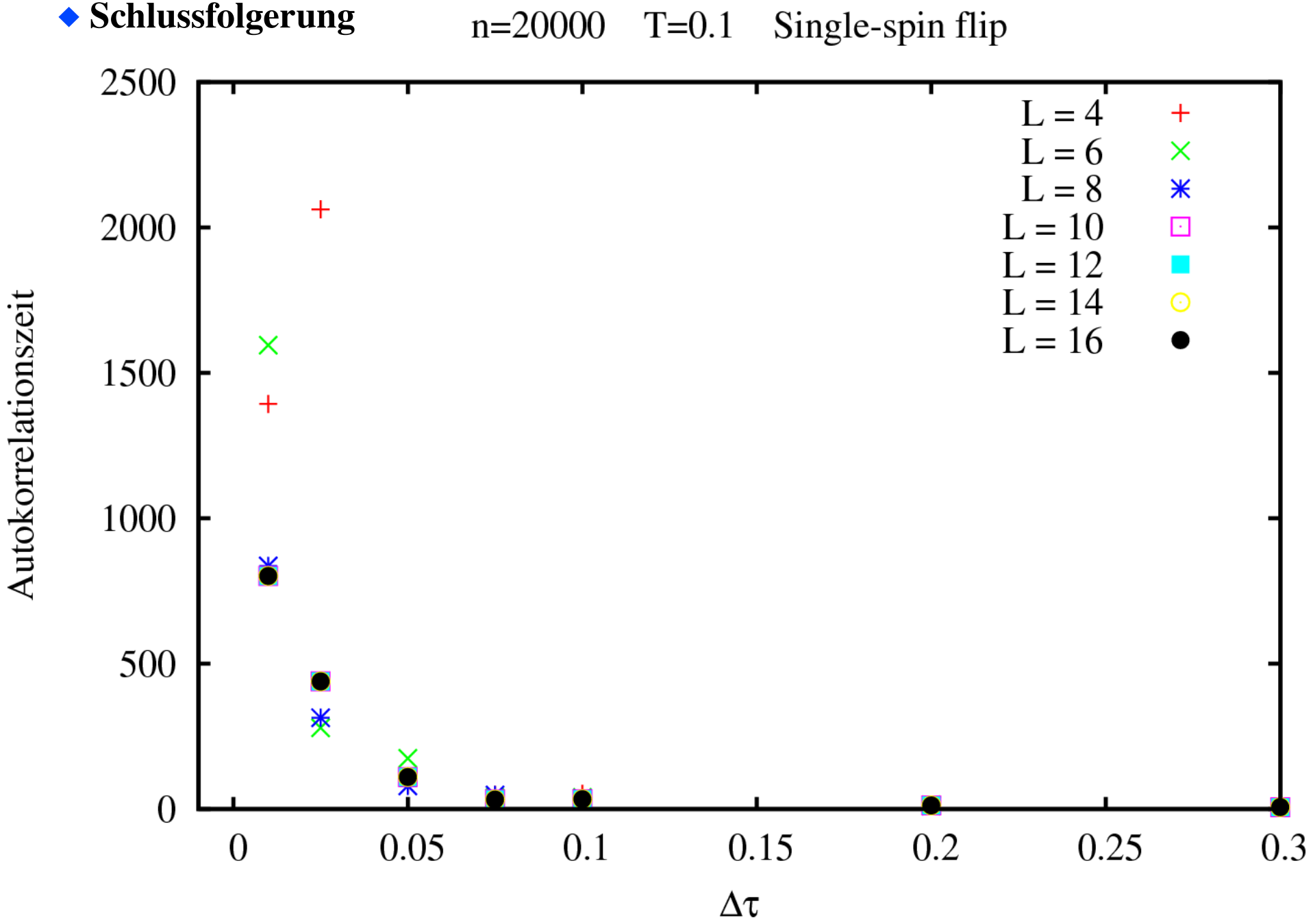
Weltlinien

- Resultat



Weltlinien

- Schlussfolgerung



Weltlinien

Schlussfolgerung

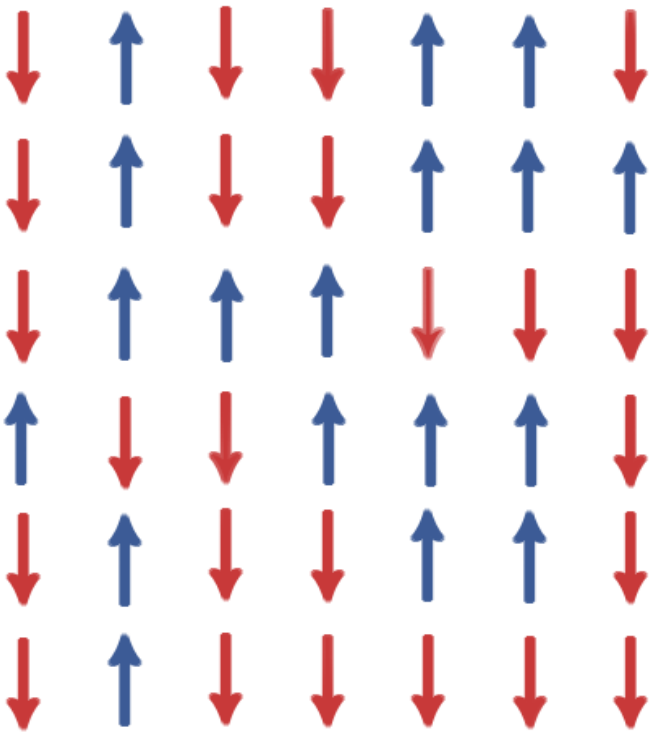
Probleme:

- Beim Weltlinien-MC ist die **Zahl der Weltlinien konstant** => systematischer Fehler
- **Autokorrelationszeit wird sehr groß** für kleine Zeitschritte => große Fehler
- Dadurch Extrapolation $\Delta\tau \to 0$ **wenig Aussagekraft**
- Ist bei großen Gittern (also kleinem $\Delta\tau$ oder langer Spinkette) sehr teuer
- Wir kennen dieses Verhalten schon bei Single-spin flips im Ising-Modell

Lösung bei Ising: Cluster-moves

Cluster-Moves

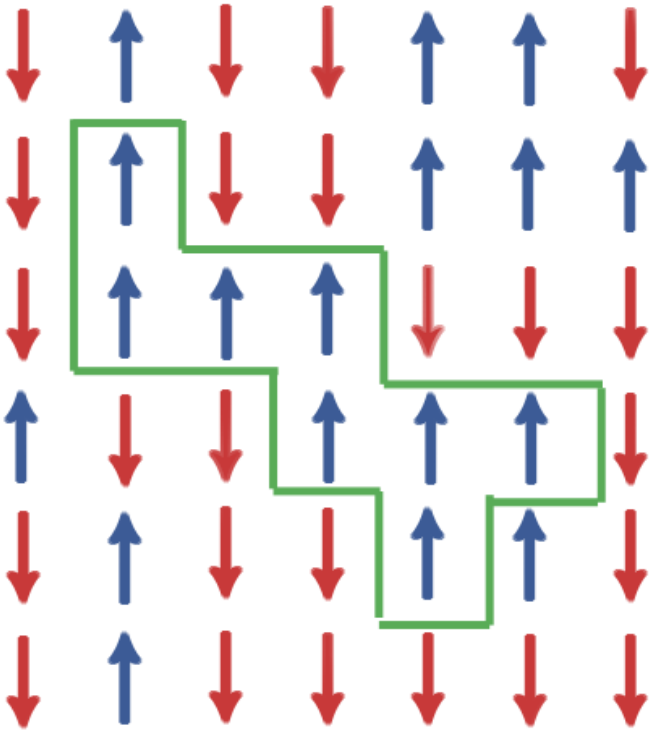
Idee



- Betrachte z.B. ein 2D-Gitter
- Wähle Startpunkt

Cluster-Moves

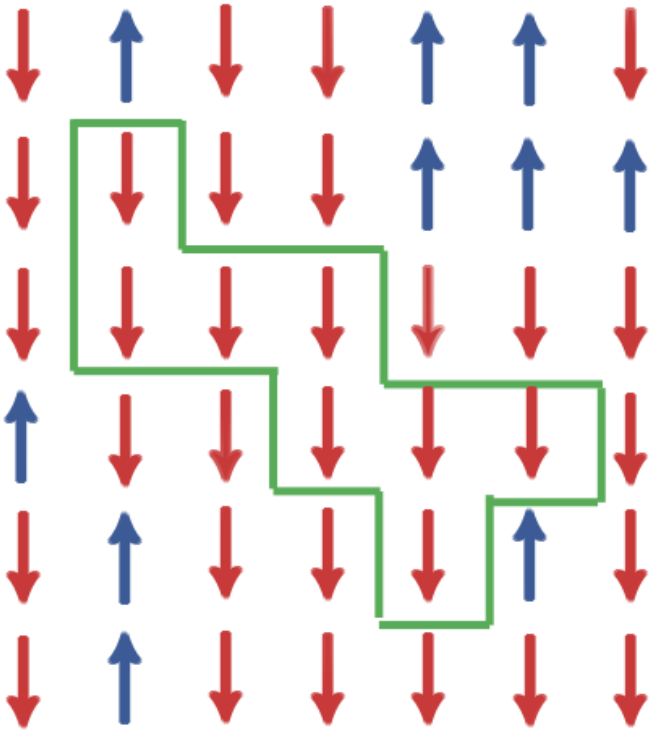
Idee



- Betrachte z.B. ein 2D-Gitter
- Wähle Startpunkt
- Besuche Nachbarn, überprüfe Richtung auf Übereinstimmung
- Wenn ja füge ihn zum Cluster hinzu (ggf. mit gewisser Wahrscheinlichkeit)

Cluster-Moves

Idee



- Betrachte z.B. ein 2D-Gitter
- Wähle Startpunkt
- Besuche Nachbarn, überprüfe Richtung auf Übereinstimmung
- Wenn ja füge ihn zum Cluster hinzu (ggf. mit gewisser Wahrscheinlichkeit)
- Klappe ganzes Cluster

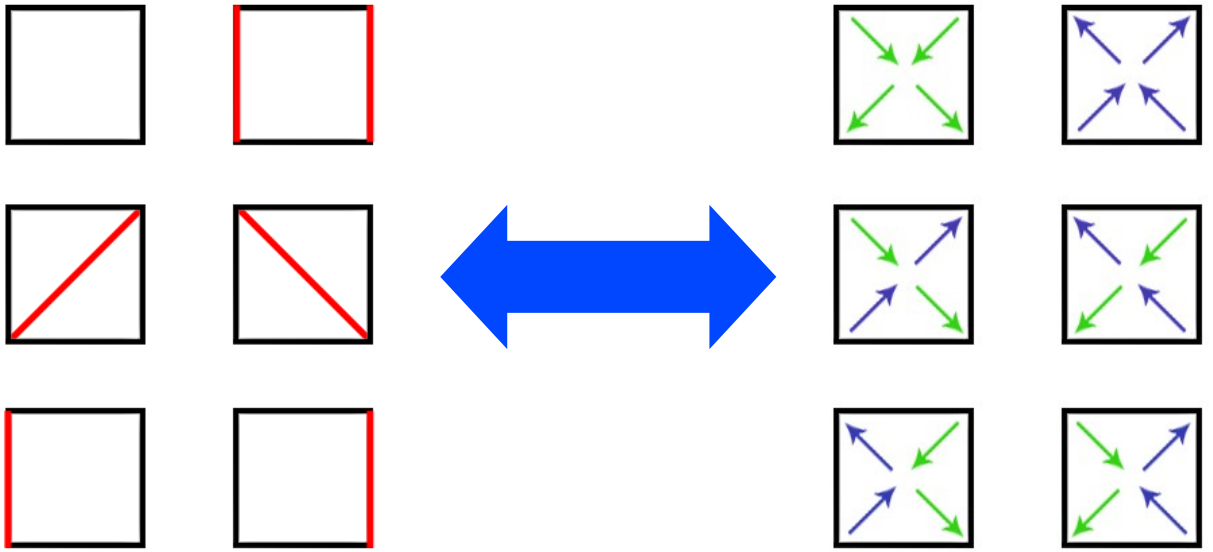
Ungewichteter Loop

- **Algorithmus**

- Weiße Plaketten werden eindeutigen, sog. „Vertizes“ zugeordnet

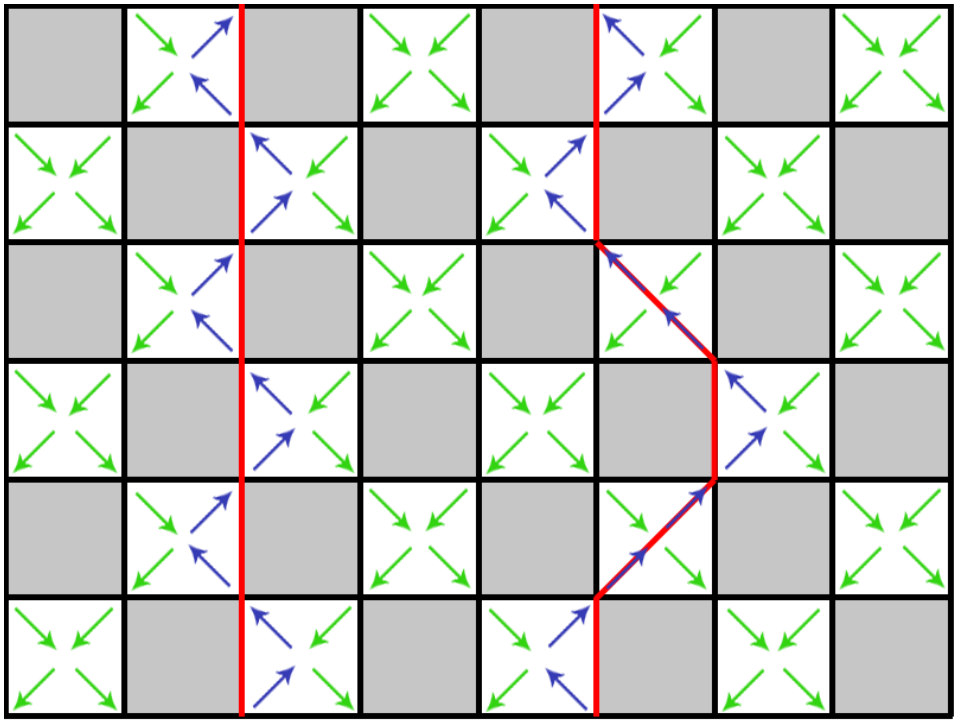
- Jedes Vertex hat **2 Ein- und Ausgänge**

- Nach oben gerichtete Pfeile (blau) entsprechen Weltlinien, nach unten entsprechend einer „freien“ Flanke



Ungewichteter Loop

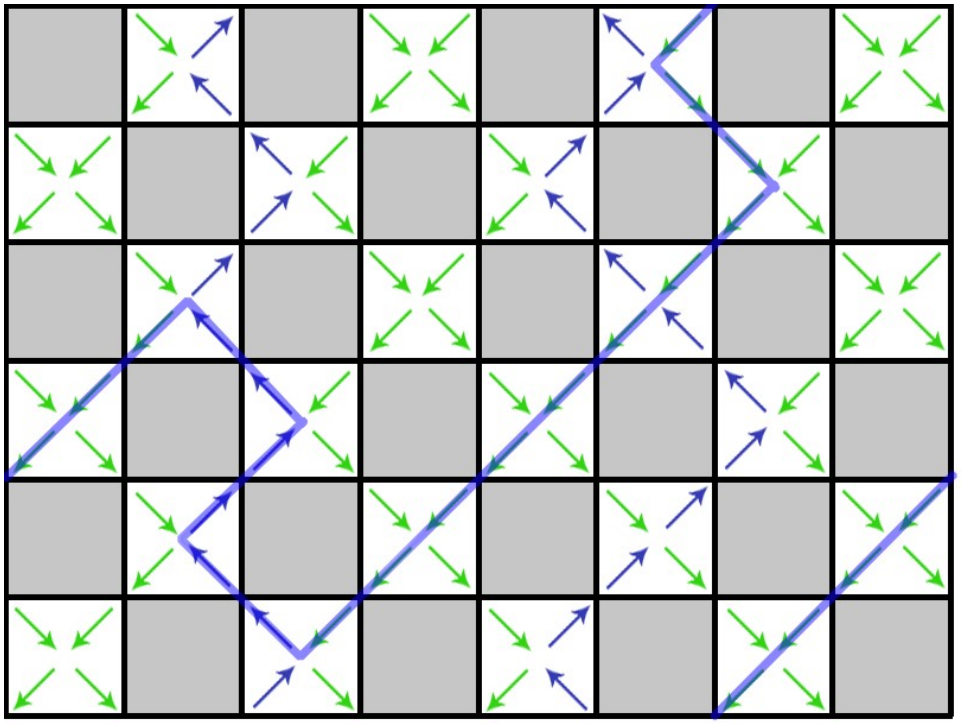
Algorithmus



- Die Vertizes bilden ein **differgenzfreies** Gitter

Ungewichteter Loop

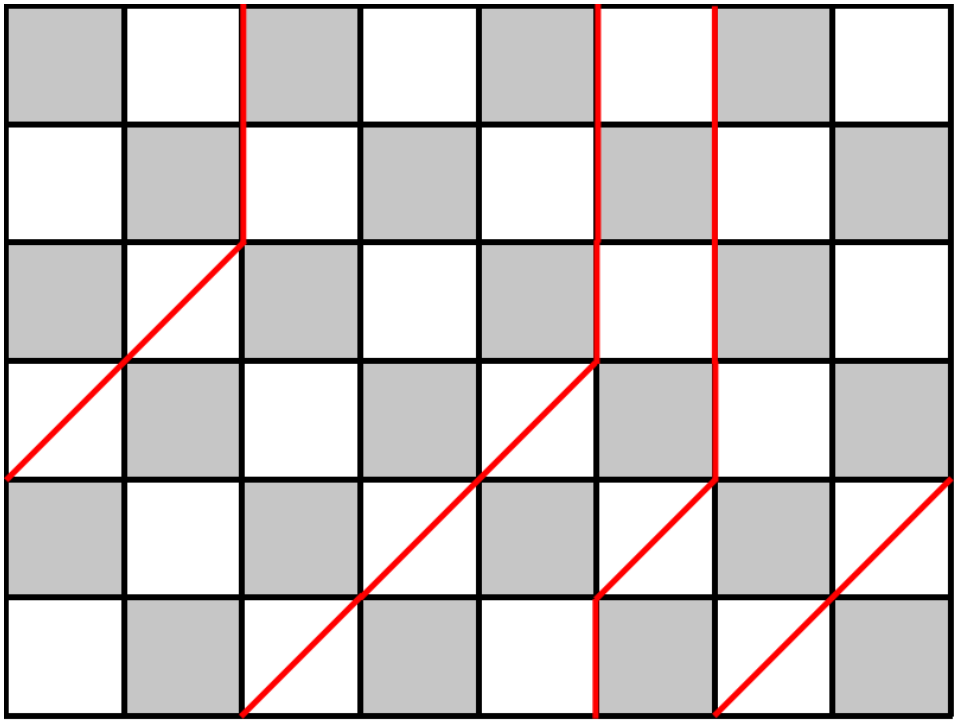
- Algorithmus



- Die Vertizes bilden ein differgenzfreies Gitter
- Finden von geschlossenem Pfad entlang den Pfeilen

Ungewichteter Loop

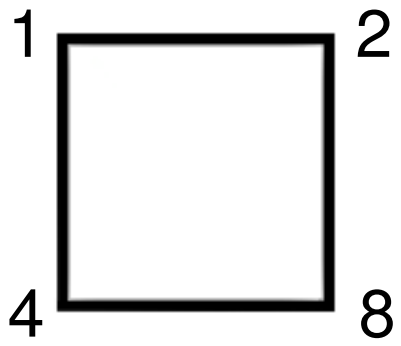
Algorithmus

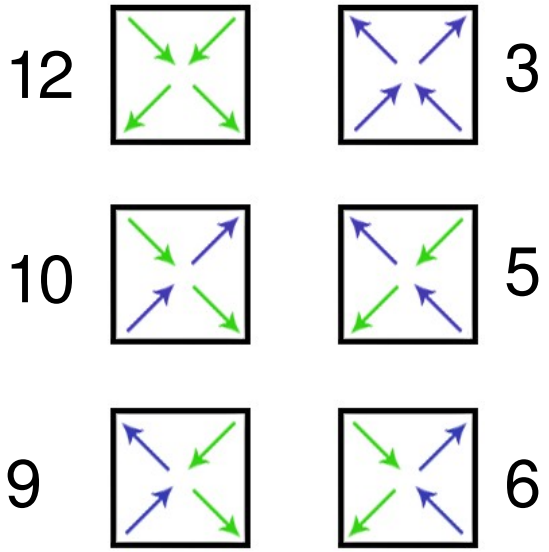


- Die Vertizes bilden ein differgenzfreies Gitter
- Finden von geschlossenem Pfad entlang den Pfeilen
- Alle Pfeile entlang des Pfades klappen (also Eingang -> Ausgang)
- Akzeptiere Move z.B. nach heat-bath

Ungewichteter Loop

Implementierung

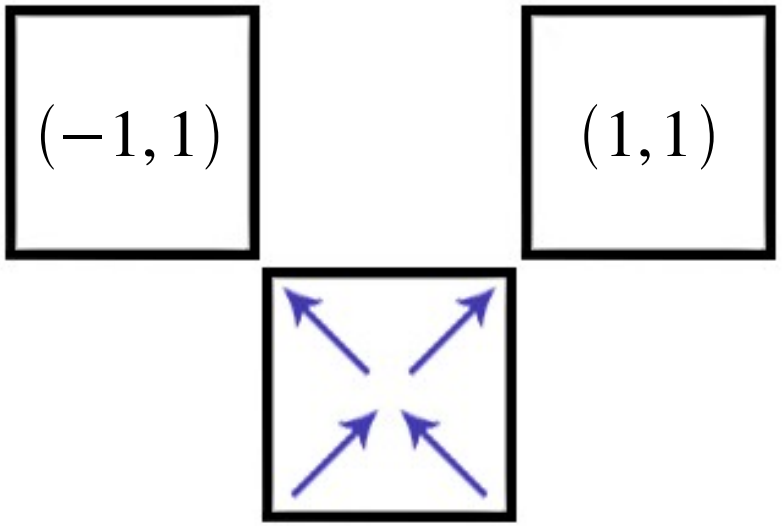




- Alle Ausgänge bekommen eine 2er-Potenz (1,2,4,8)
- Ein Vertex wird nun beschrieben aus der Summe der Ausgangs-Zahlen
- Nun ist jede der 6 Vertizes eindeutig identifiziert
- Will man nun entlang eines Loops die Pfeile klappen, muss man nur den jeweilig benutzen Ausgang abziehen und den Eingang addieren

Ungewichteter Loop

Implementierung



- Um nun einen Pfad zu finden muss man lediglich den Pfeilen der Plaketten folgen
- Hierzu kann man den Pfeilen Änderungen der momentanen Position zuordnen
- Dies sieht für die Plakette 3 z.B. so aus:

Plakette 3 => Ausgänge 1 und 2

Ausgang 1 => $d\vec{r}_1 = (-1, 1)$

Ausgang 2 => $d\vec{r}_2 = (1, 1)$

- Realisierung mittels Look-Up-Table

Ungewichteter Loop

Implementierung

- Während der Pfadsuche wird jeweils der verwendete Ein- und Ausgang gespeichert sowie die Plakette markiert

- Dadurch, dass der Pfad nicht vom Startpunkt aus eindeutig festgelegt ist, wird ein Randomwalk durch das Gitter entlang der Pfeile gemacht

- Jeder Ausgang wird mit Wahrscheinlichkeit ½ benutzt (sofern beide möglich sind)

- Durchkreuzen der Linien (2-maliges Besuchen einer Plakette) ist verboten

- Dadurch sind Sackgassen möglich; sollte der Pfad in eine Sackgasse geraten wird er verworfen und neu angefangen.

Ungewichteter Loop

Implementierung

```
Solange kein geschlossener Pfad {
        Loesche den Pfad
        waehle zufaellige weisse Plakette
        Markiere die Plakette als Startpunkt
        suche Pfad
}

w_alt berechnen
entlang des Pfades klappen
w_neu berechnen

w = w_neu/w_alt
p = w / (1+w)

wenn random > p {
        entlang des Pfades zurueckklappen
}
Messe Observablen
```

- Pfadsuche ist rekursiv
- Pfadsuche muss determinieren, auch wenn sie keinen geschlossenen Pfad findet
- Pfadsuche muss Indizi dafür liefern, ob sie erfolgreich war oder nicht
- Wieder wird erst geklappt und bei nicht akzeptieren zurückgeklappt

Ungewichteter Loop

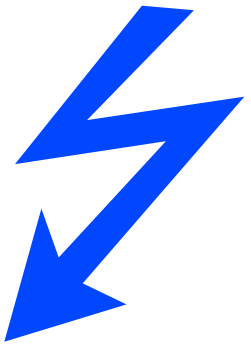
- **Problem: Gewichte**

- Durch die Randomwalk Analogie ist am Startpunkt weder Länge **noch Abschluss des Pfades bekannt**

- Deshalb kann nur schwer eine Wahrscheinlichkeit für einen bestimmten Pfad angegeben werden

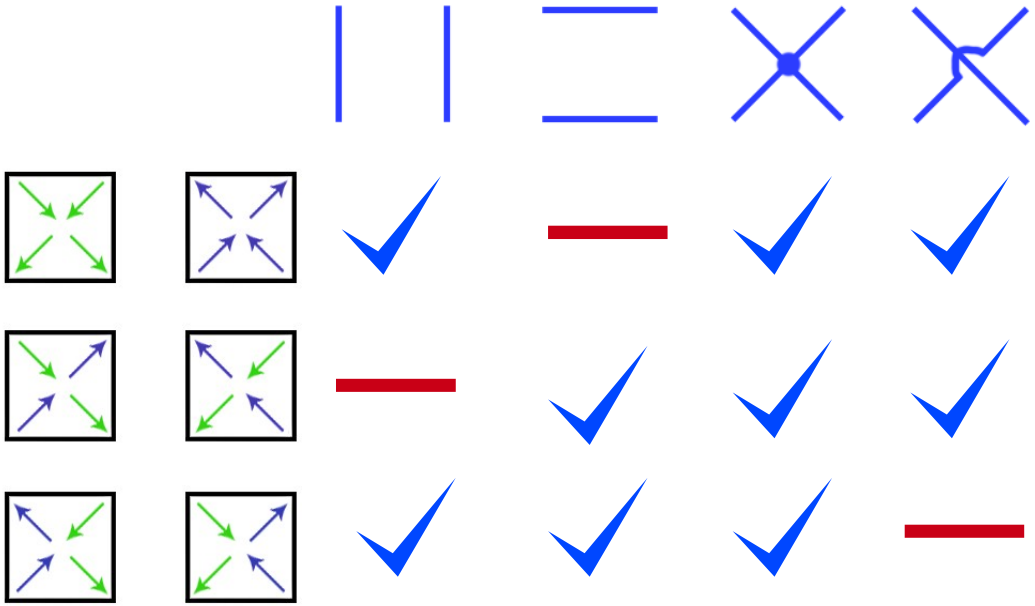
- Entsprechend ist detailliertes Gleichgewicht nicht gewährleistet

- Selbst wenn Detailbalance, wäre Akzeptanzrate sehr gering



Loop-updates

Algorithmus



- Verlagerung des **Akzeptanzschrittes auf Konstruktion der Schleife**
- Führe sog. „**Graphen**“ ein, die mit bestimmten Wahrscheinlichkeiten gesetzt werden
- Graphen sind ungerichtet (entgegen der Vertizes)
- Graphen entsprechen den möglichen Vertex-Pfaden einer Plakette

Loop-updates

Algorithmus

- Gewichte ergeben sich aus dem Gleichungssystem

$$1.\ \sum_G W(S,G) = W(S)$$

$$2.\ W(S,G) = W(S',G) \qquad \forall\, S, S'\ kompatibel\ mit\ G$$

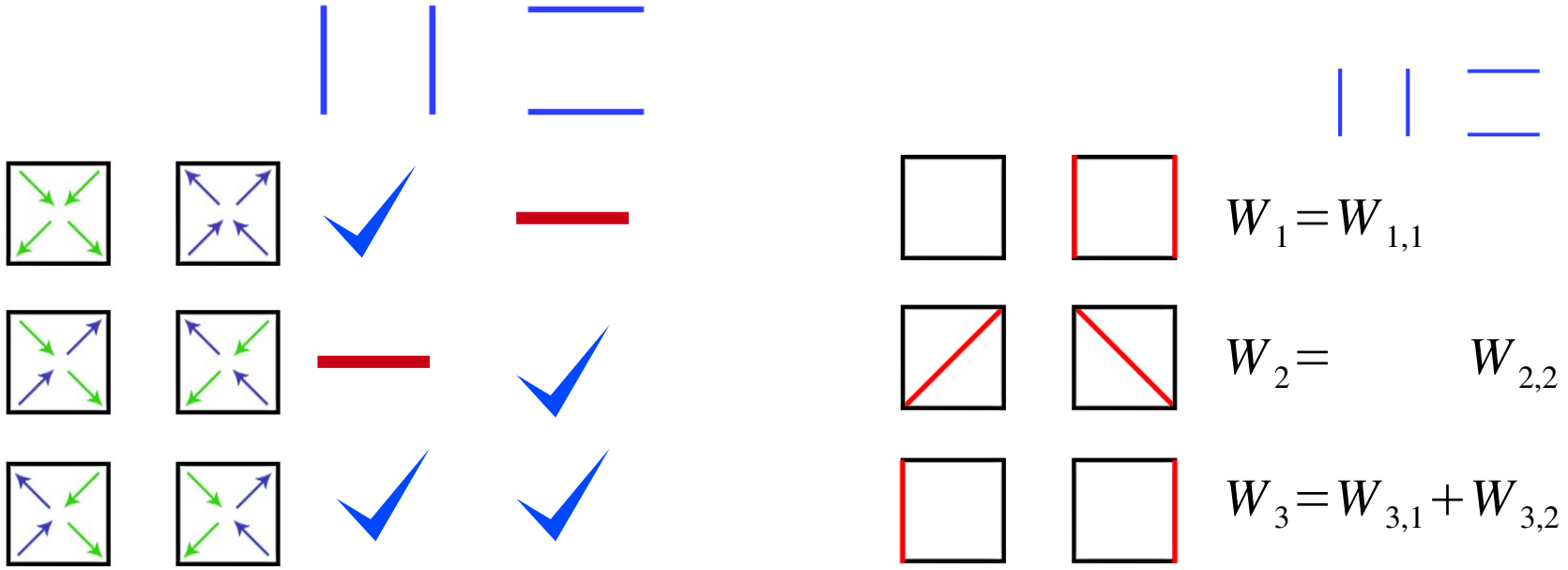
Probleme:

- W(S,G) nicht eindeutig
- Gefrorene Graphen erzeugen zu große Cluster

Loop-updates

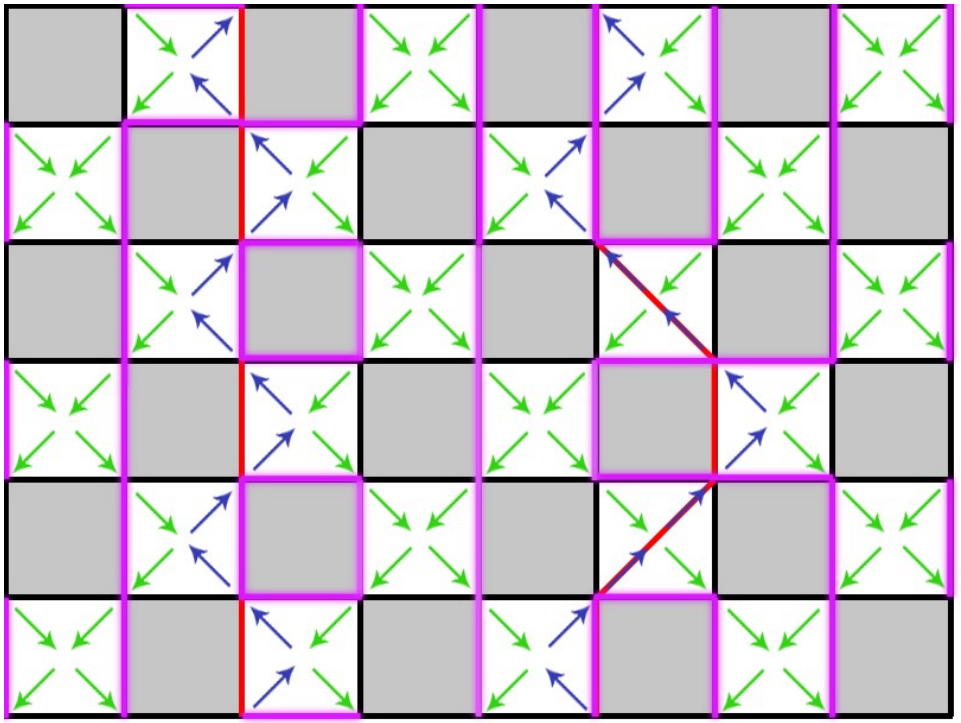
Algorithmus

- Wir betrachten Fall mit zwei Graphen
- Wobei die Graphen bei 4 der 6 möglichen Konfigurationen eindeutig sind
- Nur ferromagnetische Plaketten haben beide Möglichkeiten



Loop-updates

- Algorithmus

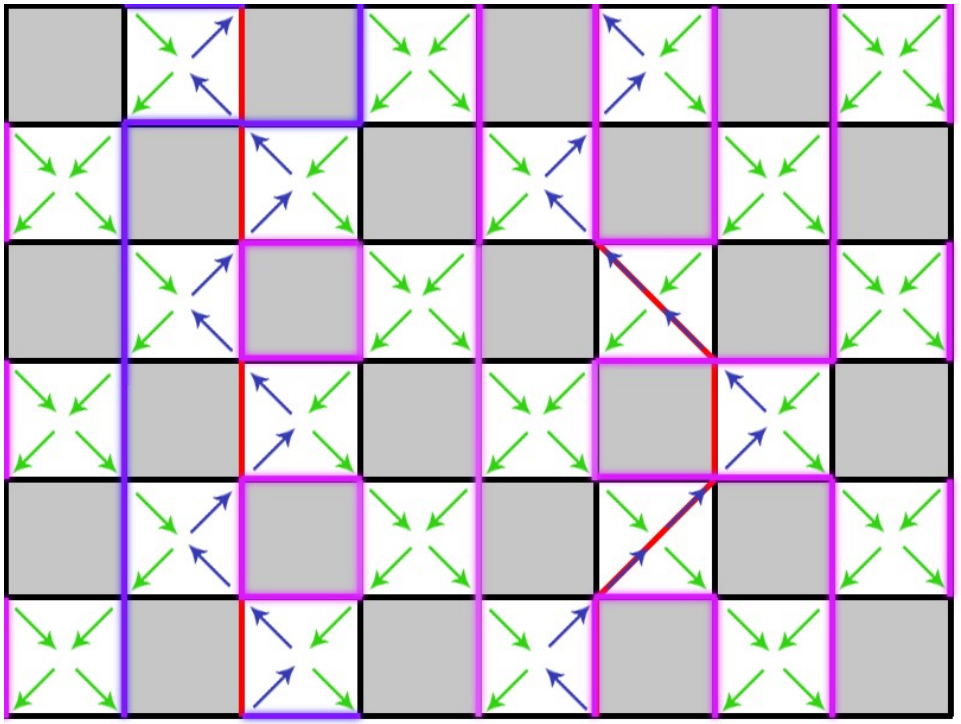


$$P(\square\square, \| \; \|) = \tanh(\Delta\tau \frac{J}{2})$$

- Die Graphen sind ungerichtet und spannen eindeutige Schleifen auf

Loop-updates

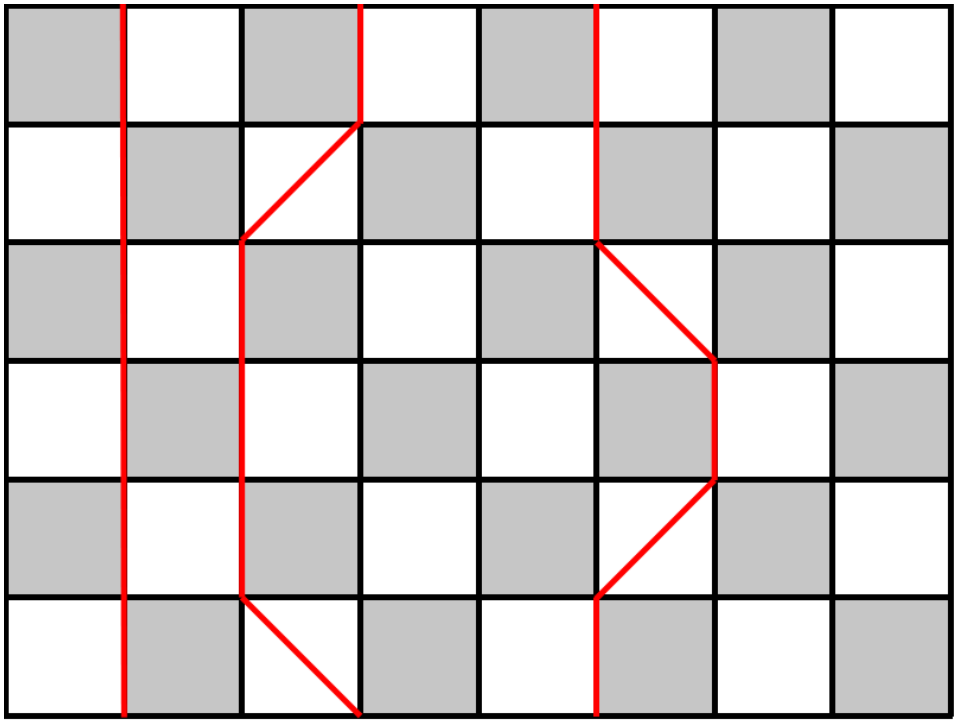
- Algorithmus



- Die Graphen sind ungerichtet und spannen eindeutige Schleifen auf
- Nun wählen wir eine oder mehrere Schleifen aus

Loop-updates

Algorithmus



- Die Graphen sind ungerichtet und spannen eindeutige Schleifen auf
- Nun wählen wir eine oder mehrere Schleifen aus
- Flippen aller Pfeile entlang der Graphen-Schleife und rekonstruieren Weltlinien

Loop-updates

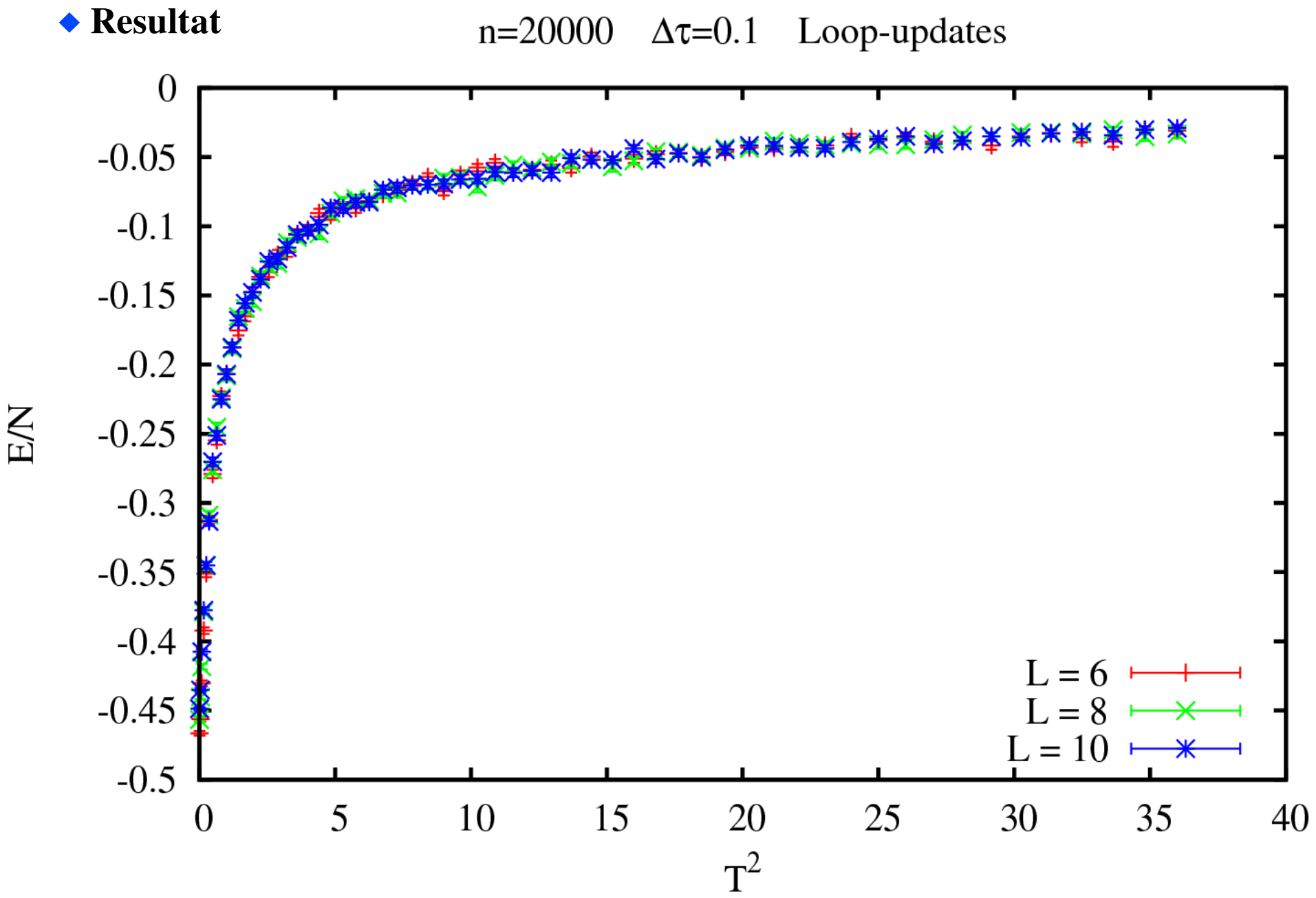
Implementierung

```
Fuer alle Plaketten {
        identifiziere Weltlinienkonfiguration
        weise Graphen zu
}

waehle Startplakette/Kante/Richtung
markiere Startplakette entsprechend

folge der Schleife bis Startpunkt {
        klappe Vertex-Pfeile
}

Messe Observablen
```

- Kompletter MC-Anteil liegt in Graphenzuweisung
- In der Tat ist das Folgen der Schleife einfach, z.B. mittels Look-Up-Table
- Da Schleife eindeutig können direkt Pfeile geklappt werden

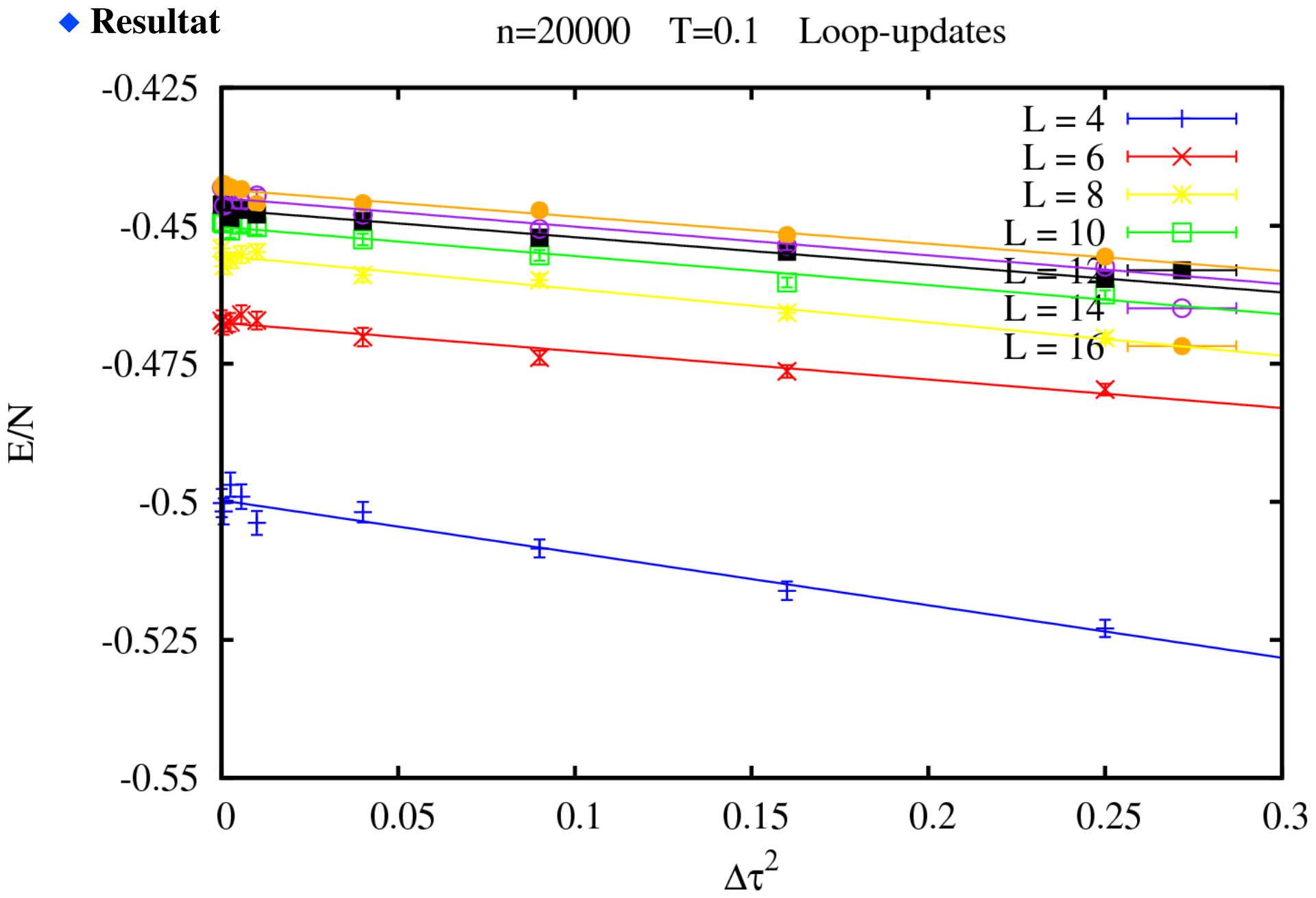
Loop-updates

- Resultat



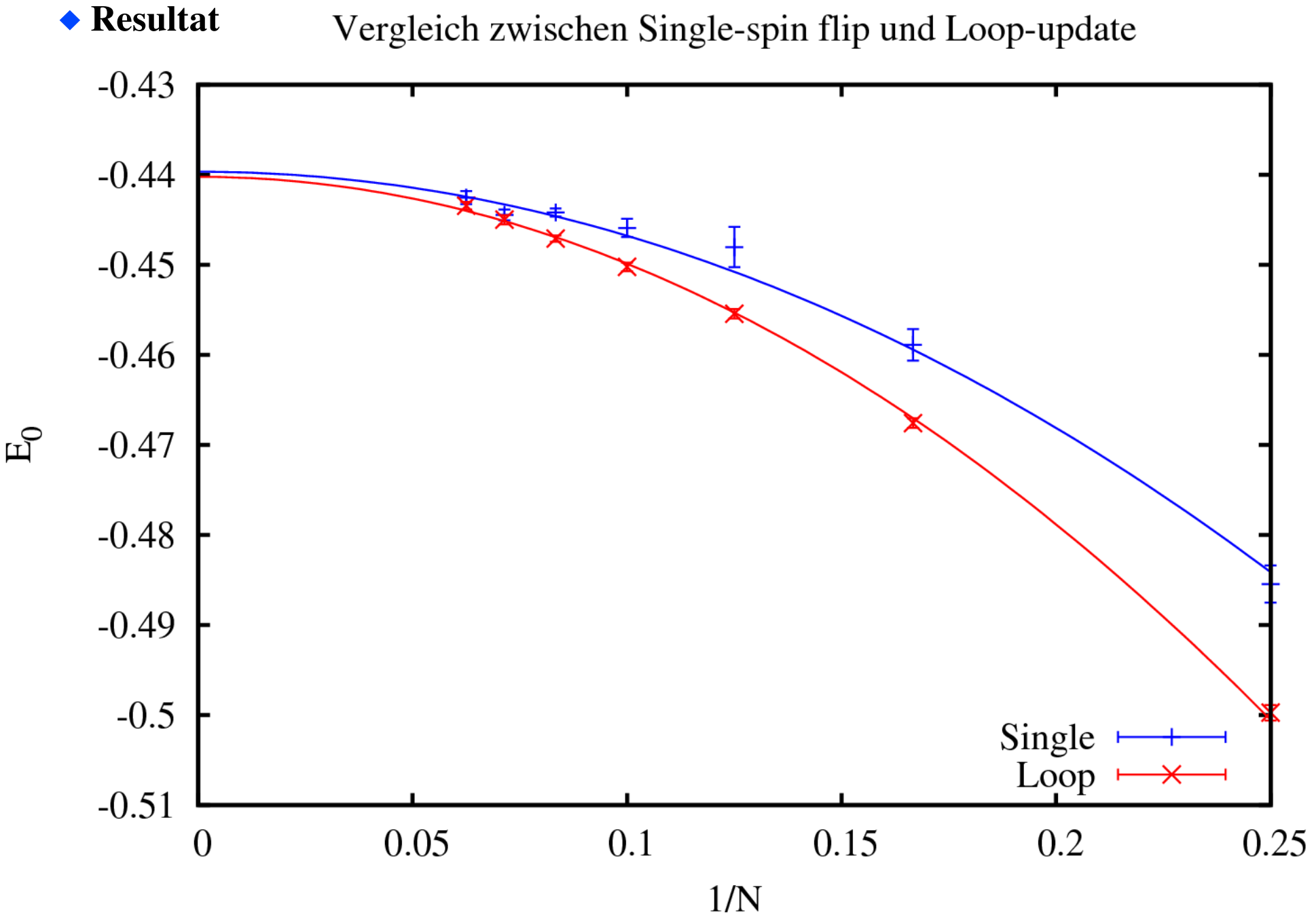
Loop-updates

- Resultat



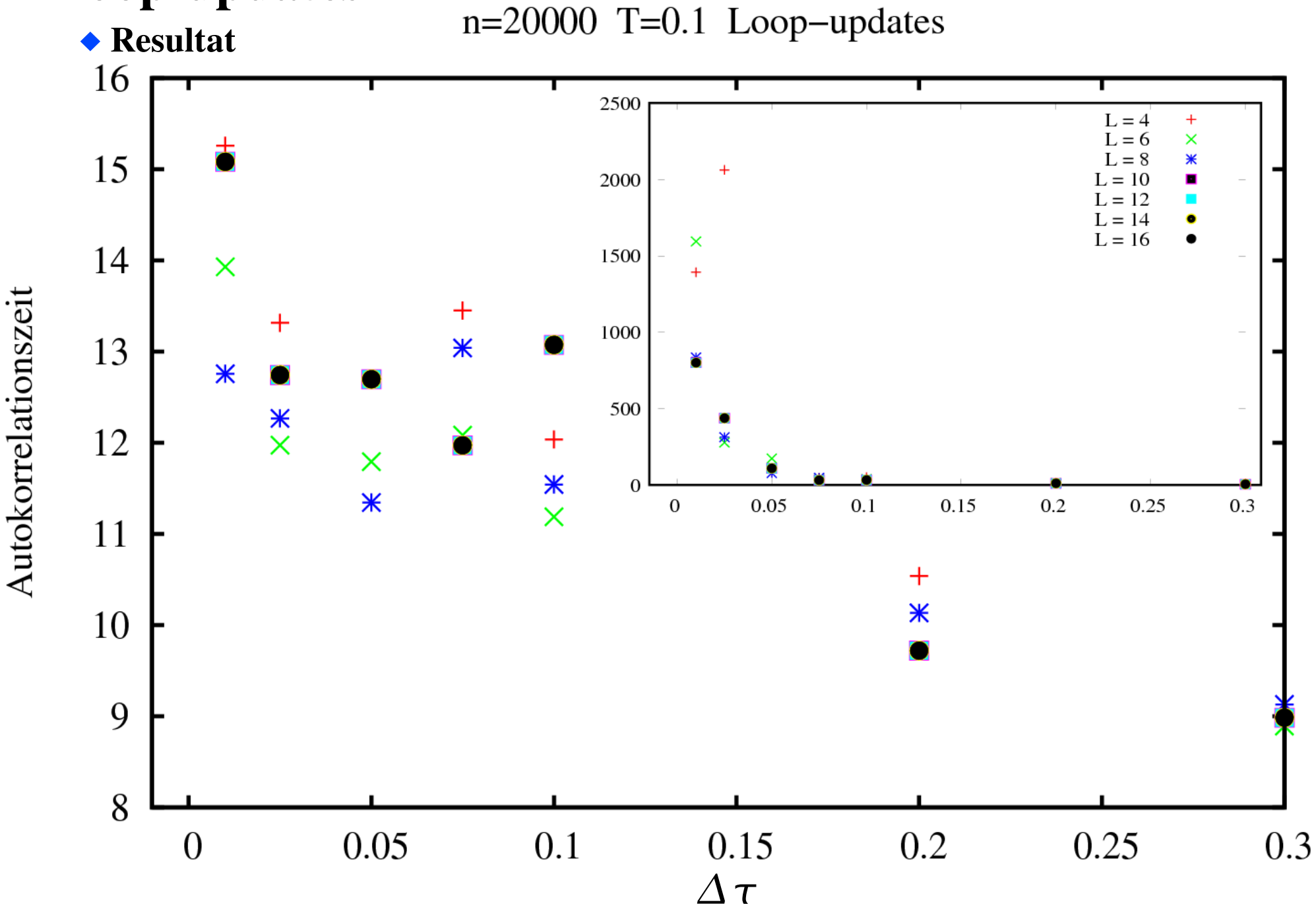
Loop-updates

- Resultat



Loop-updates

- Resultat



Alps Looper

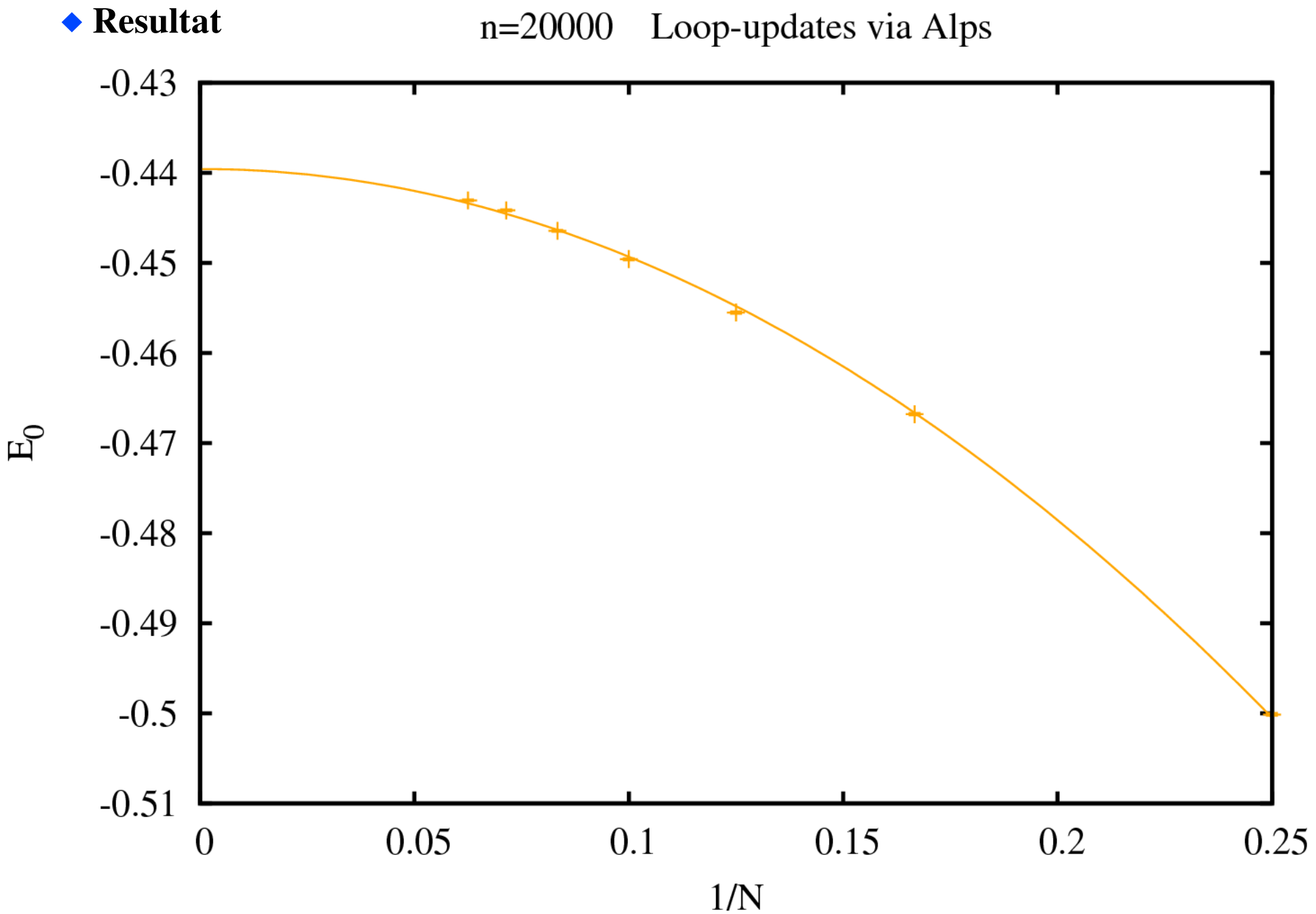
Verwendung

```
LATTICE="chain lattice"
  MODEL="spin"
  local_S=1/2
  T=0.1
  J=1
  THERMALIZATION=15000
  SWEEPS=20000
  REPRESENTATION="SSE"
  { L=2;}
  { L=4;}
  { L=6;}
  { L=8;}
  { L=10;}
  { L=12;}
  { L=14;}
  { L=16;}
```

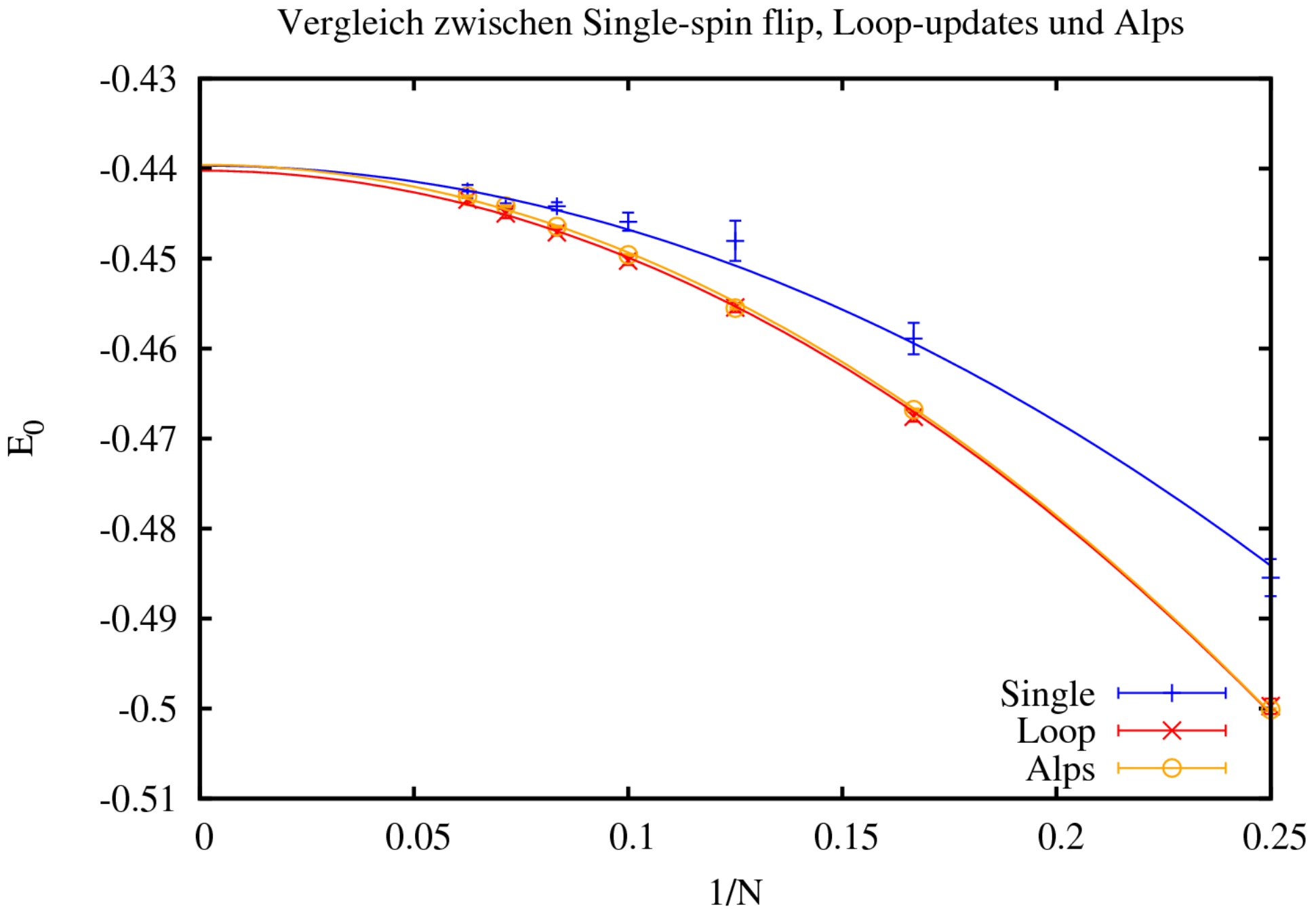
- Die Alps Anwendungen haben bereits einen Loop-Algorithmus implementiert
- Zur Verwendung wird lediglich eine Parameterdatei angelegt und das Programm gestartet
- Alps scheint einen „continuous-time“ Algorithmus zu verwenden, da sowohl Konfig- als auch Ergebnis-Datei keine Hinweise auf Zeit-Diskretisierungsschritte zulassen
- Für 2D und 3D Gitter verwende „square lattice“ und „cubic lattice“

Alps Looper

- Resultat



Vergleich


Vergleich zwischen Single-spin flip, Loop-updates und Alps
-0.43
-0.44
-0.45
-0.46
-0.47
-0.48
-0.49
-0.5
-0.51
E0
0
0.05
0.1
0.15
0.2
0.25
1/N
Single
Loop
Alps

Alps Looper

- Resultat