Vom Dreiecksbaum zur Gravitationskraft

Erste Implementierung und nachvollziehbare Prüfung

1 Was diese erste Version macht

Wir behalten den bisherigen konformen CFK-Baum vollständig bei und ergänzen eine zweite Ebene: Jedes Teilchen bekommt eine Masse; jeder belegte Teilbaum bekommt eine Gesamtmasse und einen Massenschwerpunkt. Damit können wir weit entfernte Gruppen bei der Kraftberechnung durch jeweils eine einzige Quelle ersetzen. Das ist die Grundidee von Barnes–Hut.

Die Teilchen bleiben vorerst an ihren echten Positionen. Es gibt weder eine Verschiebung zum Dreiecksmittelpunkt noch eine flächige Massenverteilung. Wir berechnen zunächst nur Beschleunigungen einer festen Konfiguration, noch keine Bewegung im Lauf der Zeit.

Die Positionen liegen in einer Ebene, aber wir verwenden das gewöhnliche dreidimensionale Newtonsche Gravitationsgesetz. Das ist nicht die eigenständige zweidimensionale Theorie mit logarithmischem Potential. Für ein Zielteilchen i lautet die direkte Referenzrechnung

$$\boldsymbol{a}_i = G \sum_{j \neq i} m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\left(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2 + \varepsilon^2\right)^{3/2}}.$$

Standardmässig gilt $G = 1$, $m_j = 1/n$ und $\varepsilon = 0$. Die Gesamtmasse ist damit eins. Individuelle positive Massen sind bereits unterstützt und werden dann unverändert verwendet. Das Ergebnis ist eine *Beschleunigung*; für die Kraft muss noch mit der Zielmasse m_i multipliziert werden. Es gibt keine periodischen Bilder und keine Randkräfte.

1.1 Masse von unten nach oben

Ein belegtes Blatt kennt Masse und Position seines Teilchens. Für einen inneren Knoten S mit belegten Kindern L, R gilt

$$M_S = M_L + M_R, \qquad \boldsymbol{c}_S = \frac{M_L \boldsymbol{c}_L + M_R \boldsymbol{c}_R}{M_S}.$$

Leere Teilbäume tragen nichts bei. Existiert nur ein belegtes Kind, werden dessen Masse und Schwerpunkt übernommen. Jeder Schwerpunkt wird einmal beim Aufbau berechnet und anschliessend wiederverwendet, nicht bei jeder Begegnung mit einem Zielteilchen. Der Kraftbaum speichert nur massetragende Knoten; die leeren Zellen des konformen Gitters bleiben im ursprünglichen Baum erhalten.

1.2 Warum zunächst ohne Glättung?

Verschiedene Teilchenpositionen reichen für diese statische Rechnung aus. Ein Teilchen pro Blatt garantiert allerdings keinen Mindestabstand. Sehr nahe Paare erzeugen deshalb sehr grosse Beschleunigungen. Ein Test mit Abstand 10^{-20} und Einheitsmassen reproduziert 10^{40}. Eine optionale Plummer-Glättung ist vorgesehen, in allen grossen Versuchen aber ausgeschaltet. Ihre spätere Wahl ist eine physikalische Modellentscheidung, keine Voraussetzung des CFK-Baums.

2 Wie entschieden wird, ob ein Knoten geöffnet wird

Für jedes Zielteilchen beginnt die Suche an der Wurzel. Ein entfernter Teilbaum darf als Punktmasse M_S an seinem Schwerpunkt $\boldsymbol{c}_S$ wirken. Andernfalls besuchen wir seine Kinder. An einem fremden Blatt wird die Teilchenwechselwirkung direkt ausgewertet. Das eigene Blatt wird ausgelassen.

2.1 Was der Integer-Code unmittelbar liefert

Die Wurzel hat Code 1, die Kinder von k haben Codes $2k$ und $2k+1$. Die Tiefe ist $d = \text{bitlength}(k) - 1$. Für unsere Wurzel $(0,0),(2,0),(0,2)$ sind Fläche und quadrierter Durchmesser

$$A(k) = 2^{1-d}, \qquad D(k)^2 = 2^{3-d}.$$

Ein Beispiel: $5 = (101)_2$ ist Vorfahr von $22 = (10110)_2$, denn $22 \texttt{>>} 2 = 5$. Mit diesem Präfixvergleich erkennen wir, ob ein Knoten das Zielblatt enthält. Ein solcher Knoten wird *immer* geöffnet, damit die eigene Masse nicht in einer angenommenen Gruppe mitwirkt.

2.2 Variante A: klassischer geometrischer Test

Eine Gruppe wird angenommen, falls sie das Ziel nicht enthält und

$$D_S^2 < \theta^2 \|\boldsymbol{x}_i - \boldsymbol{c}_S\|^2.$$

Kleineres θ bedeutet strengeres Öffnen. Im Test selbst gibt es keine Wurzel, Division oder trigonometrische Winkelberechnung. Die Komponenten des Abstands werden bei Annahme direkt für die Kraft wiederverwendet.

2.3 Variante B: konservativer Test aus den Codes

Ein anderer Binärpräfix bedeutet noch keinen grossen Abstand: Zwei Zellen können an einer Kante aneinanderliegen. Deshalb rekonstruieren wir aus den Codes einmalig achsenparallele Begrenzungsrechtecke. Diese umschliessen die Dreiecke und haben exakt darstellbare dyadische Koordinaten.

Mit h als grösster Tiefe eines belegten Blattes werden die Koordinaten mit $s = 2^h$ skaliert. Sämtliche Eckpunkte sind dann ganzzahlig. Kinder entstehen aus dem Kantenmittelpunkt mittels Addition und Rechtsverschiebung um ein Bit. Seien B_T, B_S die Rechtecke von Zielblatt und Quelle. In jeder Achse ist der Rechteckabstand die positive Lücke zwischen den Intervallen, sonst null:

$$\Delta_x = \max(0, x_T^- - x_S^+, x_S^- - x_T^+), \quad \Delta_y = \max(0, y_T^- - y_S^+, y_S^- - y_T^+), \quad \delta_g^2 = \Delta_x^2 + \Delta_y^2.$$

Für den eingegebenen Gleitkommawert $\theta = p/q$ wird ausschliesslich mit Integern entschieden:

$$2^{3-d+2h} q^2 < p^2 \delta_g^2.$$

Die Rechtecklücke ist eine untere Abstandsschranke: Zielposition und Quellschwerpunkt liegen in ihren jeweiligen Dreiecken. Daher ist dieser Test beim gleichen θ konservativer als Variante A (abgesehen von Gleitkommarundung der Schwerpunkte). Berührende Rechtecke erzwingen Absteigen.

Einordnung: Das ist exakte, zwischengespeicherte Integer-Geometrie aus Codes, nicht ein vermeintlicher Abstand aus XOR allein. Auch ganzzahlige Multiplikationen und zusätzliche Knotenbesuche kosten Zeit. Keine der beiden Regeln garantiert unmittelbar eine bestimmte relative Kraftgenauigkeit.

3 Versuch mit jeweils 100000 Teilchen

Wir verwenden drei feste Wolken: uniform im Einheitsquadrat, eine schmale Gauss-Wolke um $(0.5, 0.5)$ und zwei Gauss-Wolken um $(0.3, 0.5)$ sowie $(0.7, 0.5)$ mit 90:10-Aufteilung. Die Gauss-Breite beträgt jeweils $\sigma = 0.025$ pro Koordinate; ausserhalb des Quadrats liegende Stichproben würden paarweise neu gezogen. Alle Massen sind $1/n$.

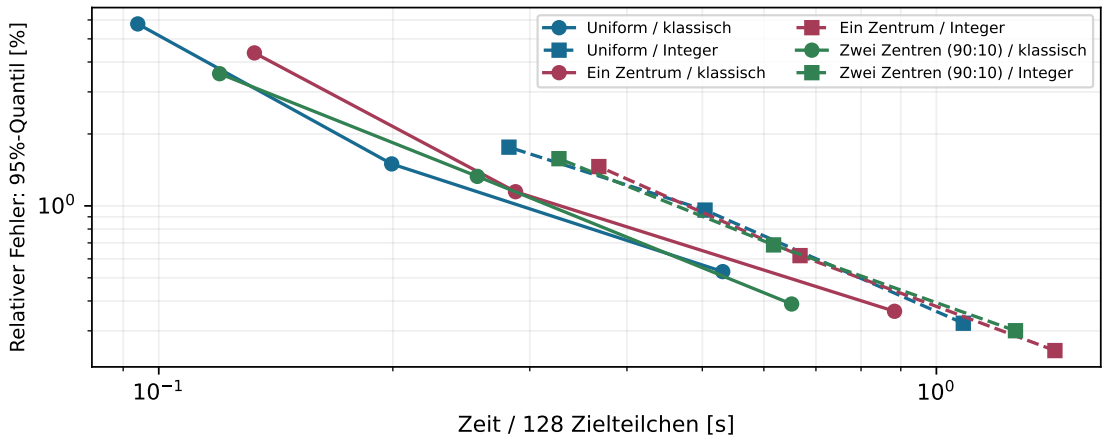
Pro Wolke wählen wir 128 Zielteilchen ohne Zurücklegen. Für jedes dieser Ziele summiert die direkte Referenz *alle 100 000 Quellen*. Beide Baumvarianten verwenden dieselben Eingaben und Zielindizes. Wir prüfen $\theta = 0.3, 0.5, 0.8$ und wiederholen nur die Zeitmessung dreimal in gemischter Reihenfolge; die Kräfte sind dabei identisch. Das sind **keine drei unabhängigen Teilchenrealisierungen**.

Wolke	Blätter	Gitter [s]	Vorbereitung [s]	Direkt [s]
Uniform	334 282	4.09	5.15	1.51
Ein Zentrum	329 978	3.95	5.48	1.85
Zwei Zentren (90:10)	330 076	3.78	4.84	1.39

Die Vorbereitung umfasst Blattzuordnung, Massenaggregate und Integer-Rechtecke. Sie wird für beide Varianten gemeinsam bezahlt; eine rein klassische Implementierung könnte die Rechtecke weglassen. Gitteraufbau mit bis zu vier Prozessen einschliesslich Startkosten, Kraftsuche derzeit seriell.

Wolke	Kriterium	Zeit [s]	Fehler Q_{95} [%]	Max. [%]	Besuche/Ziel
Uniform	Klassisch	0.199	1.498	3.757	1057
Uniform	Integer	0.504	0.960	1.722	1586
Ein Zentrum	Klassisch	0.288	1.146	1.977	1468
Ein Zentrum	Integer	0.668	0.619	1.319	2120
Zwei Zentren (90:10)	Klassisch	0.257	1.329	7.635	1394
Zwei Zentren (90:10)	Integer	0.617	0.687	2.387	2018

Diese Tabelle gilt für $\theta = 0.5$. Zeiten sind Mediane der drei Messungen für 128 Ziele, *nicht* für alle Teilchen. Der relative Fehler ist $\|\boldsymbol{a} - \boldsymbol{a}_{\text{direkt}}\| / \max(\|\boldsymbol{a}_{\text{direkt}}\|, 10^{-12} a_{\text{RMS}})$. Q_{95} bezeichnet das 95%-Quantil der 128 Fehler, kein Konfidenzintervall.



Je Kurve stehen die drei Punkte für die drei getesteten Toleranzen. Links und unten bedeutet günstiger. Die Verbindungslinien dienen nur der Lesbarkeit und liefern keine Messungen zwischen den Punkten.

Erster Befund: Bei $\theta = 0.5$ ist das Integer-Kriterium 2.32- bis 2.53-mal langsamer; sein 95%-Fehlerquantil liegt aber auch 36–48% niedriger. Es besucht etwa 44–50% mehr Knoten. Die Codierung liefert somit ein funktionierendes konservatives Kriterium, in dieser Python-Implementierung jedoch keine Beschleunigung beim gleichen θ. Ein fairer Optimierungsvergleich muss gleiche erzielte Genauigkeit anstreben, nicht bloss gleiche Toleranz. Die gesamte Messserie dauerte rund 64 Sekunden.

4 Prüfung, Benutzung und Grenzen

4.1 Was geprüft wurde

Die neun neuen Tests decken leere Eingaben, ein einzelnes Teilchen, analytische Zweikörperkräfte, optionale Glättung, individuelle Massen, Schwerpunkte, Ausschluss der Selbstkraft, genaue Rechtecke aus den Codes, sehr nahe Paare und ungültige Eingaben ab. Beim vollständigen Absteigen ($\theta = 0$) stimmt die Baumrechnung bis auf Gleitkomma-Summationsfehler mit direkter Summation überein. Die Massenskalierung und die Impulsbilanz der direkten Kräfte werden ebenfalls getestet.

Die gesamte bestehende Testsammlung wurde erneut ausgeführt. Die grossen Versuche sind zusätzliche Kraftvergleiche, keine erneute vollständige Konformitätsprüfung. Das zugrunde liegende Gitterverfahren bleibt unverändert. **Ergebnis:** Alle 66 Tests bestehen, einschliesslich der neun neuen Gravitationstests. Alle 55 Dateien des archivierten Versuchs wurden gegen ihre Prüfsummen geprüft; die 18 Fehlerauswertungen wurden aus den gespeicherten Kräften erneut berechnet. Die alten 8808 Dateien (ohne generierte Python-Caches) sind gegenüber der Sicherung unverändert.

4.2 Ein kleines Anwendungsbeispiel

Die neue Datei `gravity.py` liegt neben den bisherigen Gittermodulen. Der Hauptprogramm-Schutz im Beispiel ermöglicht auch den parallelen Gitterbau.

```
from gravity import GravityTree, direct_accelerations

if __name__ == "__main__":
    points = [(0.2, 0.3), (0.7, 0.6), (0.4, 0.8)]
    tree = GravityTree.build(points, workers=4)
    a, stats = tree.accelerations(theta=0.5, mode="classic")
    b, stats_b = tree.accelerations(theta=0.5, mode="code")
    exact = direct_accelerations(points)
    print(a)
```

Mit `masses=[1,2,3]` beim Aufbau werden individuelle Massen gesetzt. `targets=[0,2]` beschränkt nur die Ziele, nicht die Quellen. `tree.nodes[code]` liefert die gespeicherten Eigenschaften eines massetragenden Knotens. Bei geänderten Positionen oder Massen muss diese Momentaufnahme neu aufgebaut werden.

4.3 Reproduzierbarkeit und Wiederherstellung

`benchmark_gravity.py` erzeugt die Versuchsserie. Der Startwert ist 2026091705; NumPy-PCG64 verwendet pro Wolke eine eigene SeedSequence. Gespeichert werden alle Positionen, Zielindizes, Referenz- und Baumkräfte, Parameter, Einzelzeiten, Besuchszahlen, Softwareversionen und Quellcode. SHA-256-Prüfsummen sichern den Stand. Die Berichtsdaten liegen im Unterordner `data`; `analyze.py` prüft sie und erzeugt Tabellen und Grafik erneut. Die CSV-Datei enthält alle 18 Vergleiche, auch absolute Fehler.

Vor dem ersten Eingriff wurden `CFK/2D` und `CFK/Berichte` vollständig archiviert. Das Archiv liegt unter `CFK/Sicherungen/2026-09-17_vor_Gravitation`; die dortige Anleitung beschreibt eine Wiederherstellung ohne Überschreiben neuer Arbeit. Ein früher Probelauf bleibt separat als `gravity_g4_pilot` erhalten und geht nicht in diesen Bericht ein. Die ausgewertete Serie lief ohne gleichzeitig von uns gestartete Tests.

4.4 Was noch nicht gezeigt ist

Die drei Wolken und 128 Ziele pro Wolke belegen Funktionsfähigkeit, keine allgemeine Fehlerschranke oder gesicherte Skalierung. Nahe Begegnungen und Kraftauslöschung können relative Fehler stark verändern. Die einseitige Barnes–Hut-Näherung erzwingt keine exakt paarweise symmetrischen Kräfte. Zeitintegration, Energieerhaltung, bewegte Bäume und homogene Dreiecksquellen sind noch nicht implementiert. Gitterkoordinaten bleiben exakt, Massen und Kräfte werden in doppelter Gleitkommagenauigkeit berechnet; dort zusammenfallende Positionen oder nicht darstellbare Beschleunigungen werden zurückgewiesen.

Hintergrund: J. Barnes, Treecode Guide; J. Barnes, Gravitational softening as a smoothing operation (2012).