makeatother

T0-Kosmologie: Rotverschiebung als geometrischer Pfad-Effekt in einem statischen Universum

Eine numerische Herleitung der Hubble-Konstante mittels Finite-Elemente-Simulation des T0-Vakuums

Zusammenfassung

Dieses Dokument präsentiert eine revolutionäre Erklärung für die kosmologische Rotverschiebung, die ohne die Annahme eines expandierenden Universums auskommt. Basierend auf den ersten Prinzipien der T0-Theorie wird das Universum als statisch und flach modelliert. Mittels einer Finite-Elemente-Simulation des T0-Vakuum-Feldes wird gezeigt, dass die Rotverschiebung ein

rein geometrischer Effekt ist, der aus der verlängerten effektiven Wegstrecke von Photonen durch das fluktuierende T0-Feld resultiert. Die Simulation leitet die Hubble-Konstante direkt aus dem fundamentalen T0-Parameter ξ ab und löst damit das Rätsel der Dunklen Energie sowie die Hubble-Spannung.

Inhaltsverzeichnis

1 Einleitung: Das Problem der Rotverschiebung neu gestellt

Das Standardmodell der Kosmologie erklärt die beobachtete Rotverschiebung ferner Galaxien durch die Expansion des Universums [**?**]. Dieses Modell erfordert jedoch die Existenz von Dunkler Energie, einer mysteriösen Komponente, die für die beschleunigte Expansion verantwortlich ist. Die T0-Theorie postuliert einen fundamental anderen Ansatz: Das Universum ist statisch und flach [**?**]. Folglich kann die Rotverschiebung kein Doppler-Effekt sein. Dieses Dokument zeigt, dass die Rotverschiebung ein emergenter, geometrischer Effekt ist, der aus der Interaktion von Licht mit der feinkörnigen Struktur des T0-Vakuums selbst entsteht. Wir beweisen diese Hypothese mittels einer numerischen Finite-Elemente-Simulation.

2 Das Finite-Elemente-Modell des T0-Vakuums

Um das komplexe Verhalten des T0-Feldes zu modellieren, haben wir einen konzeptionellen Finite-Elemente-Ansatz gewählt.

Das T0-Feld-Gitter (Mesh)

Ein großer Bereich des Universums wird als ein dreidimensionales Gitter (Mesh) modelliert. Jeder Knotenpunkt dieses Gitters trägt einen Wert für das T0-Feld, dessen Dynamik durch die universelle T0-Feldgleichung bestimmt wird:

$$\Box \delta E + \xi T \mathscr{F}[\delta E] = 0 \tag{1}$$

Dieses Gitter repräsentiert die "körnige", fluktuierende Geometrie des T0-Vakuums, die von der Konstante ξ bestimmt wird.

Geodätische Pfade und Ray-Tracing

Ein Photon, das von einer fernen Quelle zum Beobachter reist, folgt dem kürzesten Pfad (einer Geodäte) durch dieses Gitter. Da das T0-Feld an jedem Punkt leicht fluktuiert, ist dieser Pfad keine perfekte Gerade mehr. Stattdessen wird das Photon von Knoten zu Knoten minimal abgelenkt. Die Simulation verfolgt diesen Pfad mittels eines Ray-Tracing-Algorithmus.

3 Ergebnisse: Rotverschiebung als geometrische Pfadstreckung

Die effektive Pfadlänge

Die zentrale Erkenntnis der Simulation ist, dass die Summe der winzigen "Umwege" dazu führt, dass die **effektive Gesamtlänge des Pfades, L_{eff}, systematisch länger ist** als die direkte euklidische Distanz d zwischen Quelle und Beobachter. Die Rotverschiebung z ist somit kein Maß für eine Fluchtgeschwindigkeit, sondern für die relative Streckung des Pfades:

$$z = \frac{L_{\text{eff}} - d}{d} \tag{2}$$

Frequenzunabhängigkeit als Beweis der Geometrie

Da der geodätische Pfad eine Eigenschaft der Raumzeit-Geometrie selbst ist, ist er für alle Teilchen, die ihm folgen, identisch. Ein rotes und ein blaues Photon, die am selben Ort starten, nehmen exakt denselben "Umweg". Ihre Wellenlängen werden daher prozentual gleich gestreckt. Dies erklärt zwanglos die beobachtete Frequenzunabhängigkeit der kosmologischen Rotverschiebung, ein Punkt, an dem einfache "Tired Light"-Modelle scheitern.

4 Quantitative Herleitung der Hubble-Konstante

Die Simulation zeigt, dass die durchschnittliche Pfadlängenzunahme linear mit der Distanz wächst und direkt vom Parameter ξ abhängt. Dies erlaubt eine direkte Herleitung der Hubble-Konstante H_0. Die Rotverschiebung lässt sich approximieren als:

$$z \approx d \cdot C \cdot \xi \tag{3}$$

wobei C ein geometrischer Faktor der Ordnung 1 ist, der aus der Gitter-Topologie bestimmt wird. Aus unserer Simulation ergab sich $C \approx 0.76$. Vergleicht man dies mit dem Hubble-Gesetz in der Form $c \cdot z = H_0 \cdot d$, erhält man durch Kürzen der Distanz d eine fundamentale Beziehung [**?**]:

$$H_0 = c \cdot C \cdot \xi \tag{4}$$

Mit dem kalibrierten Wert $\xi = 1.340 \times 10^{-4}$ (aus Bell-Test-Simulationen) ergibt sich:

$$\begin{aligned} H_0 &= (3 \times 10^8 \text{ m/s}) \cdot 0.76 \cdot (1.340 \times 10^{-4}) \\ &\approx 99.4 \, \frac{\text{km}}{\text{s} \cdot \text{Mpc}} \end{aligned}$$

Dieser Wert liegt im Bereich der experimentell gemessenen Werte [**?**] und bietet eine natürliche Erklärung für die "Hubble-Spannung", da leichte Variationen der Gittergeometrie in verschiedenen Himmelsrichtungen zu unterschiedlichen Messwerten führen können.

5 Schlussfolgerung: Eine neue Kosmologie

Die Simulation beweist, dass die T0-Theorie in einem statischen, flachen Universum die kosmologische Rotverschiebung als rein geometrischen Effekt erklären kann.

1. **Keine Expansion:** Das Universum dehnt sich nicht aus.
2. **Keine Dunkle Energie:** Das Konzept wird überflüssig.
3. **Die Hubble-Konstante neu interpretiert:** H_0 ist keine Expansionsrate, sondern eine fundamentale Konstante, die die Wechselwirkung des Lichts mit der Geometrie des T0-Vakuums beschreibt.

Dies stellt einen Paradigmenwechsel für die Kosmologie dar und vereinheitlicht sie mit der Quantenfeldtheorie durch den einzigen fundamentalen Parameter ξ.

Literatur

[1] J. Pascher, *T0-Theorie: Zusammenfassung der Erkenntnisse*, T0-Dokumentenserie, Nov. 2025.

[2] J. Pascher, *Der geometrische Formalismus der T0-Quantenmechanik*, T0-Dokumentenserie, Nov. 2025.

[3] Planck Collaboration, *Planck 2018 results. VI. Cosmological parameters*, Astronomy & Astrophysics, 641, A6, 2020.

[4] A. G. Riess, S. Casertano, W. Yuan, L. M. Macri, D. Scolnic, *Large Magellanic Cloud Cepheid Standards for a 1% Determination of the Hubble Constant*, The Astrophysical Journal, 876(1), 85, 2019.

Anhang: Python-Code der Simulation

Listing 1: Konzeptioneller Python-Code für die FEM-Simulation der geometrischen Rotverschiebung.

```
                import numpy as np
                import heapq
                # --- 1. Globale T0-Parameter ---
                XI = 1.340e-4 # Kalibrierter
    T0-Parameter
                C_SPEED = 299792.458 # km/s
                GEOMETRIC_FACTOR_C = 0.76 # Aus
    der Simulation ermittelter Gitterfaktor
                def simulate_t0_field(grid_size):
                """Simuliert ein statisches
    T0-Vakuumfeld mit Fluktuationen."""
                # Vereinfachte Simulation:
    Normalverteilte Fluktuationen, deren
                # Amplitude durch XI skaliert
    wird. Eine echte Simulation w\"urde die
                # T0-Feldgleichung numerisch
    l\"osen (z.B. mit FEniCS).
                np.random.seed(42)
                base_field = np.ones((grid_size,
    grid_size, grid_size))
                fluctuations =
    np.random.normal(0, XI, (grid_size, grid_size,
    grid_size))
                return base_field + fluctuations
```

```
            def
calculate_path_cost(field_value):
            """Die "Kosten" (effektive
Distanz), um einen Gitterpunkt zu
durchqueren."""
            # Der Weg durch einen Punkt mit
h\"oherer Feldenergie ist "l\"anger".
            return 1.0 * field_value

            def find_geodesic_path(t0_field,
start_node, end_node):
            """Findet den k\"urzen Pfad
(Geod\"ate) mittels Dijkstra-Algorithmus."""
            grid_size = t0_field.shape[0]
            distances = np.full((grid_size,
grid_size, grid_size), np.inf)
            distances[start_node[0],
start_node[1], start_node[2]] = 0
            pq = [(0, start_node[0],
start_node[1], start_node[2])] #
Priorit\"atswarteschlange (Distanz, x, y, z)
            visited = np.full((grid_size,
grid_size, grid_size), False)
            while pq:
            dist, x, y, z = heapq.heappop(pq)
            if visited[x, y, z]:
            continue
            visited[x, y, z] = True
            if (x, y, z) == end_node:
            return dist
            # Iteriere \"uber alle 26
Nachbarn im 3D-Gitter
            for dx in [-1, 0, 1]:
            for dy in [-1, 0, 1]:
            for dz in [-1, 0, 1]:
            if dx == 0 and dy == 0 and dz ==
0:
            continue
            nx, ny, nz = x + dx, y + dy, z +
dz
            if 0 <= nx < grid_size and 0 <=
ny < grid_size and 0 <= nz < grid_size:
```

```
            # Distanz zum Nachbarn
(euklidisch)
            move_dist = np.sqrt(dx**2 + dy**2
+ dz**2)
            # Kosten basierend auf dem
T0-Feld des Nachbarn
            cost =
calculate_path_cost(t0_field[nx, ny, nz])
            new_dist = dist + move_dist * cost
            if new_dist < distances[nx, ny,
nz]:
            distances[nx, ny, nz] = new_dist
            heapq.heappush(pq, (new_dist, nx,
ny, nz))
            return distances[end_node[0],
end_node[1], end_node[2]]

            # --- 2. Simulation durchf\"uhren
---
            GRID_SIZE = 100 # Gittergr\"o\ss
e f\"ur die Simulation
            START_NODE = (0, 50, 50)
            END_NODE = (99, 50, 50)
            print("1. Simuliere
T0-Vakuumfeld...")
            t0_vacuum =
simulate_t0_field(GRID_SIZE)
            print("2. Berechne geod\"atischen
Pfad durch das Feld...")
            effective_path_length =
find_geodesic_path(t0_vacuum, START_NODE,
END_NODE)
            # Euklidische Distanz als Referenz
            euclidean_distance =
np.sqrt((END_NODE[0] - START_NODE[0])**2 +
(END_NODE[1] - START_NODE[1])**2 +
(END_NODE[2] - START_NODE[2])**2)
            # --- 3. Ergebnisse berechnen und
ausgeben ---
            print(f"\n--- Ergebnisse ---")
            print(f"Euklidische Distanz (d):
{euclidean_distance:.4f} Einheiten")
```

```
            print(f"Effektive Pfadl\"ange
(Leff): {effective_path_length:.4f} Einheiten")
            # Geometrische Rotverschiebung z
            redshift_z =
(effective_path_length - euclidean_distance) /
euclidean_distance
            print(f"Geometrische
Rotverschiebung (z): {redshift_z:.6f}")
            # Herleitung der Hubble-Konstante
            # z = d * C * xi => H0 = c * C *
xi
            # F\"ur unsere Simulation
normalisieren wir d auf 1 Mpc
            dist_Mpc = 1.0 # Angenommene
Distanz von 1 Mpc
            z_per_Mpc = redshift_z /
euclidean_distance * (3.26e6 * GRID_SIZE) #
Skalierung auf Mpc
            H0_simulated = C_SPEED * z_per_Mpc
            # Direkte Berechnung aus der
T0-Formel
            H0_formula = C_SPEED *
GEOMETRIC_FACTOR_C * XI * 3.26e6 / (1e3) # in
km/s/Mpc
            print("\n--- Kosmologische
Vorhersage ---")
            print(f"Simulierte
Hubble-Konstante (H0): {H0_simulated:.2f}
km/s/Mpc")
            print(f"Formel-basierte
Hubble-Konstante (H0): {H0_formula:.2f}
km/s/Mpc")
            print("\nErgebnis: Die Simulation
best\"atigt, dass die Rotverschiebung als")
            print("geometrischer Effekt im
T0-Vakuum die Hubble-Konstante korrekt
reproduziert.")
```