デジタル宇宙論・技術補遺：数理モデルとPython実装

Digital Cosmology: Technical Appendix - Mathematical Models and Python Implementation

思考モード (Conceptual Architect's Assistant)

2025年11月24日

1　数理モデルによる予言 (Mathematical Predictions)

本節では、デジタル宇宙論（離散的情報処理モデル）から導かれる、既存の物理学とは異なる独自の予言を定式化する。

1.1　量子干渉の離散化 (Discrete Phase Counter Model)

粒子が持つ位相 ϕ は連続値ではなく、ビット深度 N を持つ離散値（整数）であると仮定する。 スクリーン上の位置 x における検出確率 $P(x)$ は、連続的な波動関数の二乗ではなく、以下の離散和で表される。

$$P(x) = \left| \sum_{path} \exp\left(\frac{2\pi i}{N} \lfloor \phi_{path}(x) \rfloor \right) \right|^2 \tag{1}$$

ここで $\lfloor \cdot \rfloor$ は床関数を表す。 **予言：** N が有限である場合、極低温・超精密な干渉実験において、干渉縞の強度分布は滑らかではなく、微細な**「階段状の量子化ノイズ」**を示すはずである。

1.2　重力赤方偏移の不感帯 (Energy Discretization Model)

エネルギーのやり取りもプランク単位 ϵ で離散化される。重力ポテンシャル差 $\Delta\Phi$ による振動数変化 $\Delta\nu$ は以下の通りとなる。

$$\Delta\nu_{discrete} = \left\lfloor \frac{\nu \Delta\Phi}{c^2 \epsilon} \right\rfloor \cdot \epsilon \tag{2}$$

予言： 重力差が極めて小さい領域（$\nu\Delta\Phi/c^2 < \epsilon$）では、エネルギー変化が最小単位を下回り、切り捨てが発生するため、**赤方偏移が全く発生しない「不感帯（Dead Zone）」**が観測される可能性がある。

1.3　計算リソース限界による揺らぎ (Computational Fluctuation)

宇宙の局所的な計算密度 C がシステムの限界 C_{max} に近づくと、物理定数（例：微細構造定数 α）の演算精度が低下する。

$$\alpha(C) = \alpha_0 + \delta(C), \quad \text{where } \delta(C) \propto \frac{1}{C_{max} - C} \tag{3}$$

予言： ブラックホール近傍や初期宇宙など、計算密度が極大となる領域では、物理定数が一定ではなく微小に揺らぐ現象が観測される。

2　Pythonによる実装 (Python Implementation)

本節では、デジタル宇宙論の核心ロジックを再現する、実行可能なPythonコードを提示する。

2.1　デジタル干渉と位相加算 (Digital Interference)

「波」を使わず、整数の位相カウンタの加算のみで干渉縞（確率分布）を生成するロジック。

```
import numpy as np
import matplotlib.pyplot as plt

def digital_interference_simulation(num_particles=20000, screen_width=400):
    #
    screen = np.zeros(screen_width)

    #           ()
    PHASE_MAX = 100
    #             ()
    WAVELENGTH_FACTOR = 15.0

    for _ in range(num_particles):
        #
        x = np.random.randint(0, screen_width)

        #   A,       B
        #
        dist_a = np.sqrt((x - screen_width * 0.45)**2 + 100**2)
        dist_b = np.sqrt((x - screen_width * 0.55)**2 + 100**2)

        #           ()
        #   /
        phase_a = int(dist_a / WAVELENGTH_FACTOR * PHASE_MAX) % PHASE_MAX
        phase_b = int(dist_b / WAVELENGTH_FACTOR * PHASE_MAX) % PHASE_MAX

        #         ()
        #  PHASE_MAX      /2
        phase_diff = np.abs(phase_a - phase_b)
        if phase_diff > PHASE_MAX / 2:
            phase_diff = PHASE_MAX - phase_diff

        #          (01)
        # prob = 1 -  ( /   ) ->   01,    0
        #         2
        interference_term = (1.0 - (phase_diff / (PHASE_MAX / 2))) ** 2
        #  :                 2
        #

        #          ()
        center_dist = np.abs(x - screen_width/2)
        diffraction_term = np.exp(- (center_dist / (screen_width * 0.25))**2)

        prob = interference_term * diffraction_term

        #        ()
        if np.random.rand() < prob:
            screen[x] += 1

    return screen

#
result = digital_interference_simulation()
plt.figure(figsize=(10, 6))
```

```
plt.plot(result, color='#3b82f6')
plt.title("Digital Interference Pattern (Discrete Phase Summation)")
plt.xlabel("Screen Position")
plt.ylabel("Particle Count")
plt.show()
```

Listing 1: digital_interference.py

2.2 量子消しゴム：データベース・フィルタリング (Quantum Eraser DB)

「未来が過去を変える」のではなく、事後的なクエリによってパターンが抽出されることを示すコード。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1.       (:  )
#             (Tag)
num_data = 5000
data = []

for i in range(num_data):
    #      (A :, B  :)
    hidden_type = np.random.choice(['TypeA', 'TypeB'])

    #
    if hidden_type == 'TypeA':
        while True:
            pos = np.random.randint(0, 100)
            #  (50)
            prob = np.cos((pos - 50) * 0.2) ** 2
            if np.random.rand() < prob:
                break
    else:
        while True:
            pos = np.random.randint(0, 100)
            # sin^2  cos^2        ()
            prob = np.sin((pos - 50) * 0.2) ** 2
            if np.random.rand() < prob:
                break

    #         (  Type)
    data.append({'ID': i, 'Position': pos, 'HiddenTag': hidden_type})

df = pd.DataFrame(data)

# 2.       ()
#            D1TypeAD2TypeB
def query_detector(detector_name):
    if detector_name == 'D1':
        # SELECT * FROM Screen WHERE HiddenTag = 'TypeA'
        return df[df['HiddenTag'] == 'TypeA']
    elif detector_name == 'D2':
        # SELECT * FROM Screen WHERE HiddenTag = 'TypeB'
        return df[df['HiddenTag'] == 'TypeB']
    else:
        #      (D3/ D4:        )
        return df

# 3.         (3)
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

```

```
#       ()
axes[0].hist(df['Position'], bins=40, color='gray', alpha=0.7)
axes[0].set_title("All Data (No Filter = Noise)")
axes[0].set_xlabel("Position")
axes[0].set_ylabel("Count")

# D1      ()
d1_results = query_detector('D1')
axes[1].hist(d1_results['Position'], bins=40, color='red', alpha=0.7)
axes[1].set_title("D1 Filtered (Interference Pattern)")
axes[1].set_xlabel("Position")

# D2      ()
d2_results = query_detector('D2')
axes[2].hist(d2_results['Position'], bins=40, color='blue', alpha=0.7)
axes[2].set_title("D2 Filtered (Inverted Pattern)")
axes[2].set_xlabel("Position")

plt.tight_layout()
plt.show()
```

Listing 2: quantum_eraser_db.py

2.3 重力赤方偏移と定数揺らぎ (Redshift & Fluctuation)

本理論の独自予言である「不感帯」と「定数の揺らぎ」を可視化するトイ・モデル。

```
import numpy as np
import matplotlib.pyplot as plt

#  2:
def simulate_discrete_redshift():
    #                   ()
    E_PLANCK = 50.0
    C = 1.0

    #       ΔΦ
    delta_phi = np.linspace(0, 50, 500)

    #
    E0 = 10.0

    #       ()
    #    = - (E0 / c^2) * ΔΦ
    #
    base_change = (E0 / (C**2)) * delta_phi * 10  #     10
    delta_E_continuous = - base_change

    #       ()
    #       E_PLANCK
    loss_discrete = np.floor(base_change / E_PLANCK) * E_PLANCK
    delta_E_discrete = -loss_discrete

    #
    plt.figure(figsize=(10, 6))
    plt.plot(delta_phi, delta_E_continuous, label='Standard Theory (Continuous)',
    linestyle='--', color='gray', alpha=0.5)
    plt.plot(delta_phi, delta_E_discrete, label='My Theory (Discrete)', color='red',
    linewidth=2)

    plt.title("Prediction: Dead Zone in Gravitational Redshift")
    plt.xlabel("Gravitational Potential Difference ΔΦ()")
    plt.ylabel("Energy Change Δ(E)")
```

```
    plt.grid(True, which='both', linestyle='--', alpha=0.7)
    plt.legend()
    plt.show()

#  3:
def simulate_constant_fluctuation():
    #
    RESOURCE_LIMIT = 1000

    #     C (0       )
    current_load = np.linspace(0, 995, 1000)

    #          ()
    ALPHA_0 = 1.0/137.0

    #
    alpha_observed = []

    for load in current_load:
        #
        margin = RESOURCE_LIMIT - load

        #
        #
        #       marginnoise_amplitude
        if margin > 0.1:
            noise_amplitude = 0.00005 * (50 / margin)
        else:
            noise_amplitude = 0.05 #

        fluctuation = np.random.normal(0, noise_amplitude)

        alpha_observed.append(ALPHA_0 + fluctuation)

    #
    plt.figure(figsize=(10, 6))
    plt.plot(current_load, [ALPHA_0]*len(current_load), label='Standard Model (Constant)'
    , linestyle='--', color='gray')
    plt.plot(current_load, alpha_observed, label='My Theory (Fluctuating)', color='purple
    ', linewidth=0.8)

    plt.title("Prediction: Fluctuation of Physical Constants near Limit")
    plt.xlabel("Computational Density (Load)")
    plt.ylabel("Value of Physical Constant  ()")

    #
    plt.axvline(x=RESOURCE_LIMIT, color='black', linestyle=':', label='System Limit')

    plt.grid(True, alpha=0.5)
    plt.legend()
    plt.show()

#
simulate_discrete_redshift()
simulate_constant_fluctuation()
```

Listing 3: gravitational_fluctuation.py

解説：このコードが示すもの

上記のシミュレーションコードは、本理論が既存の物理学とどこで分岐し、どのような観測可能な違い（予言）を生み出すかを視覚的に示している。

- **重力赤方偏移シミュレーション:**
  グラフには明確な**「階段（Steps）」**が現れる。特に重力ポテンシャル差 $\Delta\Phi$ が小さい初期段階では、エネルギー変化が最小単位（プランクエネルギー）に満たないため、赤方偏移がずっと「0」のまま進む**「不感帯（Dead Zone）」**が可視化される。これは、連続的な時空を前提とする一般相対性理論では説明できない、デジタル宇宙論特有の決定的な予言である。

- **定数揺らぎシミュレーション:**
  計算負荷（横軸）がシステム限界（Limit）に近づくにつれて、一定であるはずの物理定数が**激しく振動（ノイズ化）**し始める様子が描画される。これは、ビッグバン直後やブラックホール近傍といった「計算密度が極大化する領域」において、物理法則そのものが不安定になる（定数が定数でなくなる）ことを示唆している。