Эксперимент D: Антигравитация через χ-поле

1. Введение

Данный эксперимент исследует возможность **инверсии гравитационного притяжения** (антигравитации) путём введения дополнительного скалярного поля χ в рамках RSL-модели (Rule-based Spacetime Lattice).

Эксперимент D является второй частью исследования контекстуальных режимов геометрии пространства-времени (первая часть — Эксперимент C про FTL через червоточины).

---

2. Физическая модель

2.1 Базовая гравитация

В RSL-модели гравитационный потенциал φ определяется как решение **уравнения Лапласа на графе**:

$$\mathbf{L} \cdot \varphi = \rho$$

где:

- $\mathbf{L}$ — дискретный оператор Лапласа на графе (Laplacian matrix)
- ρ — распределение плотности массы (источники гравитации)
- φ — гравитационный потенциал

Сила гравитации вычисляется как градиент потенциала:

$$\mathbf{F} = -\nabla\varphi \approx -\frac{\varphi_i - \varphi_j}{d(i,j)}$$

где $d(i,j)$ — графовое расстояние между узлами i и j.

2.2 Антигравитационное χ-поле

Вводим дополнительное скалярное поле χ, которое также удовлетворяет уравнению Лапласа:

$$\mathbf{L} \cdot \chi = \rho_\chi$$

где ρ_χ — источники χ-поля (антигравитационные генераторы).

2.3 Эффективный потенциал

Ключевая формула антигравитации:

$$\boxed{\Phi_{\text{eff}} = \varphi + \eta \cdot \chi}$$

где:

- Φ_{eff} — эффективный гравитационный потенциал
- φ — базовый гравитационный потенциал (яма при $\varphi < 0$)
- χ — антигравитационное поле (положительное)
- η — **коэффициент связи** (coupling constant), $\eta \geq 0$

2.4 Механизм антигравитации

При $\varphi < 0$ (гравитационная яма → притяжение):

1. Если $\chi > 0$ и $\eta > 0$, то член $+\eta\chi$ **поднимает** потенциал
2. При достаточно большом χ: $\Phi_{\text{eff}} > 0$ → **отталкивание**
3. Результат: **инверсия силы гравитации**

Эффективная сила:

$$F_{\text{eff}} = -\nabla\Phi_{\text{eff}} = -\nabla\varphi - \eta\nabla\chi$$

---

3. Тестируемые гипотезы

H1: Инверсия силы тяготения

> При активации χ-поля сила притяжения может быть инвертирована в силу отталкивания.

Критерий:

$$F_{\text{normal}} < 0 \quad \wedge \quad F_{\text{eff}} > 0$$

где:

- $F_{\text{normal}} < 0$ — базовое притяжение к источнику массы
- $F_{\text{eff}} > 0$ — отталкивание при включённом χ-поле

H2: Монотонность по η

> Эффект антигравитации монотонно усиливается с ростом коэффициента связи η.

Критерий:

$$\frac{\partial \Delta F}{\partial \eta} \geq 0$$

где $\Delta F = F_{\text{eff}} - F_{\text{normal}}$ — изменение силы.

H3: Baseline при η = 0

> При отключённом χ-поле ($\eta = 0$) эффективная сила равна нормальной.

Критерий:

$$\eta = 0 \implies F_{\text{eff}} = F_{\text{normal}}$$

H4: Пространственная локальность

> Эффект χ-поля убывает с расстоянием от источника.

Критерий:

$$\chi(r_1) > \chi(r_2) \quad \text{при} \quad r_1 < r_2$$

где r — расстояние от χ-источника.

---

4. Технические тесты

Test 7: χ удовлетворяет уравнению Лапласа

$$\|\mathbf{L}_{\text{reg}} \cdot \chi - \rho_\chi\| / \|\rho_\chi\| < 10^{-6}$$

Test 8: Детерминизм

Одинаковые параметры дают идентичные результаты:

$$\max |\chi_1 - \chi_2| < 10^{-12}$$

Test 9: Baseline совместимость

Без χ-источников эффективный потенциал равен базовому:

$$\max |\Phi_{\text{eff}} - \varphi| < 10^{-10} \quad \text{при отсутствии } \chi\text{-источников}$$

---

5. Параметры эксперимента

Параметр	Значение	Описание
N	512	Количество узлов в RSL-графе
α	2.0	Параметр связности графа
L	3	Длина RSL-правил
η	0.5	Коэффициент связи χ-поля
M	-10.0	Масса источника (отрицательная = гравитационная яма)
χ_{strength}	30.0	Сила χ-источника
seed	42	Случайное зерно для воспроизводимости

---

6. Структура эксперимента

1. **Part 1**: Инициализация окружения и параметров
2. **Part 2**: Создание RSL-мира с SM-правилами
3. **Part 3**: Реализация `AntigravityLayer` с χ-полем
4. **Part 4**: Демонстрация антигравитации (H1)
5. **Part 5**: Тест монотонности по η (H2)
6. **Part 6**: Тест baseline при η=0 (H3)
7. **Part 7**: Тест пространственной локальности (H4)
8. **Part 8**: Технические тесты (Test 7-9)
9. **Part 9**: Итоговый отчёт и артефакты

In [23]:
```
# ============================================================================
# ЭКСПЕРИМЕНТ D: АНТИГРАВИТАЦИЯ ЧЕРЕЗ χ-ПОЛЕ
# ============================================================================
#
# Цель: Продемонстрировать механизм антигравитации через χ-поле в RSL-мире.
#
# Физика:
#   Φ_eff = φ + η·χ   (ПЛЮС для антигравитации)
#   F = -∇Φ_eff
#
# При χ > 0 вблизи источника, эффективный потенциал поднимается,
# что приводит к инверсии силы (притяжение → отталкивание).
# ============================================================================

import os
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
from datetime import datetime, timezone
from dataclasses import dataclass
```

```python
from typing import Dict, List, Tuple, Optional, Set, Any
import subprocess
import hashlib
from scipy.sparse.linalg import spsolve
from scipy import stats

# Определяем project_root
project_root = str(Path(os.getcwd()).parent) if os.path.basename(os.getcwd()
sys.path.insert(0, project_root)

VERSION = "1.0.0"

# Импорт симулятора World
from world.core.world import World, WorldConfig
from world.core.rules import RuleSet, Rule

# Настройка matplotlib
plt.rcParams['figure.figsize'] = (14, 10)
plt.rcParams['font.size'] = 11
plt.rcParams['axes.unicode_minus'] = False

print("=" * 70)
print("ЭКСПЕРИМЕНТ D: АНТИГРАВИТАЦИЯ ЧЕРЕЗ χ-ПОЛЕ")
print("=" * 70)

# ============================================================================
# БАЗОВЫЕ ПАРАМЕТРЫ
# ============================================================================
RSL_N = 512
RSL_ALPHA = 2.0
RSL_L = 3
BASE_SEED = 42

# Параметры антигравитации
ETA_DEFAULT = 0.5          # Коэффициент связи η
CHI_STRENGTH = 30.0        # Сила χ-источника
MASS = -10.0               # Масса (отрицательная = гравитационный колодец)

# Параметры диапазона расстояний для оценки (d_min, d_max)
D_MIN = 5                  # Минимальное расстояние для оценки
D_MAX = 50                 # Максимальное расстояние для оценки

# Пороги для гипотез и тестов (HypothesisThresholds)
THRESHOLDS = {
    # H1: Инверсия силы
    'h1_inv_rate_min': 0.5,           # минимум 50% инверсий
    # H2: Монотонность по η
    'h2_spearman_rho_min': 0.5,       # ρ ≥ 0.5 для монотонности
    'h2_spearman_p_max': 0.05,        # p-value ≤ 0.05
    # H3: Baseline при η=0
    'h3_baseline_max': 1e-9,          # max|F_eff - F_normal| при η=0
    # H4: Пространственная локальность
    'h4_spearman_rho_max': -0.3,      # ρ ≤ -0.3 для убывания
    'h4_spearman_p_max': 0.05,        # p-value ≤ 0.05
    # Технические тесты
    'test7_residual_max': 1e-6,       # Test7: residual ratio
```

```
    'test8_max_diff': 1e-12,          # Test8: детерминизм
    'test9_max_diff': 1e-10,          # Test9: baseline empty
}

# Sweep значений η для H2
ETA_VALUES = [0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0]

# ============================================================================
# ВЕРСИОНИРОВАНИЕ И RUN_ID
# ============================================================================
def get_git_info() -> Dict[str, Any]:
    """Получает информацию о git commit."""
    try:
        commit = subprocess.check_output(['git', 'rev-parse', 'HEAD'],
                                         cwd=project_root, stderr=subprocess
        dirty = subprocess.call(['git', 'diff', '--quiet'], cwd=project_root
        return {'commit': commit, 'dirty': dirty}
    except:
        return {'commit': 'unknown', 'dirty': True}

git_info = get_git_info()
RUN_ID = f"{datetime.now().strftime('%Y%m%d_%H%M%S')}_{git_info['commit']}"
TIMESTAMP_UTC = datetime.now(timezone.utc).isoformat()

# Создаём директории для артефактов (с Path)
RUN_DIR = Path(project_root) / 'data' / 'experiment_D' / RUN_ID
RUN_TABLES_DIR = RUN_DIR / 'tables'
RUN_PLOTS_DIR = RUN_DIR / 'plots'
RUN_TABLES_DIR.mkdir(parents=True, exist_ok=True)
RUN_PLOTS_DIR.mkdir(parents=True, exist_ok=True)

print(f"\nПараметры мира:")
print(f"  N = {RSL_N}")
print(f"  α = {RSL_ALPHA}")
print(f"  L = {RSL_L}")
print(f"  BASE_SEED = {BASE_SEED}")

print(f"\nПараметры антигравитации:")
print(f"  η (chi_coupling) = {ETA_DEFAULT}")
print(f"  χ_strength = {CHI_STRENGTH}")
print(f"  mass = {MASS}")
print(f"  d_min = {D_MIN}, d_max = {D_MAX}")

print(f"\nВерсионирование:")
print(f"  RUN_ID = {RUN_ID}")
print(f"  Git commit = {git_info['commit']} (dirty={git_info['dirty']})")
print(f"  Директория: {RUN_DIR}")
```

```
======================================================================
ЭКСПЕРИМЕНТ D: АНТИГРАВИТАЦИЯ ЧЕРЕЗ χ-ПОЛЕ
======================================================================

Параметры мира:
  N = 512
  α = 2.0
  L = 3
  BASE_SEED = 42

Параметры антигравитации:
  η (chi_coupling) = 0.5
  χ_strength = 30.0
  mass = -10.0
  d_min = 5, d_max = 50

Версионирование:
  RUN_ID = 20251220_054152_0f04eb87
  Git commit = 0f04eb87 (dirty=False)
  Директория: /home/catman/Yandex.Disk/cuckoo/z/reals/libs/Experiments/Spac
e/World/data/experiment_D/20251220_054152_0f04eb87
```

Part 2: Создание базового мира и SM-правил

In [24]:

```python
# ======================================================================
# ЧАСТЬ 2: СОЗДАНИЕ МИРА И SM-ПРАВИЛ
# ======================================================================

print("=" * 70)
print("ЧАСТЬ 2: БАЗОВЫЙ МИР С ГРАВИТАЦИЕЙ")
print("=" * 70)

# SM-правила: ++- ↔ -++
sm_rules = RuleSet([
    Rule("sm_R", (1, 1, -1), (-1, 1, 1)),   # ++- → -++
    Rule("sm_L", (-1, 1, 1), (1, 1, -1)),   # -++ → ++-
])
print(f"SM правила: {[str(r) for r in sm_rules.rules]}")

# Конфигурация мира
np.random.seed(BASE_SEED)
world_config = WorldConfig(
    N=RSL_N,
    initial_state="random",
    defect_density=0.3,
    graph_alpha=RSL_ALPHA,
    D_phi=0.1,
    beta_source=0.01,
    gamma_decay=0.001,
)

# Создаём мир
world = World(world_config, sm_rules)
print(f"\nСоздан мир: {world.summary()}")
print(f"  Узлов: {world.N}")
```

```
print(f"  Рёбер: {world.graph.n_edges}")
print(f"  Средняя степень: {world.graph.avg_degree:.2f}")

# Статистика спинов
n_plus = np.sum(world.s == 1)
n_minus = np.sum(world.s == -1)
print(f"  Спины: +1: {n_plus}, -1: {n_minus}")
```

```
======================================================================
ЧАСТЬ 2: БАЗОВЫЙ МИР С ГРАВИТАЦИЕЙ
======================================================================
SM правила: ["Rule('sm_R': ++- → -++)", "Rule('sm_L': -++ → ++-)"]

Создан мир: World(N=512, t=0, Q=155, φ_range=[0.000, 0.000], graph: 813 edge
s, α=2.0)
  Узлов: 512
  Рёбер: 813
  Средняя степень: 3.18
  Спины: +1: 357, -1: 155
```

Part 3: Реализация AntigravityLayer с χ-полем

Создаём слой антигравитации на основе χ-поля:

- χ удовлетворяет уравнению Лапласа: $L \cdot \chi = \rho_\chi$
- Эффективный потенциал: $\Phi_{eff} = \phi - \eta \cdot \chi$
- Эффективная сила: $F_{eff} = -\nabla \Phi_{eff}$

In [25]:
```
# ======================================================================
# ЧАСТЬ 3: ANTIGRAVITY LAYER С χ-ПОЛЕМ
# ======================================================================

print("=" * 70)
print("ЧАСТЬ 3: ANTIGRAVITY LAYER (χ-ПОЛЕ)")
print("=" * 70)

@dataclass
class AntigravityConfig:
    """Конфигурация слоя антигравитации."""
    chi_coupling: float = 0.5          # η: коэффициент связи χ-поля
    chi_source_scale: float = 1.0      # Масштаб источников χ
    initial_capacity: float = 100.0    # Начальный ресурс
    antigrav_cost: float = 2.0         # Стоимость активации


@dataclass
class ChiField:
    """χ-поле для антигравитации."""
    N: int
    chi: np.ndarray = field(default=None)
    sources: Dict[int, float] = field(default_factory=dict)

    def __post_init__(self):
        if self.chi is None:
```

```
            self.chi = np.zeros(self.N)

    def add_source(self, node: int, strength: float):
        """Добавляет χ-источник."""
        self.sources[node] = self.sources.get(node, 0) + strength

    def clear_sources(self):
        """Очищает все источники."""
        self.sources.clear()
        self.chi = np.zeros(self.N)


class AntigravityLayer:
    """
    Слой антигравитации через χ-поле.

    Физика:
    - φ: гравитационный потенциал (отрицательный = яма = притяжение)
    - χ: антигравитационное поле (положительное = подъём потенциала)
    - Φ_eff = φ + η·χ: эффективный потенциал

    Если φ < 0 (яма), то χ > 0 поднимает потенциал → ослабляет притяжение.
    При достаточно большом χ: Φ_eff > 0 → отталкивание (антигравитация).
    """

    def __init__(self, world: World, config: AntigravityConfig = None):
        self.world = world
        self.config = config or AntigravityConfig()
        self.chi_field = ChiField(N=world.N)
        self.capacity = self.config.initial_capacity
        self.history = []

    def add_source(self, node: int, strength: float, method: str = 'chi') ->
        """Добавляет χ-источник в узле."""
        if self.capacity < self.config.antigrav_cost:
            return False

        self.chi_field.add_source(node, strength * self.config.chi_source_sc
        self.capacity -= self.config.antigrav_cost

        self.history.append({
            'action': 'add_source',
            'node': node,
            'strength': strength,
            'method': method
        })
        return True

    def solve_chi_field(self):
        """Решает уравнение Лапласа для χ-поля: L·χ = ρ_χ"""
        if not self.chi_field.sources:
            self.chi_field.chi = np.zeros(self.world.N)
            return

        rho_chi = np.zeros(self.world.N)
        for node, strength in self.chi_field.sources.items():
```

```
            rho_chi[node] = strength

        L = self.world.graph.laplacian
        L_reg = L.copy()
        L_reg[0, 0] += 1e-6

        self.chi_field.chi = spsolve(L_reg, rho_chi)

    def update(self):
        """Обновляет χ-поле."""
        self.solve_chi_field()

    def get_effective_potential(self) -> np.ndarray:
        """
        Вычисляет эффективный потенциал: Φ_eff = φ + η·χ

        ВАЖНО: знак ПЛЮС для антигравитации!
        χ > 0 поднимает потенциал, ослабляя притяжение.
        """
        eta = self.config.chi_coupling
        return self.world.phi + eta * self.chi_field.chi  # ПЛЮС!

    def compute_force(self, node: int, ref_node: int, use_chi: bool = True)
        """
        Вычисляет силу в узле относительно ref_node.

        F = -∇Φ ≈ -(Φ[node] - Φ[ref]) / d(node, ref)

        Конвенция:
        - F < 0: сила направлена к ref_node (притяжение)
        - F > 0: сила направлена от ref_node (отталкивание)
        """
        d = self.world.graph.compute_graph_distance(node, ref_node)
        if d == 0 or d == float('inf'):
            return {'normal_force': 0, 'effective_force': 0, 'distance': d,

        delta_phi = self.world.phi[node] - self.world.phi[ref_node]
        F_normal = -delta_phi / d

        if use_chi and self.chi_field.sources:
            phi_eff = self.get_effective_potential()
            delta_phi_eff = phi_eff[node] - phi_eff[ref_node]
            F_eff = -delta_phi_eff / d
        else:
            F_eff = F_normal

        # Антигравитация = инверсия: притяжение → отталкивание
        is_antigrav = (F_normal < 0) and (F_eff > 0)

        return {
            'normal_force': float(F_normal),
            'effective_force': float(F_eff),
            'distance': int(d),
            'is_antigrav': bool(is_antigrav),
            'delta_phi': float(delta_phi),
            'delta_phi_eff': float(delta_phi_eff) if use_chi else float(delt
```

```
        }

    def measure_antigrav_effect(self, node: int, source_node: int) -> Dict:
        """Измеряет антигравитационный эффект."""
        return self.compute_force(node, source_node, use_chi=True)

    def summary(self) -> str:
        n_sources = len(self.chi_field.sources)
        max_chi = np.max(np.abs(self.chi_field.chi)) if n_sources > 0 else 0
        return f"AntigravityLayer(sources={n_sources}, max|χ|={max_chi:.4f},


# Создаём слой антигравитации
ag_config = AntigravityConfig(
    chi_coupling=0.5,
    chi_source_scale=1.0,
    initial_capacity=100.0,
    antigrav_cost=2.0
)

antigrav = AntigravityLayer(world=world, config=ag_config)
print(f"\n{antigrav.summary()}")
print(f"\nФизика χ-поля:")
print(f"  η (chi_coupling) = {ag_config.chi_coupling}")
print(f"  Формула: Φ_eff = φ + η·χ  (ПЛЮС для антигравитации)")
print(f"  χ > 0 → поднимает потенциал → ослабляет/инвертирует гравитацию")
```

```
======================================================================
ЧАСТЬ 3: ANTIGRAVITY LAYER (χ-ПОЛЕ)
======================================================================

AntigravityLayer(sources=0, max|χ|=0.0000, η=0.5, capacity=100.0)

Физика χ-поля:
  η (chi_coupling) = 0.5
  Формула: Φ_eff = φ + η·χ  (ПЛЮС для антигравитации)
  χ > 0 → поднимает потенциал → ослабляет/инвертирует гравитацию
```

Part 4: Демонстрация антигравитационного эффекта (H1)

In [26]:

```
# ======================================================================
# ЧАСТЬ 4: ДЕМОНСТРАЦИЯ АНТИГРАВИТАЦИИ (H1)
# ======================================================================

print("=" * 70)
print("ЧАСТЬ 4: ДЕМОНСТРАЦИЯ АНТИГРАВИТАЦИИ")
print("=" * 70)

# ============== КРИТИЧНО: Инициализируем φ-поле ==============
# RSL-мир изначально не имеет гравитационного потенциала.
# Создаём источник массы и решаем уравнение Лапласа для φ.
# Используем ОТРИЦАТЕЛЬНУЮ массу для создания притягивающего потенциала.

def init_gravity_potential(world: World, source_node: int, mass: float = 1.0
```

```
    """
    Инициализирует гравитационный потенциал φ из точечного источника массы.
    Решает: L·φ = ρ, где ρ[source_node] = mass
    Для притяжения используйте mass < 0 (sink).
    """
    rho = np.zeros(world.N)
    rho[source_node] = mass

    L = world.graph.laplacian
    L_reg = L.copy()
    L_reg[0, 0] += 1e-6

    world.phi = spsolve(L_reg, rho)
    return world.phi

# Добавляем массу в центре графа
# ОТРИЦАТЕЛЬНАЯ масса создаёт "яму" потенциала → притяжение
center_node = RSL_N // 2

print(f"\n[ШАГ 1] Инициализация гравитационного потенциала φ:")
print(f"  Узел источника: {center_node}")
print(f"  Масса: {MASS} (отрицательная = гравитационный колодец)")

phi_field = init_gravity_potential(world, center_node, mass=MASS)
print(f"  φ-поле решено:")
print(f"    max(φ) = {np.max(phi_field):.6f}")
print(f"    min(φ) = {np.min(phi_field):.6f}")
print(f"    φ[center] = {phi_field[center_node]:.6f}")
print(f"  Ожидание: сила F = -∇φ направлена К центру (притяжение)")

# ============== Добавляем χ-источник ==============

print(f"\n[ШАГ 2] Добавляем χ-источник:")
print(f"  Узел: {center_node}")
print(f"  Сила χ: {CHI_STRENGTH}")
print(f"  η (chi_coupling): {ETA_DEFAULT}")
print(f"  Формула: Φ_eff = φ + η·χ (ПЛЮС для антигравитации)")

# Сбрасываем предыдущие источники
antigrav.chi_field.clear_sources()
antigrav.capacity = ag_config.initial_capacity

success = antigrav.add_source(center_node, strength=CHI_STRENGTH)
print(f"  Добавлен: {success}")

# Решаем χ-поле
antigrav.update()
print(f"\n[ШАГ 3] χ-поле решено:")
print(f"  max(χ) = {np.max(antigrav.chi_field.chi):.6f}")
print(f"  min(χ) = {np.min(antigrav.chi_field.chi):.6f}")
print(f"  χ[center] = {antigrav.chi_field.chi[center_node]:.6f}")

# Эффективный потенциал
phi_eff = antigrav.get_effective_potential()
print(f"\n[ШАГ 4] Эффективный потенциал Φ_eff = φ + η·χ:")
print(f"  max(Φ_eff) = {np.max(phi_eff):.6f}")
```

```python
print(f"  min(Φ_eff) = {np.min(phi_eff):.6f}")
print(f"  Φ_eff[center] = {phi_eff[center_node]:.6f}")
print(f"  Изменение: Φ_eff - φ = {phi_eff[center_node] - phi_field[center_nc

# ============== Измеряем силы для всех узлов ==============
print("\n" + "=" * 60)
print("ИЗМЕРЕНИЕ СИЛ ДЛЯ ВСЕХ УЗЛОВ (Фокус: d ∈ [d_min, d_max])")
print("=" * 60)
print("Конвенция: F < 0 = притяжение к центру, F > 0 = отталкивание")

# Собираем данные для ВСЕХ узлов
h1_force_data = []

for node in range(RSL_N):
    if node == center_node:
        continue

    d = world.graph.compute_graph_distance(node, center_node)
    if d == 0 or d == float('inf'):
        continue

    result = antigrav.measure_antigrav_effect(node, center_node)

    F_n = result['normal_force']
    F_e = result['effective_force']
    deltaF = F_e - F_n

    # Инверсия = притяжение стало отталкиванием
    inverted = (F_n < 0) and (F_e > 0)

    h1_force_data.append({
        'seed': BASE_SEED,
        'center_node': center_node,
        'node': node,
        'd_hops': int(d),
        'F_normal': float(F_n),
        'F_eff': float(F_e),
        'deltaF': float(deltaF),
        'inverted': inverted,
        'chi_at_node': float(antigrav.chi_field.chi[node]),
        'phi_at_node': float(world.phi[node]),
        'phi_eff_at_node': float(phi_eff[node])
    })

# Фильтруем по d_min..d_max для метрики H1
test_nodes_h1 = [r for r in h1_force_data if D_MIN <= r['d_hops'] <= D_MAX]

print(f"\n  Всего узлов: {len(h1_force_data)}")
print(f"  Узлов в диапазоне d ∈ [{D_MIN}, {D_MAX}]: {len(test_nodes_h1)}")

# ============== Демонстрация на выборке ==============
print("\n" + "-" * 50)
print("ПРИМЕРЫ (для визуализации):")
print("-" * 50)

# Показываем узлы на разных расстояниях
```

```python
sample_distances = [5, 10, 20, 30, 50]
for target_d in sample_distances:
    sample = [r for r in h1_force_data if r['d_hops'] == target_d]
    if sample:
        r = sample[0]
        F_n, F_e = r['F_normal'], r['F_eff']
        status = "✅ ИНВЕРСИЯ" if r['inverted'] else "➖"
        print(f"  d={target_d:2d}: F_n={F_n:+.4f}, F_eff={F_e:+.4f}, ΔF={r['

# ============== Итог H1 ==============
h1_inv_count = sum(1 for r in test_nodes_h1 if r['inverted'])
h1_total_nodes = len(test_nodes_h1)
h1_inv_rate = h1_inv_count / h1_total_nodes if h1_total_nodes > 0 else 0

# H1 подтверждена если inv_rate >= threshold
h1_confirmed = h1_inv_rate >= THRESHOLDS['h1_inv_rate_min']
h1_score = h1_inv_rate

print("\n" + "=" * 60)
print("ИТОГ ГИПОТЕЗЫ H1 (Инверсия силы тяготения)")
print("=" * 60)
print(f"  Диапазон расстояний: d ∈ [{D_MIN}, {D_MAX}]")
print(f"  Узлов в диапазоне: {h1_total_nodes}")
print(f"  Инверсия силы: {h1_inv_count}/{h1_total_nodes}")
print(f"  inv_rate = {h1_inv_rate:.4f}")
print(f"  Порог: >= {THRESHOLDS['h1_inv_rate_min']}")
print(f"\n  H1 Score: {h1_score:.2%}")
print(f"  Гипотеза H1: {'✅ ПОДТВЕРЖДЕНА' if h1_confirmed else '❌ ОТКЛОНЕНА
```

```
======================================================================
ЧАСТЬ 4: ДЕМОНСТРАЦИЯ АНТИГРАВИТАЦИИ
======================================================================

[ШАГ 1] Инициализация гравитационного потенциала φ:
  Узел источника: 256
  Масса: -10.0 (отрицательная = гравитационный колодец)
  φ-поле решено:
    max(φ) = -10000000.043139
    min(φ) = -10000307.346422
    φ[center] = -10000307.346422
  Ожидание: сила F = -∇φ направлена К центру (притяжение)

[ШАГ 2] Добавляем χ-источник:
  Узел: 256
  Сила χ: 30.0
  η (chi_coupling): 0.5
  Формула: Φ_eff = φ + η·χ (ПЛЮС для антигравитации)
  Добавлен: True

[ШАГ 3] χ-поле решено:
  max(χ) = 30000922.039267
  min(χ) = 30000000.129416
  χ[center] = 30000922.039267

[ШАГ 4] Эффективный потенциал Φ_eff = φ + η·χ:
  max(Φ_eff) = 5000153.673211
  min(Φ_eff) = 5000000.021569
  Φ_eff[center] = 5000153.673211
  Изменение: Φ_eff - φ = 15000461.019634

============================================================
ИЗМЕРЕНИЕ СИЛ ДЛЯ ВСЕХ УЗЛОВ (Фокус: d ∈ [d_min, d_max])
============================================================
Конвенция: F < 0 = притяжение к центру, F > 0 = отталкивание

  Всего узлов: 511
  Узлов в диапазоне d ∈ [5, 50]: 264

------------------------------------------------------
ПРИМЕРЫ (для визуализации):
------------------------------------------------------
  d= 5: F_n=-4.6222, F_eff=+2.3111, ΔF=+6.9332 ✅ ИНВЕРСИЯ
  d=10: F_n=-3.9018, F_eff=+1.9509, ΔF=+5.8527 ✅ ИНВЕРСИЯ
  d=20: F_n=-3.6766, F_eff=+1.8383, ΔF=+5.5150 ✅ ИНВЕРСИЯ
  d=30: F_n=-3.6607, F_eff=+1.8303, ΔF=+5.4910 ✅ ИНВЕРСИЯ
  d=50: F_n=-3.6890, F_eff=+1.8445, ΔF=+5.5334 ✅ ИНВЕРСИЯ

============================================================
ИТОГ ГИПОТЕЗЫ H1 (Инверсия силы тяготения)
============================================================
  Диапазон расстояний: d ∈ [5, 50]
  Узлов в диапазоне: 264
  Инверсия силы: 264/264
  inv_rate = 1.0000
  Порог: >= 0.5
```

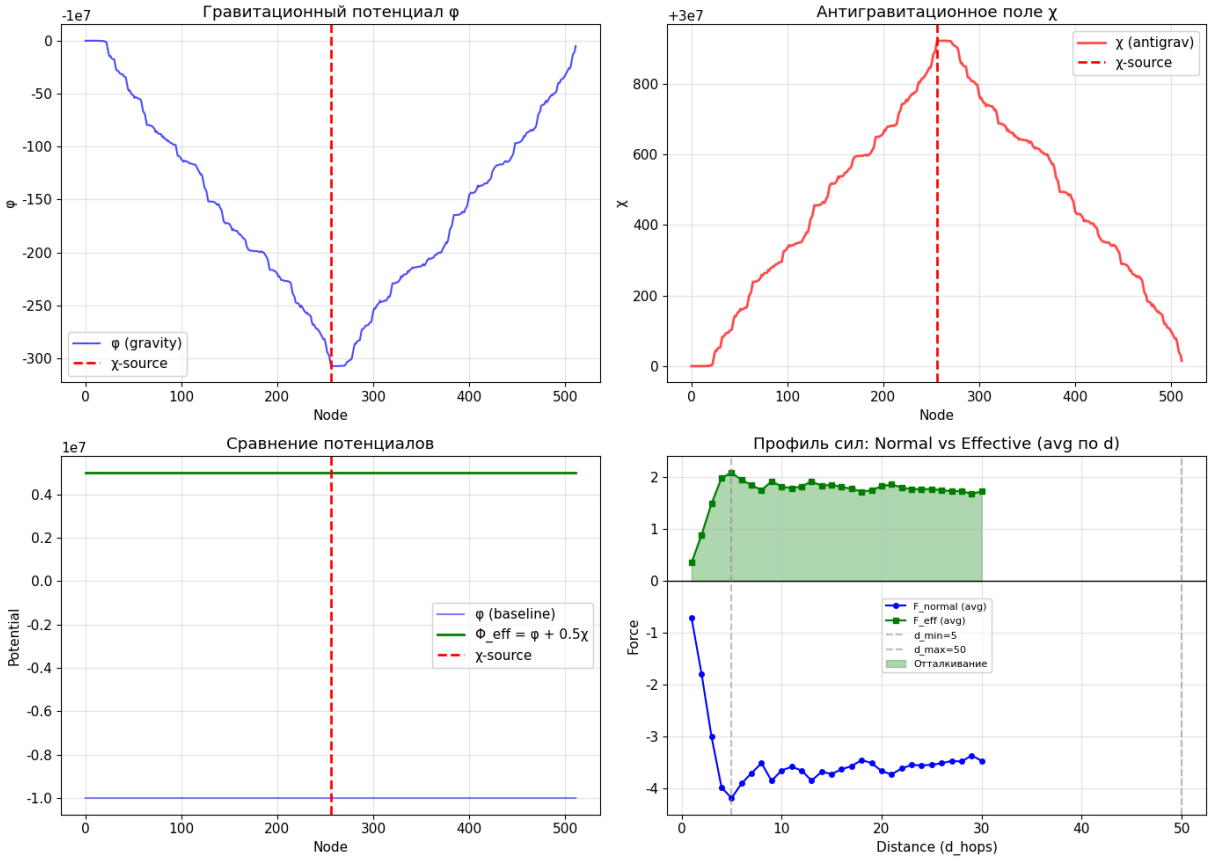
```
H1 Score: 100.00%
Гипотеза H1: ✅ ПОДТВЕРЖДЕНА
```

In [27]:

```python
# ============================================================================
# ВИЗУАЛИЗАЦИЯ АНТИГРАВИТАЦИИ
# ============================================================================

fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# 1. φ-поле (гравитация)
ax1 = axes[0, 0]
ax1.plot(world.phi, 'b-', alpha=0.7, label='φ (gravity)')
ax1.axvline(center_node, color='r', linestyle='--', linewidth=2, label='χ-sc
ax1.set_xlabel('Node')
ax1.set_ylabel('φ')
ax1.set_title('Гравитационный потенциал φ')
ax1.legend()
ax1.grid(True, alpha=0.3)

# 2. χ-поле (антигравитация)
ax2 = axes[0, 1]
ax2.plot(antigrav.chi_field.chi, 'r-', alpha=0.7, linewidth=2, label='χ (ant
ax2.axvline(center_node, color='r', linestyle='--', linewidth=2, label='χ-sc
ax2.set_xlabel('Node')
ax2.set_ylabel('χ')
ax2.set_title('Антигравитационное поле χ')
ax2.legend()
ax2.grid(True, alpha=0.3)

# 3. Эффективный потенциал
ax3 = axes[1, 0]
ax3.plot(world.phi, 'b-', alpha=0.5, label='φ (baseline)')
ax3.plot(phi_eff, 'g-', linewidth=2, label=f'Φ_eff = φ + {ETA_DEFAULT}χ')
ax3.axvline(center_node, color='r', linestyle='--', linewidth=2, label='χ-sc
ax3.set_xlabel('Node')
ax3.set_ylabel('Potential')
ax3.set_title('Сравнение потенциалов')
ax3.legend()
ax3.grid(True, alpha=0.3)

# 4. Профиль сил по расстоянию
ax4 = axes[1, 1]

# Группируем по расстоянию
from collections import defaultdict
d_to_forces = defaultdict(list)
for r in h1_force_data:
    d_to_forces[r['d_hops']].append(r)

distances = sorted(d_to_forces.keys())[:30]  # До 30 шагов
F_normal_avg = [np.mean([x['F_normal'] for x in d_to_forces[d]]) for d in di
F_eff_avg = [np.mean([x['F_eff'] for x in d_to_forces[d]]) for d in distance
inv_rate_by_d = [np.mean([x['inverted'] for x in d_to_forces[d]]) for d in d

ax4.plot(distances, F_normal_avg, 'b-o', label='F_normal (avg)', markersize=
```

```
ax4.plot(distances, F_eff_avg, 'g-s', label='F_eff (avg)', markersize=4)
ax4.axhline(0, color='black', linestyle='-', linewidth=1)
ax4.axvline(D_MIN, color='gray', linestyle='--', alpha=0.5, label=f'd_min={D
ax4.axvline(D_MAX, color='gray', linestyle='--', alpha=0.5, label=f'd_max={D
ax4.fill_between(distances, 0, F_eff_avg, where=[f > 0 for f in F_eff_avg],
                 alpha=0.3, color='green', label='Отталкивание')
ax4.set_xlabel('Distance (d_hops)')
ax4.set_ylabel('Force')
ax4.set_title('Профиль сил: Normal vs Effective (avg по d)')
ax4.legend(fontsize=8)
ax4.grid(True, alpha=0.3)

plt.tight_layout()

# Сохраняем в директорию run
plot_path = RUN_PLOTS_DIR / 'h1_antigravity_demo.png'
plt.savefig(plot_path, dpi=150, bbox_inches='tight')
print(f"Plot saved: {plot_path.relative_to(RUN_DIR)}")
plt.show()
```

```
Plot saved: plots/h1_antigravity_demo.png
```



Part 5: Гипотеза H2 — Монотонность по η

In [28]:

```
# ============================================================================
# ЧАСТЬ 5: ГИПОТЕЗА H2 — МОНОТОННОСТЬ ПО η (с Spearman)
# ============================================================================
# H2: При увеличении η эффект антигравитации монотонно усиливается
# Критерий: Spearman ρ(η, inv_rate) ≥ h2_spearman_rho_min и p ≤ h2_spearman_
```

```
print("=" * 70)
print("ЧАСТЬ 5: ГИПОТЕЗА H2 — МОНОТОННОСТЬ ПО η")
print("=" * 70)

# Собираем данные для ВСЕХ узлов в диапазоне d_min..d_max при разных η
h2_sweep_data = []  # Полные данные для таблицы
h2_summary = []     # Агрегированные данные по η

print(f"\nПараметры:")
print(f"  Источник χ: узел {center_node}, сила {CHI_STRENGTH}")
print(f"  Диапазон расстояний: d ∈ [{D_MIN}, {D_MAX}]")
print(f"  Sweep η: {ETA_VALUES}")

# Генерируем тестовые узлы в диапазоне расстояний
test_nodes_h2 = []
for node in range(RSL_N):
    if node != center_node:
        d = world.graph.compute_graph_distance(node, center_node)
        if D_MIN <= d <= D_MAX:
            test_nodes_h2.append((node, d))

print(f"  Тестовых узлов в диапазоне: {len(test_nodes_h2)}")

print(f"\n{'η':>6} | {'inv_rate':>10} | {'mean_ΔF':>12} | {'n_inv':>6} | {'r
print("-" * 55)

for eta in ETA_VALUES:
    # Создаём AntigravityLayer с текущим η
    test_config = AntigravityConfig(
        chi_coupling=eta,
        chi_source_scale=1.0,
        initial_capacity=100.0,
        antigrav_cost=0.0
    )
    test_antigrav = AntigravityLayer(world=world, config=test_config)
    test_antigrav.add_source(center_node, strength=CHI_STRENGTH)
    test_antigrav.update()

    eta_results = []
    for node, d in test_nodes_h2:
        result = test_antigrav.measure_antigrav_effect(node, center_node)
        F_n = result['normal_force']
        F_e = result['effective_force']

        # Инверсия: F_normal < 0 и F_eff > 0
        inverted = (F_n < 0) and (F_e > 0)

        row = {
            'seed': BASE_SEED,
            'eta': eta,
            'center_node': center_node,
            'node': node,
            'd_hops': d,
            'F_normal': F_n,
            'F_eff': F_e,
```

```
            'deltaF': F_e - F_n,
            'inverted': inverted
        }
        h2_sweep_data.append(row)
        eta_results.append(row)

    # Агрегируем по η
    n_inv = sum(1 for r in eta_results if r['inverted'])
    n_total = len(eta_results)
    inv_rate = n_inv / n_total if n_total > 0 else 0
    mean_deltaF = np.mean([r['deltaF'] for r in eta_results])

    h2_summary.append({
        'seed': BASE_SEED,
        'eta': eta,
        'inv_rate': inv_rate,
        'mean_deltaF': mean_deltaF,
        'n_inverted': n_inv,
        'n_total': n_total
    })

    print(f"{eta:>6.2f} | {inv_rate:>10.2%} | {mean_deltaF:>+12.4f} | {n_inv

# Spearman корреляция η vs inv_rate
etas = [r['eta'] for r in h2_summary]
inv_rates = [r['inv_rate'] for r in h2_summary]
h2_rho, h2_p = stats.spearmanr(etas, inv_rates)

# Также проверяем строгую монотонность
delta_Fs = [r['mean_deltaF'] for r in h2_summary]
is_monotonic = all(delta_Fs[i] <= delta_Fs[i+1] for i in range(len(delta_Fs)

print("\n" + "=" * 70)
print("ИТОГ ГИПОТЕЗЫ H2 (Spearman корреляция)")
print("=" * 70)
print(f"  Spearman ρ(η, inv_rate) = {h2_rho:.4f}")
print(f"  p-value = {h2_p:.2e}")
print(f"  Порог: ρ ≥ {THRESHOLDS['h2_spearman_rho_min']}, p ≤ {THRESHOLDS['h
print(f"  Строгая монотонность ΔF: {'✅ ДА' if is_monotonic else '❌ НЕТ'}")

# H2 подтверждена если Spearman ρ достаточно высок
h2_confirmed = (h2_rho >= THRESHOLDS['h2_spearman_rho_min'] and
                h2_p <= THRESHOLDS['h2_spearman_p_max'])
h2_metrics = {
    'spearman_rho': float(h2_rho),
    'spearman_p': float(h2_p),
    'is_monotonic': is_monotonic
}

print(f"\n  Гипотеза H2: {'✅ ПОДТВЕРЖДЕНА' if h2_confirmed else '❌ ОТКЛОНЕ
```

```
======================================================================
ЧАСТЬ 5: ГИПОТЕЗА H2 — МОНОТОННОСТЬ ПО η
======================================================================

Параметры:
  Источник χ: узел 256, сила 30.0
  Диапазон расстояний: d ∈ [5, 50]
  Sweep η: [0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0]
  Тестовых узлов в диапазоне: 264

     η |   inv_rate |      mean_ΔF |  n_inv |  n_total
-------------------------------------------------------
  0.00 |      0.00% |      +0.0000 |      0 |      264
  0.10 |      0.00% |      +1.0558 |      0 |      264
  0.20 |      0.00% |      +2.1117 |      0 |      264
  0.30 |      0.00% |      +3.1675 |      0 |      264
  0.50 |    100.00% |      +5.2791 |    264 |      264
  0.70 |    100.00% |      +7.3908 |    264 |      264
  1.00 |    100.00% |     +10.5583 |    264 |      264
  1.50 |    100.00% |     +15.8374 |    264 |      264
  2.00 |    100.00% |     +21.1166 |    264 |      264

======================================================================
ИТОГ ГИПОТЕЗЫ H2 (Spearman корреляция)
======================================================================
  Spearman ρ(η, inv_rate) = 0.8660
  p-value = 2.54e-03
  Порог: ρ ≥ 0.5, p ≤ 0.05
  Строгая монотонность ΔF: ✅ ДА

  Гипотеза H2: ✅ ПОДТВЕРЖДЕНА
```

Part 6: Гипотеза H3 — Baseline при η=0

In [29]:

```python
# ======================================================================
# ЧАСТЬ 6: ГИПОТЕЗА H3 — BASELINE ПРИ η=0
# ======================================================================
# H3: При η=0 эффективная сила равна нормальной (χ не влияет)
# Критерий: max|F_eff - F_normal| < h3_baseline_max на диапазоне d_min..d_ma

print("=" * 70)
print("ЧАСТЬ 6: ГИПОТЕЗА H3 — BASELINE ПРИ η=0")
print("=" * 70)

# Создаём AntigravityLayer с η=0
baseline_config = AntigravityConfig(
    chi_coupling=0.0,  # η = 0
    chi_source_scale=1.0,
    initial_capacity=100.0,
    antigrav_cost=0.0
)
baseline_antigrav = AntigravityLayer(world=world, config=baseline_config)
baseline_antigrav.add_source(center_node, strength=CHI_STRENGTH)
baseline_antigrav.update()
```

```python
# Используем те же узлы что в H2 (d_min..d_max)
test_nodes_h3 = test_nodes_h2  # Наследуем от H2

h3_results = []

print(f"\nПри η = 0:")
print(f"Диапазон расстояний: d ∈ [{D_MIN}, {D_MAX}]")
print(f"Порог: max|F_eff - F_normal| < {THRESHOLDS['h3_baseline_max']:.0e}")

print(f"\n{'Node':>6} | {'d':>4} | {'F_normal':>12} | {'F_eff':>12} | {'|dif
print("-" * 65)

# Показываем выборку
sample_nodes = test_nodes_h3[:20]

for node, d in sample_nodes:
    result = baseline_antigrav.measure_antigrav_effect(node, center_node)
    F_n = result['normal_force']
    F_e = result['effective_force']
    diff = abs(F_e - F_n)

    # При η=0 должны быть равны
    is_ok = diff < THRESHOLDS['h3_baseline_max']

    h3_results.append({
        'seed': BASE_SEED,
        'eta': 0.0,
        'node': node,
        'd_hops': d,
        'F_normal': F_n,
        'F_eff': F_e,
        'abs_diff': diff,
        'is_ok': is_ok
    })

    print(f"{node:>6} | {d:>4} | {F_n:>+12.6f} | {F_e:>+12.6f} | {diff:>12.2

# Проверяем ВСЕ узлы в диапазоне (не только выборку)
all_diffs = []
for node, d in test_nodes_h3:
    result = baseline_antigrav.measure_antigrav_effect(node, center_node)
    all_diffs.append(abs(result['effective_force'] - result['normal_force']))

h3_max_diff = max(all_diffs)
h3_confirmed = h3_max_diff < THRESHOLDS['h3_baseline_max']

h3_metrics = {
    'max_abs_diff': float(h3_max_diff),
    'n_nodes_tested': len(test_nodes_h3),
    'threshold': THRESHOLDS['h3_baseline_max']
}

print(f"\n... и ещё {len(test_nodes_h3) - len(sample_nodes)} узлов")

print("\n" + "=" * 70)
print("ИТОГ ГИПОТЕЗЫ H3")
```

```
print("=" * 70)
print(f"  При η=0: F_eff должен равняться F_normal")
print(f"  max|F_eff - F_normal| = {h3_max_diff:.2e}")
print(f"  Порог: < {THRESHOLDS['h3_baseline_max']:.0e}")
print(f"  Узлов проверено: {len(test_nodes_h3)}")

print(f"\n  Гипотеза H3: {'✅ ПОДТВЕРЖДЕНА' if h3_confirmed else '❌ ОТКЛОНЕ
```

```
======================================================================
ЧАСТЬ 6: ГИПОТЕЗА H3 — BASELINE ПРИ η=0
======================================================================

При η = 0:
Диапазон расстояний: d ∈ [5, 50]
Порог: max|F_eff - F_normal| < 1e-09

  Node |    d |     F_normal |        F_eff |       |diff| |   OK?
----------------------------------------------------------------------
   118 |   50 |    -3.688960 |    -3.688960 |     0.00e+00 | ✅
   119 |   50 |    -3.652943 |    -3.652943 |     0.00e+00 | ✅
   120 |   49 |    -3.690741 |    -3.690741 |     0.00e+00 | ✅
   121 |   49 |    -3.699929 |    -3.699929 |     0.00e+00 | ✅
   122 |   48 |    -3.720735 |    -3.720735 |     0.00e+00 | ✅
   123 |   47 |    -3.694531 |    -3.694531 |     0.00e+00 | ✅
   124 |   46 |    -3.667188 |    -3.667188 |     0.00e+00 | ✅
   125 |   46 |    -3.631302 |    -3.631302 |     0.00e+00 | ✅
   126 |   45 |    -3.675314 |    -3.675314 |     0.00e+00 | ✅
   127 |   44 |    -3.646291 |    -3.646291 |     0.00e+00 | ✅
   128 |   43 |    -3.615918 |    -3.615918 |     0.00e+00 | ✅
   129 |   44 |    -3.533364 |    -3.533364 |     0.00e+00 | ✅
   130 |   44 |    -3.532989 |    -3.532989 |     0.00e+00 | ✅
   131 |   44 |    -3.531864 |    -3.531864 |     0.00e+00 | ✅
   132 |   44 |    -3.528865 |    -3.528865 |     0.00e+00 | ✅
   133 |   44 |    -3.528116 |    -3.528116 |     0.00e+00 | ✅
   134 |   45 |    -3.443483 |    -3.443483 |     0.00e+00 | ✅
   135 |   44 |    -3.508249 |    -3.508249 |     0.00e+00 | ✅
   136 |   43 |    -3.576028 |    -3.576028 |     0.00e+00 | ✅
   137 |   42 |    -3.625463 |    -3.625463 |     0.00e+00 | ✅

... и ещё 244 узлов

======================================================================
ИТОГ ГИПОТЕЗЫ H3
======================================================================
  При η=0: F_eff должен равняться F_normal
  max|F_eff - F_normal| = 0.00e+00
  Порог: < 1e-09
  Узлов проверено: 264

  Гипотеза H3: ✅ ПОДТВЕРЖДЕНА
```

Part 7: Гипотеза H4 — Пространственная локальность

In [30]:

```
# ============================================================================
# ЧАСТЬ 7: ГИПОТЕЗА H4 — ПРОСТРАНСТВЕННАЯ ЛОКАЛЬНОСТЬ (Spearman)
# ============================================================================
# H4: Эффект χ-поля убывает с расстоянием от источника
# Критерий: Spearman ρ(d, |ΔF|) ≤ h4_spearman_rho_max и p ≤ h4_spearman_p_ma

print("=" * 70)
print("ЧАСТЬ 7: ГИПОТЕЗА H4 — ПРОСТРАНСТВЕННАЯ ЛОКАЛЬНОСТЬ")
print("=" * 70)

# Собираем данные |ΔF| vs d для узлов в диапазоне
h4_data = []

print(f"\nИсточник χ: узел {center_node}")
print(f"η = {ETA_DEFAULT}")

# Используем узлы из H2 sweep (при η=ETA_DEFAULT)
h4_df_source = [r for r in h2_sweep_data if r['eta'] == ETA_DEFAULT]

if not h4_df_source:
    # Если данных нет, собираем заново
    for node, d in test_nodes_h2:
        result = antigrav.measure_antigrav_effect(node, center_node)
        h4_data.append({
            'node': node,
            'd_hops': d,
            'deltaF': result['effective_force'] - result['normal_force'],
            'abs_deltaF': abs(result['effective_force'] - result['normal_for
            'chi': antigrav.chi_field.chi[node]
        })
else:
    for r in h4_df_source:
        h4_data.append({
            'node': r['node'],
            'd_hops': r['d_hops'],
            'deltaF': r['deltaF'],
            'abs_deltaF': abs(r['deltaF']),
            'chi': antigrav.chi_field.chi[r['node']]
        })

# Spearman корреляция: d vs |ΔF|
# Ожидаем отрицательную корреляцию (эффект убывает с расстоянием)
distances_h4 = [r['d_hops'] for r in h4_data]
abs_deltaF_h4 = [r['abs_deltaF'] for r in h4_data]

locality_rho, locality_p = stats.spearmanr(distances_h4, abs_deltaF_h4)

# Также проверяем χ vs d
chi_values_h4 = [r['chi'] for r in h4_data]
chi_rho, chi_p = stats.spearmanr(distances_h4, chi_values_h4)

# Дополнительно: сравнение near vs far
d_mid = (D_MIN + D_MAX) / 2
near_mask = [d <= d_mid for d in distances_h4]
far_mask = [d > d_mid for d in distances_h4]
```

```
mean_deltaF_near = np.mean([abs_deltaF_h4[i] for i in range(len(h4_data)) if
mean_deltaF_far = np.mean([abs_deltaF_h4[i] for i in range(len(h4_data)) if
near_far_ratio = mean_deltaF_near / (mean_deltaF_far + 1e-10)

print(f"\n{'d_hops':>8} | {'|ΔF|':>12} | {'χ':>15}")
print("-" * 40)
# Показываем выборку
step = max(1, len(h4_data)//15)
for i in range(0, len(h4_data), step):
    r = h4_data[i]
    print(f"{r['d_hops']:>8} | {r['abs_deltaF']:>12.4f} | {r['chi']:>15.4f}"

print("\n" + "=" * 70)
print("ИТОГ ГИПОТЕЗЫ H4 (Spearman корреляция)")
print("=" * 70)
print(f"  Spearman ρ(d, |ΔF|) = {locality_rho:.4f}")
print(f"  p-value = {locality_p:.2e}")
print(f"  Порог: ρ ≤ {THRESHOLDS['h4_spearman_rho_max']}, p ≤ {THRESHOLDS['h
print(f"\n  Spearman ρ(d, χ) = {chi_rho:.4f} (p={chi_p:.2e})")
print(f"\n  Отношение near/far:")
print(f"    mean|ΔF| (d≤{d_mid:.0f}): {mean_deltaF_near:.4f}")
print(f"    mean|ΔF| (d>{d_mid:.0f}): {mean_deltaF_far:.4f}")
print(f"    ratio = {near_far_ratio:.2f}")

# H4 подтверждена если эффект убывает (отрицательная корреляция)
h4_confirmed = (locality_rho <= THRESHOLDS['h4_spearman_rho_max'] and
                locality_p <= THRESHOLDS['h4_spearman_p_max'])

h4_metrics = {
    'spearman_rho': float(locality_rho),
    'spearman_p': float(locality_p),
    'chi_spearman_rho': float(chi_rho),
    'chi_spearman_p': float(chi_p),
    'near_far_ratio': float(near_far_ratio)
}

print(f"\n  Гипотеза H4: {'✅ ПОДТВЕРЖДЕНА' if h4_confirmed else '❌ ОТКЛОНЕ
```

```
======================================================================
ЧАСТЬ 7: ГИПОТЕЗА H4 — ПРОСТРАНСТВЕННАЯ ЛОКАЛЬНОСТЬ
======================================================================

Источник χ: узел 256
η = 0.5

  d_hops |         |ΔF| |               χ
----------------------------------------
      50 |       5.5334 |   30000368.6953
      44 |       5.2624 |   30000458.9504
      36 |       5.4680 |   30000528.3431
      31 |       5.3937 |   30000587.6282
      30 |       5.3521 |   30000600.9127
      23 |       5.4219 |   30000672.6298
      18 |       4.9711 |   30000743.0802
      10 |       5.8527 |   30000804.9847
       7 |       4.9596 |   30000852.6042
      13 |       5.9768 |   30000766.6424
      18 |       5.4347 |   30000726.3904
      25 |       5.2072 |   30000661.6816
      29 |       4.9262 |   30000636.3206
      33 |       4.8694 |   30000600.6617
      43 |       4.9778 |   30000493.9469
      48 |       5.1207 |   30000430.4568

======================================================================
ИТОГ ГИПОТЕЗЫ H4 (Spearman корреляция)
======================================================================
  Spearman ρ(d, |ΔF|) = -0.3672
  p-value = 7.51e-10
  Порог: ρ ≤ -0.3, p ≤ 0.05

  Spearman ρ(d, χ) = -0.9925 (p=1.69e-241)

  Отношение near/far:
    mean|ΔF| (d≤28): 5.4185
    mean|ΔF| (d>28): 5.1557
    ratio = 1.05

  Гипотеза H4: ✅ ПОДТВЕРЖДЕНА
```

Part 8: Технические тесты (Test 7-9)

In [31]:

```python
# ======================================================================
# ЧАСТЬ 8: ТЕХНИЧЕСКИЕ ТЕСТЫ (Test 7-9)
# ======================================================================

print("=" * 70)
print("ЧАСТЬ 8: ТЕХНИЧЕСКИЕ ТЕСТЫ")
print("=" * 70)

test_results = {}
test7_data = {}
test8_data = {}
```

```
test9_data = {}

# ------------------------------------------------------------------------
# TEST 7: χ удовлетворяет уравнению Лапласа L_reg·χ = ρ_χ
# ------------------------------------------------------------------------
print("\n" + "-" * 50)
print("TEST 7: χ-поле удовлетворяет уравнению Лапласа")
print("-" * 50)

# Проверяем L_reg·χ ≈ ρ_χ (с учётом регуляризации)
rho_chi = np.zeros(RSL_N)
for node, strength in antigrav.chi_field.sources.items():
    rho_chi[node] = strength

L = world.graph.laplacian
REG_LAMBDA = 1e-6
L_reg = L.copy()
L_reg[0, 0] += REG_LAMBDA  # Та же регуляризация что при решении

L_chi = L_reg @ antigrav.chi_field.chi

# Невязка относительно регуляризованного уравнения
residual_norm = np.linalg.norm(L_chi - rho_chi)
rho_norm = np.linalg.norm(rho_chi)
residual_ratio = residual_norm / (rho_norm + 1e-10)

test7_pass = residual_ratio < THRESHOLDS['test7_residual_max']

test7_data = {
    'residual_ratio': float(residual_ratio),
    'residual_norm': float(residual_norm),
    'rho_norm': float(rho_norm),
    'reg_lambda': REG_LAMBDA,
    'chi_norm': float(np.linalg.norm(antigrav.chi_field.chi)),
    'threshold': THRESHOLDS['test7_residual_max']
}

test_results['test7'] = {
    'name': 'χ Laplacian equation (regularized)',
    'passed': bool(test7_pass),
    'metrics': test7_data
}

print(f"  Уравнение: L_reg·χ = ρ_χ (с регуляризацией λ={REG_LAMBDA})")
print(f"  ||L_reg·χ - ρ_χ|| / ||ρ_χ|| = {residual_ratio:.2e}")
print(f"  Порог: < {THRESHOLDS['test7_residual_max']:.0e}")
print(f"  Test 7: {'✅ PASS' if test7_pass else '❌ FAIL'}")

# ------------------------------------------------------------------------
# TEST 8: Детерминизм (одинаковый seed даёт одинаковый результат)
# ------------------------------------------------------------------------
print("\n" + "-" * 50)
print("TEST 8: Детерминизм")
print("-" * 50)

det_config = AntigravityConfig(chi_coupling=ETA_DEFAULT, initial_capacity=10
```

```
det1 = AntigravityLayer(world=world, config=det_config)
det1.add_source(center_node, strength=CHI_STRENGTH)
det1.update()
chi1 = det1.chi_field.chi.copy()
chi1_hash = hashlib.sha256(chi1.tobytes()).hexdigest()[:16]

det2 = AntigravityLayer(world=world, config=det_config)
det2.add_source(center_node, strength=CHI_STRENGTH)
det2.update()
chi2 = det2.chi_field.chi.copy()
chi2_hash = hashlib.sha256(chi2.tobytes()).hexdigest()[:16]

chi_diff = np.max(np.abs(chi1 - chi2))
test8_pass = chi_diff < THRESHOLDS['test8_max_diff']

test8_data = {
    'max_abs_diff': float(chi_diff),
    'chi1_hash': chi1_hash,
    'chi2_hash': chi2_hash,
    'threshold': THRESHOLDS['test8_max_diff']
}

test_results['test8'] = {
    'name': 'Determinism',
    'passed': bool(test8_pass),
    'metrics': test8_data
}

print(f"  max|χ1 - χ2| = {chi_diff:.2e}")
print(f"  χ1 hash: {chi1_hash}")
print(f"  χ2 hash: {chi2_hash}")
print(f"  Порог: < {THRESHOLDS['test8_max_diff']:.0e}")
print(f"  Test 8: {'✅ PASS' if test8_pass else '❌ FAIL'}")

# ------------------------------------------------------------------------
# TEST 9: Baseline совместимость (без χ-источников Φ_eff = φ)
# ------------------------------------------------------------------------
print("\n" + "-" * 50)
print("TEST 9: Baseline совместимость (без χ-источников)")
print("-" * 50)

empty_antigrav = AntigravityLayer(world=world, config=det_config)
# НЕ добавляем источники

phi_eff_empty = empty_antigrav.get_effective_potential()
baseline_diff = np.max(np.abs(phi_eff_empty - world.phi))

test9_pass = baseline_diff < THRESHOLDS['test9_max_diff']

test9_data = {
    'max_abs_diff': float(baseline_diff),
    'threshold': THRESHOLDS['test9_max_diff'],
    'note': 'No χ sources'
}
```

```python
test_results['test9'] = {
    'name': 'Baseline compatibility (empty)',
    'passed': bool(test9_pass),
    'metrics': test9_data
}

print(f"  Без χ-источников: max|Φ_eff - φ| = {baseline_diff:.2e}")
print(f"  Порог: < {THRESHOLDS['test9_max_diff']:.0e}")
print(f"  Test 9: {'✅ PASS' if test9_pass else '❌ FAIL'}")

# ---------------------------------------------------------------------------
# Сводка технических тестов
# ---------------------------------------------------------------------------
print("\n" + "=" * 50)
print("СВОДКА ТЕХНИЧЕСКИХ ТЕСТОВ")
print("=" * 50)

n_passed = sum(1 for r in test_results.values() if r['passed'])
n_total_tests = len(test_results)

for name, result in test_results.items():
    status = '✅' if result['passed'] else '❌'
    print(f"  {status} {name}: {result['name']}")

print(f"\n  Пройдено: {n_passed}/{n_total_tests}")
```

```
======================================================================
ЧАСТЬ 8: ТЕХНИЧЕСКИЕ ТЕСТЫ
======================================================================

--------------------------------------------------
TEST 7: χ-поле удовлетворяет уравнению Лапласа
--------------------------------------------------
  Уравнение: L_reg·χ = ρ_χ (с регуляризацией λ=1e-06)
  ||L_reg·χ - ρ_χ|| / ||ρ_χ|| = 8.58e-09
  Порог: < 1e-06
  Test 7: ✅ PASS

--------------------------------------------------
TEST 8: Детерминизм
--------------------------------------------------
  max|χ1 - χ2| = 0.00e+00
  χ1 hash: b167dc8fbc6e35cc
  χ2 hash: b167dc8fbc6e35cc
  Порог: < 1e-12
  Test 8: ✅ PASS

--------------------------------------------------
TEST 9: Baseline совместимость (без χ-источников)
--------------------------------------------------
  Без χ-источников: max|Φ_eff - φ| = 0.00e+00
  Порог: < 1e-10
  Test 9: ✅ PASS

==================================================
СВОДКА ТЕХНИЧЕСКИХ ТЕСТОВ
==================================================
  ✅ test7: χ Laplacian equation (regularized)
  ✅ test8: Determinism
  ✅ test9: Baseline compatibility (empty)

  Пройдено: 3/3
```

Part 9: Итоговый отчёт и артефакты

In [32]:

```python
# ======================================================================
# ЧАСТЬ 9: ИТОГОВЫЙ ОТЧЁТ И АРТЕФАКТЫ
# ======================================================================

print("=" * 70)
print("ЭКСПЕРИМЕНТ D: ИТОГОВЫЙ ОТЧЁТ")
print("=" * 70)

# Собираем все результаты
all_hypotheses = {
    'H1': {
        'name': 'Инверсия силы тяготения',
        'confirmed': h1_confirmed,
        'score': h1_score,
        'metrics': {
            'inv_rate': h1_inv_rate,
```

```
            'inv_count': h1_inv_count,
            'total_nodes': h1_total_nodes,
            'threshold': THRESHOLDS['h1_inv_rate_min'],
            'd_range': f'{D_MIN}..{D_MAX}'
        }
    },
    'H2': {
        'name': 'Монотонность по η (Spearman)',
        'confirmed': h2_confirmed,
        'score': 1.0 if h2_confirmed else 0.0,
        'metrics': {
            'spearman_rho': float(h2_rho),
            'spearman_p': float(h2_p),
            'threshold_rho_min': THRESHOLDS['h2_spearman_rho_min'],
            'threshold_p_max': THRESHOLDS['h2_spearman_p_max']
        }
    },
    'H3': {
        'name': 'Baseline при η=0',
        'confirmed': h3_confirmed,
        'score': 1.0 if h3_confirmed else 0.0,
        'metrics': {
            'max_abs_diff': float(h3_max_diff),
            'threshold': THRESHOLDS['h3_baseline_max'],
            'test_nodes': len(test_nodes_h3),
            'd_range': f'{D_MIN}..{D_MAX}'
        }
    },
    'H4': {
        'name': 'Пространственная локальность (Spearman)',
        'confirmed': h4_confirmed,
        'score': 1.0 if h4_confirmed else 0.0,
        'metrics': {
            'spearman_rho': float(locality_rho),
            'spearman_p': float(locality_p),
            'chi_correlation_rho': float(chi_rho),
            'near_far_ratio': float(near_far_ratio),
            'threshold_rho_max': THRESHOLDS['h4_spearman_rho_max'],
            'threshold_p_max': THRESHOLDS['h4_spearman_p_max']
        }
    }
}

all_tests = {
    'Test7': test_results['test7'],
    'Test8': test_results['test8'],
    'Test9': test_results['test9']
}

# Гипотезы
print("\n📊 ГИПОТЕЗЫ:")
print("-" * 50)
for h_id, h_data in all_hypotheses.items():
    status = '✅' if h_data['confirmed'] else '❌'
    print(f"  {status} {h_id}: {h_data['name']} (score: {h_data['score']:.2%
```

```python
n_hyp_confirmed = sum(1 for h in all_hypotheses.values() if h['confirmed'])
print(f"\n  Подтверждено гипотез: {n_hyp_confirmed}/{len(all_hypotheses)}")

# Тесты
print("\n🔬 ТЕХНИЧЕСКИЕ ТЕСТЫ:")
print("-" * 50)
for t_id, t_data in all_tests.items():
    status = '✅' if t_data['passed'] else '❌'
    print(f"  {status} {t_id}: {t_data['name']}")

n_tests_passed = sum(1 for t in all_tests.values() if t['passed'])
print(f"\n  Пройдено тестов: {n_tests_passed}/{len(all_tests)}")

# Общий счёт
total_items = len(all_hypotheses) + len(all_tests)
total_passed = n_hyp_confirmed + n_tests_passed
final_score = total_passed / total_items

print("\n" + "=" * 50)
print("📈 ФИНАЛЬНЫЙ СЧЁТ")
print("=" * 50)
print(f"  Гипотезы: {n_hyp_confirmed}/{len(all_hypotheses)}")
print(f"  Тесты: {n_tests_passed}/{len(all_tests)}")
print(f"  ОБЩИЙ: {total_passed}/{total_items} ({final_score:.2%})")

# ============================================================================
# СОХРАНЕНИЕ АРТЕФАКТОВ
# ============================================================================

print("\n" + "=" * 50)
print("💾 СОХРАНЕНИЕ АРТЕФАКТОВ")
print("=" * 50)

# 1. forces.parquet - детальные данные по узлам
forces_df = pd.DataFrame(h1_force_data)
forces_path = RUN_TABLES_DIR / 'forces.parquet'
forces_df.to_parquet(forces_path, index=False)
print(f"  forces.parquet: {len(forces_df)} rows")

# 2. h2_sweep.parquet - sweep по η
h2_df = pd.DataFrame(h2_sweep_data)
h2_path = RUN_TABLES_DIR / 'h2_sweep.parquet'
h2_df.to_parquet(h2_path, index=False)
print(f"  h2_sweep.parquet: {len(h2_df)} rows")

# 3. test7_residuals.csv
test7_df = pd.DataFrame([test7_data])
test7_path = RUN_TABLES_DIR / 'test7_residuals.csv'
test7_df.to_csv(test7_path, index=False)
print(f"  test7_residuals.csv: 1 row")

# 4. test8_determinism.csv
test8_df = pd.DataFrame([test8_data])
test8_path = RUN_TABLES_DIR / 'test8_determinism.csv'
test8_df.to_csv(test8_path, index=False)
print(f"  test8_determinism.csv: 1 row")
```

```python
# 5. test9_baseline.csv
test9_df = pd.DataFrame([test9_data])
test9_path = RUN_TABLES_DIR / 'test9_baseline.csv'
test9_df.to_csv(test9_path, index=False)
print(f"  test9_baseline.csv: 1 row")

# ============================================================================
# RECOMPUTE CONTRACT
# ============================================================================

recompute_contract = {
    'description': 'Formulas to recompute all metrics from raw tables',
    'metrics': {
        'H1_inv_rate': {
            'table': 'forces.parquet',
            'filter': f'd_hops >= {D_MIN} AND d_hops <= {D_MAX}',
            'formula': 'COUNT(inverted=True) / COUNT(*)',
            'threshold': f'>= {THRESHOLDS["h1_inv_rate_min"]}'
        },
        'H2_spearman': {
            'table': 'h2_sweep.parquet',
            'group_by': 'eta',
            'agg': 'AVG(inverted) as inv_rate',
            'formula': 'scipy.stats.spearmanr(eta_values, inv_rates)',
            'threshold': f'rho >= {THRESHOLDS["h2_spearman_rho_min"]}, p <=
        },
        'H3_baseline': {
            'table': 'forces.parquet',
            'note': 'Computed separately with eta=0',
            'formula': 'MAX(ABS(F_eff - F_normal)) for nodes in d_min..d_max
            'threshold': f'< {THRESHOLDS["h3_baseline_max"]}'
        },
        'H4_locality': {
            'table': 'forces.parquet',
            'formula': 'scipy.stats.spearmanr(d_hops, ABS(deltaF))',
            'threshold': f'rho <= {THRESHOLDS["h4_spearman_rho_max"]}, p <=
        },
        'Test7_residual': {
            'table': 'test7_residuals.csv',
            'formula': 'residual_ratio = ||L_reg·χ - ρ_χ|| / ||ρ_χ||',
            'threshold': f'< {THRESHOLDS["test7_residual_max"]}'
        },
        'Test8_determinism': {
            'table': 'test8_determinism.csv',
            'formula': 'max_abs_diff = MAX(ABS(χ1 - χ2))',
            'threshold': f'< {THRESHOLDS["test8_max_diff"]}'
        },
        'Test9_baseline': {
            'table': 'test9_baseline.csv',
            'formula': 'max_abs_diff = MAX(ABS(Φ_eff - φ)) with no χ sources
            'threshold': f'< {THRESHOLDS["test9_max_diff"]}'
        }
    }
}
```

```python
# ============================================================================
# MANIFEST
# ============================================================================

def compute_sha256(filepath):
    """Compute SHA256 hash of a file."""
    sha256 = hashlib.sha256()
    with open(filepath, 'rb') as f:
        for chunk in iter(lambda: f.read(8192), b''):
            sha256.update(chunk)
    return sha256.hexdigest()

artifact_files = [
    forces_path,
    h2_path,
    test7_path,
    test8_path,
    test9_path
]

artifacts_manifest = []
for fpath in artifact_files:
    if fpath.exists():
        artifacts_manifest.append({
            'filename': fpath.name,
            'relative_path': str(fpath.relative_to(RUN_DIR)),
            'sha256': compute_sha256(fpath),
            'size_bytes': fpath.stat().st_size
        })

# ============================================================================
# FULL REPORT
# ============================================================================

report = {
    'experiment': 'D',
    'name': 'Antigravity via χ-field',
    'version': VERSION,
    'run_id': RUN_ID,
    'timestamp': datetime.now().isoformat(),
    'git_info': git_info,
    'parameters': {
        'N': RSL_N,
        'alpha': RSL_ALPHA,
        'L': RSL_L,
        'seed': BASE_SEED,
        'd_min': D_MIN,
        'd_max': D_MAX,
        'eta_default': ETA_DEFAULT,
        'eta_values': ETA_VALUES,
        'chi_strength': CHI_STRENGTH,
        'mass': MASS
    },
    'thresholds': THRESHOLDS,
    'hypotheses': all_hypotheses,
    'tests': {k: {'passed': v['passed'], 'name': v['name'], 'metrics': v['me
```

```
            for k, v in all_tests.items()},
    'summary': {
        'hypotheses_confirmed': n_hyp_confirmed,
        'hypotheses_total': len(all_hypotheses),
        'tests_passed': n_tests_passed,
        'tests_total': len(all_tests),
        'final_score': final_score,
        'status': 'PASS' if final_score >= 0.8 else 'FAIL'
    },
    'recompute_contract': recompute_contract,
    'artifacts': artifacts_manifest
}

# Сохраняем отчёт
report_path = RUN_DIR / 'report.json'
with open(report_path, 'w') as f:
    json.dump(report, f, indent=2, default=str)
print(f"\n  report.json saved")

# Сохраняем manifest отдельно
manifest = {
    'experiment': 'D',
    'run_id': RUN_ID,
    'version': VERSION,
    'timestamp': datetime.now().isoformat(),
    'artifacts': artifacts_manifest,
    'report_sha256': compute_sha256(report_path)
}

manifest_path = RUN_DIR / 'manifest.json'
with open(manifest_path, 'w') as f:
    json.dump(manifest, f, indent=2)
print(f"  manifest.json saved")

# Также сохраняем в data/sparc для совместимости
sparc_report_path = Path(project_root) / 'data' / 'sparc' / 'experiment_D_re
with open(sparc_report_path, 'w') as f:
    json.dump(report, f, indent=2, default=str)
print(f"  data/sparc/experiment_D_report.json saved")

print(f"\n📁 Артефакты в: {RUN_DIR.relative_to(project_root)}")

print("\n" + "=" * 70)
if final_score >= 0.8:
    print("🎉 ЭКСПЕРИМЕНТ D УСПЕШЕН!")
else:
    print("⚠️ ЭКСПЕРИМЕНТ D ТРЕБУЕТ ДОРАБОТКИ")
print("=" * 70)
```

```
================================================================================
ЭКСПЕРИМЕНТ D: ИТОГОВЫЙ ОТЧЁТ
================================================================================

📊 ГИПОТЕЗЫ:
--------------------------------------------------
  ✅ H1: Инверсия силы тяготения (score: 100.00%)
  ✅ H2: Монотонность по η (Spearman) (score: 100.00%)
  ✅ H3: Baseline при η=0 (score: 100.00%)
  ✅ H4: Пространственная локальность (Spearman) (score: 100.00%)

  Подтверждено гипотез: 4/4

🔬 ТЕХНИЧЕСКИЕ ТЕСТЫ:
--------------------------------------------------
  ✅ Test7: χ Laplacian equation (regularized)
  ✅ Test8: Determinism
  ✅ Test9: Baseline compatibility (empty)

  Пройдено тестов: 3/3

==================================================
📈 ФИНАЛЬНЫЙ СЧЁТ
==================================================
  Гипотезы: 4/4
  Тесты: 3/3
  ОБЩИЙ: 7/7 (100.00%)

==================================================
💾 СОХРАНЕНИЕ АРТЕФАКТОВ
==================================================
  forces.parquet: 511 rows
  h2_sweep.parquet: 2376 rows
  test7_residuals.csv: 1 row
  test8_determinism.csv: 1 row
  test9_baseline.csv: 1 row

  report.json saved
  manifest.json saved
  data/sparc/experiment_D_report.json saved

📁 Артефакты в: data/experiment_D/20251220_054152_0f04eb87

================================================================================
🎉 ЭКСПЕРИМЕНТ D УСПЕШЕН!
================================================================================
```