Challenge 02:

Gravitational Positivity & Causality Bounds

Comprehensive Technical Report

Domain:	Quantum Gravity & Particle Physics
Difficulty:	High
Timeline:	3–9 months
Prerequisites:	Scattering amplitudes, dispersion relations, convex optimization, sum-of-squares programming

Pure Thought AI Challenges
January 19, 2026

Contents

1 Executive Summary

This challenge addresses a fundamental question in quantum gravity: **Which low-energy effective field theories can consistently couple to gravity?** Effective field theories (EFTs) with gravity cannot be arbitrary—quantum consistency, unitarity, causality, and analyticity impose stringent constraints on the Wilson coefficients of higher-derivative terms in the gravitational action.

Analysis Note

This problem lies at the heart of the **Swampland program**, which aims to distinguish theories that can be UV-completed into consistent quantum gravity (the "landscape") from those that cannot (the "swampland"). Rigorous bounds derived here would be *mathematical no-go theorems*, not heuristic arguments.

The approach uses **positivity bounds** from dispersion relations, combined with **causality constraints** from shockwave scattering, formulated as **semidefinite programs (SDPs)** with machine-verifiable certificates.

2 Scientific Context and Motivation

2.1 The Swampland Program

Not every consistent-looking quantum field theory can arise as the low-energy limit of a theory of quantum gravity. The **Swampland program** seeks to identify the constraints that separate:

- **Landscape:** Theories that *can* be UV-completed into quantum gravity
- **Swampland:** Theories that *cannot* be UV-completed

Physical Insight

Why This Matters for Physics: If we observe certain Wilson coefficients in nature (e.g., through gravitational wave observations or particle physics experiments), Swampland bounds tell us whether quantum gravity *allows* those values. Violations would indicate inconsistency in our theoretical framework.

2.2 Higher-Derivative Gravity

Consider the most general effective action for gravity in four dimensions:

$$\boxed{S = \int d^4x\sqrt{-g}\left[M_{\rm pl}^2 R + a_1 R^2 + a_2 R_{\mu\nu}R^{\mu\nu} + a_3 R_{\mu\nu\rho\sigma}R^{\mu\nu\rho\sigma} + \cdots\right]} \tag{1}$$

where:

- $M_{\rm pl}$ is the Planck mass
- R is the Ricci scalar
- a_1, a_2, a_3 are dimensionful **Wilson coefficients**
- Higher-order terms (R^3, R^4, etc.) are suppressed by higher powers of $M_{\rm pl}^{-1}$

2.3 The Core Question

Central Research Question

What are the sharp bounds on Wilson coefficients $\{a_1, a_2, a_3, \ldots\}$ imposed purely by consistency conditions?
Specifically, which ranges of these coefficients are compatible with:

1. **Unitarity:** Positive-norm states in the quantum theory
2. **Causality:** No superluminal signal propagation
3. **Analyticity:** Scattering amplitudes satisfy dispersion relations
4. **Crossing symmetry:** $s \leftrightarrow t \leftrightarrow u$ symmetry of amplitudes

2.4 Why Rigorous Bounds Matter

(1) Swampland Constraints: Determines which low-energy theories can couple to gravity

(2) Phenomenology: Constrains quantum gravity corrections to General Relativity

(3) Mathematical Rigor: Produces actual no-go theorems, not heuristics

(4) Testability: Bounds can be confronted with observations (gravitational waves, cosmology)

3 Mathematical Formulation

3.1 Graviton-Graviton Scattering

Consider the scattering amplitude $\mathcal{M}(s,t)$ for graviton-graviton scattering, where s, t, u are **Mandelstam variables**:

$$s = (p_1 + p_2)^2 \quad \text{(center-of-mass energy squared)} \tag{2}$$
$$t = (p_1 - p_3)^2 \quad \text{(momentum transfer squared)} \tag{3}$$
$$u = (p_1 - p_4)^2 \quad \text{(crossing channel)} \tag{4}$$

with the constraint $s + t + u = 0$ (for massless gravitons).

Analysis Note

The amplitude $\mathcal{M}(s,t)$ encodes all information about graviton interactions. The Wilson coefficients $\{a_i\}$ appear as polynomial corrections to the Einstein gravity amplitude at low energies.

3.2 Physical Constraints

3.2.1 Constraint 1: Unitarity

The optical theorem relates the imaginary part of the forward amplitude to the total cross-section:

$$\boxed{\text{Im}\mathcal{M}(s,t) \geq 0 \quad \text{in the physical region}} \tag{5}$$

This follows from probability conservation in quantum mechanics.

3.2.2 Constraint 2: Analyticity

The amplitude $\mathcal{M}(s,t)$ is **analytic** in the complex s-plane except on **physical cuts** (where particles can go on-shell):

- Right-hand cut: $s \geq s_{\text{th}}$ (threshold for particle production)
- Left-hand cut: $u \geq u_{\text{th}}$ (crossed channel)

3.2.3 Constraint 3: Crossing Symmetry

For identical particles, the amplitude must be symmetric under exchange of Mandelstam variables:

$$\boxed{\mathcal{M}(s,t) = \mathcal{M}(t,s) = \mathcal{M}(u,t)} \tag{6}$$

3.2.4 Constraint 4: Causality (Shockwave Positivity)

Causality implies no superluminal propagation. In the **eikonal approximation**, scattering off a gravitational shockwave gives a time delay:

$$\delta t(b) \geq 0 \quad \text{for all impact parameters } b \tag{7}$$

This translates to **positivity constraints** on certain combinations of Wilson coefficients.

3.2.5 Constraint 5: Regge Boundedness

At high energies (large $|s|$ with fixed $t < 0$), the amplitude must satisfy:

$$|\mathcal{M}(s,t)| \lesssim s^2 \quad \text{as } |s| \to \infty \tag{8}$$

This ensures the dispersion relation converges.

3.3 Dispersion Relations

For fixed $t < 0$, analyticity and the Regge bound imply a **dispersion relation**:

$$\boxed{\mathcal{M}(s,t) = \mathcal{M}(0,t) + \frac{s^2}{\pi} \int_{s_{\text{th}}}^{\infty} ds' \frac{\text{Im}\mathcal{M}(s',t)}{s'^2(s'-s)}} \tag{9}$$

Analysis Note

Key Insight: The dispersion relation expresses the amplitude in terms of its imaginary part on the physical cut. Combined with unitarity ($\text{Im}\mathcal{M} \geq 0$), this gives **positivity bounds** on the low-energy expansion coefficients.

3.4 Wilson Coefficient Extraction

The low-energy expansion of the amplitude is:

$$\mathcal{M}(s,t) = \mathcal{M}_{\text{GR}}(s,t) + \sum_{n,m} c_{nm} s^n t^m \tag{10}$$

where c_{nm} are related to the Wilson coefficients $\{a_i\}$. The dispersion relation constrains these coefficients.

3.5 Optimization Formulation

The problem becomes a **semidefinite program (SDP)**:

$$\begin{aligned} \text{Maximize/Minimize:} \quad & a_i \\ \text{Subject to:} \quad & \text{Dispersion relation holds} \\ & \text{Im}\mathcal{M}(s,t) \geq 0 \quad \text{(unitarity)} \\ & \mathcal{M}(s,t) = \mathcal{M}(t,s) \quad \text{(crossing)} \\ & \delta t(b) \geq 0 \quad \text{(causality)} \\ & |\mathcal{M}| \lesssim s^2 \quad \text{(Regge bound)} \end{aligned} \tag{11}$$

Research Direction

Sum-of-Squares (SoS) Formulation: The positivity constraints can be written as polynomial inequalities, which are certified by expressing them as sums of squares. This gives *machine-verifiable proofs* of the bounds.

4 Certificates of Correctness

4.1 For Allowed Regions

If a point $\{a_1^*, a_2^*, \ldots\}$ is in the allowed region:

1. Provide explicit Wilson coefficients $\{a_i^*\}$
2. Construct explicit scattering amplitude $\mathcal{M}(s,t)$ satisfying all constraints
3. Verify unitarity numerically at multiple (s,t) points
4. Verify crossing symmetry and dispersion relation

4.2 For Forbidden Regions

If a region is forbidden, provide a **dual SoS certificate**:

1. A polynomial $p(s,t)$ such that:
   - $p(s,t)$ is positive on the physical region
   - $\int p(s,t) \cdot [\text{violated constraint}]\, ds\, dt < 0$
2. This proves mathematically that no amplitude can satisfy all constraints in that region
3. Export to SoS verification format for independent checking

5 Implementation Approach

5.1 Phase 1: Tree-Level Graviton Scattering (Months 1–2)

Goal: Build amplitude calculator for graviton-graviton scattering.

5.1.1 Einstein Gravity Amplitude

The tree-level amplitude in pure General Relativity:

```python
def einstein_amplitude(s: float, t: float, M_pl: float) -> complex:
    """
    Pure GR amplitude for graviton-graviton -> graviton-graviton

    At tree level, this is proportional to the famous expression
    involving the Mandelstam variables.

    Args:
        s: Center-of-mass energy squared
        t: Momentum transfer squared
        M_pl: Planck mass

    Returns:
        Tree-level amplitude M(s,t)
    """
    u = -s - t  # Mandelstam relation for massless particles

    # Schematic form (full expression involves polarization tensors)
    # The actual amplitude has poles at s=0, t=0, u=0
    amplitude = (s * t * u) / M_pl**2

    return amplitude
```

Listing 1: Einstein gravity amplitude

5.1.2 Higher-Derivative Corrections

```python
def corrected_amplitude(s: float, t: float, M_pl: float,
                        wilson_coeffs: dict) -> complex:
    """
    Amplitude including R^2, R^3, ... corrections

    The Wilson coefficients parametrize higher-derivative terms:
    - a1: coefficient of R^2
    - a2: coefficient of (Ricci)^2
    - a3: coefficient of (Riemann)^2
    - etc.

    Args:
        s, t: Mandelstam variables
        M_pl: Planck mass
        wilson_coeffs: Dictionary of Wilson coefficients

    Returns:
        Corrected amplitude
    """
    u = -s - t

    # Leading Einstein gravity contribution
    M_0 = einstein_amplitude(s, t, M_pl)

    # R^2 correction (dimension 4 operator)
    # Contributes at order s^2, t^2, u^2
    a1 = wilson_coeffs.get('a1', 0)
    M_R2 = a1 * (s**2 + t**2 + u**2) / M_pl**4

    # R^3 correction (dimension 6 operator)
    a2 = wilson_coeffs.get('a2', 0)
```

```
    M_R3 = a2 * (s**3 + t**3 + u**3) / M_pl**6

    # Gauss-Bonnet combination
    a3 = wilson_coeffs.get('a3', 0)
    M_GB = a3 * (s**2 * t + s * t**2 + s**2 * u +
                 s * u**2 + t**2 * u + t * u**2) / M_pl**6

    return M_0 + M_R2 + M_R3 + M_GB
```

Listing 2: Amplitude with Wilson coefficient corrections

5.1.3 Crossing Symmetry Verification

```
def verify_crossing_symmetry(amplitude_func, s: float, t: float,
                             tol: float = 1e-10) -> bool:
    """
    Verify that M(s,t) = M(t,s) = M(u,t)

    Args:
        amplitude_func: Function M(s, t) -> complex
        s, t: Test point
        tol: Tolerance for equality check

    Returns:
        True if crossing symmetry holds
    """
    u = -s - t

    M_st = amplitude_func(s, t)
    M_ts = amplitude_func(t, s)
    M_ut = amplitude_func(u, t)

    check1 = abs(M_st - M_ts) < tol
    check2 = abs(M_st - M_ut) < tol

    if not (check1 and check2):
        print(f"Crossing violation: M(s,t)={M_st}, M(t,s)={M_ts}, M(u,t)={M_ut}
    ")
        return False

    return True
```

Listing 3: Crossing symmetry check

Critical Consideration

Polarization Structure: The full graviton amplitude involves polarization tensors $\epsilon_{\mu\nu}$. For positivity bounds, we typically work with specific helicity configurations (e.g., all-plus or MHV) or average over polarizations. Ensure consistency in the chosen convention.

5.2 Phase 2: Dispersion Relations (Months 2–3)

Goal: Implement forward dispersion relation and derive positivity constraints.

```
import numpy as np
from scipy.integrate import quad

def check_dispersion_relation(M_func, s: float, t: float,
                              s_th: float, s_max: float,
                              epsilon: float = 1e-6) -> float:
```

```
    """
    Check if M(s,t) satisfies the dispersion relation

    M(s,t) = M(0,t) + s^2/pi * integral ds' Im M(s',t)/(s'^2(s'-s))

    Args:
        M_func: Amplitude function M(s, t) -> complex
        s: Evaluation point
        t: Fixed momentum transfer (should be < 0)
        s_th: Threshold energy squared
        s_max: Upper cutoff for integral
        epsilon: Imaginary part for contour

    Returns:
        Residual |LHS - RHS| (should be ~ 0 if dispersion holds)
    """
    # Left-hand side: direct evaluation
    lhs = M_func(s, t)

    # Subtraction constant
    M_0 = M_func(0, t)

    # Dispersive integral
    def integrand(s_prime):
        # Evaluate slightly above real axis to get imaginary part
        M_above = M_func(s_prime + 1j * epsilon, t)
        Im_M = M_above.imag

        # Dispersion kernel
        denominator = s_prime**2 * (s_prime - s)
        return Im_M / denominator

    # Numerical integration
    integral, error = quad(integrand, s_th, s_max, limit=100)

    dispersive_part = (s**2 / np.pi) * integral

    rhs = M_0 + dispersive_part

    residual = abs(lhs - rhs)
    return residual
```

Listing 4: Dispersion relation implementation

Analysis Note

Subtracted Dispersion Relations: For amplitudes with polynomial growth, we use *subtracted* dispersion relations. The number of subtractions is determined by the Regge behavior. For gravity, $\mathcal{M} \sim s^2$ at large s, requiring two subtractions.

5.3 Phase 3: Causality from Shockwave Scattering (Months 3–4)

Goal: Implement causality constraints from eikonal scattering.

5.3.1 Eikonal Phase Shift

The eikonal phase shift $\delta(b)$ at impact parameter b is:

$$\delta(b) = \frac{1}{2s} \int \frac{d^2\mathbf{q}}{(2\pi)^2} e^{i\mathbf{q}\cdot\mathbf{b}} \mathcal{M}(s, t = -\mathbf{q}^2) \tag{12}$$

Causality requires: $\delta(b) \geq 0$ for all b.

```python
from scipy.special import j0
from scipy.integrate import quad

def eikonal_phase(b: float, s: float, M_func,
                  q_max: float = 100) -> float:
    """
    Compute eikonal phase shift delta(b)

    delta(b) = (1/2s) * FT of M(s, t=-q^2)

    For azimuthally symmetric case, this becomes a Hankel transform:
    delta(b) = (1/4*pi*s) * integral dq q J_0(qb) M(s, -q^2)

    Args:
        b: Impact parameter
        s: Center-of-mass energy squared
        M_func: Amplitude function
        q_max: UV cutoff for integral

    Returns:
        Phase shift delta(b)
    """
    def integrand(q):
        t = -q**2
        M_val = M_func(s, t).real  # Take real part for phase
        return q * j0(q * b) * M_val

    integral, _ = quad(integrand, 0, q_max)

    delta = integral / (4 * np.pi * s)
    return delta


def check_causality(M_func, s: float, b_values: list) -> dict:
    """
    Check causality constraint: delta(b) >= 0 for all b

    Args:
        M_func: Amplitude function
        s: Energy
        b_values: List of impact parameters to check

    Returns:
        Dictionary with causality check results
    """
    results = {'passed': True, 'violations': []}

    for b in b_values:
        delta = eikonal_phase(b, s, M_func)
        if delta < -1e-10:  # Allow small numerical errors
            results['passed'] = False
            results['violations'].append((b, delta))

    return results
```

Listing 5: Eikonal phase shift calculation

Physical Insight

Physical Meaning: The eikonal phase $\delta(b)$ represents the phase acquired by a graviton wave packet scattering off a gravitational shockwave at impact parameter b. A negative phase would correspond to the wave arriving *before* it was sent—a violation of causality.

5.4 Phase 4: SDP Formulation and Solver (Months 4–6)

Goal: Set up the optimization problem and solve for bounds.

```python
import cvxpy as cp
import numpy as np

def setup_positivity_sdp(n_coeffs: int, grid_s: np.ndarray,
                         grid_t: np.ndarray,
                         impact_params: np.ndarray) -> cp.Problem:
    """
    Set up SDP for bounding Wilson coefficients

    Variables: Wilson coefficients a = [a1, a2, ..., a_n]
    Constraints:
        1. Unitarity: Im M >= 0
        2. Causality: delta(b) >= 0
        3. Crossing: M(s,t) = M(t,s)
        4. Regge bound: implicit in dispersion relation

    Args:
        n_coeffs: Number of Wilson coefficients
        grid_s, grid_t: Grid points for constraint sampling
        impact_params: Impact parameters for causality

    Returns:
        cvxpy Problem object
    """
    # Variables: Wilson coefficients
    a = cp.Variable(n_coeffs)

    constraints = []

    # 1. Unitarity: Im M(s,t) >= 0 at grid points
    for s_i in grid_s:
        for t_i in grid_t:
            if s_i > 0 and t_i < 0:  # Physical region
                # Im M is linear in Wilson coefficients (for EFT expansion)
                Im_M = compute_imaginary_amplitude(s_i, t_i, a)
                constraints.append(Im_M >= 0)

    # 2. Causality: eikonal phase positivity
    for b_i in impact_params:
        # delta(b) is linear in Wilson coefficients
        delta = compute_eikonal_linear(b_i, a)
        constraints.append(delta >= 0)

    # 3. Crossing symmetry is automatic if we parametrize
    # the amplitude in crossing-symmetric combinations

    return a, constraints


def bound_wilson_coefficient(coeff_index: int, n_coeffs: int,
                             direction: str = 'max') -> dict:
    """
```

```
    Find upper or lower bound on a single Wilson coefficient

    Args:
        coeff_index: Which coefficient to bound (0-indexed)
        n_coeffs: Total number of coefficients
        direction: 'max' for upper bound, 'min' for lower bound

    Returns:
        Dictionary with bound value and dual certificate
    """
    a, constraints = setup_positivity_sdp(n_coeffs, ...)

    # Objective
    if direction == 'max':
        objective = cp.Maximize(a[coeff_index])
    else:
        objective = cp.Minimize(a[coeff_index])

    # Solve
    problem = cp.Problem(objective, constraints)
    problem.solve(solver=cp.MOSEK, verbose=True)

    if problem.status == cp.OPTIMAL:
        return {
            'status': 'optimal',
            'bound': a[coeff_index].value,
            'coefficients': a.value,
            'dual_certificate': [c.dual_value for c in constraints]
        }
    else:
        return {'status': problem.status}
```

Listing 6: SDP setup for Wilson coefficient bounds

5.5 Phase 5: Sum-of-Squares Certificates (Months 6–9)

Goal: Generate machine-verifiable SoS certificates for the bounds.

```
from sympy import symbols, expand, Poly
from sympy.polys.polytools import factor_list

def find_sos_certificate(bound_polynomial, variables):
    """
    Find Sum-of-Squares decomposition for proving bounds

    Given a polynomial p(x) that should be non-negative on a region,
    find polynomials {g_i} such that:
        p(x) = sum_i g_i(x)^2 + sum_j h_j(x) * constraint_j(x)

    where constraint_j are the defining constraints of the region.

    Args:
        bound_polynomial: Polynomial to prove non-negative
        variables: List of symbolic variables

    Returns:
        SoS decomposition (list of polynomials)
    """
    # This uses SDP to find the Gram matrix Q such that
    # p(x) = v(x)^T Q v(x) where v(x) is vector of monomials
    # and Q is positive semidefinite

    # Implementation uses SOSTOOLS-like approach
```

```python
    # (Details depend on specific SDP solver)

    pass  # Implementation details


def verify_sos_certificate(polynomial, sos_decomposition,
                           test_points: int = 1000) -> bool:
    """
    Verify that SoS decomposition is correct

    Check:
    1. Sum of squares equals original polynomial
    2. Evaluate at random points to confirm non-negativity
    """
    # Reconstruct from SoS
    reconstructed = sum(p**2 for p in sos_decomposition)
    diff = expand(polynomial - reconstructed)

    # Check algebraic equality
    if diff != 0:
        print(f"SoS reconstruction error: {diff}")
        return False

    # Numerical spot-check
    import numpy as np
    for _ in range(test_points):
        point = np.random.randn(len(variables))
        val = float(polynomial.subs(list(zip(variables, point))))
        if val < -1e-8:
            print(f"Negative value found: {val} at {point}")
            return False

    return True
```

Listing 7: SoS certificate generation

Analysis Note

SoS Background: A polynomial $p(x)$ is a *sum of squares* if $p(x) = \sum_i g_i(x)^2$ for some polynomials g_i. Finding such a decomposition is a convex problem (SDP), and the existence of an SoS decomposition proves $p(x) \geq 0$ everywhere. This gives a *certificate* of non-negativity.

6 Detailed Research Directions

6.1 Direction 1: Single-Coefficient Bounds

Research Direction

First Target: Bound the coefficient a_1 of the R^2 term.
Approach:

1. Set all other coefficients $a_2 = a_3 = \cdots = 0$
2. Maximize and minimize a_1 subject to unitarity + causality
3. Extract both upper and lower bounds
4. Generate SoS certificate for each bound

Expected Outcome: A rigorous interval $[a_1^{\min}, a_1^{\max}]$ with machine-checkable proof.

6.2 Direction 2: Multi-Parameter Bounds

Research Direction

Goal: Map the allowed region in (a_1, a_2, a_3) space.
Approach:

1. Fix one coefficient and bound the others
2. Repeat for different fixed values to trace out boundary
3. Use convex hull algorithms to characterize the allowed region
4. Generate SoS certificates for the boundary

Visualization: 3D plot with allowed region (green) and forbidden region (red).

6.3 Direction 3: Spinning Amplitudes

Extend beyond graviton-graviton scattering:

- Graviton + scalar (matter coupling)
- Graviton + photon (gravity-EM coupling)
- Multiple graviton scattering (higher-point amplitudes)

These probe different Wilson coefficients and give complementary bounds.

6.4 Direction 4: Loop Corrections

The tree-level analysis can be extended to include:

- One-loop corrections (UV divergences require renormalization)
- Anomalous dimensions of higher-derivative operators
- Running of Wilson coefficients with energy scale

6.5 Direction 5: Connection to String Theory

Research Direction

Consistency Check: String theory provides explicit UV completions of gravity. Compute Wilson coefficients in string theory and verify they lie within the derived bounds. This serves as both a check on the bounds and a test of string theory's consistency with the positivity constraints.

7 Success Criteria

7.1 Minimum Viable Result (3 months)

- ✓ Tree-level amplitude calculator implemented and tested
- ✓ Crossing symmetry verified numerically
- ✓ Dispersion relation implemented
- ✓ **Single coefficient bound:** Rigorous bounds on a_1 (R^2 coefficient)
- ✓ Dual certificate extracted and verified independently

7.2 Strong Result (6 months)

- ✓ Multi-parameter bounds on $\{a_1, a_2, a_3\}$
- ✓ Allowed region in 3D parameter space characterized
- ✓ Boundary certified with SoS decompositions
- ✓ Novel bounds (tighter than literature or new multi-coupling constraints)
- ✓ Machine-checkable certificates in standard format

7.3 Publication-Quality Result (9 months)

- ✓ All Wilson coefficients up to dimension 8 operators bounded
- ✓ Full allowed region characterized with phase diagram
- ✓ Formal proofs in Lean 4 or Isabelle/HOL
- ✓ Comparison with string theory predictions
- ✓ Publication with proof repository

8 Verification Protocol

```
def verify_bound_certificate(coeff_bounds: dict,
                             dual_certificate: list) -> str:
    """
    Verify that the dual certificate proves the claimed bound.

    Steps:
    1. Check SoS decomposition (if applicable)
    2. Verify positivity on physical region via sampling
    3. Check that bound is actually saturated
```

```
    Args:
        coeff_bounds: Dictionary with claimed bounds
        dual_certificate: The certificate proving the bound

    Returns:
        "VERIFIED" if all checks pass, else error description
    """
    # 1. Check SoS decomposition
    sos_polynomials = extract_sos_from_certificate(dual_certificate)
    reconstructed = sum(p**2 for p in sos_polynomials)

    if not is_polynomial_equal(reconstructed, constraint_polynomial):
        return "FAILED: SoS reconstruction mismatch"

    # 2. Verify positivity on physical region
    test_points = generate_physical_region_samples(n=1000)
    for s, t in test_points:
        cert_val = evaluate_certificate(dual_certificate, s, t)
        if cert_val < -1e-10:
            return f"FAILED: Negative certificate at (s,t)=({s},{t})"

    # 3. Check bound is saturated correctly
    critical_amplitude = construct_amplitude_from_certificate(
        dual_certificate
    )

    if not verify_unitarity(critical_amplitude):
        return "FAILED: Unitarity violated at critical point"
    if not verify_causality(critical_amplitude):
        return "FAILED: Causality violated at critical point"
    if not verify_crossing(critical_amplitude):
        return "FAILED: Crossing violated at critical point"

    return "VERIFIED"
```

Listing 8: Bound verification function

8.1 Exported Artifacts

1. **Bound certificate:** `bound_a1.sos`
   - Sum-of-Squares decomposition
   - Exact rational coefficients (when possible)
   - Verifiable by independent SoS checkers
2. **Allowed region:** `allowed_region.json`

```
{
    "coefficients": ["a1", "a2", "a3"],
    "bounds": {
        "a1": {"min": -0.5, "max": 0.5},
        "a2": {"min": 0.0, "max": 1.0},
        "a3": {"min": -0.3, "max": 0.7}
    },
    "constraints_used": ["unitarity", "causality", "crossing"],
    "certificate_files": ["bound_a1.sos", "bound_a2.sos", ...]
}
```

3. **Formal proof:** `gravitational_bounds.lean`

- Lean 4 formalization of the positivity argument
- Machine-checked theorem statements
- Importable certificate data

9 Common Pitfalls and Mitigations

9.1 Numerical Instabilities in Dispersion Integrals

Critical Consideration

Problem: The dispersive integral $\int ds' \, \mathrm{Im}\mathcal{M}/(s'^2(s'-s))$ has a pole at $s'=s$ and may converge slowly.
Solutions:

- Use principal value integration or contour deformation
- Employ adaptive quadrature with error control
- For numerical bounds, verify convergence by varying cutoffs

9.2 Insufficient Grid Resolution

Critical Consideration

Problem: Constraints sampled on a coarse grid may miss violations between grid points.
Solutions:

- Start with coarse grid, then refine near constraint boundaries
- Use adaptive grid refinement based on dual variables
- Final verification should use dense grid ($> 10^4$ points)

9.3 Crossing Symmetry Breaking

Critical Consideration

Problem: Parametrizing amplitudes in a way that breaks crossing symmetry.
Solution:

- Always use manifestly crossing-symmetric variables
- Examples: $\sigma_2 = s^2 + t^2 + u^2$, $\sigma_3 = s^3 + t^3 + u^3$
- Verify crossing numerically at every step

10 Milestone Checklist

10.1 Phase 1: Amplitude Infrastructure (Months 1–2)

☐ Einstein gravity amplitude implemented

☐ Higher-derivative corrections added

☐ Crossing symmetry verified numerically

☐ Polarization/helicity conventions documented

10.2 Phase 2: Dispersion Relations (Months 2–3)

☐ Forward dispersion relation implemented

☐ Subtracted dispersion relation for $\mathcal{M} \sim s^2$

☐ Numerical convergence verified

☐ Connection to positivity bounds established

10.3 Phase 3: Causality Constraints (Months 3–4)

☐ Eikonal phase shift calculator implemented

☐ Shockwave positivity constraints derived

☐ Constraints translated to Wilson coefficient space

☐ Combined with unitarity constraints

10.4 Phase 4: Optimization (Months 4–6)

☐ SDP formulation complete

☐ Single-coefficient bounds obtained

☐ Dual certificates extracted

☐ Multi-parameter scans initiated

10.5 Phase 5: Certificates & Verification (Months 6–9)

☐ SoS decompositions computed

☐ Independent verification performed

☐ Lean formalization begun

☐ Publication draft prepared

11 Resources and References

11.1 Key Papers

1. Adams et al. (2006): "Causality, analyticity and an IR obstruction to UV completion" [hep-th/0602178] — Foundational paper on positivity bounds

2. Bellazzini et al. (2016): "Softness and amplitudes' positivity for spinning particles" [arXiv:1605.06111] — Extension to spinning particles

3. Caron-Huot et al. (2021): "Sharp boundaries for the swampland" [arXiv:2102.08951] — State-of-the-art gravitational bounds

4. Tolley et al. (2020): "Positivity bounds from multiple vacua and their cosmological consequences" [arXiv:2011.06081]

5. de Rham et al. (2022): "Gravitational positivity bounds" [arXiv:2201.05178]

11.2 Software

- **CVXPY:** Convex optimization in Python — `pip install cvxpy`
- **MOSEK:** Commercial SDP solver (free academic license) — https://mosek.com
- **SOSTOOLS:** MATLAB toolbox for SoS programming
- **Macaulay2:** Computer algebra for algebraic geometry
- **Lean 4:** Proof assistant — https://lean-lang.org

11.3 Background Reading

- Elvang & Huang: "Scattering Amplitudes in Gauge Theory and Gravity" (Cambridge)
- Eden et al.: "The Analytic S-Matrix" (Cambridge) — Classic on dispersion relations
- Parrilo: "Semidefinite programming relaxations for semialgebraic problems" (PhD thesis) — SoS theory

12 Conclusion

Gravitational positivity bounds represent a frontier where fundamental physics (unitarity, causality) meets modern optimization (SDP, SoS). The bounds derived here are not heuristics but *mathematical theorems* with machine-verifiable proofs.

Success in this challenge would:

1. Provide rigorous constraints on the Swampland
2. Give testable predictions for quantum gravity corrections
3. Establish new connections between optimization theory and theoretical physics
4. Produce a library of certified bounds for future research

The methodology—combining dispersion relations, causality, and convex optimization with formal verification—represents a paradigm for rigorous theoretical physics.