PRD 10: Flat Chern Bands with Provable Geometry

Pure Thought AI Challenge 10

Pure Thought AI Challenges Project

January 18, 2026

Abstract

This document presents a comprehensive Product Requirement Document (PRD) for implementing a pure-thought computational challenge. The problem can be tackled using only symbolic mathematics, exact arithmetic, and fresh code—no experimental data or materials databases required until final verification. All results must be accompanied by machine-checkable certificates.

Contents

Domain: Materials Science
Timeline: 6-9 months
Difficulty: High
Prerequisites: Topological band theory, quantum geometry, Lie algebras, algebraic geometry

---

0.1 1. Problem Statement

0.1.1 Scientific Context

Flat bands—bands with near-zero dispersion (dE/dk 0)—are platforms for strongly correlated physics because the kinetic energy is quenched, making interactions dominant. When flat bands also carry non-trivial **Chern numbers**, they can host exotic quantum states:

- **Fractional Chern Insulators (FCI)**: Lattice analogs of fractional quantum Hall states
- **Fractional Quantum Anomalous Hall Effect**: Quantized Hall conductance at fractional filling
- **Topological Superconductivity**: Pairing instabilities in flat Chern bands

The **quantum geometry** of flat bands—encoded in the **quantum metric tensor** g*(k) and* ***Berry curvature*** *F*(k)—determines many physical properties:

- **Ideal Flatness**: F(k) = const (uniform Berry curvature)
- **Stability Ratio**: S = ⟨F⟩^{2} / ⟨g⟩ quantifies susceptibility to interactions
- **Trace Condition**: Tr(g) = (C/Area) × (integral over BZ) for Chern number C

Recent Developments:

- Twisted bilayer graphene (TBG) exhibits narrow Chern bands
- Moiré materials show tunable flat band physics
- "Magic angle" corresponds to optimal quantum geometry

0.1.2 Core Question

Can we construct tight-binding models with perfectly flat Chern bands (zero dispersion) and prove that their quantum geometry is optimal for fractional Chern insulator states?

Specifically:

- Given target Chern number C, construct a tight-binding Hamiltonian with:
- Exactly flat band: E(k) = E = const
- Uniform Berry curvature: F(k) = C/(2 × Area$_BZ$)

- Maximal stability ratio S
- Prove the model is "ideal" (cannot be improved)
- Certify geometric properties using exact algebra

0.1.3 Why This Matters

Theoretical Impact:

- Identifies fundamental limits on flat band geometry
- Provides benchmarks for realistic materials (TBG, moiré systems)
- Connects topology to quantum information (Fubini-Study metric)

Practical Benefits:

- Guides design of photonic/cold atom simulators
- Predicts optimal platforms for fractional Chern insulators
- Enables engineering of interaction-dominated regimes

Pure Thought Advantages:

- Quantum metric is purely geometric (no experimental input)
- Flatness can be proven algebraically via energy eigenvalues
- Ideal models often have exact solutions (Landau levels, symmetric spaces)
- No need for DFT or materials databases

0.2 2. Mathematical Formulation

0.2.1 Problem Definition

A **flat Chern band** is a single-particle Hamiltonian H(k) on the Brillouin zone BZ (a torus T^2) such that:

- **Flatness**: One band has constant energy E$_n(k) = EkBZ$
- **Topology**: The flat band has Chern number C 0
- **Quantum Geometry**: The quantum metric g*(k) and Berry curvature F*(k) satisfy optimality conditions

Quantum Metric Tensor:

```
g _    (k) =   R  e  _   u_n(k) | (1 - | u _ n u_n |) |    _   u_n(k)
```

where $|$u*n(k)⟩ is the Bloch wavefunction for the flat band (= /*k$_{)}$.

Berry Curvature:

```
F_xy(k) =   I  m  _x   u_n(k) |   _y   u_n(k)    - (x     y)
```

Ideality Criterion:
A flat Chern band is "ideal" if:

- **Uniform curvature**: F(k) = C/(Area$_BZ) = const$
- **Trace bound**: Tr(g(k)) = |F(k)| (saturates Cauchy-Schwarz inequality)

Ideal flat bands have wavefunctions forming **coherent states** on a complex manifold (e.g., Landau levels are coherent states on ^{1}).

0.2.2 Input/Output Specification

Input:

```python
from sympy import Symbol, Matrix, sqrt, exp, I, pi
from typing import Callable, Tuple

class FlatBandModel:
    hamiltonian: Callable[[np.ndarray], np.ndarray]  # H(k)
    flat_band_index: int  # Which band is flat (0-indexed)
    target_chern: int  # Desired C
    num_orbitals: int  # Dimension of H(k)
```

Output:

```python
class FlatBandCertificate:
    model: FlatBandModel

    # Flatness verification
    energy_dispersion: float  # max_k E(k) - min_k E(k), should be ~0
    flatness_error: float  #  _E  / E_mean

    # Topological invariants
    chern_number: int  # Exact integer
    berry_curvature_variance: float  # Var[F(k)], should be ~0 for ideal

    # Quantum geometry
    quantum_metric: Callable[[np.ndarray], np.ndarray]  # g_   (k)
    trace_condition: float  #     Tr(g) - |C| / Area_BZ
    stability_ratio: float  # S =     F     /  Tr (g)

    # Ideality proof
    is_ideal: bool
    coherent_state_manifold: Optional[str]  # e.g., "CP^1", "Flag(2,4)"
    embedding_map: Optional[Callable]  # k     point on complex manifold

    # Verification artifacts
    wavefunction_samples: List[np.ndarray]  # |u_n(k)    at grid points
    proof_of_flatness: str  # Algebraic proof that E(k) = const
```

0.3 3. Implementation Approach

0.3.1 Phase 1: Landau Level Benchmarks (Months 1-2)

Start with exactly solvable models—Landau levels in a magnetic field:

```python
import numpy as np
from scipy.special import hermite
from sympy import *
import mpmath as mp

def landau_level_wavefunction(n: int, k: np.ndarray, magnetic_length: float = 1.0) -> complex:
    """
    Landau level wavefunction in symmetric gauge.
```

```

     _n (x,y) = (1/   (2^n n!          )) exp(-r  /4     ) H_n(r/  2     ) exp(i n   )

    where     = magnetic length =    (   /eB)
    """
    x, y = k[0], k[1]
    r = np.sqrt(x**2 + y**2)
    theta = np.arctan2(y, x)

    ell = magnetic_length
    normalization = 1.0 / np.sqrt(2**n * mp.factorial(n) * np.sqrt(np.pi) * ell)

    # Hermite polynomial H_n
    H_n = hermite(n)

    psi = normalization * np.exp(-r**2 / (4*ell**2)) * H_n(r / (np.sqrt(2)*ell)) * np.exp(1j*n*theta)

    return psi

def landau_berry_curvature(n: int) -> float:
    """
    Berry curvature for Landau level n (constant across k-space).

    F_xy = 1 /       = eB/    (magnetic field strength)

    Chern number for lowest Landau level: C = 1
    """
    magnetic_length = 1.0
    return 1.0 / magnetic_length**2

def landau_quantum_metric(n: int, k: np.ndarray, magnetic_length: float = 1.0) -> np.ndarray:
    """
    Quantum metric for Landau level (Fubini-Study metric on         ).

    For LLL (n=0):
     g_    = (1/4     )    _

    For higher LL:
     g_    = (1/4     ) [   _    + (x_   x_  / 2     (n+1))]
    """
    ell = magnetic_length

    if n == 0:
        # Lowest Landau level: isotropic metric
        return np.eye(2) / (4 * ell**2)
    else:
        # Higher Landau levels: anisotropic correction
        x, y = k[0], k[1]
        r2 = x**2 + y**2
        g = np.eye(2) / (4*ell**2)
        g += np.outer([x, y], [x, y]) / (8*ell**4 * (n+1))
        return g
```

```python

def verify_trace_condition_landau(n: int) -> bool:
    """
    Verify that Tr(g) = |F| for Landau levels (ideality condition).

    For LLL:
    Tr(g) = 2    (1/4     ) = 1/(2     )
    F = 1/

    Ratio: Tr(g) / F = 1/2     1     LLL is NOT ideal for trace condition!

    (But it IS ideal in a different sense: maximal stability)
    """
    ell = 1.0
    Tr_g = 2 * (1 / (4*ell**2))  # = 1/(2     )
    F = 1 / ell**2

    print(f"Landau level {n}:")
    print(f"  Tr(g) = {Tr_g:.6f}")
    print(f"  F = {F:.6f}")
    print(f"  Ratio = {Tr_g / F:.6f}")

    return np.isclose(Tr_g, F)
```

Validation: Verify LLL has C=1, perfectly flat E(k)=/2, and uniform F(k).

0.3.2 Phase 2: Lattice Flat Band Models (Months 2-4)

Construct discrete lattice versions of flat Chern bands:

```python
def kapit_mueller_model(N: int, alpha: float) -> Tuple[Callable, int]:
    """
    Kapit-Mueller model: Lattice Landau levels on a square lattice.

    N: Linear system size
    alpha: Effective flux per plaquette (   = p/q rational)

    Returns: (Hamiltonian, Chern number)

    For    = 1/4, has exactly flat C=1 band (lattice LLL).
    """
    def H(k: np.ndarray) -> np.ndarray:
        kx, ky = k[0], k[1]

        # Model parameters
        flux = alpha * 2*np.pi

        # Hopping with Peierls substitution
        t_x = np.cos(kx)
        t_y = np.cos(ky + flux*kx)  # Landau gauge

        # Magnetic Hamiltonian
        H_mat = np.zeros((N, N), dtype=complex)

        # Fill in hopping terms...
        # (Full implementation requires band projection operators)

        return H_mat

    C = int(np.round(alpha))  # Chern number     flux
    return H, C

def wannier_flatband_projector(H_func: Callable, band_idx: int,
                               N_k: int = 100) -> Callable:
    """
    Construct projector onto a single flat band using Wannier states.

    P(k) = |u_n(k)    u_n    (k)|

    For exactly flat bands, this projector has special properties.
    """
    # Discretize BZ
    kx_grid = np.linspace(0, 2*np.pi, N_k, endpoint=False)
    ky_grid = np.linspace(0, 2*np.pi, N_k, endpoint=False)

    projectors = {}

    for kx in kx_grid:
        for ky in ky_grid:
            k = np.array([kx, ky])
            evals, evecs = np.linalg.eigh(H_func(k))

            # Select flat band
            sorted_idx = np.argsort(evals)
            flat_state = evecs[:, sorted_idx[band_idx]]

            P_k = np.outer(flat_state, flat_state.conj())
            projectors[(kx, ky)] = P_k

    def get_projector(k: np.ndarray) -> np.ndarray:
        # Nearest neighbor interpolation
        kx_idx = np.argmin(np.abs(kx_grid - k[0]))
        ky_idx = np.argmin(np.abs(ky_grid - k[1]))
        return projectors[(kx_grid[kx_idx], ky_grid[ky_idx])]

    return get_projector

def compute_quantum_metric(H_func: Callable, band_idx: int,
                           k: np.ndarray, delta: float = 1e-5) ->
                               np.ndarray:
    """
    Compute quantum metric tensor g_    (k) via finite differences.

     g_    =  R e _  u |  _   u  -  R e _  u |  u u  |
          _   u
    """
    # Get wavefunction at k
    evals, evecs = np.linalg.eigh(H_func(k))
    sorted_idx = np.argsort(evals)
    u_k = evecs[:, sorted_idx[band_idx]]

```

```python
    g = np.zeros((2, 2))

    for mu in range(2):
        dk_mu = np.zeros(2)
        dk_mu[mu] = delta

        evals_plus, evecs_plus = np.linalg.eigh(H_func(k + dk_mu))
        sorted_idx_plus = np.argsort(evals_plus)
        u_k_plus = evecs_plus[:, sorted_idx_plus[band_idx]]

        # Finite difference derivative
        du_mu = (u_k_plus - u_k) / delta

        for nu in range(2):
            dk_nu = np.zeros(2)
            dk_nu[nu] = delta

            evals_plus_nu, evecs_plus_nu = np.linalg.eigh(H_func(k + dk_nu))
            sorted_idx_nu = np.argsort(evals_plus_nu)
            u_k_plus_nu = evecs_plus_nu[:, sorted_idx_nu[band_idx]]

            du_nu = (u_k_plus_nu - u_k) / delta

            # Quantum metric formula
            overlap = np.vdot(du_mu, du_nu)
            proj_mu = np.vdot(du_mu, u_k)
            proj_nu = np.vdot(u_k, du_nu)

            g[mu, nu] = np.real(overlap - proj_mu * proj_nu)

    return g
```

Test Cases:

- Kapit-Mueller at =1/4: exactly flat C=1 band
- Hofstadter model at rational flux
- Chern-Simons-matter duals

0.3.3 Phase 3: Ideal Flat Band Construction (Months 4-6)

Systematically construct ideal flat bands using algebraic geometry:

```python
def coherent_state_flatband(manifold: str, embedding_dim: int) -> FlatBandModel:
    """
    Construct flat Chern band from coherent states on a complex manifold.

    Ideal flat bands correspond to holomorphic line bundles on:
    -        ^n: Complex projective space (Landau levels)
    - Flag( n  , n  , ..., n  ): Flag manifolds (SU(N) WZW models)
    - G/H: Symmetric spaces (coset constructions)

    Returns tight-binding model with exactly flat band.
    """
    if manifold == "CP1":
        #              S      Landau level on sphere
        return construct_cp1_model(chern=1)

    elif manifold == "CP2":
        #              Generalized Landau levels, C can be higher
        return construct_cp2_model(chern=2)

    elif manifold.startswith("Flag"):
        # Flag manifolds     Multi-component flat bands
        return construct_flag_manifold_model(manifold)

    else:
        raise ValueError(f"Unknown manifold: {manifold}")

def construct_cp1_model(chern: int) -> FlatBandModel:
    """
    Construct tight-binding model with flat band from          geometry.

    Uses Hopf map: S       S

    Chern number = winding of Hopf fibration
    """
    def H(k: np.ndarray) -> np.ndarray:
        kx, ky = k[0], k[1]

        # Stereographic coordinates on S
        z = kx + 1j*ky

        # Bloch Hamiltonian from coherent states
        # H = |  z  z  | where | z   is coherent state

        # Explicit parameterization:
        # | z   = (1/   (1+|z|  )) * [1, z]

        norm_sq = 1 + np.abs(z)**2
        psi = np.array([1, z], dtype=complex) / np.sqrt(norm_sq)

        # Flat band projector
        P = np.outer(psi, psi.conj())

        # Add non-flat bands (orthogonal)
        psi_orth = np.array([-np.conj(z), 1], dtype=complex) /
           np.sqrt(norm_sq)
        P_orth = np.outer(psi_orth, psi_orth.conj())

        # Full Hamiltonian: flat band at E=0, other band at E=1
        H_full = 0 * P + 1 * P_orth

        return H_full

    return FlatBandModel(
        hamiltonian=H,
        flat_band_index=0,
        target_chern=chern,
        num_orbitals=2
    )

def prove_exact_flatness(model: FlatBandModel) -> str:
    """
    Prove algebraically that a band is exactly flat.

    For coherent state models, flatness follows from:
    H |u(k)   =  E   |u(k)     k

    where  E   is independent of k.
    """
    proof = "Proof of Exact Flatness:\n\n"

    # Sample Hamiltonian at multiple k-points
    k_samples = [
        np.array([0, 0]),
        np.array([np.pi, 0]),
        np.array([0, np.pi]),
        np.array([np.pi, np.pi]),
        np.array([np.pi/2, np.pi/3])
    ]

    eigenvalues = []

    for k in k_samples:
        H_k = model.hamiltonian(k)
        evals = np.linalg.eigvalsh(H_k)
        sorted_evals = np.sort(evals)
        flat_band_energy = sorted_evals[model.flat_band_index]
        eigenvalues.append(flat_band_energy)

    # Check variance
    energy_mean = np.mean(eigenvalues)
    energy_std = np.std(eigenvalues)

    proof += f"Flat band index: {model.flat_band_index}\n"
    proof += f"Sampled energies: {eigenvalues}\n"
    proof += f"Mean: {energy_mean:.10f}\n"
    proof += f"Std deviation: {energy_std:.2e}\n\n"

    if energy_std < 1e-10:
        proof += "    Band is EXACTLY flat (   < 10        )\n"
        proof += f"Constant energy:  E   = {energy_mean:.10f}\n"
    else:
        proof += f"    Band has dispersion:    = {energy_std:.2e}\n"

    return proof

def verify_ideal_geometry(model: FlatBandModel, N_k: int = 50) ->
   Tuple[bool, dict]:
    """
    Check if flat band saturates ideality bounds.

    Ideality criteria:
```

```python
    1. Uniform Berry curvature: Var[F(k)] = 0
    2. Trace condition:     Tr(g) = 2  |C|
    3. Stability ratio: S = max possible
    """
    kx_grid = np.linspace(0, 2*np.pi, N_k)
    ky_grid = np.linspace(0, 2*np.pi, N_k)

    F_values = []
    Tr_g_values = []

    for kx in kx_grid:
        for ky in ky_grid:
            k = np.array([kx, ky])

            # Berry curvature
            F = berry_curvature_2d(model.hamiltonian, k,
                [model.flat_band_index])
            F_values.append(F)

            # Quantum metric trace
            g = compute_quantum_metric(model.hamiltonian,
                model.flat_band_index, k)
            Tr_g = np.trace(g)
            Tr_g_values.append(Tr_g)

    # Statistics
    F_mean = np.mean(F_values)
    F_var = np.var(F_values)

    Tr_g_integral = np.mean(Tr_g_values) * (2*np.pi)**2
    expected_integral = 2*np.pi * abs(model.target_chern)

    # Ideality checks
    uniform_curvature = (F_var < 1e-8)
    trace_satisfied = (abs(Tr_g_integral - expected_integral) < 0.01)

    is_ideal = uniform_curvature and trace_satisfied

    diagnostics = {
        'F_mean': F_mean,
        'F_variance': F_var,
        'Tr_g_integral': Tr_g_integral,
        'expected_Tr_g': expected_integral,
        'uniform_curvature': uniform_curvature,
        'trace_condition': trace_satisfied
    }

    return is_ideal, diagnostics
```

0.3.4 Phase 4: Stability Ratio Optimization (Months 6-7)

Optimize quantum geometry for fractional Chern insulator stability:

```python
def compute_stability_ratio(model: FlatBandModel, N_k: int = 100) ->
    float:
    """
    Compute stability ratio S =     F     /   Tr  (g)    .

    Higher S     better platform for fractional Chern insulators.
    Ideal bound: S     2   (saturated by Landau levels on sphere).
    """
    kx_grid = np.linspace(0, 2*np.pi, N_k)
    ky_grid = np.linspace(0, 2*np.pi, N_k)

    F_squared_sum = 0
    Tr_g_sum = 0

    for kx in kx_grid:
        for ky in ky_grid:
            k = np.array([kx, ky])

            F = berry_curvature_2d(model.hamiltonian, k,
               [model.flat_band_index])
            g = compute_quantum_metric(model.hamiltonian,
               model.flat_band_index, k)

            F_squared_sum += F**2
            Tr_g_sum += np.trace(g).real

    F_squared_avg = F_squared_sum / (N_k**2)
    Tr_g_avg = Tr_g_sum / (N_k**2)

    S = F_squared_avg / Tr_g_avg if Tr_g_avg > 0 else 0

    return S

def optimize_model_for_stability(initial_model: FlatBandModel,
                                 param_ranges: dict) -> FlatBandModel:
    """
    Scan parameter space to maximize stability ratio.
    """
    from scipy.optimize import minimize

    def objective(params):
        # Update model with new parameters
        updated_model = update_model_parameters(initial_model, params)

        # Compute stability (negative because we minimize)
        S = compute_stability_ratio(updated_model)

        return -S

    # Constraints: maintain flatness and Chern number
    constraints = [
        {'type': 'eq', 'fun': lambda p: verify_flatness_constraint(p)},
        {'type': 'eq', 'fun': lambda p: verify_chern_constraint(p,
           initial_model.target_chern)}
    ]

    result = minimize(objective, x0=list(param_ranges.values()),
                     method='SLSQP', constraints=constraints)

```

```python
    optimal_params = result.x
    optimal_model = update_model_parameters(initial_model,
        optimal_params)

    return optimal_model
```

0.3.5 Phase 5: Certificate Generation (Months 7-8)

Produce comprehensive verification certificates:

```python
def generate_flat_band_certificate(model: FlatBandModel) ->
    FlatBandCertificate:
    """
    Generate complete certificate for flat Chern band.
    """
    # Compute dispersion
    k_samples = [np.random.uniform(-np.pi, np.pi, 2) for _ in
        range(1000)]
    energies = []

    for k in k_samples:
        evals = np.linalg.eigvalsh(model.hamiltonian(k))
        energies.append(evals[model.flat_band_index])

    dispersion = max(energies) - min(energies)
    flatness_error = np.std(energies) / abs(np.mean(energies)) if
        np.mean(energies) != 0 else 0

    # Compute Chern number
    C = compute_chern_number_exact(model.hamiltonian,
        [model.flat_band_index])

    # Berry curvature statistics
    F_values = [berry_curvature_2d(model.hamiltonian, k,
        [model.flat_band_index])
                for k in k_samples]
    berry_variance = np.var(F_values)

    # Quantum geometry
    g_samples = [compute_quantum_metric(model.hamiltonian,
        model.flat_band_index, k)
                 for k in k_samples]
    Tr_g_mean = np.mean([np.trace(g).real for g in g_samples])
    trace_condition_error = abs(Tr_g_mean * (2*np.pi)**2 -
        2*np.pi*abs(C))

    # Stability ratio
    S = compute_stability_ratio(model)

    # Ideality check
    is_ideal, diagnostics = verify_ideal_geometry(model)

    # Algebraic proof
    proof = prove_exact_flatness(model)

    cert = FlatBandCertificate(
        model=model,
        energy_dispersion=dispersion,
        flatness_error=flatness_error,
        chern_number=C,
        berry_curvature_variance=berry_variance,
        quantum_metric=lambda k:
            compute_quantum_metric(model.hamiltonian,
                                                            model.flat_band_index,
                                                               k),
        trace_condition=trace_condition_error,
        stability_ratio=S,
        is_ideal=is_ideal,
        coherent_state_manifold="CP1" if is_ideal else None,
        proof_of_flatness=proof
    )

    return cert

def export_certificate(cert: FlatBandCertificate, filename: str):
    """Export certificate with all data."""
    import json

    cert_dict = {
        'chern_number': cert.chern_number,
        'flatness_error': float(cert.flatness_error),
        'energy_dispersion': float(cert.energy_dispersion),
        'berry_curvature_variance':
            float(cert.berry_curvature_variance),
        'trace_condition_error': float(cert.trace_condition),
        'stability_ratio': float(cert.stability_ratio),
        'is_ideal': cert.is_ideal,
        'manifold': cert.coherent_state_manifold,
        'num_orbitals': cert.model.num_orbitals,
        'proof_of_flatness': cert.proof_of_flatness
    }

    with open(filename, 'w') as f:
        json.dump(cert_dict, f, indent=2)
```

0.3.6 Phase 6: Database and Applications (Months 8-9)

Build database of optimal flat Chern bands:

```
def generate_flatband_database(max_chern: int = 5) -> dict:
    """
    Generate database of ideal flat Chern bands for C = 1,...,max_chern.
    """
    database = {
        'models': [],
        'timestamp': datetime.now().isoformat()
    }

    for C in range(1, max_chern + 1):
        print(f"Constructing ideal flat band for C = {C}...")

        # Try different manifolds
        for manifold in [f"CP{C}", f"Flag({C},{C+1})"]:
            try:
                model = coherent_state_flatband(manifold, embedding_dim=C+1)
                cert = generate_flat_band_certificate(model)

                if cert.is_ideal:
                    database['models'].append({
                        'chern_number': C,
                        'manifold': manifold,
                        'stability_ratio': cert.stability_ratio,
                        'certificate_path': export_certificate(cert, f'flatband_C{C}.json')
                    })
                    break

            except Exception as e:
                print(f"  Failed for {manifold}: {e}")
                continue

    return database
```

0.4 4. Example Starting Prompt

```
You are a condensed matter theorist specializing in topological flat bands and quantum geometry.
Your task is to construct tight-binding models with perfectly flat Chern bands and prove their
quantum geometry is optimal for fractional Chern insulator physics.

OBJECTIVE: Build ideal flat Chern band models with C = 1,2,3, verify exact flatness, and
certify optimal quantum geometry using algebraic methods only.

PHASE 1 (Months 1-2): Landau level benchmarks
- Implement Landau level wavefunctions in symmetric gauge
- Compute Berry curvature (should be uniform: F = 1/     )
- Calculate quantum metric tensor g_    (k)
- Verify Chern number C = 1 for lowest Landau level

PHASE 2 (Months 2-4): Lattice flat band models
- Implement Kapit-Mueller model at     = 1/4 (lattice Landau level)
- Verify exact flatness:  _E   / E_mean < 10
- Compute quantum metric via finite differences
- Check trace condition:     Tr(g) = 2  |C|

PHASE 3 (Months 4-6): Ideal flat band construction
- Construct           coherent state model (Hopf fibration)
- Prove exact flatness algebraically
- Verify uniform Berry curvature: Var[F(k)] < 10
- Check ideality: F(k) = C/(2      Area)

PHASE 4 (Months 6-7): Stability ratio optimization
```

```
- Compute S =     F     /   Tr  (g)     for all models
- Optimize hopping parameters to maximize S
- Compare to theoretical bound: S     2   (Landau sphere)

PHASE 5 (Months 7-8): Certificate generation
- For each model, generate FlatBandCertificate with:
  * Exact Chern number (integer)
  * Flatness error (should be ~0)
  * Berry curvature variance (should be ~0 for ideal)
  * Stability ratio S
  * Algebraic proof of flatness
- Export as JSON with full quantum geometry data

PHASE 6 (Months 8-9): Database and applications
- Build database of ideal flat bands for C = 1,...,5
- Identify best platforms for fractional Chern insulators
- Predict moir  material parameters matching ideal geometry

SUCCESS CRITERIA:
- MVR: Landau level and Kapit-Mueller models with verified flat bands
- Strong:           model with proven ideal geometry, S optimized
- Publication: Complete database C     5, applications to TBG/moir
    systems

VERIFICATION:
- Flatness verified: energy dispersion < 10
- Chern number exact (integer via Fukui method)
- Ideality proven: uniform curvature + trace condition satisfied
- All certificates exported with algebraic proofs

Use exact symbolic math where possible. No experimental data or DFT
    calculations.
All results must be mathematically rigorous and certificate-based.
```

0.5 5. Success Criteria

0.5.1 Minimum Viable Result (MVR)

Within 2-3 months:

- **Landau Level Implementation**:
- LLL wavefunctions with C = 1 verified
- Berry curvature computed: F = 1/^{2} (uniform)
- Quantum metric: g = *(1/4^2)*
- **Kapit-Mueller Lattice Model**:
- = 1/4 model has exactly flat band
- Flatness error < 10
- Chern number C = 1 verified

- **Basic Quantum Geometry**:
- Finite difference computation of $g_{(k)}$

- Berry curvature variance measured
- Trace condition checked for 2 models

Deliverable: Verified flat bands for Landau level + Kapit-Mueller

0.5.2 Strong Result

Within 5-6 months:

- **Ideal Flat Band Models**:
- ^{1} coherent state model constructed
- Exact flatness proven algebraically
- Uniform Berry curvature: Var[F] $< 10^1$
- Trace condition: error < 1
- **Stability Ratio Analysis**:
- S computed for 10+ flat band models
- Optimization: find model with maximal S for each C
- Comparison to theoretical bounds
- **Certificate System**:
- FlatBandCertificate generated for 10 models
- All certificates exported as JSON
- Proofs of flatness and ideality included

Metrics:

- 10 models with exact flat bands
- 3+ ideal models (uniform curvature)
- Stability ratios documented

0.5.3 Publication-Quality Result

Within 8-9 months:

- **Complete Classification**:
- Ideal flat bands for C = 1,2,3,4,5
- Minimal orbital counts determined
- Connection to K-theory and cobordism
- **Application to Real Materials**:
- Predict optimal moiré twist angles matching ideal geometry
- Identify TBG parameter regimes closest to ^{1} model
- Propose photonic/cold atom realizations
- **Fractional Chern Insulator Predictions**:
- Compute interaction matrix elements for ideal bands
- Predict FCI phase diagram using stability ratio
- Identify best platforms (highest S)
- **Formal Verification**:
- Translate flatness proofs to Lean/Isabelle
- Formally verify trace condition theorem
- Machine-checkable certificates

Publications:

- "Ideal Flat Chern Bands from Complex Geometry"
- "Quantum Geometry Optimization for Fractional Chern Insulators"
- "Pure-Thought Design of Topological Flat Bands"

0.6 6. Verification Protocol

0.6.1 Automated Checks

```python
def verify_flat_band_certificate(cert: FlatBandCertificate) ->
    bool:
    """Verify all certificate claims."""
    checks = []

    # Check 1: Flatness
    checks.append(('Flatness', cert.flatness_error < 1e-6))

    # Check 2: Chern number
    C_recomputed = compute_chern_number_exact(cert.model.hamiltonian,
                                               [cert.model.flat_band_index])
    checks.append(('Chern number', C_recomputed == cert.chern_number))

    # Check 3: Ideality (if claimed)
    if cert.is_ideal:
        checks.append(('Uniform curvature', cert.berry_curvature_variance < 1e-8))
        checks.append(('Trace condition', cert.trace_condition < 0.01))

    # Check 4: Stability ratio bounds
    checks.append(('Stability ratio > 0', cert.stability_ratio > 0))

    for name, passed in checks:
        print(f"{'   ' if passed else '   '} {name}")

    return all(p for _, p in checks)
```

0.6.2 Cross-Validation

- Compare to Landau level exact solutions
- Reproduce twisted bilayer graphene at magic angle
- Check against fractional Chern insulator literature

0.6.3 Exported Artifacts

Certificates in JSON format with all quantum geometry data, plus visualization of Berry curvature fields and quantum metric heatmaps.

0.7 7. Resources Milestones

0.7.1 Key References

- Parameswaran et al. (2013): "Fractional Quantum Hall Physics in Topological Flat Bands"
- Neupert et al. (2011): "Fractional Quantum Hall States at Zero Magnetic Field"
- Roy (2014): "Band Geometry of Fractional Topological Insulators"
- Herzog-Arbeitman et al. (2022): "Quantum Geometry and Stability of Moiré Flatbands"

0.7.2 Common Pitfalls

- **Numerical vs Exact Flatness**: Use symbolic math to verify exact E(k) = const
- **Gauge Dependence**: Quantum metric depends on gauge choice—use gauge-invariant formulas
- **Finite-Size Effects**: Ensure BZ discretization doesn't introduce spurious dispersion

0.7.3 Milestone Checklist

- **Month 2**: Landau level + Kapit-Mueller verified
- **Month 4**: Quantum metric computation working
- **Month 6**: Ideal ^{1} model constructed
- **Month 8**: Database of C=1-5 models complete
- **Month 9**: Application to TBG parameters

0.8 8. Extensions and Open Questions

- **Higher Chern Numbers**: C $>$ 5 ideal models
- **3D Flat Bands**: Weyl semimetals with flat Fermi arcs
- **Interacting Flat Bands**: Many-body Hamiltonians in flat band limit

Long-Term Vision: Provide blueprints for quantum simulators realizing fractional Chern insulator phases without magnetic fields.

End of PRD 10