Assignment: Abstract Math in C++ for AI & Performance Systems

Overview

This curriculum introduces math-rich ideas while progressively developing your C++ skills. Each stage reinforces learning through proofs, linear algebra, and practical modeling. Bonus "black diamond" challenges push the boundaries on performance and parallelization.

1 Stage 1: Class and Function Synthesis — Mathematical Objects in Code

Theme: Modeling Abstract Math Concepts

Project 1: Peano Arithmetic Synthesizer

- **Goal:** Implement natural numbers using the Peano axioms (Zero, Successor).
- **Part 1:** Write a function-based model of addition and multiplication using recursion.
- **Part 2:** Develop a `PeanoNumber` class with operator overloads.
- **Bonus:** Use compile-time recursion with `constexpr` (template metaprogramming preview).
- **Rationale:** Reinforces recursion and inductive thinking (directly related to *How to Prove It*).

Project 2: Vector Space Playground

- **Goal:** Implement a minimal vector space class for 2D/3D.
- **Part 1:** Define a `Vector` class with dot product, scalar multiplication, and magnitude.
- **Part 2:** Enforce axioms via function contracts/assertions.
- **Part 3:** Implement transformations such as projection and extraction of orthogonal components.
- **Bonus:** Optimize the `dot()` function for float vectors using SIMD.

2 Stage 2: Templates, Overloads, and Abstraction — Building Generic Math Libraries

Project 3: Template Matrix Library ($\mathbb{R}^n$)

- **Goal:** Build a templated `Matrix<T, Rows, Cols>` class.
- **Part 1:** Implement basic operations: addition, scalar multiplication, and transpose.
- **Part 2:** Add support for operator overloading and slicing.
- **Part 3:** Develop a Gaussian elimination function.
- **Bonus:** Use `std::span` with alignment-aware storage and benchmark the performance.

Project 4: Symbolic Logic Expression Tree

- **Goal:** Parse and evaluate propositional logic formulas.
- **Part 1:** Construct an `Expr` class hierarchy (including `And`, `Or`, `Not`, `Var`).
- **Part 2:** Recursively evaluate expressions based on a truth assignment.
- **Bonus:** Create a simplifier that applies De Morgan's laws and eliminates double negations.

3 Stage 3: Memory Management & System Modeling — Low-Level Behavior of Abstract Systems

Project 5: Markov Chain Simulator

- **Goal:** Model discrete Markov processes using dynamic memory (state machine).
- **Part 1:** Develop a `State` class with transitions and associated probabilities.
- **Part 2:** Implement Monte Carlo simulation of state transitions.
- **Bonus:** Compare performance between `std::vector` and a custom arena allocator.

Project 6: Multivariate Function Optimizer (Gradient Descent)

- **Goal:** Implement a simple gradient descent system without autodiff.
- **Part 1:** Design a `Function<T>` class with evaluation and gradient estimation.
- **Part 2:** Optimize convex functions such as L2 and quadratic bowls.
- **Bonus:** Use OpenMP to parallelize the gradient estimation.

4 Stage 4: Advanced Structures, Algebraic Systems — Encoding Deep Mathematical Objects

Project 7: Finite Field Arithmetic Library (GF(p))

- **Goal:** Build modular arithmetic types to facilitate polynomial math.
- **Part 1:** Implement a `ModInt<p>` class with overloaded operators and the extended Euclidean algorithm.
- **Part 2:** Add polynomial division and compute the GCD over GF(p).
- **Bonus:** Integrate SIMD acceleration and lookup table optimizations.

Project 8: Hilbert Curve Encoder/Decoder

- **Goal:** Develop a 2D and 3D spatial locality-preserving indexer.
- **Part 1:** Implement the recursive Hilbert curve algorithm from scratch.
- **Part 2:** Encapsulate the solution in a `HilbertMapper` class with unit tests.
- **Bonus:** Benchmark against a Z-order curve and explore cache-aware matrix block layouts.

5 Stage 5: Probabilistic Modeling & Inference — Foundations of AI/ML in Systems Code

Project 9: Bayesian Coin Inference Engine

- **Goal:** Compute the posterior distribution of a coin's bias from observed flips.
- **Part 1:** Develop a `BetaDistribution` class with appropriate update rules.
- **Part 2:** Simulate updates and visualize convergence (export data as CSV).
- **Bonus:** Implement a Monte Carlo sampler using the Box-Muller transform for noise modeling.

Project 10: Kernels & RKHS Prototyper

- **Goal:** Implement common kernel functions and compute the Gram matrix.
- **Part 1:** Code Gaussian/RBF and polynomial kernels.
- **Part 2:** Classify points in toy datasets using the computed kernels.
- **Bonus:** SIMD-optimize the kernel matrix computation and measure its performance.

Optional Side Quests: Black Diamond Challenges

- Write a unit test suite in pure C++.
- Use OpenMP or `std::thread` to parallelize workloads.
- Implement custom memory pools for allocation-heavy classes.
- Add CSV logging/profiling hooks to track runtime performance.