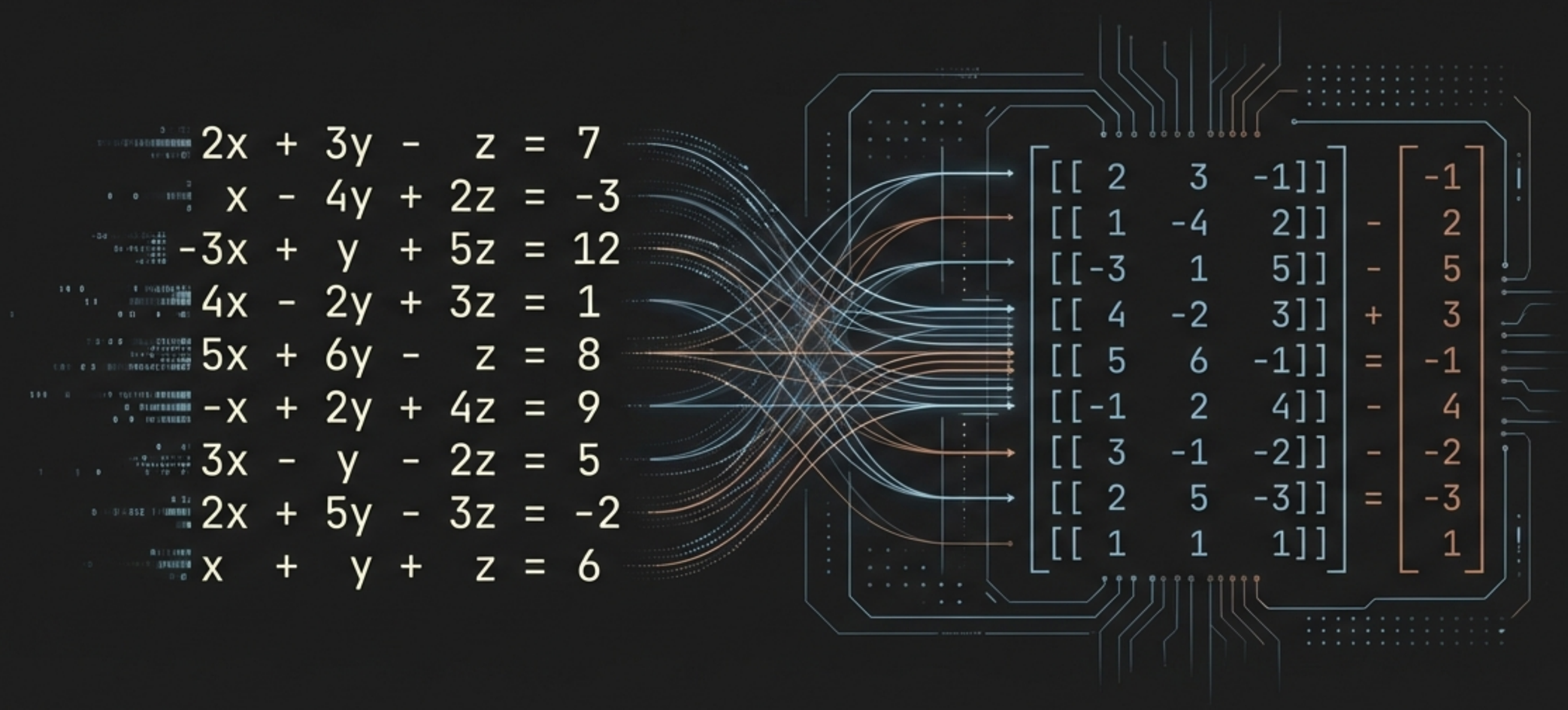
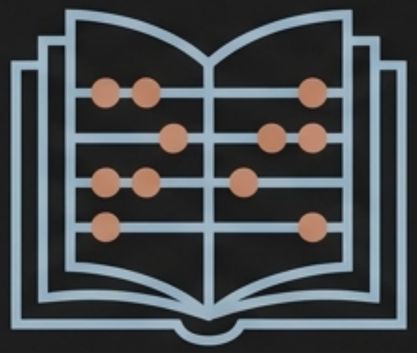
Linear Equations Solver

Building a Dependency-Free Mathematical Engine in Modern C++



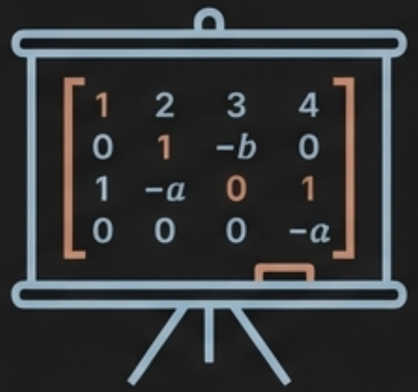
Empowering Mathematical Computation

Students



A lightweight tool to instantly verify hand calculations in linear algebra or numerical methods.

Educators



A zero-setup demonstration tool for teaching Gaussian elimination, Cramer's Rule, and matrix determinants.

Researchers

A dependency-free equation manipulation module suitable for constrained C++ environments.

Engineering Under Strict Constraints

Core Toolkit

C++11 standard, cross-platform compatible (g++, clang++, MSVC).

Zero External Libraries

Complete reliance on raw standard C++ capabilities. No mathematical extensions or UI frameworks.

Zero STL Containers

The ultimate challenge. No std::vector, no std::map. Memory management, dynamic resizing, and data associations handled entirely through custom algorithms and raw pointer manipulation.

Seamless Algebraic Parsing

- Normalizes lowercase variables
- Reduces coefficients of 1 or -1 to signs
- Formats RHS as clean integers

The Interactive Command Loop

Information Retrieval

`num_vars`	Total unique variables
`equation <n>`	Print normalized form
`column <var>`	Extract variable coefficients

Equation Arithmetic

`add <i> <j>`	Add two equations
`subtract <i> <j>`	Subtract equation j from i
`substitute <var> <i> <j>`	Direct algebraic substitution

Solver Execution

`D`	Print coefficient matrix
`D_value`	Compute determinant
`solve`	Execute full Cramer's Rule logic

4-Phase System Architecture Pipeline

1. Input Phase

Text tokenization via `parseLeftSide()` and extraction of variable/coefficient pairs.

2. Command Processing

Active dispatch loop reading from standard input and routing commands to handlers.

3. Computation Phase

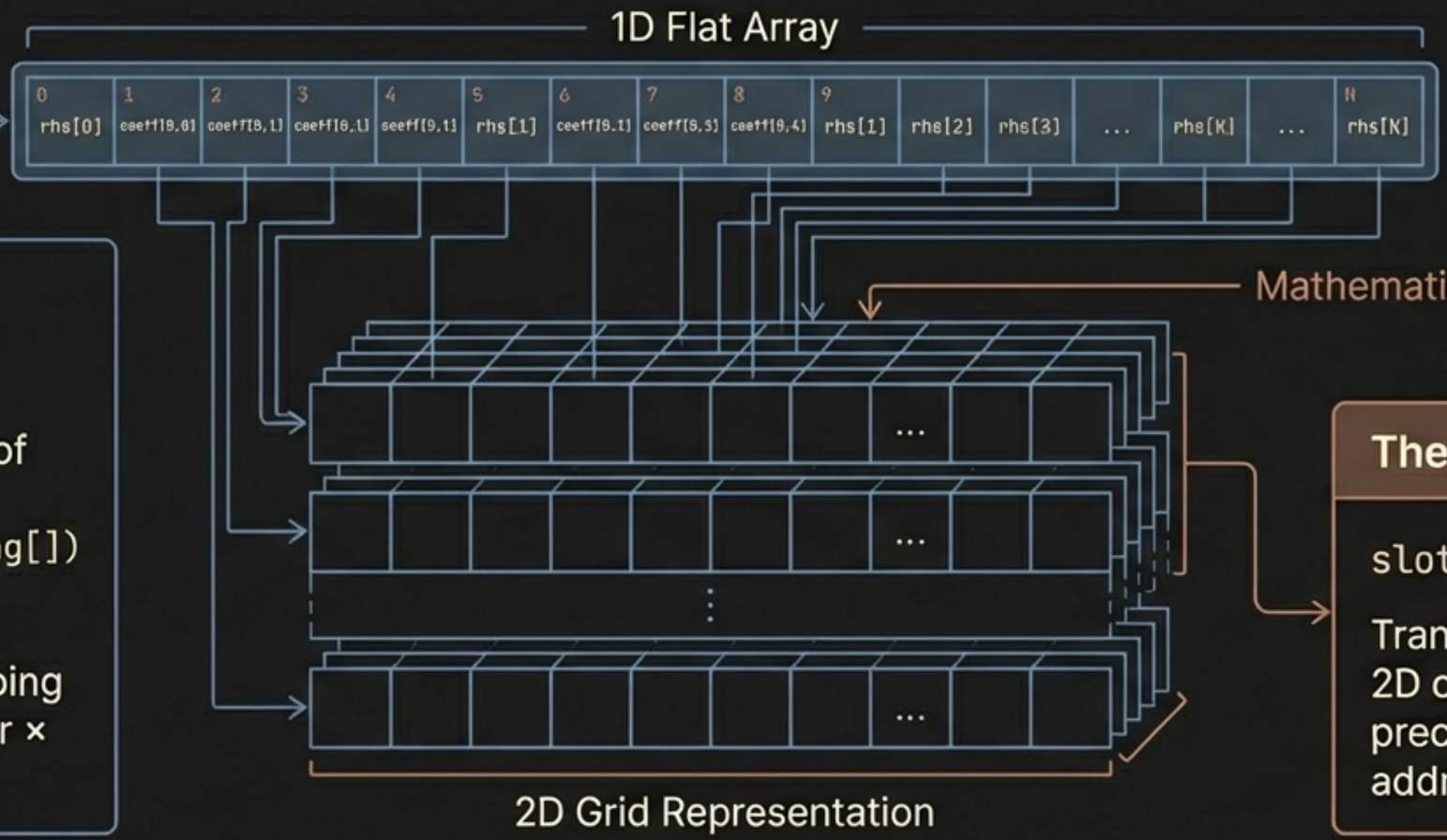
Matrix construction via `buildCoeffMatrix()` and recursive determinant evaluation.

4. Output Phase

Converting stored flat-array data back into human-readable strings via `equationToString()`.

Memory Management via Flat Arrays

Without standard vectors, dynamic equations are managed using 1D flat arrays acting as 2D structures.



The Blueprint:

- `rightHandSide (float[])`: Stores the RHS of each equation.
- `varNames (string[])` & `varCoeffs (float[])`: Flat arrays mapping equation number × variable slot.

The Key to Indexing:

`slot(eq, varSlot)`

Translates simulated 2D coordinates into precise 1D memory addresses.

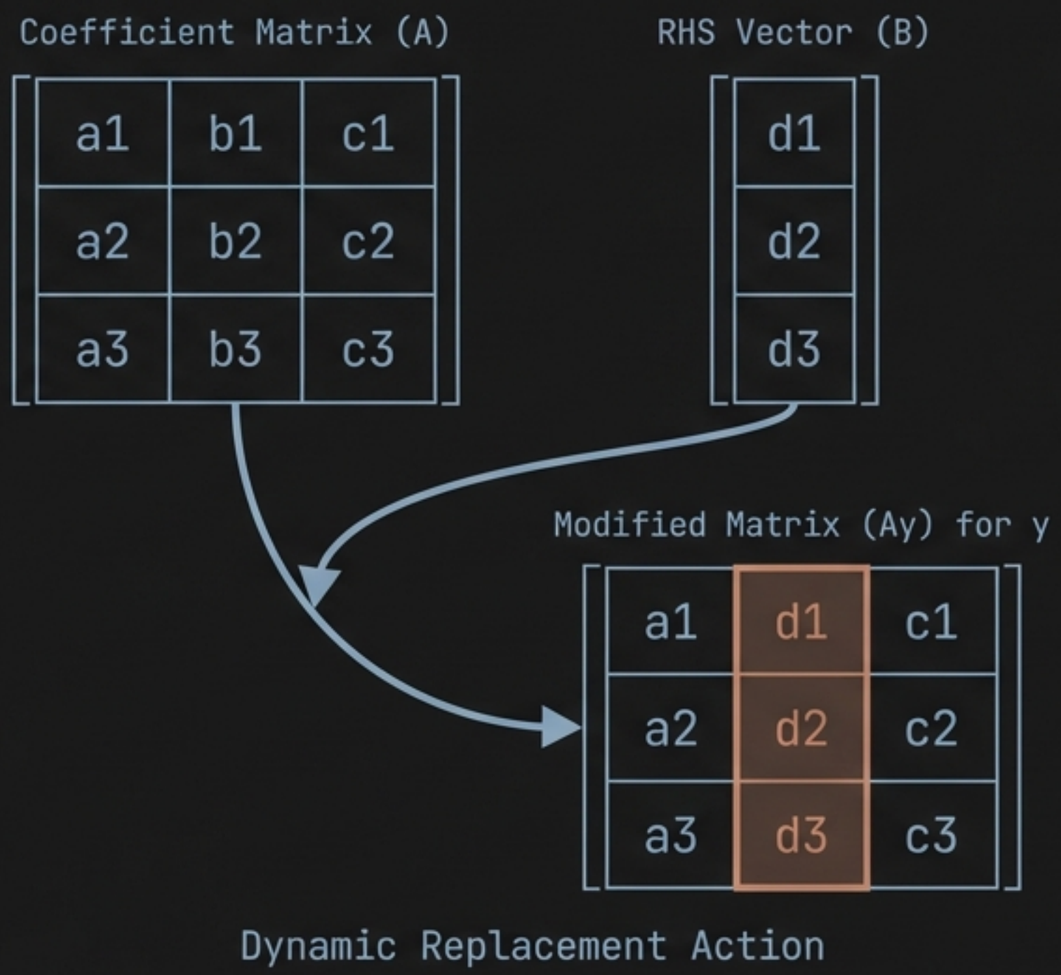
The Matrix Engine and Cramer’s Rule

Laplace Expansion

Determinants are calculated recursively using cofactor expansion. This yields O(n!) time complexity, optimized for systems where n ≤ 5.

Cramer’s Implementation

Dynamically replaces the target variable's column in the coefficient matrix with the RHS vector. Requires executing n+1 total determinant calculations to output final states.



```
Validating 2-Variable Systems

System Input:
  equation 1: 2x+y=5
  equation 2: x-y=1

Arithmetic Processing:
> add 1 2
    3x=6
> subtract 1 2
    x+2y=4

Solver Output:
> D_value
    -3
> solve
    x=2
    y=1
```

Scaling to 3-Variable Matrices

Validating 3-Variable Systems

System Input:
equation 1: x+y+z=6
equation 2: 2x-y+z=3
equation 3: x+2y-z=2

Determinant Processing:
Engine constructs 3x3 matrix.
> D_value
-9

Solver Output:
> solve
x=1
y=2
z=3

Advanced Substitution:

Applying substitute z 1 2 proves the engine can dynamically eliminate z from the first equation using the second.

Resolving Edge Cases and Inconsistencies

Implicit Parsing

Flawlessly reads missing explicit coefficients, treating inputs like +x securely as +1x.

Sparse Systems

Handles variables appearing in only specific equations by treating missing slots securely as 0 in memory.

The ‘No Solution’ Trap

Parallel Inputs:
`x+y=1 and x+y=2`

`JetBrains Mono:`
`Engine computes`
`D_value = 0`

Elegantly halts to output **No Solution** rather than crashing with a fatal division by zero error.

Project Outcomes and Technical Mastery

A highly capable mathematical tool built entirely from scratch using raw C++ memory management and custom algorithms.

Functional Toolkit

Successfully delivered a complete, functional linear equations toolkit natively in C++.

Secure Routing

Proved complex dynamic data routing and string parsing can be achieved securely without modern STL safety nets.

Portable Binary

Engineered a dependency-free, highly portable binary capable of performing reliable linear algebra computations across any OS.

Future Horizons

Gaussian Elimination

Shifting algorithms to process complex systems far exceeding n=5 without the O(n!) mathematical bottleneck.

Infinite Solution States

Expanding the determinant logic to distinguish between inconsistent systems (No Solution) and under-determined systems (Infinite Solutions).

Interface Evolution

Integrating the raw, high-performance core logic into intuitive graphical or web-based frontend interfaces.