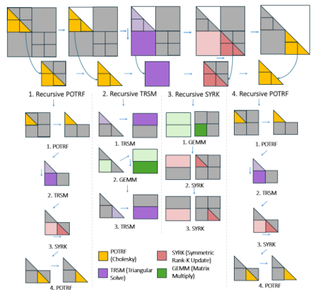
Vicki Carrica (author), Rabab Alomairy (advisor), Pedro Valero-Lara (advisor), Alan Edelman (advisor)

AGENTIC AI AUGMENTATION FOR RECURSIVE MIXED-PRECISION LINEAR SOLVERS

RECURSIVE MIXED-PRECISION ENABLES FAST MXU LINEAR SOLVERS

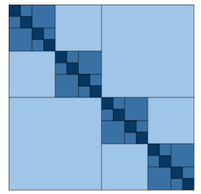


Recursive Cholesky Decomposition Tree

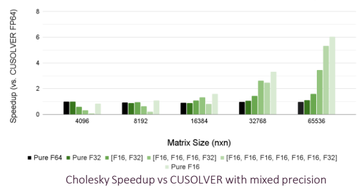
The Problem: Low precision computation on modern hardware sacrifices accuracy for speed. Linear solvers are essential for scientific computing, where we need numerical accuracy.

Our Cholesky Solution: Nested recursive framework with hierarchical precision assignment using a custom data structure. ~ 5× speedup over CUSOLVER FP32 while being 100x more accurate than pure FP16 for Cholesky.

The Goal: Extending this framework to other linear solvers like LU and QR.



Recursive Mixed-Precision Data Structure



Cholesky Speedup vs CUSOLVER with mixed precision

EXTENDING RECURSIVE MIXED PRECISION SOLVERS IS CHALLENGING. HOW CAN LLMS HELP?

Translating our mixed-precision framework from Cholesky to LU requires a **new data structure, mixed-precision recursive GEMM**, and following casting and quantization rules.

The Question: Can we use LLMs to generate high-performance LU?

LU Implementation Evaluation Checklist

	Recursive Algorithm
☐	Checks `A.BaseCase !== nothing` or `n <= block_size` before recursing
☐	Calls `lu_recursive!` on `A11` before triangular solves
☐	Computes `U12` via `unified_rectrxm!`, not vendor fallback
☐	Uses correct `U12` arguments: `('L','L','U')`
☐	Computes `L21` via `unified_rectrxm!`, not vendor fallback
☐	Uses correct `L21` arguments: `('R','U','N')`
☐	Updates Schur complement via `recgemm!`
☐	Schur operation: `A22 = A22 − L21 * U12` (negative multiplier)
☐	Calls `lu_recursive!` on `A22` after Schur update
	Data Structure & Framework
☐	Accesses `FullMixedPrec` fields (`.A11`, `.A12`, `.A21`, `.A22`, `.BaseCase`); no slicing
☐	Separate dispatch methods for `FullMixedPrec` and dense arrays
☐	Vendor `trsm!`/`gemm!` only in recursive helpers/base-case, not main LU
☐	Uses allocation-free views: `@view` or `view(...)` for partitions
	Mixed-Precision & GPU
☐	Upcasts Float16 to Float32 before `CUSOLVER.getrf!`
☐	No scalar indexing on GPU arrays (no element-wise loops)

ICL METHODOLOGY

In Context Learning (ICL): injecting task-specific context directly into the prompt.

Five ICL prompt levels, from a **minimal baseline prompt** (L1) to **varying levels of provided code (context) and mathematical formulation**.

We evaluated the generated code with a **15-item LU checklist, numerical accuracy, and runtime performance**.

Math \ Context	Restricted	Full
Vague	Level 2	Level 4
Explicit	Level 3	Level 5

FULL CONTEXT IS NECESSARY FOR RELIABILITY AND CODE QUALITY

Restricting framework context (L1-L3) frequently resulted in **execution failures** and **lower implementation quality**. Providing full context (L4, L5) led to **100% execution success** across all four LLMs and **higher checklist scores**. Successful implementations followed the nested recursive framework, enabling the fastest runtimes across models. This shows that **full context is essential for reliable AI-generated HPC code**.

Model Execution Success Across Prompting Levels

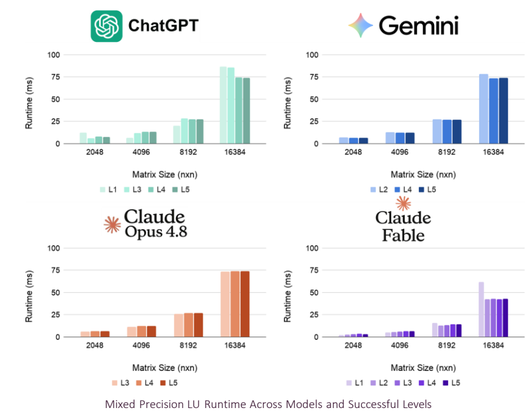
Model	L1	L2	L3	L4	L5
ChatGPT 5.3 Codex	✓	×	✓	✓	✓
Gemini Pro 3.1	×	✓	×	✓	✓
Claude Opus 4.8	×	×	✓	✓	✓
Claude Fable	✓	✓	✓	✓	✓

Model Checklists out of 15

Model	L1	L2	L3	L4	L5
ChatGPT 5.3 Codex	8	14	14	14	14
Gemini Pro 3.1	8	10	10	10	10
Claude Opus 4.8	8	9	9	10	10
Claude Fable	8	11	11	11	**15**

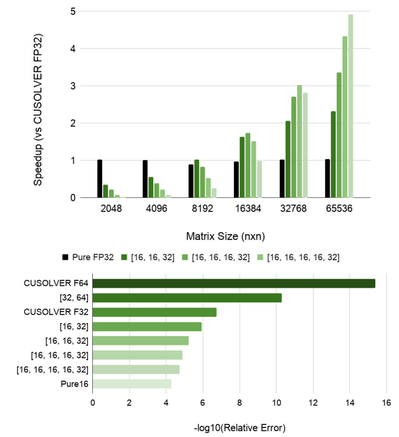
Nested Recursive LU Implementation Generated By the Model

Model	L1	L2	L3	L4	L5
ChatGPT 5.3 Codex	No	Full	Full	Full	Full
Gemini Pro 3.1	No	Partial	Partial	Partial	Partial
Claude Opus 4.8	No	Partial	Partial	Partial	Partial
Claude Fable	No	Partial	Partial	Partial	Full



Mixed Precision LU Runtime Across Models and Successful Levels

ICL ACHIEVES EXPERT-LEVEL SPEEDUPS VERSUS STATE-OF-THE-ART LU



Speedup vs Non-Pivoting CUSOLVER FP32 LU (Top) and Accuracy (Bottom) of Fable Level 5 Nested-Recursive Mixed-Precision LU

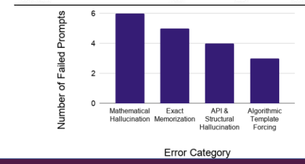
Claude Fable's Level 5 generated code achieved a **4.9x speedup** over the vendor-optimized cuSOLVER FP32 baseline at n=65536 while maintaining expected accuracy.

The nested recursive LU formulation exposes **GEMM operations on the off-diagonal blocks**. By computing these in low precision, the AI-generated code successfully leverages **NVIDIA H200 Tensor Cores** while maintaining expected numerical accuracy.

FINE-TUNING LEARNS EXISTING FRAMEWORKS BUT DOES NOT GENERALIZE

Fine-Tuned StarCoder2-15B Accuracy Across Prompt Categories

Evaluation Category	Total Prompts	Successful	Accuracy (%)
In-Distribution (Exact Match)	10	10	100%
In-Distribution (Paraphrased)	10	10	100%
OOD (Code Synthesis)	10	0	0%
OOD (Conceptual Reasoning)	10	2	20%
Overall	**40**	**22**	**55%**

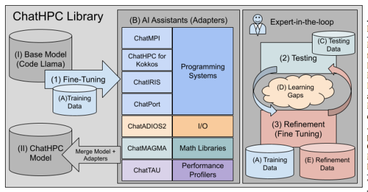


Fine-tuning on StarCoder2-15B **increased task accuracy from a baseline of 0% (0/40) to 55% (22/40)** overall.

The model **memorized the in-distribution framework** (100% success), but **failed to generalize** to Out-of-Distribution (OOD).

The Trade-off: Fine-tuning provides fast, token-efficient inference using a smaller model (15B) but defaults to **math hallucinations and exact memorization** on novel tasks. In contrast, ICL **requires massive token context windows and more expensive** state-of-the-art models, but **successfully generalizes to new algorithms** like the LU implementation when prompted with full context.

FINE-TUNING METHODOLOGY



Summary of Fine-Tuning Dataset by Module

Dataset Module	Prompt Count
Mixed-precision block-recursive linear algebra framework	68
Recursive mixed-precision data structures	60
NVIDIA hardware wrappers	52
Hardware routing and type dispatch for GPU linear algebra	28
Mathematical formulations for block-recursive linear algebra	28
Algorithm dependencies and data structure mapping	24
Quantization rules and overflow prevention	16
Mathematical limitations and non-recursive formulations	4
Total	**280**

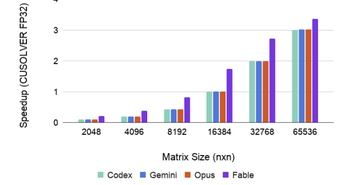
We advanced the **ChatHPC ecosystem** by upgrading its base model to **StarCoder2-15B** (selected for its Julia pretraining), training lightweight adapters without modifying base weights.

We created a **custom dataset** of 280 prompt-context-answer tuples across eight core modules.

To ensure generalization to diverse inputs, we **generated five stylistic variations of each initial prompt**.

ChatHPC QR Code

ALL MODELS SURPASS STATE-OF-THE-ART AT SCALE WITH LEVEL 5 PROMPTING



Speedup vs Non-Pivoting CUSOLVER FP32 LU Across All Level 5 Implementations

Using the [16, 16, 32] mixed-precision hierarchy, all four models successfully **pass accuracy** validation when prompted at Level 5, **matching or exceeding the unpivoted CUSOLVER FP32** baseline at large matrix sizes, **n ≥ 16384**.

FUTURE WORK

Extending the recursive mixed-precision framework and ICL pipeline to QR.

Exploring lower precision types like FP8 and BF16.

Extending the mixed-precision recursive framework to sparse matrices.

ACKNOWLEDGEMENTS AND REFERENCES

This material is in part based upon work supported by the US Department of Energy, Office of Science, Advanced Scientific Computing Research program as part of the Advancements in Artificial Intelligence for Science program, Ellora and Durban projects.

P. Valero-Lara et al., "ChatHPC: Building the Foundations for a Productive and Trustworthy AI-Assisted HPC Ecosystem," SC '25, St. Louis, MO, 2025, pp. 458–474.
V. Carrica, R. Alomairy, E. Ringoot, and A. Edelman, "Hierarchical Recursive Precision for Accelerating Symmetric Linear Solves on MXUs," arXiv:2601.08082, 2026.

Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, endorses, recommend or favors, or assumes any legal liability for anything in this report, or represents that its use would not infringe privately owned rights. The views expressed herein are those of the authors alone.