JHPCN Field Workshop
State-of-the-Art in Code Generative AI for High-Performance Computing
名古屋大学, 2025年12月5日

AI & HPC Synergies
Developments and Opportunities

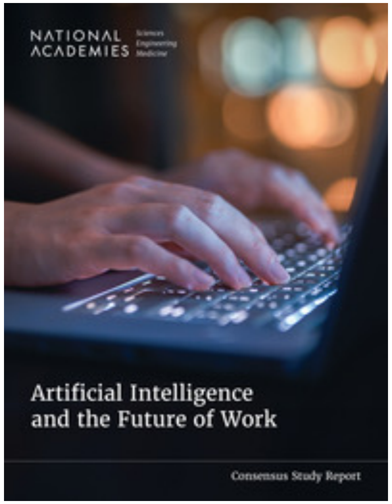
Osni Marques
Lawrence Berkeley National Laboratory
(retiree affiliate)

Contents

- Introduction
  - General directions
  - AI in CS&E / HPC
- Synergies
  - Focus on topics I have worked on
  - What could (have) be(en) done better ?
- Summary / conclusions

The Future of the Workforce in the Age of AI

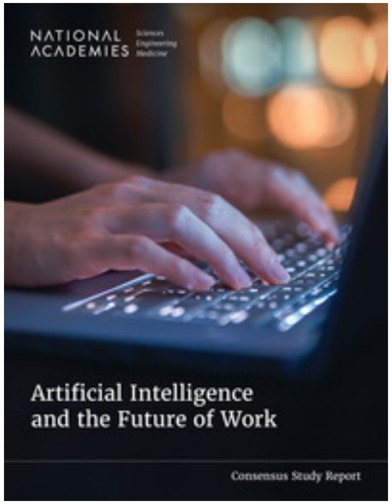
- Opportunities: AI redefining tasks, roles, and productivity
  - Up to 80% of jobs will see some tasks affected by AI.
  - Routine, text-heavy, and analytical tasks are most exposed.
  - AI drives augmentation over automation, enhancing human performance.
  - New hybrid workflows: AI assists, humans curate and decide.
  - Rising demand for AI literacy, adaptability, creativity, and ethics.
- Challenges: adapting for an AI-augmented future
  - Organizations: must redesign jobs, upskill workers, and integrate AI responsibly.
  - Workers: lifelong learning and hybrid skills are essential to stay competitive.
  - Policy & Education: focus on inclusion, critical thinking, and fair transition support.
  - By 2030 most jobs will evolve rather than disappear: new “human + AI” roles dominate.
  - Success depends on collaboration, not competition, between humans and AI.


NATIONAL ACADEMIES
Sciences Engineering Medicine
Artificial Intelligence and the Future of Work
Consensus Study Report

- *Brynjolfsson, Chandar and Chen (2025). Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence. Stanford Digital Economy Lab.*
- *Bankins, Hu and Yuan (2024). Artificial intelligence, workers, and future of work skills. Current Opinion in Psychology*
- *Zirar, Ali and Islam (2023). Worker and workplace AI coexistence: emerging themes and research agenda. Technovation.*

The Future of the Workforce in the Age of AI

- Opportunities: AI redefining tasks, roles, and productivity
  - Up to 80% of jobs will see some tasks affected by AI.
  - Routine, text-heavy, and analytical tasks are most exposed.
  - AI drives augmentation over automation, enhancing human performance.
  - New hybrid workflows: AI assists, humans curate and decide.
  - Rising demand for AI literacy, adaptability, creativity, and ethics.
- Challenges: adapting for an AI-augmented future
  - Organizations: must redesign jobs, upskill workers, and integrate AI responsibly.
  - Workers: lifelong learning and hybrid skills are essential to stay competitive.
  - Policy & Education: focus on inclusion, critical thinking, and fair transition support.
  - By 2030 most jobs will evolve rather than disappear: new “human + AI” roles dominate.
  - Success depends on collaboration, not competition, between humans and AI.



NATIONAL ACADEMIES

Artificial Intelligence and the Future of Work

Consensus Study Report

- *Brynjolfsson, Chandar and Chen (2025). Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence. Stanford Digital Economy Lab.*
- *Bankins, Hu and Yuan (2024). Artificial intelligence, workers, and future of work skills. Current Opinion in Psychology*
- *Zirar, Ali and Islam (2023). Worker and workplace AI coexistence: emerging themes and research agenda. Technovation.*

Eric Schmidt, https://youtu.be/L5jhEYofpaQ: “Within the next 12 months, most programming jobs could be replaced by AI… AI systems will be able to surpass the brightest graduate-level mathematicians in tasks like advanced math and coding.” (04/10/2025)

Large Language Models (LLMs) abound

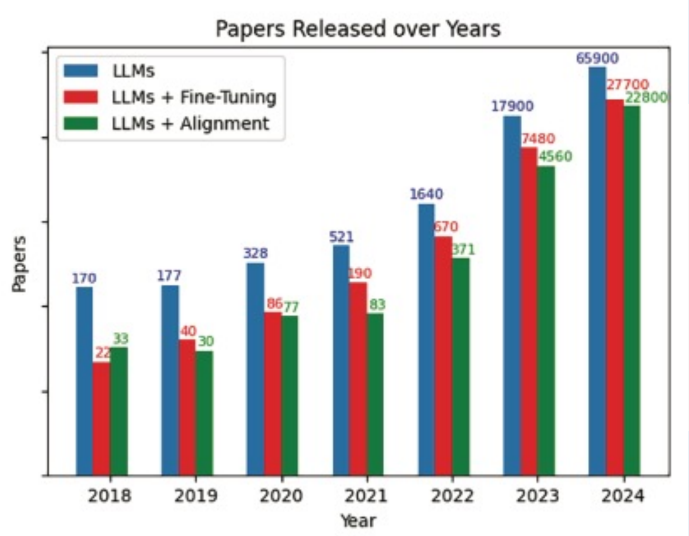
- GPT-4 (OpenAI)
- GPT-3 (OpenAI)
- Gemini (Google DeepMind)
- Claude 3 (Anthropic)
- Llama 3 (Meta AI)
- Falcon 180B (Technology Innovation Institute)
- BLOOM (BigScience)
- PaLM 2 (Google)
- Ernie 4.5 (Baidu)
- Qwen 3 (Alibaba Cloud)

⋮

Large Language Models (LLMs) abound

- GPT-4 (OpenAI)
- GPT-3 (OpenAI)
- Gemini (Google DeepMind)
- Claude 3 (Anthropic)
- Llama 3 (Meta AI)
- Falcon 180B (Technology Innovation Institute)
- BLOOM (BigScience)
- PaLM 2 (Google)
- Ernie 4.5 (Baidu)
- Qwen 3 (Alibaba Cloud)

⋮



Naveed et al. (2025). A Comprehensive Overview of Large Language Models. ACM Transactions on Intelligent Systems and Technology.

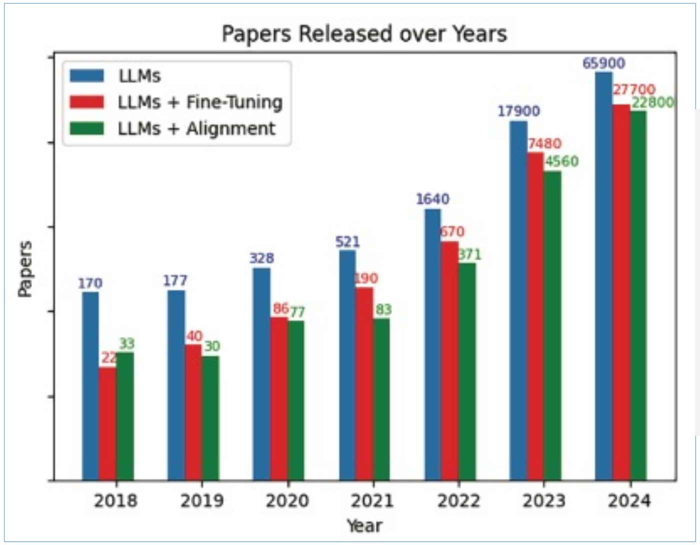
Large Language Models (LLMs) abound

- GPT-4 (OpenAI)
- GPT-3 (OpenAI)
- Gemini (Google DeepMind)
- Claude 3 (Anthropic)
- Llama 3 (Meta AI)
- Falcon 180B (Technology Innovation Institute)
- BLOOM (BigScience)
- PaLM 2 (Google)
- Ernie 4.5 (Baidu)
- Qwen 3 (Alibaba Cloud)

⋮

- Customer support & chatbots
- Content creation & copywriting
- Language translation & localization
- Sentiment analysis & classification
- Legal & contract review
- Supply chain, operations & decision-making support
- Personalized education & tutoring
- Document summarization & knowledge extraction
- Code generation & software development support
- Research assistance & scientific applications

⋮



Naveed et al. (2025). A Comprehensive Overview of Large Language Models. ACM Transactions on Intelligent Systems and Technology.

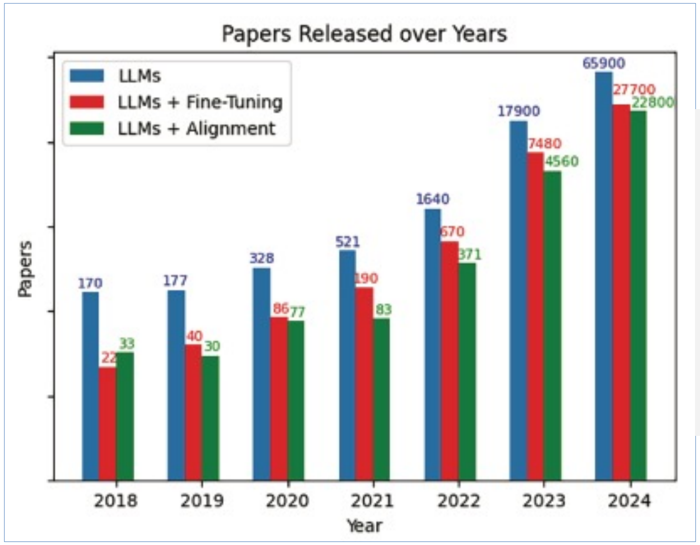
Large Language Models (LLMs) abound

- GPT-4 (OpenAI)
- GPT-3 (OpenAI)
- Gemini (Google DeepMind)
- Claude 3 (Anthropic)
- Llama 3 (Meta AI)
- Falcon 180B (Technology Innovation Institute)
- BLOOM (BigScience)
- PaLM 2 (Google)
- Ernie 4.5 (Baidu)
- Qwen 3 (Alibaba Cloud)

⋮

- Customer support & chatbots
- Content creation & copywriting
- Language translation & localization
- Sentiment analysis & classification
- Legal & contract review
- Supply chain, operations & decision-making support
- Personalized education & tutoring
- Document summarization & knowledge extraction
- Code generation & software development support
- Research assistance & scientific applications

⋮



Naveed et al. (2025). A Comprehensive Overview of Large Language Models. ACM Transactions on Intelligent Systems and Technology.

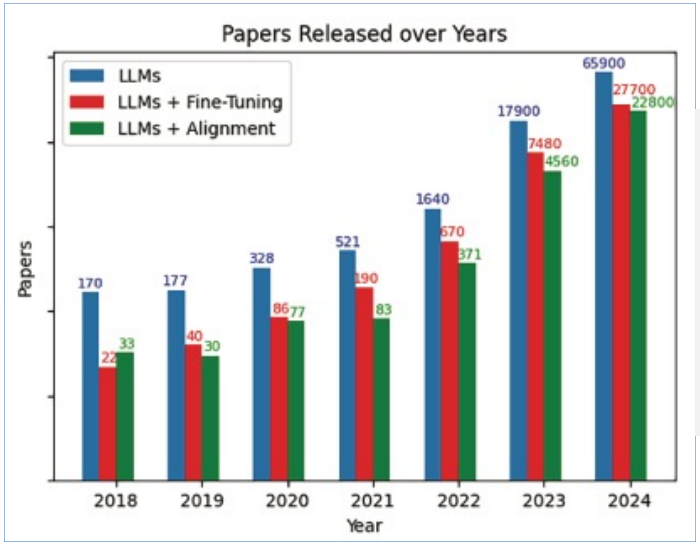
Large Language Models (LLMs) abound

- GPT-4 (OpenAI)
- GPT-3 (OpenAI)
- Gemini (Google DeepMind)
- Claude 3 (Anthropic)
- Llama 3 (Meta AI)
- Falcon 180B (Technology Innovation Institute)
- BLOOM (BigScience)
- PaLM 2 (Google)
- Ernie 4.5 (Baidu)
- Qwen 3 (Alibaba Cloud)

⋮

- Customer support & chatbots
- Content creation & copywriting
- Language translation & localization
- Sentiment analysis & classification
- Legal & contract review
- Supply chain, operations & decision-making support
- Personalized education & tutoring
- Document summarization & knowledge extraction
- Code generation & software development support
- Research assistance & scientific applications

⋮



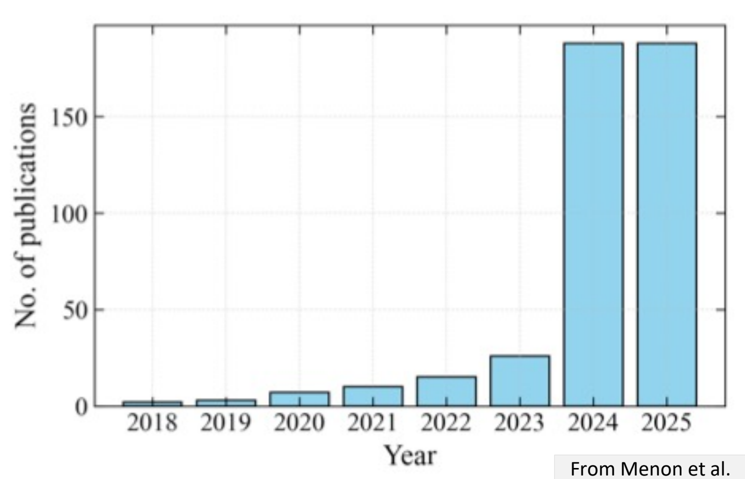
Naveed et al. (2025). A Comprehensive Overview of Large Language Models. ACM Transactions on Intelligent Systems and Technology.

NYT, Oct 5, 2025

これは和牛ではありません。

From LLMs to Foundation Models for Science

- Foundation models are large-scale AI systems trained on broad and diverse datasets, designed to serve as a base for multiple downstream applications
  - Characteristics:
    - Trained on massive, multimodal data (text, images, audio, code, etc.)
    - Capable of transfer learning across diverse domains
    - Enable rapid development of customized AI tools and applications
  - Relevance:
    - Drive advances in automation, research, and human–AI collaboration
    - Serve as the infrastructure layer of AI ecosystems
    - Lower barriers for innovation across domains
- AuroraGPT
  - improve methodologies for the science community
  - pre-trained, instruct-tuned and aligned models
  - collecting and refining large-scale scientific datasets (models at 8×10^9 to 400×10^9 parameters)
  - DOE's Frontiers in Artificial Intelligence for Science, Security and Technology (FASST)
- SciGPT
  - https://www.stat.berkeley.edu/~mmahoney
  - anchored on scientific principles and ML principles
  - inclusion of scientific computing concepts
  - implementations?
  - applications?



From Menon et al.

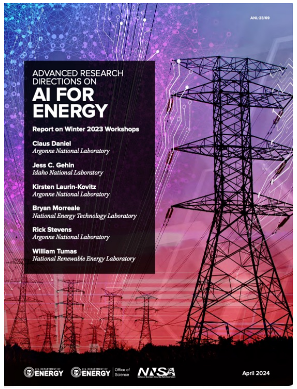
- *Bommasani et al. (2022). On the Opportunities and Risks of Foundation Models. arXiv:2108.07258.*
- *Menon et al. (2025). On Scientific Foundation Models: Rigorous Definitions, Key Applications, and a Survey. https://dx.doi.org/10.2139/ssrn.5409063.*
- *Choi et al. (2025). Defining Foundation Models for Computational Science: A Call for Clarity and Rigor. arXiv:2505.22904.*

AI-driven tools and applications in CS&E

- Chemistry and Materials Science
  - prediction of molecular and material properties
  - self-driving laboratories
  - ChemCrow (LLM chemistry agent)
- Drug discovery
  - analyzes of vast amounts of data
  - identification of risks factors
  - prediction of patient outcomes
- Vaccine development
  - simulation of the immune system
  - creation of predictive models
- Network operations
  - predictive analytics
  - automated management
  - enhanced security
- Advanced data analysis
  - automation of tasks
  - generation of predictive insights
- Predictive maintenance
- Design optimization
- Fault diagnosis

⋮

2024



ANL-23/69

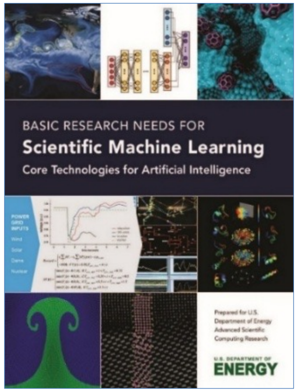
ADVANCED RESEARCH DIRECTIONS ON
AI FOR ENERGY
Report on Winter 2023 Workshops

Claus Daniel
Argonne National Laboratory
Jess C. Gehin
Idaho National Laboratory
Kirsten Laurin-Kovitz
Argonne National Laboratory
Bryan Morreale
National Energy Technology Laboratory
Rick Stevens
Argonne National Laboratory
William Tumas
National Renewable Energy Laboratory

U.S. DEPARTMENT OF ENERGY
U.S. DEPARTMENT OF ENERGY | Office of Science
NNSA
April 2024

- ❖ Nuclear energy
- ❖ Power grid
- ❖ Carbon management
- ❖ Energy storage
- ❖ Energy materials

2019



BASIC RESEARCH NEEDS FOR
Scientific Machine Learning
Core Technologies for Artificial Intelligence

Prepared for U.S. Department of Energy Advanced Scientific Computing Research

U.S. DEPARTMENT OF ENERGY

- ❖ Domain-aware scientific ML
- ❖ Interpretable scientific ML
- ❖ Robust scientific ML
- ❖ Data-intensive scientific ML
- ❖ ML-enhanced modeling and simulation
- ❖ Automation and decision support

Retrospective on Exascale Panel @ ECPAM 2022

Jeffrey Vetter
Pete Beckman
Jack Dongarra
Bob Lucas
Kathy Yelick

- Misses:
  - ❖ *Impact of ML/AI on applications and systems*
  - ❖ *AI does not appear in the 2010 Exascale Report*
- Predictions for 2032:
  - ❖ *Scientists will use AI/ML*
  - ❖ *AI-based automated code writing, testing and improvement*
  - ❖ *Simulation apps will use AI hardware*

White House's Genesis Mission: *dedicated, coordinated national effort to unleash a new age of AI-accelerated innovation and discovery that can solve the most challenging problems of this century.*

The Japan News: *About ¥190 billion will be earmarked for AI-related initiatives, with ¥45 billion allocated for the application of AI to scientific research.*

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

- **20+ public repositories in *ai-coding-assistant***
  - awesome-vibe-coding
  - awesome-ai-coding-tools
  - clauder

⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

- **20+ public repositories in *ai-coding-assistant***
  - awesome-vibe-coding
  - awesome-ai-coding-tools
  - clauder

  ⋮
- **30+ public repositories in *ai-code-generation***
  - fragments
  - awesome-ai-coding-tools
  - The-Creator-AI

  ⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

- **20+ public repositories in *ai-coding-assistant***
  - awesome-vibe-coding
  - awesome-ai-coding-tools
  - clauder

    ⋮
- **30+ public repositories in *ai-code-generation***
  - fragments
  - awesome-ai-coding-tools
  - The-Creator-AI

    ⋮
- **35+ public repositories in *ai-code-review***
  - typedai
  - kodus-ai
  - bitoai

    ⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

- **20+ public repositories in *ai-coding-assistant***
  - awesome-vibe-coding
  - awesome-ai-coding-tools
  - clauder

  ⋮
- **30+ public repositories in *ai-code-generation***
  - fragments
  - awesome-ai-coding-tools
  - The-Creator-AI

  ⋮
- **35+ public repositories in *ai-code-review***
  - typedai
  - kodus-ai
  - bitoai

  ⋮
- **45+ public repositories in *ai-testing***
  - giskard-oss
  - Scenario
  - langtest

  ⋮

AI-driven tools to support coding

https://www.qodo.ai

- **AI-Powered Development Assistants**
  - Qodo
  - Cursor AI
  - Codeium
  - AskCodi
- **Code Intelligence & Completion**
  - Github Copilot
  - Tabnine
  - IntelliCode
- **Security & Analysis**
  - DeepCode AI
  - Codiga
  - Amazon CodeWhisperer
- **Cross-Language & Translation**
  - CodeT5
  - Figstack
  - CodeGeeX
- **Educational & Learning Tools**
  - Replit
  - OpenAI Codex
  - SourceGraph Cody
  - StarCoder

⋮

- **20+ public repositories in *ai-coding-assistant***
  - awesome-vibe-coding
  - awesome-ai-coding-tools
  - clauder

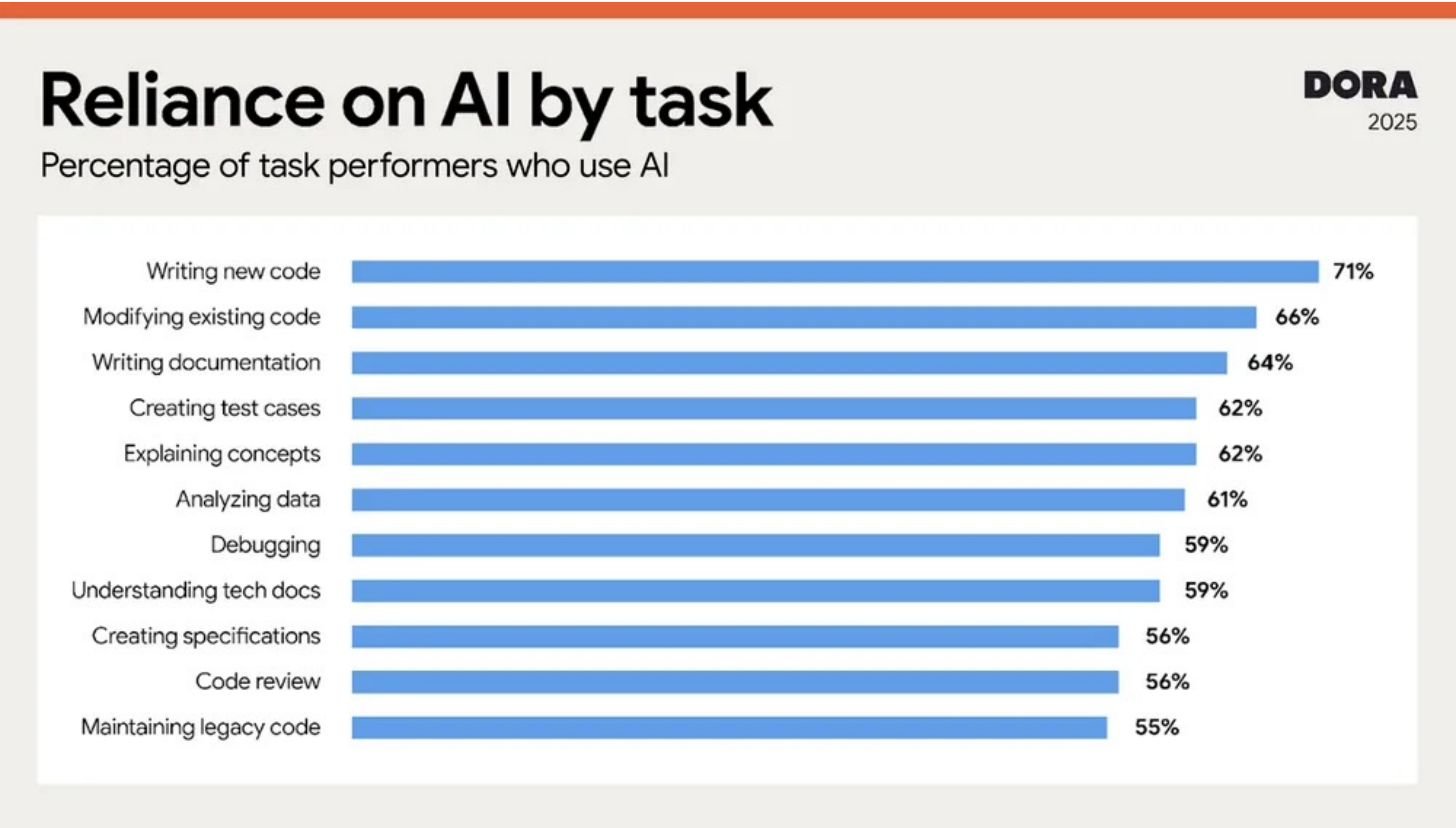
  ⋮
- **30+ public repositories in *ai-code-generation***
  - fragments
  - awesome-ai-coding-tools
  - The-Creator-AI

  ⋮
- **35+ public repositories in *ai-code-review***
  - typedai
  - kodus-ai
  - bitoai

  ⋮
- **45+ public repositories in *ai-testing***
  - giskard-oss
  - Scenario
  - langtest

  ⋮

- *Jiang et. (2024). A Survey on Large Language Models for Code Generation. arXiv:2406.00515.*
- *Peng et al. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590.*
- *Dakhel et al. (2023). GitHub Copilot AI pair programmer: Asset or Liability? Journal of Systems and Software.*
- *Li et al. (2022). Competition-level code generation with AlphaCode. arXiv:2203.07814.*



Reliance on AI by task

Percentage of task performers who use AI

Source: Google Cloud’s DORA (DevOps Research and Assessment)

AI & HPC Synergies – a partial list

- Code generation
- Code refactoring
- Code translation
- Developer productivity
- Automatic tuning
- Software testing
- Algorithm selection

⋮

- *Guo (2023). Six Opportunities for Scientists and Engineers to Learn Programming Using AI Tools Such as ChatGPT. IEEE CS&E.*
- *Ozkaya (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. IEEE CS&E.*
- *Haque (2025). LLMs: A Game-Changer for Software Engineers? arXiv:2411.00932.*
- *Ding et al. (2023). HPC-GPT: Integrating Large Language Model for High-Performance Computing, arXiv:2311.12833.*
- *Yin et al. (2025). chatHPC: Empowering HPC users with large language models. J Supercomputing.*
- *Teranishi et al. (2025). Leveraging AI for Productive and Trustworthy HPC Software: Challenges and Research Directions. arXiv:2505.08135.*
- *Research Software Engineering in the Age of AI, panel at SC25.*

AI & HPC Synergies – a partial list

- Code generation
- Code refactoring
- Code translation
- Developer productivity
- Automatic tuning
- Software testing
- Algorithm selection

⋮

What could (have) be(en) done better ?

- *Guo (2023). Six Opportunities for Scientists and Engineers to Learn Programming Using AI Tools Such as ChatGPT. IEEE CS&E.*
- *Ozkaya (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. IEEE CS&E.*
- *Haque (2025). LLMs: A Game-Changer for Software Engineers? arXiv:2411.00932.*
- *Ding et al. (2023). HPC-GPT: Integrating Large Language Model for High-Performance Computing, arXiv:2311.12833.*
- *Yin et al. (2025). chatHPC: Empowering HPC users with large language models. J Supercomputing.*
- *Teranishi et al. (2025). Leveraging AI for Productive and Trustworthy HPC Software: Challenges and Research Directions. arXiv:2505.08135.*
- *Research Software Engineering in the Age of AI, panel at SC25.*

Code generation/refactoring/translation

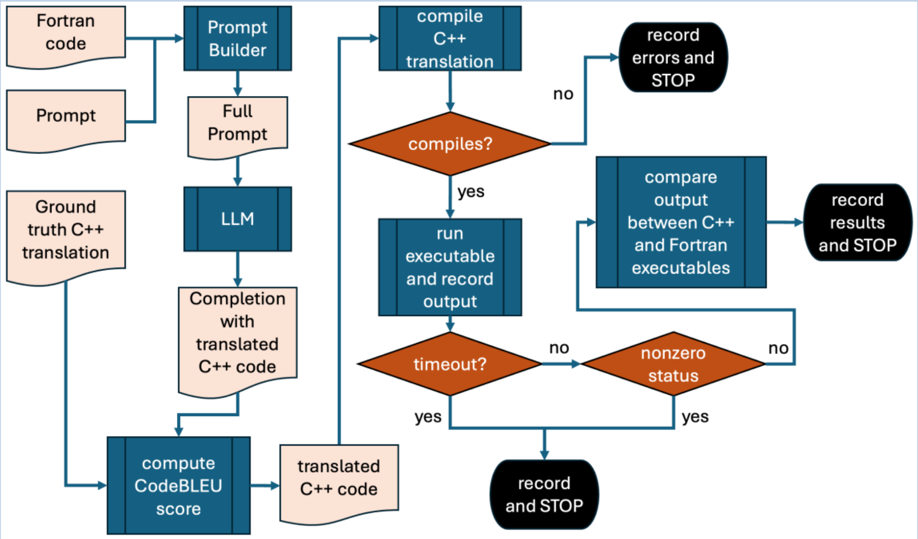
- Basic Linear Algebra Subprograms (BLAS) Generation
  - ChatBLAS
    - based on Julia
    - orchestrates with LLMs models
    - ability to target different architectures
  - Evaluation of BLAS 1, 2 and 3 generated with GPT-4.1 and o4-mini
    - "LLMs do not always learn the reference code as the best correct answer but that they learn the specifications of routines with documents available on the Internet."
- Code translation
  - FORTRAN to C++
    - See https://fortran-lang.discourse.group: "Scientists are using artificial intelligence and large language models to rewrite old code in modern languages"
  - OpenMP to CUDA, CUDA to OpenMP
- Parallelization
  - HPC-Coder
  - LASSI

- *Valero-Lara et al. (2024). ChatBLAS: The First AI-Generated and Portable BLAS Library. SC24.*
- *Mukunoki et al. (2025). Performance Evaluation of General-Purpose Large Language Models for Basic Linear Algebra Subprograms Code Generation. arXiv:2507.04697.*

- *Ranasinghe et al. (2025). LLM-Assisted Translation of Legacy FORTRAN Codes to C++: A Cross-Platform Study. arXiv:2504.15424.*
- *Dhruv and Dubey (2025). Leveraging Large Language Models for Code Translation and Software Development in Scientific Computing. arXiv:2410.24119.*
- *Chen et al. (2025). Fortran2CPP: Automating Fortran-to-C++ Translation using LLMs via Multi-Turn Dialogue and Dual-Agent Integration. arXiv:2412.19770.*
- *Bhattarai et al. (2024). Enhancing Code Translation in Language Models with Few-Shot Learning via Retrieval-Augmented Generation. arXiv:2407.19619.*
- *Nichols et al. (2024). HPC-Coder: Modeling Parallel Programs using Large Language Models, arXiv:2306.17281.*
- *Dearing et al. (2024). LASSI: An LLM-Based Automated Self-Correcting Pipeline for Translating Parallel Scientific Codes. arXiv:2407.01638.*

Code development/refactoring/translation

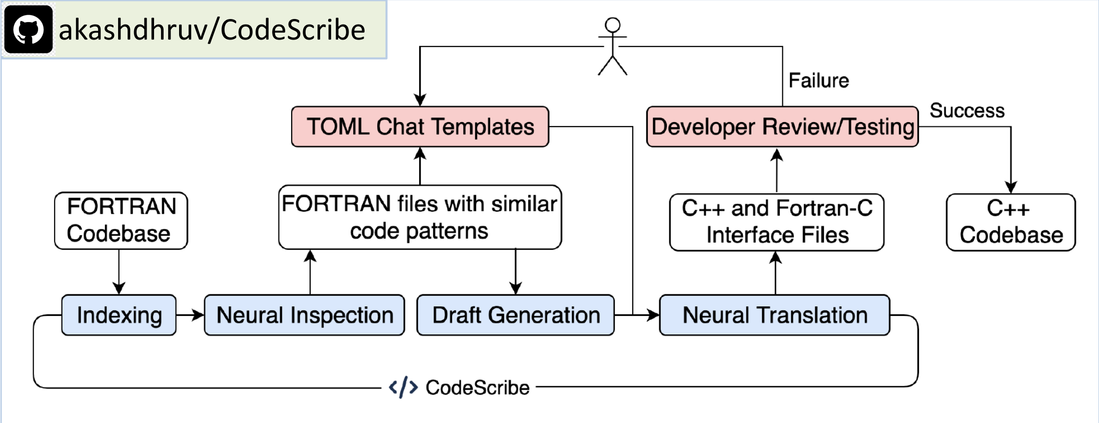
Ranasinghe et al.'s workflow



CodeBLEU: metric for evaluating code generation, assessing code quality by evaluating both syntax and semantic logic.

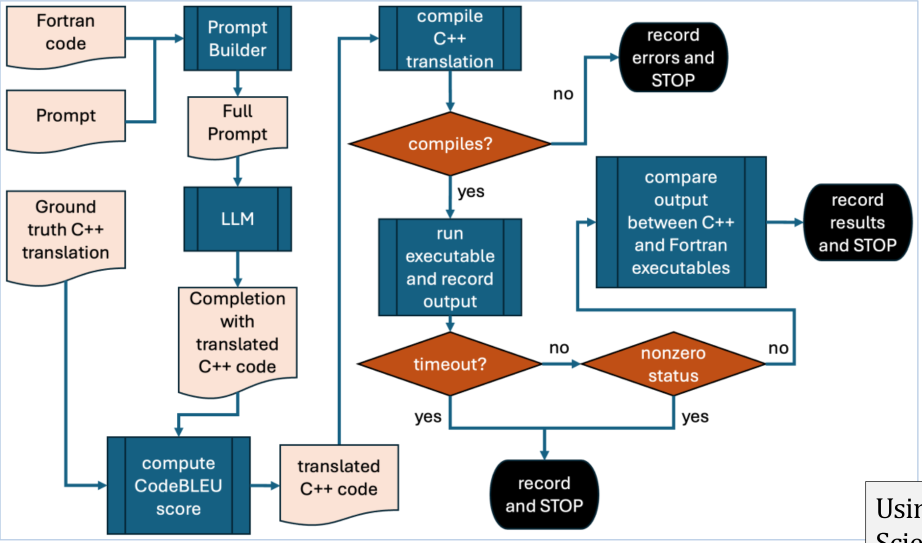
Dhruv and Dubey's CodeScribe

akashdhruv/CodeScribe



Code development/refactoring/translation

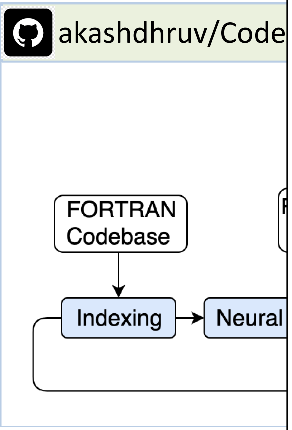
Ranasinghe et al.'s workflow

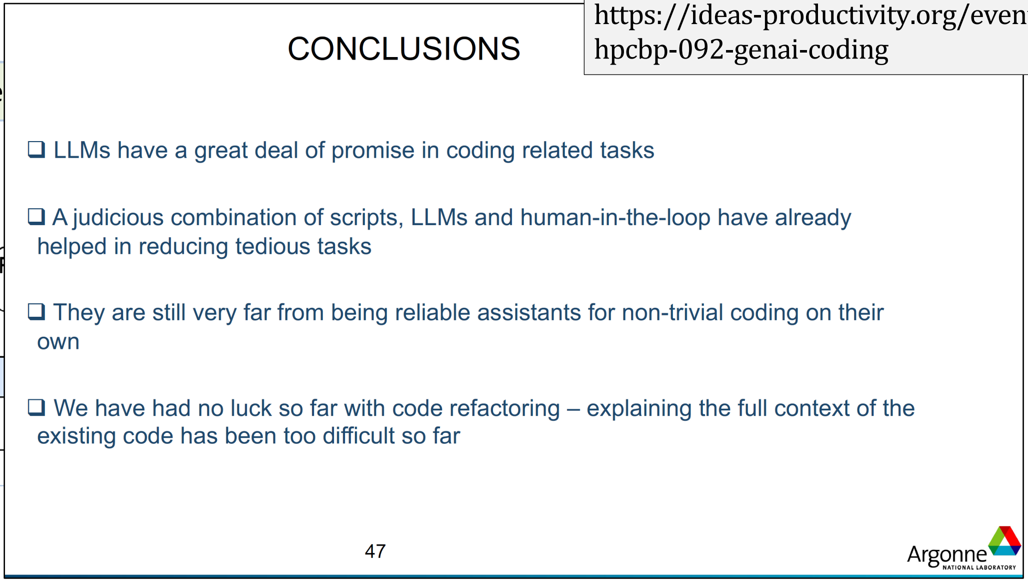


CodeBLEU: metric for evaluating code generation, assessing code quality by evaluating both syntax and semantic logic.

Using Generative AI for Coding Tasks in Scientific Software, 07/09/2025 webinar: https://ideas-productivity.org/events/hpcbp-092-genai-coding

Dhruv and Dubey's CodeScribe





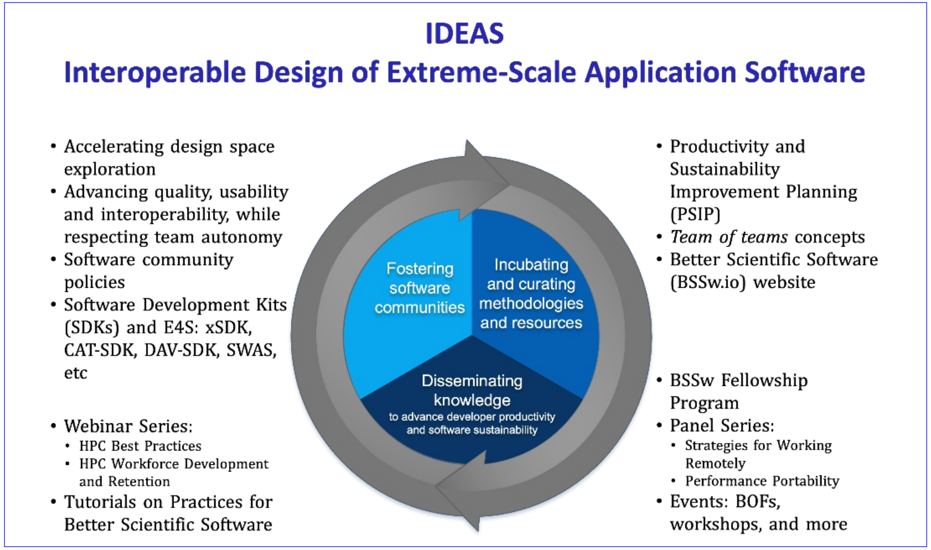
CONCLUSIONS

- ❑ LLMs have a great deal of promise in coding related tasks
- ❑ A judicious combination of scripts, LLMs and human-in-the-loop have already helped in reducing tedious tasks
- ❑ They are still very far from being reliable assistants for non-trivial coding on their own
- ❑ We have had no luck so far with code refactoring – explaining the full context of the existing code has been too difficult so far

47

Developer Productivity: The IDEAS Project

- ❖ Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.
- ❖ Respond to trend of continuous refactoring with efficient agile software engineering methodologies & improved software design.
- ❖ IDEAS began in 2014 as a DOE ASRC/BER partnership to improve application software productivity, quality, and sustainability, and continued as part of DOE's ECP.



IDEAS

Interoperable Design of Extreme-Scale Application Software

- Accelerating design space exploration
- Advancing quality, usability and interoperability, while respecting team autonomy
- Software community policies
- Software Development Kits (SDKs) and E4S: xSDK, CAT-SDK, DAV-SDK, SWAS, etc

- Webinar Series:
  - HPC Best Practices
  - HPC Workforce Development and Retention
- Tutorials on Practices for Better Scientific Software

Fostering software communities
Incubating and curating methodologies and resources
Disseminating knowledge
to advance developer productivity and software sustainability

- Productivity and Sustainability Improvement Planning (PSIP)
- *Team of teams* concepts
- Better Scientific Software (BSSw.io) website

- BSSw Fellowship Program
- Panel Series:
  - Strategies for Working Remotely
  - Performance Portability
- Events: BOFs, workshops, and more

Motto: *Software Quality Drives Quality Science*

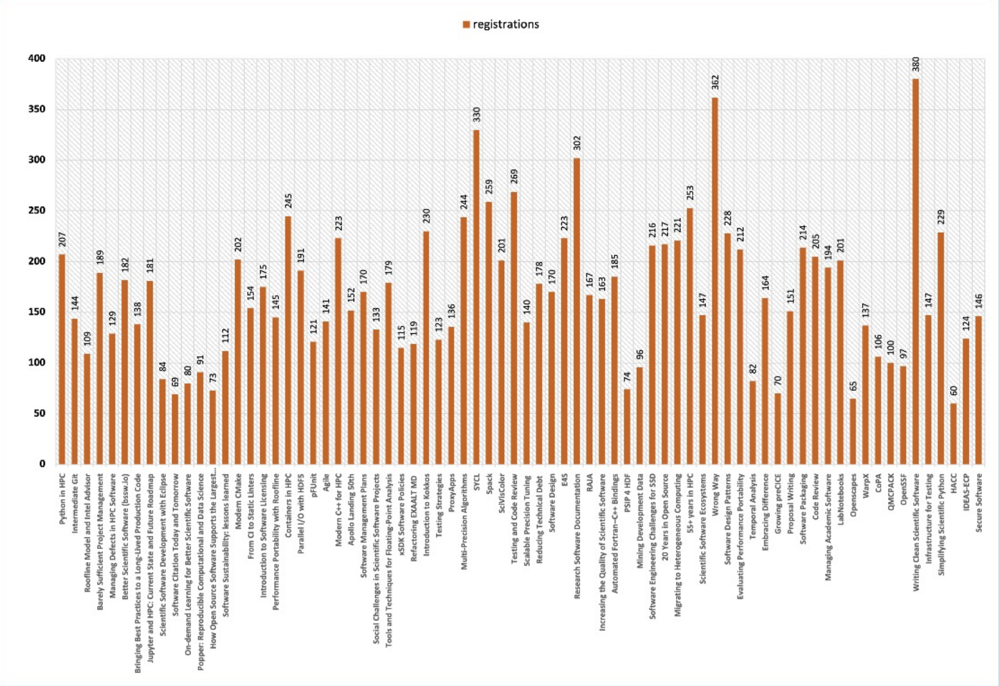
- Do you develop and use HPC software?
  - Investigate resources for software improvement
  - Advocate for and lead change in your projects
  - Disseminate insights about software improvement from your own work (blogs, presentations, posters, papers, etc.)
  - Check out community activities, such as the Research Software Engineering (RSE) movement
- Do you lead projects or organizations where teams develop and use HPC software?
  - Encourage continual software quality improvement
  - Provide clear career paths and mentoring for scientific software professionals, such as research software engineers
- Are you a stakeholder or supporter of projects that develop and use HPC software?
  - Incorporate expectations of software quality and sustainability, including funding for people to do this important work
  - Incorporate expectations for transparency and reproducibility
- Everyone
  - Work toward changes in software citations/credit models, metrics
  - Work toward changes in incentives, training and education

- *L. C. McInnes et al. (2024). A Cast of Thousands: How the IDEAS Productivity Project Has Advanced Software Productivity and Sustainability. IEEE CS&E.*
- *https://ideas-productivity.org*
- *https://bssw.io*

IDEAS-ECP: Outreach

Webinar Series:
Best Practices for HPC Software Developers

- *73 webinars (June 2017 – December 2023)*
- *12,000+ registrations in total*
- *average: 170 registrations (45 ECP-affiliated), 80 attendees*



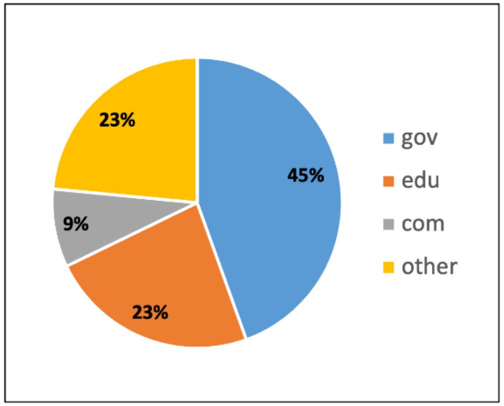
Slides and videos available at
https://ideas-productivity.org/events

Tutorials

- ***Overview of Best Practices in HPC Software Development***
- ***Code Coverage and Continuous Integration***
- ***An Introduction to Software Licensing***
- ***Agile Project Management***
- ***Continuous Integration***
- ***Software Refactoring***
- ***Software Design***
- ***Software Testing***
- ***Reproducibility***
- ***Git Workflows***

Panel Series

- ***Performance Portability & ECP***
- ***Strategies for Working Remotely***

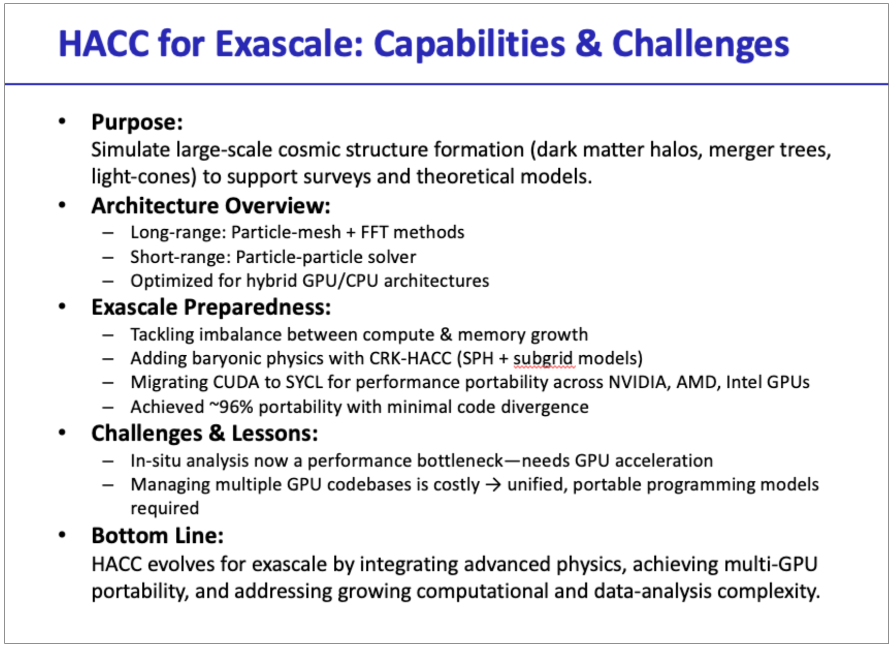


Opportunities (w.r.t. education/training)

- Categorize and summarize slides
  - Summary of meetings can already be done with Zoom's AI Companion
- Combine training materials
- Create customized material, targeting users' needs, interests and time
- Help with the selection of topics for future events

⋮

October 11, 2023 Webinar
Taking HACC into the Exascale Era: New Code Capabilities, and Challenges
Esteban Rangel (Argonne National Laboratory)
- one slide summary -



HACC for Exascale: Capabilities & Challenges

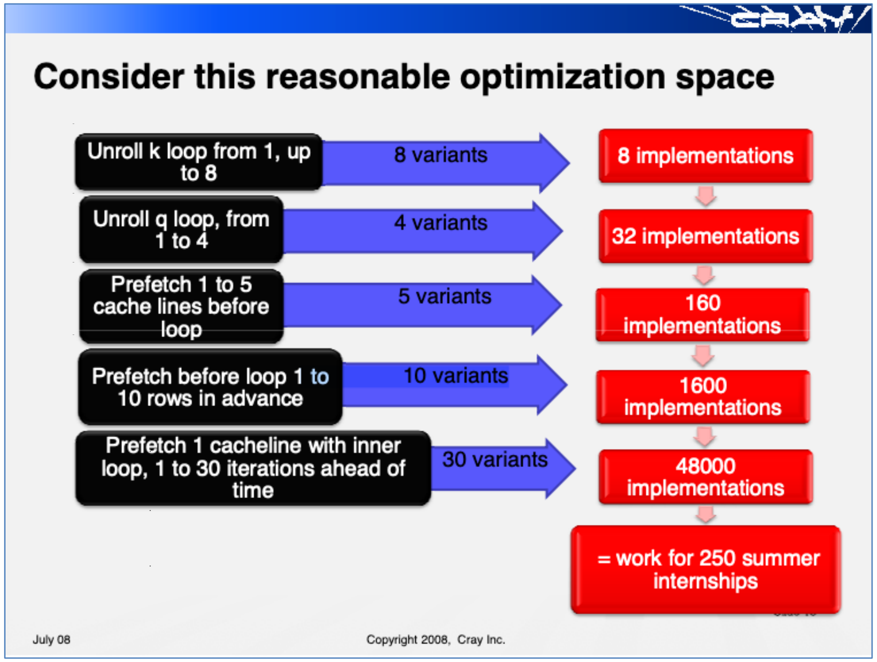
- **Purpose:**
  Simulate large-scale cosmic structure formation (dark matter halos, merger trees, light-cones) to support surveys and theoretical models.
- **Architecture Overview:**
  - Long-range: Particle-mesh + FFT methods
  - Short-range: Particle-particle solver
  - Optimized for hybrid GPU/CPU architectures
- **Exascale Preparedness:**
  - Tackling imbalance between compute & memory growth
  - Adding baryonic physics with CRK-HACC (SPH + subgrid models)
  - Migrating CUDA to SYCL for performance portability across NVIDIA, AMD, Intel GPUs
  - Achieved ~96% portability with minimal code divergence
- **Challenges & Lessons:**
  - In-situ analysis now a performance bottleneck—needs GPU acceleration
  - Managing multiple GPU codebases is costly → unified, portable programming models required
- **Bottom Line:**
  HACC evolves for exascale by integrating advanced physics, achieving multi-GPU portability, and addressing growing computational and data-analysis complexity.

Automatic Tuning

- Numerical software often has many parameters that affects its performance on HPC platforms
  - determining optimal values for such parameters in an impromptu way is very difficult (if not impossible)
  - automatically adjusting parameters – autotuning – is of great interest

Automatic Tuning

- Numerical software often has many parameters that affects its performance on HPC platforms
  - determining optimal values for such parameters in an impromptu way is very difficult (if not impossible)
  - automatically adjusting parameters – autotuning – is of great interest



Consider this reasonable optimization space

Unroll k loop from 1, up to 8 → 8 variants → 8 implementations

Unroll q loop, from 1 to 4 → 4 variants → 32 implementations

Prefetch 1 to 5 cache lines before loop → 5 variants → 160 implementations

Prefetch before loop 1 to 10 rows in advance → 10 variants → 1600 implementations

Prefetch 1 cacheline with inner loop, 1 to 30 iterations ahead of time → 30 variants → 48000 implementations

= work for 250 summer internships

July 08

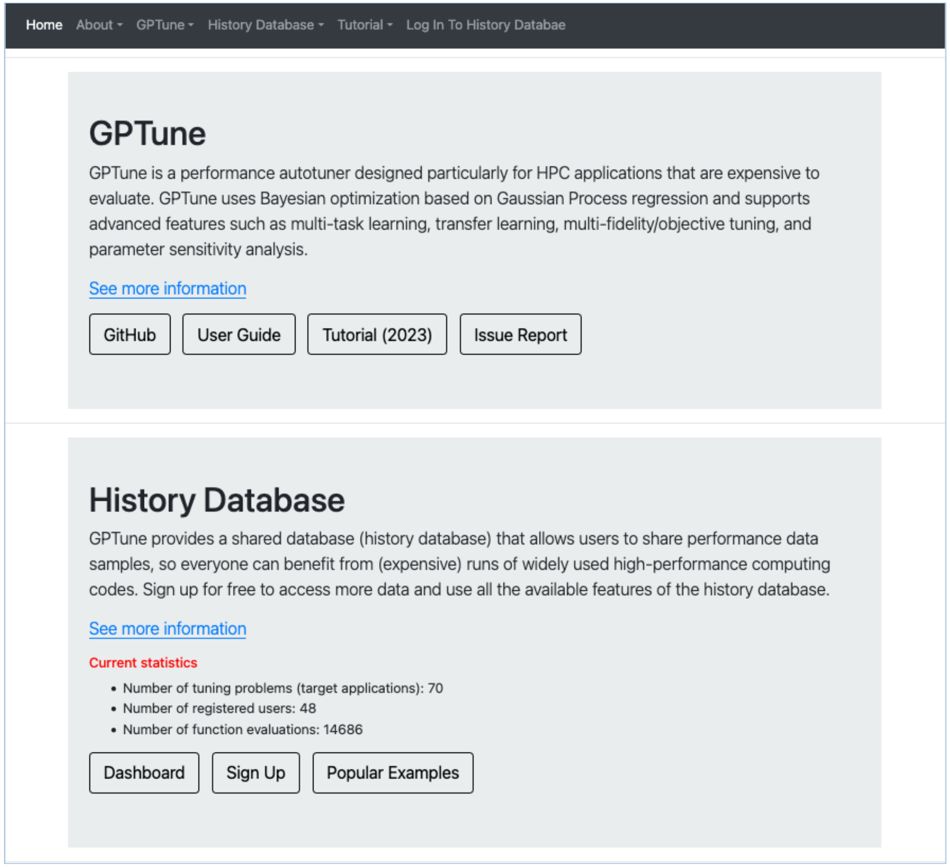
Copyright 2008, Cray Inc.

Automatic Tuning

- Numerical software often has many parameters that affects its performance on HPC platforms
  - determining optimal values for such parameters in an impromptu way is very difficult (if not impossible)
  - automatically adjusting parameters – autotuning – is of great interest
- Autotuning
  - directives: OSKI, ppOpen-AT, Xevolve …
  - compilers: CHiLL, Orio, POET, moTuner …
  - code generators: cuDNN, FFTW, Spiral …
  - frameworks: black-box optimization problem (models)
    - OpenTuner: *https://opentuner.org*
    - Auto-Tuning Framework (ATF): *https://atf-tuner.org*
    - Optuna: *https://optuna.org*
    - GPTune: *https://github.com/gptune/GPTune*

Automatic Tuning: GPTune

- Auto-tuning framework developed under the US Department of Energy's ECP
- It uses Bayesian optimization based on Gaussian Process regression and supports advanced features such as multi-objective tuning and parameter sensitivity analysis
- It targets the autotuning of HPC codes
- Readily available and adaptable
- History database



GPTune

GPTune is a performance autotuner designed particularly for HPC applications that are expensive to evaluate. GPTune uses Bayesian optimization based on Gaussian Process regression and supports advanced features such as multi-task learning, transfer learning, multi-fidelity/objective tuning, and parameter sensitivity analysis.

See more information

History Database

GPTune provides a shared database (history database) that allows users to share performance data samples, so everyone can benefit from (expensive) runs of widely used high-performance computing codes. Sign up for free to access more data and use all the available features of the history database.

See more information

Current statistics

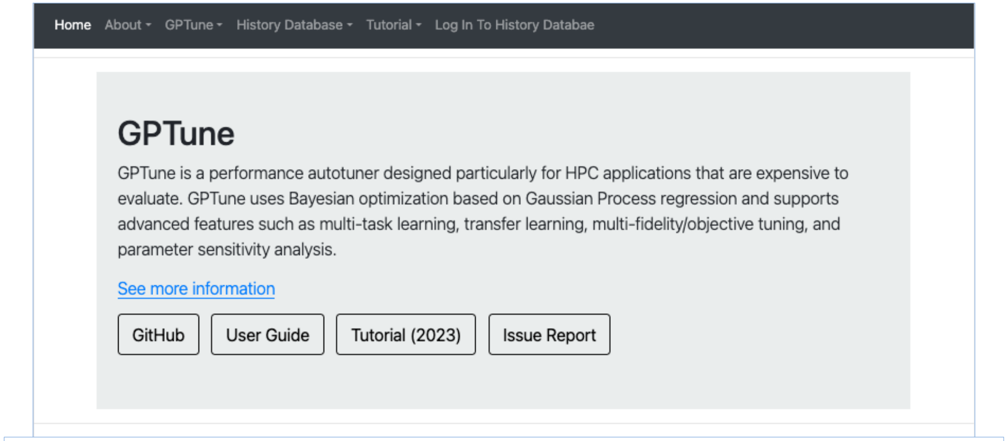
- Number of tuning problems (target applications): 70
- Number of registered users: 48
- Number of function evaluations: 14686

Liu et al. (2021). GPTune: multitask learning for autotuning exascale applications. Proc. 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.

Automatic Tuning: GPTune

- Auto-tuning framework developed under the US Department of Energy’s ECP
- It uses Bayesian optimization based on Gaussian Process regression and supports advanced features such as multi-objective tuning and parameter sensitivity analysis
- It targets the autotuning of HPC codes
- Readily available and adaptable
- History database

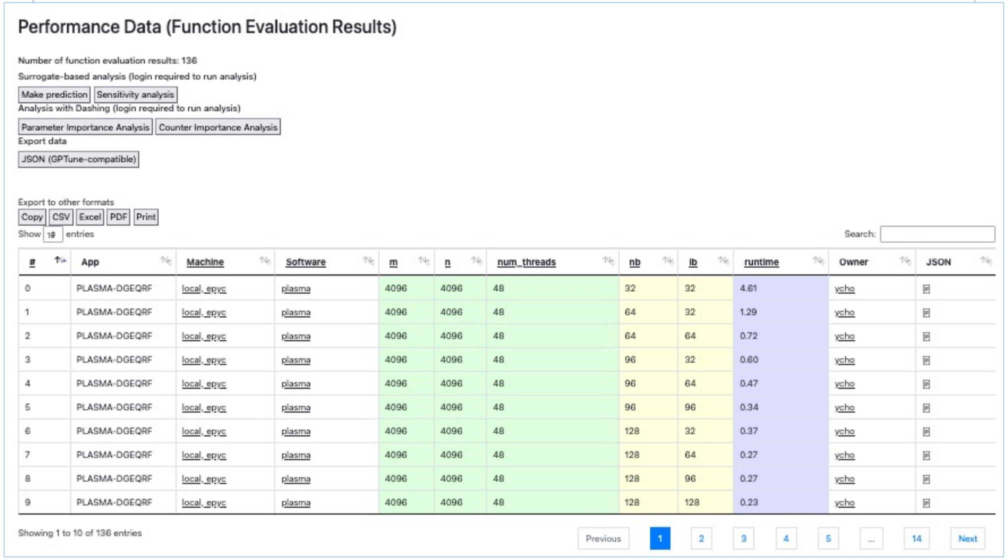
Liu et al. (2021). GPTune: multitask learning for autotuning exascale applications. Proc. 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.



GPTune

GPTune is a performance autotuner designed particularly for HPC applications that are expensive to evaluate. GPTune uses Bayesian optimization based on Gaussian Process regression and supports advanced features such as multi-task learning, transfer learning, multi-fidelity/objective tuning, and parameter sensitivity analysis.

See more information



Performance Data (Function Evaluation Results)

Number of function evaluation results: 136

Surrogate-based analysis (login required to run analysis)

Analysis with Dashing (login required to run analysis)

Export data

Export to other formats

Show 10 entries

Search:

#	App	Machine	Software	m	n	num_threads	nb	lb	runtime	Owner	JSON
0	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	32	32	4.61	ycho	🗎
1	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	64	32	1.29	ycho	🗎
2	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	64	64	0.72	ycho	🗎
3	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	96	32	0.60	ycho	🗎
4	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	96	64	0.47	ycho	🗎
5	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	96	96	0.34	ycho	🗎
6	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	128	32	0.37	ycho	🗎
7	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	128	64	0.27	ycho	🗎
8	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	128	96	0.27	ycho	🗎
9	PLASMA-DGEQRF	local, epyc	plasma	4096	4096	48	128	128	0.23	ycho	🗎

Showing 1 to 10 of 136 entries

GPTune: spaces

- Input (IS)
  - defines the problems to be tuned
  - each point in this space represents one instance of the problem
- Parameter (PS)
  - defines the parameters to be tuned
  - a point in this space corresponds a combination of the parameters
  - the tuner finds the best possible combination of parameters that minimizes the objective function associated with the problem to be tuned
- Output (OS)
  - Defines what needs to be optimized (e.g., runtime, accuracy, memory)

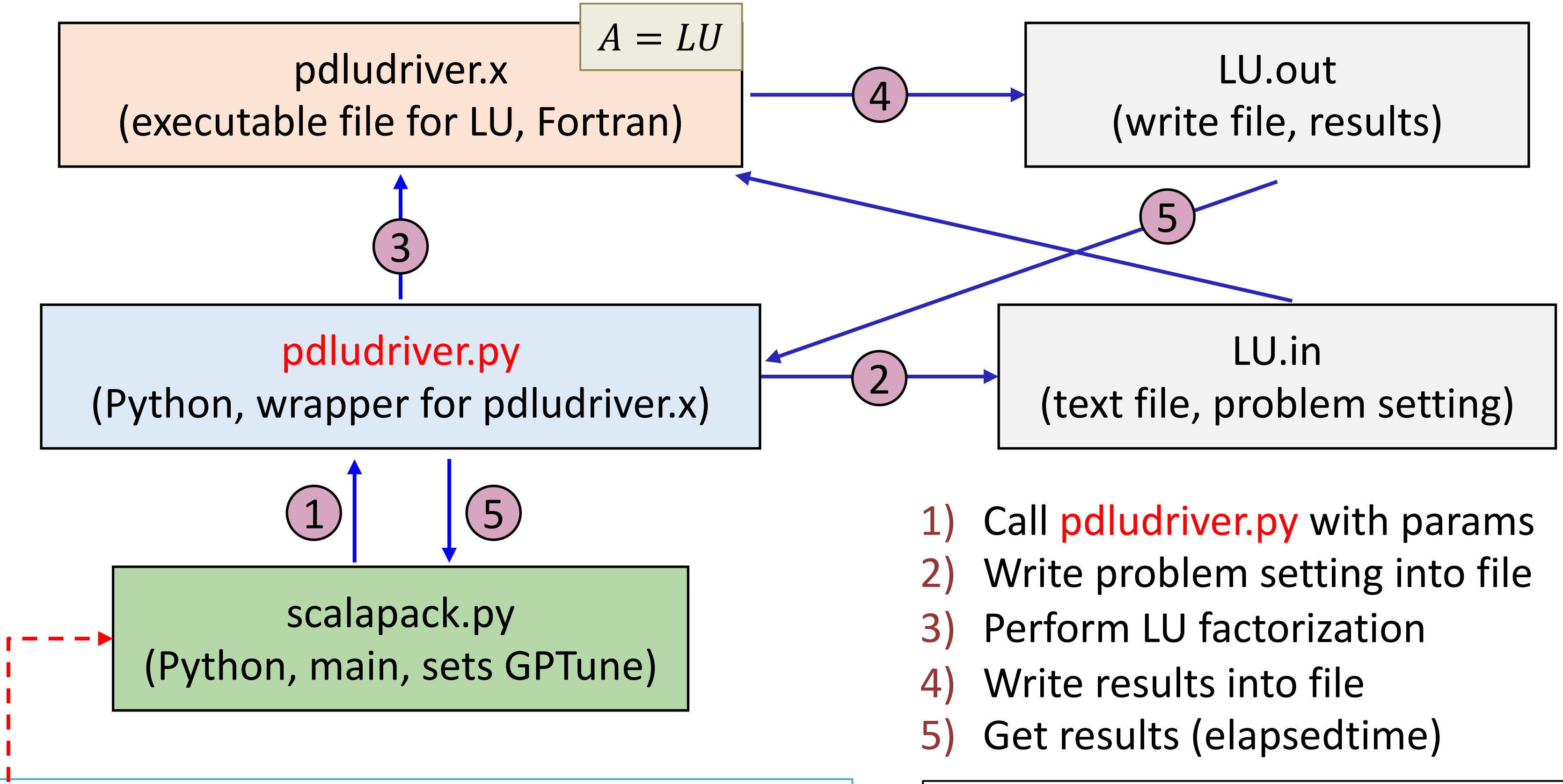
GPTune: spaces

- Input (IS)
  - defines the problem
  - each point in this sp represents one insta problem
- Parameter (PS)
  - defines the paramet tuned
  - a point in this space a combination of the
  - the tuner finds the b combination of para minimizes the objec associated with the tuned
- Output (OS)
  - Defines what needs optimized (e.g., runt accuracy, memory)

code snippets

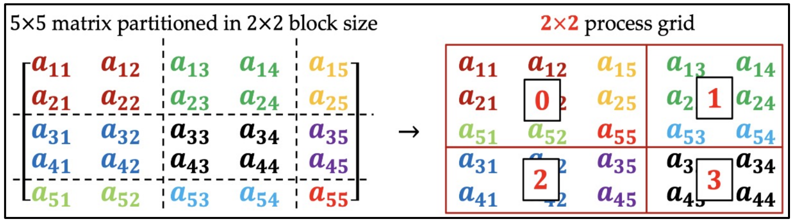
```
# matrix dimensions and number of processes

  m     = Integer(512, mmax, transform="normalize", name="m")
  n     = Integer(512, nmax, transform="normalize", name="n")
  nb    = Integer(1, 16, transform="normalize", name="nb")
  nproc = Integer(nprocmin, nprocmax, transform="normalize", name="nproc")
  p     = Integer(1, nprocmax, transform="normalize", name="p")
  r     = Real(float("-Inf"), float("Inf"), name="r")

# input, parameters and output spaces

  IS = Space([m, n])              # problem to be tuned
  PS = Space([nb, nproc, p])      # parameters to be tuned
  OS = Space([r])                 # what needs to be optimized
      ⋮

  problem = TuningProblem(IS, PS, OS, objective, constraints, …)
  computer = Computer(nodes, cores, hosts)

  options = …

  data = Data(problem)
  gt = GPTune(problem, computer, data, options)

  NI = ntask      # number of points in IS
  NS = nruns      # number of samples for each point in IS
  (data, model, stats) = gt.MLA(NS=NS, NI=NI, NS1=max(NS//2,1))
```

GPTune: interface with an application


$A = LU$
pdludriver.x
(executable file for LU, Fortran)
4
LU.out
(write file, results)
3
5
pdludriver.py
(Python, wrapper for pdludriver.x)
2
LU.in
(text file, problem setting)
1
5
scalapack.py
(Python, main, sets GPTune)

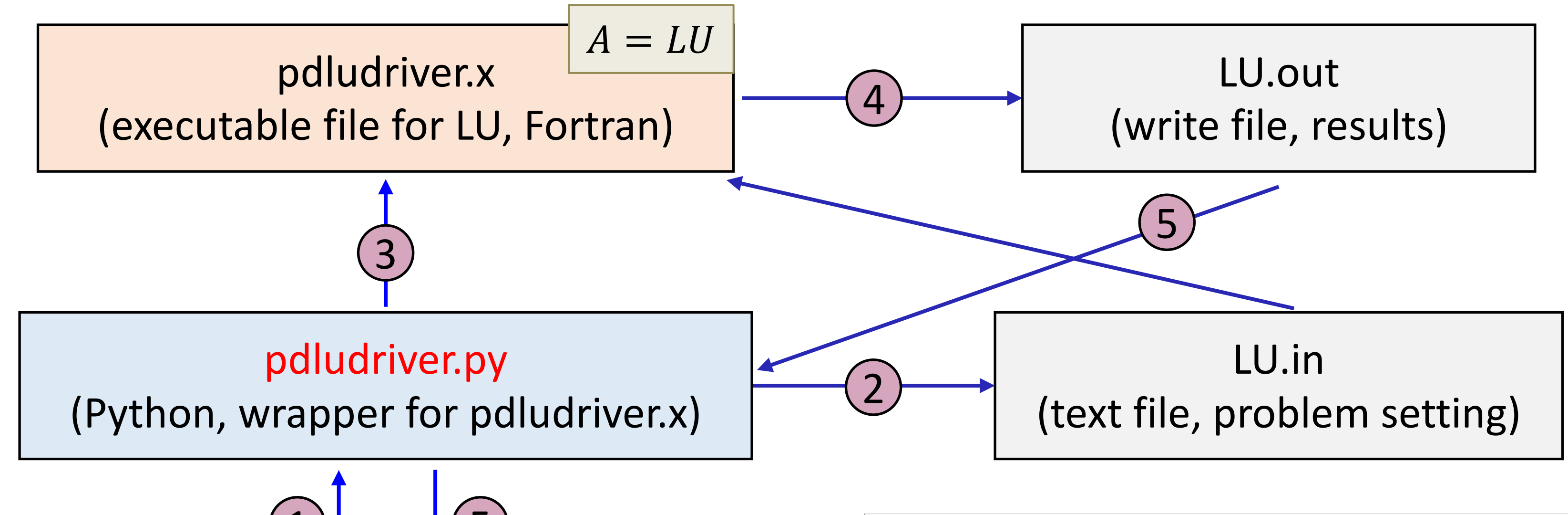
1) Call pdludriver.py with params
2) Write problem setting into file
3) Perform LU factorization
4) Write results into file
5) Get results (elapsedtime)

```
def objective(point):

    params = [('LU', n, nodes, cores, nb, nthreads, nproc, p, q, 1., npernode)]
    print(params, ' scalapack starts ‘)
    elapsedtime = pdludriver(params, niter=2, JOBID=JOBID)
    print(params, ' scalapack time: ', elapsedtime)

    return elapsedtime
```



nb	row block size
mb	column block size
p, q	process grid

GPTune: interface with an application



$A = LU$

pdludriver.x
(executable file for LU, Fortran)

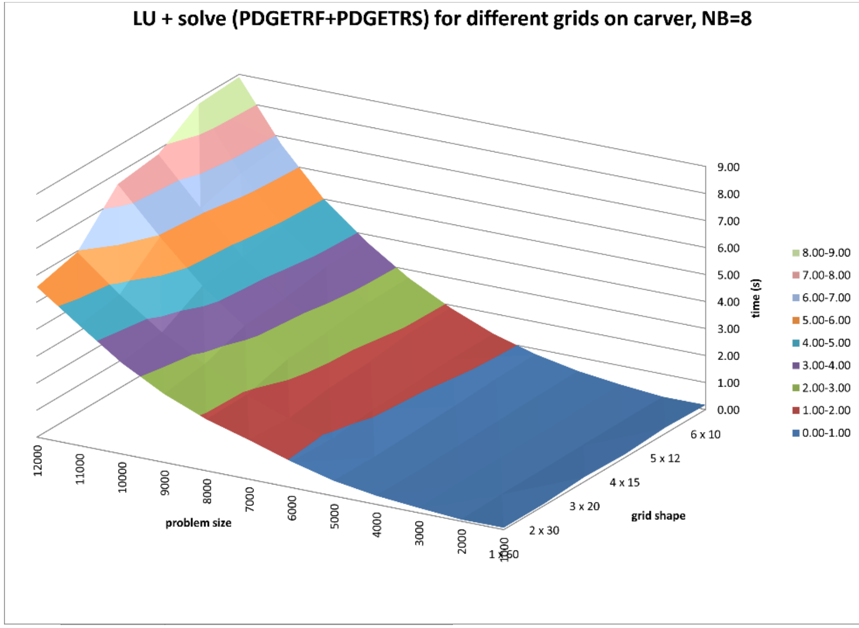
LU.out
(write file, results)

pdludriver.py
(Python, wrapper for pdludriver.x)

LU.in
(text file, problem setting)

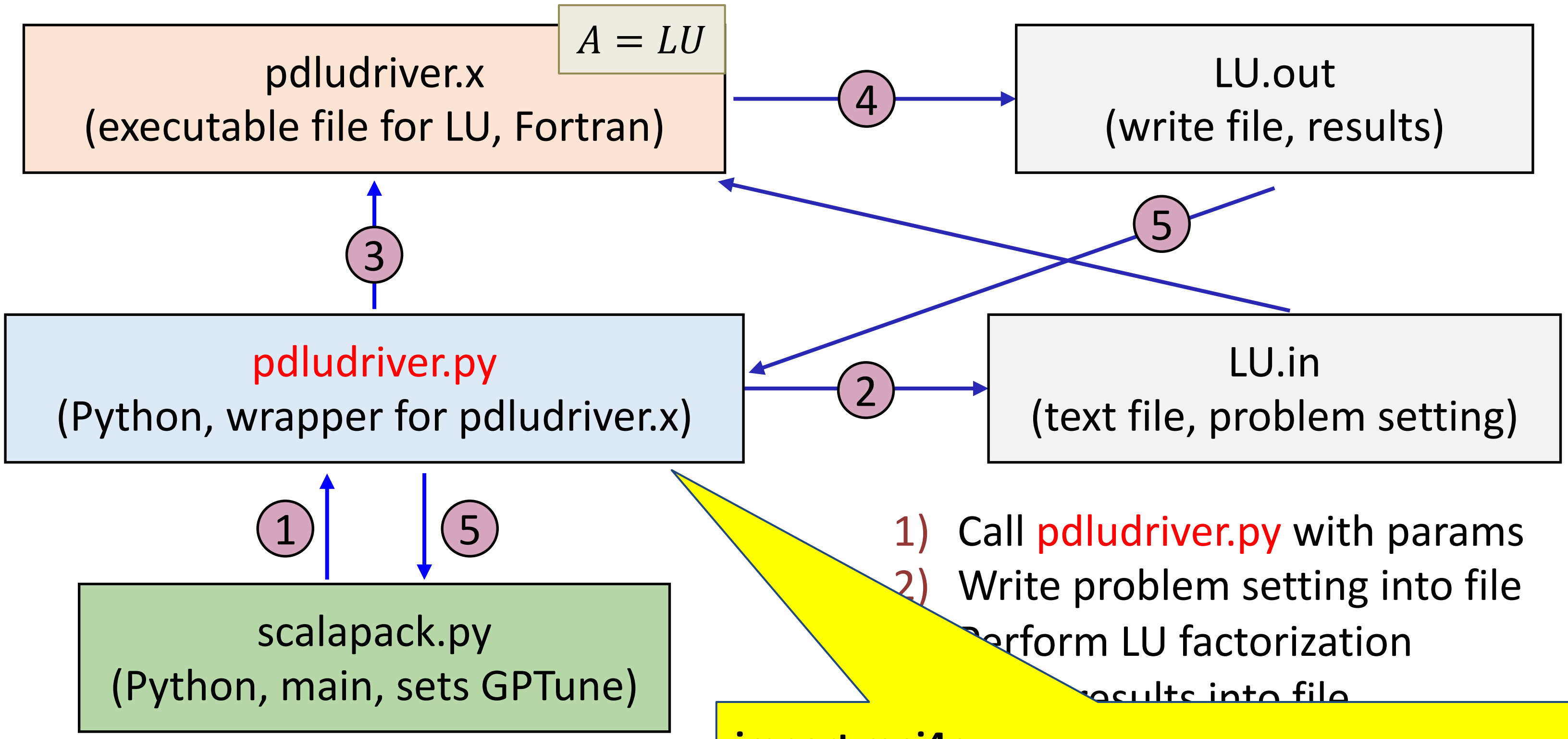
scalapack.py
(Python, main, sets GPTune)

```
def objective(point):

   params = [('LU', n, nodes, cores, nb, nthreads, nproc, p, q, 1., npernode)]
   print(params, ' scalapack starts ‘)
   elapsedtime = pdludriver(params, niter=2, JOBID=JOBID)
   print(params, ' scalapack time: ', elapsedtime)

   return elapsedtime
```



LU + solve (PDGETRF+PDGETRS) for different grids on carver, NB=8

GPTune: running



$A = LU$

pdludriver.x
(executable file for LU, Fortran)

LU.out
(write file, results)

pdludriver.py
(Python, wrapper for pdludriver.x)

LU.in
(text file, problem setting)

scalapack.py
(Python, main, sets GPTune)

1) Call pdludriver.py with params
2) Write problem setting into file
erform LU factorization
esults into file

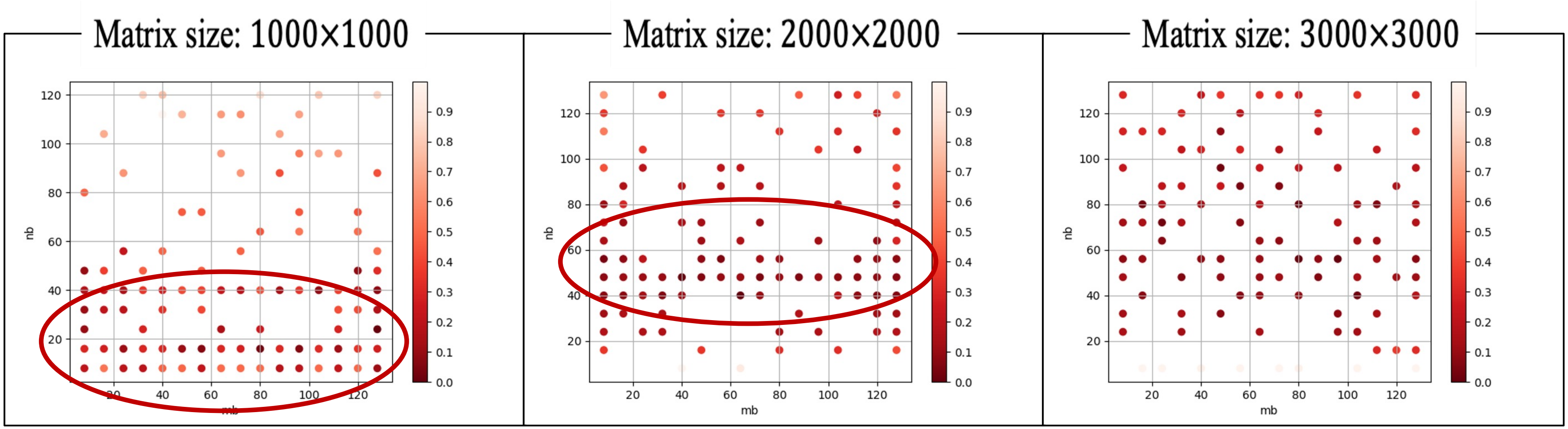
```
import mpi4py
  ⋮
comm = MPI.COMM_SELF.Spawn("pdludriver", … )
```

```
mpirun -n 1 python ./scalapack.py -nodes 1 -cores 32 …
```

Experiments: ScaLAPACK QR

$A = QR$
Q is orthogonal
R is upper triangular

- Experiments by Makoto Morishita
- MacBookAir (M1, 2020), Apple M1 chip, 8 cores, 8 GB, Ventura 13.4.1
- nb: 8~128; mb: 8~128; p, q: 1~8 (p×q <= 8)



Matrix size: 1000×1000

Matrix size: 2000×2000

Matrix size: 3000×3000

Execution time for 2 tuning parameters, block size (nb, mb), p=1

- *When the problem size is small (N = 1000) we get lower values for nb*
- *As the problem size increases (N = 3000) we get a larger set of possible nb's and mb's*
- *In general, the closer the block size is to a square (nb = mb), the more efficient the processing is*

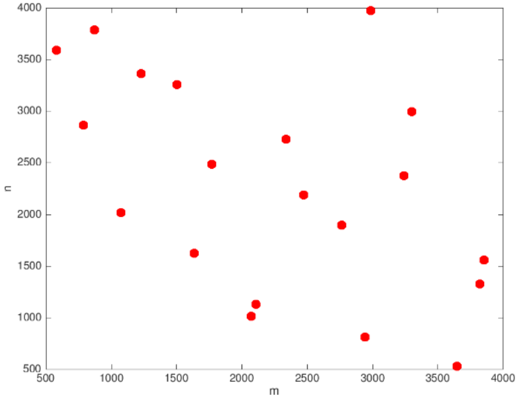
matrix size	(nb, mb)
1000×1000	(128, 24)
2000×2000	(40, 48)
3000×3000	(80, 56)

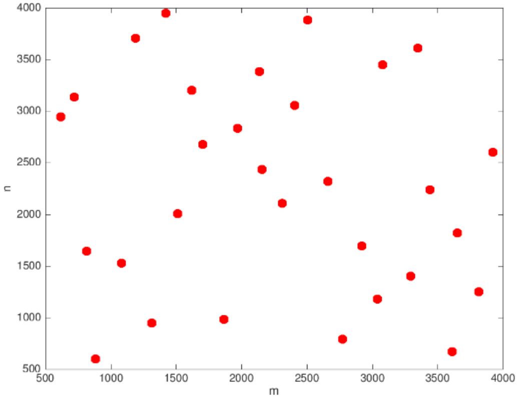
Experiments: ScaLAPACK PDGESVD

$A = USV^T$
U and V are orthog.
S is diagonal

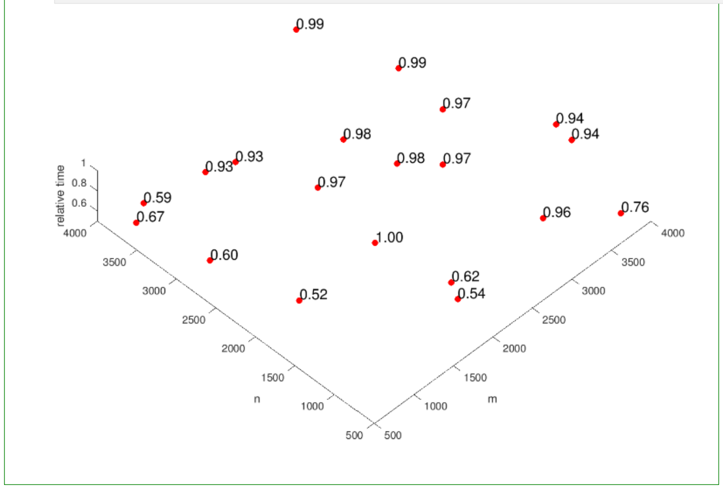
parameters for scalapack-SVD.py; min(m,n) = 512

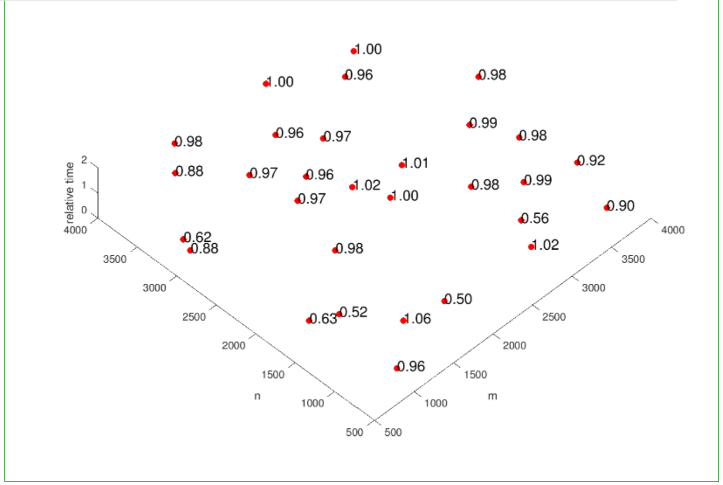
case	nodes	cores	m_{max}	n_{max}	ntask	nrun
1	1	32	4000	4000	20	30
2	1	32	4000	4000	30	20





Relative times for the matrix dimensions in the above figures, i.e. times for the optimal configurations obtained by GPTune divided by the time for standalone PDGESVD runs with nb = 64, p = 5 and q = 6





Experiments: ScaLAPACK PDSYEVX

$Ax = \lambda x$

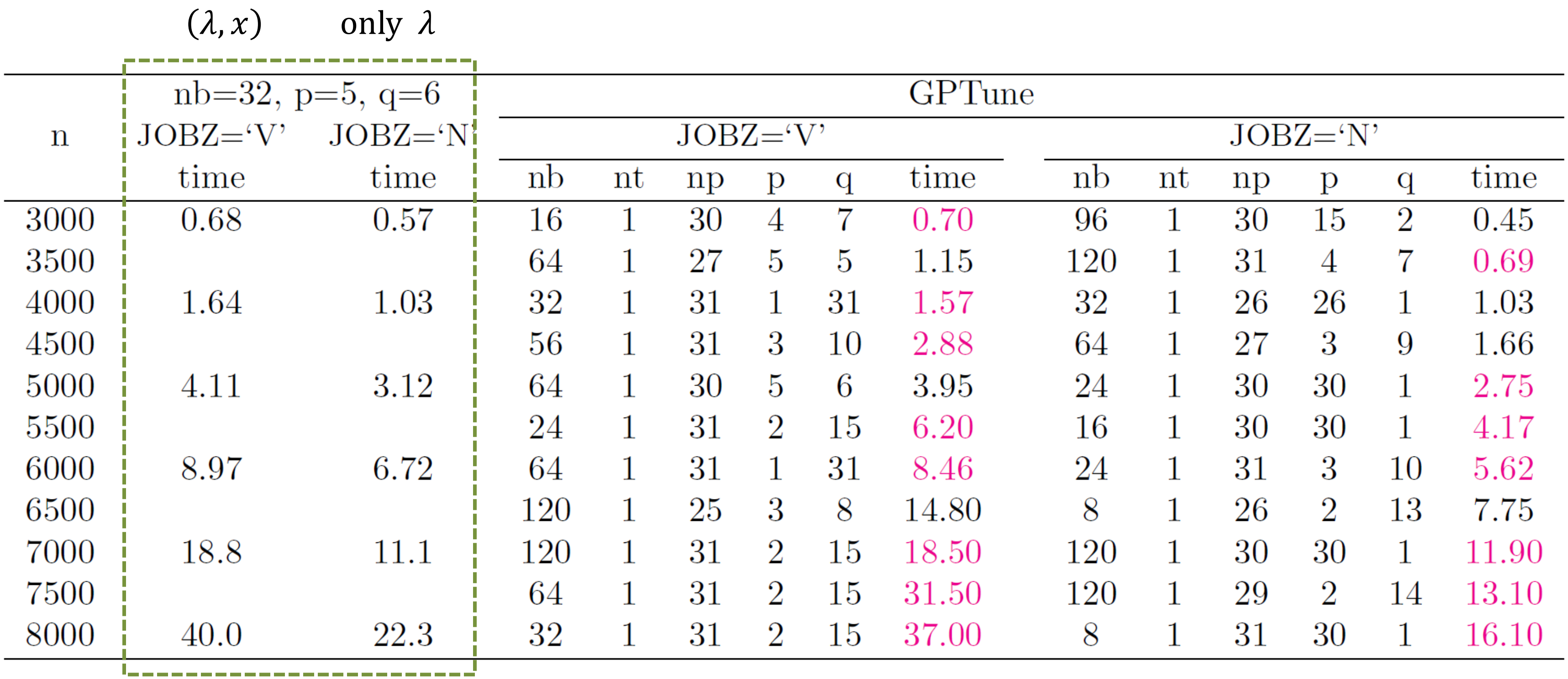
λ is a scalar

$X = [x_1\ x_1 \cdots x_n\ ]$ is orthogonal

A is first reduced to tridiagonal

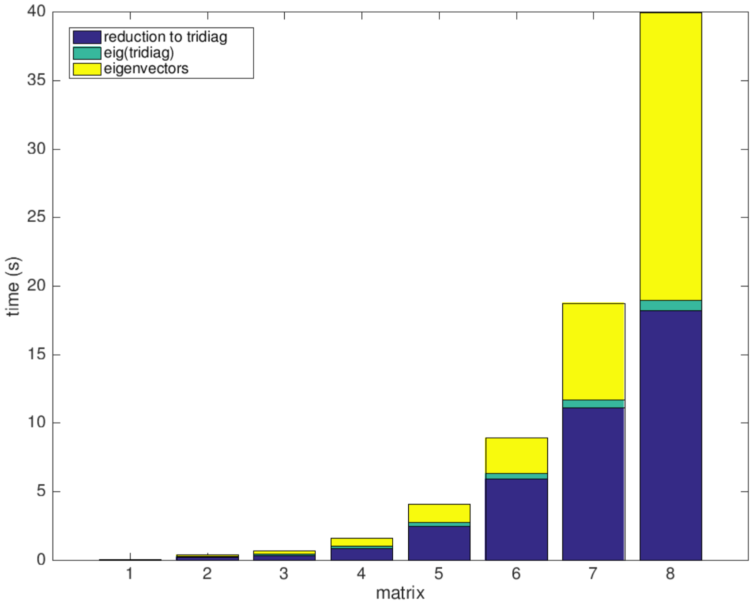
$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}$$

Times (s) for PDSYEVX on one node (haswell), standalone mode (dashed box) and configurations obtained with GPTune, NS = 10, "-nodes 1 -cores 32". In the table, nt = number of threads and np = number of processes. The times in magenta correspond to configurations found in the search phase.

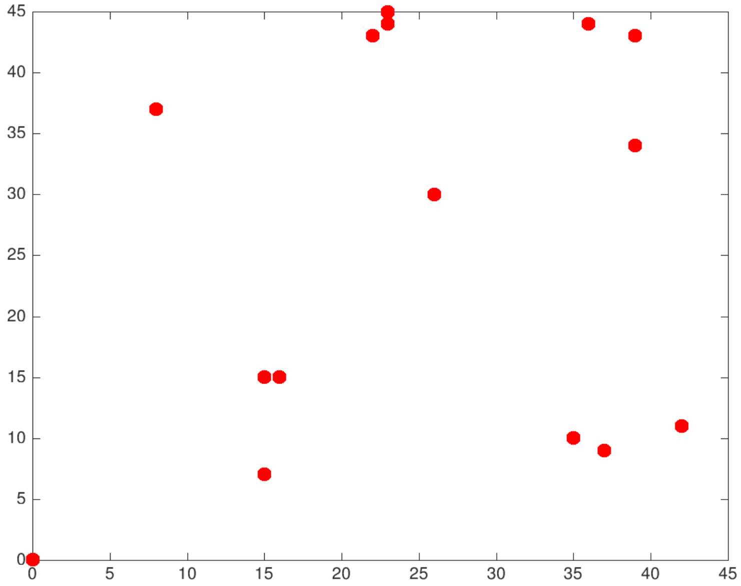


	(λ, x)	only λ												
	nb=32, p=5, q=6		GPTune											
n	JOBZ='V'	JOBZ='N'	JOBZ='V'						JOBZ='N'					
	time	time	nb	nt	np	p	q	time	nb	nt	np	p	q	time
3000	0.68	0.57	16	1	30	4	7	0.70	96	1	30	15	2	0.45
3500			64	1	27	5	5	1.15	120	1	31	4	7	0.69
4000	1.64	1.03	32	1	31	1	31	1.57	32	1	26	26	1	1.03
4500			56	1	31	3	10	2.88	64	1	27	3	9	1.66
5000	4.11	3.12	64	1	30	5	6	3.95	24	1	30	30	1	2.75
5500			24	1	31	2	15	6.20	16	1	30	30	1	4.17
6000	8.97	6.72	64	1	31	1	31	8.46	24	1	31	3	10	5.62
6500			120	1	25	3	8	14.80	8	1	26	2	13	7.75
7000	18.8	11.1	120	1	31	2	15	18.50	120	1	30	30	1	11.90
7500			64	1	31	2	15	31.50	120	1	29	2	14	13.10
8000	40.0	22.3	32	1	31	2	15	37.00	8	1	31	30	1	16.10

Experiments: ScaLAPACK PDSYEVX



Time breakdown for the main phases of PDSYEVX: reduction to tridiagonal form T, eigval(T), and eigvec(T) plus back transformation of eigenvectors. Matrices of dimension 1000 to 8000. JOBZ =`V', RANGE = `A’ and UPLO = `U'.



Finding: A large workspace (argument LWORK in PDSYEVX) can lead to a poor load balancing and poor performance because the algorithm takes into account load balance and memory in an intricate way. The figure shows active processes (eigvec(T) plus back transformation) for a matrix of dimension 10K in a 44x46 grid: process (23,44) computes 6 vectors while process (0,0) computes 2,026 vectors.

Opportunities (w.r.t. code performance)

- Performance models integrated with the generation of codes
- Use the feedback from running the generated code to fine-tune models
- Comparison of models
- Address model limitations
- Evaluation benchmarks that go beyond correctness to include metrics for code robustness and performance
- Detection and interpretability of hidden outliers

⋮

- *Wei et al. (2025). Improving Parallel Program Performance with LLM Optimizers via Agent-System Interfaces. Proc. 42nd Int. Conf. on Machine Learning.*
- *Cribeiro-Ramallo, Arzamasov and Boehm (2024). Efficient Generation of Hidden Outliers for Improved Outlier Detection. ACM Transactions on Knowledge Discovery from Data.*

Testing: LAPACK as a case study

- User end
  - set of tests performed at installation time
  - single, double, real, complex
- Developer end
  - no specific guidelines
  - single, double, real, complex
  - tests expected to be (very) difficult
  - very tedious and time-consuming process
  - bugs reported by users may take years to be fixed
- Focus: $Ax = \lambda x$ (symmetric A) and $A = USV^T$ (general A)
  - tests performed with all kinds of matrices (“torture” cases)
  - oamarques/STCollection
    - test cases
    - software

Symmetric Tridiagonal Eigensolver

$Ax = \lambda x$

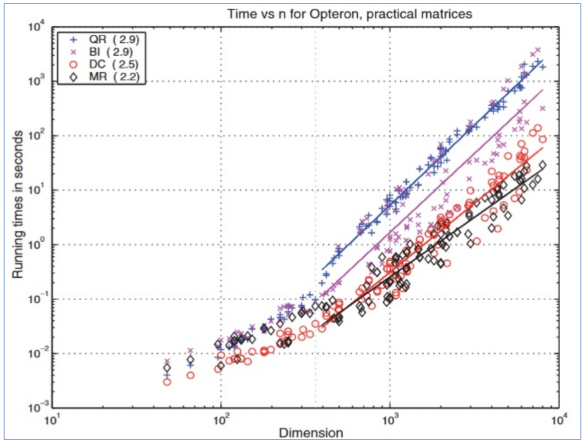
λ is a scalar

$X = [x_1\ x_1 \cdots x_n\,]$ is orthogonal

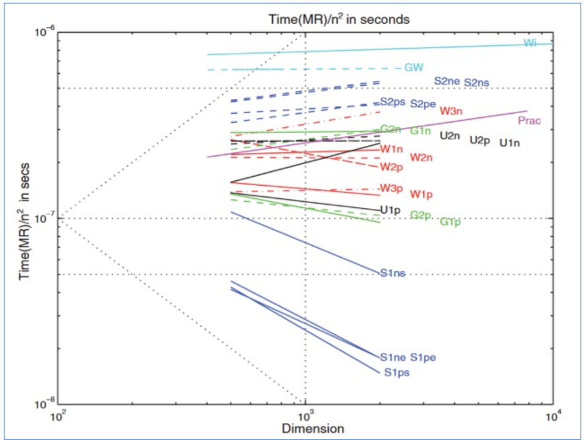
A is first reduced to tridiagonal

$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}$$

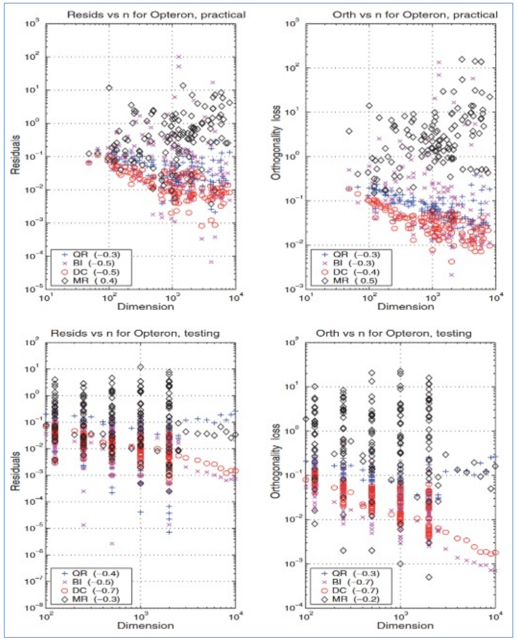
- Four algorithms (going back to LAPACK 3.1…)
  - QR iteration
  - bisection and inverse iteration (BI)
  - divide-and-conquer (DC)
  - multiple relatively robust representations (MRRR)
- Experiments performed on 8 different architectures
- Large amount of data (performance and accuracy)



Run time of all algorithms on Opteron. The slopes of the least-squares fit, shown in parentheses, are computed from the larger matrices.



Performance trend lines of MRRR for run time divided by n² on Opteron.



Residuals and losses of orthogonality for all matrices on Opteron. (Top: All practical matrices. Bottom: All synthetic testing matrices. Note the difference in vertical scales.)

Demmel et al. (2008). Performance and Accuracy of LAPACK's Symmetric Tridiagonal Eigensolvers. SIAM J. Sci. Comp.

BDSVDX

$A = USV^T$
U and V are orthogonal
S is diagonal
A is first reduced to bidiag. form

- $B = \widehat{U}\hat{S}\widehat{V}^T$ through the eigenpairs of an associated symmetric tridiagonal matrix
- Computation of only a subset of singular values and corresponding vectors (potential performance gains)

$$B = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \end{bmatrix} \Rightarrow \; C = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} = P T_{GK} P^T; \;\; P = [e_{n+1}, e_1, e_{n+2}, e_2, e_{n+2}, \cdots e_{2n}]$$

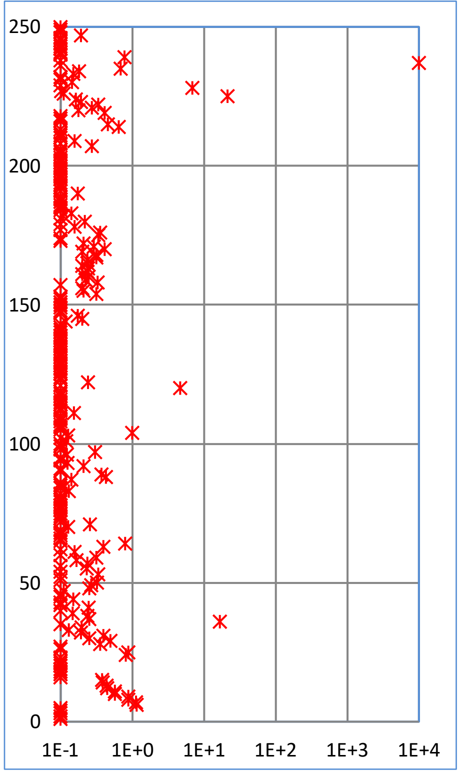
$$T_{GK} = \begin{bmatrix} 0 & a_1 & & & & \\ a_1 & 0 & b_1 & & & \\ & b_1 & 0 & \ddots & & \\ & & \ddots & \ddots & b_{n-1} & \\ & & & b_{n-1} & 0 & a_n \\ & & & 0 & a_n & 0 \end{bmatrix}$$

OM: If a and/or b is zero the matrix splits → the algorithm and the testing become complicated…

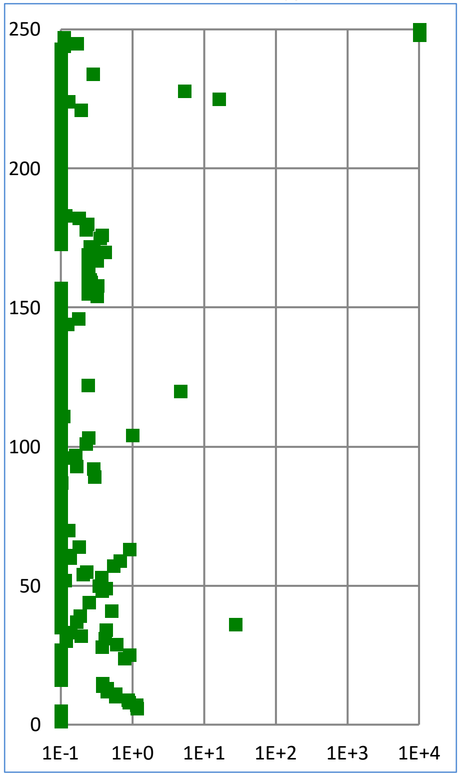
Marques, Demmel and *Vasconcelos (2020). Bidiagonal SVD Computation via an Associated Tridiagonal Eigenproblem. ACM TOMS.*

BDSVDX: residual and orthogonality, RANGE="A"

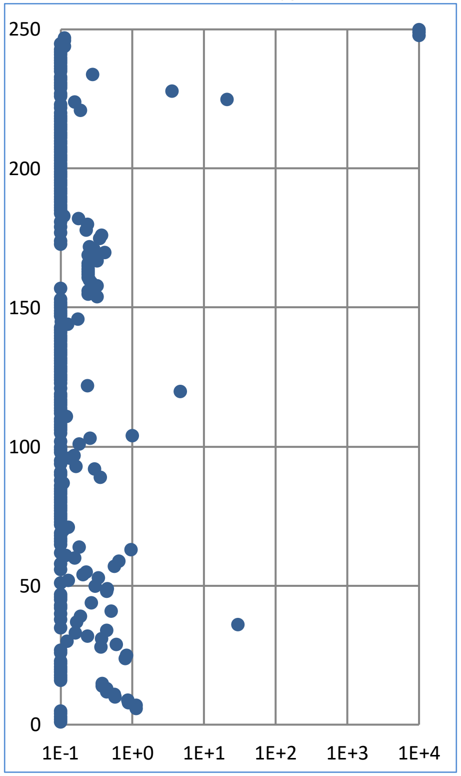
$$\text{resid} = \frac{\|\hat{U}^T B \hat{V} - \hat{S}\|}{\|B\| n\varepsilon}$$

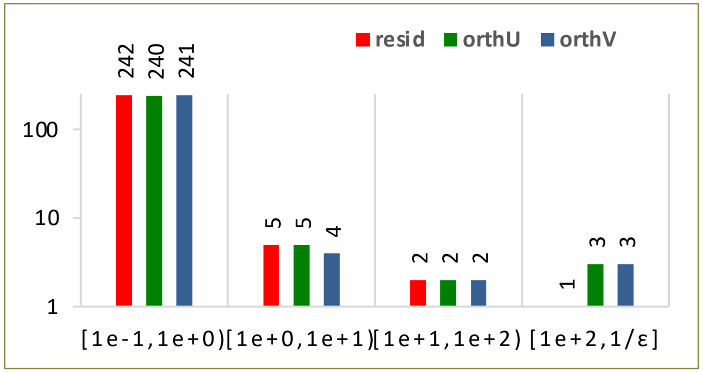


$$\text{orthU} = \frac{\|I - \hat{U}^T \hat{U}\|}{n\varepsilon}$$



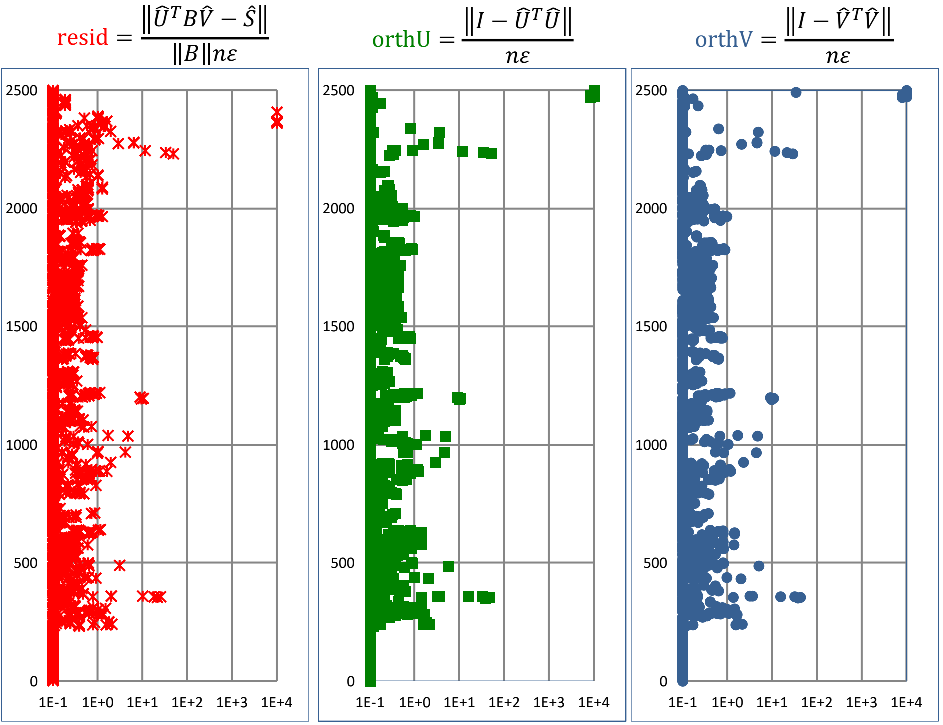
$$\text{orthV} = \frac{\|I - \hat{V}^T \hat{V}\|}{n\varepsilon}$$



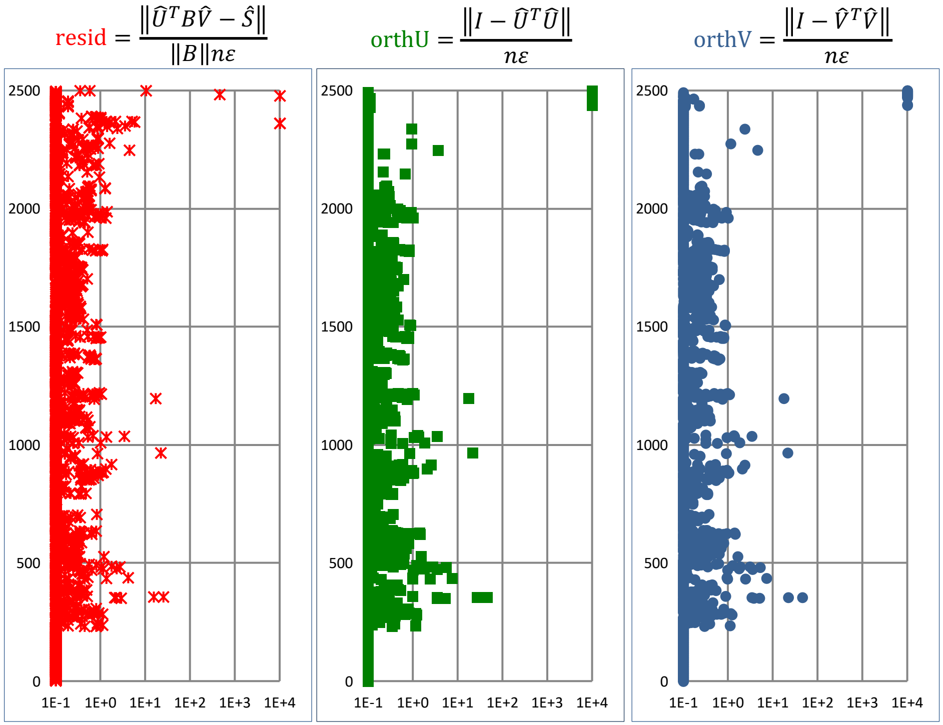


BDSVDX: residual and orthogonality, RANGE= “I” and “V”

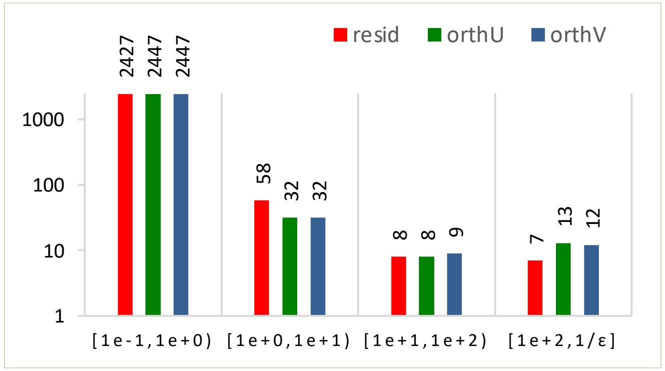
results for “I” (left) and “V” (right) generated randomly, 10 pairs per matrix

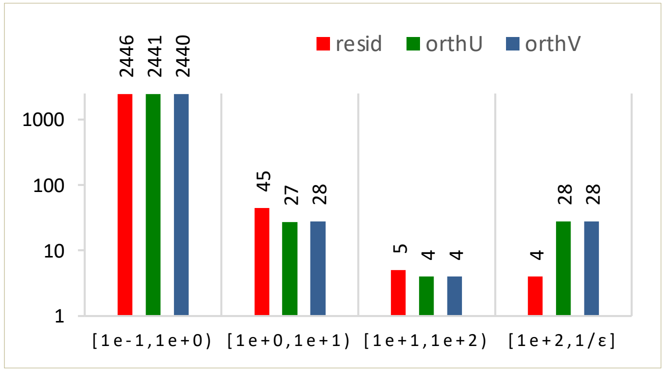


$\text{resid} = \frac{\|\hat{U}^T B \hat{V} - \hat{S}\|}{\|B\| n\varepsilon}$ $\text{orthU} = \frac{\|I - \hat{U}^T \hat{U}\|}{n\varepsilon}$ $\text{orthV} = \frac{\|I - \hat{V}^T \hat{V}\|}{n\varepsilon}$



$\text{resid} = \frac{\|\hat{U}^T B \hat{V} - \hat{S}\|}{\|B\| n\varepsilon}$ $\text{orthU} = \frac{\|I - \hat{U}^T \hat{U}\|}{n\varepsilon}$ $\text{orthV} = \frac{\|I - \hat{V}^T \hat{V}\|}{n\varepsilon}$

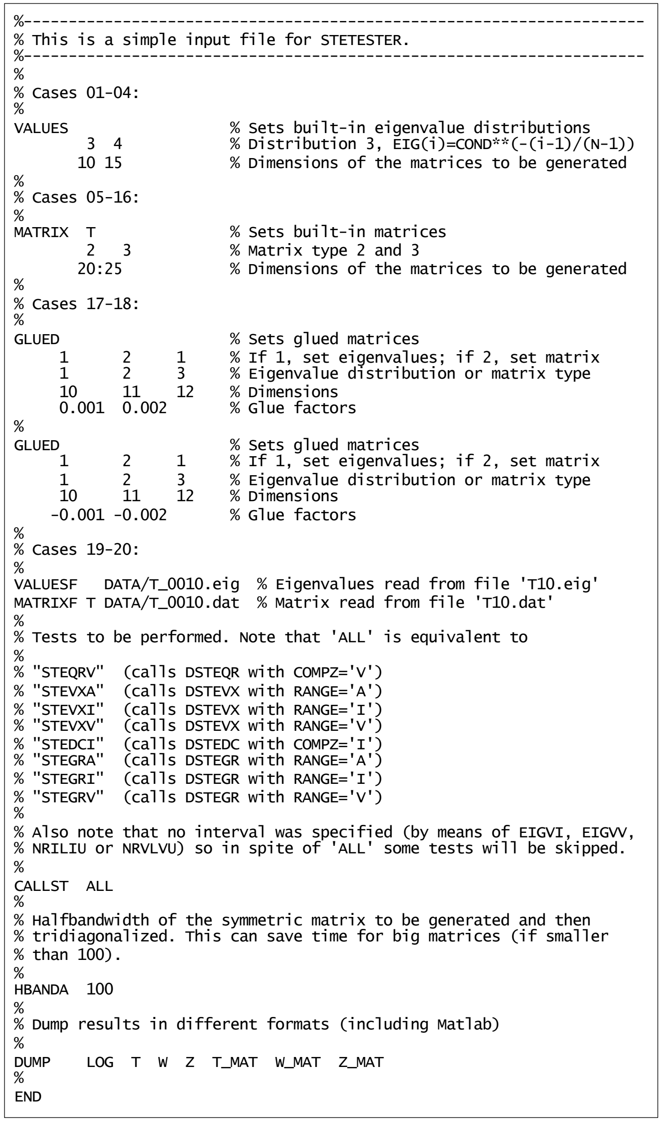




STCollection: hard tests behind the scenes

- Developed for $Ts = \theta s$
- Extended to $B = \hat{U}\hat{S}\hat{V}^T$
- Matrices and eigenvalue distributions (and corresponding plots)
  - Built-in eigenvalue distributions (e.g. geometric) and matrices (e.g. Wilkinson)
  - Tridiagonal matrices derived from real world applications
    - Chemistry (analysis of molecules)
    - Harwell-Boeing Collection (structural engineering, etc)
    - SuiteSparse Matrix Collection (FEM analysis, NASA)
    - Matrices provided by LAPACK users (some related to bugs)
    - Lanczos algorithm without reorthogonalization to provoke very close eigenvalues
- Software (various versions)
- Flexible input file

Marques et al. (2008). Algorithm 880: A testing infrastructure for symmetric tridiagonal eigensolvers. ACM TOMS.

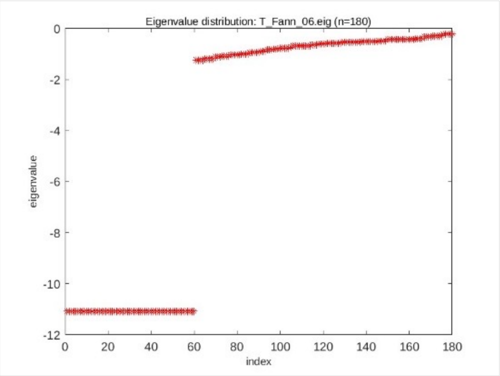


```
%--------------------------------------------------------------------
% This is a simple input file for STETESTER.
%--------------------------------------------------------------------
%
% Cases 01-04:
%
VALUES                  % Sets built-in eigenvalue distributions
        3  4            % Distribution 3, EIG(i)=COND**(-(i-1)/(N-1))
       10 15            % Dimensions of the matrices to be generated
%
% Cases 05-16:
%
MATRIX  T               % Sets built-in matrices
        2   3           % Matrix type 2 and 3
       20:25            % Dimensions of the matrices to be generated
%
% Cases 17-18:
%
GLUED                   % Sets glued matrices
     1      2     1     % If 1, set eigenvalues; if 2, set matrix
     1      2     3     % Eigenvalue distribution or matrix type
     10     11    12    % Dimensions
     0.001  0.002       % Glue factors
%
GLUED                   % Sets glued matrices
     1      2     1     % If 1, set eigenvalues; if 2, set matrix
     1      2     3     % Eigenvalue distribution or matrix type
     10     11    12    % Dimensions
    -0.001 -0.002       % Glue factors
%
% Cases 19-20:
%
VALUESF   DATA/T_0010.eig  % Eigenvalues read from file 'T10.eig'
MATRIXF T DATA/T_0010.dat  % Matrix read from file 'T10.dat'
%
% Tests to be performed. Note that 'ALL' is equivalent to
%
% "STEQRV"  (calls DSTEQR with COMPZ='V')
% "STEVXA"  (calls DSTEVX with RANGE='A')
% "STEVXI"  (calls DSTEVX with RANGE='I')
% "STEVXV"  (calls DSTEVX with RANGE='V')
% "STEDCI"  (calls DSTEDC with COMPZ='I')
% "STEGRA"  (calls DSTEGR with RANGE='A')
% "STEGRI"  (calls DSTEGR with RANGE='I')
% "STEGRV"  (calls DSTEGR with RANGE='V')
%
% Also note that no interval was specified (by means of EIGVI, EIGVV,
% NRILIU or NRVLVU) so in spite of 'ALL' some tests will be skipped.
%
CALLST  ALL
%
% Halfbandwidth of the symmetric matrix to be generated and then
% tridiagonalized. This can save time for big matrices (if smaller
% than 100).
%
HBANDA  100
%
% Dump results in different formats (including Matlab)
%
DUMP    LOG  T  W  Z  T_MAT  W_MAT  Z_MAT
%
END
```

STCollection: eigenvalue distributions

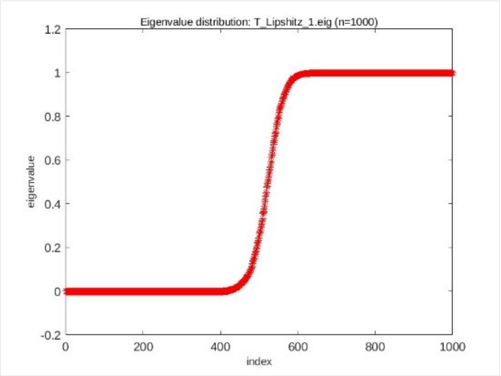
T_Fann_06.eig
n=180, 20 smallest eigenvalues

-1.107582174359294E+01
-1.107582174359294E+01
-1.107582174359292E+01
-1.107582174359291E+01
-1.107582174359290E+01
-1.107580538638403E+01
-1.107580538638399E+01
-1.107580538638398E+01
-1.107580538638397E+01
-1.107579468129299E+01
-1.107579468129296E+01
-1.107579468129296E+01
-1.107579468129292E+01
-1.107579468129291E+01
-1.107574609356799E+01
-1.107574609356797E+01
-1.107574609356795E+01
-1.107574609356793E+01
-1.107571441013466E+01
-1.107571441013465E+01



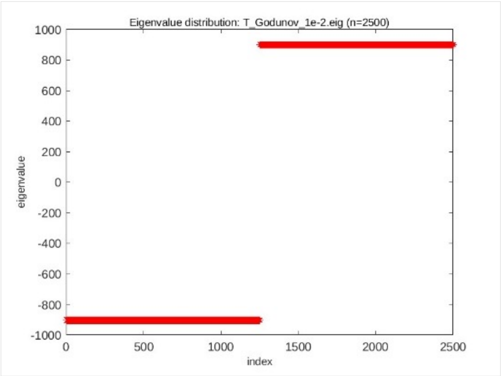
T_Lipshitz_1.eig
n=1000, 20 smallest and largest eigenvalues

-2.6823372031320144e-16	1.0000000000000020e+00
-2.2237154507547389e-16	1.0000000000000022e+00
-2.2204461419491012e-16	1.0000000000000022e+00
-2.2204460492503131e-16	1.0000000000000022e+00
-1.8342609148280882e-16	1.0000000000000022e+00
-1.3899996366291143e-16	1.0000000000000024e+00
-1.1135703515315642e-16	1.0000000000000024e+00
-5.5511152667964514e-17	1.0000000000000024e+00
-4.9656459539928424e-17	1.0000000000000024e+00
-3.6224205091068070e-17	1.0000000000000024e+00
-3.4769060438593381e-17	1.0000000000000024e+00
-2.9457359363955050e-17	1.0000000000000024e+00
-1.8099200983560987e-17	1.0000000000000024e+00
-9.9708593735412647e-18	1.0000000000000029e+00
-4.4639982898565932e-18	1.0000000000000029e+00
-4.1246352014896293e-18	1.0000000000000029e+00
-3.9284577008713904e-18	1.0000000000000029e+00
-3.6526711419576054e-18	1.0000000000000031e+00
-2.2491630371466091e-18	1.0000000000000031e+00
-1.8965391276166353e-18	1.0000000000000033e+00



T_Godunov_1e-2.eig
n=2500, 20 smallest and largest eigenvalues

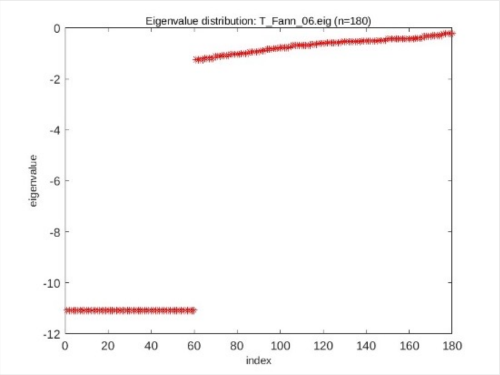
-9.000099999684682E+02	9.000099873898861E+02
-9.000099998738725E+02	9.000099886191389E+02
-9.000099997162141E+02	9.000099897853999E+02
-9.000099994954936E+02	9.000099908886612E+02
-9.000099992117126E+02	9.000099919289162E+02
-9.000099988648727E+02	9.000099929061586E+02
-9.000099984549761E+02	9.000099938203817E+02
-9.000099979820255E+02	9.000099946715800E+02
-9.000099974460236E+02	9.000099954597482E+02
-9.000099968469744E+02	9.000099961848813E+02
-9.000099961848813E+02	9.000099968469744E+02
-9.000099954597482E+02	9.000099974460236E+02
-9.000099946715800E+02	9.000099979820255E+02
-9.000099938203817E+02	9.000099984549761E+02
-9.000099929061586E+02	9.000099988648727E+02
-9.000099919289162E+02	9.000099992117126E+02
-9.000099908886612E+02	9.000099994954936E+02
-9.000099897853999E+02	9.000099997162141E+02
-9.000099886191389E+02	9.000099998738725E+02
-9.000099873898861E+02	9.000099999684682E+02



STCollection: eigenvalue distributions

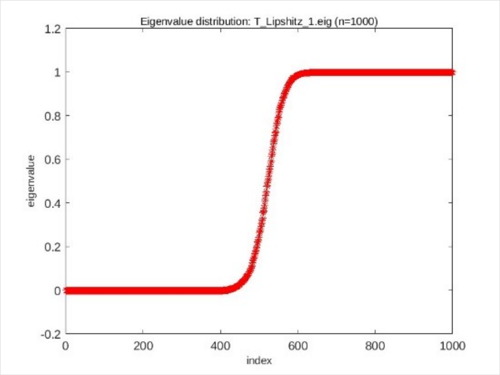
T_Fann_06.eig
n=180, 20 smallest eigenvalues

```
-1.107582174359294E+01
-1.107582174359294E+01
-1.107582174359292E+01
-1.107582174359291E+01
-1.107582174359290E+01
-1.107580538638403E+01
-1.107580538638399E+01
-1.107580538638398E+01
-1.107580538638397E+01
-1.107579468129299E+01
-1.107579468129296E+01
-1.107579468129296E+01
-1.107579468129292E+01
-1.107579468129291E+01
-1.107574609356799E+01
-1.107574609356797E+01
-1.107574609356795E+01
-1.107574609356793E+01
-1.107571441013466E+01
-1.107571441013465E+01
```



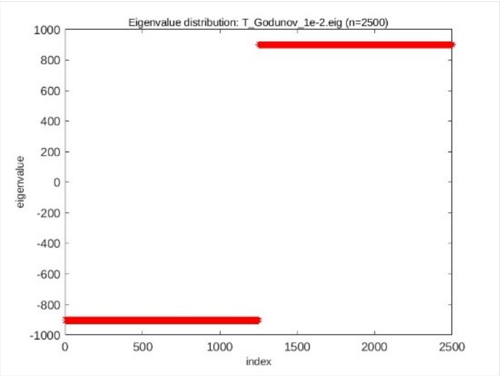
T_Lipshitz_1.eig
n=1000, 20 smallest and largest eigenvalues

```
-2.6823372031320144e-16     1.0000000000000020e+00
-2.2237154507547389e-16     1.0000000000000022e+00
-2.2204461419491012e-16     1.0000000000000022e+00
-2.2204460492503131e-16     1.0000000000000022e+00
-1.8342609148280882e-16     1.0000000000000022e+00
-1.3899996366291143e-16     1.0000000000000024e+00
-1.1135703515315642e-16     1.0000000000000024e+00
-5.5511152667964514e-17     1.0000000000000024e+00
-4.9656459539928424e-17     1.0000000000000024e+00
-3.6224205091068070e-17     1.0000000000000024e+00
-3.4769060438593381e-17     1.0000000000000024e+00
-2.9457359363955050e-17     1.0000000000000024e+00
-1.8099200983560987e-17     1.0000000000000024e+00
-9.9708593735412647e-18     1.0000000000000029e+00
-4.4639982898565932e-18     1.0000000000000029e+00
-4.1246352014896293e-18     1.0000000000000029e+00
-3.9284577008713904e-18     1.0000000000000029e+00
-3.6526711419576054e-18     1.0000000000000031e+00
-2.2491630371466091e-18     1.0000000000000031e+00
-1.8965391276166353e-18     1.0000000000000033e+00
```

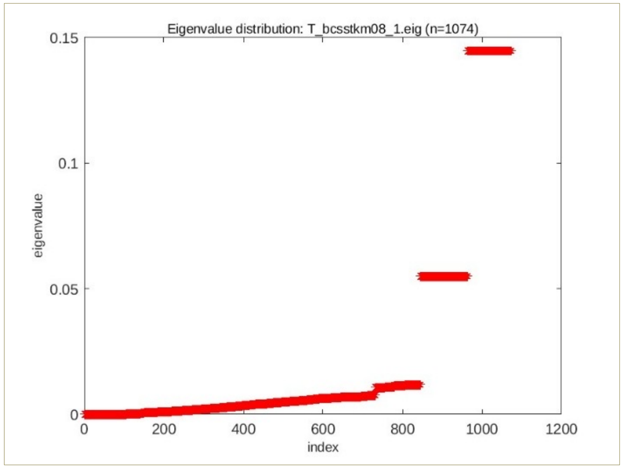


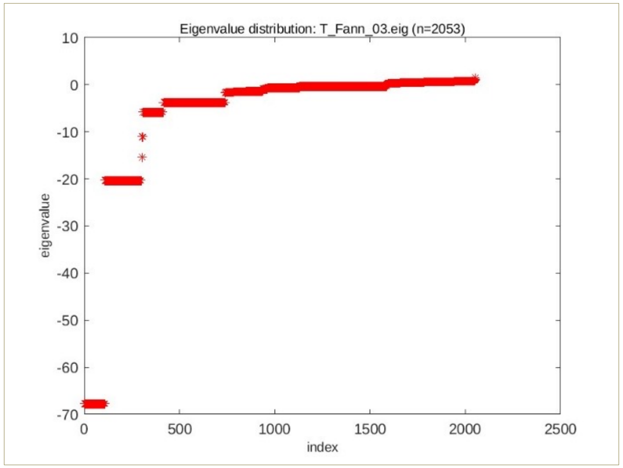
T_Godunov_1e-2.eig
n=2500, 20 smallest and largest eigenvalues

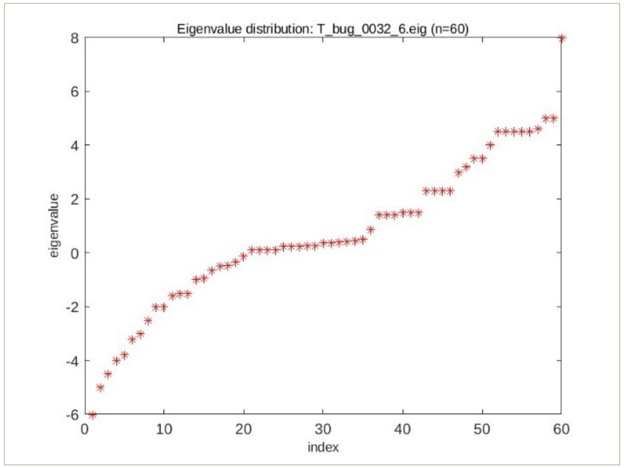
```
-9.000099999684682E+02     9.000099873898861E+02
-9.000099998738725E+02     9.000099886191389E+02
-9.000099997162141E+02     9.000099897853999E+02
-9.000099994954936E+02     9.000099908886612E+02
-9.000099992117126E+02     9.000099919289162E+02
-9.000099988648727E+02     9.000099929061586E+02
-9.000099984549761E+02     9.000099938203817E+02
-9.000099979820255E+02     9.000099946715800E+02
-9.000099974460236E+02     9.000099954597482E+02
-9.000099968469744E+02     9.000099961848813E+02
-9.000099961848813E+02     9.000099968469744E+02
-9.000099954597482E+02     9.000099974460236E+02
-9.000099946715800E+02     9.000099979820255E+02
-9.000099938203817E+02     9.000099984549761E+02
-9.000099929061586E+02     9.000099988648727E+02
-9.000099919289162E+02     9.000099992117126E+02
-9.000099908886612E+02     9.000099994954936E+02
-9.000099897853999E+02     9.000099997162141E+02
-9.000099886191389E+02     9.000099998738725E+02
-9.000099873898861E+02     9.000099999684682E+02
```

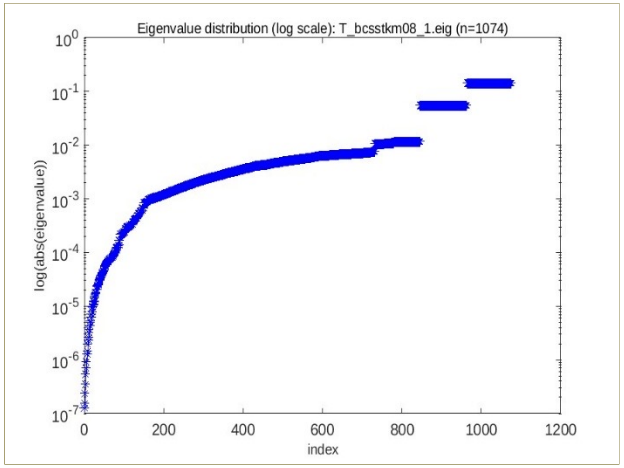


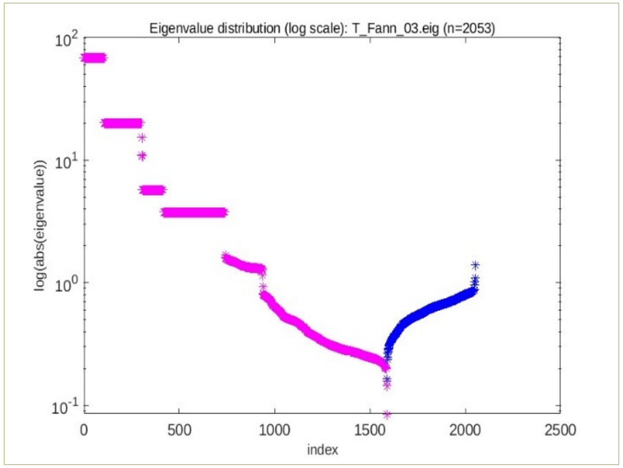
STCollection: plots of eigenvalue distributions

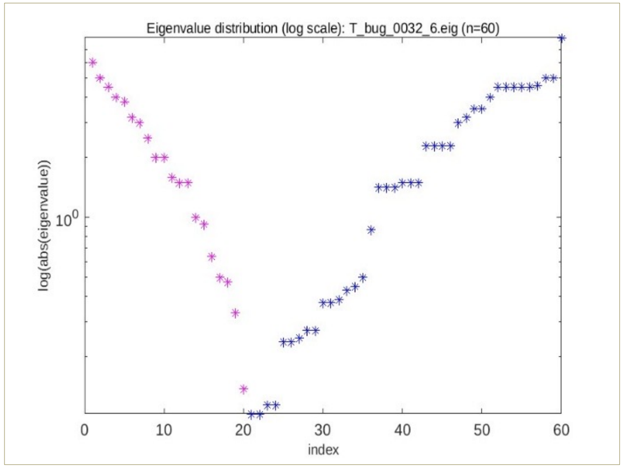

Eigenvalue distribution: T_bcsstkm08_1.eig (n=1074)


Eigenvalue distribution: T_Fann_03.eig (n=2053)


Eigenvalue distribution: T_bug_0032_6.eig (n=60)


Eigenvalue distribution (log scale): T_bcsstkm08_1.eig (n=1074)

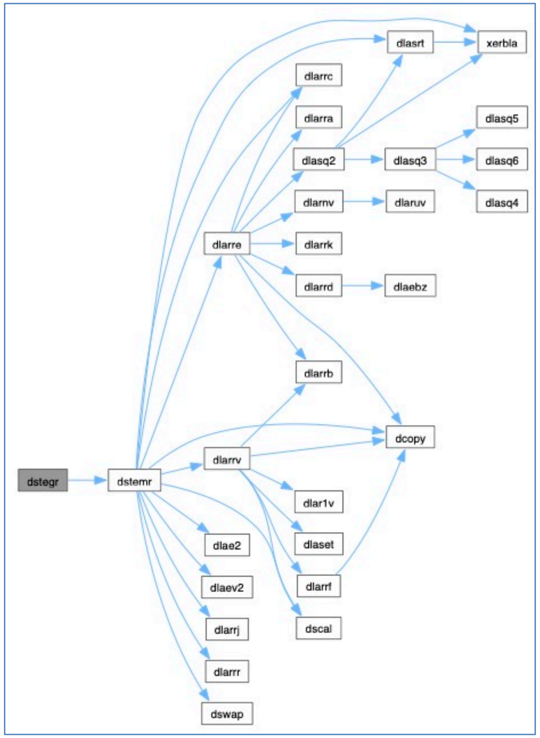

Eigenvalue distribution (log scale): T_Fann_03.eig (n=2053)


Eigenvalue distribution (log scale): T_bug_0032_6.eig (n=60)

Opportunities (w.r.t. software testing)

- Categorize tests
- Streamline the analysis of results (for publications)
- Which algorithm is preferable under certain conditions?
- Integration of meaningful tests in the generation of codes
- Choose tests based of data or plot characteristics
- Test of code segments (akin to unit tests)
- Manage versions (e.g. real/complex) instead of using ”extract” (base files)
- Algorithm selection (next slides)

⋮

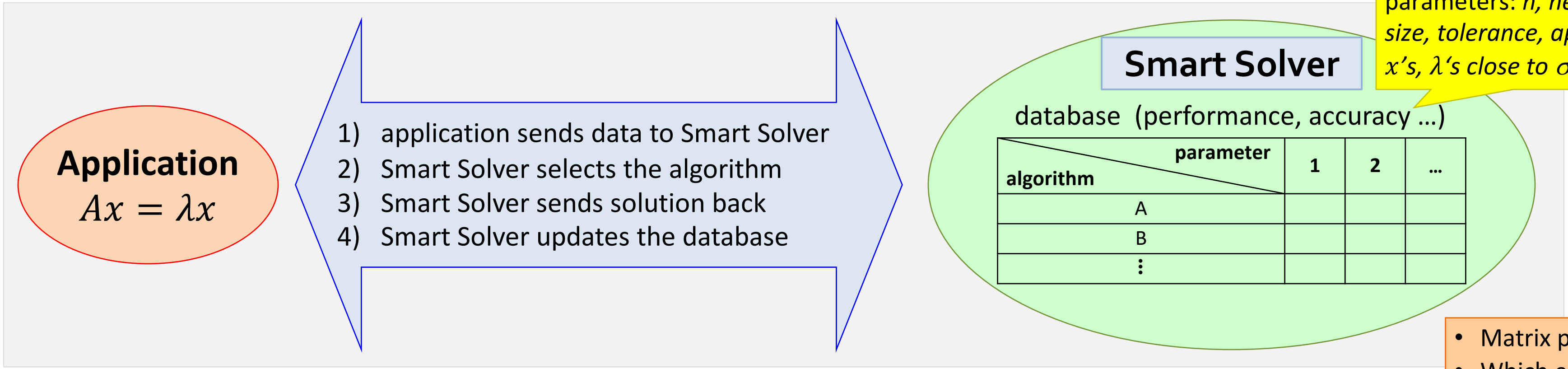


OM: I fed STCollection into https://gitingest.com and obtained as output something that I am not sure it would be much useful for an LLM-based model…

Oba, Tezuka and Hasegawa (2025). Prediction of Iterative Solvers Convergence Using Pretraining by Natural Images. 27th DaWaK.

Algorithm Selection: A Poor Man's AI

20+ years ago 😊



Application

$Ax = \lambda x$

1) application sends data to Smart Solver
2) Smart Solver selects the algorithm
3) Smart Solver sends solution back
4) Smart Solver updates the database

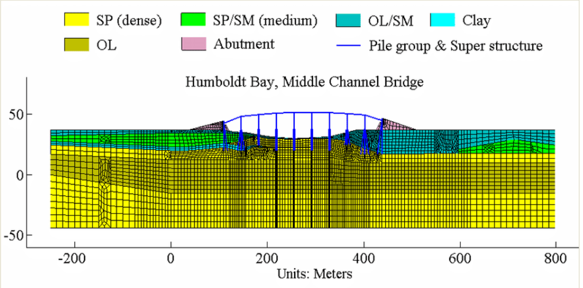
Smart Solver

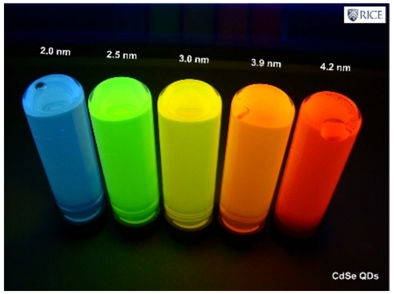
database (performance, accuracy …)

parameter / **algorithm**	**1**	**2**	**…**
A			
B			
⋮			

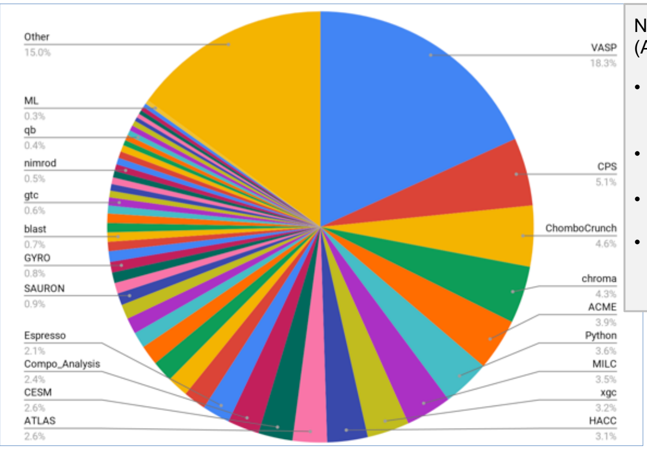
parameters: *n, neig, block size, tolerance, approximate x's, λ's close to σ, restart…*

- Matrix properties?
- Which eigenvalues?
- Eigenvectors?





- ❖ Given a set of parameters:
  - How to select an algorithm?
  - How to update the database?
- ❖ Is the user allowed to bias the selection?
- ❖ How to package the algorithms?
- ❖ What to do in case of failure?
- ❖ What to use as database?



NERSC System Utilization (Aug'17 - Jul'18)

- electronic structure DFT eigenvalue problems ~ 25% of the workload
- 10 codes > 50% of the workload
- 35 codes > 75% of the workload
- Over 600 codes comprise the remaining 25% of the workload

- *Demmel, Li and Marques (2004). An Intelligent and Scalable Sparse Eigensolver Toolkit for High-End Computing. Submitted to NSF.*
- *Kajiyama et al. (2006). SILC: A Flexible and Environment-Independent Interface for Matrix Computation Libraries, LNCS 3911.*
- *Sakurai et al. (2013). A Sparse Matrix Library with Automatic Selection of Iterative Solvers and Preconditioners, Procedia Computer Science 18.*
- *Yu et al. (2018). ELSI: A unified software interface for Kohn–Sham electronic structure solvers. Computer Physics Communications.*

Question to ChatGPT

Given a very large, sparse, real symmetric matrix, for which about 5% of the smallest eigenvalues and corresponding eigenvectors need to be computed, which eigenvalue solvers could be used for that task, considering accuracy and speed ?

Answer (next slides):

- summary of good options
- practical recommendations
- implementation suggestions / libraries
- quick decision tree

Answer: summary of good options

- **Block Lanczos / Krylov-Schur (ARPACK / ARPACK-NG / SLEPc / PRIMME)**
  - **How:** Krylov subspace method with restarting; use block variants when many eigenpairs required.
  - **Pros:** Robust, well-tested, good for relatively small numbers of eigenpairs (hundreds → low thousands). Mature libraries (ARPACK-NG, SLEPc, PRIMME).
  - **Cons:** Orthogonalization cost grows with number of wanted vectors; memory ~ O(n·m) with m = block/subspace size. Can be slow if you need many (tens of thousands) eigenpairs.
- **LOBPCG (Locally Optimal Block Preconditioned Conjugate Gradient)**
  - **How:** Block method that uses a few vectors per iteration and accepts preconditioning.
  - **Pros:** Excellent when a good preconditioner is available (AMG, ILU, domain decomposition). Good scaling to many eigenpairs if block size chosen well. Low communication cost for parallel runs. Implementations: BLOPEX, SLEPc, scipy.sparse.linalg.lobpcg.
  - **Cons:** Needs good preconditioner for speed; still needs reorthogonalization.
- **Davidson / Jacobi-Davidson (block versions)**
  - **How:** Subspace expansion with a correction equation solved approximately (often with preconditioning).
  - **Pros:** Very effective for interior or clustered eigenvalues, and when good preconditioners exist for the correction equation. Good for many physics applications. Implementations: PRIMME (Jacobi-Davidson), SLEPc.
  - **Cons:** Each step may require inner linear solves; implementation complexity higher.
- **Shift-invert strategy (with any Krylov method)**
  - **How:** Transform smallest eigenvalues to largest by solving $(A - \sigma I)^{-1}x$ with σ near target (here $\sigma = 0$). Then run Lanczos/Arnoldi on the transformed operator.
  - **Pros:** Very fast convergence to extreme or interior eigenvalues.
  - **Cons:** Requires solving large sparse linear systems efficiently each iteration — i.e., a fast/robust direct solver or preconditioned iterative solver (multigrid/AMG, ILU) is essential. Memory and factorization cost can be the limiter.
- **Spectrum-slicing / FEAST / Contour integration methods**
  - **How:** Slice the spectral interval containing the wanted eigenvalues into subintervals and solve many independent problems (often via contour integrals or rational filtering).
  - **Pros:** Highly parallel (each slice independent), excellent when you want a *large fraction* of the spectrum (like 5%). FEAST and spectrum-slicing libraries work well for thousands → tens of thousands of eigenpairs.
  - **Cons:** Each slice typically requires solving several linear systems (shifted matrices), so good linear solvers/preconditioners are again required.
- **Polynomial/Chebyshev filtering + subspace iteration (filter-diagonalization)**
  - **How:** Apply polynomial filter that amplifies the wanted part of spectrum, then perform block subspace iterations/orthonormalization.
  - **Pros:** Avoids inner linear solves (cheaper per apply if A is sparse); very suitable for computing many eigenpairs and for GPU/parallel architectures. Good for large fractions of spectrum.
  - **Cons:** Need to design filters (degree vs cost) and handle orthogonalization overhead.
- **Randomized / approximate methods**
  - **How:** Randomized subspace projection plus refinement (useful when only approximate eigenpairs are acceptable).
  - **Pros:** Very fast and memory-friendly for approximations.
  - **Cons:** Not ideal if high accuracy for many smallest eigenpairs is required.

Answer: summary of good options

- **Block Lanczos / Krylov-Schur (ARPACK / ARPACK-NG / SLEPc / PRIMME)**
  - **How:** Krylov subspace method with restarting; use block variants when many eigenpairs required.
  - **Pros:** Robust, well-tested, good for relatively small numbers of eigenpairs (hundreds → low thousands). Mature libraries (ARPACK-NG, SLEPc, PRIMME).
  - **Cons:** Orthogonalization cost grows with number of wanted vectors; memory ~ O(n·m) with m = block/subspace size. Can be slow if you need many (tens of thousands) eigenpairs.
- **LOBPCG (Locally Optimal Block Preconditioned Conjugate Gradient)**
  - **How:** Block method that uses a few vectors per iteration and accepts preconditioning.
  - **Pros:** Excellent when a good preconditioner is available (AMG, ILU, domain decomposition). Good scaling to many eigenpairs if block size chosen well. Low communication cost for parallel runs. Implementations: BLOPEX, SLEPc, scipy.sparse.linalg.lobpcg.
  - **Cons:** Needs good preconditioner for speed; still needs reorthogonalization.
- **Davidson / Jacobi-Davidson (block versions)**
  - **How:** Subspace expansion with a correction equation solved approximately (often with preconditioning).
  - **Pros:** Very effective for interior or clustered eigenvalues, and when good preconditioners exist for the correction equation. Good for many physics applications. Implementations: PRIMME (Jacobi-Davidson), SLEPc.
  - **Cons:** Each step may require inner linear solves; implementation complexity higher.
- **Shift-invert strategy (with any Krylov method)**
  - **How:** Transform smallest eigenvalues to largest by solving $(A - \sigma I)^{-1}x$ with σ near target (here $\sigma\ = 0$). Then run Lanczos/Arnoldi on the transformed operator.
  - **Pros:** Very fast convergence to extreme or interior eigenvalues.
  - **Cons:** Requires solving large sparse linear systems efficiently each iteration — i.e., a fast/robust direct solver or preconditioned iterative solver (multigrid/AMG, ILU) is essential. Memory and factorization cost can be the limiter.
- **Spectrum-slicing / FEAST / Contour integration methods**
  - **How:** Slice the spectral interval containing the wanted eigenvalues into subintervals and solve many independent problems (often via contour integrals or rational filtering).
  - **Pros:** Highly parallel (each slice independent), excellent when you want a *large fraction* of the spectrum (like 5%). FEAST and spectrum-slicing libraries work well for thousands → tens of thousands of eigenpairs.
  - **Cons:** Each slice typically requires solving several linear systems (shifted matrices), so good linear solvers/preconditioners are again required.
- **Polynomial/Chebyshev filtering + subspace iteration (filter-diagonalization)**
  - **How:** Apply polynomial filter that amplifies the wanted part of spectrum, then perform block subspace iterations/orthonormalization.
  - **Pros:** Avoids inner linear solves (cheaper per apply if A is sparse); very suitable for computing many eigenpairs and for GPU/parallel architectures. Good for large fractions of spectrum.
  - **Cons:** Need to design filters (degree vs cost) and handle orthogonalization overhead.
- **Randomized / approximate methods**
  - **How:** Randomized subspace projection plus refinement (useful when only approximate eigenpairs are acceptable).
  - **Pros:** Very fast and memory-friendly for approximations.
  - **Cons:** Not ideal if high accuracy for many smallest eigenpairs is required.

Answer: practical recommendations

- If “5%” corresponds to a modest number (≤ a few thousand eigenpairs) and you have moderate memory:
  - Use block Krylov (ARPACK-NG, PRIMME) or LOBPCG with a good preconditioner.
  - If eigenvalues are tightly clustered or convergence is slow, use Jacobi-Davidson or shift-invert (if you can solve linear systems fast).
- If “5%” is large (many thousands — e.g. n=1e6 → 50k eigenpairs):
  - Prefer spectrum-slicing / FEAST or Chebyshev polynomial filtering + block subspace. These scale better when you need a big chunk of the spectrum and are embarrassingly parallel over slices or frequency intervals.
  - Shift-invert becomes expensive at this scale unless you have an extremely fast parallel direct solver or extremely good preconditioners.
- If you have a powerful preconditioner (AMG, domain-decomp):
  - LOBPCG or Jacobi-Davidson (block) + preconditioning is often the fastest and most memory-efficient route.
- If you need to run on distributed memory (MPI) or GPUs:
  - Use parallel packages: SLEPc (with PETSc), PRIMME (has parallel support), or FEAST with parallel linear solvers. For GPU-focused workflows, look at libraries that support GPU linear solves for the shift-invert inner solves.

Answer: implementation suggestions / libraries

- ARPACK-NG — classic, but single-vector ARPACK struggles when many eigenpairs are needed; use block-enabled variants where available.
- PRIMME — modern, high-performance; supports Davidson/JD/GD and preconditioning; good for many practical cases.
- SLEPc (on top of PETSc) — rich set of methods (Krylov-Schur, LOBPCG, Jacobi-Davidson, spectrum slicing), MPI-ready.
- FEAST — contour integration / spectrum slicing approach.
- BLOPEX / LOBPCG — simple, effective with preconditioning (integrated into SLEPc/Trilinos).
- Trilinos/Anasazi and ScaLAPACK for large parallel workflows.

Answer: implementation suggestions / libraries

- ARPACK-NG — classic, but single-vector ARPACK struggles when many eigenpairs are needed; use block-enabled variants where available.
- PRIMME — modern, high-performance; supports Davidson/JD/GD and preconditioning; good for many practical cases.
- SLEPc (on top of PETSc) — rich set of methods (Krylov-Schur, LOBPCG, Jacobi-Davidson, spectrum slicing), MPI-ready.
- FEAST — contour integration / spectrum slicing approach.
- BLOPEX / LOBPCG — simple, effective with preconditioning (integrated into SLEPc/Trilinos).
- Trilinos/Anasazi and ScaLAPACK for large parallel workflows. ?

Answer: quick decision tree

- Small → moderate k (hundreds): Block Lanczos / PRIMME / ARPACK-NG.
- Moderate → large k and have good preconditioner: LOBPCG or Jacobi-Davidson (block) with preconditioning.
- Large k (5% large) or extreme scale: spectrum slicing / FEAST or Chebyshev filtering + block subspace.
- If fast direct solves available: use shift-invert + Krylov/JD for fastest convergence.

Question to ChatGPT

Are eigenvalue problems solved in AI ?

Yes, **eigenvalue problems** are indeed solved in artificial intelligence (AI) algorithms and large language models (LLMs), though the way they appear may differ from traditional contexts (like solving $Ax = \lambda x$) to more generalized linear algebra and optimization frameworks. Here are the key areas where they arise:

1. **Principal Component Analysis (PCA) — Dimensionality Reduction**
   - **Where:** Used in preprocessing, feature reduction, data visualization, and sometimes in embeddings.
   - **Role:** PCA involves computing the **eigenvalues and eigenvectors** of the covariance matrix of data to find directions (principal components) of maximum variance.
   - **References:** 1) Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer. 2) Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective* (Chapter 12).
2. **Spectral Clustering — Unsupervised Learning**
   - **Where:** Used in clustering tasks where the data has non-convex structures.
   - **Role:** Involves computing the **eigenvectors** of the **Laplacian matrix** derived from the data graph. These eigenvectors are then used for clustering in a lower-dimensional space.
   - **References:** 1) Ng, Jordan, and Weiss (2002). "On spectral clustering: Analysis and an algorithm." NIPS. 2) von Luxburg, U. (2007). "A tutorial on spectral clustering." *Statistics and Computing*.
3. **Word Embeddings & Latent Semantic Analysis (LSA)**
   - **Where:** Early NLP methods and still foundational for understanding representations.
   - **Role: Singular Value Decomposition (SVD)** — a close relative to eigenvalue decomposition — is used in LSA to reduce the dimensionality of term-document matrices.
   - **References:** 1) Deerwester et al. (1990). "Indexing by latent semantic analysis." *JASIS*. 2) Turney & Pantel (2010). "From frequency to meaning: Vector space models of semantics." *JAIR*.
4. **Optimization Problems in Deep Learning**
   - **Where:** In training neural networks, eigenvalues of the **Hessian matrix** (second derivative of the loss) provide insight into curvature and convergence.
   - **Role:** Understanding the spectrum of the Hessian can inform learning rate schedules, saddle point avoidance, and generalization properties.
   - **References:** 1) Sagun, Bottou, & LeCun (2017). "Eigenvalues of the Hessian in deep learning: Singularity and beyond." arXiv:1611.07476. 2) Ghorbani, Krishnan, & Xiao (2019). "An Investigation into Neural Net Optimization via Hessian Eigenvalue Density." ICML.
5. **Transformers and Attention: Low-rank Approximations**
   - **Where:** Efficient transformers often use low-rank matrix approximations to speed up attention computations.
   - **Role:** These approximations can involve **SVD or eigen decompositions** to approximate attention matrices (e.g., in Linformer, Performer).
   - **References:** 1) Wang et al. (2020). "Linformer: Self-Attention with Linear Complexity." arXiv:2006.04768. 2) Choromanski et al. (2021). "Rethinking attention with performers." ICLR.
6. **Graph Neural Networks (GNNs)**
   - **Where:** Applied in recommendation systems, molecule modeling, knowledge graphs.
   - **Role:** Many GNNs rely on **spectral graph theory**, where eigenvalues of the graph Laplacian define convolutional operations.
   - **References:** 1) Kipf & Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks." ICLR. 2) Defferrard et al. (2016). "Convolutional neural networks on graphs with fast localized spectral filtering."

Question to ChatGPT

Are eigenvalue problems solved in AI ?

Yes, **eigenvalue problems** are indeed solved in artificial intelligence (AI) algorithms and large language models (LLMs), though the way they appear may differ from traditional contexts (like solving $Ax = \lambda x$) to more generalized linear algebra and optimization frameworks. Here are the key areas where they arise:

1. **Principal Component Analysis (PCA) — Dimensionality Reduction**
   - **Where:** Used in preprocessing, feature reduction, data visualization, and sometimes in embeddings.
   - **Role:** PCA involves computing the **eigenvalues and eigenvectors** of the covariance matrix of data to find directions (principal components) of maximum variance.
   - **References:** 1) Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer. 2) Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective* (Chapter 12).

Application Area	Eigenvalue Connection	Used In
PCA	Covariance matrix eigendecomposition	Feature reduction, NLP
Spectral Clustering	Graph Laplacian eigenvectors	Clustering, vision
Latent Semantic Analysis	SVD (related to eigenvalue problems)	NLP
Optimization in Deep Nets	Hessian eigenvalues	Training diagnostics
Efficient Transformers	Low-rank matrix approximations (SVD)	Attention speed-up
Graph Neural Networks (GNNs)	Spectral filters using eigenvalues of Laplacian	Graph data modeling

Investigation into Neural Net Optimization via Hessian Eigenvalue Density." ICML.

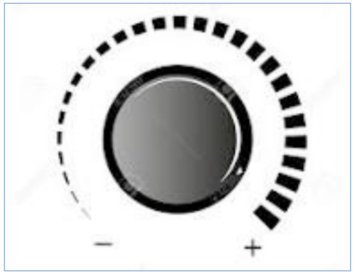
5. **Transformers and Attention: Low-rank Approximations**
   - **Where:** Efficient transformers often use low-rank matrix approximations to speed up attention computations.
   - **Role:** These approximations can involve **SVD or eigen decompositions** to approximate attention matrices (e.g., in Linformer, Performer).
   - **References:** 1) Wang et al. (2020). "Linformer: Self-Attention with Linear Complexity." arXiv:2006.04768. 2) Choromanski et al. (2021). "Rethinking attention with performers." ICLR.
6. **Graph Neural Networks (GNNs)**
   - **Where:** Applied in recommendation systems, molecule modeling, knowledge graphs.
   - **Role:** Many GNNs rely on **spectral graph theory**, where eigenvalues of the graph Laplacian define convolutional operations.
   - **References:** 1) Kipf & Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks." ICLR. 2) Defferrard et al. (2016). "Convolutional neural networks on graphs with fast localized spectral filtering."

Summary – AI & HPC Synergies (at a glance)

- Within reach
  - Summary and generation of specific education/training material, help with the selection of education/training topics, etc.
  - Support for code translation and generation of specific kernels
  - Support for algorithm selection (with some fine-tuning)
- Wish list
  - Code generation integrated with performance models
  - Support for interpretation of results (publications)
  - Generation/selection of tests/benchmarking
  - Support for debugging

Summary – AI & HPC Synergies (at a glance)

- Within reach
  - Summary and generation of specific education/training material, help with the selection of education/training topics, etc.
  - Support for code translation and generation of specific kernels
  - Support for algorithm selection (with some fine-tuning)
- Wish list
  - Code generation integrated with performance models
  - Support for interpretation of results (publications)
  - Generation/selection of tests/benchmarking
  - Support for debugging



Andrej Karpathy's talk at AI Startup School (Y Combinator):

- *keep AI on a leash (autonomy slider)*
- *generation and verification*

Thank you !