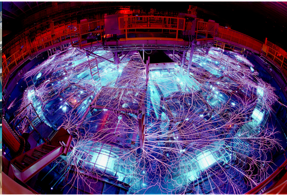
Exceptional service in the national interest

Sandia National Laboratories

Linear Algebra and Computing: Behind the Scenes

Jennifer Loe

October 26, 2022

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.
SAND NO. 2022-15012 PE

About Me:

- Bachelor’s degree from Oklahoma Christian University
- Math major with minors in computer science and international studies (After some switching!)
- Masters and Ph.D. in Mathematics from Baylor University
- Now Staff at Sandia National Laboratories: Scalable Algorithms Department in the Center for Computing Research (`https://www.sandia.gov/ccr/`)
- I study numerical analysis (how computers do math)
- Focus on numerical linear algebra (how computers solve linear systems and find eigenvalues)
- I specialize in Krylov subspace methods and am a developer of the Belos linear solvers package in Trilinos.
- I’ve also done research in mixed precision algorithms and am starting to look at quantum computing!

The National Labs

- 17 U.S. Department of Energy National Labs
- 10 Office of Science Labs:
  including Argonne, Berkeley, and Oak Ridge
- 3 National Nuclear Security Agency (NNSA) Labs:
  Sandia, Lawrence Livermore, Los Alamos

`https://www.energy.gov/national-laboratories`

About Sandia National Laboratories

- Campuses in Albuquerque, NM and Livermore, CA
- About $15{,}000$ employees (as of 2020)
- Origins in the Manhattan project
- `www.sandia.gov`

Nuclear Weapons Research

- Stockpile stewardship: safe, secure, and reliable to support nuclear deterrence.
- "Always and Never"
- Develop non-nuclear weapons components.
- Extreme Environments Testing

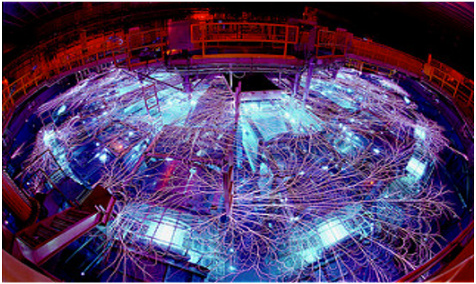


Figure: https://www.sandia.gov/z-machine/ Z Machine allows scientists to study materials under conditions similar to those produced by the detonation of a nuclear weapon, and it produces key data used to validate physics models in computer simulations.

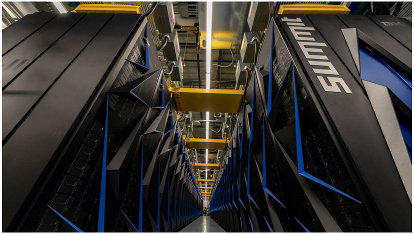
Other Research and National Security Projects:

- Biological and chemical threat reduction
- COVID-19 research and antiviral countermeasures
- Remote sensing
- Counterterrorism
- Cybersecurity
- Microelectronics
- See virtual tours at `https://tours.sandia.gov/tours.html`.

Figure: (Screenshot from MESA facilities virtual tour.)

Center for Computing Research:

- High Performance computing - Software and Hardware Architectures
- Scalable Software- Trilinos, Kokkos, Physics Modeling libraries
- Neuromorphic computing
- Quantum computing
- Data Science and Machine Learning; Artificial Intelligence



Summit supercomputer at Oak Ridge Leadership Computing Facility. 4th fastest in the world as of June 2022.

Exascale Computing Project

- First Exascale computer Frontier launched in 2022 at Oak Ridge Leadership Computing Facility. Aurora (Argonne National Lab) on the way!
- Fastest computers previously: Petascale- a quadrillion (10^{15}) operations per second (plus some)
- Now: Exascale- a quintillion (10^{18}) operations per second
- Writing new software that will scale on new supercomputers and take advantage of GPUs (highly parallel!)
- `https://www.top500.org/`

What you'll see in this talk:

1. An real-life example that requires solving a large linear system.
2. Three reasons that traditional method of solving linear systems don't work for large-scale applications.
3. A brief glimpse of Krylov solvers for linear systems.

Now, what is a linear system again??

Example

$$\begin{cases} 2x_1 + \ \ 2x_2 + \ \ 2x_3 = 12 \\ 4x_1 + \ \ 7x_2 + \ \ 7x_3 = 24 \\ 6x_1 + 18x_2 + 22x_3 = 12 \end{cases}$$

$$A = \begin{pmatrix} 2 & 2 & 2 \\ 4 & 7 & 7 \\ 6 & 18 & 22 \end{pmatrix} x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} b = \begin{pmatrix} 12 \\ 24 \\ 12 \end{pmatrix}$$

Need to solve

$$Ax = b.$$

(For this talk only, assume a unique solution exists.)

Example: ExaWind Application

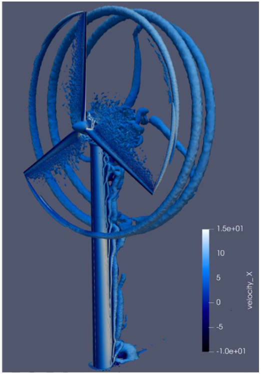


Image credit: Domino, Barone, & Bruner, 2018

Nalu-Wind simulates fluid dynamics of one (or many) wind turbines.

- Linear systems are very large (*n* $> 1,000,000$) [This problem has *n* $= 95$ million!] and very sparse (mostly zeros!)

Why solving large, sparse Ax=b is hard:

Big Ideas in Numerical Analysis:

- Computational Cost
- Memory Requirements
- Conditioning and Stability

First attempts at solving $Ax = b$?

If you're a pure mathematician, it's easy!

- Calculate A^{-1}.
- Multiply on both sides: $A^{-1}Ax = A^{-1}b$
- Then $x = A^{-1}b$. Ta-Da!

In real life: (A is large and sparse)

- Never, ever, ever compute and store A^{-1}!!
  (Too computationally expensive! Too much memory!)

Counting the cost:

Approximate number of floating-point operations $(+, -, \cdot, \div)$ to solve $Ax = b$ for A_{nxn}:

- Computing A^{-1}: $2n^3$ operations
- Gauss-Jordan elimination: n^3 operations
- *LU* factorization with back solve: $\frac{2}{3}n^3$ operations

But this is still too expensive for very large matrices!

Why solving large, sparse Ax=b is hard:

Big Ideas in Numerical Analysis:

- Computational Cost
- Memory Requirements
- Conditioning and Stability

Sparse vs Dense Storage:

Dense Storage: Store the value for every element in a matrix.
Sparse Storage: Store only the nonzero values of a matrix and their locations. (Look up: Compressed Sparse Row (CSR) format)

Small example from computational fluid dynamics: ”AF23560”
n = 23,560
Number of nonzeros: 460,598 (About .08% nonzero.)

Sparse storage: 7.56 MB
Full storage: 4.44 GB

Full matrix storage $\Longrightarrow$ Out of Memory Errors
(Yes, even on a supercomputer!)

CAUTION: A^{-1} may be dense even when A is sparse.

What about LU factorization?

Great for dense A!
No extra storage needed! (Store L and U in place of A.)

But if A is sparse, its LU factorization may not be...

(Previous small ex: A has 460,598 nonzeros.
Its LU factorization has over **13 million more nonzero elements** than A! Over 26 times more storage than sparse A!)

Conclusion: LU factorization is great for small problems, and sometimes okay for small sparse matrices...

But the storage costs are still too expensive for sparse matrices!

Why solving large, sparse Ax=b is hard:

Big Ideas in Numerical Analysis:

- Computational Cost
- Memory Requirements
- Conditioning and Stability

An ill-conditioned problem:

$$A = \begin{pmatrix} .835 & .667 \\ .333 & .266 \end{pmatrix} x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} b = \begin{pmatrix} .168 \\ .067 \end{pmatrix}$$

The true solution is:

$$x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

One small change...

$$A = \begin{pmatrix} .835 & .667 \\ .333 & .266 \end{pmatrix} x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} b = \begin{pmatrix} .168 \\ .066 \end{pmatrix}$$

The true solution is:

$$\cancel{x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}} \qquad x = \begin{pmatrix} -666 \\ 834 \end{pmatrix}$$

Definition

A problem is said to be **ill-conditioned** when small changes in the input create large changes in the output.

Ill-Conditioned: My Mental Picture

The higher the condition number of a matrix, the more ill-conditioned it is. (Small changes have bigger effect.)

The older and “more conditioned” a piece of furniture is (more wear-and-tear), the more likely that one wrong move makes it fall apart!

(Left: Photo by William Warby on Unsplash. Right: Image by Amy Moore from Pixabay.)

Stability: Is the algorithm making things worse?

Definition

The **stability** of an algorithm refers to the likelihood that the algorithm induces round-off error.

Examples:

- $x + y - x$ is not a very stable algorithm to get y.
- Modified Gram-Schmidt is more stable than Gram-Schmidt.
- Add pivoting to *LU* factorization for stability.

This is separate from conditioning.

Why solving large, sparse Ax=b is hard:

Big Ideas in Numerical Analysis:

- Computational Cost
- Memory Requirements
- Conditioning and Stability

So how **CAN** we solve a large, sparse linear system??

Solution: Krylov Methods

Krylov methods are a specific type of **iterative methods.**

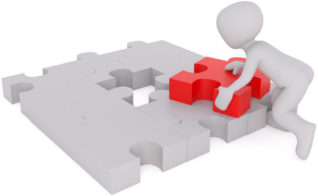
<u>Big idea of Krylov methods:</u>
Narrow down the search area!
Start with an initial guess; then keep moving closer and closer until you get close enough to the real solution.

(Think about getting ”hotter” or ”colder” in I Spy.)

The idea behind Krylov Methods:

To solve $Ax = b$, where A is $n \times n$:

1. Pick a good, small **subspace** of $\mathbb{R}^n$ in which to search for a solution.
2. Find the "best" solution in that subspace via a **projection**.



(Left: Image by Peggy Marco from Pixabay. Right: Photo by Martino Pietropoli on Unsplash.)

The idea behind Krylov Methods:

To solve $Ax = b$, where A is $n \times n$:

1. Build an orthonormal basis for a Krylov subspace:

$$\text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

2. Use an orthogonal projection to find an approximate solution $\hat{x}$ which minimizes the residual: $\|b - A\hat{x}\|_2$ (Specific to method GMRES.)

Many flavors of Krylov methods depending on your matrix! (symmetric, non-symmetric, eigenvalue distribution, etc.)

GMRES: Generalized Minimum RESidual method

Sandia National Laboratories

Algorithm GMRES (Modified Gram-Schmidt)

$\gamma = \|b\|_2$ and $v_1 = b/\gamma$
for $j = 1 : m$ **do**
  $w_j = Av_j$
  for $i = 1 : j$ **do**
    $h_{ij} = v_i^T w_j$
    $w_j = w_j - h_{ij} v_i$
  end for
  $h_{j+1,j} = \|w_j\|_2.$
  $v_{j+1} = w_j / h_{j+1,j}$
end for
Define the $(m+1) \times m$ matrix $\overline{H} = \{h_{ij}\}$
Solve least-squares problem $\overline{H}d = \gamma e_1$ for d.
$\hat{x} = V_m d$

Sparse Matrix-Vector Product (SpMV)

Orthogonalize next basis vector in Krylov subspace

Project for best solution in the space

Key Takeaways:

- National labs do research in areas important to national security.
- Lots of physics applications solve HUGE spares linear systems!
- In numerical linear algebra, we must consider computational cost, computer memory limitations, conditioning of the problem, and stability of algorithms.
- Krylov methods like GMRES provide an approximate solution to a large linear system at a fraction of the cost and memory required for a direct (full) solve.

Want to learn more?

- Take a course in Numerical Methods or Numerical Linear Algebra.
- Learn C++.
- Linux/Unix command line (Udacity- Linux command line basics.)
- Git/Github (Many free courses on this!)
- Using a command line text editor (Vim or emacs) (See VimAdventures.)
- Working in an existing code base (Google summer of code)
- Basics of MPI and OpenMP (Virtual HPC workshops from XSEDE)
- Kokkos Tutorials (if you know some C++): `https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series`
- Other good resources: Argonne Training Program on Extreme-Scale Computing (Link: ANL_Training_YouTube)

Internships at Sandia Center for Computing Research

- Internships available for undergrad and graduate students.
- Opportunity for year-round internship after successful summer internship.
- Applications are competitive due to limited space.
- Apply NOW. `https://sandia.jobs`
  Search "CSRI" and/or "TITANS."
- Finally back in person! Albuquerque, NM and Livermore, CA locations.
- Knowledge of C++, parallel programming, and Kokkos library is a plus!
- Internships in other departments for physics, engineering, chemistry, etc.
- Contact me to learn more!

Thank you!

Sandia National Laboratories

Jennifer Loe
jloe@sandia.gov

EXASCALE COMPUTING PROJECT