CAS 741 (Development of Scientific Computing Software)

Winter 2025

Mixed-Precision Iterative Solver

Xunzhou (Joe) Ye

Faculty of Engineering, McMaster University

January 28, 2025

McMaster University

Linear Solver: the Good Old $\mathbf{Ax} = \mathbf{b}$ Problem

- Direct method: Gaussian elimination

$$\mathbf{Ax} = \begin{bmatrix} 1 & -1 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \\ 3 \end{bmatrix} = \mathbf{b}$$

$$\mathbf{A}|\mathbf{b} = \left[\begin{array}{ccc|c} 1 & -1 & 3 & 11 \\ 1 & 1 & 0 & 3 \\ 3 & -2 & 1 & 3 \end{array}\right] \begin{array}{cc} \times 1 & \times 3 \\ \downarrow & \\ & \downarrow \end{array}$$

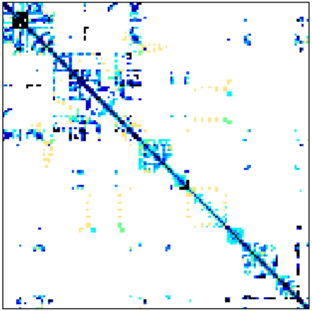
$$\mathbf{A}|\mathbf{b} \leftarrow \left[\begin{array}{ccc|c} 1 & -1 & 3 & 11 \\ 0 & 2 & -3 & -8 \\ 0 & 1 & -8 & -30 \end{array}\right]$$

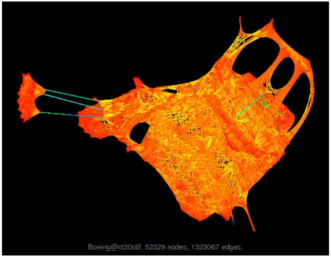
Think Big, Think Sparse

- In structural simulations (e.g., finite element analysis), structures are modeled based on real-world physics, where forces and constraints are often localized.
- Each element or node in a structure is typically connected only to a few nearby elements or nodes.
- System of equations in structural simulations is assembled from local contributions.
- In contrast, in neural networks, dense weight matrices are used in fully connected layers.

Sparse Matrix Example: Boeing/ct20stif

- $52\,329 \times 52\,329$
- $2\,600\,295$ non-zeros
- $\approx 33\,\text{MB}$ in file size

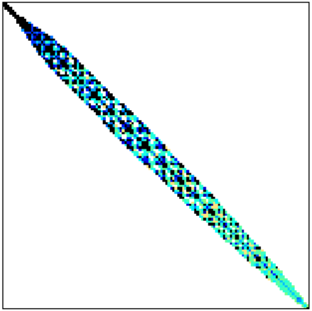




Boeing/ct20stif: CT20 Engine Block – Stiffness matrix

Sparse Matrix Example: Janna/Serena

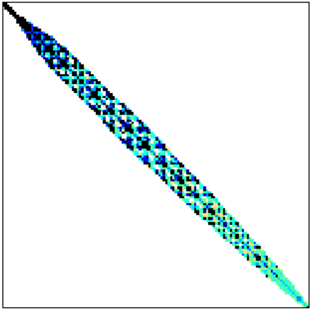
- $1\,391\,349 \times 1\,391\,349$
- $64\,131\,971$ non-zeros
- $\approx 847\,\text{MB}$ in file size



Janna/Serena: gas resevoir simulation for CO_2 sequestration

The Problem with Gaussian Elimination

- Store the whole matrix
- Memory bounded
- Sparsity preservation



Janna/Serena: gas resevoir simulation for CO_2 sequestration

Iterative Refinement

- A method to enhance the accuracy of a solution to $\mathbf{Ax} = \mathbf{b}$

The algorithm goes:

1. Solve $\mathbf{Ax}_0 = \mathbf{b}$ *approximately* to get an initial solution $\mathbf{x}_0$.
2. Compute the residual $\mathbf{r} = \mathbf{b} - \mathbf{Ax}_0$, which measures how far $\mathbf{x}_0$ is from being an exact solution.
3. Solve $\mathbf{Ae} = \mathbf{r}$ for the correction vector $\mathbf{e}$.
4. Update ("refine") the solution as $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{e}$.
5. Repeat steps 2–4 until the residual $\mathbf{r}$ is small enough.

Iterative Refinement in Mixed-Precision

1. Solve $\mathbf{A}\mathbf{x}_0 = \mathbf{b}$ *approximately* in **low** precision.
2. Compute the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ in **high** precision.
3. Solve $\mathbf{A}\mathbf{e} = \mathbf{r}$ for the correction vector $\mathbf{e}$ in **low** precision.
4. Update the solution as $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{e}$ in **medium** precision.
5. Repeat steps 2–4 until the residual $\mathbf{r}$ is small enough.

Key points:

- Trade computational complexity for space complexity
- Preserve sparsity by carefully selecting algorithms: sparse LU factorization, General Minimal Residual Method (GMRES)
- Inner solves could also be iterative and in mixed-precision
- Possible matrix-free implementation

Goals

GS1 Given some matrix $\mathbf{A}$ and column vector $\mathbf{b}$, the solver should iteratively find $\mathbf{x}$ satisfying the equation $\mathbf{Ax} = \mathbf{b}$ until the norm of the residual $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$ is smaller than some tolerance ϵ, or the maximum number of iterations n_{iter} is exhausted, whichever comes first.

GS2 Given some combinations of floating point precision configuration, the solver should perform internal steps such as matrix factorizations, triangular solves, and residual computation in these configured precisions.

Goals (Non-functional)

- GS3 The solver should offer a quantifiable performance or resource utilization advantage over other competing sparse linear solvers.
- GS4 The library should offer a set of streamlined public application programming interfaces (APIs), such that when integrated into other software as a dependent library, the interfaces are self-contained, readable and easy to consume.

Stretch Goals

GS5 With the existing solver implementation being the baseline, the refactored solver should produce more accurate results, lowering the norm of the residual by at least 1 order of magnitude.

GS6 The solver should optimize existing algorithms such that given the same set of inputs, it produces results with the same accuracy in notably less time.

Inputs

Variable	Description
$\mathbf{A}$	$n \times n$ matrix
$\mathbf{b}$	n-vector
ϵ	a solution is found if the norm of the residual is less than ϵ
n_{iter}	the maximum number of iterations to perform
u_f	factorization precision
u_w	working precision
u_r	precision in which the residuals are computed

Outputs

Variable	Description
$\mathbf{x}$	a numerical solution to the linear system
err	the norm of the residual

Assumptions

A1 Matrix $\mathbf{A}$ is quasi-definite.

A2 The precisions follow the order $u_f \leq u_w \leq u_r$, with u_r being the highest precision.

Questions