Solving a system via QR Decomposition On NVIDIA GPUs provided Armadillo's sparse matrices (sp_mat)

John C Crow 14DEC2021

Problem Description

- Create a highly sparse matrix with random values utilizing Armadillo’s sp_mat class.
- Factorize and solve the matrix on a GPU utilizing cusolver.
- Compare the results with a sequential factorization.

Create a Highly Sparse Matrix utilizing Armadillo's 'sp_mat' Class

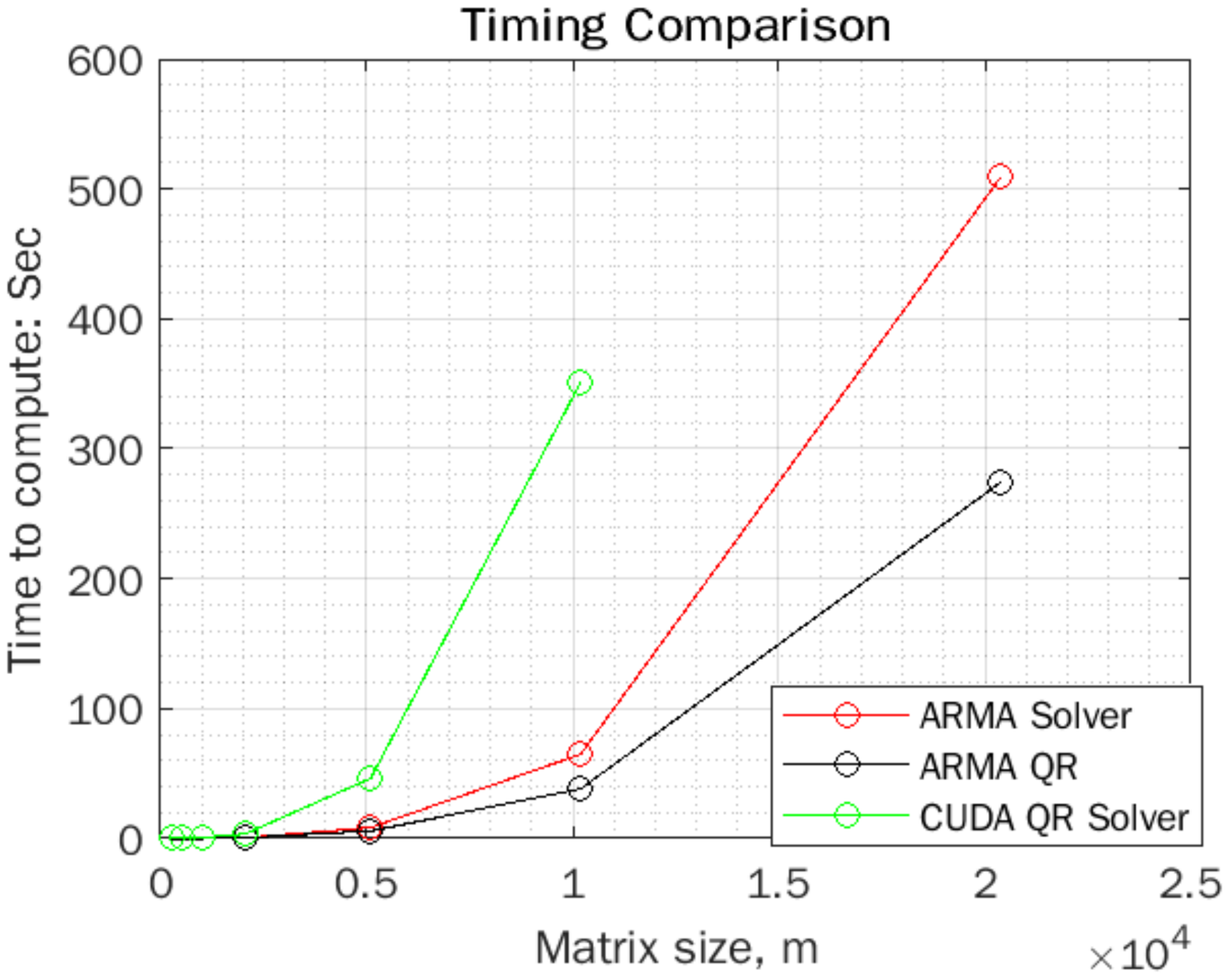
Easiest Part

- int size = 512;
- sp_mat A = sprandn<sp_mat>(size, size, 0.03);
- vec b(size, fill::randu);
  - For Ax=b

Factorize the matrix on a GPU utilizing cusolver.

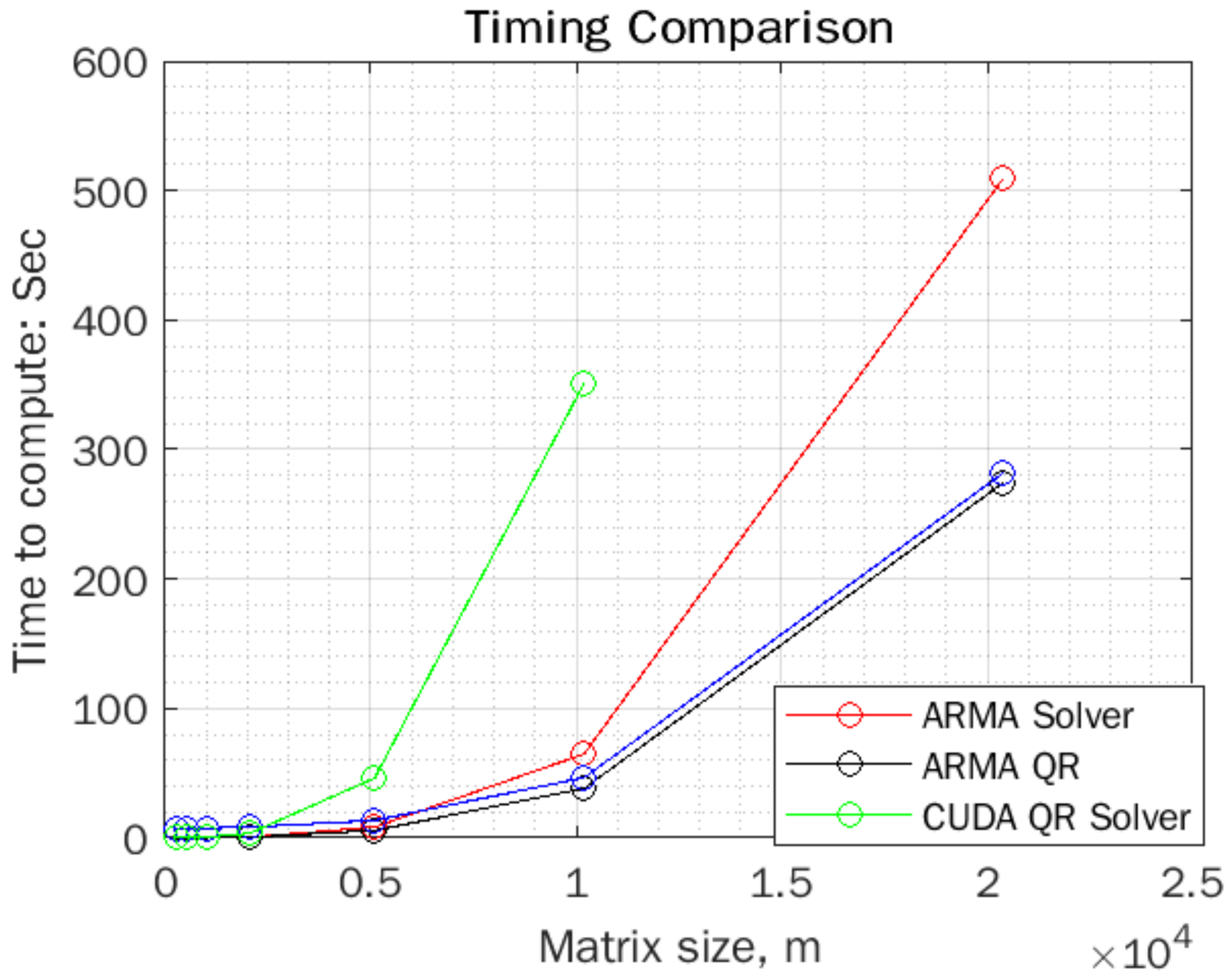
- Non-Trivial
    - Will go into greater depth

Compare the results with a sequential factorization.



NVIDIA
CUDA

Compare the results with a sequential factorization.



NVIDIA
CUDA

Factorize the matrix on a GPU utilizing cusolver.

Problems

- There is no public documentation for how Armadillo stores the sp_mat data besides that it is in CSC format.
- Armadillo stores it’s Sparse Matrices in CSC format while CUSOLVER requires CSR format.
- CUDA cusolverLU is only supported for host side computation and does not have GPU support.
    - Solving a sparse system in parallel is not a trivial problem.

Factorize the matrix on a GPU utilizing cusolver.

Problems

- It is not trivial to link and combine Armadillo Code and CUDA code

- Cusolver and cusparse libraries have poor documentation, and at least one inconsistency.

Sparse Matrix Storage

CSC vs CSR

- Because so many values in a sparse matrix are 0. We do not need to store them all.
- 2 common “compression” formats for sparse matrices

Sparse Matrix Storage

CSR **Ref[1]**

- V = [10 20 30 40 50 60 70 80]
- COL_INDEX = [0 1 1 3 2 3 4 5]
- ROW_INDEX = [0 2 4 7 8]

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

NVIDIA CUDA

Sparse Matrix Storage

CSC

- V = [10 20 30 40 50 60 70 80]
- COL_INDEX = [0 1 3 4 6 7 8]
- ROW_INDEX = [0 0 1 2 1 2 2 3]

$$\begin{pmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{pmatrix}$$

NVIDIA CUDA

Factorize the matrix on a GPU utilizing cusolver.

Problems

- There is no public documentation for how Armadillo stores the sp_mat data besides that it is in CSC format.
- Armadillo stores it’s Sparse Matrices in CSC format while CUSOLVER requires CSR format.
- CUDA cusolverLU is only supported for host side computation and does not have GPU support.
    - Solving a sparse system in parallel is not a trivial problem.

Factorize the matrix on a GPU utilizing cusolver.

Problems

- It is not trivial to link and combine Armadillo Code and CUDA code

- Cusolver and cusparse libraries have poor documentation, and at least one inconsistency.

Credit

- Special thanks to PHD Candidate Jared Brzenski for emotional support and details on problem scope.

References

- [1] https://en.wikipedia.org/wiki/Sparse_matrix#Compressed_sparse_row_(CSR,_CRS_or_Yale_format)