Possible Class Projects for Ma221, Fall 2024

Preproposals are due (on Gradescope) by Oct 15. There should be one proposal per team. Teams should be mostly of size 2 (the number of enrolled students is even, so no obstacle there!) but larger ones can be considered.

Given the breadth of topics the class covers, and the diverse backgrounds and interests of students in the class (from 11 different programs and majors), we expect and welcome a wide range of project proposals. Below, we list the project titles from the 2022 and 2023 offerings of Ma221 as examples.

Projects can involve

1) covering a topic in class in more depth than we had time (eg describing and proving (more) properties of a (different) algorithm for a given problem),
2) implementing and evaluating the accuracy and/or performance of some algorithms on a well-chosen diverse set of test problems,
3) describing the design space of algorithms relevant to a particular application you care about, and comparing the accuracy and/or performance of different algorithms in the design space,
4) reviewing one or more papers that go into more detail on a relevant topic of interest to you, and finally
5) implementing and analyzing some algorithms suggested by the instructor, that are relevant to an ongoing research project.

Prof. Demmel is part of a team responsible for maintaining and improving the BLAS, LAPACK, and related libraries. There are a variety of on-going projects that Prof. Demmel leads (other team members might suggest projects too, TBD):

1) Dealing with floating point exceptions (eg overflows) in a reliable and consistent way. In a survey of LAPACK users, 67% said consistent exception handling was either important or very important. The Ma221 webpage also has links to reports of rocket and car crashes caused by mishandling exceptions. The report arxiv.org/abs/2207.09281 (shorter version available) describes bugs caused by exceptions in the current implementations, and ways to fix them.
2) “Grading” the BLAS on accuracy. Vendors are exploring a wider variety of ways to implement the BLAS faster, sometimes using lower precision arithmetic (even integer matrix-multiply accelerators) to “simulate” higher precision, or different algorithms (eg Strassen), or combinations. These new (combinations of) algorithms do not generally satisfy the same error bounds from homework questions 1.10 or 1.11. Since the vendor’s source code is generally not available, we need to provide test code that works with the BLAS as a “black box” to reverse engineer the actual algorithms used, and provide reliable error bounds. We have a rough design of how to do this for some BLAS, but there is a lot of work to do.
3) RandLAPACK. We are in the process of designing and implementing a version of LAPACK that uses randomized algorithms, to get answers for large problems much

faster, sometimes with less accuracy. A design document is available at arxiv.org/pdf/2302.11474 (under revision). A large number of routines, and test code, remain to be written.

4) How to reliably use 8-bit floating point arithmetic. Many computer vendors are building 8-bit floating point arithmetic (FP8), including matrix-multiple accelerators, into their hardware, motivated by accelerating machine learning. A new standards committee is meeting to try to agree on a common way for them all to do this, much like the IEEE 754 Floating Point Arithmetic standard discussed in class. It is an open question (also being pursued by the Ma221 grader Sudhanva Kulkarni) as to how useful such low precision can be to solve more general linear algebra problems, either by just implementing most or all of an existing algorithm using FP8, or using a different algorithm.
5) Cheaply updating the QR decomposition of A after a low rank change to A. Analogously to homework question 2.13, which describes a cheap way to update the inverse of a matrix A after a low-rank change to A, it is also possible to cheaply update other factorizations of A, like QR. We have a project to add such functionality to LAPACK, which could be expanded to include other factorizations.
6) The CUR decomposition of a matrix A. This is meant to be an alternative to the SVD for computing a low-rank approximation of a matrix A. The difference is that C is a subset of columns of A, R is a subset of rows, and U is a small matrix that combines C and R. This factorization is popular in data science, because using C and R instead of singular vectors, which are linear combinations of all columns and all rows of A, makes the factorization easier to interpret. We also have a project to add this functionality (and variations) to LAPACK.

Finally, here are some titles of projects from two previous offerings of Ma221:

- Matrix completion algorithms with applications to financial time series forecasting
- Randomized Numerical Linear Algebra for Machine Learning
- Rational Krylov Method for Eigenvalue Problem
- Tuning Doubly Randomized Block Kaczmarz Method
- Three MP2 Energy Solvers in the Localized Orbital Representation
- Randomized Preconditioned Least Squares Solvers
- Solution of saddle point systems arising in constrained optimal control
- Integrative Non-Negative Matrix Factorization and Acceleration
- Application of multigrid methods to constrained linear systems
- Fast Solvers for the Finite Element Method
- Speeding up Log Determinants
- Eigenvalues using a neurodynamic approach
- The Effect of Prefetching on Matrix Multiplication
- On the exact solution and numerical solution of the Sylvester equation
- Cache reuse in Multiple Layer Convolutional Neural Network
- Fast solutions to Solving Newton's Method in Nonlinear Finite Elements for Solids
- Sparse + Low Rank Decomposition

- Recommender Systems and the SVD
- Implementing GELS for Linear Least Square Problem in TLAPACK
- Incremental Cholesky Preconditioners for Iterative Methods for SLAM Backend
- Matrix Gallery Extension for testing TLAPACK
- A study on the L-curve model and its applications for Tikhonov Regularization
- An investigation and Extension of Hager's Condition Estimator
- Implementing some routines in the TLAPACK Library
- Matrix Sketching Algorithms and their application to high-dimensional optimization
- Pivoted Cholesky Decomposition: Towards Unbiased Construction of Core Focused RI Basis for ab initio Molecular NMR Calculations
- Theoretical and Empirical Analysis of Anisotropic Poisson's Equation
- Numerical Methods for Option Pricing
- Randomly Pivoted Cholesky: Analysis and Applications
- The Wilson Dirac Operator, the Shwinger Model, and Sparse Linear Solvers
- Finite Element Model Updating Using Randomized Linear Algebra
- An Empirical Investigation of Sequential Quadratic Programming Methods and Hessian-Sketching and Hessian-Subsampling
- Sparse Matrix Factorization in Structural Analysis
- Mesh Reordering for Incomplete Cholesky Preconditioning
- Enhancing Sparse Matrix Operation Efficiency with CSR in Machine Learning
- Rank and Sensitivity Study for LLM Compression
- Extending TLAPACK: Implementation of Symmetric Eigenvalue Problem Subroutines
- Implementation of Various 8-bit Data Types in TLAPACK
- Using PCA in Genomics