Despite continuing advances in fast multigrid and domain decomposition preconditioners for iterative methods, commercial finite element codes continue to rely on direct factorization methods. These methods are reliable and make effective use of modern memory hierarchies to achieve high flop rates, but their memory usage scales poorly with problem size: standard direct factorization methods based on nested dissection typically require $O(N^{3/2})$ memory for "blocky" three-dimensional problems. Recently, "superfast" direct methods have been proposed that significantly reduce the time and memory complexity of standard direct methods by exploiting the fact that Cholesky factors in PDE problems often include large, nearly low-rank off-diagonal block. These methods show the potential to combine the reliability of direct solvers with the speed of iterative methods; but, as with standard sparse solvers, careful engineering is required to actually produce efficient implementations.

In this talk, we describe our code for "superfast" Cholesky factorization for large, sparse linear systems arising from PDE discretizations. Our code is based on a state-of-the art supernodal left-looking Cholesky solver, CHOLMOD; like CHOLMOD, our code is organized around dense matrix operations, and thus much of the work can be done in tuned level-3 BLAS routines that make efficient use of the memory hierarchy. After discussing the high-level idea behind our approach, we describe some details of how our code combines sparsity and low-rank structure and how we directly compute low-rank approximate block factorizations using randomized algorithms. We also illustrate the performance of our method with a challenging model problem from nearly-inompressible elasticity. We show that conjugate gradients with our rank-structured Cholesky factorization converges far more quickly than CG with standard preconditioners based on incomplete Cholesky and the ML multigrid solver, both in iteration counts and in run time on an eight-core Intel Xeon. We also describe the relative memory scalability of our approach compared to exact factorization; for an instance with roughly 10^6 degrees of freedom, our approach uses only 3 GB of memory, while exact factorization with CHOLMOD requires 30 GB.