FAST SOLVERS FOR MESH-BASED COMPUTATIONS

Direct solvers are the core part of many engineering analyses performed using different mesh-based methods, such as the finite difference method, the collocation method, the finite element method, and the isogeometric finite element or collocation methods. Existing direct solvers of linear equations (for example, MUMPS [1], SuperLU [2], PARDISO [3], or HSL [4]) are based on solving a linear system given by a global matrix and one or several right-hand sides. The global matrix is provided either as a list of non-zero entries, or it is obtained from merging a sequence of element frontal matrices. In both cases, the additional available knowledge about the structure of the computational mesh is lost.

In this talk I would like to present a new paradigm for designing direct solvers based on the structure of the computational mesh. The construction of the solver algorithm is based on the additional available knowledge concerning the structure of the computational mesh [5]. The alternative method presented in this talk allows us to outperform traditional direct solver algorithms.

The construction of the direct solver algorithm based on the structure of computational mesh allows for better decomposition of the computational problem into sets of independent tasks. This in turn allows us to obtain a solver algorithm that delivers more efficient parallel implementation (for distributed memory Linux cluster [6], shared-memory Linux node [7] or for GPGPU [8]. Additionally it allows us to implement some special tricks such as the reuse of computations for identical sub-parts of the mesh [9], and the reutilization of LU factorizations over unrefined parts of the mesh [10]. These techniques are not easily available for classic direct solvers.

Additionally, by analyzing the structure of the computational mesh we can generate better ordering algorithms [11], that result in lower number of floating point operations than the one obtained from classical ordering algorithms analyzing only the sparsity of the matrix (for example nested-dissections from METIS [12], MD, AMD or AMF [13], or PORD [14]).
REFERENCES