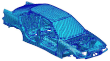
Le solveur MUMPS

et les travaux récents

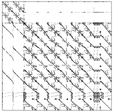
MUMPS group, CERFACS, CNRS, ENS Lyon, INPT, Inria, Univ. Bordeaux 1

September 23, 2013

Sparse Direct Methods



Discretization of a physical problem (eg, finite elements) ⇒ **Solution of sparse systems** $Ax = b$



Often the most expensive part of a simulation process

Sparse direct methods:

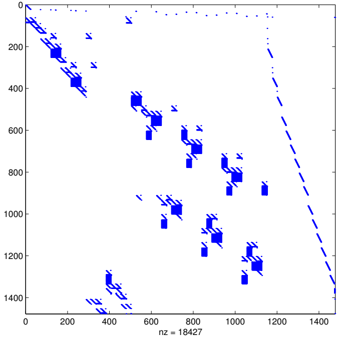
- Solve $Ax = b$ by decomposing A under the form LU,LDL^t or LL^t then solve triangular systems ($Ly = b$, then $Ux = y$)

Black box?

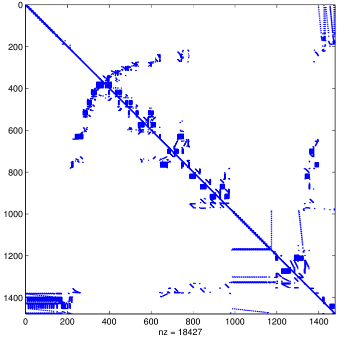
- Default (automatic/adaptive) setting of options is often available
- A better knowledge and setting of the preprocessing and algorithmic options can help the user improving:
  - size of factors and memory needed
  - operation count and computational time
  - reliability of the flops/memory estimates
  - numerical accuracy

Preprocessing - illustration

Original ($A = \text{LHR01}$)



Preprocessed matrix ($A'(\text{LHR01})$)

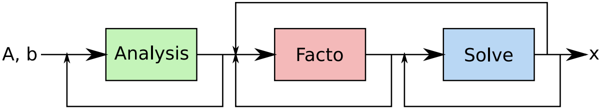


Modified Problem:$A'x' = b'$ with $A' = P_n P D_r A Q D_c P^t Q_n$

Three-phase scheme to solve $Ax = b$

1. Analysis
   - Preprocessing of A (permutations, scalings)
   - Build dependency graph (tree)
   - Prepare factorization (mapping, memory estimates)
2. Factorization: $A = LU$ (or LDL^t, or LL^t, or QR)

   dynamic pivoting for numerical stability
3. Solve:
   - the solution x is computed by means of forward and backward substitutions
   - improvement of solution (iterative refinement), error analysis



Some (shared memory) sparse direct codes

Code	Technique	Scope	Availability (www.)
BCSLIB	Multifrontal	SYM/UNS	Boeing → Access Analytics
HSL MA87	Supernodal	SPD	cse.clrc.ac.uk/Activity/HSL
MA41	Multifrontal	UNS	cse.clrc.ac.uk/Activity/HSL
MA49	Multifr. QR	RECT	cse.clrc.ac.uk/Activity/HSL
PanelLLT	Left-looking	SPD	Ng
PARDISO	Left-right	SYM/UNS	Schenk
PSL[†]	Left-looking	SPD/UNS	SGI product
SuperLU_MT	Left-looking	UNS	nersc.gov/∼xiaoye/SuperLU
SuiteSparseQR	Multifr. QR	RECT	cise.ufl.edu/research/sparse/SPQR
TAUCS	Left/Multifr.	SYM/UNS	tau.ac.il/∼stoledo/taucs
WSMP[†]	Multifrontal	SYM/UNS	IBM product

[†] Only object code available.

Some distributed-memory sparse direct codes

Code	Technique	Scope	Availability (www.)
DSCPACK	Multifr./Fan-in	SPD	cse.psu.edu/∼raghavan/Dscpack
MUMPS	Multifrontal	SYM/UNS	`graal.ens-lyon.fr/MUMPS` and `mumps.enseeiht.fr`
PaStiX	Fan-in	SYM/UNS	labri.fr/perso/ramet/pastix
PSPASES	Multifrontal	SPD	cs.umn.edu/∼mjoshi/pspases
SPOOLES	Fan-in	SYM/UNS	netlib.org/linalg/spooles
SuperLU	Fan-out	UNS	nersc.gov/∼xiaoye/SuperLU
S+	Fan-out	UNS	cs.ucsb.edu/research/S+
WSMP †	Multifrontal	SYM	IBM product

† Only object code available.

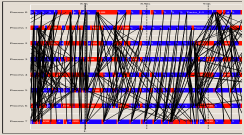
Main features

How does MUMPS compare with the others?

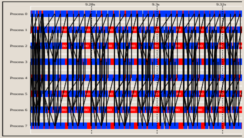
- Address wide classes of problems and architectures
- Good numerical stability (dynamic pivoting, preprocessing, postprocessing, error analysis)
- Wide range of numerical and algorithmic features

Management of parallelism

- Mainly MPI-based
- Dynamic and asynchronous approach



MUMPS



SuperLU_DIST

Main features (I)

Interfaces

Fortran, C, Matlab/Octave and Scilab interfaces

Input/Output formats

- Assembled format
- Distributed assembled format
- Sum of elemental matrices
- Sparse, multiple right-hand sides, centralized/distributed solution

Types of matrices

- Symmetric (positive definite, indefinite), Unsymmetric
- Single/Double precision with Real/Complex arithmetic
- Singular matrices

Main features (II)

Preprocessing and Postprocessing

- Reduce fill-in: symmetric orderings interfaced: AMD, QAMD, AMF, PORD, (par)METIS, (pt)SCOTCH
- Numerical preprocessing: unsymmetric orderings and scalings
- Iterative refinement and backward error analysis

Numerical pivoting

- Partial pivoting and two-by-two pivots (symmetric)
- Static pivoting and "Null" pivot detection, null-space basis

Main features (III)

Solving larger problems

- Hybrid scheduling: performance under memory constraints
- Out-of-core
- Parallel analysis

Miscellaneous

- Partial factorization and Schur complement
- Forward elimination during factorization
- Exploit user-provided memory
- Inertia, determinant, entries of A^{-1}, discard factors ...

MUMPS (MUltifrontal Massively Parallel Solver)

Initially funded by European project PARASOL (1996-1999)

http://graal.ens-lyon.fr/MUMPS and http://mumps.enseeiht.fr

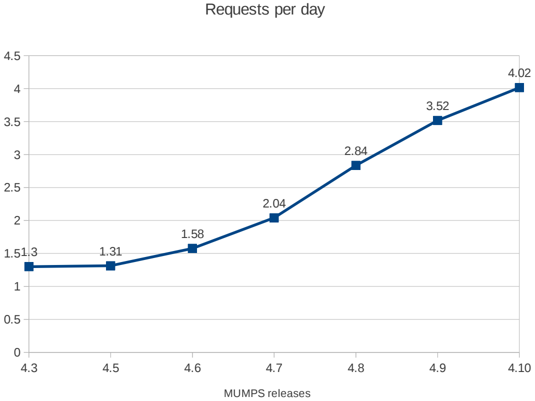
Platform for research

- Research projects, PhD thesis
- Hybrid direct-iterative methods: block Cimmino, Schur-based

Competitive software package used worldwide

- Co-developed by CERFACS, CNRS, ENS Lyon, INPT, Inria, Univ. Bordeaux 1
- Latest release: MUMPS 4.10.0, May 2011, $\approx$ 250 000 lines of C and Fortran code
- Integrated within commercial and open-source packages (e.g., Samcef from Samtech, Actran from Free Field Technologies, *Code_Aster* or Telemac from EDF, PAM-Crash from ESI Group, debian packages, IPOPT, Petsc, Trilinos, . . .).
- 1000+ downloads per year from our website, academia and industry

1000+ downloads per year from the MUMPS website



- ≈ 400 subscribers to the mumps-users mailing list: 2753 mail exchanges in 6 years
- exchanges developers ↔ users: ≈ 870 mail exchanges per year

Recent/ongoing work since last release (4.10.0)

- **Low-rank multifrontal Solver** (PhD of C. Weisbecker (INPT-IRIT, 2010→ Oct. 28, 2013) on Block-Low-Rank method) Collaboration with EDF (O. Boiteau) and C. Ashcraft (Livermore, USA)
- **Memory scalability and quality of memory estimates**: Robust memory-aware mappings (PhD of Agullo (ENS Lyon, 2005-2008) and Rouet (INPT-IRIT, 2009-2012))
- **Solution phase** performance (time and memory) (PhD of Rouet (INPT-IRIT, 2009-2012)): dense and sparse, IC and OOC environments (increased block size for multiple RHS)
- **Hybrid parallelism** on clusters of multicore architectures (PhD of W. Sid-Lakdhar, ENS Lyon, 2011-) and on heterogeneous architectures (PhD F. of Lopez, UPS-IRIT, 2012-)

Objective: integrate results of these major research tracks in future MUMPS releases (medium-term engineering work)

Related project: the qr_mumps code (i)

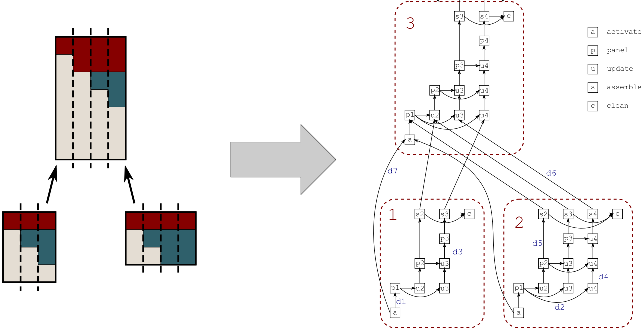
- multifrontal sparse QR solver
- designed for multicore systems (single-node).
- mainly developed by A. Buttari
- qr_mumps v1.0 available at
  `http://buttari.perso.enseeiht.fr/qr_mumps`

Main characteristics

- 40 000 lines of Fortran2003 code (full C interface)
- OpenMP multithreading (can now use StarPU)
- S, D, C, Z arithmetics
- COLAMD, METIS and SCOTCH orderings
- multithreaded solve ($R^{-1}x$, $R^{-T}x$, Qx, Q^Tx)
- singletons detection

Related project: the qr_mumps code (ii)

If a task is defined as the execution of one elementary operation on a block-column or a front, then the entire multifrontal factorization can be represented as a Directed Acyclic Graph (DAG)



where nodes represent tasks and edges the dependencies among them: "DAG-based approach"

Multifrontal solvers on top of **runtime systems**

- Subject of the PhD thesis of F. Lopez (UPS-IRIT, 2012-)
- Current focus: `qr_mumps` (rather than MUMPS) + StarPU (RUNTIME team, Bordeaux) to address both multicores and GPUs
- Work continues in the context of the ANR project SOLHAR (coordinator: A. Guermouche)

References:

- E. Agullo, A. Buttari, A. Guermouche, F. Lopez, *Multifrontal QR Factorization for Multicore Architectures over Runtime Systems*, Europar 2013, LNCS 8097, pp. 521–532 (2013).
- A. Buttari. *Fine-grained multithreading for the multifrontal QR factorization of sparse matrices*, SISC 35(3): pp. C323-C345 (2013).

Thanks for your attention!