Using low-rank approximation techniques for engineering problems

François-Henry Rouet

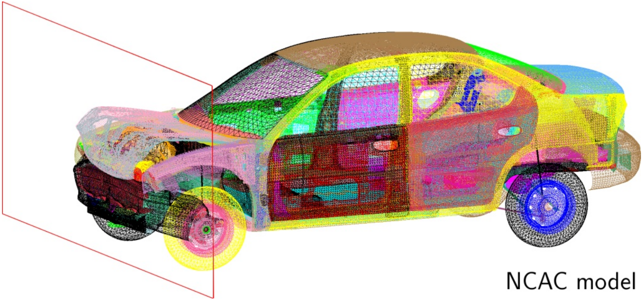
Livermore Software Technology Corporation

Joint work with: J. Anton, C. Ashcraft, R. Grimes, P. L'Eplattenier, C. Weisbecker

International Conference on Computational Methods, August 4th, 2016

LS-DYNA

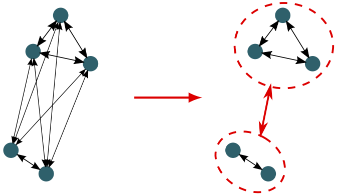
- Originated at the Lawrence Livermore Lab (DYNA3D, 1976).
- Applications: automotive crash and occupant safety, metal forming, CFD, FSI, electromagnetism, acoustics, thermal...
- Finite elements, boundary elements, meshless, SPH...
- About 50% of the explicit crash market, a share of the growing implicit market.
- Linear algebra team: Bob Lucas, Cleve Ashcraft, Roger Grimes, Julie Anton, Clément Weisbecker, F.-H. Rouet.



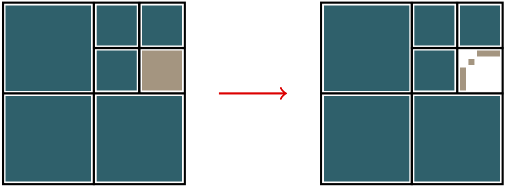
NCAC model

Low-rankness

- Low-rank/structured methods rely on data sparsity, similar to the Fast Multipole Method.



- In algebraic terms: some off-diagonal blocks of the input matrix are low-rank; they can be compressed.



- NB: sometimes this applies to intermediate matrices (not the input matrix), e.g., in sparse factorizations.

Classes of structured matrices

Most structured matrices belong to the class of Hierarchical matrices ($\mathcal{H}-$matrices) [Hackbusch, Bebendorf, Börm, Grasedyck. . .].

- $\mathcal{H}^2$ (Hackbusch, Börm, et al.)
- HSS (Chandrasekaran, Jia, et al.)
- HODLR (Darve et al.)
- BLR (Amestoy, Ashcraft, et al.)
- $+$ SSS, MHS, . . .

In this talk:

- We focus on HSS and BLR.
- We report results with problems from LSTC's applications.

Differences

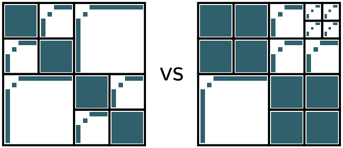
Three criteria differentiate all the low-rank formats:

Differences

Three criteria differentiate all the low-rank formats:

- Clustering/partitioning: off-diagonal blocks can be refined or not.

The partitioning is defined by a single tree whose leaves cluster $[1, n]$.



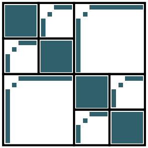
The partitioning is defined by the product of two trees (rows $\times$ columns).

Differences

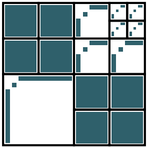
Three criteria differentiate all the low-rank formats:

- Clustering/partitioning: off-diagonal blocks can be refined or not.

The partitioning is defined by a single tree whose leaves cluster $[1, n]$.



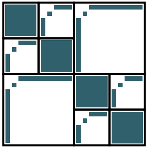
vs



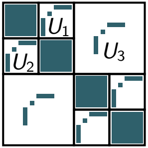
The partitioning is defined by the product of two trees (rows $\times$ columns).

- Nested basis or not.

Blocks have independent compressed representations (bases).



vs



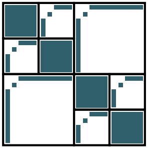
Shared information:

$$U_3^{\text{big}} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} U_3$$

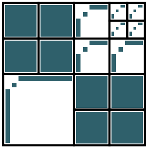
Three criteria differentiate all the low-rank formats:

- Clustering/partitioning: off-diagonal blocks can be refined or not.

The partitioning is defined by a single tree whose leaves cluster $[1, n]$.



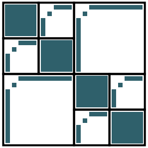
vs



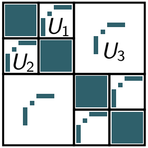
The partitioning is defined by the product of two trees (rows $\times$ columns).

- Nested basis or not.

Blocks have independent compressed representations (bases).



vs

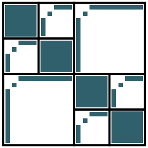


Shared information:

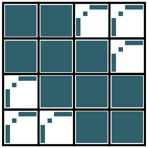
$$U_3^{\text{big}} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} U_3$$

- Buffer zone next to the diagonal or not (“strong admissibility”).

Assumes interaction between two clusters is low-rank.



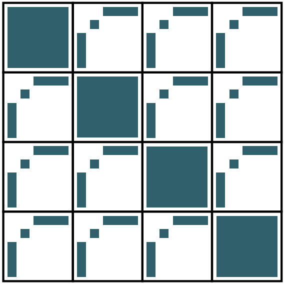
vs



Blocks next to the diagonal not “admitted” (compressed).

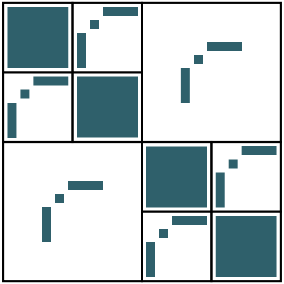
Two classes of hierarchical matrices

Block Low-Rank (BLR):



- Flat partitioning.
- No nested basis; indep. blocks.
- Buffer zone possible.

Hierarchical Semi-Separable (HSS):



- Tree-based partitioning.
- Nested basis.
- No buffer zone.

Two extremes of the low-rank spectrum. HODLR in the middle.
HSS essentially algebraic Fast Multipole Method.

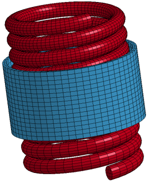
Experiments with dense problems – setup

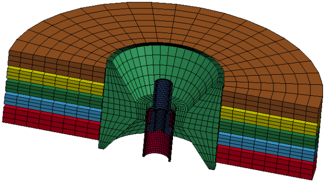
Codes

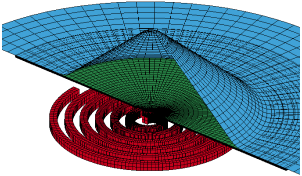
- BLR used at LSTC for 12 years, mostly for electromagnetism. Shared-memory code, MPI in progress.
- Benchmarking HSS, STRUMPACK (R., Li, Ghysels, LBNL).

Problems – mixed BEM/FEM for electromagnetism

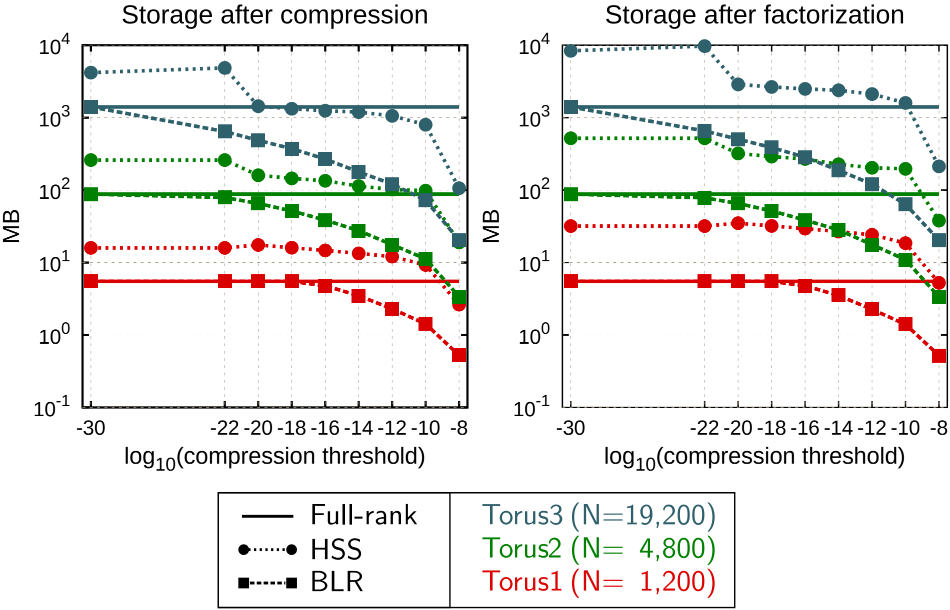
- Easy problems: a family of tori.
- Harder: complex geometries involving coupling of different pieces/materials. 10,000-50,000 rows.



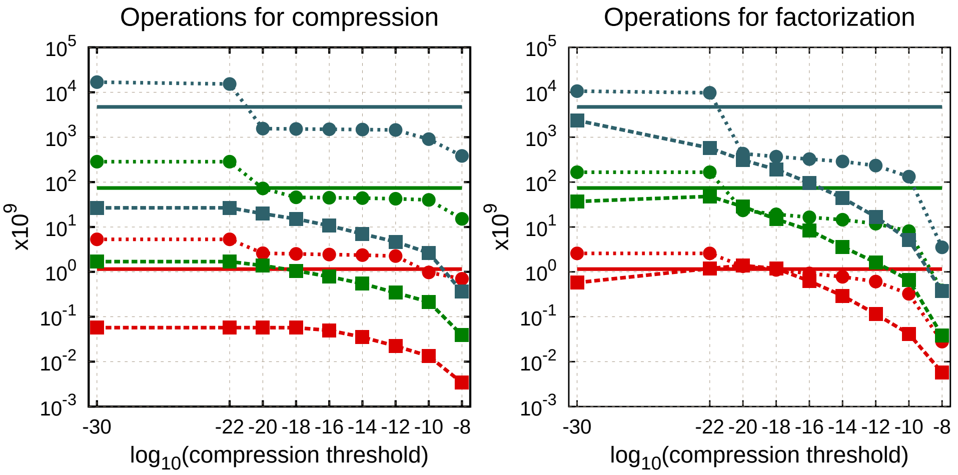




Experiments with dense problems – storage



Experiments with dense problems – operations



Full-rank	Torus3 (N=19,200)
HSS	Torus2 (N= 4,800)
BLR	Torus1 (N= 1,200)

Experiments with dense problems – observations

Tori

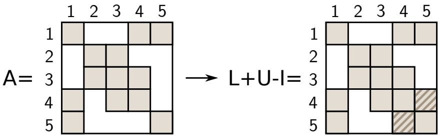
- Storage and operations: HSS doesn't catch up with BLR as size increases. Ranks grow too fast.
- Fill-in (compressed matrix $\rightarrow$ factored matrix):
  - HSS: factors typically 2-4x larger than the compressed form.
  - BLR: almost no fill-in. $A_{I,J} \leftarrow A_{I,J} - \sum_K L_{I,K} U_{K,J}$; 95% of the time rank doesn't increase. 50% it actually decreases.
- Backward error: for both BLR and HSS, $\frac{||Ax-b||}{||b||} \simeq 10^5 \cdot \varepsilon$, with ε the compression threshold.
- Run time: strongly correlates with #flops. BLR lower flop rate than HSS (smaller tasks).

Harder problems

Similar observations. HSS yields larger storage and operation count.

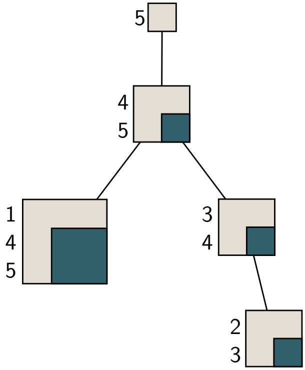
Low-rank representations and sparse solvers

Many LR techniques embedded in the multifrontal method [Duff & Reid '83]. Related: "sweeping preconditioner" [Ying, Engquist,. . .], elliptic solver [Chavez et al. '16]. . .



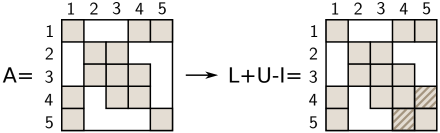
Traverse the tree bottom-up; at each node (a.k.a. frontal matrix):

- Partial factorization (yields parts of L/U).
- Compute a contribution block (Schur complement) to be used at the parent.



Elimination tree

Many LR techniques embedded in the multifrontal method [Duff & Reid '83]. Related: "sweeping preconditioner" [Ying, Engquist,...], elliptic solver [Chavez et al. '16]...

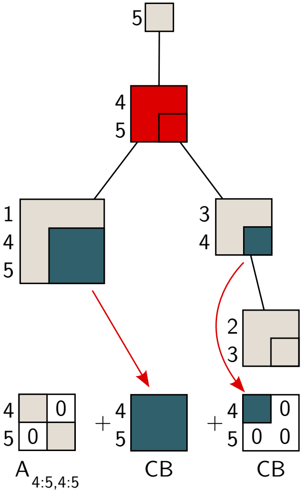


Traverse the tree bottom-up; at each node (a.k.a. frontal matrix):

- Partial factorization (yields parts of L/U).
- Compute a contribution block (Schur complement) to be used at the parent.

  Assembly/extend-add operation:

  $F = A_{I,J} \mathbin{\leftrightarrow\!\!\!\!\!\updownarrow} \mathrm{CB}_{\text{child 1}} \mathbin{\leftrightarrow\!\!\!\!\!\updownarrow} \mathrm{CB}_{\text{child 2}} \mathbin{\leftrightarrow\!\!\!\!\!\updownarrow} \dots$



Extend-add and low-rank representations

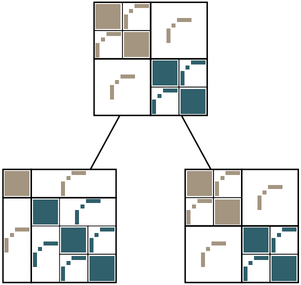
How to do extend-add with compressed matrices?

- BLR: contribution blocks not compressed [Amestoy et al. '12].

Extend-add and low-rank representations

How to do extend-add with compressed matrices?

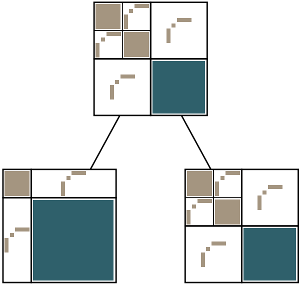
- BLR: contribution blocks not compressed [Amestoy et al. '12].
- HSS, [Jia et al. '09], serial 2D code. HSS extend-add. Slow and hard to parallelize. Hard to extend to algebraic.



Extend-add and low-rank representations

How to do extend-add with compressed matrices?

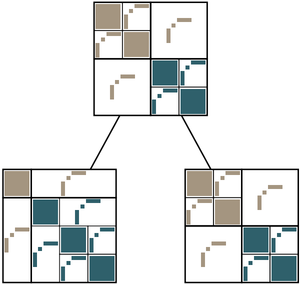
- BLR: contribution blocks not compressed [Amestoy et al. '12].
- HSS, [Jia et al. '09], serial 2D code. HSS extend-add. Slow and hard to parallelize. Hard to extend to algebraic.
- HSS, [Wang et al. '15], parallel geometric code. Contribution blocks not compressed. Simple but penalty on memory footprint.



Extend-add and low-rank representations

How to do extend-add with compressed matrices?

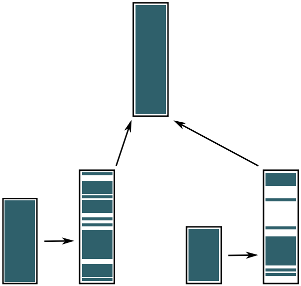
- BLR: contribution blocks not compressed [Amestoy et al. '12].
- HSS, [Jia et al. '09], serial 2D code. HSS extend-add. Slow and hard to parallelize. Hard to extend to algebraic.
- HSS, [Wang et al. '15], parallel geometric code. Contribution blocks not compressed. Simple but penalty on memory footprint.
- HSS, [Ghysels et al. '15], parallel algebraic code with randomized sampling:
  1. After partial factorization, compute $Y = CB \cdot R$ with R random tall-skinny matrix. Y is a sample of the Schur Complement.
  2. At parent node, compute a sample of the frontal matrix F as:
     $$F \cdot R = A \cdot R \updownarrow Y_1 \updownarrow Y_2 \ldots$$
     Extend-add with "extension" only along rows.



Extend-add and low-rank representations

How to do extend-add with compressed matrices?

- BLR: contribution blocks not compressed [Amestoy et al. '12].
- HSS, [Jia et al. '09], serial 2D code. HSS extend-add. Slow and hard to parallelize. Hard to extend to algebraic.
- HSS, [Wang et al. '15], parallel geometric code. Contribution blocks not compressed. Simple but penalty on memory footprint.
- HSS, [Ghysels et al. '15], parallel algebraic code with randomized sampling:
  1. After partial factorization, compute $Y = CB \cdot R$ with R random tall-skinny matrix. Y is a sample of the Schur Complement.
  2. At parent node, compute a sample of the frontal matrix F as:
     $$F \cdot R = A \cdot R \updownarrow Y_1 \updownarrow Y_2 \ldots$$
     Extend-add with "extension" only along rows.



Sparse solvers at LSTC

- BCSLIB-EXT:
  - Shared-memory multifrontal code.
  - Boeing's "Industrial Strength" direct solver.
  - Gordon Bell Award 1988 using Cray YMP-8.
  - Still active in LS-DYNA, used in a new development project.
- MF2:
  - MPI+OpenMP multifrontal code by Bob Lucas.
  - LS-DYNA default solver for twenty years.
- MUMPS:
  - MPI(+OpenMP) multifrontal code by Amestoy et al., France.
  - BLR in Weisbecker's thesis (2013) and Mary's thesis (on-going).

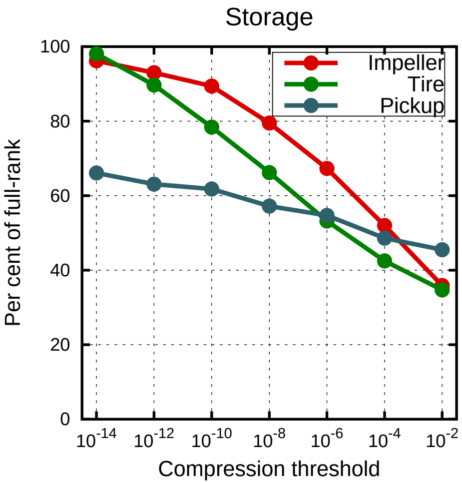
Experiments with sparse problems – setup

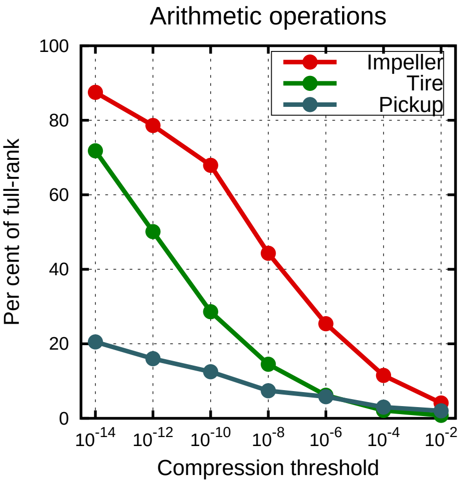
Three matrices from implicit mechanics:

- Impeller: jet turbine, 96 fan blades on hub, 21M dof (3D).
- Tire: model of tire rubber, 3M dof (3D).
- Pickup: full scale model of a pickup truck, 40M dof (2.5D).

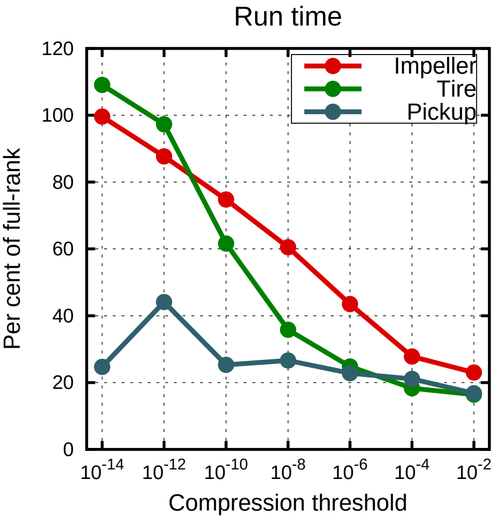
Experiments with MUMPS-BLR, 16 MPI processes.

Experiments with sparse problems – storage, operations





Experiments with sparse problems – run time (16 MPI)



Experiments with sparse problems – observations

- Run time strongly correlates with #flop reduction, much faster than classical/full-rank factorization.
- Component-wise scaled residual $\max_i \frac{|Ax-b|_i}{(|A||x|+|b|)_i} \simeq (10^2-10^3) \cdot \varepsilon$. Iterative refinement recovers lost digits.
- Future work: same experiments with the MPI version of STRUMPACK-sparse (in development, Ghysels et al., LBNL).

Conclusion

Low-rank approximations are a robust acceleration technique. Can be used as near-direct solvers or aggressive preconditioners.

- Dense problems:
  - BLR very effective for our BEM problems.
  - HSS doesn't pay off now, but maybe in the future as our problems get bigger and bigger.
- Sparse problems:
  - BLR (in MUMPS) effective for our implicit mechanics problems. On-going experiments with MF2.
  - HSS shown to be effective as an aggressive preconditioner ($\varepsilon = 10^{-2}, 10^{-1}, \ldots$) [Ghysels et al, SISC 2016]. On-going experiments.

End

Thank you for your attention!

Any questions?