MUMPS-BLR inside a preconditioned eigensolver

François-Henry Rouet
Ansys, Inc.

June 22th, 2023

LS-DYNA multiphysics solver

History:

- Started at Lawrence Livermore Lab in the 70's by **John Hallquist**;
- Private company (LSTC) for 30+ years;
- Acquired by **Ansys** in 2019.

Capabilities:

- Initial focus was **mechanical engineering** (weapons, car crash. . .);
- Strongly-coupled **multiphysics** added over the years: fluids, electromagnetism. . .

Linear algebra team of 6:

- Two are **MUMPS graduates**;
- Sparse, dense; direct, iterative; low-rank; eigensolvers; constraints; reordering. . .
- MPI, OpenMP, GPUs, and "exotic" architectures.
- Lots of in-house codes, some external codes; MUMPS is one of them.

Eigenanalysis – the vibration problem

Vibration problem in structural mechanics:

- FEM discretization of laws of conservation lead to

  $$M\ddot{u}(t) + Ku(t) = 0$$

  K **stiffness matrix**, M **mass matrix**; real, symmetric, semi-definite.

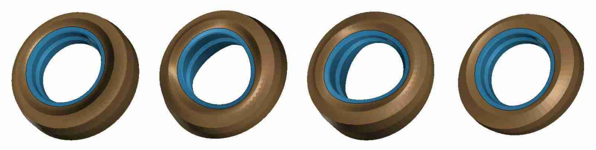
- Looking for solutions of the form $\phi e^{i\omega t}$ leads to

  $$K\phi = \lambda M\phi \qquad \text{with } \lambda = \omega^2$$

Eigenanalysis – the vibration problem

Vibration problem in structural mechanics:

- FEM discretization of laws of conservation lead to
$$M\ddot{u}(t) + Ku(t) = 0$$
K **stiffness matrix**, M **mass matrix**; real, symmetric, semi-definite.
- Looking for solutions of the form $\phi e^{i\omega t}$ leads to
$$K\phi = \lambda M\phi \qquad \text{with } \lambda = \omega^2$$
- Eigenvalues are **frequencies (squared)**; the **smallest ones** matter. **Eigenvectors are "mode shapes"**:



Eigenanalysis – the vibration problem

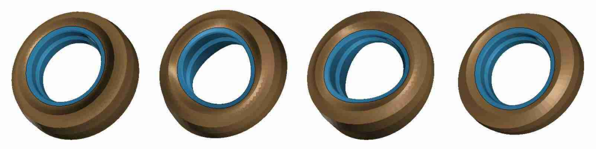
Vibration problem in structural mechanics:

- FEM discretization of laws of conservation lead to

  $$M\ddot{u}(t) + Ku(t) = 0$$

  K **stiffness matrix**, M **mass matrix**; real, symmetric, semi-definite.
- Looking for solutions of the form $\phi e^{i\omega t}$ leads to

  $$K\phi = \lambda M\phi \qquad \text{with } \lambda = \omega^2$$
- Eigenvalues are **frequencies (squared)**; the **smallest ones** matter.
  Eigenvectors are "mode shapes":



Other problems can lead to unsymmetric, complex, quadratic eigenvalue problems: rotational dynamics, break squeal. . .

Eigenanalysis – how many modes?

How many modes do analysts want?

- Quick model checking: ***O*(10) modes**.
  - Typically to find **rigid-body modes** = zero eigenmodes. Structure (or subpart) floating freely in space. Typically 6 RBMs (translations + rotations over the 3 axes).
  - Can point to missing connection or constraint.
- Typical modal analysis: **tens or a few hundred of modes**.
  - One standard criterion is **modal effective mass**: the amount of mass that the first eigenmodes move along.
  - Typically, analysts want 80% or more cumulative mass participation.
- Automotive Noise-Vibration-Harshness: **thousands of modes, low accuracy**;
  - Maybe 0.1% error for the lowest frequencies then 1%-5% error for the higher ones.
  - Sometimes only a few selected entries of the eigenvectors are needed.

Eigenanalysis – solvers

Solvers:

- Block Shift-Invert **Lanczos**:
  - Uses a direct solver to factor and solve with $K - \sigma M$, σ shift(s).
  - Very robust, almost never misses anything. Guided by inertia count/Sturm sequence.
  - The Boeing version **[Grimes, Lewis, Simon '94]** is the gold standard.
- Subspace iteration: good for a few modes.
- **AMLS** (Automated MultiLevel Substructuring **[Bennighof '92]**): good for coarse approximation of lots of modes.
- **LOBPCG** (and other preconditioned eigensolvers): see next.
- Davidson methods: popular for some fields like Quantum Chemistry, haven't seen it much for mechanics.
- . . .

LOBPCG

Locally Optimal Block Preconditioned Conjugate Gradient [Knyazev '01]:

- **Rayleigh quotient minimization** technique; $\lambda(u) = \frac{u^T K u}{u^T M u}$.
- Minimization done over a subspace spanned by:
  - U current eigenvector approximations;
  - W **preconditioned residuals**; does **NOT** need an "exact" solve like Lanczos.
  - P search directions.
- Algorithm in a nutshell:
  Repeat until residual norms are small enough:
  1. M-orthogonalize W against U and P, then W itself.
  2. **Rayleigh-Ritz procedure** on the projected eigenproblem $\widehat{K}\widehat{X} = \widehat{M}\widehat{\Lambda}\widehat{X}$

$$\widehat{K} = \begin{bmatrix} U & P & W \end{bmatrix}^T K \begin{bmatrix} U & P & W \end{bmatrix}$$
$$\widehat{M} = \begin{bmatrix} U & P & W \end{bmatrix}^T M \begin{bmatrix} U & P & W \end{bmatrix}$$

  3. Update the search space: $P \leftarrow W\widehat{X}_W + P\widehat{X}_P, \quad X \leftarrow X\widehat{X}_X + P$
- The BLOPEX paper **[Knyazev et al. '07]** has practical recommendations.

LOBPCG in LS-DYNA

LOBPCG code:

- Serial code implemented by **Eugene Vecharynski** (Knyazev's student).
- Distributed-memory implementation is fairly straightforward.
- Eigenpairs are computed in blocks, using shifts.

Choice of preconditioner:

- For our mechanical problems, simple preconditioners (block diagonal, ILU0...) simply don't cut it.
- Multigrid can work but is very finicky, needs lots of physical info.
- **MUMPS-BLR** is very attractive thanks to the tunable accuracy.

LOBPCG in LS-DYNA – MUMPS usage

MUMPS usage:

- MUMPS **5.6.0.c**, ifort + gcc + Intel MKL.
- Distributed matrix (`ICNTL(18)=3`), distributed dense RHS and solution.
- Ordering with **ParMETIS** or our partitioner **LS-GPart**.
  Waiting on automatic graph compression :-)
- Aggressive scheduling is on.
- BLR: `ICNTL(35)=2`.
- Simple sequences of calls, 1 MUMPS instance:
  - Analyze once;
  - Factor, solve, solve, solve. . .
  - New shift σ: factor $K - \sigma M$, solve, solve, solve. . . .
    (New shift doesn't change the structure.)
  - . . .
  - Release instance.

A first example

Synthetic problem:

- Regular grid of solid elements (cubes), elastic material.
- Stiffness matrix: $n = 7.3$M; $nz = 295.1$M. Computing **50 modes**.

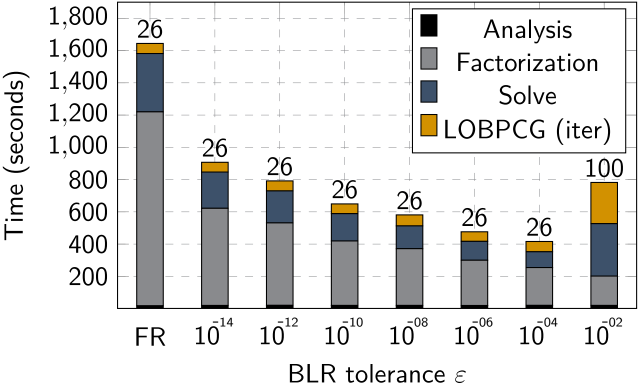
This is a **best-case scenario**:

- BLR likes these bulky 3D geometries;
- Nice mesh, "easy" elements. . . not hard to precondition.

We look at:

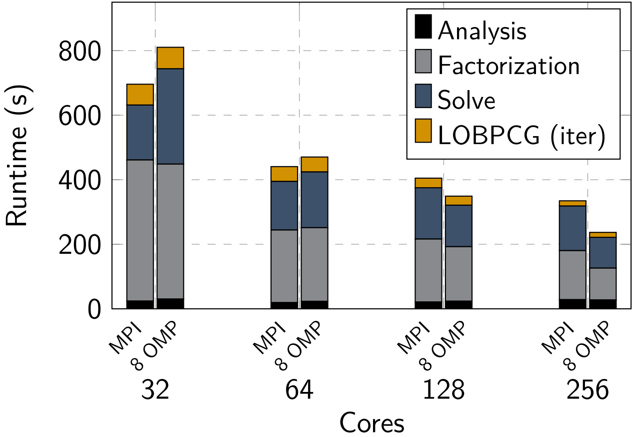
- Performance as a function of BLR tolerance ε;
- Scalability: pure MPI and hybrid parallel using the "MPI to k OMP" feature.

A first example – BLR tolerance ε



- Number of iterations stays constant till $\varepsilon = 10^{-4}$, then LOBPCG breaks down.
- **BLR buys us a factor of 4 compared to a full-rank solution.**
  Factor compression: 19%; opc compression: 10%.
- But remember, this is a very easy problem!

A first example – scalability



- Speed-up of 3 out of 8 when going from 32 to 256 cores.
- Hybrid parallelism using the “MPI to k OMP” feature helps at high core count.
- The **triangular solve** is the limiting factor.

A real-life example

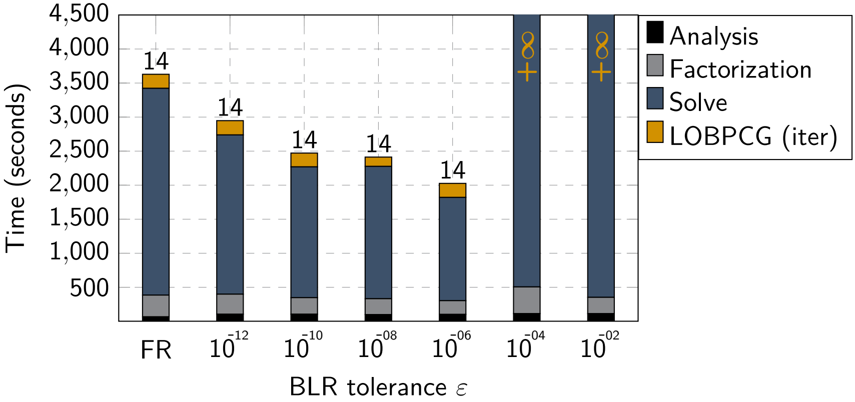
9M element electric pick-up truck:

- Discretized with solids, shells, beams. . . with different integration rules. . .
- 30+ types of materials: metals, glass, rubber, foam. . .
- Constraints and boundary conditions: rigid bodies, joints, spotwelds, contacts. . .
- Stiffness matrix: $n = 37.8\text{M}$; $nz = 1.29\text{B}$. Computing **50 modes**.

This is much harder:

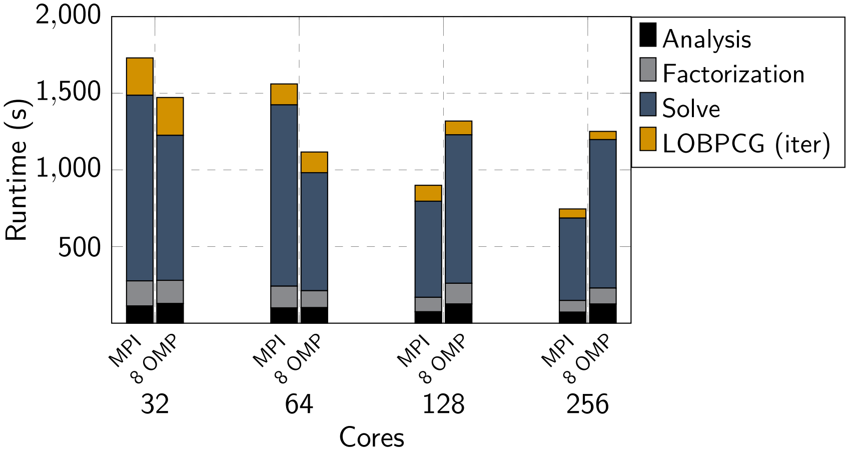
- Hollow "2.5D" geometry is **less amenable to compression**.
  E.g., the root node is just a few thousands equations, some kind of 1D manifold.
  (How do you cut a pick-up truck in half?)
- Shell elements are notoriously hard for iterative solvers.
- All kinds of heterogeneities and nonlinearities are present.

A real-life example – BLR tolerance



- Can't push ε as far. Tried GMRES on top of MUMPS-BLR and it didn't help.
- Now the triangular solve dominates.
- **BLR buys us a factor of 1.8 compared to a full-rank solution.**
  Factor compression: 64%; opc compression: 21%.

A real-life example – scalability



- Pure MPI gives a speed-up of 2.3 out of 8 when going from 32 to 256 cores.
- Not sure what's happening with hybrid parallelism for 128 and 256 cores.

A real-life example – trying mixed precision BLR. . .

- The manual recommends combining mixed precision with **contribution block compression**.
- For $\varepsilon = 10^{-6}$ (our default), this slowed down LOBPCG convergence dramatically (43 vs 14 iterations).
- For $\varepsilon = 10^{-8}$:

Mode	CB compr?	Mixed?	Factor(s)	Solve(s)	Memory(GB)
FR	-	-	309.4	2190.1	348.8
BLR	no	no	242.8	1223.6	300.7
BLR	yes	no	278.1	1236.2	280.5
BLR	yes	yes	250.8	1094.6	**262.8**

Clear impact on memory usage.

An NVH example

2.8M element electric sedan (body in white with battery packs):

- Same “spirit” as the previous model, just smaller.
- Stiffness matrix: $n = 11.9\text{M}$; $nz = 405.8\text{M}$.
- Computing **2000 modes**; only want **a few digits of accuracy**.

Questions:

- Can we lower the LOBPCG tolerance to get to a decent approximation faster?
- How do we setup BLR in these conditions?

An NVH example – results

Tolerances:

- We can decrease the convergence criterion from 10^{-12} to 10^{-8}. Past this, things get iffy, LOBPCG can get lost in some clusters of eigenvalues. The threshold can be increased towards the end of the spectrum.
- We can decrease the BLR tolerance a tiny bit. But **still can't be too aggressive** ($10^{-4}, 10^{-2} \ldots$ break down) :-(

Results (2000 modes, 40 shifts):

Mode	Analysis	Factor	Solve	LOBPCG	Total	#iter
Default	26.9	1042.1	5702.3	4448.7	11220.0	659
"Fast"	27.2	920.0	2517.5	2376.1	5840.8	321

Observations:

- Probably not as fast as AMLS, but better quality (**$<$ 0.01% error** across the spectrum).
- **Orthogonalization becomes a bottleneck**.

Conclusion

MUMPS-BLR as a preconditioner for LOBPCG:

- Robust, BLR-$\varepsilon = 10^{-6}$ seems to be the sweetspot. Can be relaxed a bit for a coarse eigensolve but not too much.
- Fairly scalable. The performance of **the triangular solve is the critical piece** here: many calls + many RHS every call.

Future work:

- Play with new BLR enhancements.
- Try new orthogonalization schemes (new hire **Daniel Bielich**).
- Redistribute our data to conform to the MUMPS distribution/tree (`ISOL_loc`) after each shift/factorization instead of going back and forth at each iteration.

The End

Thank you for your attention!

Any questions?