Scaling of normal modes computation in SunShine using MUMPS solver

MUMPS Users Day – 2023

Jason Pavlidis
Moduleering

22 June 2023

Table of contents

SunShine Overview

Moduleering is a Research and Development company, the Greek branch of *TechnoStar Co., Ltd. Japan*

SunShine is a modern, powerful and robust multiphysics simulation software, that supports a wide variety of analysis needed in engineering:

- Structural linear / nonlinear analysis
- **Normal modes analysis**
- Buckling analysis
- Complex eigenvalue analysis
- Frequency / Transient response analysis
- Steady-state / Transient heat transfer analysis
- Electrostatic / Magnetostatic analysis

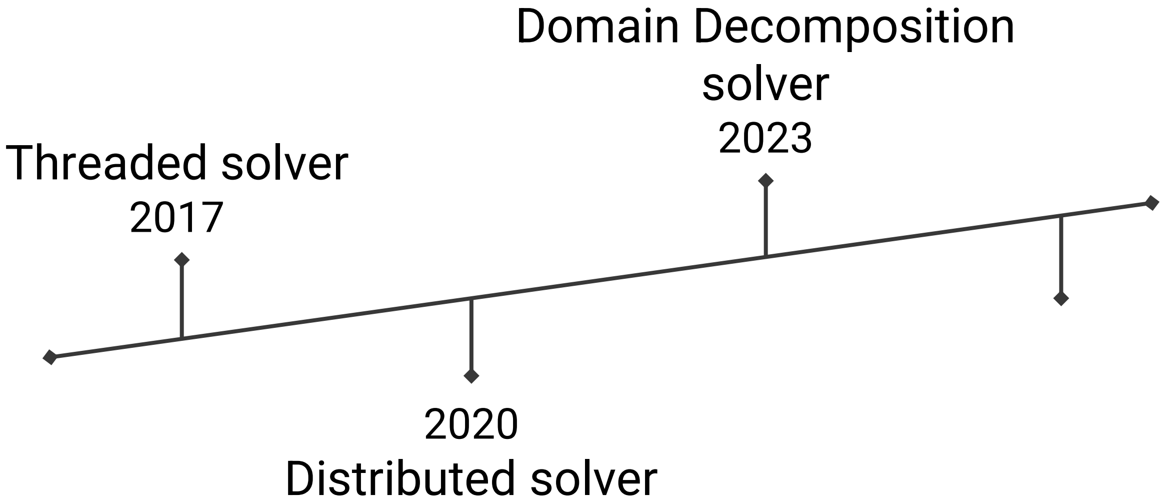
MUMPS in SunShine

SunShine uses various MUMPS features to enhance its performance and its capabilities.

- Distributed factorization and solution
- Distributed right-hand side / solution vector
- Out-of-core factorization and solution
- Null pivot detection

SunShine

SunShine History



Normal Modes Analysis

Natural frequencies and normal modes characterize the basic dynamic behavior of a structure.

They can be computed using the equation of motion for undamped free vibration.

$$[M][\ddot{u}] + [\mathrm{K}][\mathrm{u}] = 0 \xRightarrow{u = \{\phi\}\, sin\omega t}$$

$$([K] - \omega^2[M])\{\phi\} = 0 \Longrightarrow$$

$$([K] - \lambda[M])\{\phi\} = 0 \Longrightarrow$$

$$[K]\{\phi\} = \lambda[M]\{\phi\}$$

Generalized eigenproblem with real symmetric matrices. We are usually interested to a few of the smallest eigenvalues.

Shifted Block Lanczos

A very robust and efficient method to compute a few eigenpairs of large sparse matrices on a desired frequency interval.

Instead of

$$Kx = \lambda M x$$

Use spectral transformation and solve

$$M(K - \sigma M)^{-1} M x = \mu M x \quad , \qquad \mu = \frac{1}{\lambda - \sigma}$$

The method can be described by the following transformation

$$Q_j^T M (K - \sigma M)^{-1} M Q_j = T_j \qquad T_j = \begin{bmatrix} A_1 & B_2^T & 0 & \dots & 0 \\ B_2 & A_2 & B_3^T & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & B_{j-1} & A_{j-1} & B_j^T \\ 0 & \cdots & 0 & B_j & A_j \end{bmatrix}$$

MUMPS Usage in Lanczos

Linear system solution

$$(K - \sigma M)\, X = B$$

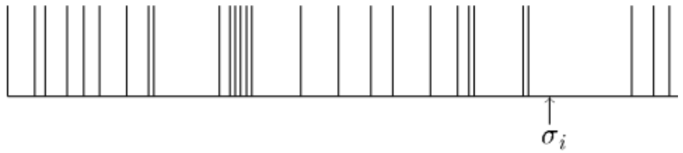
Computation of Inertia

$$(K - \sigma M) = LDL^T$$

Number of negative pivots in D $\longrightarrow$ number of eigenvalues λ < σ

Null pivot in D $\longrightarrow$ σ is very near to an eigenvalue

Very important information to choose a new shift and seek for remaining eigenvalues



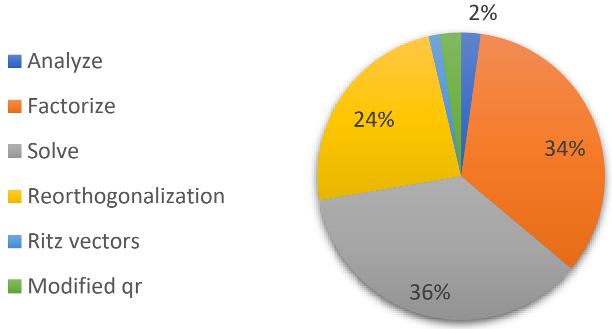
Lanczos Cost Analysis

Algorithm 1 Shifted Block Lanczos algorithm

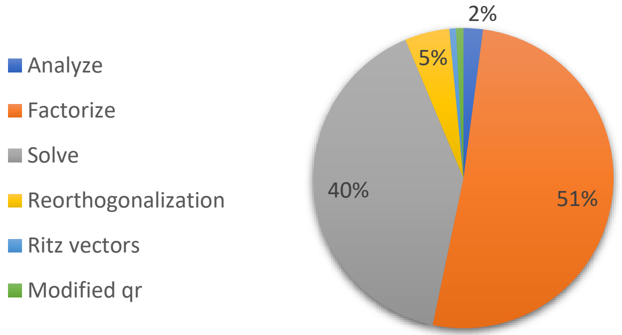
```
for $j = 1, \ldots, m$ do
    $U_j = \boxed{(K - \sigma M)^{-1}(MQ_j)} - Q_{j-1}B_{j-1}^T$
    $A_j = U_j^T(MQ_j)$
    $R_{j+1} = U_j - Q_jA_j$
    $Q_{j+1}B_{j+1} = qr(R_{j+1})$ such that $Q_{j+1}^T MQ_{j+1} = I$

    $VDV^T = eig(T_j)$ and check for convergence

    Reorthogonalize $Q_{j+1}$ and $Q_j$
end for
$Y = QV$
return $(D, Y)$
```

- The structure of matrix $K - \sigma M$ does not change with shift value so we can analyse only once
- We factorize once for each shift value and solve in each iteration
- Factorization and solution phases require $70\% +$ of the total execution time

Model 6M | Shell | 100 modes



Model 6M | Solid | 100 modes



Benchmark Environment

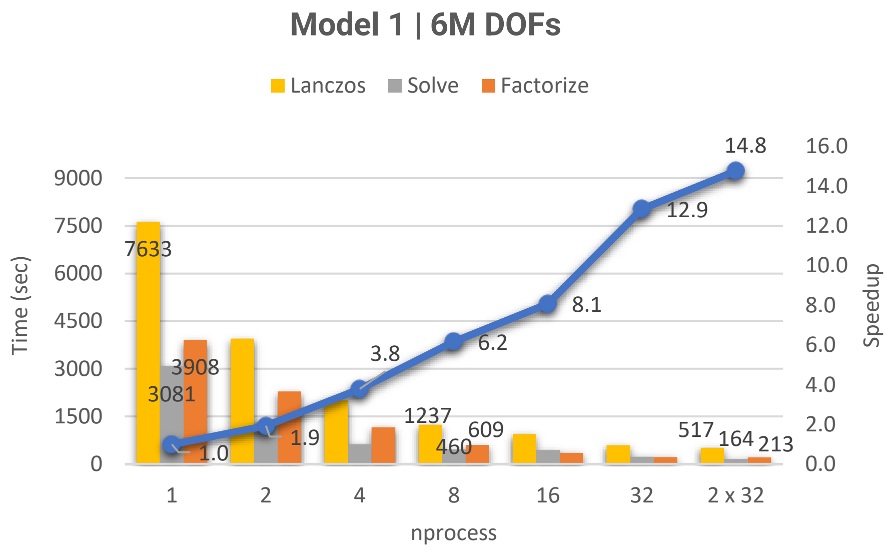
Hardware Configuration	
Nodes	2
CPU / node	2 x AMD EPYC 7301 16-Core / 32-Thread
Cores / node	32
RAM / node	250 GB
Disk / node	2TB NVME

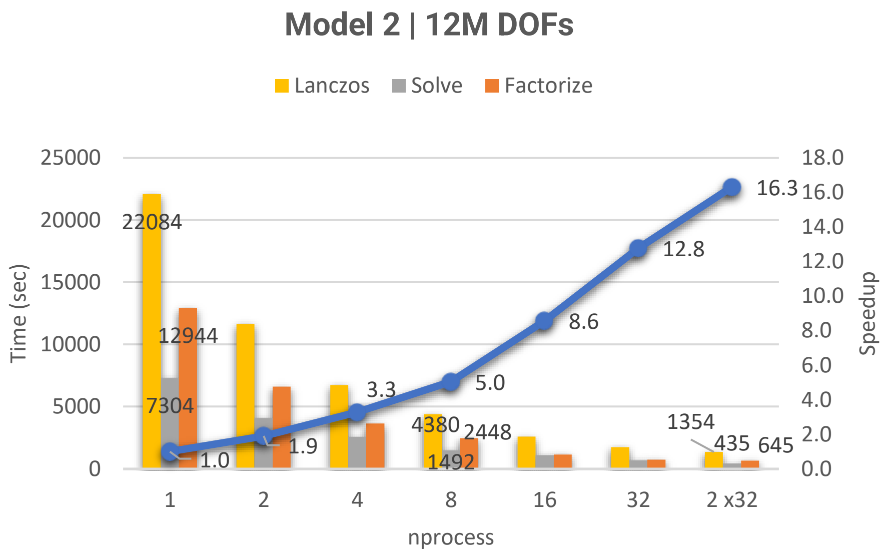
Software Configuration	
OS	CentOS Linux 7
MUMPS	5.5.1
Analysis	METIS
Factorization	Both in-core and out-of-core
Solve	Distributed right-hand side / distributed solution

Model	# DOFs	# Non zeros	Element Type
Model 1	6M	257M	Solid - Ctetra10
Model 2	12M	477M	Solid – Ctetra10
Model 3	7M	117M	Shell – Cquad4/Ctria3
Model 4	12M	199M	Shell – Cquad4/Ctria3

Results (1/2)

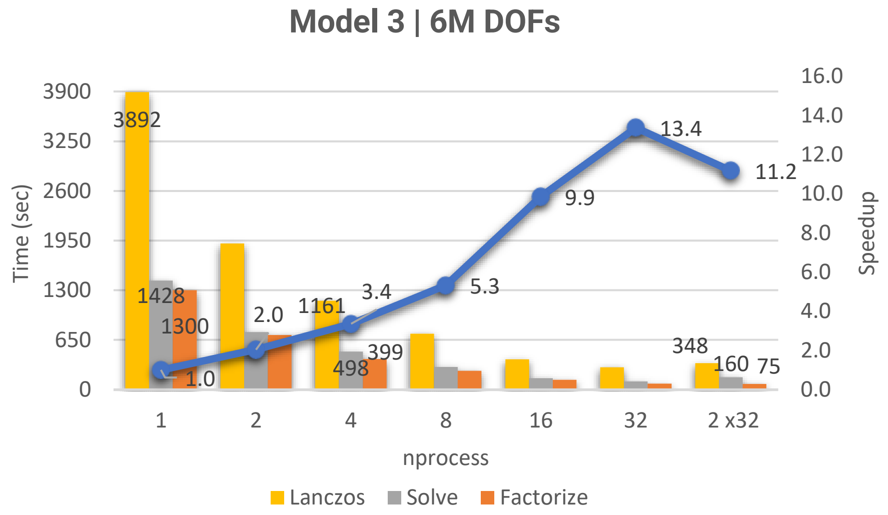
Solid elements

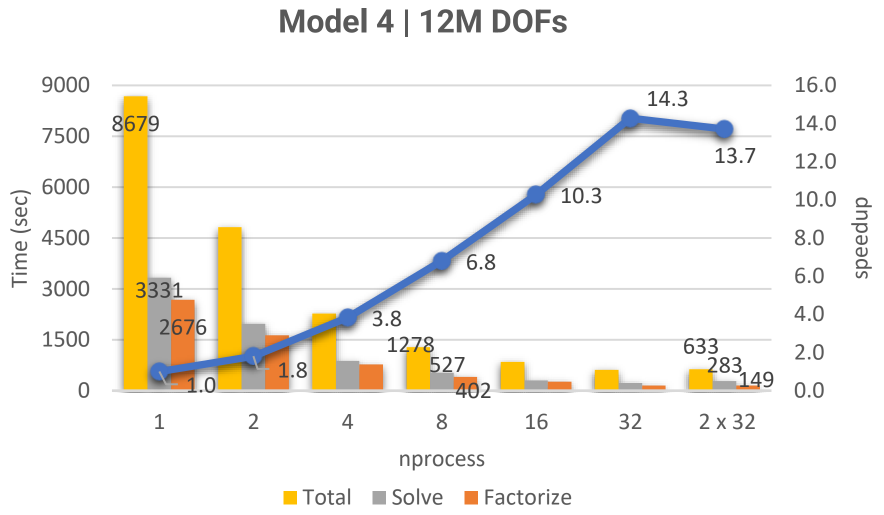




Results (2/2)

Shell elements





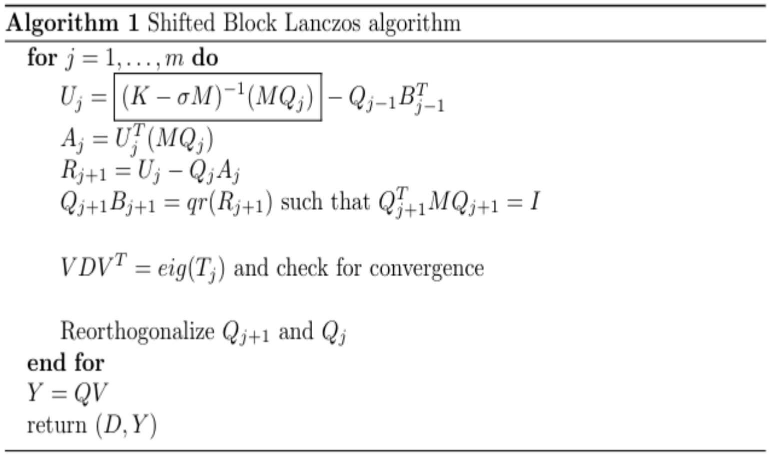
BLR usage in Lanczos

One factorization and many solve steps

MUMPS Block Low-Rank feature

- ✓ Reduce the number of FLOPS in factorization and solve operations
- ✓ Reduce memory footprint

- × Correlation between BLR parameter ε and the accuracy of the eigenvalues is unknown
- × Reduced number of FLOPS may not result in proportionally reduced wall time



Algorithm 1 Shifted Block Lanczos algorithm

```
for $j = 1, \ldots, m$ do
    $U_j = \boxed{(K - \sigma M)^{-1}(MQ_j)} - Q_{j-1}B_{j-1}^T$
    $A_j = U_j^T(MQ_j)$
    $R_{j+1} = U_j - Q_jA_j$
    $Q_{j+1}B_{j+1} = qr(R_{j+1})$ such that $Q_{j+1}^TMQ_{j+1} = I$

    $VDV^T = eig(T_j)$ and check for convergence

    Reorthogonalize $Q_{j+1}$ and $Q_j$
end for
$Y = QV$
return $(D, Y)$
```

Wish list

Sparse right-hand side

- ✓ Reduce cost of scattering data
- ✓ Accelerate the solution phase
- × Only centralized input

Distributed right-hand side

- ✓ Avoid gathering and scattering data
- ✓ Reduce memory footprint on the host
- × Only for dense data

Distributed sparse right-hand side

- ✓ Avoid gathering and scattering data
- ✓ Accelerate the solution phase

Conclusion

- Distributed scaling of MUMPS library result in an eigensolver that can handle large problems efficiently
    - Models with solid elements are benefited the most

- Block Low Rank feature may improve the performance of the eigensolver even more

THANK YOU

Do you have any questions?