MUMPS v 5.1.1

Kostas Sikeli, Phd
Software Development specialist, Optistruct
MUMPS User Days, June 2017
Outline

• Optistruct - Overview
• Comparison of MUMPS 5.1.1 vs 5.0.1
• 64 bit MUMPS
• BLR/ABLR – Preliminary results
• Wish list
OptiStruct - Overview

- Linear and Nonlinear Analysis
- Vibrations and Acoustics
- Fatigue
- Composites
- Multiphysics

Optimization

Large Scale Computing and Parallelization
Optistruct – MUMPS usage

- **Linear static analysis**
  - Symmetric positive definite or indefinite systems

- **Nonlinear static/transient analysis**
  - Unsymmetric systems

- **Lanczos Eigensolver**
  - Symmetric indefinite systems

- **Direct Frequency response Analysis**
  - Symmetric complex systems
Optistruct – MUMPS Benchmark

- **Incore** linear static analysis.

<table>
<thead>
<tr>
<th>Model</th>
<th># of Equations</th>
<th># of Nonzeros</th>
<th># of Elements</th>
<th>Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knuckle</td>
<td>2.8M</td>
<td>117M</td>
<td>650K</td>
<td>Solid (CTETRA10)</td>
</tr>
<tr>
<td>Car body</td>
<td>12M</td>
<td>320M</td>
<td>1.8M</td>
<td>Shell (CQUAD4/CTRIA3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>366K</td>
<td>Solid (CTETRA/CPENTA/CHEXA)</td>
</tr>
</tbody>
</table>
Optistruct – MUMPS Benchmark

Data extracted from MUMPS report

• Time
  • ELAPSED TIME IN ANALYSIS DRIVER
  • ELAPSED TIME IN FACTORIZATION DRIVER
  • ELAPSED TIME IN SOLVE DRIVER

• Total Memory
  • TOTAL space in MBYTES for IC factorization
  • ** EFF Min: Avg. Space in MBYTES per working proc (x # of MPI-Processes)
  • ** Avg. Space in MBYTES per working proc during solve (x # of MPI-Processes)
MUMPS in Optistruct – 5.1.1 vs 5.0.1

- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Significantly improved SMP scaling both in Factorization and Solve phases

![Analysis Wall time - Shell](image)

![Analysis Wall time - Solid](image)
MUMPS in Optistruct – 5.1.1 vs 5.0.1

- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Significantly improved SMP scaling both in Factorization and Solve phases

### Factorization Wall time (s) - Shell

<table>
<thead>
<tr>
<th># of mpi</th>
<th>5.1.1</th>
<th>5.0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>335</td>
<td>328</td>
</tr>
<tr>
<td>2</td>
<td>168</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>108</td>
<td>116</td>
</tr>
<tr>
<td>8</td>
<td>77</td>
<td>72</td>
</tr>
<tr>
<td>16</td>
<td>55</td>
<td>60</td>
</tr>
</tbody>
</table>

### Factorization Wall time (s) - Solid

<table>
<thead>
<tr>
<th># of mpi</th>
<th>5.1.1</th>
<th>5.0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7172</td>
<td>7072</td>
</tr>
<tr>
<td>2</td>
<td>4567</td>
<td>3817</td>
</tr>
<tr>
<td>4</td>
<td>2515</td>
<td>3230</td>
</tr>
<tr>
<td>8</td>
<td>2227</td>
<td>2302</td>
</tr>
<tr>
<td>12</td>
<td>1248</td>
<td>1316</td>
</tr>
</tbody>
</table>
MUMPS in Optistruct – 5.1.1 vs 5.0.1

- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Significantly improved SMP scaling both in Factorization and Solve phases (even more potential with aggressive setting)

---

**Factorization Wall time (s) - Shell**

<table>
<thead>
<tr>
<th># of threads</th>
<th>5.1.1</th>
<th>5.0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>335</td>
<td>328</td>
</tr>
<tr>
<td>2</td>
<td>225</td>
<td>270</td>
</tr>
<tr>
<td>4</td>
<td>169</td>
<td>201</td>
</tr>
<tr>
<td>8</td>
<td>163</td>
<td>207</td>
</tr>
<tr>
<td>16</td>
<td>163</td>
<td>201</td>
</tr>
</tbody>
</table>

-2.1\% 16.7\% 15.9\% 21.2\% 18.9\%

---

**Factorization Wall time (s) - Solid**

<table>
<thead>
<tr>
<th># of threads</th>
<th>5.1.1</th>
<th>5.0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7172</td>
<td>7072</td>
</tr>
<tr>
<td>2</td>
<td>3605</td>
<td>3729</td>
</tr>
<tr>
<td>4</td>
<td>1940</td>
<td>2131</td>
</tr>
<tr>
<td>8</td>
<td>1073</td>
<td>1286</td>
</tr>
<tr>
<td>12</td>
<td>779</td>
<td>1010</td>
</tr>
</tbody>
</table>

-1.4\% 3.3\% 9.0\% 16.6\% 22.9\%
Optistruct – MUMPS 64bit (full)

• Motivation
  • Industrial models challenge 32bit integer capacity
  • Quick solution: Promote all integers to 64

• MPI-Free Implementation
  • Compile Fortran source forcing all integers to be 64bit (-i8 or -fdefault-integer-8 in OPTF)
  • Compile C source forcing all integers to be 64bit (-DINTSIZE64 in OPTC)
  • Link with 64bit BLAS libraries

• MPI : Intel supports 64 bit integer mpi with option –ilp64 in mpirun command, but with the following limitation
  • In REDUCE operations MPI_2INTEGER data type is not promoted, send and receive buffers must be INTEGER(4)
  • For custom reduction functions used in REDUCE operations the integer arguments must be INTEGER(4)
Optistruct – MUMPS 64bit (full)

i. bcast_errors.F@16 : INTEGER IN(2), OUT(2) --> INTEGER(4) IN(2), OUT(2)

ii. tools_common.F@269 : INTEGER TEMP1(2), TEMP2(2) --> INTEGER(4) TEMP1(2), TEMP2(2)

iii. [dzsc]fac_scalings_simScale_util.F@23 : INTEGER IWRK(IWSZ) --> INTEGER(4) IWRK(IWSZ)

iv. [dzsc]fac_scalings_simScale_util.F@433-436: INTEGER LEN, INV(2*LEN), INOUTV(2*LEN), DTYPE --> INTEGER(4) LEN, INV(2*LEN), INOUTV(2*LEN), DTYPE

v. [dzsc]fac_scalings_simScale_util.F@466 : INTEGER IW(IWSZ) --> INTEGER(4) IW(IWSZ)

vi. [dzsc]fac_scalings_simScale_util.F@923 : INTEGER IWRK(IWSZ) --> INTEGER(4) IWRK(IWSZ)

vii. [dzsc]mumps_driver.F@1642 : INTEGER TMP1(2),TMP(2) --> INTEGER(4) TMP1(2),TMP(2)

• MUMPS 5.0.1 : Apply above changes manually

• MUMPS 5.1.1 : -DWORKAROUNDINTELILP64MPI2INTEGER
Optistruct – MUMPS 64 bit (selective)

- MUMPS 5.0.1
  - NZ -> 32 bit
    - Stiffness definitely less than 2 billion
  - 32bit integer for internal symbolic analysis
    - No more than 1 billion terms
    - Even smaller model for unsymmetric matrix from nonlinear with friction
  - Interface to METIS 32bit API
    - Need scotch sometimes

- MUMPS 5.1.1
  - NZ -> 32 bit, NNZ -> 64bit
    - Backward compatible
    - Solve stiffness with more than 2 billion terms if NNZ is used
  - 64bit integer for internal symbolic analysis
    - Solve more than 1 billion terms
  - Interface to METIS 64bit API
    - Avoid METIS overflow
    - Better performance
MUMPS in Optistruct – selective 64 bit vs full

- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.
MUMPS in Optistruct – selective 64 bit vs full

- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

### Factorization Wall time (s) - Shell

<table>
<thead>
<tr>
<th># of mpi x # of threads</th>
<th>OS32</th>
<th>OS64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x16</td>
<td>163</td>
<td>157</td>
</tr>
<tr>
<td>2x8</td>
<td>82</td>
<td>81</td>
</tr>
<tr>
<td>4x4</td>
<td>59</td>
<td>61</td>
</tr>
<tr>
<td>8x2</td>
<td>49</td>
<td>58</td>
</tr>
<tr>
<td>16x1</td>
<td>55</td>
<td>89</td>
</tr>
</tbody>
</table>

### Factorization Wall time (s) - Solid

<table>
<thead>
<tr>
<th># of mpi x # of threads</th>
<th>OS32</th>
<th>OS64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x12</td>
<td>780</td>
<td>776</td>
</tr>
<tr>
<td>2x6</td>
<td>942</td>
<td>921</td>
</tr>
<tr>
<td>4x3</td>
<td>1020</td>
<td>1262</td>
</tr>
<tr>
<td>6x2</td>
<td>1011</td>
<td>1118</td>
</tr>
<tr>
<td>12x1</td>
<td>1248</td>
<td>1257</td>
</tr>
</tbody>
</table>
MUMPS in Optistruct – selective 64 bit vs full

- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

### Solve Wall time (s) - Shell

<table>
<thead>
<tr>
<th># of mpi x # of threads</th>
<th>OS32</th>
<th>OS64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x12</td>
<td>10.7</td>
<td>10.8</td>
</tr>
<tr>
<td>2x6</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>4x3</td>
<td>6.5</td>
<td>5.2</td>
</tr>
<tr>
<td>8x2</td>
<td>2.6</td>
<td>3.5</td>
</tr>
<tr>
<td>16x1</td>
<td>3.3</td>
<td>3.2</td>
</tr>
</tbody>
</table>

### Solve Wall time (s) - Solid

<table>
<thead>
<tr>
<th># of mpi x # of threads</th>
<th>OS32</th>
<th>OS64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x12</td>
<td>9.2</td>
<td>9.6</td>
</tr>
<tr>
<td>2x6</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td>4x3</td>
<td>5.2</td>
<td>4.8</td>
</tr>
<tr>
<td>6x2</td>
<td>5.3</td>
<td>4.6</td>
</tr>
<tr>
<td>12x1</td>
<td>5.8</td>
<td>5.4</td>
</tr>
</tbody>
</table>
MUMPS in Optistruct – selective 64 bit vs full

- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

![Total Memory Estimate (MB) - Shell](image1)

![Total Memory Estimate (MB) - Solid](image2)
MUMPS in Optistruct – BLR/ABLR

• Analysis phase is a bit slower (for solid).
• Factorization and Solve phases up to 3 times faster (for solid)
• Reduced accuracy of the solution. Solution changes with the number of MPI-processes

### Analysis Wall time (s) - Shell

<table>
<thead>
<tr>
<th></th>
<th>BLR</th>
<th>ABLR</th>
<th>5.1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x16</td>
<td>84</td>
<td>86</td>
<td>80</td>
</tr>
<tr>
<td>2x8</td>
<td>83</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>4x4</td>
<td>91</td>
<td>86</td>
<td>76</td>
</tr>
<tr>
<td>8x2</td>
<td>99</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td>16x1</td>
<td>108</td>
<td>108</td>
<td>95</td>
</tr>
</tbody>
</table>

### Analysis Wall time (s) - Solid

<table>
<thead>
<tr>
<th></th>
<th>BLR</th>
<th>ABLR</th>
<th>5.1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x12</td>
<td>73</td>
<td>72</td>
<td>55</td>
</tr>
<tr>
<td>2x6</td>
<td>72</td>
<td>72</td>
<td>53</td>
</tr>
<tr>
<td>4x3</td>
<td>72</td>
<td>72</td>
<td>53</td>
</tr>
<tr>
<td>6x2</td>
<td>76</td>
<td>76</td>
<td>57</td>
</tr>
<tr>
<td>12x1</td>
<td>73</td>
<td>73</td>
<td>55</td>
</tr>
</tbody>
</table>
MUMPS in Optistruct – BLR/ABLR

- Analysis phase is a bit slower (for solid).
- Factorization and Solve phases up to 3 times faster (for solid)
- Reduced accuracy of the solution. Solution changes with the number of MPI-processes
MUMPS in Optistruct – BLR/ABLR

- Analysis phase is a bit slower (for solid).
- Factorization and Solve phases up to 3 times faster (for solid)
- Reduced accuracy of the solution. Solution changes with the number of MPI-processes
Optistruct – Wish list

• Robust detection of ill-conditioned (near-singular) matrix
  • Often MUMPS provides physically meaningless solutions to ill-conditioned systems.
  • Preferable just error out.
  • Possibly a parameter similar to MAXRATIO used for LDLᵀ i.e \( r = \max(\frac{\text{Diag}(A)_{ii}}{D_i}) \)
Optistruct – Wish list

• Detection of NaN in input matrix
  • A failsafe in case input is erroneous.
  • Preferable to error out instead of delivering an erroneous solution

```
STATISTICS PRIOR SOLVE PHASE ............
NUMBER OF RIGHT-HAND-SIDES           =  1
BLOCKING FACTOR FOR MULTIPLE RHS     =  1
ICNTL (9)                             =  1
   --- (10)                           =  0
   --- (11)                           =  0
   --- (20)                           =  0
   --- (21)                           =  0
   --- (30)                           =  0

LEAVING SOLVE (MPI41C) WITH
RHS (first column)
  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00
  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00

VECTOR SOLUTION FOR COLUMN 1
RHS
  NaN    NaN    NaN    NaN    NaN
  NaN    NaN    NaN    NaN    NaN
** Rank of processor needing largest memory in solve :  0
** Space in MB/RES used by this processor for solve   :  174
** Avg. Space in MB/RES per working proc during solve:  174
```
Optistruct – Wish list

• Suggestion of optimal reordering
THANK YOU

Questions?

MUMPS Team, thank you for a great linear equation solver!