Singular Value Decomposition of Several Complex Matrices on a Multicore CPU and CUDA-enabled GPUs

Evgueni Ovtchinnikov

Numerical Analysis Group
STFC Rutherford Appleton Laboratory
Outline

▶ Target matrices and architecture
▶ CUDA GPU basics
▶ SVD basics
▶ SVD solvers: the bottleneck
▶ SVD solvers: a solution
▶ Recursive decomposition SVD
▶ Numerical tests
▶ Summary
Target matrices

Target matrices came from Tensor Network Theory.

- complex $m$-by-$n$ with $m \geq n$
- nearly rectangular: $m/n$ is not large
- sizes range $10^3 - 10^4$
- singular values $\sigma_k$ nearly exponential: $\sigma_k \sim q^k$, $q < 1$
- all singular values (and vectors) above $10^{-10}\sigma_1$ needed
- few dozens to SVD simultaneously

Matrices for testing provided by Prof. Dieter Jaksch team, Oxford University.
Target architecture

A multicore CPU with one or more CUDA-enabled GPUs.
GPU basics

- lots of cores
  - K40m: $15 \times 192$

- relatively small fast memory
  - K40m: $15 \times (256K + 48K + 16K)$

$\Rightarrow$ GPU implementations of numerical algorithms are more often than not memory bound.
CUDA basics

- GPU executes simultaneously a large number of threads
- CUDA C kernel is a set of instructions for each thread
- threads can be arranged in blocks of not more than 1024
- threads within the block can exchange data via shared memory
- CUDA kernels can be arranged into *streams*: kernels of each stream are executed one after another but asynchronously with kernels of other streams

```c
my_kernel<<< nb, nt, sm, sid >>>(arg1,arg2,...);
// nb: number of thread blocks
// nt: number of threads in a block
// sm: shared memory size (bytes)
// sid: stream id
```
SVD basics
SVD basics: notation

$A^*$ is the complex transpose of $A$, i.e. the elements $a_{ij}^*$ of $A^*$ are

$$a_{ij}^* = \overline{a_{ji}}.$$

$I$ is the identity matrix: its elements are

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

$U$ is called a unitary matrix if $U^* U = I$. 
SVD basics: generalization of the eigen-decomposition

- If $A = A^*$ then there exists a unitary $U$ such that

$$A = U\Lambda U^*,$$

where $\Lambda$ is a diagonal matrix. Columns of $U$ are called eigenvectors and the diagonal elements of $\Lambda$ eigenvalues.
SVD basics: generalization of the eigen-decomposition

- If $A = A^*$ then there exists a unitary $U$ such that

$$A = U\Lambda U^*,$$

where $\Lambda$ is a diagonal matrix. Columns of $U$ are called eigenvectors and the diagonal elements of $\Lambda$ eigenvalues.

- For any $A$ there exist unitary $U$ and $V$ such that

$$A = U\Sigma V^*$$

where $\Sigma$ is a diagonal matrix with positive elements on the diagonal. Columns of $U$ and $V$ are called left and right singular vectors and the diagonal elements of $\Sigma$ singular values.
SVD basic: how it is not done

\[ A = U \Sigma V^* \Rightarrow \]

\[ AA^* = U \Sigma^2 U^* = U \Lambda U^* \]

\[ A = U \Sigma V^* \Rightarrow \]

\[ AV = U \Sigma \]
\[ A^* U = V \Sigma \]

\[
\begin{bmatrix}
A & A^*
\end{bmatrix}
\begin{bmatrix}
U & V
\end{bmatrix} =
\begin{bmatrix}
U & V
\end{bmatrix} \Sigma
\]
SVD basic: how it is done

- Find unitary $U_A$ and $V_A$ such that $B = U_A^* AV_A$ is real bi-diagonal.

- SVD $B = U_B \Sigma V_B^*$.

- $A = U \Sigma V^*$, where $U = U_A U_B$, $V = V_A V_B$. 
SVD solvers: software

- CPU:
  - zgesvd
  - zgesdd
SVD solvers: software

- CPU:
  - zgesvd
  - zgesdd

- GPU:
  - magma_zgesvd
  - magma_zgesdd
# SVD solvers: the bottleneck

## Percentage of time for reduction to bi-diagonal form

<table>
<thead>
<tr>
<th>matrix size</th>
<th>zgesdd</th>
<th>magma_zgesdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$500 \times 500$</td>
<td>52</td>
<td>72</td>
</tr>
<tr>
<td>$1000 \times 1000$</td>
<td>51</td>
<td>69</td>
</tr>
<tr>
<td>$2006 \times 2002$</td>
<td>56</td>
<td>72</td>
</tr>
<tr>
<td>$4006 \times 4002$</td>
<td>69</td>
<td>74</td>
</tr>
<tr>
<td>$6010 \times 6006$</td>
<td>68</td>
<td>76</td>
</tr>
<tr>
<td>$8008 \times 8004$</td>
<td>77</td>
<td>74</td>
</tr>
</tbody>
</table>
SVD solvers: the bottleneck

- \( U_A = U_1 U_2 \cdots U_{n-1} \) and \( V_A = V_1 V_2 \cdots V_{n-1} \), where \( U_i \) and \( V_i \) are Householder reflections:

\[
U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^* 
\]
Singular Value Decomposition of Several Complex Matrices on a Multicore CPU and CUDA-enabled GPUs

E Ovtchinnikov

SVD solvers: the bottleneck

- $U_A = U_1 U_2 \cdots U_{n-1}$ and $V_A = V_1 V_2 \cdots V_{n-1}$, where $U_i$ and $V_i$ are Householder reflections:

  \[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^* \]

- For $m \times n$ matrix $A$, $u_i$ and $v_i$ have $m - i + 1$ and $n - i$ non-zeros, hence the reduction to bi-diagonal form requires about

  \[ \frac{4mn^2}{3} \]

  multiplications
SVD solvers: the bottleneck

- $U_A = U_1 U_2 \cdots U_{n-1}$ and $V_A = V_1 V_2 \cdots V_{n-1}$, where $U_i$ and $V_i$ are Householder reflections:
  
  \[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^* \]

- for $m \times n$ matrix $A$, $u_i$ and $v_i$ have $m - i + 1$ and $n - i$ non-zeros, hence the reduction to bi-diagonal form requires about
  
  \[ \frac{4mn^2}{3} \]

  multiplications and the same amount of memory transactions.
SVD solvers: a solution

Use block Householder reflections

\[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^*, \]

where now \( u_i \) and \( v_i \) have \( h > 1 \) columns.
SVD solvers: a solution

Use block Householder reflections

\[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^*, \]

where now \( u_i \) and \( v_i \) have \( h > 1 \) columns.

- the same number of multiplications,
SVD solvers: a solution

Use block Householder reflections

\[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^*, \]

where now \( u_i \) and \( v_i \) have \( h > 1 \) columns.

- the same number of multiplications,
- the number of memory transactions is reduced by \( h \),
SVD solvers: a solution

Use block Householder reflections

\[ U_i = I - u_i u_i^*, \quad V_i = I - v_i v_i^*, \]

where now \( u_i \) and \( v_i \) have \( h > 1 \) columns.

- the same number of multiplications,
- the number of memory transactions is reduced by \( h \),
- but \( B = U_A^* A V_A \) is block-bidiagonal and complex.
### SVD solvers: a solution

#### Reduction to block-bidiagonal form (sec)

<table>
<thead>
<tr>
<th>matrix size</th>
<th>$h = 1$</th>
<th>$h = 4$</th>
<th>$h = 8$</th>
<th>$h = 16$</th>
<th>$h = 24$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$500 \times 500$</td>
<td>0.08</td>
<td>0.12</td>
<td>0.08</td>
<td>0.07</td>
<td>0.04</td>
</tr>
<tr>
<td>$1000 \times 1000$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.18</td>
<td>0.11</td>
<td>0.13</td>
</tr>
<tr>
<td>$2006 \times 2002$</td>
<td>1.60</td>
<td>1.19</td>
<td>0.62</td>
<td>0.36</td>
<td>0.35</td>
</tr>
<tr>
<td>$4006 \times 4002$</td>
<td>8.79</td>
<td>5.90</td>
<td>3.04</td>
<td>1.77</td>
<td>1.44</td>
</tr>
<tr>
<td>$6010 \times 6006$</td>
<td>25.3</td>
<td>18.00</td>
<td>9.59</td>
<td>5.70</td>
<td>4.31</td>
</tr>
<tr>
<td>$8012 \times 8004$</td>
<td>57.0</td>
<td>38.67</td>
<td>18.60</td>
<td>11.71</td>
<td>8.97</td>
</tr>
</tbody>
</table>
SVD solvers: a solution

Reduction to block-bidiagonal form (sec)

<table>
<thead>
<tr>
<th>matrix size</th>
<th>$h = 1$</th>
<th>$h = 4$</th>
<th>$h = 8$</th>
<th>$h = 16$</th>
<th>$h = 24$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$500 \times 500$</td>
<td>0.08</td>
<td>0.12</td>
<td>0.08</td>
<td>0.07</td>
<td>0.04</td>
</tr>
<tr>
<td>$1000 \times 1000$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.18</td>
<td>0.11</td>
<td>0.13</td>
</tr>
<tr>
<td>$2006 \times 2002$</td>
<td>1.60</td>
<td>1.19</td>
<td>0.62</td>
<td>0.36</td>
<td>0.35</td>
</tr>
<tr>
<td>$4006 \times 4002$</td>
<td>8.79</td>
<td>5.90</td>
<td>3.04</td>
<td>1.77</td>
<td>1.44</td>
</tr>
<tr>
<td>$6010 \times 6006$</td>
<td>25.3</td>
<td>18.00</td>
<td>9.59</td>
<td>5.70</td>
<td>4.31</td>
</tr>
<tr>
<td>$8012 \times 8004$</td>
<td>57.0</td>
<td>38.67</td>
<td>18.60</td>
<td>11.71</td>
<td>8.97</td>
</tr>
</tbody>
</table>

But for $h > 1$, we have to SVD a complex block-bidiagonal matrix, which is much more complicated and computationally expensive: had to use $h = 8$ in tests :(((
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix}
  d_1 & e_1 \\
  d_2 & e_2 \\
  d_3 & e_3 \\
  d_4 & e_4 \\
  d_5 & e_5 \\
  d_6 & e_6 \\
  d_7 &
\end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix}
  d_1 & e_1 &\vline&
  d_2 & e_2 &\vline&
  d_3 & e_3 \\
  \hline
  d_4 & e_4 &\vline&
  d_5 & e_5 &\vline&
  d_6 & e_6 &\vline&
  d_7 \\
\end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix} d_1 & e_1 & & & & & \\ & d_2 & e_2 & & & & \\ & & d_3 & & & & \\ & & & d_4 & e_4 & & \\ & & & & d_5 & e_5 & \\ & & & & & d_6 & e_6 \\ & & & & & & d_7 \end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix} d_1 & e_1 & e_2 & e_4 \\ d_2 & e_2 & d_3 \\ d_5 & e_5 & d_6 & e_6 \\ d_7 \end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix}
  d_1 & e_1 & & & \\
  & d_2 & e_2 & & \\
  & & d_3 & e_3 & \\
  & & & d_4 & e_4 \\
  & & & & d_5 & e_5 \\
  & & & & & d_6 & e_6 \\
  & & & & & & d_7
\end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

$$B = \begin{bmatrix}
  d_1 & e_1 & \quad & \quad & \quad & \\
  d_2 & e_2 & \quad & \quad & \quad & \\
  d_3 & e_3 & \quad & \quad & \quad & \\
  d_4 & e_4 & \quad & \quad & \quad & \\
  d_5 & e_5 & \quad & \quad & \quad & \\
  d_6 & e_6 & \quad & \quad & \quad & \\
  d_7 & \quad & \quad & \quad & \quad & \\
\end{bmatrix}$$
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix} d_1 & e_1 & \ & \ \\ & d_2 & e_2 & \ \\ & & d_3 \end{bmatrix} \begin{bmatrix} & & & \ \\ & & & e_3 \ \\ & & & \ \\ & e_4 & \ & \\ & d_5 & e_5 & \ \\ & & d_6 & e_6 \ \\ & & & d_7 \end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix}
  U_1 \Sigma_1 V_1^* & e_3 \\
  d_4 & U_2 \Sigma_2 V_2^*
\end{bmatrix} \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ B = \begin{bmatrix} U_1 & \Sigma_1 & V_1^* \\ & e_3 & \\
U_2 & \Sigma_2 & V_2^* \end{bmatrix} = U_C \begin{bmatrix} \tilde{\Sigma} & b \\ c & \end{bmatrix} V_C^* \]

\[ \tilde{\Sigma} = \begin{bmatrix} \Sigma_1 \\ \Sigma_2 \end{bmatrix}, \quad U_C^* U_C = I, \quad V_C^* V_C = I. \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ C = \begin{bmatrix} \tilde{\Sigma} & b \\ c & \end{bmatrix}. \]
Recursive decomposition SVD for a block-bidiagonal matrix

\[ C = \begin{bmatrix} \tilde{\Sigma} & b \\ c & \end{bmatrix}. \]

Singular values of \( C \) are the positive eigenvalues of

\[ H = \begin{bmatrix} C^* & C \\ C^* & \end{bmatrix} = \begin{bmatrix} \tilde{\Sigma} & b \\ b^* & c^* \\ \end{bmatrix}. \]
Recursive decomposition SVD for a block-bidiagonal matrix

If $\sigma$ is not on the diagonal of $\tilde{\Sigma}$, then there exists non-degenerate matrix $W(\sigma)$ such that

$$W(\sigma)^*(H - \sigma I)W(\sigma) = S(\sigma) - \sigma I,$$

where

$$S(\sigma) = \begin{bmatrix} \frac{1}{\sigma} \tilde{\Sigma}^2 & \ast \\ \ast & s(\sigma) \end{bmatrix}, \quad s(\sigma) = \frac{1}{\sigma} c^* c - \sum_i \frac{\sigma}{\tilde{\sigma}_i^2 - \sigma^2} b_i^* b_i.$$
Recursive decomposition SVD for a block-bidiagonal matrix

If $\sigma$ is not on the diagonal of $\tilde{\Sigma}$, then there exists non-degenerate matrix $W(\sigma)$ such that

$$W(\sigma)^* (H - \sigma I) W(\sigma) = S(\sigma) - \sigma I,$$

where

$$S(\sigma) = \begin{bmatrix} \frac{1}{\sigma} \tilde{\Sigma}^2 \\ s(\sigma) \end{bmatrix}, \quad s(\sigma) = \frac{1}{\sigma} c^* c - \sum_i \frac{\sigma}{\tilde{\sigma}_i^2 - \sigma^2} b_i^* b_i.$$

$\sigma$ is a singular value of $C$ if and only if it is an eigenvalue of $s(\sigma)$. 
Recursive decomposition SVD for a block-bidiagonal matrix

If \( \sigma \) is not on the diagonal of \( \tilde{\Sigma} \), then there exists non-degenerate matrix \( W(\sigma) \) such that

\[
W(\sigma)^* (H - \sigma I) W(\sigma) = S(\sigma) - \sigma I,
\]

where

\[
S(\sigma) = \begin{bmatrix}
\frac{1}{\sigma} \tilde{\Sigma}^2 \\
s(\sigma)
\end{bmatrix} ,
\]

\[
s(\sigma) = \frac{1}{\sigma} c^* c - \sum_i \frac{\sigma}{\tilde{\sigma}_i^2 - \sigma^2} b_i^* b_i.
\]

\( \sigma \) is a singular value of \( B \) if and only if it is an eigenvalue of \( s(\sigma) \).
Recursive decomposition SVD for a block-bidiagonal matrix

If $\sigma$ is not on the diagonal of $\tilde{\Sigma}$, then there exists non-degenerate matrix $W(\sigma)$ such that

$$W(\sigma)^*(H - \sigma I)W(\sigma) = S(\sigma) - \sigma I,$$

where

$$S(\sigma) = \begin{bmatrix} \frac{1}{\sigma} \tilde{\Sigma}^2 & \ast \\ \ast & s(\sigma) \end{bmatrix}, \quad s(\sigma) = \frac{1}{\sigma} c^* c - \sum_i \frac{\sigma}{\tilde{\sigma}_i^2 - \sigma^2} b_i^* b_i.$$

$\sigma$ is a singular value of $A$ if and only if it is an eigenvalue of $s(\sigma)$.
Recursive decomposition SVD for a block-bidiagonal matrix

The algorithm outline

1. If problem size is small enough, use zgesvd or zgesdd.
2. Split the matrix into two submatrices and a column. Perform SVD of the submatrices recursively.
3. Compute blocks $b$ and $c$ of matrix $C$.
4. Compute all singular values and vectors of $C$ by solving independent nonlinear eigenvalue problems $s(\sigma)v = \sigma v$ in parallel.
5. Orthogonalize the singular vectors of $C$.
6. Compute the singular vectors of $B$ from those of $C$ and $U_1$, $U_2$, $V_1$ and $V_2$. 
Recursive decomposition SVD for a block-bidiagonal matrix

Parallel features

GPU:

- cublasZgemm for the orthogonalization of singular vectors of $C$ (Gram-Schmidt) and the computation of singular vectors of $B$
- one CUDA block of $h^2$ threads per singular value for computing $s(\sigma)$

CPU:

- on second and further levels of the decomposition tree several matrices are processed $\Rightarrow$ distribute OMP and MKL threads between them
What it takes

<table>
<thead>
<tr>
<th>step</th>
<th>flops total</th>
<th>kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to block bi-diagonal</td>
<td>$a_1 mn^2$</td>
<td>cublasZgemm</td>
</tr>
<tr>
<td>computing $s(\sigma)$</td>
<td>$a_2 h^2 n^2$</td>
<td>own kernel</td>
</tr>
<tr>
<td>computing eigenvalues and eigenvectors</td>
<td>$a_3 h^3 n \log(n)$</td>
<td>currently on CPU</td>
</tr>
<tr>
<td>of $s(\sigma)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orthonormalization of singular vectors</td>
<td>$a_4 n^3$</td>
<td>cublasZgemm</td>
</tr>
<tr>
<td>of $C$ (block Gram-Schmidt)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>computation of singular</td>
<td>$a_5 n^3$</td>
<td>cublasZgemm</td>
</tr>
<tr>
<td>vectors of $B$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>computation of singular</td>
<td>$a_6 mn^2$</td>
<td>cublasZgemm</td>
</tr>
<tr>
<td>vectors of $A$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

▶ None of $a_i$ is large.
Numerical tests
Numerical tests

Test matrices and their sizes.

- TNT matrices provided by Dr. Sarah Al-Assam, of Prof. Dieter Jaksch team:
  500 × 500
  1000 × 1000
  2006 × 2002
  4006 × 4002

- ‘Pseudo-TNT’ matrices generated by imitating the singular values distribution of TNT matrices:
  6010 × 6006
  8008 × 8004
Numerical tests

**Hardware used.**

CPU: 8-core Intel(R) Xeon(R) E5-2650 v2 @ 2.60GHz

GPU: 2 Tesla K40m GPUs

(Emerald High Performance Computing facility)
Numerical tests

Software tested.

- zgesvd (Intel MKL 11.2)
- zgesdd (Intel MKL 11.2)
- magma_zgesvd (MAGMA 1.6)
- magma_zgesdd (MAGMA 1.6)
- recursive decomposition code, working title svdral

CUDA version: 7.0.
SVD of a single matrix

Timings for a single matrix SVD

- zgesvd
- zgesdd
- magma_zgesvd
- magma_zgesdd
- svdral
Singular Value Decomposition of Several Complex Matrices on a Multicore CPU and CUDA-enabled GPUs

SVD of a single matrix

Timings for smaller matrices

- zgesvd
- zgesdd
- magma_zgesvd
- magma_zgesdd
- svdral

SVD time (sec)

matrix width
Simultaneous SVD of several matrices

- A CUDA stream for each matrix.
- One OMP thread for each matrix.

OpenMP and MKL environment used for simultaneous SVD:

OMP_NUM_THREADS = 8
OMP_NESTED = TRUE
MKL_NUM_THREADS = 2
MKL_DYNAMIC = FALSE
Simultaneous SVD of several matrices
Simultaneous SVD of several matrices
Simultaneous SVD of several matrices

SVD of $6010 \times 6006$ matrices: time per matrix

![Graph showing SVD time per matrix for different numbers of matrices.

- Blue bars represent MAGMA.
- Orange bars represent SVDRAL.

Number of matrices: 1, 2, 4, 8.

SVD time (sec): 0, 5, 10, 15, 20, 25, 30, 35, 40, 45.
Numerical tests

**Compute Utilization (%)**
(NVIDIA Visual Profiler *nvvp*)

- **MAGMA**
- **SVDRAL**

Matrix width: 2002, 4002, 6006
Numerical tests

Percentage of total cublasZgemm time for svdral (NVIDIA profiler nvprof).

![Bar chart showing percentage of total cublasZgemm time for svdral for different matrix widths (2002, 4002, 6006, 8004). The chart indicates the percentage of time taken by cublasZgemm compared to the total time, with the rest of the time accounted for by other operations. The chart shows a clear trend where as the matrix width increases, the percentage of time taken by cublasZgemm also increases.]
Numerical tests

Percentage of total cublasZgemm/v times for \texttt{magma\_zgesdd}
(NVIDIA profiler \texttt{nvprof})

![Bar chart showing the percentage of total cublasZgemm/v times for magma\_zgesdd. The chart displays the percentage of the rest (yellow), \texttt{zgemv} (red), and \texttt{zgemm} (blue) for different matrix widths (2002, 4002, 6006, 8004).]
The accuracy of singular vectors

\[ Av_i = \sigma_i u_i, \quad A^* u_i = \sigma_i v_i \]

\[ \tilde{u}_i \approx u_i, \quad \tilde{v}_i \approx v_i \]

\[ r_i = \sqrt{\| A\tilde{v}_i - \tilde{\sigma}_i \tilde{u}_i \|^2 + \| A^* \tilde{u}_i - \tilde{\sigma}_i \tilde{v}_i \|^2} \]

\[ \sqrt{\| \tilde{u}_i - u_i \|^2 + \| \tilde{v}_i - v_i \|^2} \leq \frac{r_i}{\delta_i}, \quad \delta_i = \min_{j \neq i} |\tilde{\sigma}_i - \sigma_j| \]
The accuracy of singular vectors

Singular values of 4006-by-4002 matrix
The accuracy of singular vectors

Residuals for right singular vectors computed by zgesvd

residual norms for v (zgesvd)
The accuracy of singular vectors

Residuals for right singular vectors computed by \texttt{magma_zgesvd}
The accuracy of singular vectors

Residuals for right singular vectors computed by zgesdd

residual norms for v (zgesdd)

0  500  1000  1500  2000  2500  3000  3500  4000  4500
The accuracy of singular vectors

Residuals for right singular vectors computed by \texttt{magma_zgesdd}
The accuracy of singular vectors

Residuals for right singular vectors computed by *svdral*
Summary

- Below size 4K, GPU codes performance is far from impressive, making their comparison pointless.

- For larger sizes, the reduction to block bi-diagonal form leads to improved performance compared to the reduction to bi-diagonal: the larger the size the more visible the improvement.

- Two matrices on two GPUs take about the same time as one matrix on one GPU. With svdral, doing more than one matrix per GPU does not bring tangible gains.
Thank You!