PLASMA, MAGMA, PARSEC

Performance Bounds in Symmetric Eigensolver

Azzam Haidar

INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE

ICL Friday talk,
March 6, 2015

Performance Bounds in Symmetric Eigensolver

Motivation:

- Study the algorithm and what can we expect from it in term of performance. Acceptable or need to think about new algorithm ?

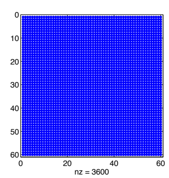
- Analyze the implementation and verify if there is room for optimizations.

General Overview: the Eigenproblem algorithms

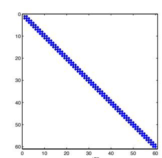
Background:

- Symmetric EVP $Ax = \lambda x$ meaning compute $A = Z \lambda Z^*$
  where λ are the Eigenvalues and Z are the eigenvectors.

1. Tri-Diagonalization Reduction: `transform A to nice form` ☺



==>



$$A = Q\,T\,Q^*$$

2. Solve: `compute the Eigenvalue and Eigenvectors of the tridiagonal`

$$T = E\,\lambda\,E^* \quad \Rightarrow \quad A = Q\,.\,(E\,\lambda\,E^*)\,.\,Q^*$$

3. Back transformation: `update the computed Eigenvectors.`

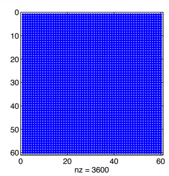
$$Z = Q * E$$

General Overview: the Eigenproblem algorithms

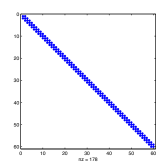
Background:

- Symmetric EVP $Ax = \lambda x$ meaning compute $A = Z \lambda Z^*$ where λ are the Eigenvalues and Z are the eigenvectors.

1. Tri-Diagonalization Reduction: `transform A to nice form` ☺



==>



$$A = Q\,T\,Q^*$$

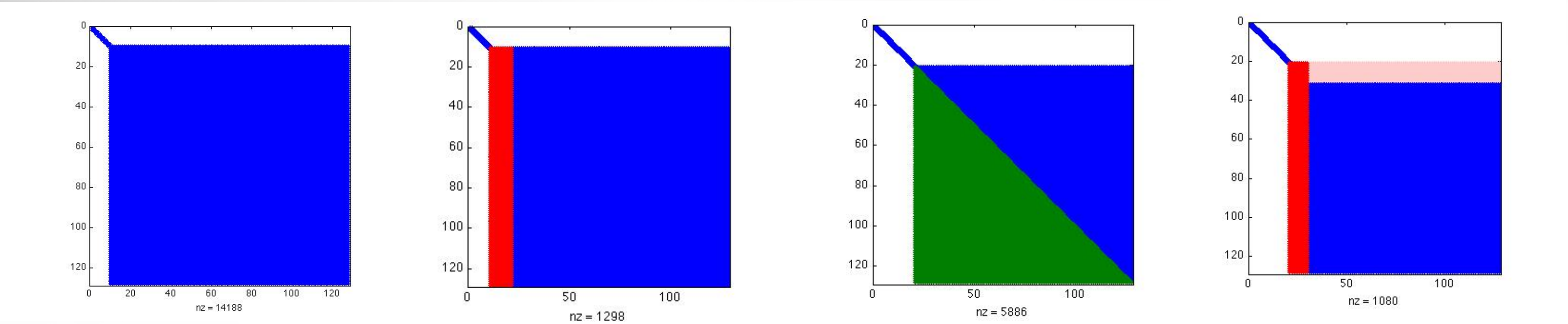
2. Solve: `compute the Eigenvalue and Eigenvectors of the tridiagonal`

$$T = E \lambda E^* \quad \Rightarrow \quad A = Q\,.\,(E \lambda E^*)\,.\,Q^*$$

3. Back transformation: `update the computed Eigenvectors.`

$$Z = Q * E$$

The standard Tridiagonal reduction *dsytrd*



step *k*: **Q A Q^H** then update ➔ **step *k+1***

✷ Characteristics

- Too many Blas-2 op,
- Relies on panel factorization,
- Total cost $4n^3/3$,
- ➔**Bulk sync phases,**
- ➔**Memory bound algorithm.**

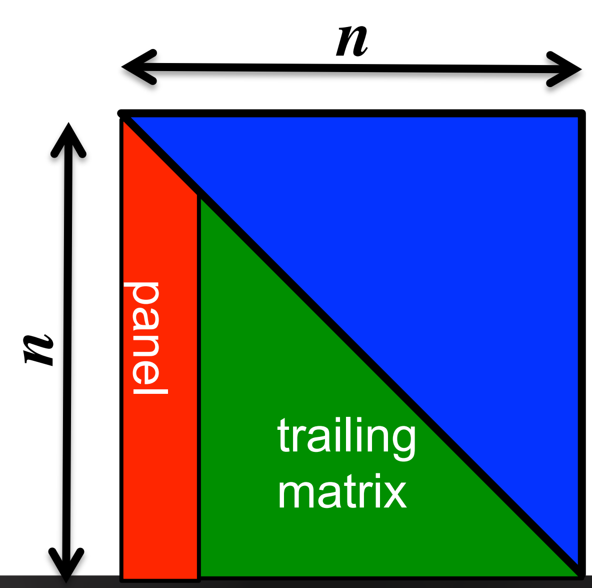
The standard Tridiagonal reduction *dsytrd*

- For each step it's the cost of the panel + cost of update:
  - Each panel is of size n_b column, and each column of the panel requires:
    - 1 SYMV with the trailing matrix, and 6 panel GEMV + O(n)
    - Thus the cost of a panel is: $n_b*(2l^2)$ + O(l).
  - The update **A := A – V*W' – W*V'** consists into:
    - SYR2K to update the trailing matrix, cost= $2*n_b*l^2$

Cost:

For all steps (n/n_b), the trailing matrix size varies from n to n_b by steps of size n_b, where l varies from n to n_b and k varies from $(n-n_b)$ to 2 n_b. Thus, the total cost for the n/n_b steps is:

$$\begin{aligned} flops &\approx 2n_b \sum_{n_b}^{n/n_b} l^2 + 2n_b \sum_{2n_b}^{\frac{n-n_b}{n_b}} k^2 \\ &\approx \tfrac{2}{3} n^3_{\text{symv}} + \tfrac{2}{3} n^3_{\text{syr2k}} \\ &\approx \tfrac{4}{3} n^3. \end{aligned} \qquad (1)$$

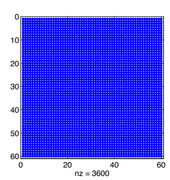


General Overview: the Eigenproblem algorithms

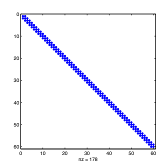
Background:

- Symmetric EVP $Ax = \lambda x$ meaning compute $A = Z \lambda Z^*$
  where λ are the Eigenvalues and Z are the eigenvectors.

1. Tri-Diagonalization Reduction: `transform A to nice form` ☺



==>



$$A = Q\,T\,Q^*$$

2. Solve: `compute the Eigenvalue and Eigenvectors of the tridiagonal`

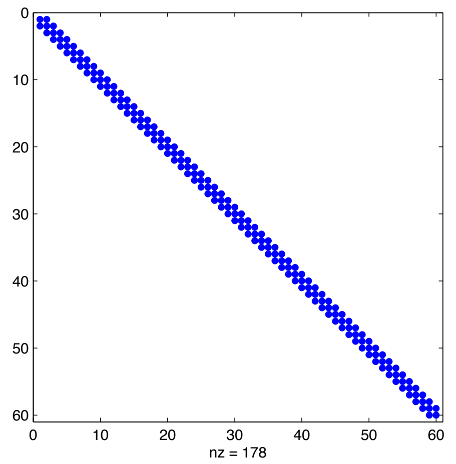
$$T = E\,\lambda\,E^* \quad \Rightarrow \quad A = Q\,.\,(E\,\lambda\,E^*)\,.\,Q^*$$

3. Back transformation: `update the computed Eigenvectors.`

$$Z = Q * E$$

ICL UT

The Divide and Conquer Algorithm *dstedc*

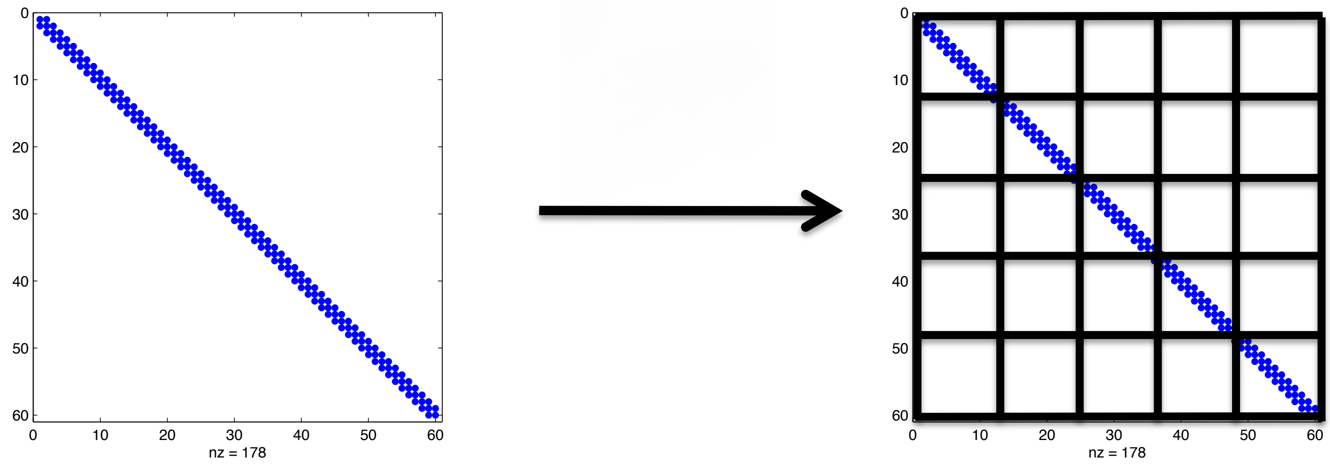


A symmetric tridiagonal eigensolver computes the spectral decomposition of a tridiagonal matrix T such that:

$$T = E\Lambda E^T \text{ with } EE^T = I \qquad (6)$$

where E are the eigenvectors, and Λ are the eigenvalues.

The Divide and Conquer Algorithm *dstedc*

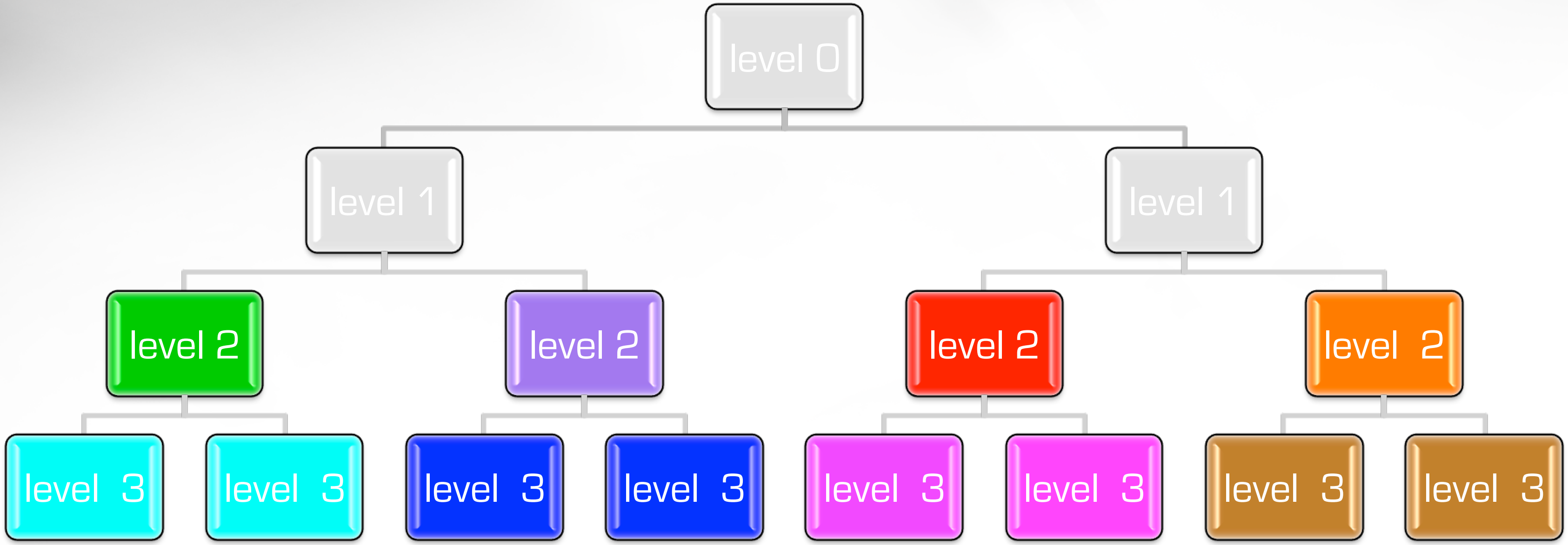


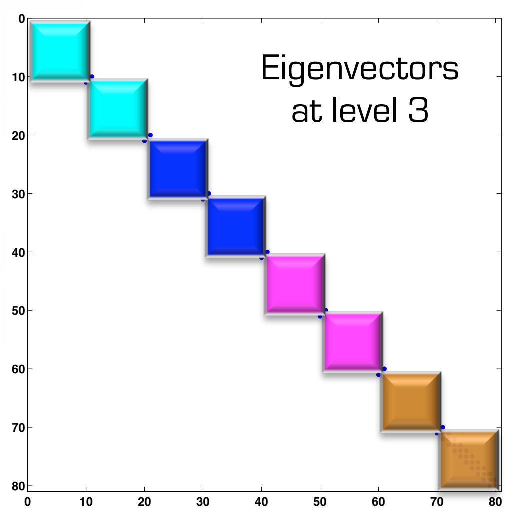
A symmetric tridiagonal eigensolver computes the spectral decomposition of a tridiagonal matrix T such that:

$$T = E \Lambda E^T \text{ with } E E^T = I \tag{6}$$

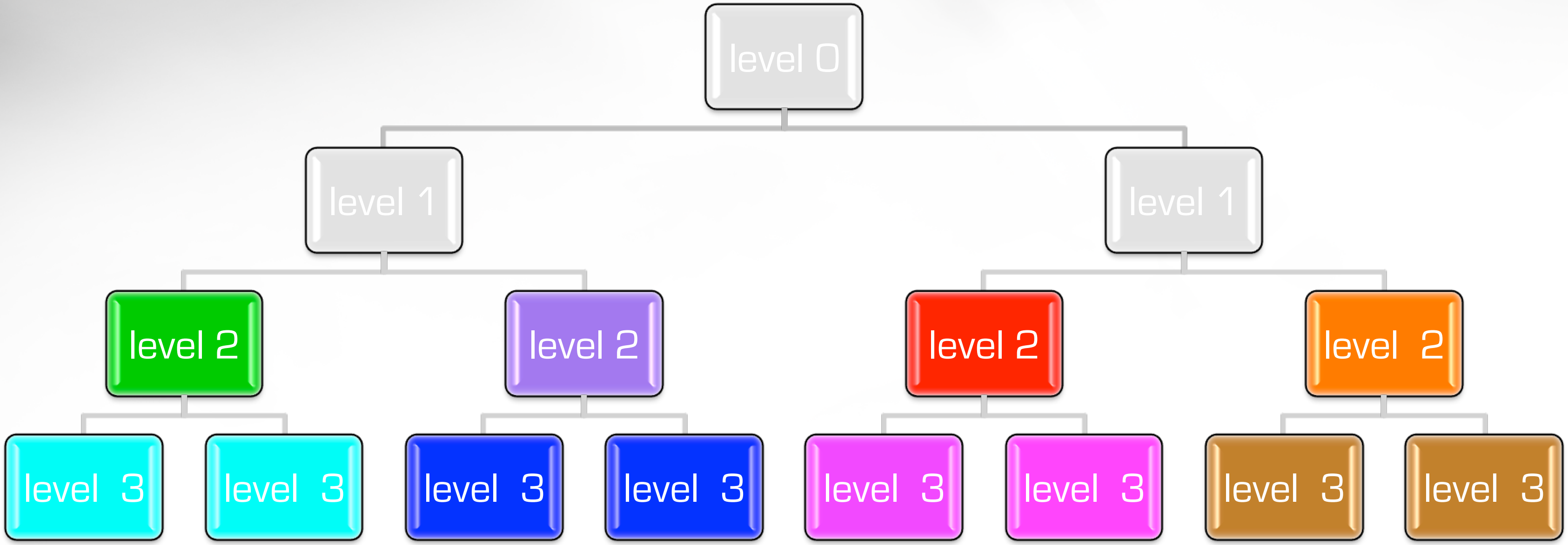
where E are the eigenvectors, and Λ are the eigenvalues.

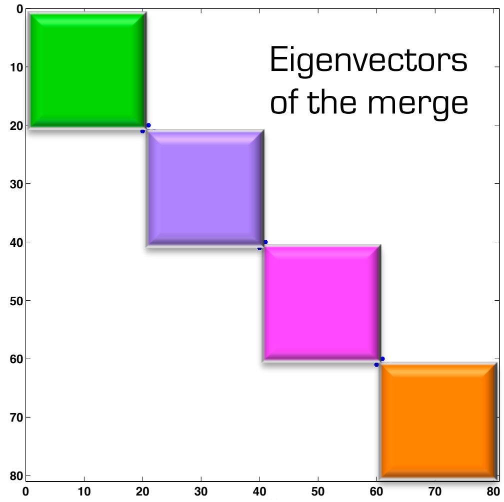
The Divide and Conquer Algorithm *dstedc*



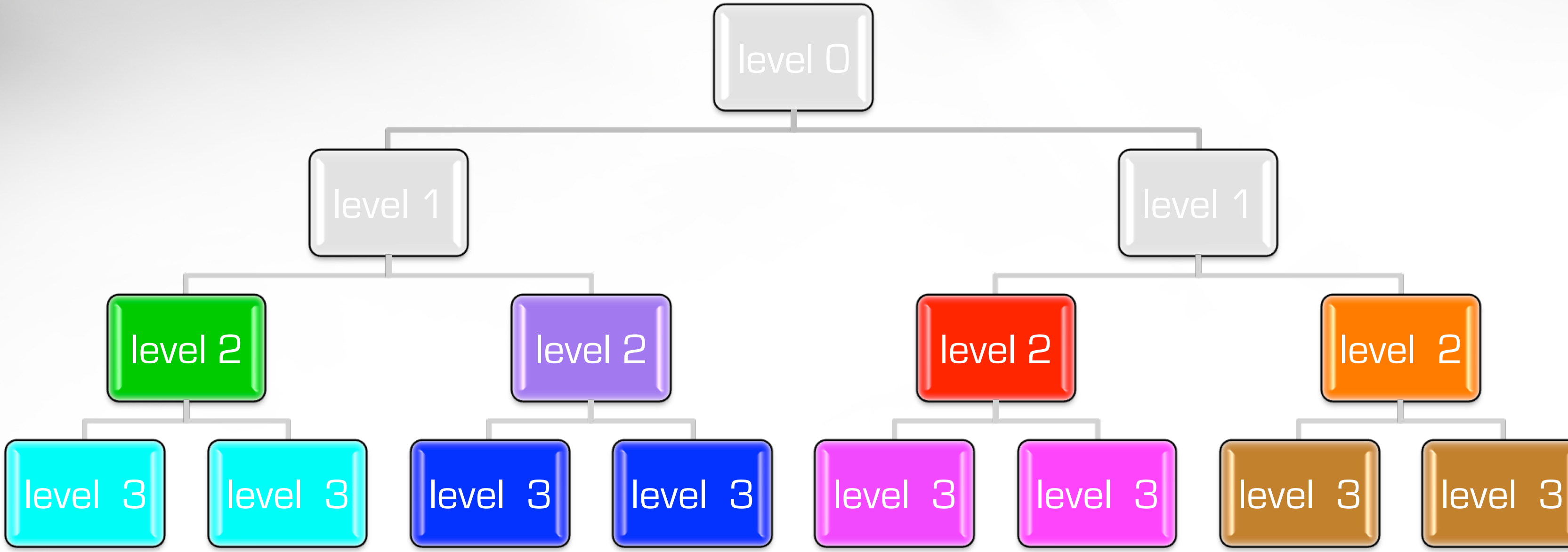


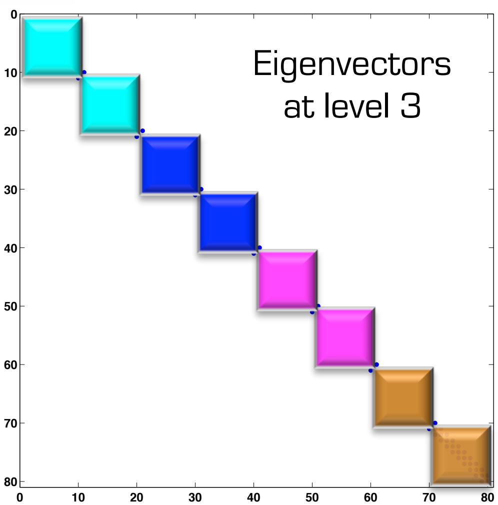
The Divide and Conquer Algorithm *dstedc*



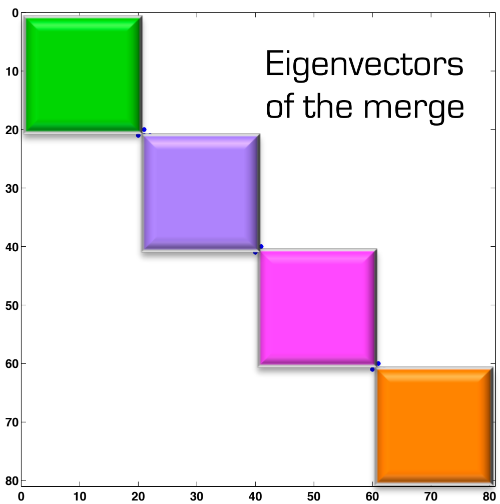


The Divide and Conquer Algorithm *dstedc*

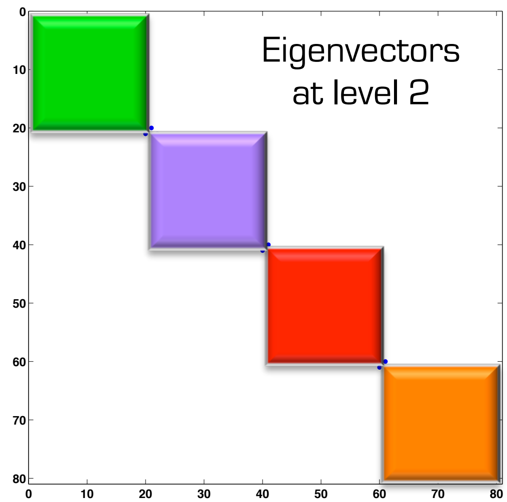




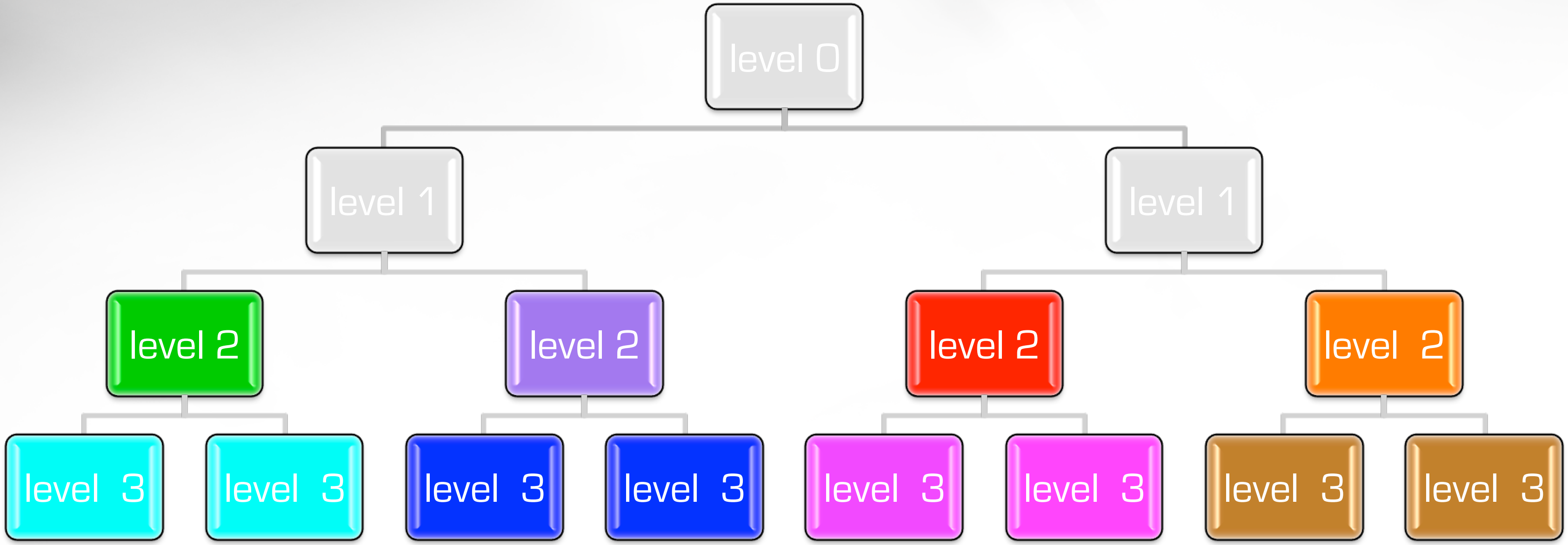
X



=



The Divide and Conquer Algorithm *dstedc*



Cost:

In the worst case, when no eigenvalue is deflated, the overall complexity can be expressed by:

$$n^3 + 2(\frac{n}{2})^3 + 4(\frac{n}{4})^3 + \cdots = \sum_{i=0}^{log(n)} \frac{n^3}{2^{2i}} = \frac{4n^3}{3} + \Theta(n^2) \quad (9)$$

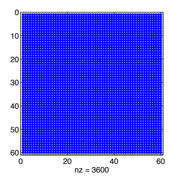
We can observe that the overall complexity is dominated by the cost of the last merge which is about n^3 operations.

General Overview: the Eigenproblem algorithms

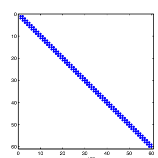
Background:

- Symmetric EVP $Ax = \lambda x$ meaning compute $A = Z \lambda Z^*$
  where λ are the Eigenvalues and Z are the eigenvectors.

1. Tri-Diagonalization Reduction: `transform A to nice form` ☺



==>



$$A = Q\,T\,Q^*$$

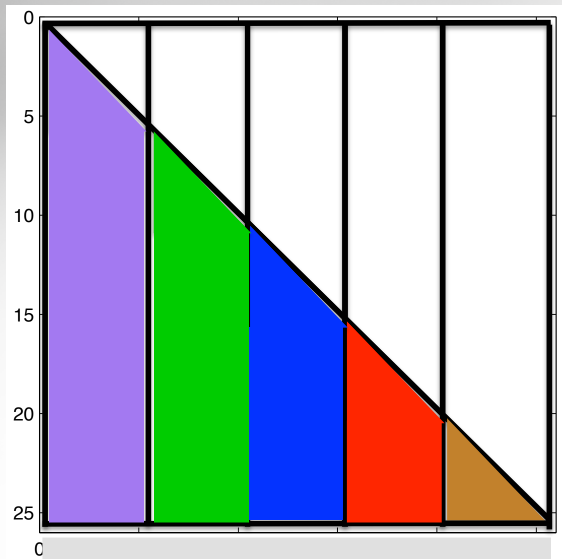
2. Solve: `compute the Eigenvalue and Eigenvectors of the tridiagonal`

$$T = E\,\lambda\,E^* \quad \Rightarrow \quad A = Q\,.\,(E\,\lambda\,E^*)\,.\,Q^*$$

3. Back transformation: `update the computed Eigenvectors.`

$$Z = Q * E$$

The Back Transformation *dormtr*



Householder reflectors

X

E eigenvectors of T

Cost:

The cost of every step is summarized as $2(n - step)i_b^2 + 4n(n - step)i_b$, where i_b is the blocking factor used by the dormtr in order to perform Level 3 BLAS operation which is usually 32 or 64. Therefore, the total floating-point cost of the $(n/i_b - 1)$ steps described in Algorithm 2 is:

$$
\begin{aligned}
flops &\approx \sum_{s=i_b}^{n} (n-s)i_b^2 + 4n(n-s)i_b \\
&\approx 2n^3 (\text{Level 3})
\end{aligned} \tag{10}
$$

The total cost of the symmetric eigenvalue solver

Total cost:

$$\underbrace{\frac{2}{3}n^3(\mathsf{dsymv}) + \frac{2}{3}n^3(\mathsf{dsyr2k})}_{\text{tridiagonalization}} \quad + \quad \underbrace{\Theta(n^\omega)}_{\text{diagonalization}} \quad + \quad \underbrace{2 \times n^3(\mathsf{dormtr})}_{\text{back-transformation}} \tag{15}$$

Thus:

$$t = \underbrace{\frac{2n^3}{3P_{symv}} + \frac{2n^3}{3P_{L3}}}_{\text{tridiagonalization}} \quad + \quad \underbrace{\frac{4n^3}{3P_w}}_{\text{diagonalization}} \quad + \quad \underbrace{\frac{2n^3}{P_{L3}}}_{\text{back-transformation}} \tag{16}$$

In practice the performance level of the divide and conquer algorithm is considered to be $P_w \longrightarrow \frac{2}{5}P_{L3}$

$$\begin{aligned} t &= n^3\left(\frac{2}{3P_{symv}} + \frac{2}{3P_{L3}} + \frac{10}{3P_{L3}} + \frac{6}{3P_{L3}}\right) \\ t &= n^3\left(\frac{2}{3P_{symv}} + \frac{18}{3P_{L3}}\right) \end{aligned} \tag{17}$$

The total cost of the symmetric eigenvalue solver

Total cost:

If we consider that the performance of a Level 3 BLAS is about 20 times higher than the one for dsymv and we integrate this in Eq. (17) we obtain:

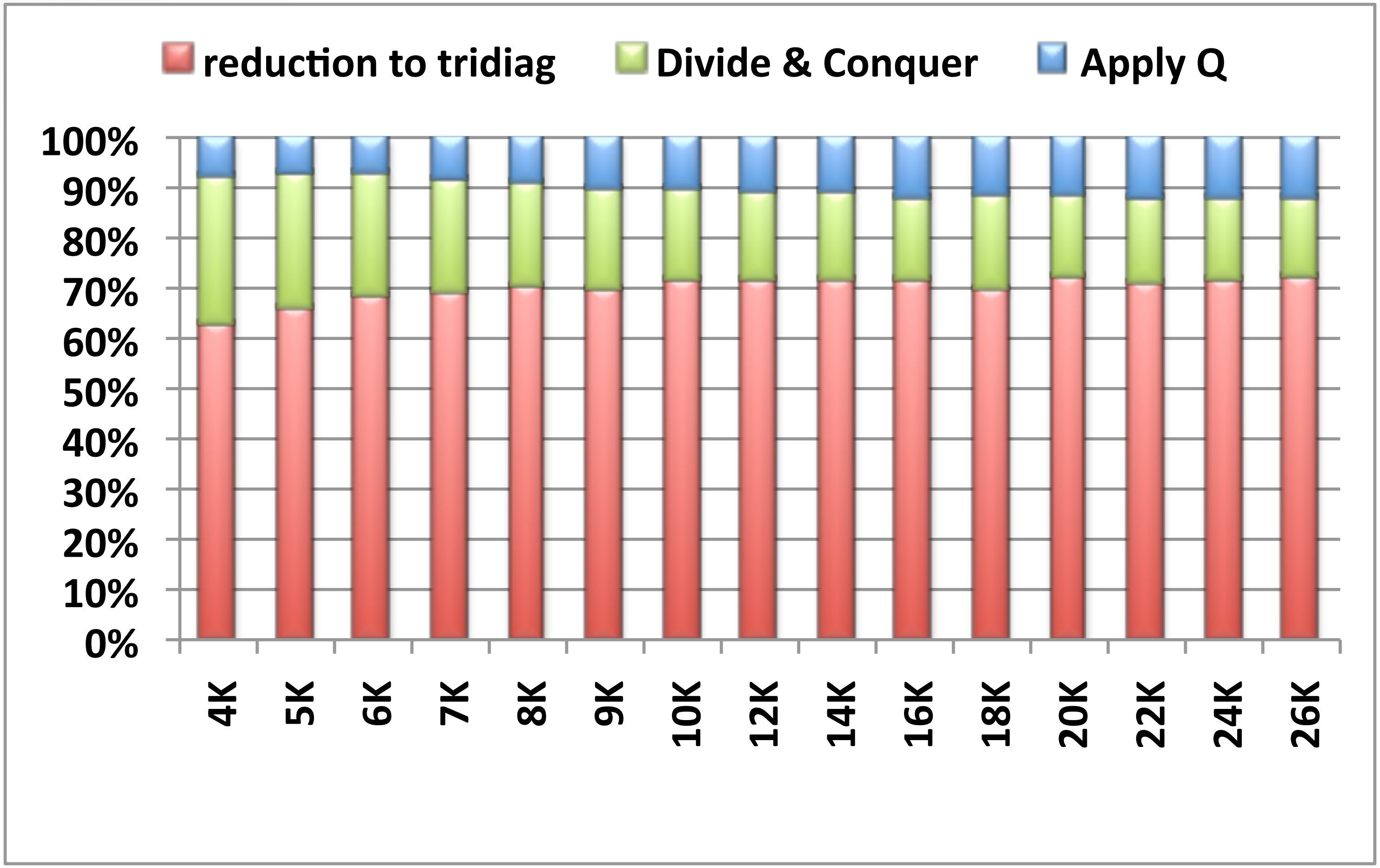
$$\begin{aligned} t = &\quad n^3 \left(\frac{2P_{L3}}{3P_{symv}P_{L3}} + \frac{18P_{symv}}{3P_{symv}P_{L3}} \right) \\ t = &\quad n^3 \frac{58}{60P_{symv}} \qquad (18) \\ \longrightarrow &\quad P_{symv} = n^3 \frac{58}{60t} \quad \text{and} \quad P_{L3} = n^3 \frac{58}{3t} \end{aligned}$$

Let us substitute P_{symv} and P_{L3} in Eq. (15) in order to find the impact of each phase.

$$\begin{aligned} t = &\quad \underbrace{\frac{42}{58}t}_{\text{tridiagonalization}} + \underbrace{\frac{10}{58}t}_{\text{diagonalization}} + \underbrace{\frac{6}{58}t}_{\text{back-transformation}} \\ t = &\quad \underbrace{0.72t}_{\text{tridiagonalization}} + \underbrace{0.18t}_{\text{diagonalization}} + \underbrace{0.10t}_{\text{back-transformation}} \qquad (19) \end{aligned}$$

The total cost of the symmetric eigenvalue solver

Total cost:

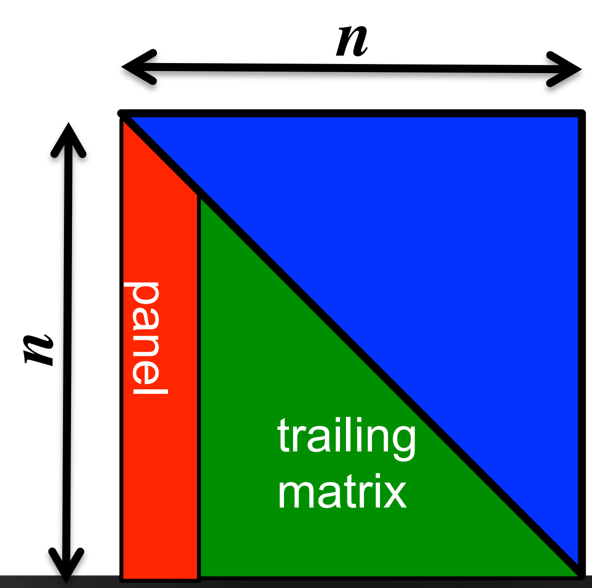


The standard Tridiagonal reduction *dsytrd*

Cost:

For all steps (n/n_b), the trailing matrix size varies from n to n_b by steps of size n_b, where l varies from n to n_b and k varies from $(n - n_b)$ to $2\ n_b$. Thus, the total cost for the n/n_b steps is:

$$\begin{aligned} flops &\approx 2n_b \sum_{n_b}^{n/n_b} l^2 + 2n_b \sum_{2n_b}^{\frac{n-n_b}{n_b}} k^2 \\ &\approx \tfrac{2}{3} n^3_{\text{symv}} + \tfrac{2}{3} n^3_{\text{syr2k}} \\ &\approx \tfrac{4}{3} n^3. \end{aligned} \tag{1}$$



The standard Tridiagonal reduction *dsytrd*

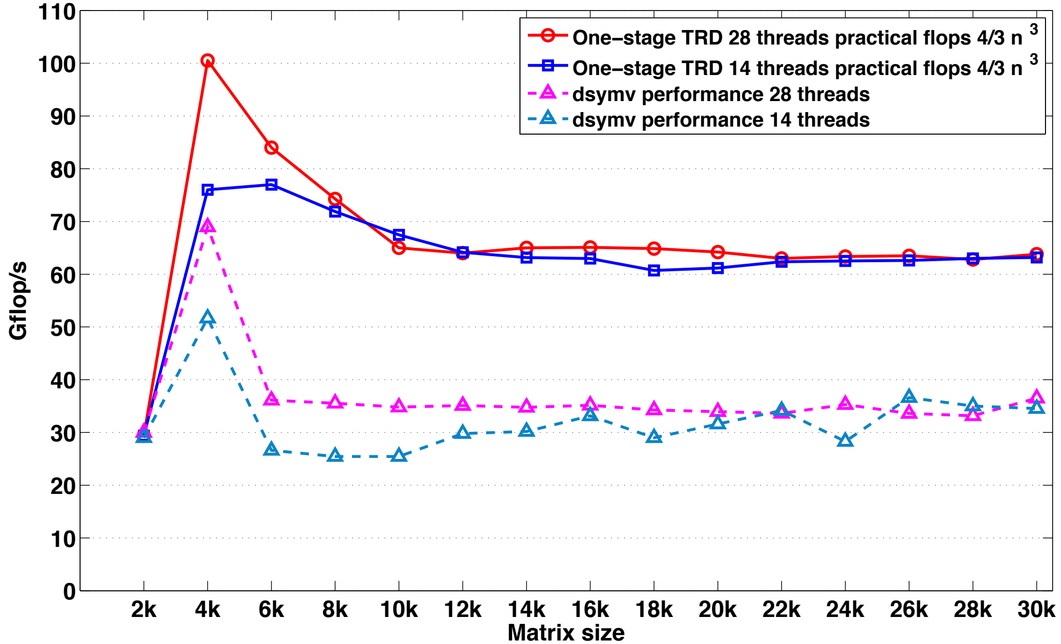
Maximal Performance bound of the TRD:

$$
\begin{aligned}
P_{max} &= \frac{number\ \ of\ \ operations}{minimum\ \ time\ \ t_{min}} \\
&= \frac{\frac{4}{3}n^3}{t_{min}(\frac{2}{3}n^3\ \ flops\ \ in\ \ symv)+t_{min}(\frac{2}{3}n^3\ \ flops\ \ in\ \ syr2k)} \\
&= \frac{\frac{4}{3}n^3}{\frac{2}{3}n^3 * \frac{1}{P_{symv}} + \frac{2}{3}n^3 * \frac{1}{P_{Level3}}} \\
&= \frac{2 * P_{Level3} * P_{symv}}{P_{Level3} + P_{symv}} \\
&\leq 2P_{symv} \quad when \quad P_{Level3} \gg P_{symv}.
\end{aligned}
\qquad (2)
$$

The standard Tridiagonal reduction *dsytrd*

Maximal Performance bound:

$$
\begin{aligned}
P_{max} &= \frac{number\ of\ operations}{minimum\ time\ t_{min}} \\
&= \frac{\frac{4}{3}n^3}{t_{min}(\frac{2}{3}n^3\ \ flops\ \ in\ \ symv)+t_{min}(\frac{2}{3}n^3\ \ flops\ \ in\ \ syr2k)} \\
&= \frac{\frac{4}{3}n^3}{\frac{2}{3}n^3 * \frac{1}{P_{symv}} + \frac{2}{3}n^3 * \frac{1}{P_{Level3}}} \\
&= \frac{2 * P_{Level3} * P_{symv}}{P_{Level3} + P_{symv}} \\
&\leq 2P_{symv} \quad when \quad P_{Level3} \gg P_{symv}.
\end{aligned}
$$



The total cost of the symmetric eigenvalue solver

✷ Characteristics:

- This algorithm does not achieve good performance, mainly due to the reduction phase
- The reduction to Tridiagonal phase relies on panel factorization,
- Too many Blas-2 op,
- ➔**Bulk sync phases,**
- ➔**Memory bound algorithm.**

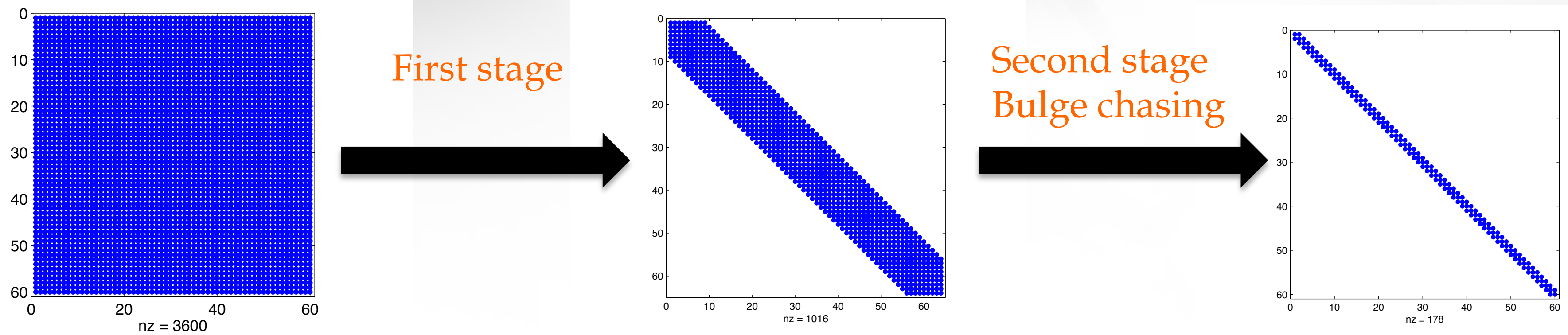
at ICL we are investigating different path to find a solution

The 2-stage tridiagonal algorithm

Ideas:

- The idea is to cast expensive memory operations, occurring during the panel factorization into fast compute intensive ones.
- Redesign the algorithm in a new fashion which increase the cache reuse.
- Design new cache friendly kernels to overcomes the memory bound limitation.
- Extract parallelism and schedule task in an asynchronous order.

The 2-stage tridiagonal algorithm

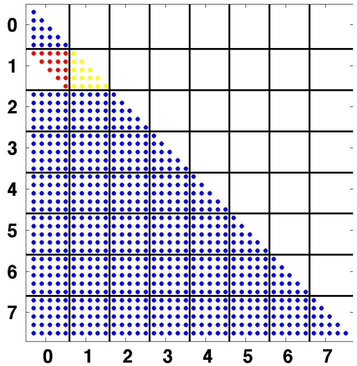


✸ Characteristics

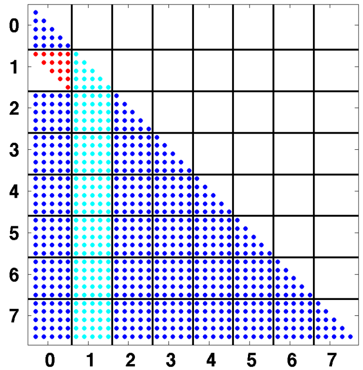
- **Stage 1:**
  - BLAS-3,
  - one shot reduction,
  - asynchronous execution,
- **Stage2:**
  - BLAS-2.5,
  - element-wise/column-wise,
  - asynchronous execution,
  - new cache friendly kernel.

The 2-stage tridiagonal algorithm

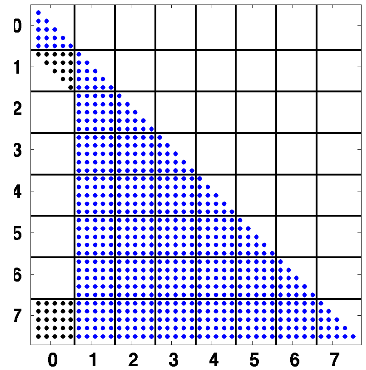
Stage 1



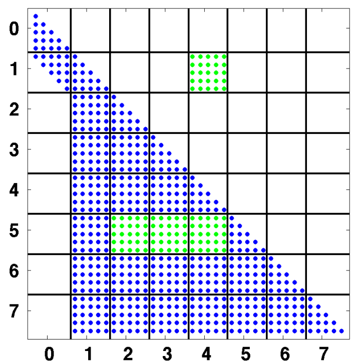
Apply Q from LEFT on tile A(1,1)



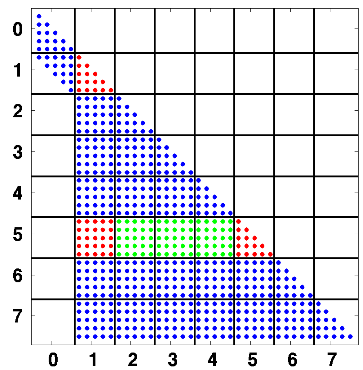
Apply Q from RIGHT on tile A(7,1)



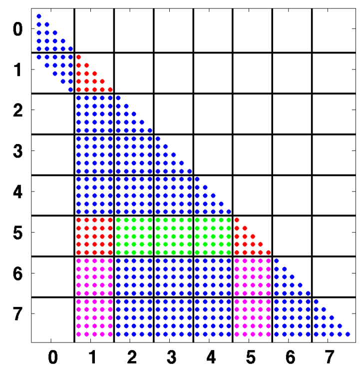
QR factorisation on tile A(1,0) A(7,0)



Apply Q from LEFT on tile $A(4,1)^T$ A(5,4)



Apply Q from LEFT and RIGHT on tiles A(1,1) A(5,1) A(5,5)



Apply Q from RIGHT on tile A(7,1) A(7,5)

```
1:for step = 1; 2 to NT-1
2:   QR factorize
3:   apply Q from LEFT
4:   for i = step+1 to NT
5:      apply Q from RIGHT
6:   end for
7:   for k = step+2 to NT do
8:      factorize 2 tiles
9:      for j = step+2 to k-1
10:          LEFT update on 2 tiles
11:      end for
12:      apply a LEFT and
         RIGHT update diagonal
13:      for m = k+1 to NT do
14:          RIGHT updates
15:      end for
16:  end for
17:end for
```

For each repetition of the outer loop in Algorithm 1, one dgeqrt, one dsyrfb, nt-step-1 dormqr, nt-step-1 dtsqrt, (nt-step-1)×(nt-step-2) dtsmqr and one dtsmqr_diag.

	dgeqrt	dsyrfb	dormqr	dtsqrt	dtsmqr	dtsmqr_diag
cost	$2n_b^3$	$3n_b^3$	$3n_b^3$	$\frac{10}{3}n_b^3$	$5n_b^3$	$10n_b^3$

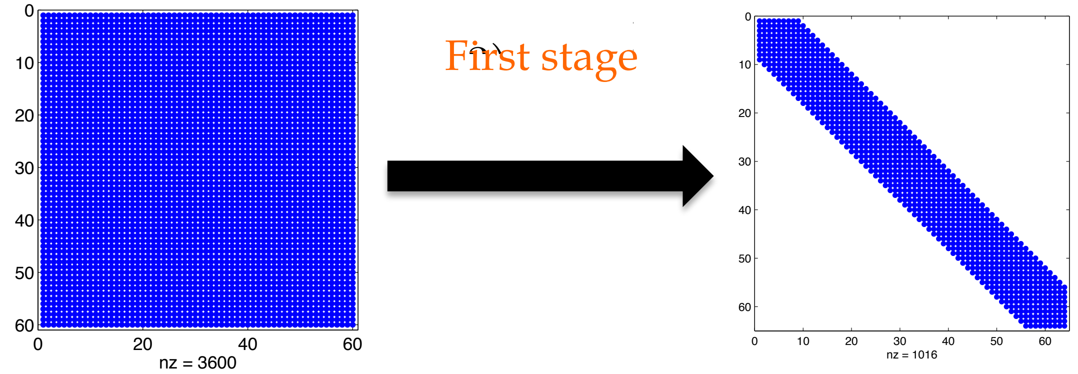
Table 1: The flops count of every routine involved in the reduction to band tri-diagonal (first stage) form.

The 2-stage tridiagonal algorithm

Stage 1

Integrating this quantity over all the $(n/n_b - 1)$ steps of the outer loop in Algorithm 1, the total operation count is:

$$
\begin{aligned}
flops &\approx \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + 3n_b^3 + (nt-s-1)3n_b^3 + (nt-s-1)\tfrac{10}{3}n_b^3 \\
&\quad +(nt-s-1)\times(nt-s-2)5n_b^3 + 10n_b^3 \\
&\approx \tfrac{5}{3}n^3 + \tfrac{5n_b}{3}n^2 + \tfrac{n_b}{3}n^3 \\
&\approx \tfrac{5}{3}n^3 (\text{Level 3})
\end{aligned}
$$



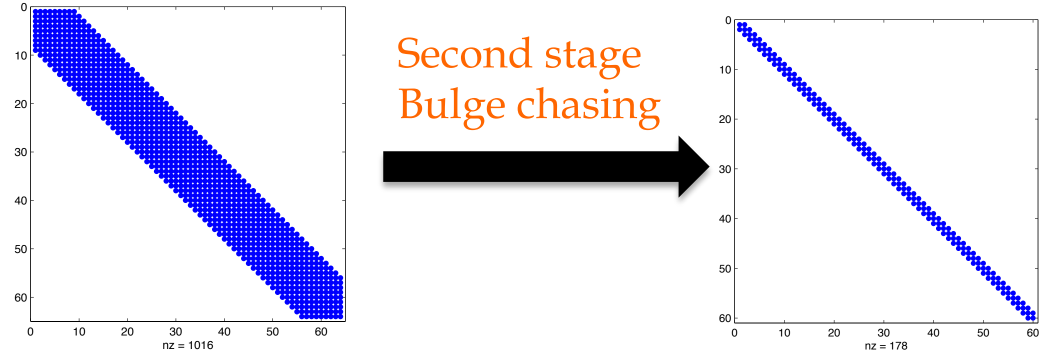
The 2-stage tridiagonal algorithm

Stage 2

Its cost can be expressed as

$$6 n_b n^2$$

operations. The second stage only performs a small portion of flops and, in addition, it is done by Level 2.5 BLAS which are our custom cache-friendly and memory aware computational kernels for the bulge chasing procedure.



The 2-stage tridiagonal algorithm

Total cost

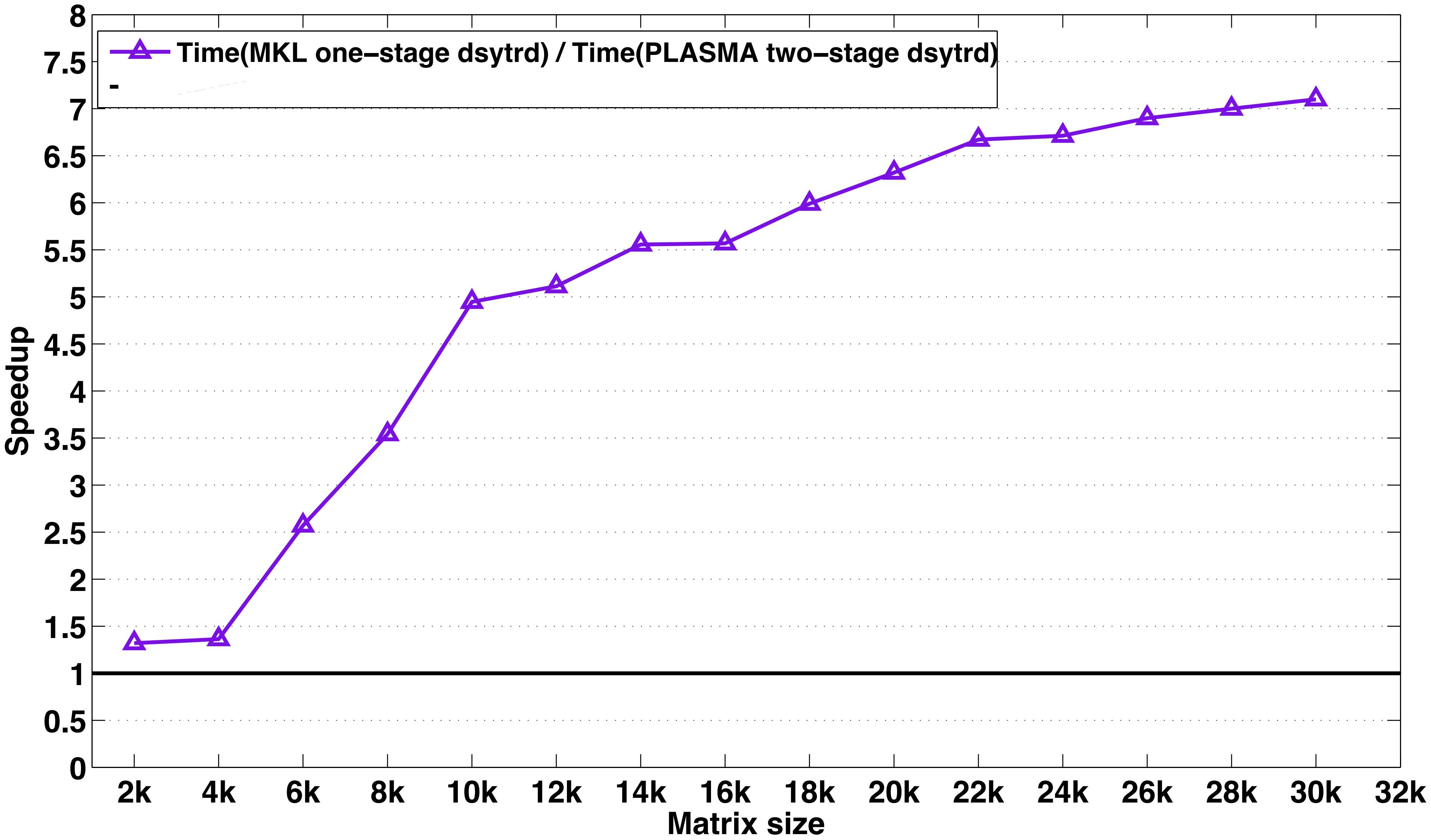
Hence, the total cost of the reduction to full tridiagonal form is:

$$\underbrace{\frac{5}{3}n^3(\text{Level 3})}_{\text{first stage}} + \underbrace{6 \times D \times n^2(\text{BLAS 2.5})}_{\text{second stage}} = \frac{5}{3}n^3(\text{Level 3}) + \Theta(n^2) \tag{5}$$

Our performance model will let us predict what speedup we can expect from the two-stage algorithm. the speedup that can be reached is formulated as:

$$\begin{aligned} \text{speedup} &= \frac{\text{time of two-stage}}{\text{time of one-stage}} \\ &= \frac{5n^3/3P_3}{2n^3/3P_{symv} + 2n^3/3P_3} \\ &= \frac{2(P_{symv}+P_3)}{5P_{symv}} \\ &\approx 8 \text{ if } P_3 \text{ is about 20x } P_{symv} \end{aligned} \tag{6}$$

The 2-stage tridiagonal algorithm



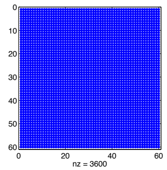
Won best paper

General Overview: the Eigenproblem algorithms

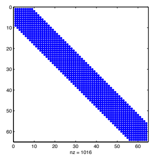
Background:

- Symmetric EVP $Ax = \lambda x$ meaning compute $A = Z \lambda Z^*$ where λ are the Eigenvalues and Z are the eigenvectors.

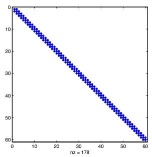
1. 2 stage Tri-Diagonalization Reduction: `transform A to nice form` ☺



==>



==>



$$A = Q_1 Q_2 \, T \, Q_2^* \, Q_1^*$$

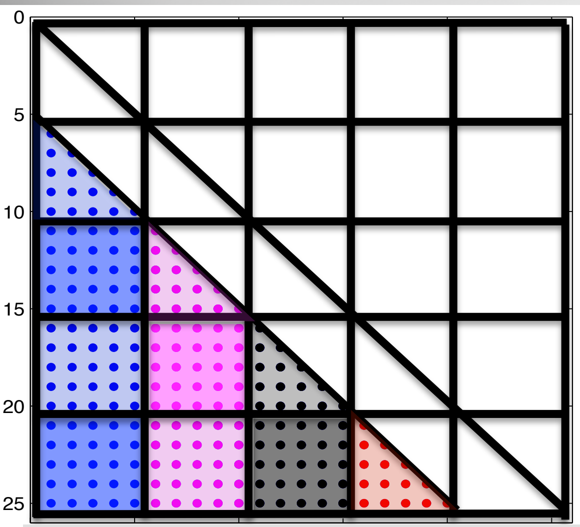
2. Solve: `compute the Eigenvalue and Eigenvectors of the tridiagonal`

$$T = E \lambda E^*$$

3. Back transformation: `update the computed Eigenvectors.`

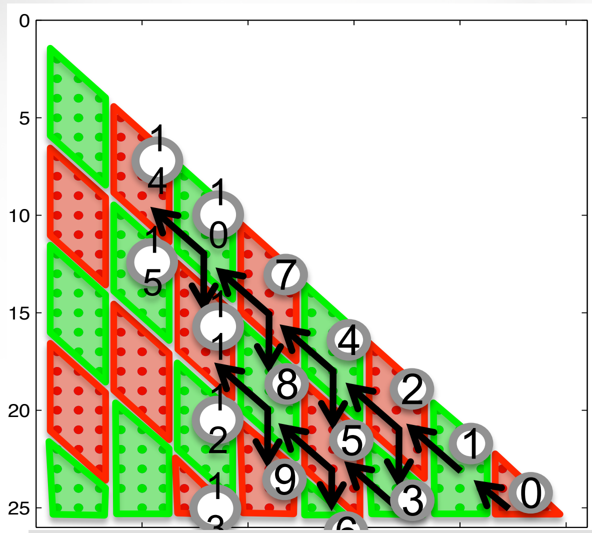
$$Z = Q_1 Q_2 * E$$

The Back Transformation `dormtr`



Householder reflectors Q_1

X



Householder reflectors Q_2

X

E eigenvectors of T

Cost:

$$Z = Q_1 Q_2 E = (I - V_1 T_1 V_1^H)(I - V_2 T_2 V_2^H)E,$$

$$flops \approx \sum_{s=i_b}^{n} 4 n i_b \left(1 + \frac{i_b}{n_b}\right) s \quad + \quad \sum_{s=i_b}^{n} 4 n i_b \left(1 + \frac{i_b}{n_b}\right) s$$

$$\approx 2\left(1 + \frac{i_b}{n_b}\right) n^3 (\text{Level 3}) \quad + \quad 2\left(1 + \frac{i_b}{n_b}\right) n^3 (\text{Level 3})$$

$$\approx \frac{5}{2} n^3 (\text{Level 3}) \quad + \quad \frac{5}{2} n^3 (\text{Level 3})$$

The total cost of the symmetric eigenvalue solver

Total cost:

The total cost of the symmetric eigensolver based on the two-stage approach.

$$\underbrace{\frac{5}{3}n^3(\mathsf{dlarfb})}_{\text{first stage}} + \underbrace{6Dn^2(\mathsf{BLAS2.5})}_{\text{second stage}} + \underbrace{\Theta(n^\omega)}_{\text{diag.}} + \underbrace{\frac{5}{2}n^3(\mathsf{dlarfb})}_{\text{back-trans. 2nd stage}} + \underbrace{\frac{5}{2}n^3(\mathsf{dlarfb})}_{\text{back-trans. 1st stage}}$$

The total cost of the symmetric eigenvalue solver

Total cost:

The total cost of the symmetric eigensolver based on the two-stage approach.

$$\underbrace{\frac{5}{3}n^3(\mathsf{dlarfb})}_{\text{first stage}} + \underbrace{6Dn^2(\mathsf{BLAS2.5})}_{\text{second stage}} + \underbrace{\Theta(n^\omega)}_{\text{diag.}} + \underbrace{\frac{5}{2}n^3(\mathsf{dlarfb})}_{\text{back-trans. 2nd stage}} + \underbrace{\frac{5}{2}n^3(\mathsf{dlarfb})}_{\text{back-trans. 1st stage}}$$

- Let S_{trd} denote the speedup obtained by the tridiagonal phase which is about 8-fold, thus the time of the two-stage tridiagonal reduction expressed in function of t is $0.7t/S_{trd}$.

- Our optimized implementation can reach a factor of about $S_{edc} = \frac{5}{2}$ times faster than the corresponding MKL routine.

The total cost of the symmetric eigenvalue solver

Speedup:

- Let S_{trd} denote the speedup obtained by the tridiagonal phase which is about 8-fold, thus the time of the two-stage tridiagonal reduction expressed in function of t is $0.7t/S_{trd}$.

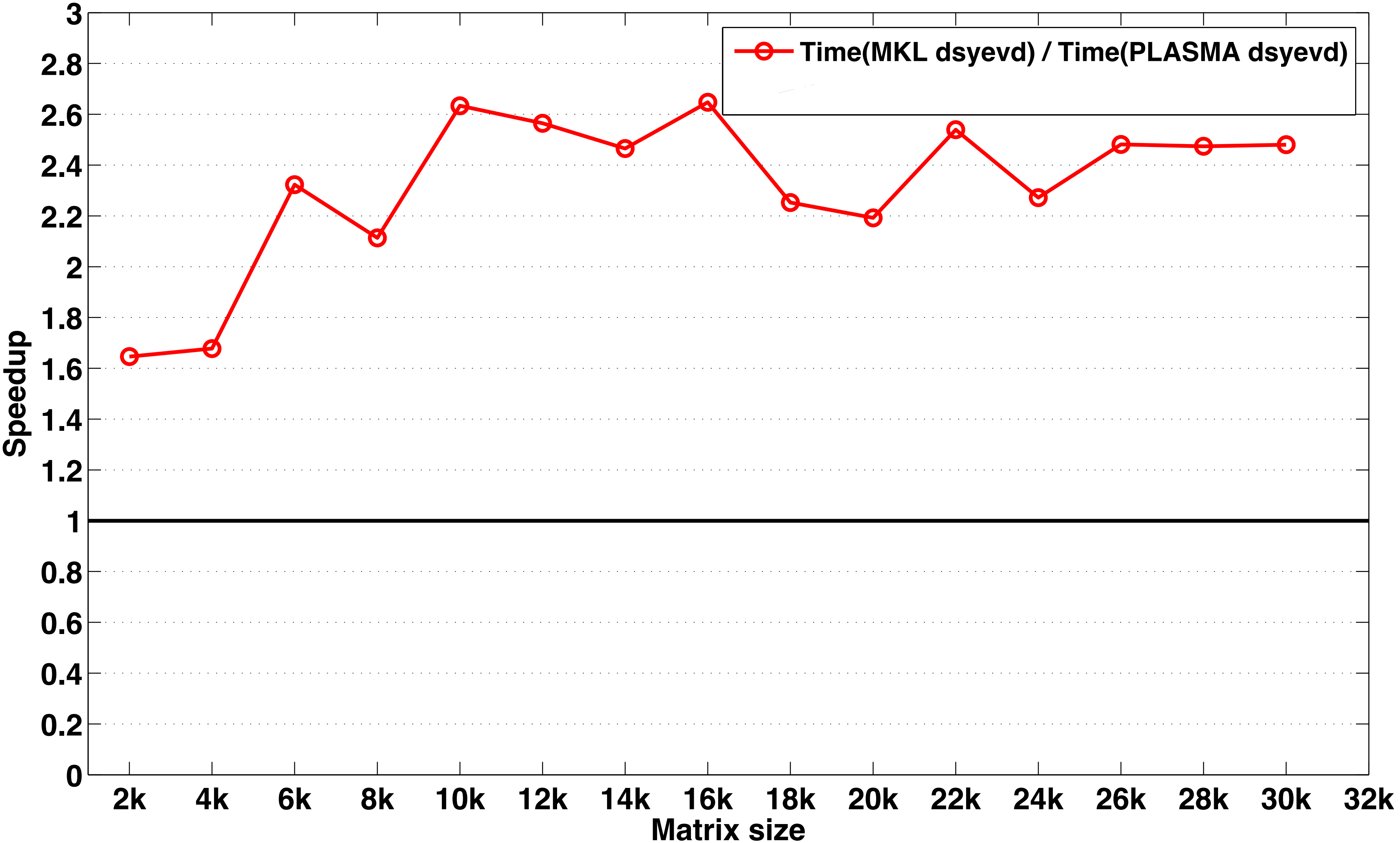
- Our optimized implementation can reach a factor of about $S_{edc} = \frac{5}{2}$ times faster than the corresponding MKL routine.

$$\text{Speedup} \approx \frac{0.72t + 0.18t + 0.10t}{\frac{0.72t}{S_{trd}} + \frac{0.18t}{S_{edc}} + 0.125t + 0.125t}$$

$$\text{Speedup} \approx \frac{0.72t + 0.18t + 0.10t}{0.09t + 0.07t + 0.125t + 0.125t}$$

$$\text{Speedup} \approx 2.5$$

The total cost of the symmetric eigenvalue solver



The total cost of the symmetric eigenvalue solver

Speedup 20%:

- Let S_{trd} denote the speedup obtained by the tridiagonal phase which is about 8-fold, thus the time of the two-stage tridiagonal reduction expressed in function of t is $0.7t/S_{trd}$.

- Our optimized implementation can reach a factor of about $S_{edc} = \frac{5}{2}$ times faster than the corresponding MKL routine.

$$\text{Speedup} \approx \frac{0.72t + 0.18t + 0.10t \times 0.2}{0.09t + 0.07t \times 0.5 + 0.125t \times 0.2 + 0.125t \times 0.2}$$

$$\text{Speedup} \approx 5.7$$

Questions ?

Thank you very much for your attention