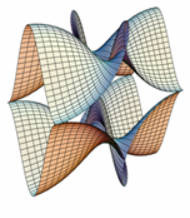
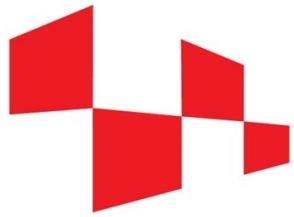


University of Zagreb
Faculty of Natural Sciences
Department of Mathematics



HrZZ
Project #9345
Croatian Science Foundation

A Householder-based algorithm for Hessenberg-triangular reduction

Zvonimir Bujanović, University of Zagreb
Lars Karlsson, Umeå University
Daniel Kressner, EPF Lausanne

NASCA 2018, Kalamata
July 4, 2018

Contents

1. Introduction
2. Opposite Householder reflectors and the basic algorithm
3. Improving performance: the blocked algorithm
4. Numerical experiments

Computing eigenpairs of a matrix pencil

The QZ-algorithm for computing eigenpairs of $A - \lambda B$

1. Use QR or RQ factorization to reduce B to upper triangular.

$$A = \left[\quad \right], B = \left[\quad \right].$$

2. Hessenberg-triangular reduction: find orthogonal Q, Z such that $A \leftarrow Q^T A Z$ is Hessenberg, $B \leftarrow Q^T B Z$ is still upper triangular.

$$A = \left[\quad \right], B = \left[\quad \right].$$

3. Iteratively reduce A to quasi-triangular form.

$$A = \left[\quad \right], B = \left[\quad \right].$$

Step 3 : significant progress in the last decade [Kågström/Kressner, 2006]

- aggressive early deflation;
- multi-shift techniques.

Our goal: accelerate Step 2.

Hessenberg-triangular reduction

Current state of the art

- [Moler/Stewart, 1973]
  Uses $\mathcal{O}(n^2)$ Givens rotations.
  LAPACK: `DGGHRD`.

- [Kågström/Kressner/Quintana-Ortí/Quintana-Ortí, 2008]
  Accumulates Givens rotations, applies them via matrix multiplication.
  LAPACK: `DGGHD3`.

- Difficult to use efficient level 3 BLAS operations.
- Difficult to parallelize.

We propose a new algorithm based on (unconventional) use of Householder reflectors.

Opposite Householder reflectors [Watkins, 2000]

Common Householder reflectors: apply from the left to annihilate a column.

$$(I - \alpha u u^T) \cdot \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, \quad (\alpha, u) = H\left(\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}\right)$$

Opposite Householder reflectors [Watkins, 2000]

Common Householder reflectors: apply from the left to annihilate a column.

$$(I - \alpha u u^T) \cdot \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, \quad (\alpha, u) = H\left(\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}\right)$$

"Opposite" Householder reflectors: apply from the right to annihilate a column.

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} \cdot (I - \beta v v^T) = \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}$$

To compute v:

1. Solve $Bx = e_1$.
2. Set $(\beta, v) = H(x)$.

Opposite Householder reflectors [Watkins, 2000]

Common Householder reflectors: apply from the left to annihilate a column.

$$(I - \alpha u u^T) \cdot \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, \quad (\alpha, u) = H\left(\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}\right)$$

"Opposite" Householder reflectors: apply from the right to annihilate a column.

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} \cdot (I - \beta v v^T) = \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}$$

To compute v:

1. Solve $Bx = e_1$.
2. Set $(\beta, v) = H(x)$.

Theorem

If the above procedure is carried out in floating point precision, so that $(B + \triangle)x = e_1$, and we explicitly set zeros in the first column of $\tilde{B} = fl(B \cdot (I - \beta v v^T))$, then

$$\tilde{B} = (B + \tilde{\triangle}) \cdot (I - \beta v v^T), \quad \|\tilde{\triangle}\|_F \leq \|\triangle\|_F + cnu\|B\|_F.$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step 1 : start with

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}, B = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & & & \bullet \end{bmatrix}.$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step ❶ : start with

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}, B = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & & & \bullet \end{bmatrix}.$$

Step ❷ : use a conventional reflector to introduce zeros into the 1st column of A. This destroys the upper triangular structure of B (but preserves the 1st column).

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, B = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}.$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step ❶ : start with

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}, B = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & & & \bullet \end{bmatrix}.$$

Step ❷ : use a conventional reflector to introduce zeros into the 1st column of A.
This destroys the upper triangular structure of B (but preserves the 1st column).

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \otimes & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, B = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}.$$

Step ❸ : use an opposite reflector to introduce zeros into the 2nd column of B.
This preserves the upper Hessenberg part of A.

$$A = \begin{bmatrix} \bullet & \otimes & \otimes & \otimes \\ \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{bmatrix}, B = \begin{bmatrix} \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{bmatrix}.$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step 3 : use an opposite reflector to introduce zeros into the 2nd column of B.
This preserves the upper Hessenberg part of A.

$$A = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right].$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step ❸ : use an opposite reflector to introduce zeros into the 2nd column of B.
This preserves the upper Hessenberg part of A.

$$A = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right].$$

Step ❹ : use a conventional reflector to introduce zeros into the 2nd column of A.
This preserves the upper triangular part of B.

$$A = \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right].$$

HT-reduction: The basic algorithm

We apply a series of conventional reflectors from the left, and opposite from the right.

Step ❸ : use an opposite reflector to introduce zeros into the 2nd column of B.
This preserves the upper Hessenberg part of A.

$$A = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \otimes & \otimes & \otimes \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right].$$

Step ❹ : use a conventional reflector to introduce zeros into the 2nd column of A.
This preserves the upper triangular part of B.

$$A = \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ & \otimes & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right].$$

Step ❺ : use an opposite reflector to introduce zeros into the 3nd column of B.
This preserves the upper Hessenberg part of A.

$$A = \left[\begin{array}{cccc} \bullet & \bullet & \otimes & \otimes \\ \bullet & \bullet & \otimes & \otimes \\ & \bullet & \otimes & \otimes \\ & & \otimes & \otimes \end{array}\right], B = \left[\begin{array}{cccc} \bullet & \bullet & \otimes & \otimes \\ & \bullet & \otimes & \otimes \\ & & \otimes & \otimes \\ & & & \otimes \end{array}\right].$$

HT-reduction: A blocked algorithm

The algorithm requires:

- $n-2$ applications of conventional reflectors from the left,
- $n-2$ applications of opposite reflectors from the right.

Each opposite reflector requires solving a linear system with a (full) submatrix of B.

The total time complexity is $\mathcal{O}(n^4)$.
The arithmetic intensity is low.

HT-reduction: A blocked algorithm

The algorithm requires:

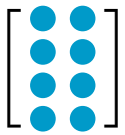
- $n-2$ applications of conventional reflectors from the left,
- $n-2$ applications of opposite reflectors from the right.

Each opposite reflector requires solving a linear system with a (full) submatrix of B.

The total time complexity is $\mathcal{O}(n^4)$.
The arithmetic intensity is low.

To improve efficiency:

1. Perform the reduction in panels $\rightsquigarrow$ increased arithmetic intensity.
   Group individual reflectors into block reflectors, using compact WY-representation.

   Left block reflector: $I - USU^T$, $U =$  , $S =$  .

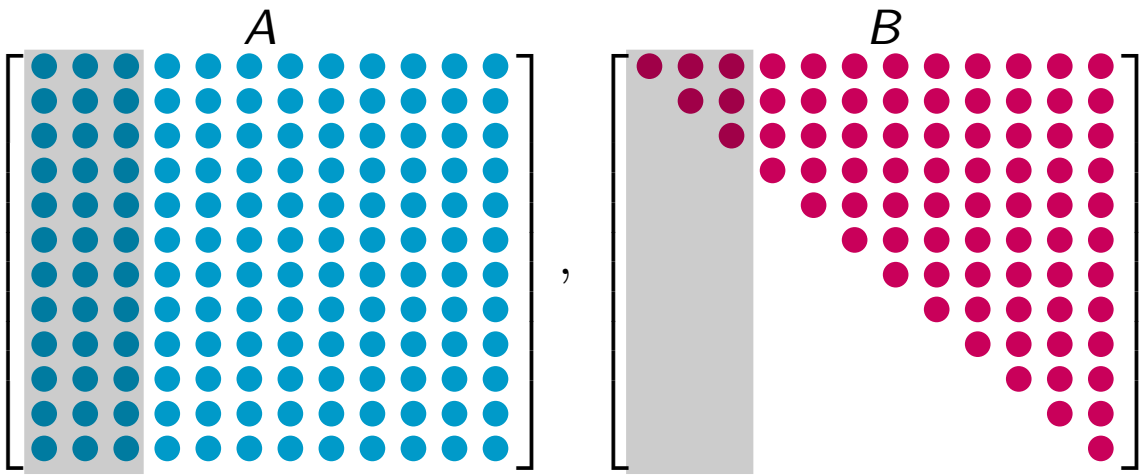
   Right block reflector: $I - VTV^T$, $V =$  , $T =$  .

2. Keep B in factored form $\rightsquigarrow$ reduce the time complexity to $\mathcal{O}(n^3)$.
   Avoid applying the reflectors to B immediately, and keep it factored as

$$\tilde{B} = (I - USU^T) \cdot B \cdot (I - VTV^T).$$

HT-reduction: Overview of the blocked algorithm

Split the matrices A, B into panels of width `nb`.



For each panel of `nb` columns:

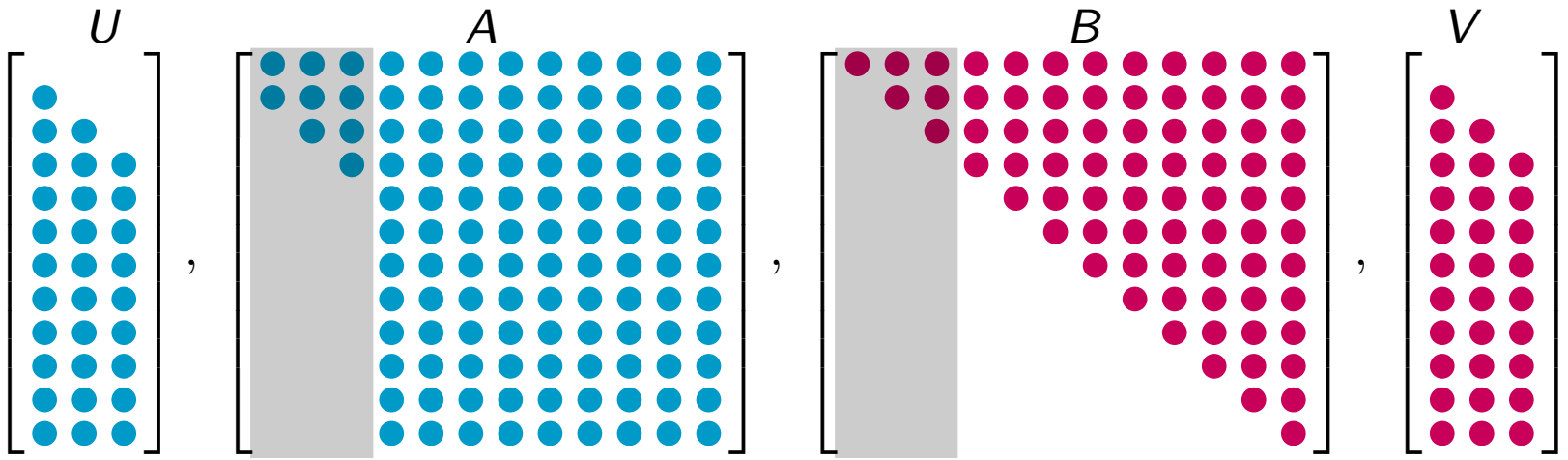
1. Initialize the left block reflector $I - USU^T$ to identity: $U = [\,]$, $S = [\,]$.
   Initialize the right block reflector $I - VTV^T$ to identity: $V = [\,]$, $T = [\,]$.
2. Reduce the panel column by column, computing $I - USU^T$, $I - VTV^T$.
   Block reflectors are applied only onto the panel of the matrix A.
3. Absorb the reflectors, i.e., update

$$A \leftarrow (I - USU^T) \cdot A \cdot (I - VTV^T),$$
$$B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T),$$

   while keeping B upper triangular.

HT-reduction: Overview of the blocked algorithm

Split the matrices A, B into panels of width `nb`.



For each panel of `nb` columns:

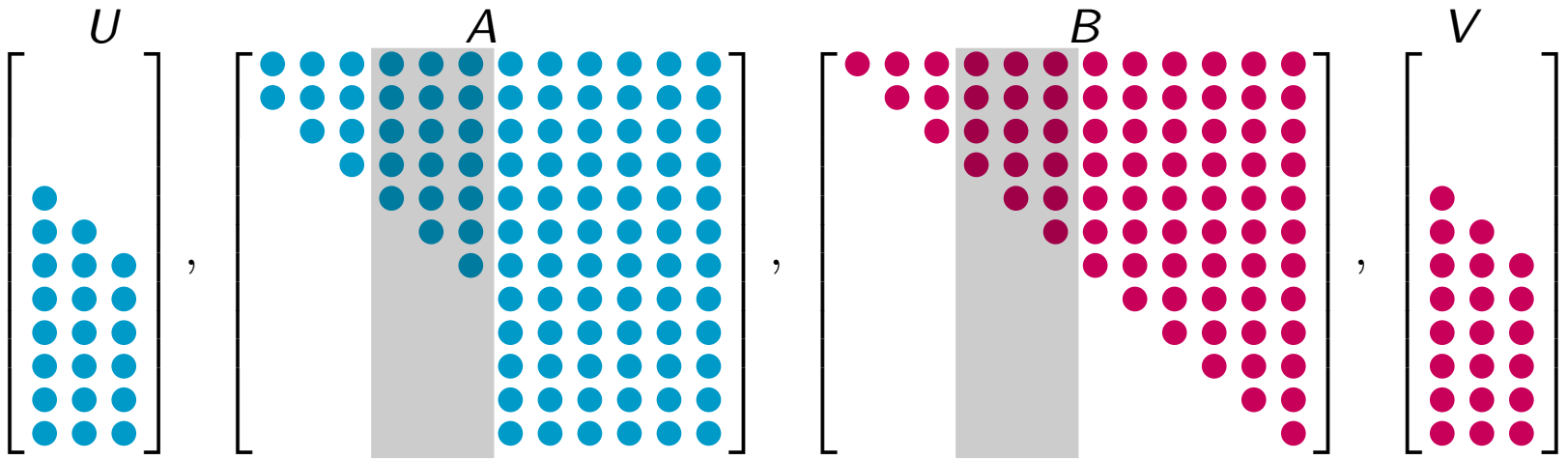
1. Initialize the left block reflector $I - USU^T$ to identity: $U = [\,]$, $S = [\,]$.
   Initialize the right block reflector $I - VTV^T$ to identity: $V = [\,]$, $T = [\,]$.
2. Reduce the panel column by column, computing $I - USU^T$, $I - VTV^T$.
   Block reflectors are applied only onto the panel of the matrix A.
3. Absorb the reflectors, i.e., update

$$A \leftarrow (I - USU^T) \cdot A \cdot (I - VTV^T),$$
$$B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T),$$

   while keeping B upper triangular.

HT-reduction: Overview of the blocked algorithm

Split the matrices A, B into panels of width `nb`.



For each panel of `nb` columns:

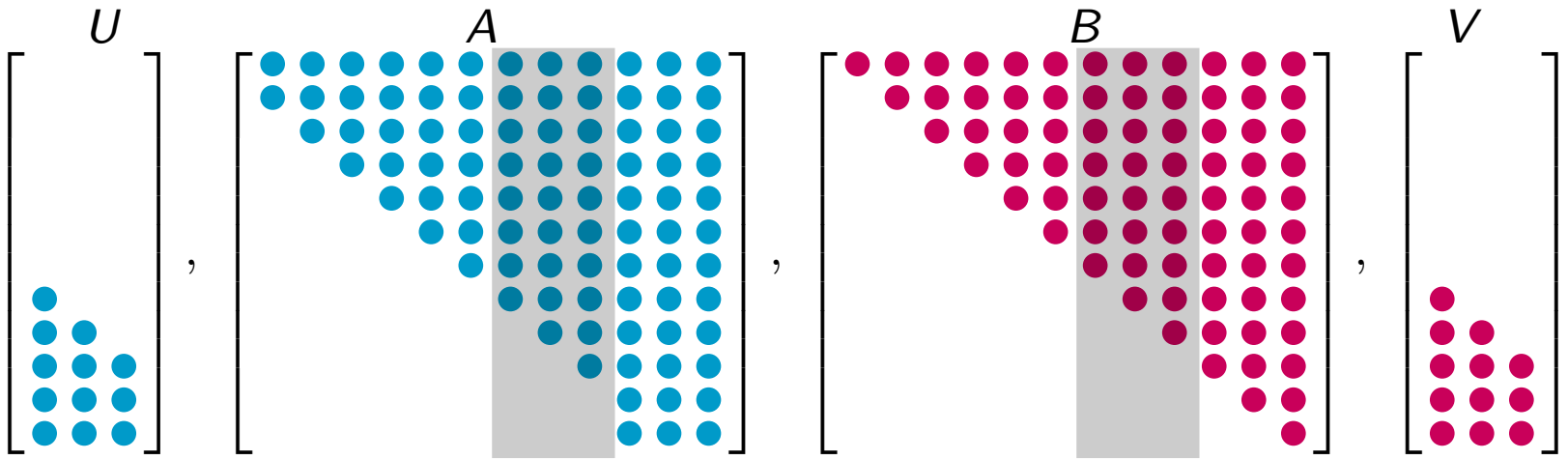
1. Initialize the left block reflector $I - USU^T$ to identity: $U = [\,]$, $S = [\,]$.
   Initialize the right block reflector $I - VTV^T$ to identity: $V = [\,]$, $T = [\,]$.
2. Reduce the panel column by column, computing $I - USU^T$, $I - VTV^T$.
   Block reflectors are applied only onto the panel of the matrix A.
3. Absorb the reflectors, i.e., update

$$A \leftarrow (I - USU^T) \cdot A \cdot (I - VTV^T),$$
$$B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T),$$

   while keeping B upper triangular.

HT-reduction: Overview of the blocked algorithm

Split the matrices A, B into panels of width `nb`.



For each panel of `nb` columns:

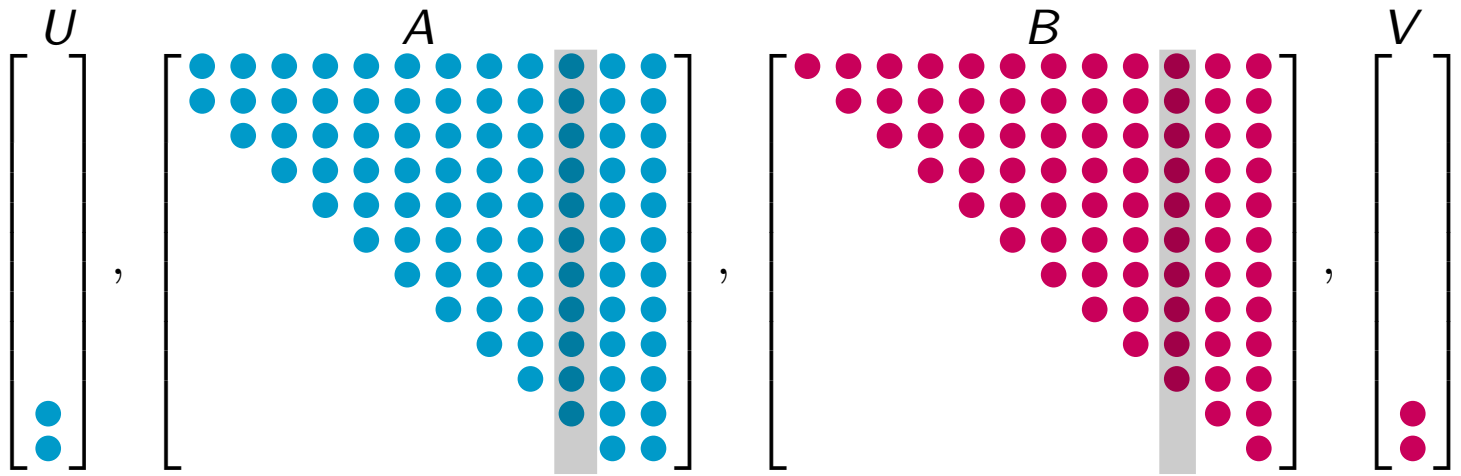
1. Initialize the left block reflector $I - USU^T$ to identity: $U = [\,]$, $S = [\,]$.
   Initialize the right block reflector $I - VTV^T$ to identity: $V = [\,]$, $T = [\,]$.
2. Reduce the panel column by column, computing $I - USU^T$, $I - VTV^T$.
   Block reflectors are applied only onto the panel of the matrix A.
3. Absorb the reflectors, i.e., update

$$A \leftarrow (I - USU^T) \cdot A \cdot (I - VTV^T),$$
$$B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T),$$

   while keeping B upper triangular.

HT-reduction: Overview of the blocked algorithm

Split the matrices A, B into panels of width `nb`.



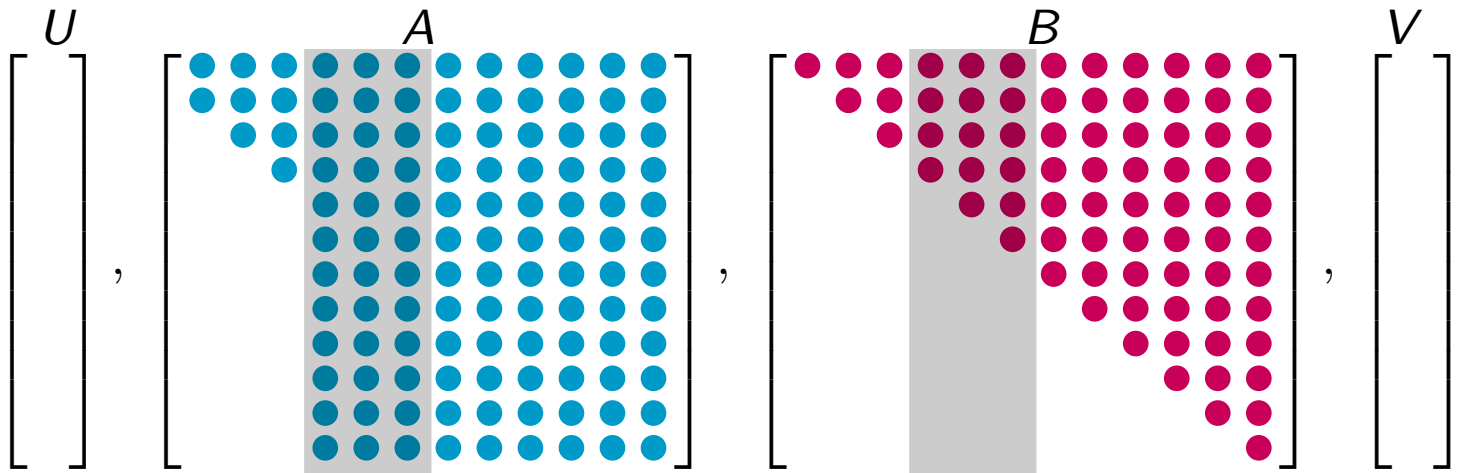
For each panel of `nb` columns:

1. Initialize the left block reflector $I - USU^T$ to identity: $U = [\,]$, $S = [\,]$.
   Initialize the right block reflector $I - VTV^T$ to identity: $V = [\,]$, $T = [\,]$.
2. Reduce the panel column by column, computing $I - USU^T$, $I - VTV^T$.
   Block reflectors are applied only onto the panel of the matrix A.
3. Absorb the reflectors, i.e., update

$$A \leftarrow (I - USU^T) \cdot A \cdot (I - VTV^T),$$
$$B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T),$$

   while keeping B upper triangular.

HT-reduction: Panel reduction



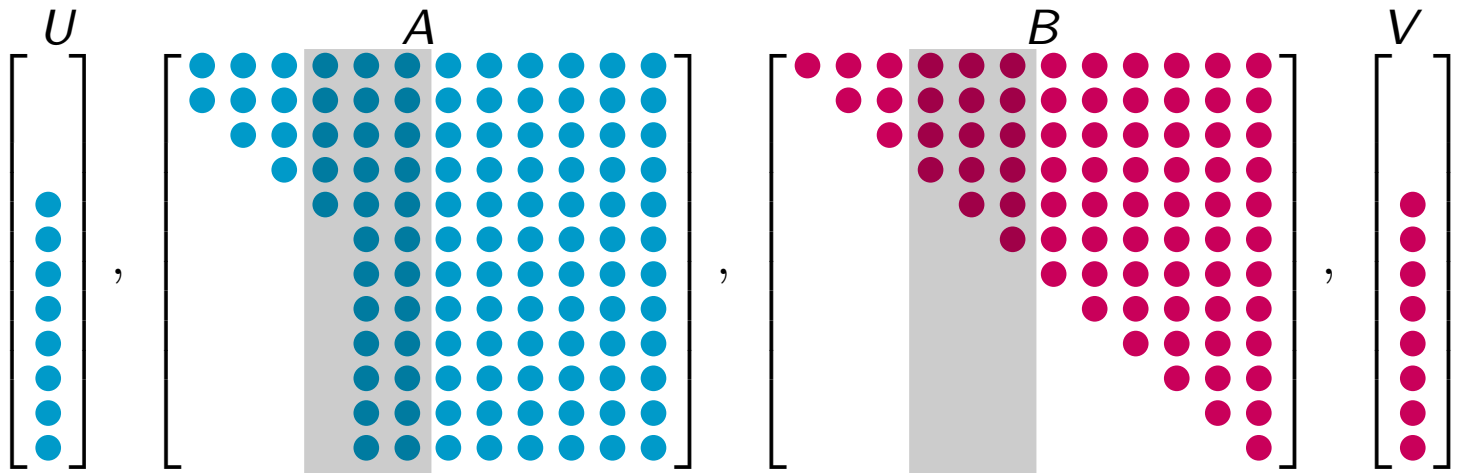
For each column A_j and B_{j+1} of the panel:

1. Apply $I - USU^T$ and $I - VTV^T$ to A_j.
2. Compute the conventional reflector (α, u) to annihilate A_j.
3. Expand U, S with (α, u).
4. Compute the opposite reflector (β, v) to annihilate

$$\tilde{B}_{j+1} = (I - USU^T)B(I - VTV^T)e_{j+1}.$$

5. Expand V, T with (β, v).

HT-reduction: Panel reduction



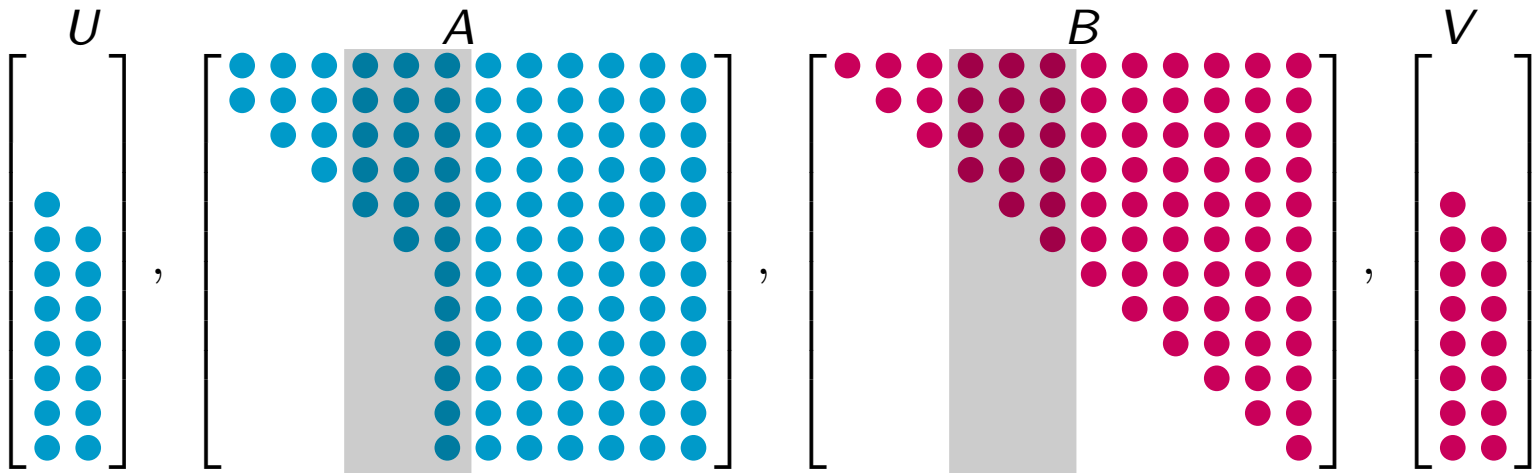
For each column A_j and B_{j+1} of the panel:

1. Apply $I - USU^T$ and $I - VTV^T$ to A_j.
2. Compute the conventional reflector (α, u) to annihilate A_j.
3. Expand U, S with (α, u).
4. Compute the opposite reflector (β, v) to annihilate

$$\tilde{B}_{j+1} = (I - USU^T)B(I - VTV^T)e_{j+1}.$$

5. Expand V, T with (β, v).

HT-reduction: Panel reduction



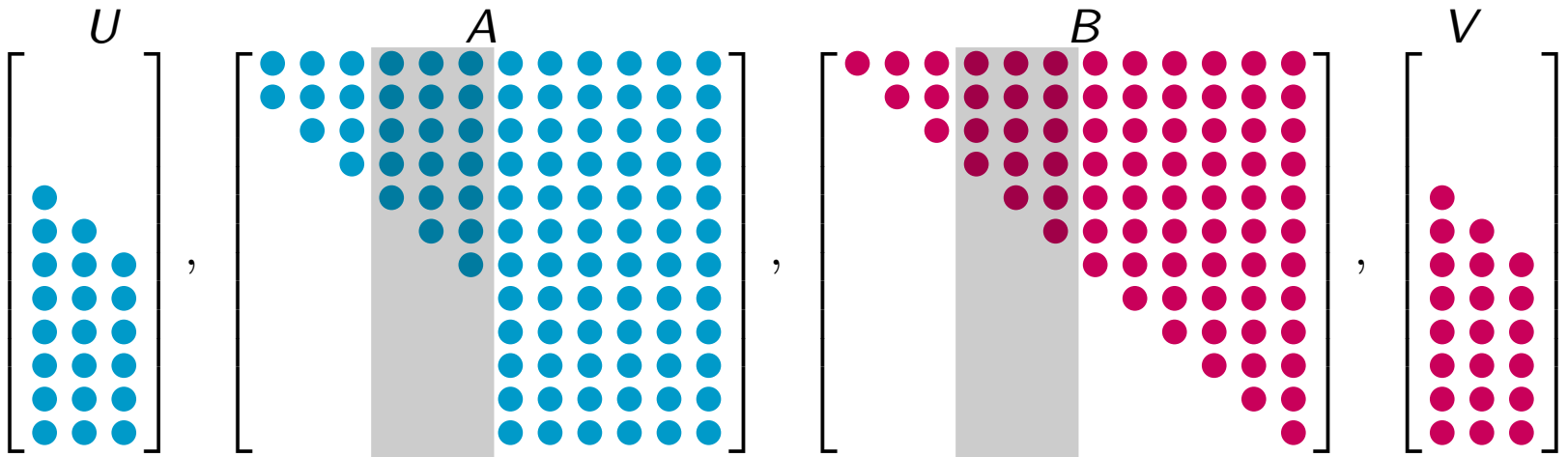
For each column A_j and B_{j+1} of the panel:

1. Apply $I - USU^T$ and $I - VTV^T$ to A_j.
2. Compute the conventional reflector (α, u) to annihilate A_j.
3. Expand U, S with (α, u).
4. Compute the opposite reflector (β, v) to annihilate

$$\tilde{B}_{j+1} = (I - USU^T)B(I - VTV^T)e_{j+1}.$$

5. Expand V, T with (β, v).

HT-reduction: Panel reduction



For each column A_j and B_{j+1} of the panel:

1. Apply $I - USU^T$ and $I - VTV^T$ to A_j.
2. Compute the conventional reflector (α, u) to annihilate A_j.
3. Expand U, S with (α, u).
4. Compute the opposite reflector (β, v) to annihilate

$$\tilde{B}_{j+1} = (I - USU^T)B(I - VTV^T)e_{j+1}.$$

5. Expand V, T with (β, v).

HT-reduction: Computing opposite reflectors

4. Compute the opposite reflector (β, v) to annihilate

$$\tilde{B}_{j+1} = (I - USU^T)B(I - VTV^T)e_{j+1}.$$

Here B is upper triangular, and

$$\tilde{B} = (I - USU^T)B(I - VTV^T) = \left[\begin{array}{cc} \blacktriangleleft & \blacksquare \\ & \boxtimes \end{array}\right].$$

We need to annihilate the first column of $\boxtimes \rightsquigarrow$ compute $x = \boxtimes^{-1} e_1$.

$$x = \left[\begin{array}{ccc} 0 & I_{n-j} \end{array}\right](I - VTV^T)^T B^{-1}(I - USU^T)\left[\begin{array}{c} 0 \\ e_1 \end{array}\right].$$

Note:

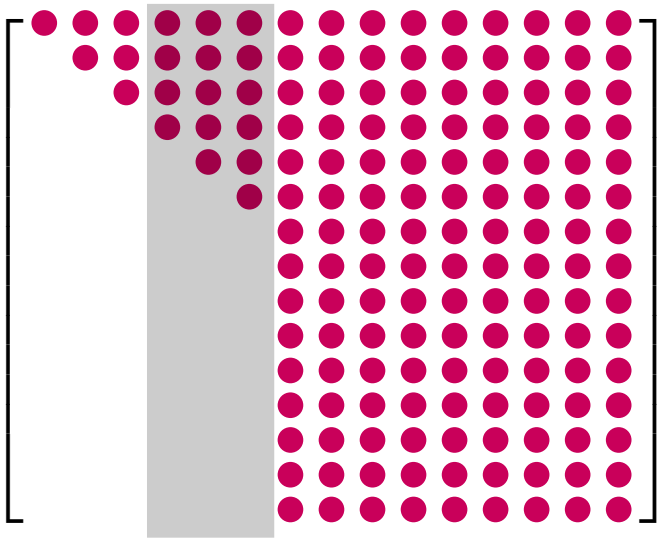
- The complexity of this step is now only $\mathcal{O}(n^2)$.
- In general, this procedure is backward stable only for the 1st column of B!
- To achieve stability, do iterative refinement until the residual $\|\boxtimes \cdot x - e_1\|$ becomes small enough.
- If more than 10 steps of IR are needed, absorb the reflectors and restart the panel.

HT-reduction: Absorbing block reflectors

Once the panel is full, the block reflectors are applied to A and B ("absorbed").
We constructed $I - USU^T$, $I - VTV^T$ so that B stays triangular within the panel.

Applying $B \leftarrow (I - USU^T) \cdot B \cdot (I - VTV^T)$ directly destroys upper triangular structure elsewhere in B.

$$B \leftarrow (I - USU^T)B(I - VTV^T) =$$

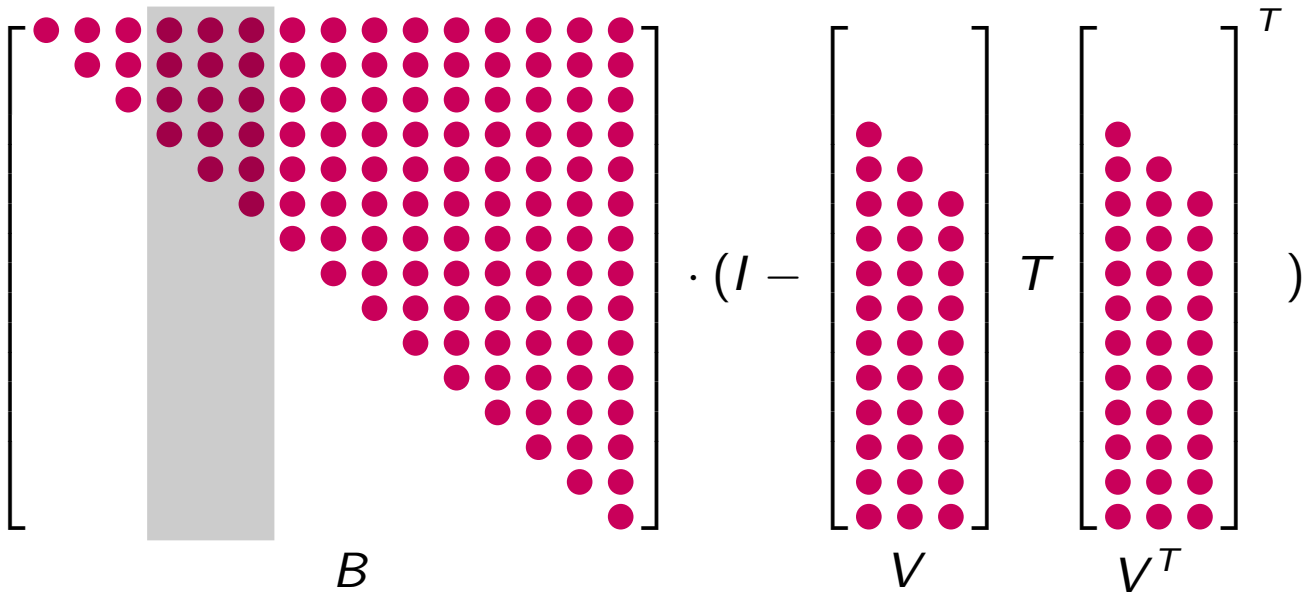


The entire bottom-right part of B would have to be made upper triangular (QR or RQ).

HT-reduction: Absorbing block reflectors

Thus we apply left/right block reflectors in stages.
We show the computation of $B \leftarrow B(I - VTV^T)$.

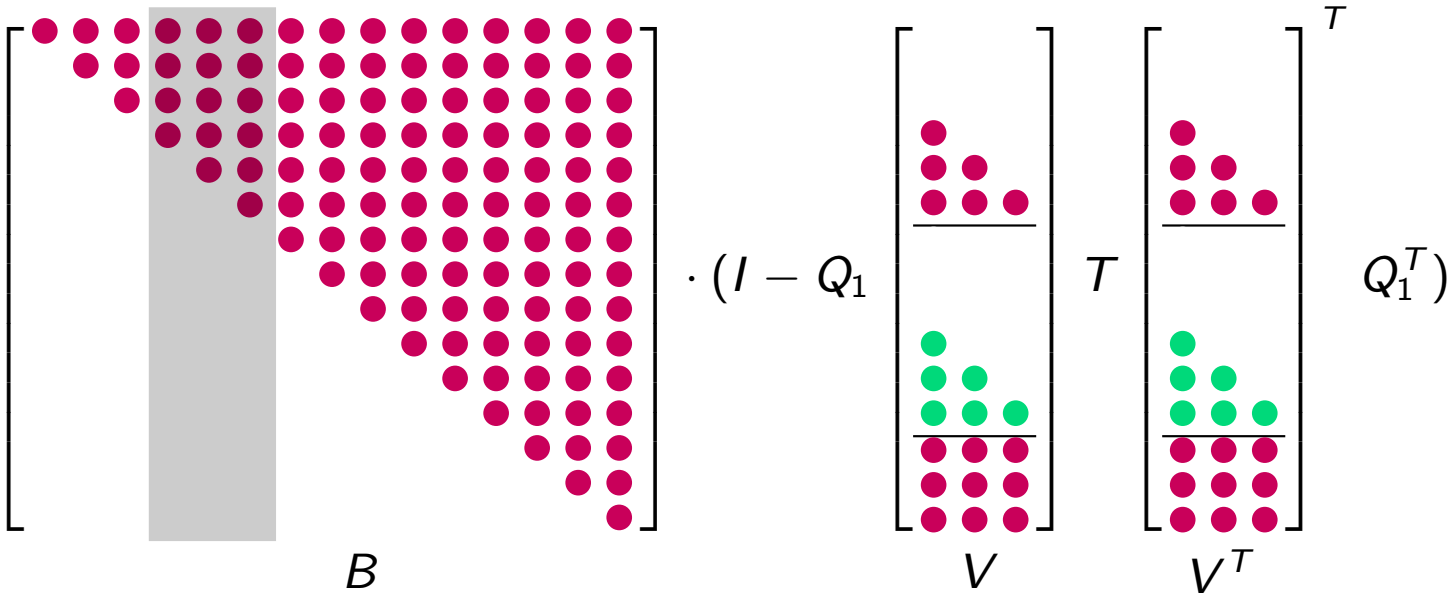
1. Start with:



HT-reduction: Absorbing block reflectors

Thus we apply left/right block reflectors in stages.
We show the computation of $B \leftarrow B(I - VTV^T)$.

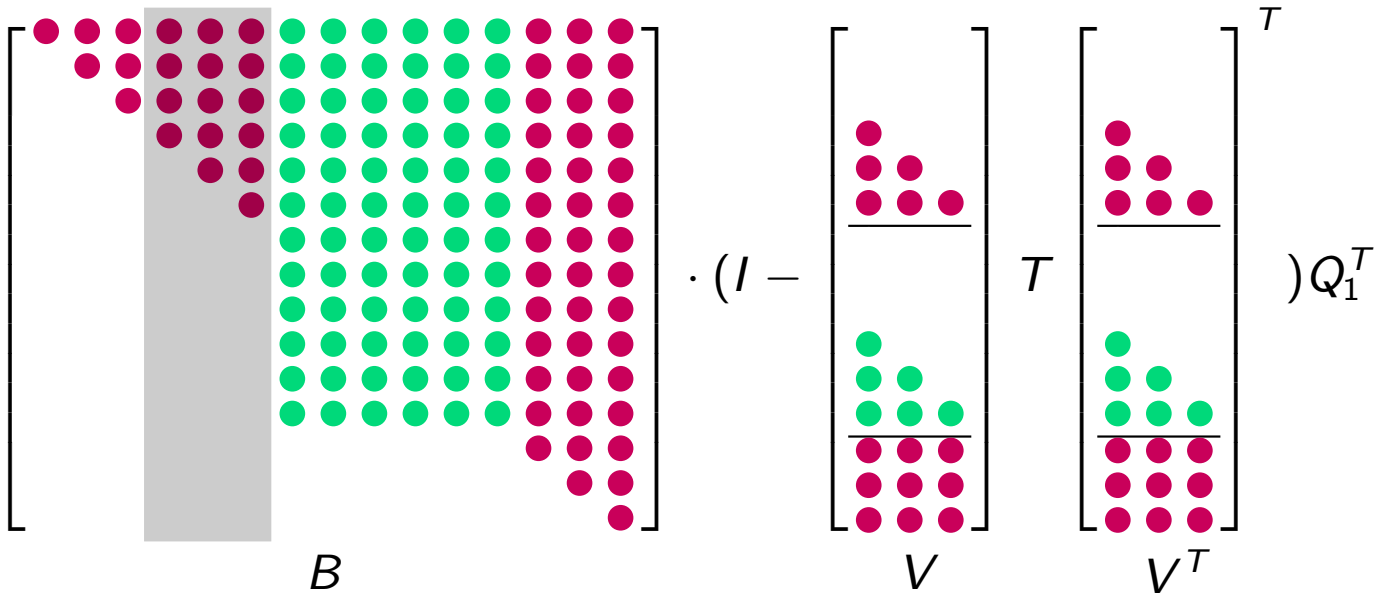
2. Compute the QL factorization of $2\mathtt{nb}$ rows in V (skip the top triangle!).



HT-reduction: Absorbing block reflectors

Thus we apply left/right block reflectors in stages.
We show the computation of $B \leftarrow B(I - VTV^T)$.

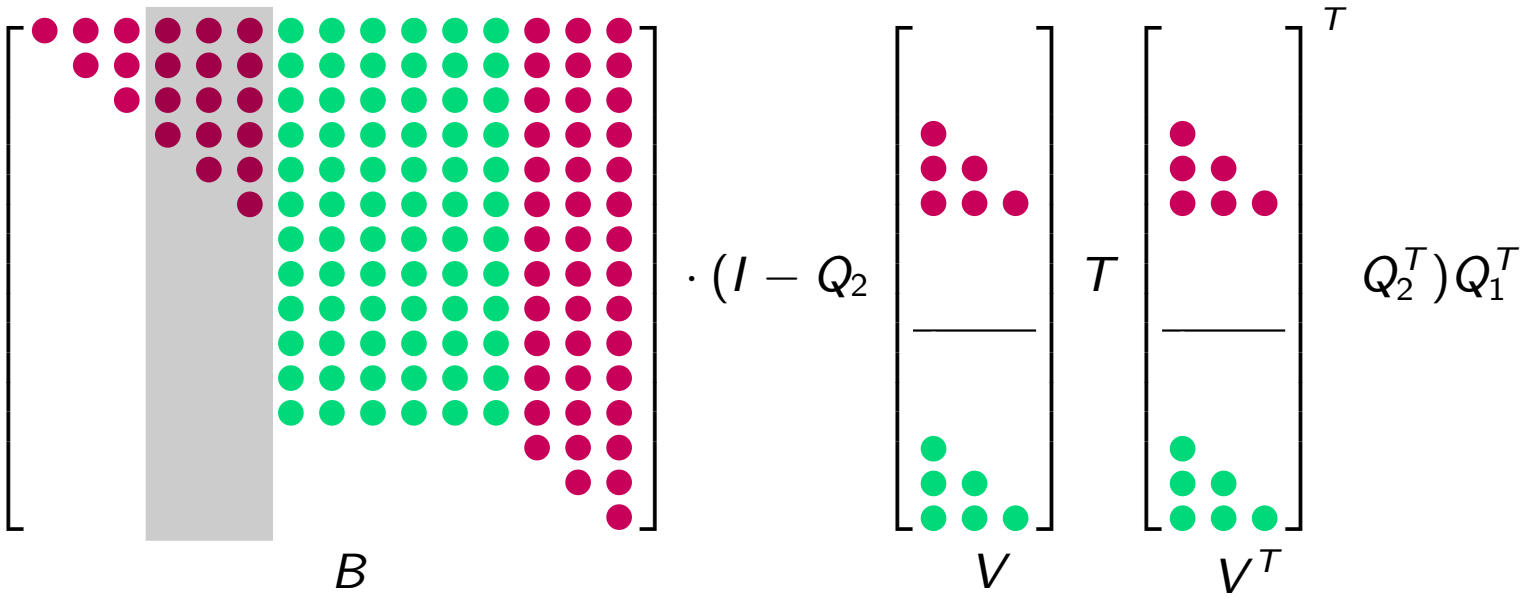
3. Apply the obtained orthogonal factor Q_1 to $B \rightsquigarrow$ a bulge of size $2\mathtt{nb} \times 2\mathtt{nb}$.



HT-reduction: Absorbing block reflectors

Thus we apply left/right block reflectors in stages.
We show the computation of $B \leftarrow B(I - VTV^T)$.

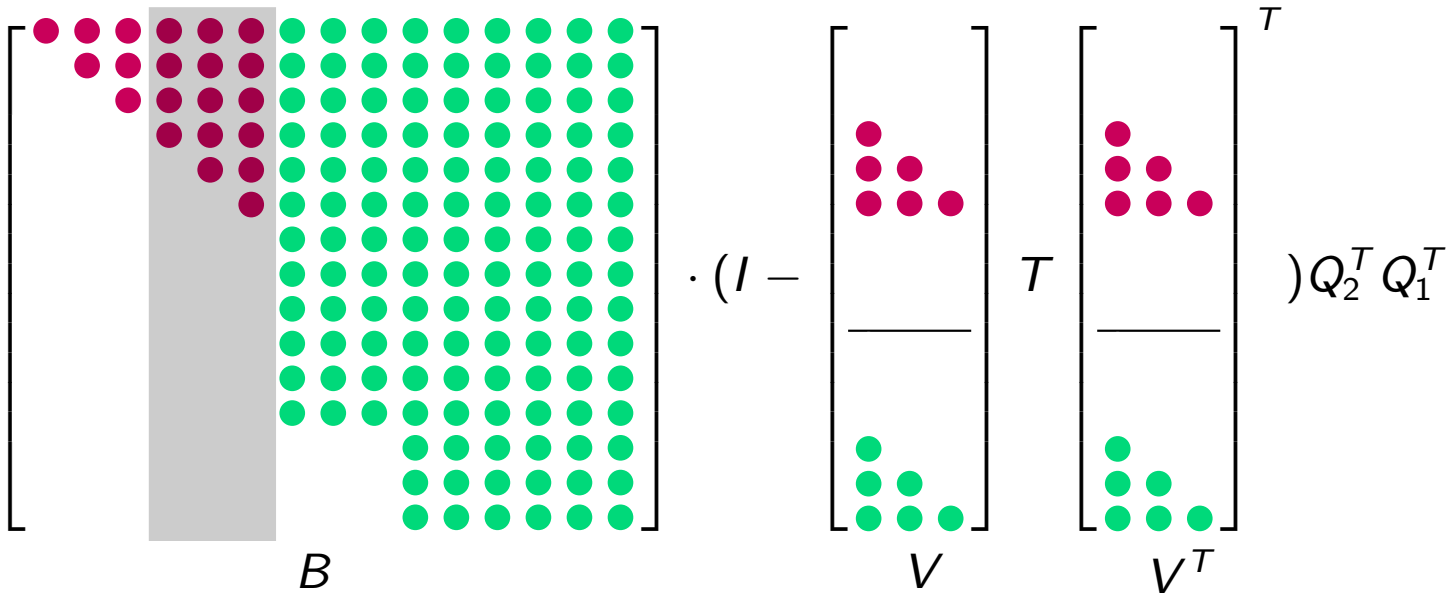
4. Compute the QL factorization of the next $2\texttt{nb}$ rows in V.



HT-reduction: Absorbing block reflectors

Thus we apply left/right block reflectors in stages.
We show the computation of $B \leftarrow B(I - VTV^T)$.

5. Apply the obtained orthogonal factor Q_2 to $B \rightsquigarrow$ a new bulge.



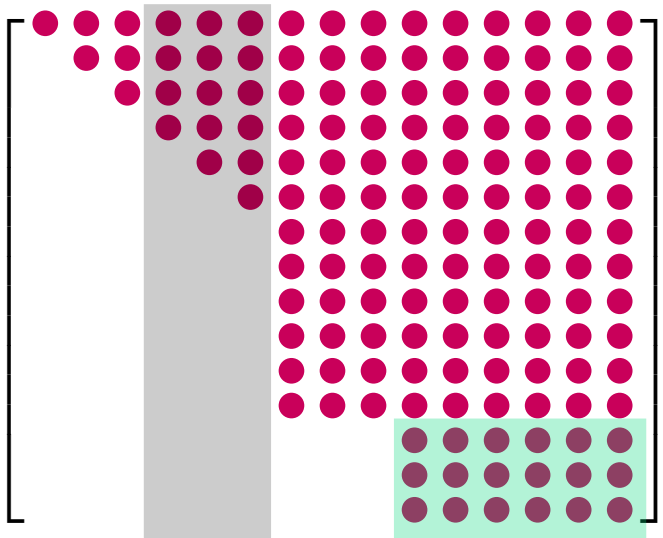
Note:

- The orthogonal factors Q_1, Q_2 of QL factorizations are applied to A without destroying its Hessenberg structure.
- Absorbing the rest of $I - VTV^T$ does not introduce further bulges to B.

HT-reduction: Absorbing block reflectors

Now we can restore the upper triangular structure of B far more cheaply:
$\rightsquigarrow$ the bulges are flattened by a series of small RQ factorizations from bottom to top.

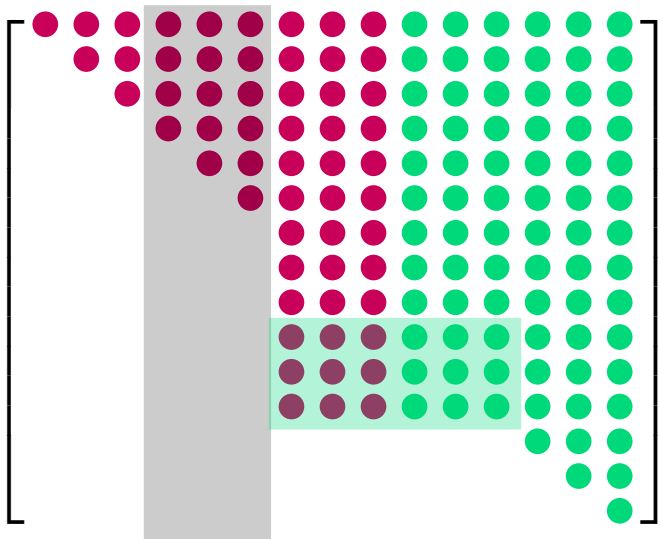
1. Compute the RQ factorization of the bottom `nb` x `2nb` block.



HT-reduction: Absorbing block reflectors

Now we can restore the upper triangular structure of B far more cheaply:
$\rightsquigarrow$ the bulges are flattened by a series of small RQ factorizations from bottom to top.

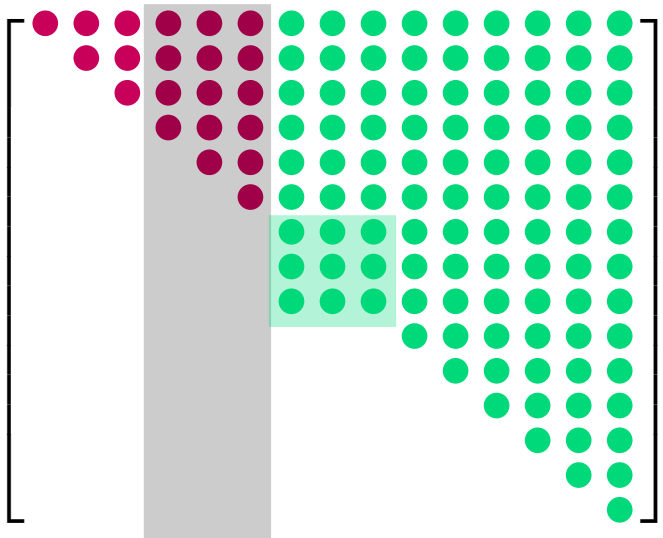
2. Compute the RQ factorization of the next `nb` x `2nb` block.



HT-reduction: Absorbing block reflectors

Now we can restore the upper triangular structure of B far more cheaply:
$\rightsquigarrow$ the bulges are flattened by a series of small RQ factorizations from bottom to top.

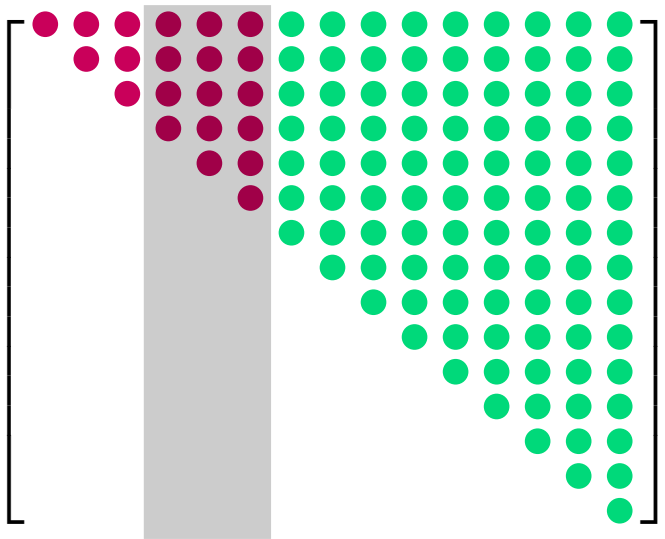
3. Finish with the RQ factorization of the remaining `nb` x `nb` block.



HT-reduction: Absorbing block reflectors

Now we can restore the upper triangular structure of B far more cheaply:
$\rightsquigarrow$ the bulges are flattened by a series of small RQ factorizations from bottom to top.

4. Now B is upper triangular once again.



Absorbing the left block reflector $I - USU^T$ is done analogously.

Further improvements

A number of other improvements to increase efficiency:

1. Keep B upper block triangular, instead of triangular.
   - It is cheaper to restore B only to block triangular after each panel.
   - Maintain LU factorizations of each diagonal block to speed up solving with B.
2. During absorption, use QL factorizations of $\ell\mathtt{nb}\times\mathtt{nb}$ submatrices of V, $\ell > 2$.
3. Use *regular* instead of *compact* WY representation of certain block reflectors.
   - During block reflector absorption phase, small reflectors coming from QL/RQ have zero structure that can be exploited.
   - Also, there the number of reflectors in the block is close to its length.
   - It is more efficient to use $I - VW^T$ form instead of $I - VTV^T$.
4. Preprocess zero columns of B in the beginning.
   - If B has zero columns, apply a column permutation to move them to the front.
   - Then apply the reduction starting with the first non-zero column of B.

Numerical experiments

The algorithm `HouseHT` implemented in C++.
Tested on two machines, with two different BLAS each.

pascal

- 2x Intel Xeon E5-2690 v3 (2x 12 cores, 2.6GHz), 256GB RAM
- Centos 7.3
- variant 1: icpc 16.0.3 + MKL 11.3.3
- variant 2: g++ 4.8.5 + OpenBLAS 0.2.19

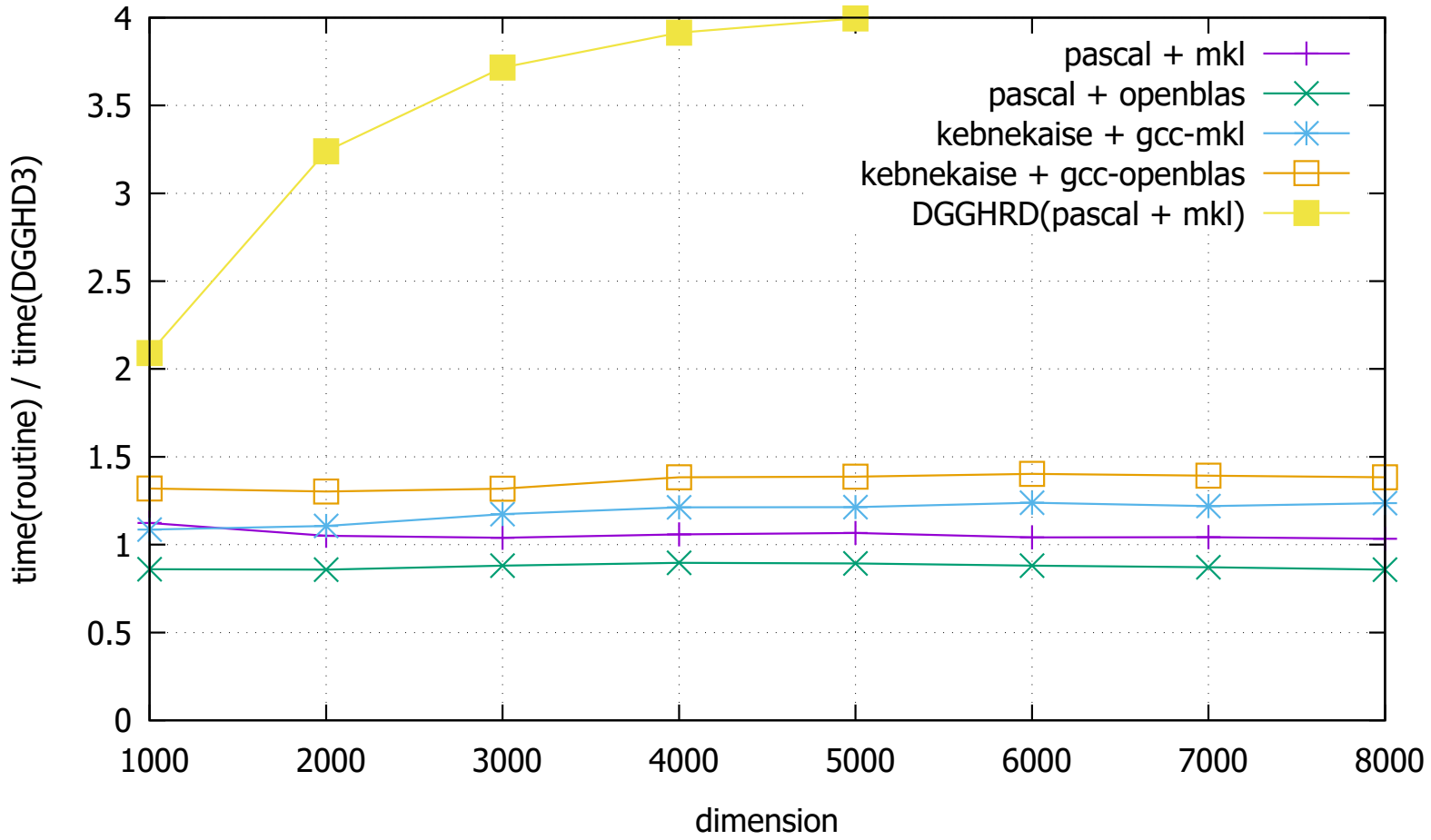
kebnekaise

- 2x Intel Xeon E5-2690 v4 (2x 14 cores, 2.6GHz), 128GB RAM
- Ubuntu 16.04
- variant 1: g++ 6.4.0 + MKL 2017.3.196
- variant 2: g++ 6.4.0 + OpenBLAS 0.2.20

Test suite 1: Random matrices

Using sequential BLAS libraries.

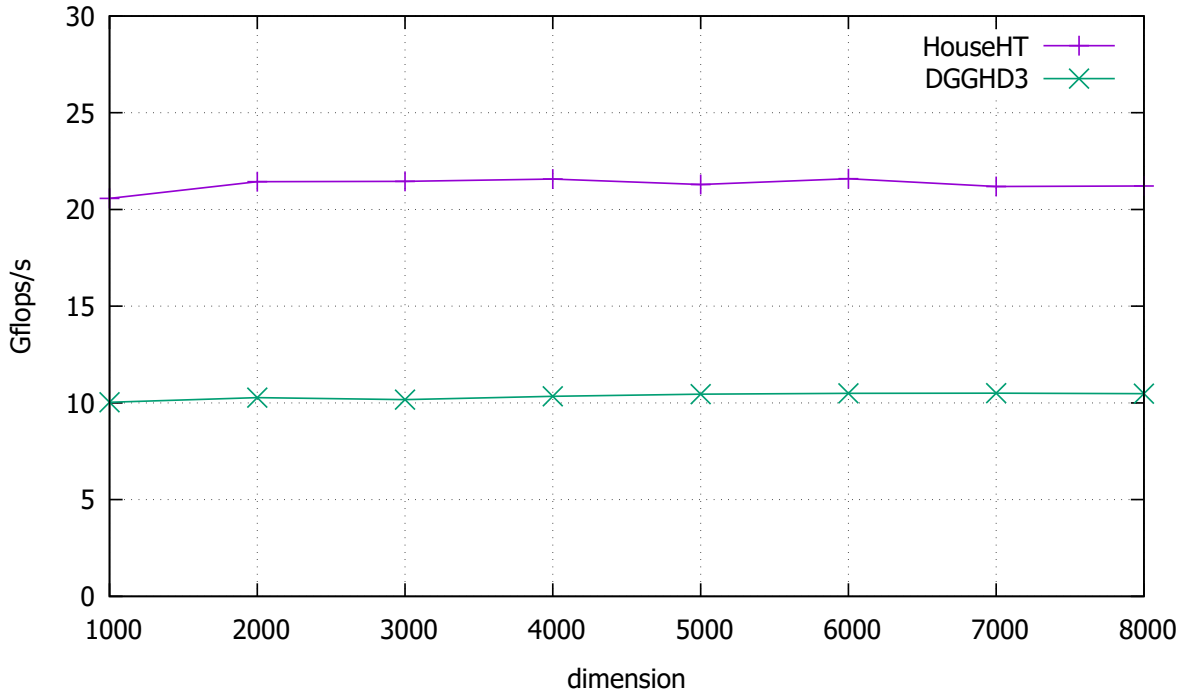
$A \rightsquigarrow$ normally distributed entries.
$B \rightsquigarrow$ R-factor of the QR decomposition of a matrix with normally distributed entries.



Test suite 1: Random matrices

Using sequential BLAS libraries.

$A \rightsquigarrow$ normally distributed entries.
$B \rightsquigarrow$ R-factor of the QR decomposition of a matrix with normally distributed entries.

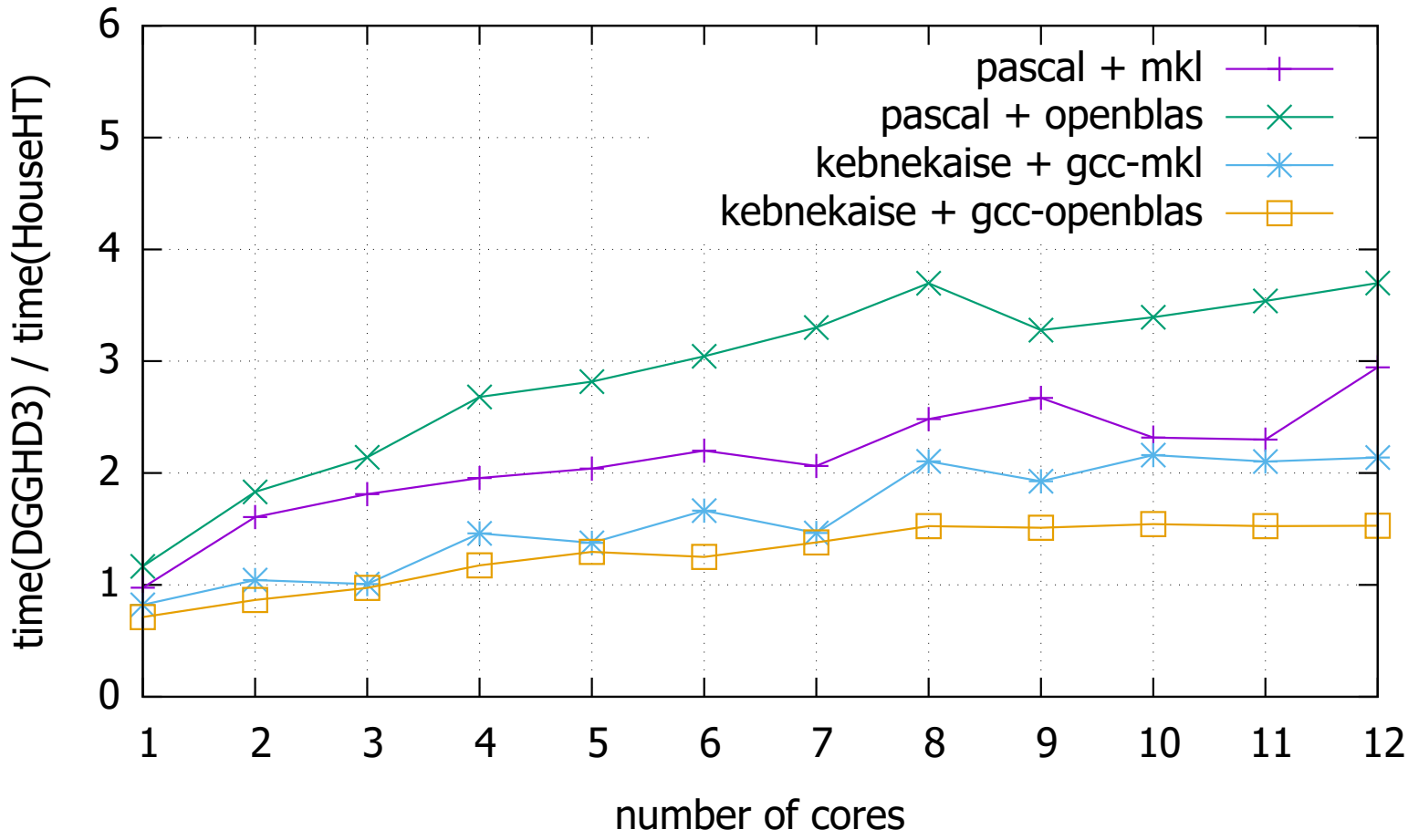


HouseHT $\rightsquigarrow$ 92.60% flops (and 52.77% time) doing level 3 BLAS.
DGGHD3 $\rightsquigarrow$ 65.35% flops (and 18.33% time) doing level 3 BLAS.

Test suite 2: Potential for parallelization

Using multithreaded BLAS libraries.

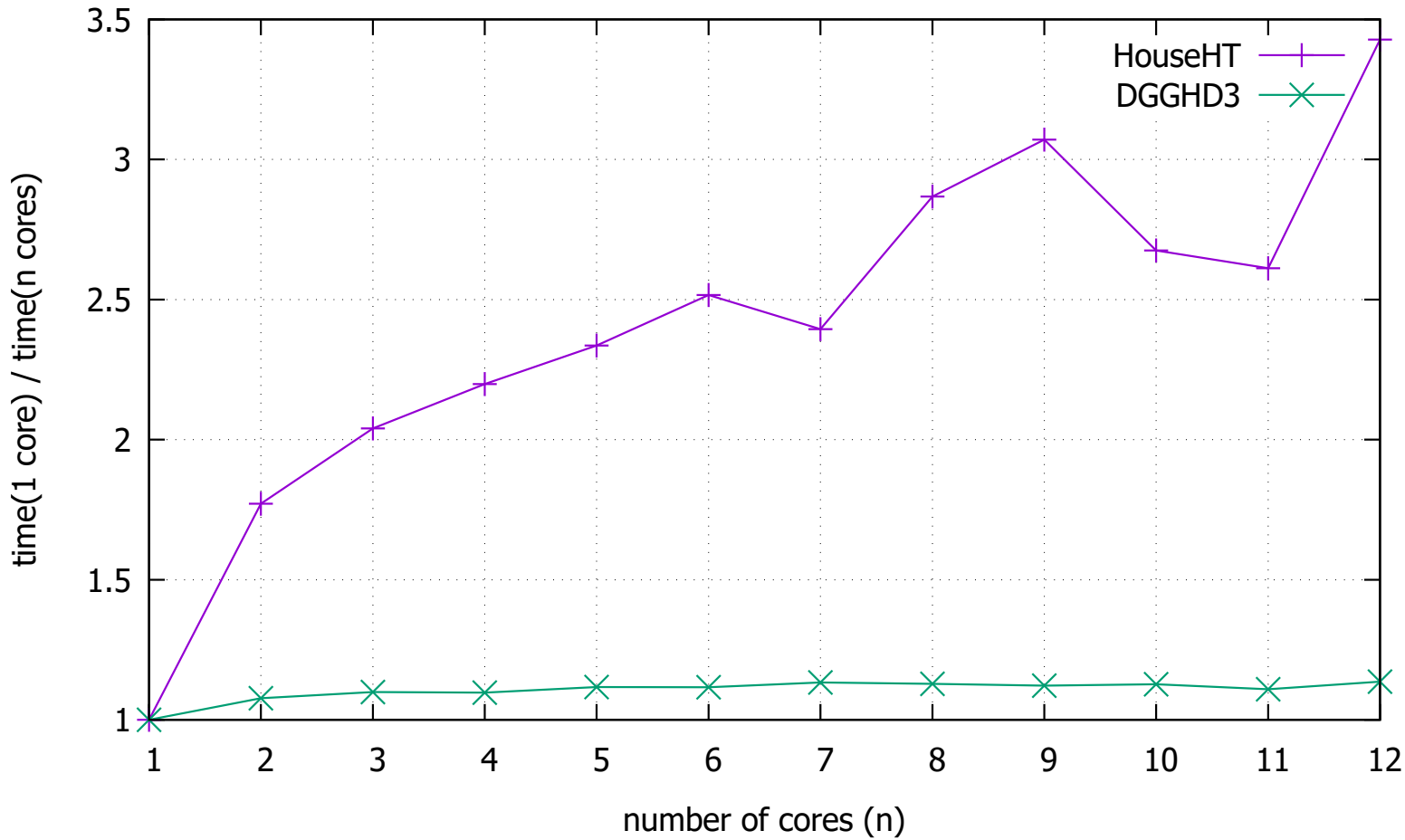
$A \rightsquigarrow$ normally distributed entries.
$B \rightsquigarrow$ R-factor of the QR decomposition of a matrix with normally distributed entries.



Test suite 2: Potential for parallelization

Using multithreaded BLAS libraries.

$A \rightsquigarrow$ normally distributed entries.
$B \rightsquigarrow$ R-factor of the QR decomposition of a matrix with normally distributed entries.



Test suite 3: Benchmark collections

name	n	time(HouseHT)/ time(DGGHD3) (1 core)	time(HouseHT)/ time(DGGHD3) (8 cores)	% cols with extra IR steps	avg. #IR steps per column
BCSST20	485	1.30	1.36	52.58	0.52
MNA_1	578	1.04	1.31	42.39	1.02
BFW782	782	1.18	0.90	0.00	0.00
BCSST19	817	0.98	1.03	55.57	0.55
MNA_4	980	1.05	0.91	34.39	0.42
BCSST08	1074	1.11	0.99	15.08	0.15
BCSST09	1083	1.13	0.93	43.49	0.43
BCSST10	1086	1.17	0.85	16.94	0.17
BCSST27	1224	1.11	0.74	24.43	0.24
RAIL	1357	1.03	0.71	0.52	0.00
SPIRAL	1434	1.04	0.68	0.00	0.00
BCSST11	1473	1.05	0.67	7.81	0.08
BCSST12	1473	1.03	0.67	1.29	0.01
FILTER	1668	1.03	0.62	0.36	0.00
BCSST26	1922	1.05	0.58	20.29	0.20
BCSST13	2003	1.05	0.59	26.21	0.28
PISTON	2025	1.06	0.57	20.79	0.27
BCSST23	3134	1.19	0.56	72.59	0.73
MHD3200	3200	1.16	0.54	26.97	0.27
BCSST24	3562	1.19	0.54	46.97	0.47
BCSST21	3600	1.11	0.48	11.53	0.11

Test suite 4: Saddlepoint matrices

Pencils designed to be particularly unfavorable for HouseHT:

$$A - \lambda B = \begin{bmatrix} X & Y \\ Y^T & 0 \end{bmatrix} - \lambda \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix};$$

X = random symmetric positive definite matrix X, 3/4 of the dimension of A.
Y = random (full-rank) matrix.

n	time(HouseHT)/ time(DGGHD3)	% columns with failed IR	% columns with extra IR steps	average #IR steps per column
1000	2.52 (0.91)	4.90 (0.10)	27.40 (0.40)	2.40 (0.02)
2000	1.72 (0.79)	1.60 (0.00)	32.55 (0.80)	1.95 (0.06)
3000	2.01 (0.77)	1.73 (0.00)	35.20 (0.77)	2.06 (0.07)
4000	2.14 (0.78)	1.55 (0.00)	37.10 (0.72)	2.06 (0.06)
5000	2.03 (0.78)	1.24 (0.00)	35.70 (0.72)	1.86 (0.06)
6000	1.91 (0.77)	0.77 (0.00)	38.00 (0.62)	1.83 (0.01)
7000	1.89 (0.76)	0.77 (0.00)	35.76 (0.67)	1.72 (0.05)
8000	1.78 (0.77)	0.54 (0.00)	36.30 (0.66)	1.65 (0.04)

In parenthesis: when preprocessed with reordering the zero columns of B.

Conclusion and outlook

- We have presented a new algorithm for reducing the matrix pencil to Hessenberg-triangular form.
- The algorithm is based on Householder reflectors rather than the Givens rotations.
- Blocking allows for much better usage of level 3 BLAS.
- Performance level on a single core is comparable to LAPACK routine `DGGHD3`.
- However, using multithreaded BLAS results in significant speedup compared to LAPACK.
- The new algorithm has much higher potential for parallelization $\rightsquigarrow$ future work.

Thank you for your attention!