MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Reduced- and Mixed-Precision Block Householder QR Factorisation

Xiaobo Liu

MPI Magdeburg, Germany

Workshop on Approximate Computing in Numerical (Multi-)Linear Algebra
MPI Magdeburg, September 1, 2026

Joint work with **Peter Benner**, **Martin Köhler**, and **Eda Oktay**

Householder QR factorization

$$A \in \mathbb{R}^{m\times n},\ m \geq n,\ \text{compute } A = QR.$$

For *active column* $x_k = A^{(k)}_{k:m,k}$ at step k,

1. choose v_k and β_k so that $H_k = I - \beta_k v_k v_k^T$ with $H_k x_k = \pm\|x_k\|_2 e_1$;
2. apply H_k to the trailing submatrix $A^{(k+1)}_{k:m,k:n} = H_k A^{(k)}_{k:m,k:n}$:

$$A^{(k)} = \left[\begin{array}{c|c|c} R_{1:k-1,\,1:k-1} & \times & \times \\ \hline 0 & a^{(k)}_{kk} & A^{(k)}_{k,\,k+1:n} \\ \hline 0 & A^{(k)}_{k+1:m,\,k} & A^{(k)}_{k+1:m,\,k+1:n} \end{array}\right] \xrightarrow{\widehat{H}_k} \left[\begin{array}{c|c|c} R_{1:k-1,\,1:k-1} & \times & \times \\ \hline 0 & r_{kk} & r_{k,\,k+1:n} \\ \hline 0 & 0 & A^{(k+1)}_{k+1:m,\,k+1:n} \end{array}\right] = A^{(k+1)}.$$

With $\widehat{H}_k = \mathrm{diag}(I_{k-1}, H_k)$, $R = \widehat{H}_n \cdots \widehat{H}_1 A$ and $Q = \widehat{H}_1 \cdots \widehat{H}_n$. (Normwise b'ward stable **[Higham, '02]**)

- **Concerns:** Repeated synchronizations limit scalability; applying $\widehat{H}_k$ is memory-bound **BLAS2** operation
- $\rightsquigarrow$ **Blocking:** group multiple HH reflectors and turn their rank-one updates into a **BLAS3** block update

Householder QR in WY representation

Main idea. Aggregates b panel HH reflectors and update trailing matrix by BLAS3 instead of b BLAS2 steps [**Bischof & Van Loan, '87**]:

$$Q_k^T = H_b \cdots H_2 H_1 = I + W_k Y_k^T$$

For active submatrix $[P_k \mid T_k]$ at block step k,

$$A^{(k)} = \left[\begin{array}{c|c|c} R_{1:(k-1)b,\,1:(k-1)b} & \times & \times \\ \hline 0 & P_k & T_k \end{array}\right] \xrightarrow{Q_k^T} \left[\begin{array}{c|c|c} R_{1:(k-1)b,\,1:(k-1)b} & \times & \times \\ \hline 0 & R_{kk} & \widehat{T}_k(1:b,:) \\ \hline 0 & 0 & [P_{k+1} \mid T_{k+1}] \end{array}\right] = A^{(k+1)}.$$

1. Factor the panel P_k and construct the WY factors.

$$\mathcal{W}_1 = -\beta_1 v_1, \quad \mathcal{Y}_1 = v_1, \qquad \mathcal{W}_j = [H_j \mathcal{W}_{j-1} \mid -\beta_j v_j], \quad \mathcal{Y}_j = [\mathcal{Y}_{j-1} \mid v_j], \quad j = 2, \ldots, b.$$

   Set $W_k = \mathcal{W}_b$ and $Y_k = \mathcal{Y}_b = V_k$; then $\boldsymbol{Q_k^T P_k = \begin{bmatrix} R_{kk} \\ 0 \end{bmatrix}}$ **(panel factorization)**.

2. Update the trailing submatrix T_k by $\boldsymbol{\widehat{T}_k = T_k + W_k(Y_k^T T_k)}$ and continue on $[P_{k+1} \mid T_{k+1}]$.

Payoff: b BLAS2 updates $\longrightarrow$ one BLAS3 block update; then repeat on the next active submatrix

Householder QR in WY representation

Main idea. Aggregates b panel HH reflectors and update trailing matrix by BLAS3 instead of b BLAS2 steps **[Bischof & Van Loan, '87]**:

$$Q_k^T = H_b \cdots H_2 H_1 = I + W_k Y_k^T$$

For active submatrix $[P_k \mid T_k]$ at block step k,

$$A^{(k)} = \left[\begin{array}{c|c|c} R_{1:(k-1)b,\,1:(k-1)b} & \times & \times \\ \hline 0 & P_k & T_k \end{array}\right] \xrightarrow{Q_k^T} \left[\begin{array}{c|c|c} R_{1:(k-1)b,\,1:(k-1)b} & \times & \times \\ \hline 0 & R_{kk} & \widehat{T}_k(1:b,:) \\ \hline 0 & 0 & [P_{k+1} \mid T_{k+1}] \end{array}\right] = A^{(k+1)}.$$

1. Factor the panel P_k and construct the WY factors.

$$\mathcal{W}_1 = -\beta_1 v_1, \quad \mathcal{Y}_1 = v_1, \qquad \mathcal{W}_j = [H_j \mathcal{W}_{j-1} \mid -\beta_j v_j], \quad \mathcal{Y}_j = [\mathcal{Y}_{j-1} \mid v_j], \quad j = 2, \ldots, b.$$

   Set $W_k = \mathcal{W}_b$ and $Y_k = \mathcal{Y}_b = V_k$; then $\boldsymbol{Q_k^T P_k = \begin{bmatrix} R_{kk} \\ 0 \end{bmatrix}}$ **(panel factorization)**.

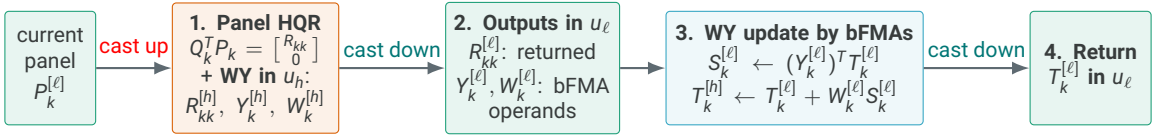
2. Update the trailing submatrix T_k by $\boldsymbol{\widehat{T}_k = T_k + W_k(Y_k^T T_k)}$ and continue on $[P_{k+1} \mid T_{k+1}]$.

Payoff: b BLAS2 updates $\longrightarrow$ one BLAS3 block update; then repeat on the next active submatrix

mpBQR3: High-precision panels, low-precision trailing updates

mpBQR3 [Yang, Fox, & Sanders, '21]:

1. Perform the panel QR factorization and WY construction in u_h
2. Cast the WY factors to u_ℓ and apply bFMAs for the trailing WY update

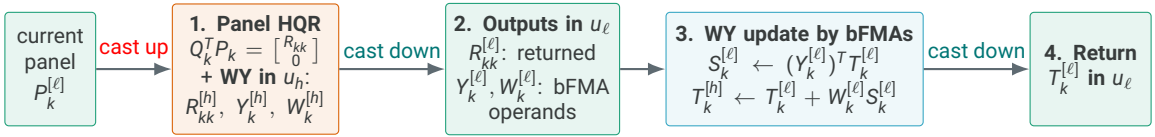


- **High-precision computational cost:** P_k has m_k rows, $\mathcal{O}(m_k b^2)$ per block step $\rightsquigarrow$ overall $\mathcal{O}(mnb)$ flops in u_h (dominant work remains in the bFMA)
- **Casting overhead:** $\mathcal{O}(m_k b)$ per block step $\rightsquigarrow \Theta(mn)$ overall

mpBQR3: High-precision panels, low-precision trailing updates

mpBQR3 [Yang, Fox, & Sanders, '21]:

1. Perform the panel QR factorization and WY construction in u_h
2. Cast the WY factors to u_ℓ and apply bFMAs for the trailing WY update



- **High-precision computational cost:** P_k has m_k rows, $\mathcal{O}(m_k b^2)$ per block step $\rightsquigarrow$ overall $\mathcal{O}(mnb)$ flops in u_h (dominant work remains in the bFMA)
- **Casting overhead:** $\mathcal{O}(m_k b)$ per block step $\rightsquigarrow \Theta(mn)$ overall

Applications of low-precision QR factorization

- **Jacobi SVD preprocessing:** as a preconditioner before low-precision SVD, followed by working-precision reorthogonalization and one-sided Jacobi **[Gao, Ma, & Shao, '25]**
- **Least-squares iterative refinement:** used to obtain the initial solution and construct the preconditioner for GMRES-based iterative refinement **[Carson, Higham, & Pranesh, '20]**
- **QR-based CP–ALS:** solves the least-squares subproblems in CP rounding via ALS **[Zhang et al., '26]**
- Schur-based refinement: initial approximations for **Sylvester equations** **[Dmytryshyn et al., '25]** and **matrix square roots** **[Gao, Kressner, & Shao, '26]** are computed from a low-precision Schur decomposition
- . . .

mpWYQR: High-precision only in norm evaluations

mpWYQR: High-precision used only in norm evaluations within the panel factorization, u_ℓ for all other operations, including WY updates.

For *active column* $x \in \mathbb{R}^\ell$, form one HH reflector:

1. Compute $\rho = \|x\|_2$ in u_h and convert the result to u_ℓ.
2. $\alpha = -\operatorname{sign}(x_1)\rho, \quad w = x - \alpha e_1$
3. Compute $\nu = \|w\|_2$ in u_h and convert the result to u_ℓ.
4. $v = w/\nu, \quad \beta = 2/(v^T v)$

- **High-precision computational cost:** two norm evaluations per column $\rightsquigarrow$ overall $4mn - 2n^2$ flops in u_h
- **Casting overhead:** two scalar conversions *in registers* per column $\rightsquigarrow \mathcal{O}(n)$ overall, with *no array data movement*

mpWYQR: High-precision only in norm evaluations

mpWYQR: High-precision used only in norm evaluations within the panel factorization, u_ℓ for all other operations, including WY updates.

For *active column* $x \in \mathbb{R}^\ell$, form one HH reflector:

1. Compute $\rho = \|x\|_2$ in u_h and convert the result to u_ℓ.
2. $\alpha = -\operatorname{sign}(x_1)\rho, \quad w = x - \alpha e_1$
3. Compute $\nu = \|w\|_2$ in u_h and convert the result to u_ℓ.
4. $v = w/\nu, \quad \beta = 2/(v^T v)$

- **High-precision computational cost:** two norm evaluations per column $\rightsquigarrow$ overall $4mn - 2n^2$ flops in u_h
- **Casting overhead:** two scalar conversions *in registers* per column $\rightsquigarrow \mathcal{O}(n)$ overall, with *no array data movement*

Performance and accuracy for square matrices in fp16–fp32

$m = n$	Method	$\|A - QR\|_F$	$\|Q^TQ - I\|_F$	rel. err. v	rel. err. β	Time (s)
4000	rWYQR	8.56e−2	9.95e−2	6.73e−1	3.59e−2	0.33
	mpWYQR	6.00e−3	7.93e−3	6.33e−1	3.22e−2	**0.32**
	mpBQR3	5.91e−3	7.71e−3	5.20e−1	2.83e−2	0.57
12000	rWYQR	5.43e−1	6.18e−1	8.13e−1	2.10e−2	1.61
	mpWYQR	5.65e−3	7.55e−3	6.72e−1	1.85e−2	**1.50**
	mpBQR3	5.54e−3	7.41e−3	6.89e−1	1.97e−2	5.09
36000	rWYQR	–	–	–	–	–
	mpWYQR	3.51e−3	5.52e−3	7.29e−1	1.33e−2	**30.14**
	mpBQR3	3.34e−3	5.51e−3	7.44e−1	1.30e−2	58.16
42000	rWYQR	–	–	–	–	–
	mpWYQR	3.01e−3	5.23e−3	7.38e−1	1.16e−2	**57.79**
	mpBQR3	3.15e−3	5.16e−3	7.45e−1	1.25e−2	79.61

Setup:

- $A \in \mathbb{R}^{m \times n}$: i.i.d. entries from $\mathcal{U}(-1, 1)$; each column scaled to unit ∞-norm
- Block size: $n_b = 64$
- **rWYQR** uses fp16 throughout (via the Low Precision Fortran library **[Köhler & Benner, '26]**).
- **mpWYQR** changes only the norm evaluations to fp32

- rWYQR overflows at $n = 36000$
- mpWYQR retains mpBQR3-level accuracy, and is about 1.4–3.4× faster

Cost comparison of mpBQR3 and mpWYQR

Block size b, The leading factorization work $F(m,n) = 2mn^2 - \frac{2}{3}n^3$.

	Computation cost in u_h	u_ℓ / bFMA work	Precision-conversion overhead
mpBQR3	$\mathcal{O}(mnb)$	$F(m,n) + \mathcal{O}(mnb)$ bFMA	$\mathcal{O}(mn)$ array entries
mpWYQR	$4mn - 2n^2$	$F(m,n) + \mathcal{O}(mnb)$	0 array; $2n$ scalar conversions

- mpWYQR has less high-precision computational cost & substantially lower precision-conversion overhead than mpBQR3

Loss-of-orthogonality bounds

Lemma: backward error for applying a HH reflector $P = I - \beta vv^T$ to a vector

$$\widehat{y} = \mathrm{fl}_\ell\Big(b - \widehat{\beta}\,\widehat{v}(\widehat{v}^T b)\Big) = (P + \Delta P)b, \quad \|\Delta P\|_F \le \varepsilon_{4m+4,\,4m+25} + \mathcal{O}\big(\varepsilon^2_{4m+4,\,4m+25}\big),$$

provided $\varepsilon_{4m+4,\,4m+25} := (4m+4)u_h + (4m+25)u_\ell < 1$.

mpWYQR:

Provided $\varepsilon_{4m+4,\,4m+25} < 1$,

$$\left\|\widehat{Q} - Q\right\|_F \lesssim n\sqrt{m}\,\varepsilon_{4m+4,\,4m+25},$$
$$\left\|\widehat{Q}^T\widehat{Q} - I\right\|_F \lesssim 2n\sqrt{m}\,\varepsilon_{4m+4,\,4m+25}.$$

mpBQR3:

With $N = n/b$, let $\widetilde{\gamma}_k^{(s)} := \frac{cku_s}{1-cku_s}$, $s \in \{h, \ell\}$ and

$$\left\|\widehat{Q}_B - Q\right\|_F \lesssim \sqrt{n}\left(\widetilde{\gamma}_N^{(\ell)} + n\widetilde{\gamma}_m^{(h)}\right),$$
$$\left\|\widehat{Q}_B^T\widehat{Q}_B - I\right\|_F \lesssim 2\sqrt{n}\left(\widetilde{\gamma}_N^{(\ell)} + n\widetilde{\gamma}_m^{(h)}\right).$$

[Yang, Fox, & Sanders, '21]

- mpWYQR bound not applicable in fp16–fp32 for $m \gtrsim 506$
- Key advantage of mpBQR3: no mu_ℓ factor in the first-order bound

Numerical experiments on an fp16–fp32 CPU platform

Software and hardware environment

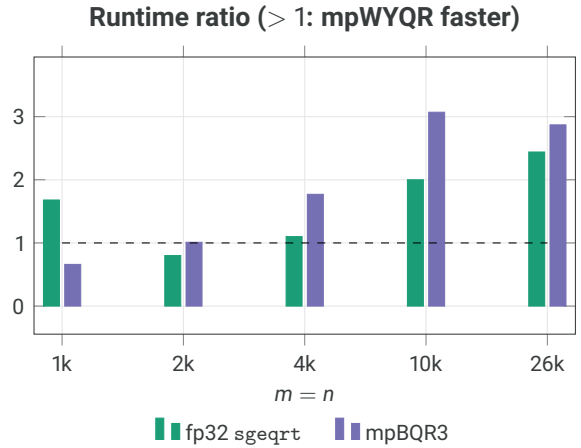
- **Fortran support:** Low Precision Fortran provides fp16 data types, arithmetic, and low-precision BLAS/LAPACK interfaces in Fortran **[Köhler & Benner, '26]**
- **BLAS library:** fp16 `GEMM` is provided by Intel MKL; the implementation is **CPU-only**
- **Compute server:** Intel Xeon Platinum, Emerald Rapids series, with native AVX-512 fp16 arithmetic

Experimental parameters

- **Precision and blocking:** $(u_\ell, u_h) = (\mathrm{fp16}, \mathrm{fp32})$, with $b = 64$
- **Timing:** the average over five runs is reported at each case
- **Test matrices:** $A \in \mathbb{R}^{m\times n}$, $a_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(-1,1)$. Each column is scaled as $a_j \leftarrow a_j / \max_i |a_{ij}|$, so $\|a_j\|_\infty = 1$

Test on square matrices

`sgeqrt`: single-precision LAPACK blocked Householder QR routine



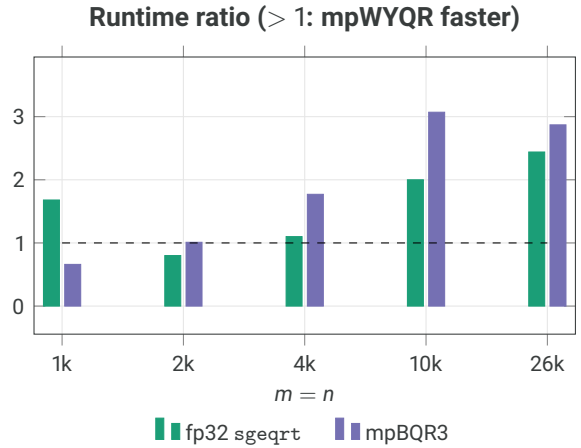
Accuracy range of mpWYQR

$$2.92\text{e}{-3} \le \|A - QR\|_F \le 6.00\text{e}{-3},$$
$$5.13\text{e}{-3} \le \left\|Q^T Q - I\right\|_F \le 7.94\text{e}{-3}.$$

- $n \le 2$k: non-GEMM costs dominate the runtime
- mpWYQR becomes competitive once GEMM dominates ($n \ge 4$k)
- 3.1× faster than mpBQR3 at $n = 10$k
- 2.4× faster than `sgeqrt` at $n = 26$k

Test on square matrices

`sgeqrt`: single-precision LAPACK blocked Householder QR routine

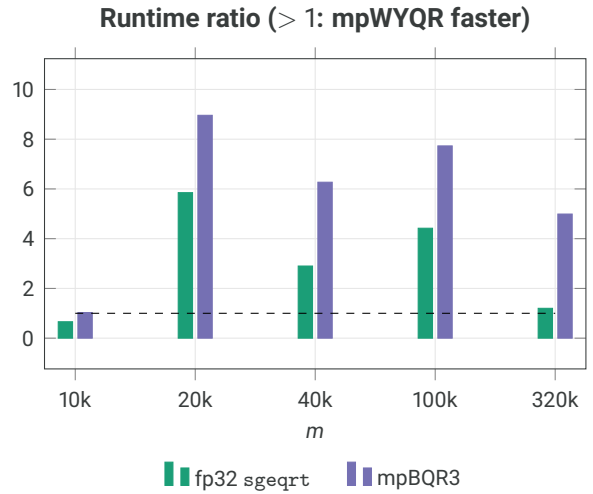


Accuracy range of mpWYQR

$$2.92\text{e}{-3} \le \|A - QR\|_F \le 6.00\text{e}{-3},$$
$$5.13\text{e}{-3} \le \left\|Q^T Q - I\right\|_F \le 7.94\text{e}{-3}.$$

- $n \le 2$k: non-GEMM costs dominate the runtime
- mpWYQR becomes competitive once GEMM dominates ($n \ge 4$k)
- 3.1× faster than mpBQR3 at $n = 10$k
- 2.4× faster than `sgeqrt` at $n = 26$k

Test on tall–skinny matrices

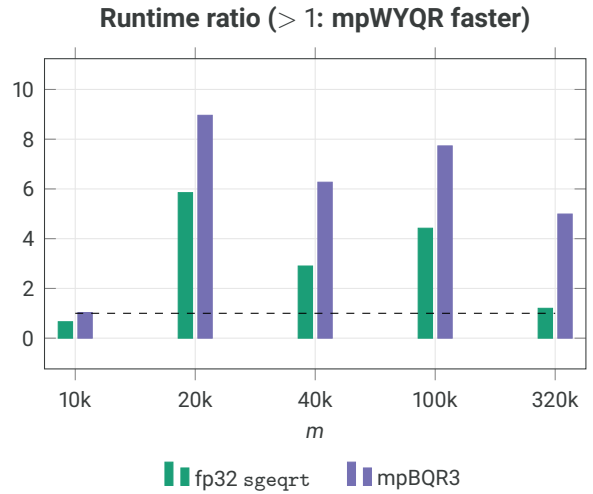


Accuracy range of mpWYQR

$$6.69\text{e}{-4} \leq \|A - QR\|_F \leq 1.10\text{e}{-3},$$
$$5.67\text{e}{-5} \leq \left\|Q^T Q - I\right\|_F \leq 1.83\text{e}{-4}.$$

- mpWYQR *up to* $9.0\times$ faster than mpBQR3 and $5.9\times$ faster than `sgeqrt`
- rWYQR overflows from $m = 40k$ (time ratio not reported)
- The drop near 320k is due to lower MKL fp16 `GEMM` performance on tall–skinny matrices

Test on tall–skinny matrices



Accuracy range of mpWYQR

$$6.69\text{e}{-4} \leq \|A - QR\|_F \leq 1.10\text{e}{-3},$$
$$5.67\text{e}{-5} \leq \left\|Q^TQ - I\right\|_F \leq 1.83\text{e}{-4}.$$

- mpWYQR *up to* 9.0× faster than mpBQR3 and 5.9× faster than `sgeqrt`
- rWYQR overflows from $m = 40k$ (time ratio not reported)
- The drop near 320k is due to lower MKL fp16 `GEMM` performance on tall–skinny matrices

Conclusions

We discussed three reduced- and mixed-precision block Householder QR factorizations:

- mpWYQR evaluates norms in high-precision and *stabilizes* reflector construction by avoiding fp16 range failures (as observed on rWYQR) $\rightsquigarrow$ remains u_ℓ-level accuracy
- mpWYQR is less memory-bound than mpBQR3 and is faster (up to $3.1\times$ for $m = n$ and $9.0\times$ for $m \gg n$).
- Existing error bounds for mpWYQR in general too *pessimistic* to explain the observed numerical behavior
- GPU implementation? Potential to outperform the CPU implementation while retaining comparable numerical accuracy
- Preprint on the way :), Ongoing work: stress tests on more badly-scaled matrices from `BlockStab` **[BlockStab, '24]**

Thanks you — Questions / Comments?

Bibliography

C. Bischof and C. Van Loan.
The WY representation for products of Householder matrices.
SIAM J. Sci. Stat. Comput., 8(1):S2–S13, 1987.

L. M. Yang, A. Fox, and G. Sanders.
Rounding error analysis of mixed precision block Householder QR algorithms.
SIAM J. Sci. Comput., 43(3):A1723–A1753, 2021.

N. J. Higham.
Accuracy and Stability of Numerical Algorithms, 2nd ed.
SIAM, 2002.

M. Köhler and P. Benner.
Low precision Fortran—enabling low precision floating point arithmetic in modern Fortran.
arXiv:2606.16709, 2026.

K. Lund, E. Oktay, E. Carson, and Y. Ma.
BlockStab.
https://github.com/katlund/BlockStab, 2026.

W. Gao, Y. Ma, and M. Shao.
A mixed precision Jacobi SVD algorithm.
ACM Trans. Math. Software, 51(1), Art. 5, 2025.

E. Carson, N. J. Higham, and S. Pranesh.
Three-precision GMRES-based iterative refinement for least squares problems.
SIAM J. Sci. Comput., 42(6):A4063–A4083, 2020.

A. Dmytryshyn, M. Fasi, N. J. Higham, and X. Liu.
Mixed-precision algorithms for solving the Sylvester matrix equation.
arXiv:2503.03456, 2025; to appear in *SIAM J. Sci. Comput.*

B. Gao, D. Kressner, and M. Shao.
A mixed precision algorithm for the matrix square root.
arXiv:2607.12430, 2026.

A. Zhang, B. D. Verma, J. Van Lent, and G. Ballard.
Efficient CP rounding using alternating least squares with QR decomposition.
SIAM J. Matrix Anal. Appl., 47(2):544–563, 2026.