

ENABLING TOLERANCE-AWARE FLEXIBLE ACCURACY VIA OZAKI-II EMULATION : THE CASE OF THE CHASE LIBRARY

2nd of November, 2026 | E. Di Napoli, C. Richefort, X. Wu | Jülich Supercomputing Centre | HANAMI - 3rd HLS |

I. BRIEF OVERVIEW OF CHASE

ChASE
CHEBYSHEV ACCELERATED SUBSPACE EIGENSOLVER

- Multi-GPU Iterative Dense Eigensolver
- Targets Hermitian & Pseudo-Hermitian matrices
- Based on multiple sequences of GEMM operations

Already tested on a broad range of platforms, including :



JUWELS BOOSTER



FUGAKU



FRONTIER



JUPITER

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: NEV wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: NEV wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.
5. Compute the primitive Ritz pairs $(\hat{\Lambda}, Y)$ by solving for $GY = Y\hat{\Lambda}$.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.
5. Compute the primitive Ritz pairs $(\hat{\Lambda}, Y)$ by solving for $GY = Y\hat{\Lambda}$.
6. Compute the approximate Ritz pairs $(\hat{\Lambda}, \hat{V} \leftarrow QY)$.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.
5. Compute the primitive Ritz pairs $(\hat{\Lambda}, Y)$ by solving for $GY = Y\hat{\Lambda}$.
6. Compute the approximate Ritz pairs $(\hat{\Lambda}, \hat{V} \leftarrow QY)$.
7. Compute the residuals of the Ritz vectors $\|A\hat{V} - \hat{V}\hat{\Lambda}\|$.

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

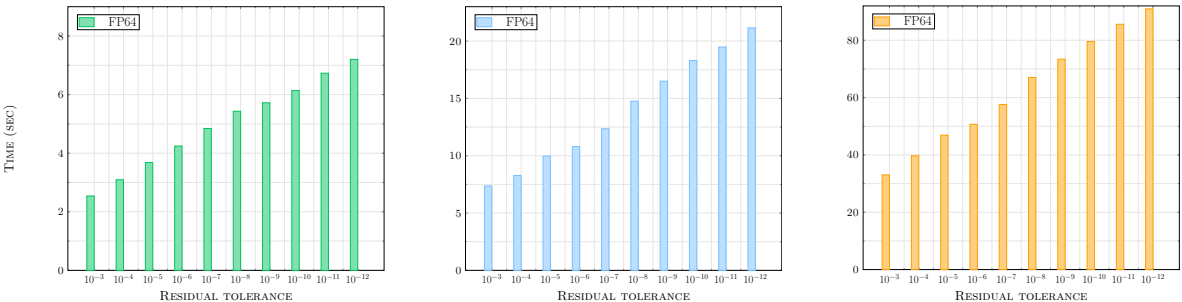
REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.
5. Compute the primitive Ritz pairs $(\hat{\Lambda}, Y)$ by solving for $GY = Y\hat{\Lambda}$.
6. Compute the approximate Ritz pairs $(\hat{\Lambda}, \hat{V} \leftarrow QY)$.
7. Compute the residuals of the Ritz vectors $\|A\hat{V} - \hat{V}\hat{\Lambda}\|$.
8. Deflate and lock the converged vectors.

END REPEAT

I. BRIEF OVERVIEW OF CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



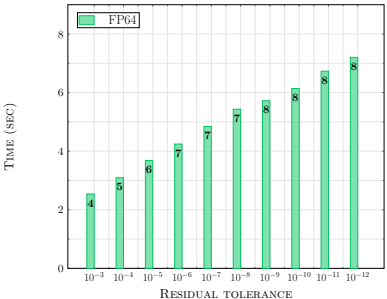
Small - N = 39k, NEV = 1192 | Medium - N = 62k, NEV = 1880 | Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

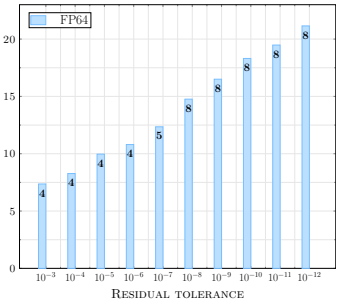
JÜLICH Forschungszentrum

I. BRIEF OVERVIEW OF CHASE

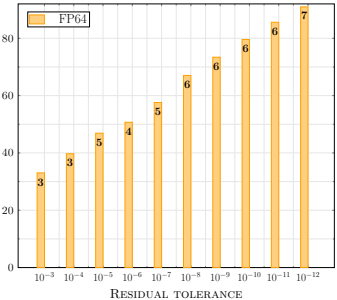
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1880



Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

JÜLICH Forschungszentrum

I. BRIEF OVERVIEW OF CHASE

INPUT: Hermitian matrix A, `tol`,`deg` OUTPUT: `NEV` wanted eigenpairs (Λ, V).

1. Identify the parameters $\{\mu_1, \mu_{\text{nevex}}, \mu_n\}$ with Lanczos

REPEAT UNTIL CONVERGENCE:

2. Filter a block of vectors $\hat{V} \longleftarrow p(A)\hat{V}$ with optimal degree.
3. Re-orthogonalize the vectors outputted by the filter; $\hat{V} = QR$.
4. Compute the Rayleigh quotient $G = Q^* AQ$.
5. Compute the primitive Ritz pairs $(\hat{\Lambda}, Y)$ by solving for $GY = Y\hat{\Lambda}$.
6. Compute the approximate Ritz pairs $(\hat{\Lambda}, \hat{V} \leftarrow QY)$.
7. Compute the residuals of the Ritz vectors $\|A\hat{V} - \hat{V}\hat{\Lambda}\|$.
8. Deflate and lock the converged vectors.

END REPEAT

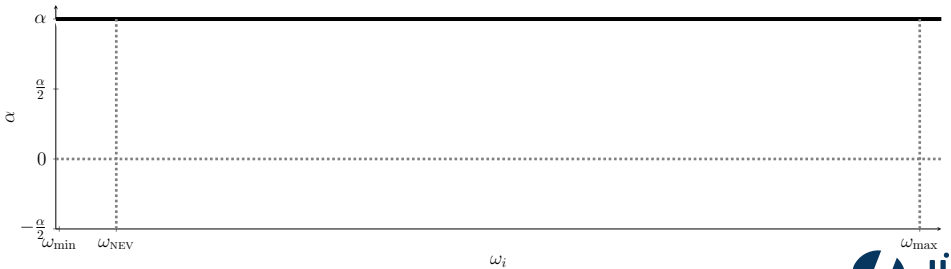
I. BRIEF OVERVIEW OF CHASE

Let a random vector $\boldsymbol{r}$ be decomposed into the eigen basis of A. Therefore,

$$\boldsymbol{r} = \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i \qquad \text{with} \qquad A\boldsymbol{v}_i = \omega_i \boldsymbol{v}_i.$$

Since $\boldsymbol{r}$ is random, let assume $\alpha_i = \alpha$. Hence, applying a polynomial of A to $\boldsymbol{r}$ gives

$$A\boldsymbol{r} = \sum_{i=1}^{m} \alpha\lambda_i \boldsymbol{v}_i \quad , \quad A^2\boldsymbol{r} = \sum_{i=1}^{m} \alpha\lambda_i^2 \boldsymbol{v}_i \quad , \quad p(A)\boldsymbol{r} = \sum_{i=1}^{m} \alpha p(\lambda_i) \boldsymbol{v}_i$$



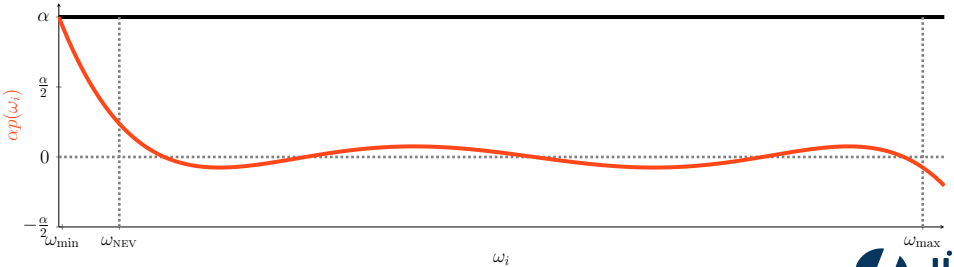
I. BRIEF OVERVIEW OF CHASE

Let a random vector $\boldsymbol{r}$ be decomposed into the eigen basis of A. Therefore,

$$\boldsymbol{r} = \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i \qquad \text{with} \qquad A\boldsymbol{v}_i = \omega_i \boldsymbol{v}_i.$$

Since $\boldsymbol{r}$ is random, let assume $\alpha_i = \alpha$. Hence, applying a polynomial of A to $\boldsymbol{r}$ gives

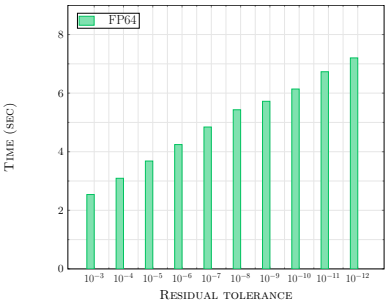
$$A\boldsymbol{r} = \sum_{i=1}^{m} \alpha\lambda_i \boldsymbol{v}_i \quad , \quad A^2\boldsymbol{r} = \sum_{i=1}^{m} \alpha\lambda_i^2 \boldsymbol{v}_i \quad , \quad p(A)\boldsymbol{r} = \sum_{i=1}^{m} \alpha p(\lambda_i)\boldsymbol{v}_i$$



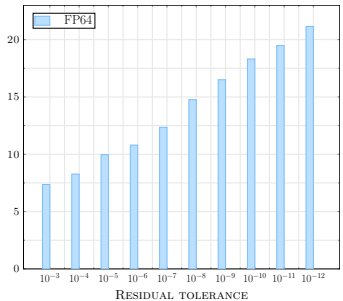
JÜLICH Forschungszentrum

II. THE FLEXIBLE RESIDUAL TOLERANCE EXPERIMENT

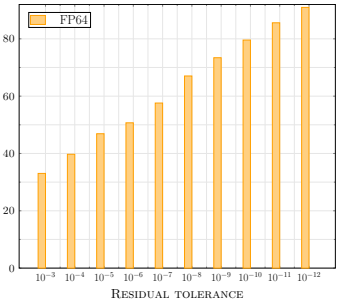
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1180



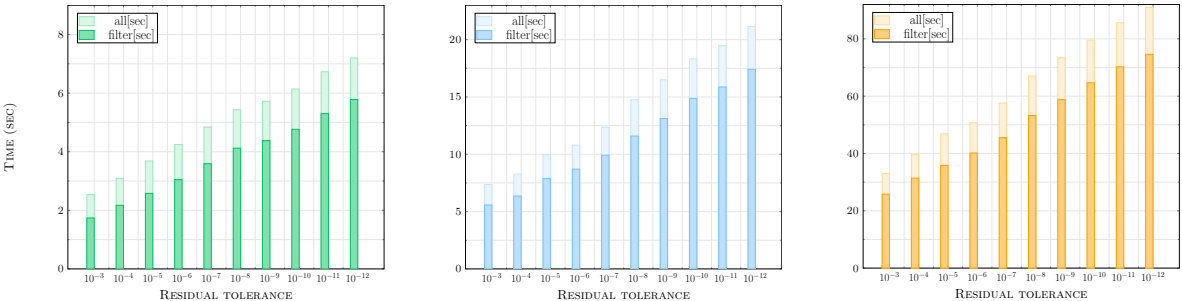
Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

JÜLICH Forschungszentrum

II. THE FLEXIBLE RESIDUAL TOLERANCE EXPERIMENT

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



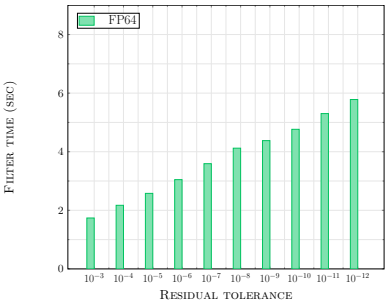
Small - N = 39k, NEV = 1192 | Medium - N = 62k, NEV = 1180 | Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

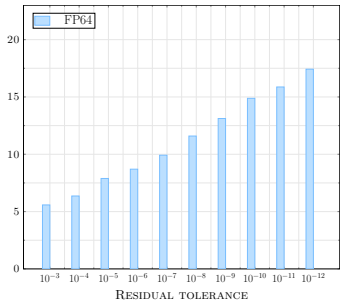
JÜLICH Forschungszentrum

II. THE FLEXIBLE RESIDUAL TOLERANCE EXPERIMENT

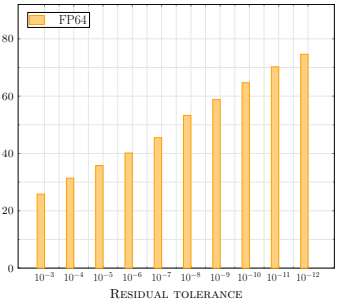
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1180



Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

Instead of computing the polynomial itself, we leverage on the three-term recurrence relation

$$V_{i+1} = \alpha_i A V_i + \beta_i V_{i-1} \quad \text{with} \quad V_1 = A V_0 \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 GEMM('N','N',alpha,A,V0,0,V1)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   GEMM('N','N',alpha,A,V1,beta,V0) //V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```


JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

Instead of computing the polynomial itself, we leverage on the three-term recurrence relation

$$V_{i+1} = \alpha_i A V_i + \beta_i V_{i-1} \quad \text{with} \quad V_1 = A V_0 \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 GEMM('N','N',alpha,A,V0,0,V1)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   GEMM('N','N',alpha,A,V1,beta,V0) //V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```

GEMM operations ($C \leftarrow \alpha A \times B + \beta C$) represent around 80% of the FLOPS !

JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

Instead of computing the polynomial itself, we leverage on the three-term recurrence relation

$$V_{i+1} = \alpha_i A V_i + \beta_i V_{i-1} \quad \text{with} \quad V_1 = A V_0 \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 OZAKI_GEMM('N','N',alpha,A,V0,0,V1,s)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   OZAKI_GEMM('N','N',alpha,A,V1,beta,V0,s)//V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```

GEMM operations ($C \leftarrow \alpha A \times B + \beta C$) represent around 80% of the FLOPS !

JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

1. Scale and Truncate :

$$A' = \mathrm{TRUNC}(2^{\mathrm{DIAG}(\boldsymbol{u})} \times A) \qquad \text{and} \qquad B' = \mathrm{TRUNC}(B \times 2^{\mathrm{DIAG}(\boldsymbol{v})})$$

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

1. Scale and Truncate :

$$A' = \text{TRUNC}(2^{\text{DIAG}(\boldsymbol{u})} \times A) \qquad \text{and} \qquad B' = \text{TRUNC}(B \times 2^{\text{DIAG}(\boldsymbol{v})})$$

2. Generate the INT8 matrices

$$A_i = A' \bmod p_i \qquad \text{and} \qquad B_i = B' \bmod p_i, \quad \text{with} \quad i \leq s.$$

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

1. Scale and Truncate :

$$A' = \text{TRUNC}(2^{\text{DIAG}(\boldsymbol{u})} \times A) \qquad \text{and} \qquad B' = \text{TRUNC}(B \times 2^{\text{DIAG}(\boldsymbol{v})})$$

2. Generate the INT8 matrices

$$A_i = A' \bmod p_i \qquad \text{and} \qquad B_i = B' \bmod p_i, \quad \text{with} \quad i \leq s.$$

3. Compute the INT8 output matrices

$$C_i = A_i B_i \bmod p_i, \quad \text{with} \quad i \leq s.$$

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

1. Scale and Truncate :

$$A' = \mathrm{TRUNC}(2^{\mathrm{DIAG}(\boldsymbol{u})} \times A) \qquad \text{and} \qquad B' = \mathrm{TRUNC}(B \times 2^{\mathrm{DIAG}(\boldsymbol{v})})$$

2. Generate the INT8 matrices

$$A_i = A' \bmod p_i \qquad \text{and} \qquad B_i = B' \bmod p_i, \quad \text{with} \quad i \le s.$$

3. Compute the INT8 output matrices

$$C_i = A_i B_i \bmod p_i, \quad \text{with} \quad i \le s.$$

4. Accumulate the results

$$C' = \sum_{i=1}^{s} C_i \left(\frac{P}{p_i} \bmod p_i \right)^{-1} \frac{P}{p_i}, \quad \text{with} \quad P = \prod_{i=1}^{s} p_i.$$

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

Let $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. Given a set $\{p_1, p_2, \ldots, p_s\}$ of s coprime modulis, the steps **[1]** are

1. Scale and Truncate :

$$A' = \mathrm{TRUNC}(2^{\mathrm{DIAG}(\boldsymbol{u})} \times A) \qquad \text{and} \qquad B' = \mathrm{TRUNC}(B \times 2^{\mathrm{DIAG}(\boldsymbol{v})})$$

2. Generate the INT8 matrices

$$A_i = A' \bmod p_i \qquad \text{and} \qquad B_i = B' \bmod p_i, \quad \text{with} \quad i \le s.$$

3. Compute the INT8 output matrices

$$C_i = A_i B_i \bmod p_i, \quad \text{with} \quad i \le s.$$

4. Accumulate the results

$$C' = \sum_{i=1}^{s} C_i \left(\frac{P}{p_i} \bmod p_i \right)^{-1} \frac{P}{p_i}, \quad \text{with} \quad P = \prod_{i=1}^{s} p_i.$$

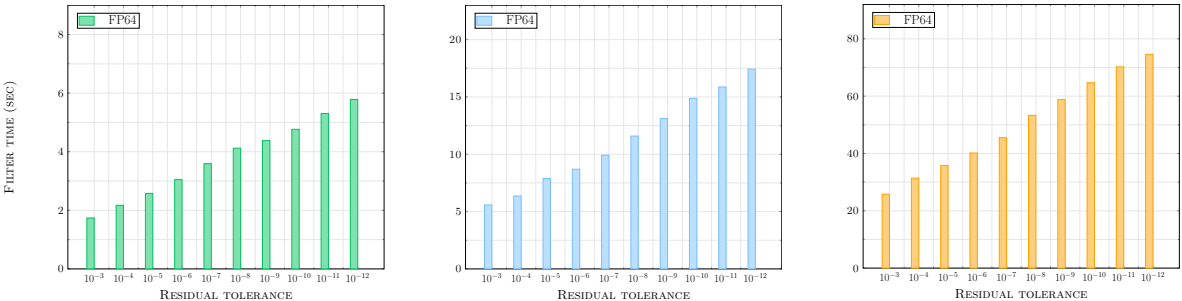
5. Unscale the accumulation

$$C = AB \approx 2^{\mathrm{DIAG}(-\boldsymbol{u})} C' 2^{\mathrm{DIAG}(-\boldsymbol{v})}.$$

[1] : *High-Performance and Power-Efficient Emulation of Matrix Multiplication using INT8 Matrix Engines*, *Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2025*

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192

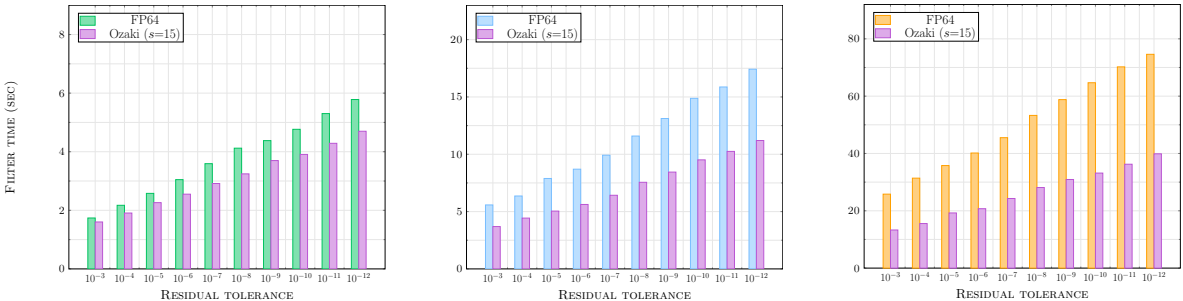
Medium - N = 62k, NEV = 1880

Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



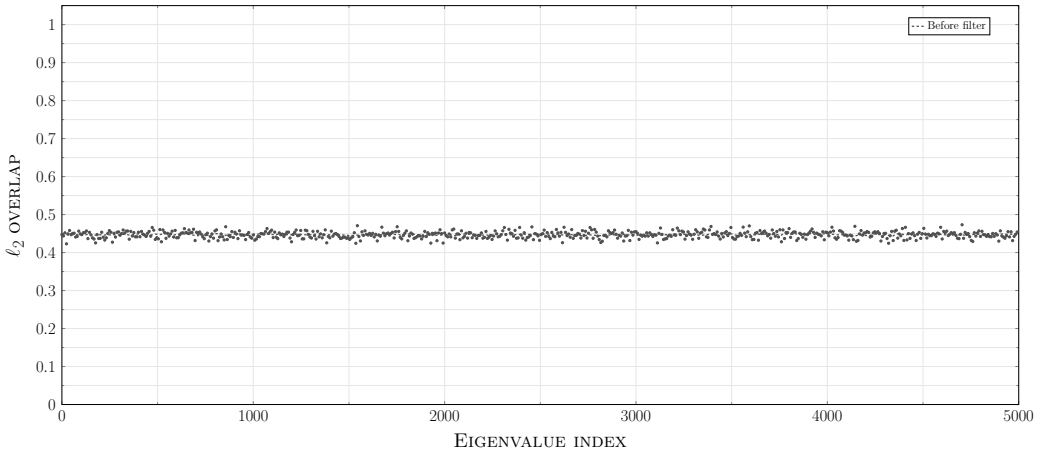
Small - N = 39k, NEV = 1192

Medium - N = 62k, NEV = 1880

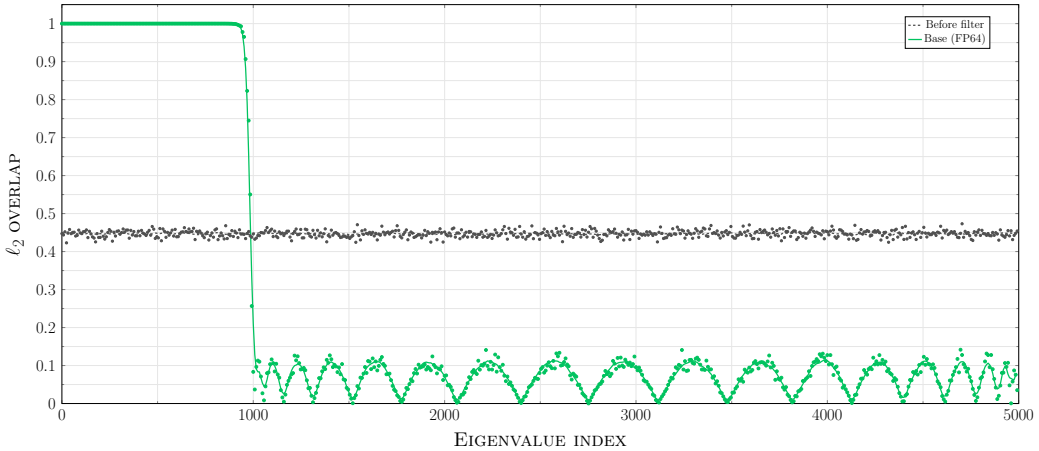
Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance*

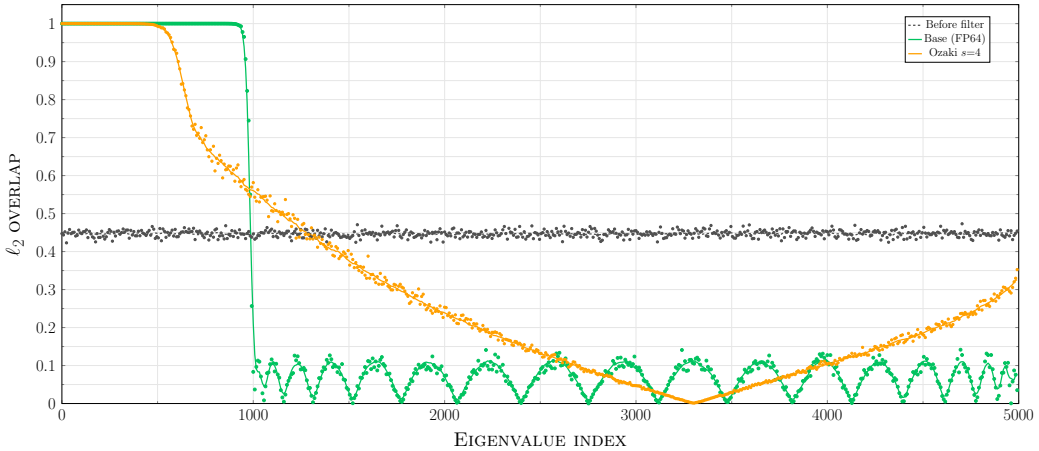
III. OZAKI-II SCHEME INTO CHASE



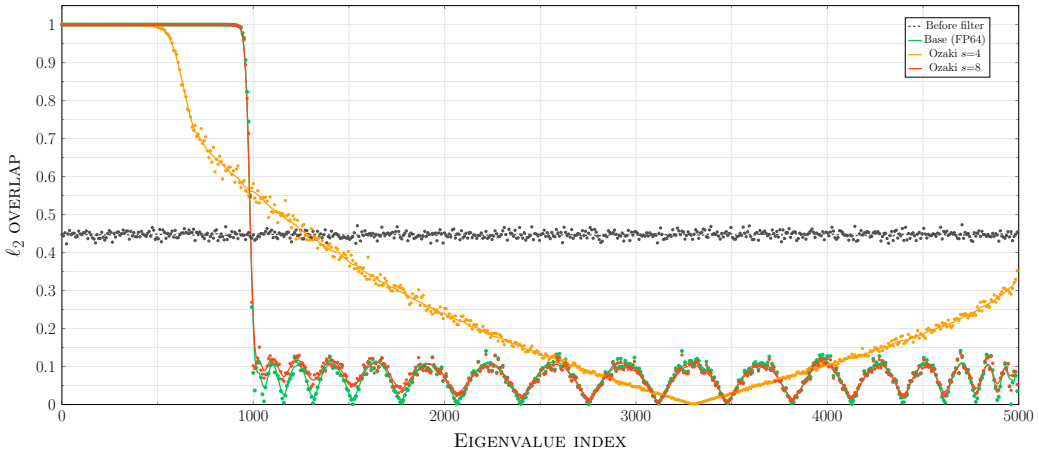
III. OZAKI-II SCHEME INTO CHASE



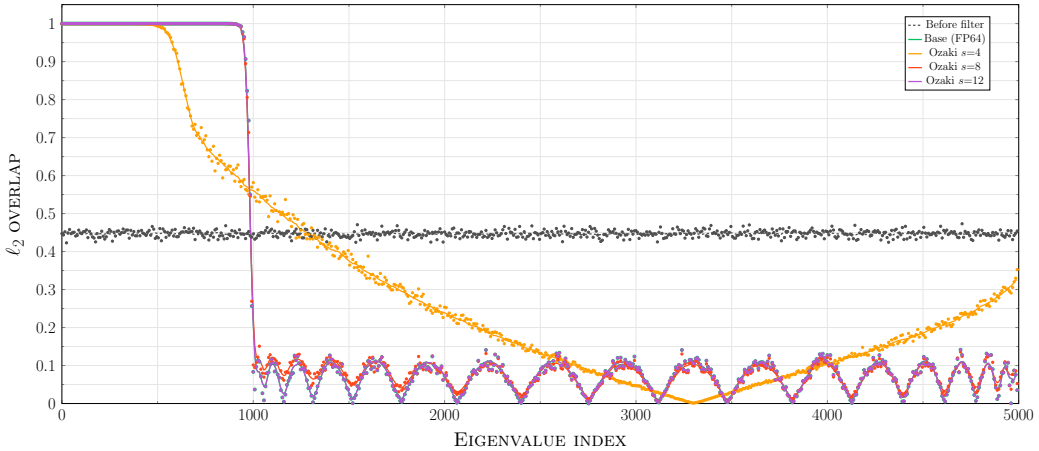
III. OZAKI-II SCHEME INTO CHASE



III. OZAKI-II SCHEME INTO CHASE



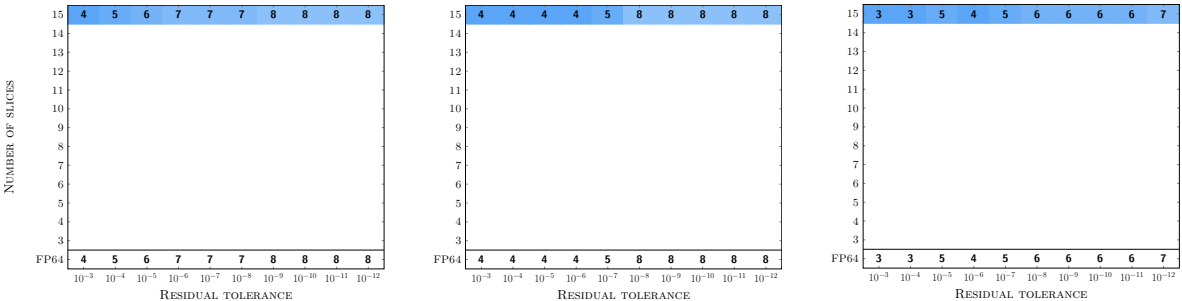
III. OZAKI-II SCHEME INTO CHASE



JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



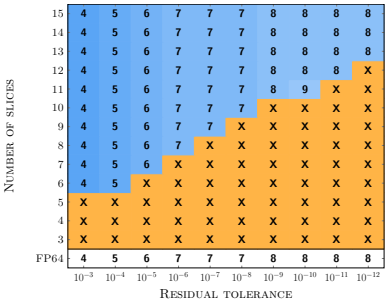
Small - N = 39k, NEV = 1192 | Medium - N = 62k, NEV = 1880 | Large - N = 115k, NEV = 3463

Figure : *Iteration* vs. *Residual Tolerance*

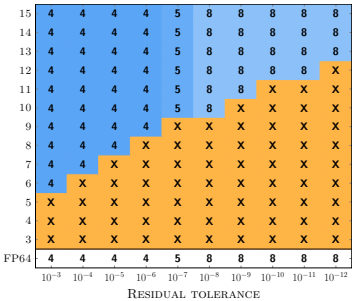
JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

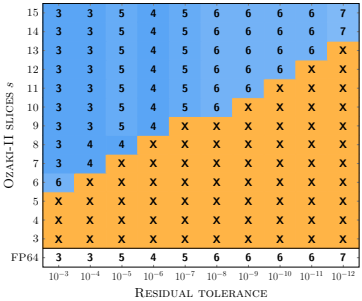
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1880

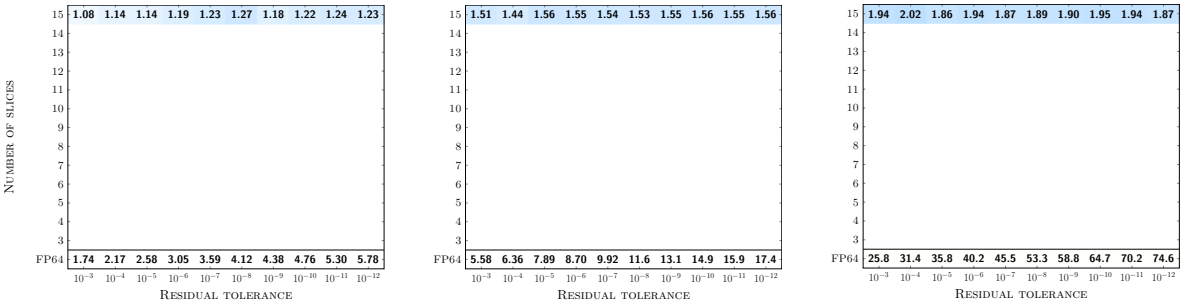


Large - N = 115k, NEV = 3463

Figure : *Iteration* vs. *Residual Tolerance*

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192

Medium - N = 62k, NEV = 1880

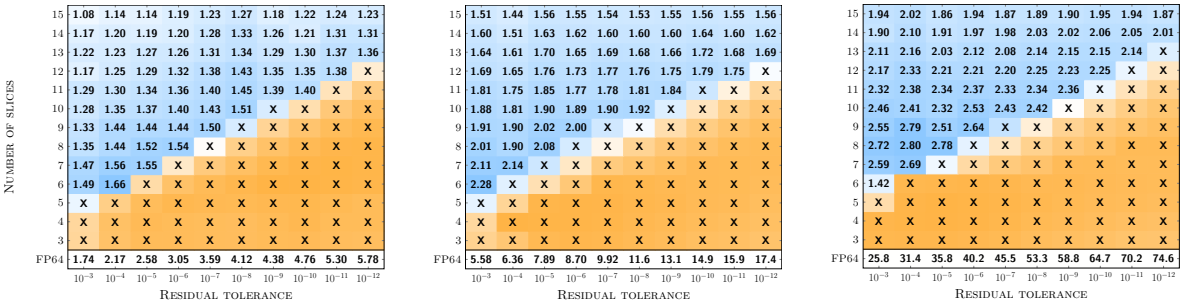
Large - N = 115k, NEV = 3463

Figure : *Speed-up* vs. *Residual Tolerance*

JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192

Medium - N = 62k, NEV = 1880

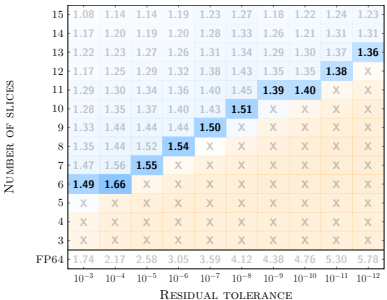
Large - N = 115k, NEV = 3463

Figure : *Speed-up* vs. *Residual Tolerance*

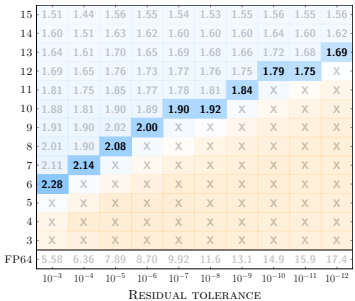
JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

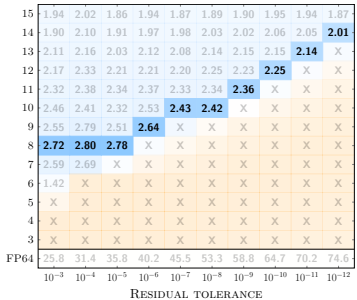
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1880



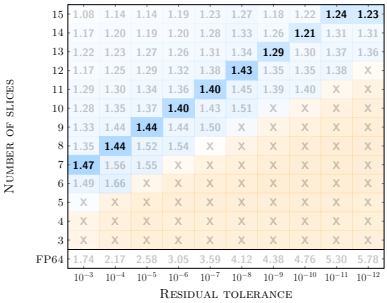
Large - N = 115k, NEV = 3463

Figure : *Speed-up* vs. *Residual Tolerance*

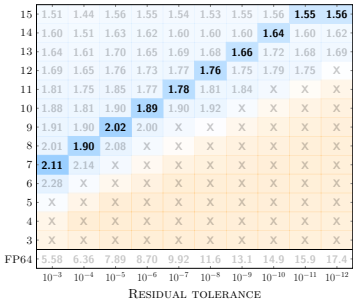
JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

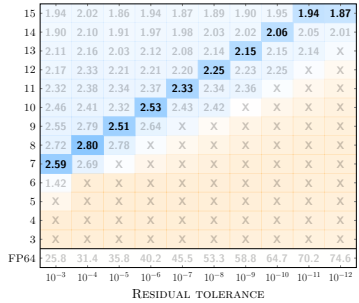
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1880



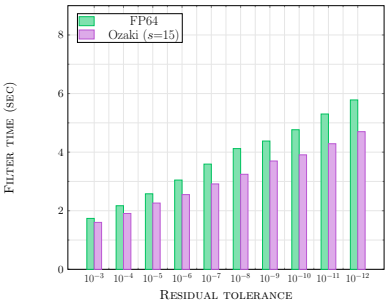
Large - N = 115k, NEV = 3463

Figure : *Speed-up* vs. *Residual Tolerance* – diag mode : $s = \max(15, 4 - \log(\mathrm{TOL}))$

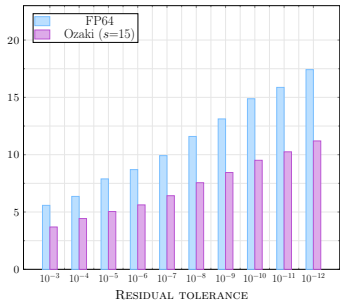
JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

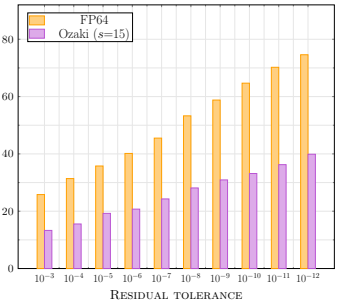
FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192



Medium - N = 62k, NEV = 1880



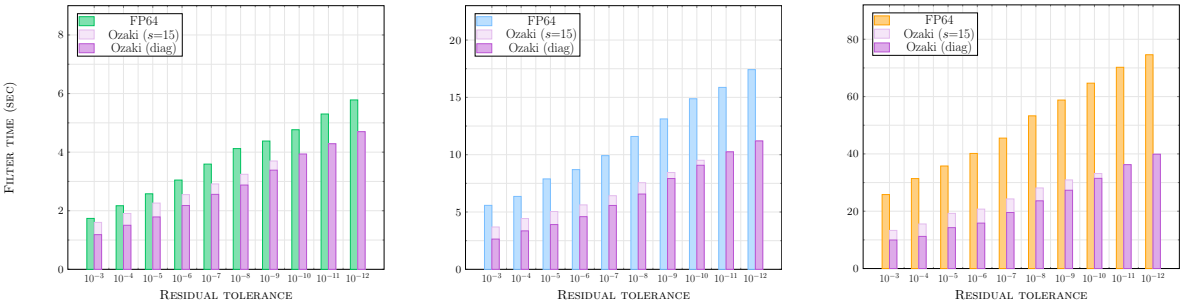
Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance* – diag mode : $s = \max(15, 4 - \log(\mathrm{TOL}))$

JÜLICH Forschungszentrum

III. OZAKI-II SCHEME INTO CHASE

FP64 Experiments with 16 GH200 GPUs on JUPITER – NEV = 3% N, NEX = NEV/2



Small - N = 39k, NEV = 1192 | Medium - N = 62k, NEV = 1880 | Large - N = 115k, NEV = 3463

Figure : *Filter Time* vs. *Residual Tolerance* – diag mode : $s = \max(15, 4 - \log(\mathrm{TOL}))$

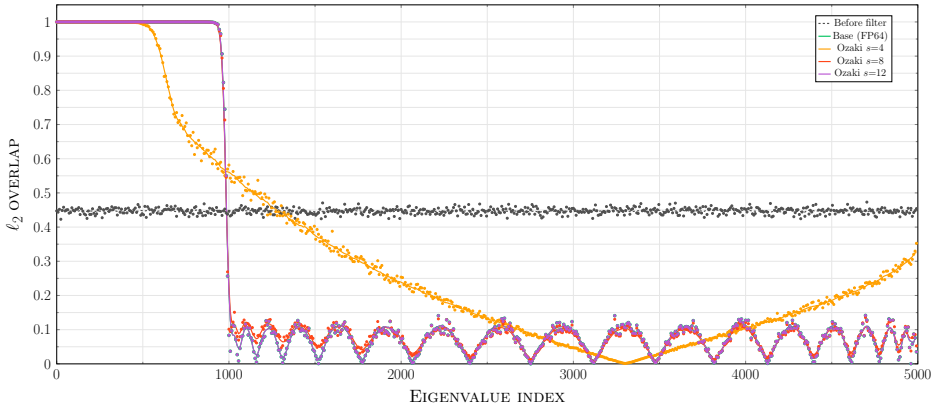
JÜLICH Forschungszentrum

Part II : Enabling tolerance-aware flexible accuracy

IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following

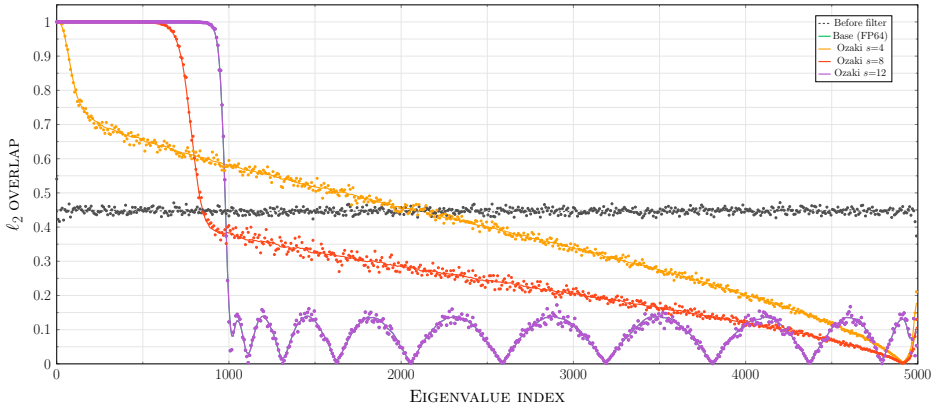
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad \text{DEGREE} = 20, \quad \phi = 0.0.$$



IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following

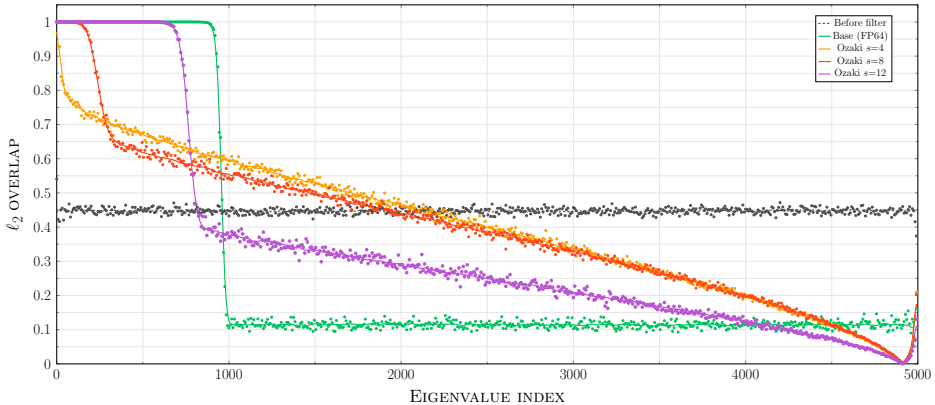
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad \text{DEGREE} = 20, \quad \phi = 1.5.$$



IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following

$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad \text{DEGREE} = 30, \quad \phi = 1.5.$$



IV. BUILDING A THEORETICAL FRAMEWORK

Instead of computing the polynomial itself, we leverage on the three-term recurrence relation

$$V_{i+1} = \alpha_i A V_i + \beta_i V_{i-1} \quad \text{with} \quad V_1 = A V_0 \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 OZAKI_GEMM('N','N',alpha,A,V0,0,V1,s)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   OZAKI_GEMM('N','N',alpha,A,V1,beta,V0,s)//V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```


JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

In practice, the three-terms recurrence relation relies on floating-point arithmetic, such that

$$\hat{V}_{i+1} = \alpha_i[A\hat{V}_i] + \beta_i \hat{V}_{i-1} \quad \text{with} \quad \hat{V}_1 = [AV_0] \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 OZAKI_GEMM('N','N',alpha,A,V0,0,V1,s)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   OZAKI_GEMM('N','N',alpha,A,V1,beta,V0,s)//V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```

Therefore, a local error $F_j := A\hat{V}_j - [A\hat{V}_j]$ is introduced at each recurrence step...

IV. BUILDING A THEORETICAL FRAMEWORK

In practice, the three-terms recurrence relation relies on floating-point arithmetic, such that

$$\hat{V}_{i+1} = \alpha_i[A\hat{V}_i] + \beta_i \hat{V}_{i-1} \quad \text{with} \quad \hat{V}_1 = [AV_0] \quad \text{and} \quad V_0 = \text{RANDOM}$$

Therefore, the implementation of the filter looks like :

```
1 V0 = random(N,nevex)
2 OZAKI_GEMM('N','N',alpha,A,V0,0,V1,s)  // V1 = alpha AV0
3 for d <= degree:
4   alpha, beta = GetCoefficients(iter,lambda,lowerb,upperb)
5   OZAKI_GEMM('N','N',alpha,A,V1,beta,V0,s)//V0 = alpha AV1 + beta V0
6   V0,V1 = V1,V0 //swap
```

Therefore, a local error $F_j := A\hat{V}_j - [A\hat{V}_j]$ is introduced at each recurrence step...

... and propagates into an accumulated error $E_{\text{deg}} := V_{\text{deg}} - \hat{V}_{\text{deg}}$.

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

The ChASE algorithm converged when

$$\|AV - V\tilde{\Lambda}\|_{1,2} < \mathcal{O}(\texttt{tol}),$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

The ChASE algorithm converged when

$$\|AV - V\tilde{\Lambda}\|_{1,2} < \mathcal{O}(\texttt{tol}),$$

Including the propagated error of the filter, it gives

$$\|A\hat{V} - \hat{V}\tilde{\Lambda}\|_{1,2} \quad = \quad \|A(V-E) - (V-E)\tilde{\Lambda}\|_{1,2} \quad \leq \quad \|AV - V\tilde{\Lambda}\|_{1,2} + \|AE - E\tilde{\Lambda}\|_{1,2}$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

The ChASE algorithm converged when

$$\|AV - V\tilde{\Lambda}\|_{1,2} < \mathcal{O}(\texttt{tol}),$$

Including the propagated error of the filter, it gives

$$\|A\hat{V} - \hat{V}\tilde{\Lambda}\|_{1,2} \quad = \quad \|A(V-E) - (V-E)\tilde{\Lambda}\|_{1,2} \quad \leq \quad \|AV - V\tilde{\Lambda}\|_{1,2} + \|AE - E\tilde{\Lambda}\|_{1,2}$$

Through Cauchy's interlace theorem, we have

$$\|AE - E\tilde{\Lambda}\|_{1,2} \quad \leq \quad \max_j \|A - \tilde{\lambda}_j I\|_2 \|E\|_{1,2} \quad \leq \quad |\lambda_n - \lambda_1| \|E\|_{1,2}.$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

The ChASE algorithm converged when

$$\|AV - V\tilde{\Lambda}\|_{1,2} < \mathcal{O}(\texttt{tol}),$$

Including the propagated error of the filter, it gives

$$\|A\hat{V} - \hat{V}\tilde{\Lambda}\|_{1,2} \quad = \quad \|A(V - E) - (V - E)\tilde{\Lambda}\|_{1,2} \quad \leq \quad \|AV - V\tilde{\Lambda}\|_{1,2} + \|AE - E\tilde{\Lambda}\|_{1,2}$$

Through Cauchy's interlace theorem, we have

$$\|AE - E\tilde{\Lambda}\|_{1,2} \quad \leq \quad \max_j \|A - \tilde{\lambda}_j I\|_2 \|E\|_{1,2} \quad \leq \quad |\lambda_n - \lambda_1| \|E\|_{1,2}.$$

A good criterion for convergence when applying a polynomial filter of degree m and s Ozaki-II moduli is

$$\|E_{\texttt{deg}}\|_{1,2} \cdot |\lambda_n - \lambda_1| \ < \ \mathcal{O}(\texttt{tol}).$$

IV. BUILDING A THEORETICAL FRAMEWORK

The ChASE algorithm converged when

$$\|AV - V\tilde{\Lambda}\|_{1,2} < \mathcal{O}(\texttt{tol}),$$

Including the propagated error of the filter, it gives

$$\|A\hat{V} - \hat{V}\tilde{\Lambda}\|_{1,2} \quad = \quad \|A(V - E) - (V - E)\tilde{\Lambda}\|_{1,2} \quad \leq \quad \|AV - V\tilde{\Lambda}\|_{1,2} + \|AE - E\tilde{\Lambda}\|_{1,2}$$

Through Cauchy's interlace theorem, we have

$$\|AE - E\tilde{\Lambda}\|_{1,2} \quad \leq \quad \max_j \|A - \tilde{\lambda}_j I\|_2 \|E\|_{1,2} \quad \leq \quad |\lambda_n - \lambda_1| \|E\|_{1,2}.$$

A good criterion for convergence when applying a polynomial filter of degree m and s Ozaki-II moduli is

$$\|E_{\texttt{deg}}\|_{1,2} \cdot |\lambda_n - \lambda_1| \ < \ \mathcal{O}(\texttt{tol}).$$

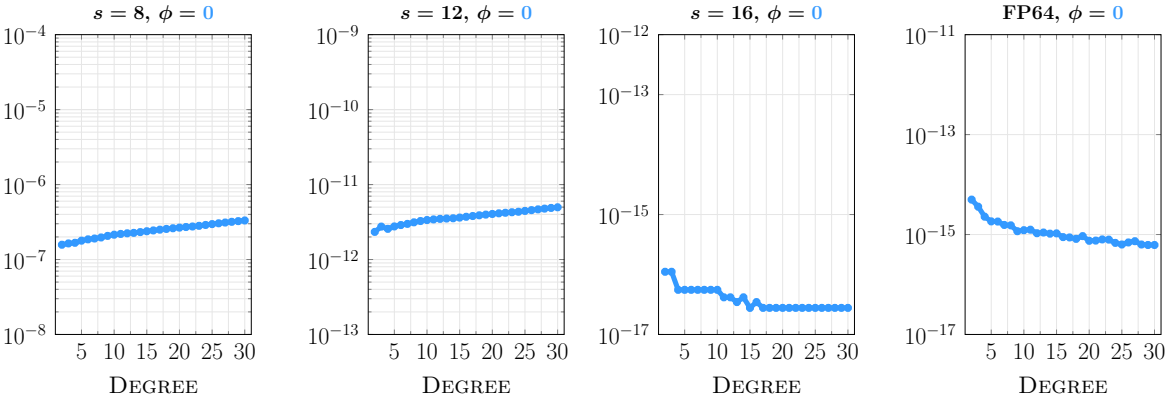
Hence, bounding the propagated error $E_{\texttt{deg}}$ to connect it to the prescribed tolerance `tol` is essential.

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

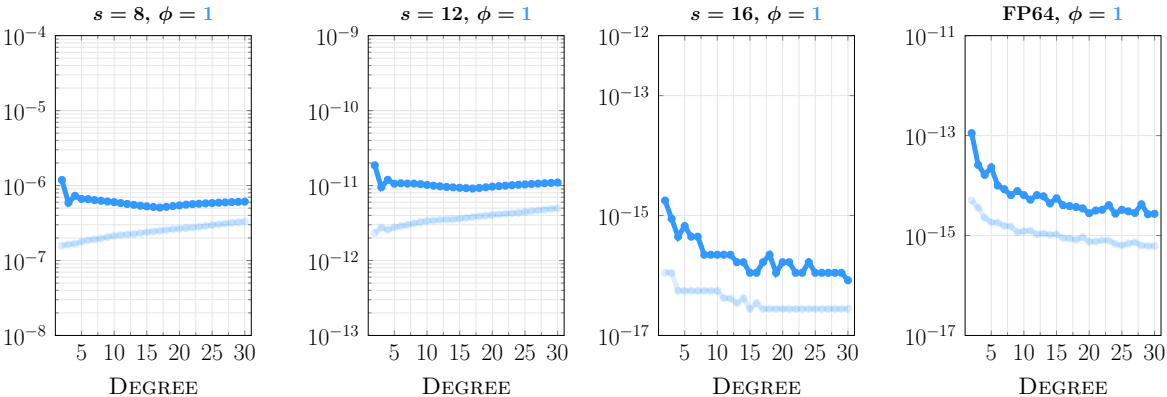
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad E_{\text{deg}} = \text{FP128}(V_{\text{deg}}) - \text{Oz-II}(V_{\text{deg}}).$$



IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

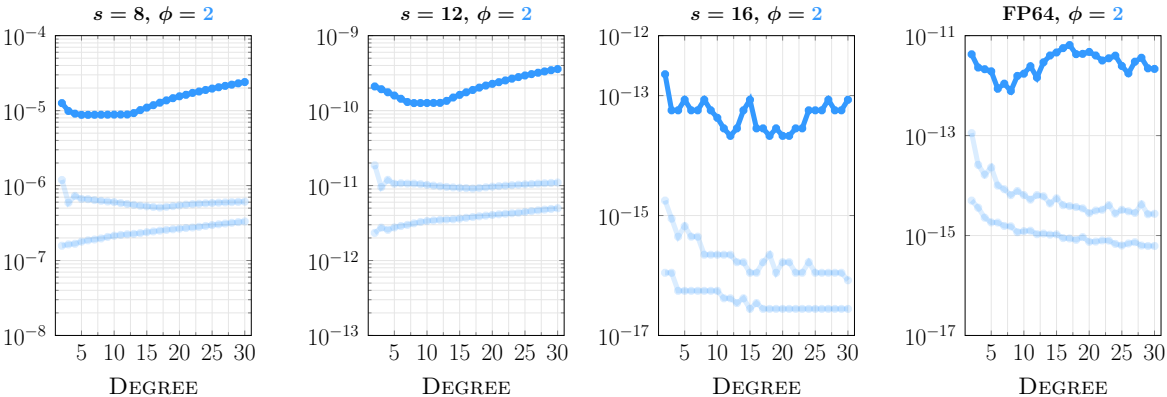
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad E_{\text{deg}} = \text{FP128}(V_{\text{deg}}) - \text{Oz-II}(V_{\text{deg}}).$$



IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad \text{with} \quad \alpha_{ij} \sim \mathcal{U}(0,1), \quad \beta_{ij} \sim \mathcal{N}(0,1) \quad | \quad E_{\text{deg}} = \text{FP128}(V_{\text{deg}}) - \text{Oz-II}(V_{\text{deg}}).$$



IV. BUILDING A THEORETICAL FRAMEWORK

In ChASE, the filtered-space is given by the three-terms recurrence relation

$$V_{i+1} = 2\sigma_{i+1}\bar{A}V_i - \sigma_{i+1}\sigma_i V_{i-1} \quad \text{with} \quad V_1 = \sigma_1 \bar{A} V_0 \quad \text{and} \quad V_0 = \text{RANDOM}(N, nev + nex)$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

In ChASE, the filtered-space is given by the three-terms recurrence relation

$$V_{i+1} = 2\sigma_{i+1}\bar{A}V_i - \sigma_{i+1}\sigma_i V_{i-1} \quad \text{with} \quad V_1 = \sigma_1 \bar{A} V_0 \quad \text{and} \quad V_0 = \text{RANDOM}(N, nev + nex)$$

In practice, each GEMM operation introduces a floating-point error $F_j := \bar{A}\hat{V}_j - [\bar{A}\hat{V}_j]$, such that

$$\hat{V}_{i+1} = 2\sigma_{i+1}[\bar{A}\hat{V}_i] - \sigma_{i+1}\sigma_i \hat{V}_{i-1} \quad \text{with} \quad \hat{V}_1 = \sigma_1[\bar{A}V_0].$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

In ChASE, the filtered-space is given by the three-terms recurrence relation

$$V_{i+1} = 2\sigma_{i+1}\bar{A}V_i - \sigma_{i+1}\sigma_i V_{i-1} \quad \text{with} \quad V_1 = \sigma_1 \bar{A} V_0 \quad \text{and} \quad V_0 = \text{RANDOM}(N, nev + nex)$$

In practice, each GEMM operation introduces a floating-point error $F_j := \bar{A}\hat{V}_j - [\bar{A}\hat{V}_j]$, such that

$$\hat{V}_{i+1} = 2\sigma_{i+1}[\bar{A}\hat{V}_i] - \sigma_{i+1}\sigma_i \hat{V}_{i-1} \quad \text{with} \quad \hat{V}_1 = \sigma_1[\bar{A}V_0].$$

This error propagates as the following non-homogeneous three-terms recurrence relation

$$E_{i+1} = 2\sigma_{i+1}\bar{A}E_i - \sigma_{i+1}\sigma_i E_{i-1} + 2\sigma_{i+1}F_i.$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

In ChASE, the filtered-space is given by the three-terms recurrence relation

$$V_{i+1} = 2\sigma_{i+1}\bar{A}V_i - \sigma_{i+1}\sigma_i V_{i-1} \quad \text{with} \quad V_1 = \sigma_1 \bar{A} V_0 \quad \text{and} \quad V_0 = \texttt{RANDOM}(N, nev + nex)$$

In practice, each GEMM operation introduces a floating-point error $F_j := \bar{A}\hat{V}_j - [\bar{A}\hat{V}_j]$, such that

$$\hat{V}_{i+1} = 2\sigma_{i+1}[\bar{A}\hat{V}_i] - \sigma_{i+1}\sigma_i \hat{V}_{i-1} \quad \text{with} \quad \hat{V}_1 = \sigma_1[\bar{A}V_0].$$

This error propagates as the following non-homogeneous three-terms recurrence relation

$$E_{i+1} = 2\sigma_{i+1}\bar{A}E_i - \sigma_{i+1}\sigma_i E_{i-1} + 2\sigma_{i+1}F_i.$$

Let U_j be a Chebyshev Polynomial of the 2ND kind and of degree j. At degree DEG, the propagated error is given by

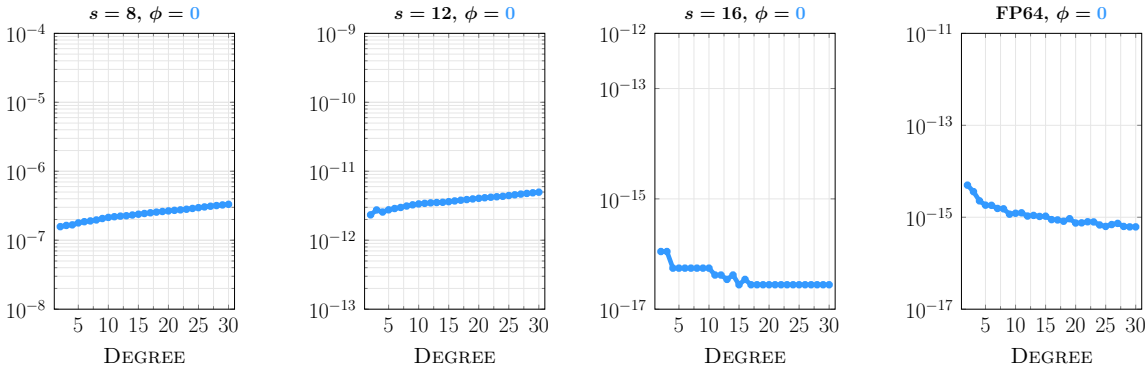
$$E_{\text{DEG}} = \sum_{j=0}^{\text{DEG}-1} w_j \frac{\eta_j}{\eta_{\text{DEG}}} U_{\text{DEG}-1-j}(\bar{A})\, F_j$$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

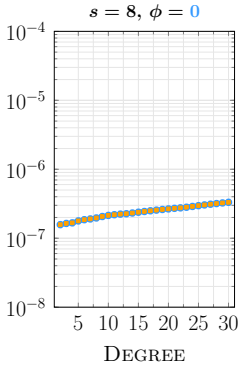
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad | \quad E_{\text{deg}} = \text{FP128}(V_{\text{deg}}) - \text{Oz-II}(V_{\text{deg}}).$$

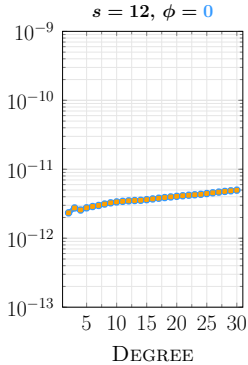


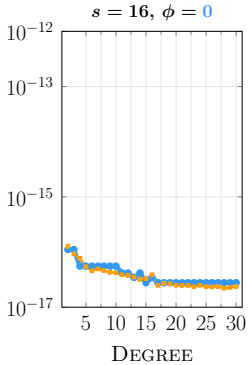
IV. BUILDING A THEORETICAL FRAMEWORK

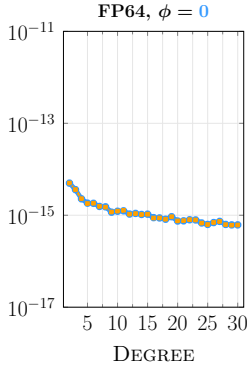
Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

$$(A_{ij}|\,V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad | \quad E_{\mathrm{DEG}} = \sum_{j=0}^{\mathrm{DEG}-1} w_j \frac{\eta_j}{\eta_{\mathrm{DEG}}} U_{\mathrm{DEG}-1-j}(\bar{A})\, F_j.$$





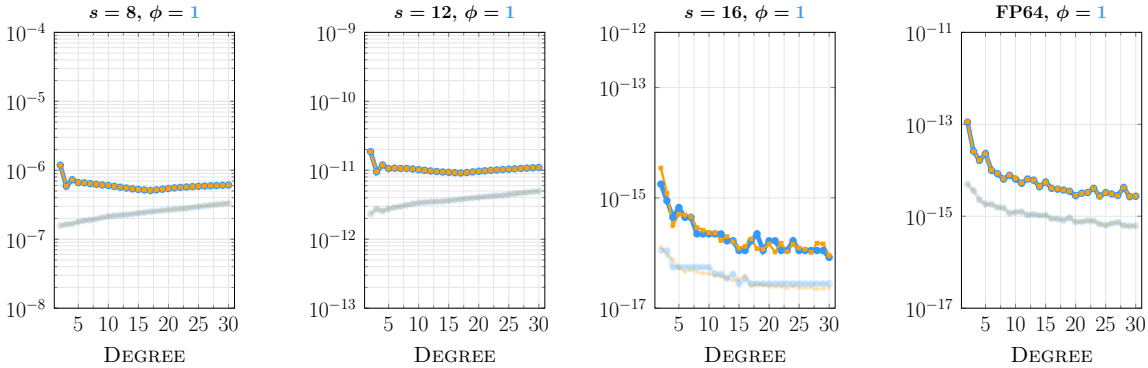




IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

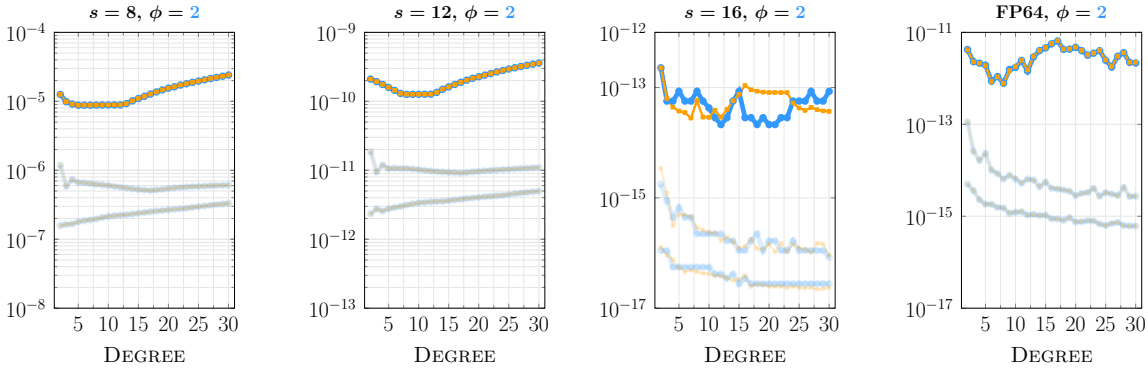
$$(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad | \quad E_{\text{DEG}} = \sum_{j=0}^{\text{DEG}-1} w_j \frac{\eta_j}{\eta_{\text{DEG}}} U_{\text{DEG}-1-j}(\bar{A})\, F_j.$$



IV. BUILDING A THEORETICAL FRAMEWORK

Let A be the input hermitian matrix and V the initial subspace, with $m = n = 10000$ and $nev + nex = 1000$, such that

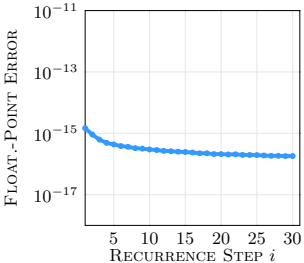
$$(A_{ij}|\,V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}} \quad | \quad E_{\text{DEG}} = \sum_{j=0}^{\text{DEG}-1} w_j \frac{\eta_j}{\eta_{\text{DEG}}} U_{\text{DEG}-1-j}(\bar{A})\, F_j.$$



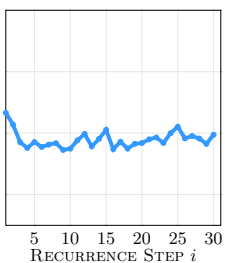
IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following $(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}}$, with $m = n = 4096$ and $nev + nex = 512$.

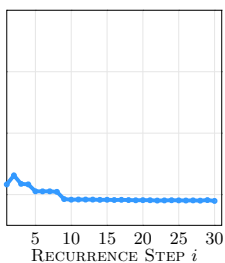
$$F_j := FP128(\bar{A}\hat{V}_j) - [\bar{A}\hat{V}_j].$$



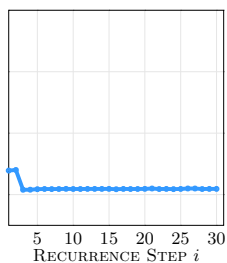
FP64 GEMM, $\phi = 0$



FP64 GEMM, $\phi = 2$



Ozaki-II $s = 16$, $\phi = 0$



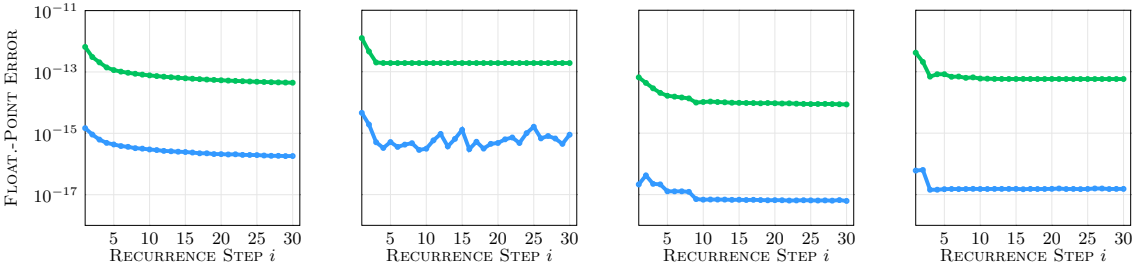
Ozaki-II $s = 16$, $\phi = 2$

JÜLICH Forschungszentrum

IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following $(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}}$, with $m = n = 4096$ and $nev + nex = 512$.

*FP*64 : $|F_j| \leq \frac{n\mu_{64}}{1 - n\mu_{64}} |\bar{A}||\hat{V}_j|$ *vs.* Ozaki-II (**[2]**): $|F_j| \leq t_s(|\bar{A}|\mathbb{1}(2^{v'})^T + 2^{a'}\mathbb{1}^T|\hat{V}_j|) + t_s^2(k + r_{b,s})2^{a'}(2^{v'})^T$.



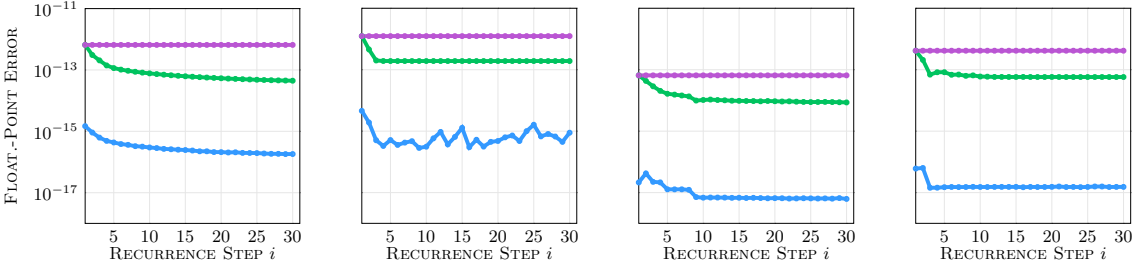
FP64 GEMM, $\phi = 0$ | FP64 GEMM, $\phi = 2$ | Ozaki-II $s = 16$, $\phi = 0$ | Ozaki-II $s = 16$, $\phi = 2$

[2] : *Error Analysis of Matrix Multiplication Emulation Using Ozaki-II Scheme, Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2026*

IV. BUILDING A THEORETICAL FRAMEWORK

Matrices are generated following $(A_{ij}|V_{ij}) = (\alpha_{ij} - 0.5) \times e^{\phi\beta_{ij}}$, with $m = n = 4096$ and $nev + nex = 512$.

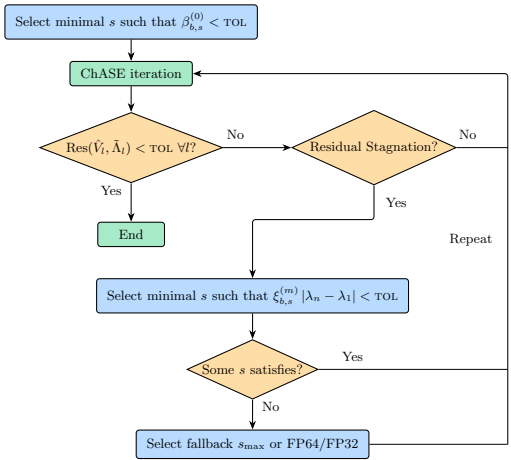
FP64 : $|F_j| \leq \dfrac{n\mu_{64}}{1 - n\mu_{64}} |\bar{A}||\hat{V}_0|$ *vs.* Ozaki-II (**[2]**): $|F_j| \leq t_s(|\bar{A}|\mathbb{1}(2^{v'})^T + 2^{a'}\mathbb{1}^T|\hat{V}_0|) + t_s^2(k + r_{b,s})2^{a'}(2^{v'})^T$.



FP64 GEMM, $\phi = 0$ | FP64 GEMM, $\phi = 2$ | Ozaki-II $s = 16$, $\phi = 0$ | Ozaki-II $s = 16$, $\phi = 2$

[2] : *Error Analysis of Matrix Multiplication Emulation Using Ozaki-II Scheme, Yuki Uchino, Katsuhisa Ozaki, Toshiyuki Imamura, 2026*

V. AUTOMATED CONTROL OF THE ACCURACY



Therefore we derive two bounds :

1 Loose condition : $F_j \leq \beta_{b,s}^{(0)} < \texttt{tol}$, with

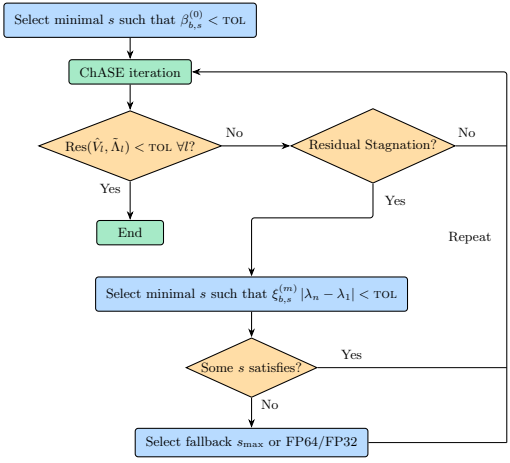
$$\begin{aligned} \beta_{b,s}^{(0)} &:= t_s \sqrt{n} (\|\bar{A}\|_\infty \cdot \max 2^{\nu'} + \|2^{a'}\|_2) \\ &+ t_s^2 (k + r_{b,s}) \cdot \|2^{a'}\|_2 \cdot \max 2^{\nu'}. \end{aligned}$$

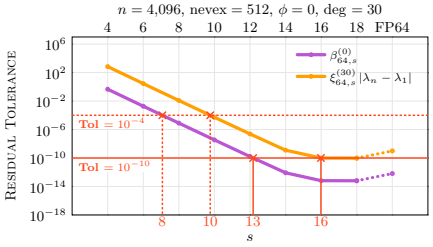
2 Conservative condition : $E_m \leq \xi_{b,s}^{(m)} |\lambda_n - \lambda_1| < \texttt{tol}$, with

$$\xi_{b,s}^{(m)} := \beta_{b,s}^{(0)} \cdot \sum_{j=0}^{m-1} \left| \omega_j \frac{\eta_j \nu_{m-j-1}}{\eta_m} \right|.$$

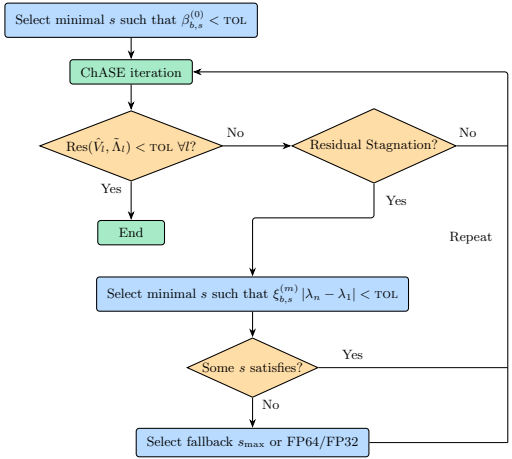
JÜLICH Forschungszentrum

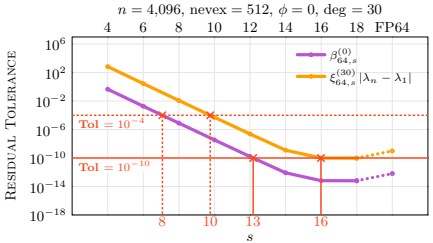
V. AUTOMATED CONTROL OF THE ACCURACY

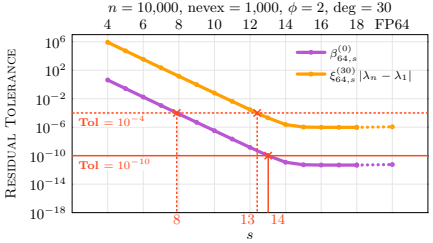




V. AUTOMATED CONTROL OF THE ACCURACY



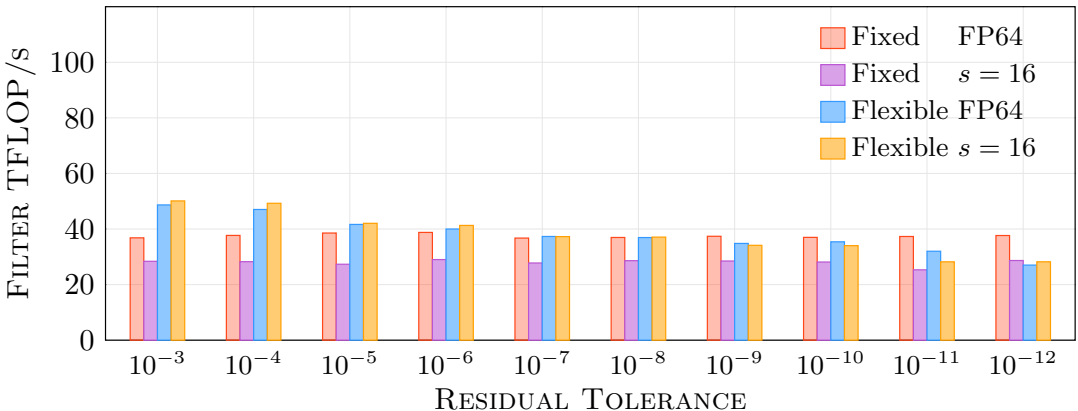




JÜLICH Forschungszentrum

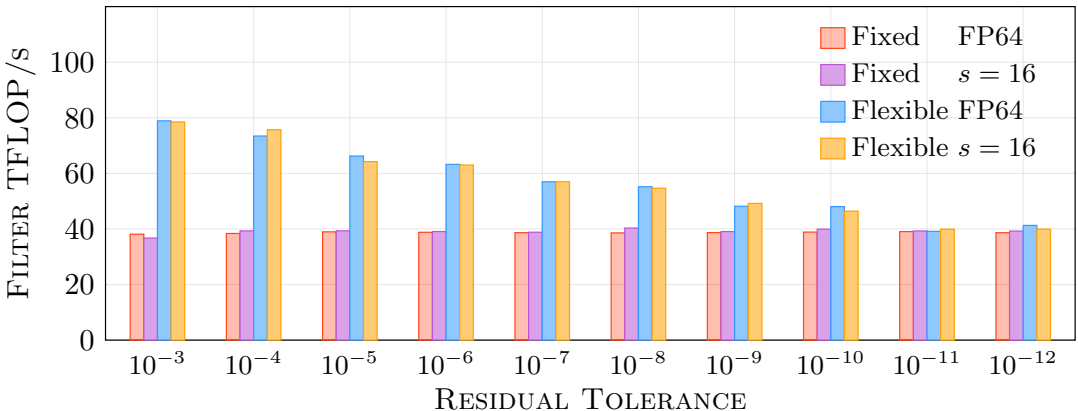
V. AUTOMATED CONTROL OF THE ACCURACY

Si vacancy: $n = 11{,}017$, $\texttt{nev} = \texttt{nex} = 550$.



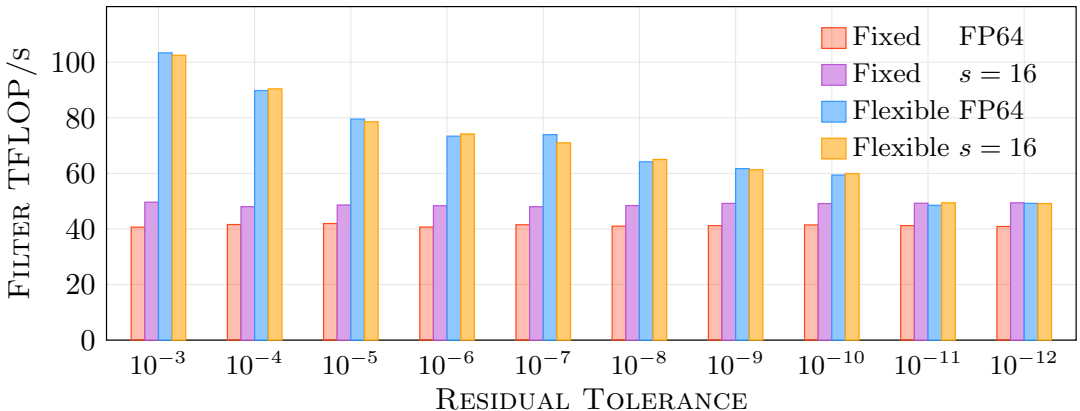
V. AUTOMATED CONTROL OF THE ACCURACY

Diamond: $n = 20{,}197$, $\texttt{nev} = \texttt{nex} = 1{,}009$.



V. AUTOMATED CONTROL OF THE ACCURACY

cBN: $n = 29{,}671$, $\texttt{nev} = \texttt{nex} = 1{,}483$.



CONCLUSION

Interconnecting Application Codes with NLA Libraries is essential for :

- Selecting a solver based on hardware configuration

CONCLUSION

Interconnecting Application Codes with NLA Libraries is essential for :

- Selecting a solver based on hardware configuration
- Abstracting data redistribution when necessary

CONCLUSION

Interconnecting Application Codes with NLA Libraries is essential for :

- Selecting a solver based on hardware configuration
- Abstracting data redistribution when necessary
- Connecting the application error tolerance with compute precision

CONCLUSION

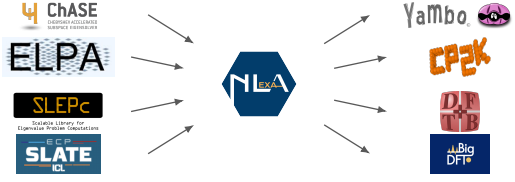
Interconnecting Application Codes with NLA Libraries is essential for :

- Selecting a solver based on hardware configuration
- Abstracting data redistribution when necessary
- Connecting the application error tolerance with compute precision

Building this bridge is one of the motivation of the ExaNLA initiative



First ExaNLA Gathering - Feb. 2025



APPENDIX

cBN material, $n = 29{,}671$. $\texttt{nev} = \texttt{nex} = 1{,}483$. Varying $\texttt{tol} \in [10^{-3}, 10^{-12}]$.

$\texttt{tol}$	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
Fixed-Accuracy Num. of Iterations	3	4	4	4	4	4	4	5	5	6
Flexible-Accuracy Num. of Iterations	3	4	4	4	4	5	5	5	5	6
Selected s for $\beta^{(0)}_{64,s} < \texttt{tol}$	8	9	10	11	11	12	13	14	16	16
Selected s for $\xi^{(m)}_{64,s} \lvert\lambda_n - \lambda_1\rvert < \texttt{tol}$	11	12	13	13	14	16*	16*	16*	16*	16*
Subspace Iteration index fallback			2			2	2	2	2	2

JÜLICH Forschungszentrum