MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

求解线性矩阵方程的混合精度算法

Xiaobo Liu (刘晓波)

德国马克斯·普朗克复杂技术系统动力学研究所

湘潭大学数韵学术报告

中国湖南 · 湘潭

2026 年 9 月 16 日

基于以下合作研究

Dmytryshyn, Fasi, Higham, & L. [SIAM J. Sci. Comput., 待刊]

Benner & L. [ArXiv:2510.02126 [math.NA], 2026 年 5 月修订]

报告提纲

1. 混合精度简介
2. Sylvester 矩阵方程
3. 低秩 Lyapunov 矩阵方程
4. 总结

浮点运算

典型的（规格化）浮点格式

1 位　　w 位　　$t-1$ 位

$$x = (-1)^s \cdot (1.f_1 \cdots f_{t-1})_2 \cdot 2^{E-(2^{w-1}-1)}, \quad 2^{w-1}-1 \text{ 为指数偏置}$$

表：四种浮点格式的参数。

格式	有效数位数 (t)	指数位数 (w)	数值范围	$u = 2^{-t}$
bf16	8	8	$10^{\pm 38}$	3.9×10^{-3}
fp16	11	5	$10^{\pm 5}$	4.9×10^{-4}
fp32	24	8	$10^{\pm 38}$	6.0×10^{-8}
fp64	53	11	$10^{\pm 308}$	1.1×10^{-16}

- 更低精度 $\rightsquigarrow$ 更快的浮点运算、更少的通信和更低的能耗。

浮点运算

典型的（规格化）浮点格式

1 位　　w 位　　$t-1$ 位

$$x = (-1)^s \cdot (1.f_1 \cdots f_{t-1})_2 \cdot 2^{E-(2^{w-1}-1)}, \quad 2^{w-1}-1 \text{ 为指数偏置}$$

表：四种浮点格式的参数。

格式	有效数位数 (t)	指数位数 (w)	数值范围	$u = 2^{-t}$
bf16	8	8	$10^{\pm 38}$	3.9×10^{-3}
fp16	11	5	$10^{\pm 5}$	4.9×10^{-4}
fp32	24	8	$10^{\pm 38}$	6.0×10^{-8}
fp64	53	11	$10^{\pm 308}$	1.1×10^{-16}

- 更低精度 $\rightsquigarrow$ 更快的浮点运算、更少的通信和更低的能耗。

浮点运算

典型的（规格化）浮点格式

1 位　　w 位　　$t-1$ 位

$$x = (-1)^s \cdot (1.f_1 \cdots f_{t-1})_2 \cdot 2^{E-(2^{w-1}-1)}, \quad 2^{w-1}-1 \text{ 为指数偏置}$$

表：四种浮点格式的参数。

格式	有效数位数 (t)	指数位数 (w)	数值范围	$u = 2^{-t}$
bf16	8	8	$10^{\pm 38}$	3.9×10^{-3}
fp16	11	5	$10^{\pm 5}$	4.9×10^{-4}
fp32	24	8	$10^{\pm 38}$	6.0×10^{-8}
fp64	53	11	$10^{\pm 308}$	1.1×10^{-16}

- 更低精度 $\rightsquigarrow$ 更快的浮点运算、更少的通信和更低的能耗。

低精度与混合精度的使用

何时使用低精度？

- 只需较低的计算精度
- 与其他来源的误差相平衡：
  模型降阶、近似（如低秩近似、离散化）
- 具有自我修正机制（如迭代改进）✓

⇝ 混合精度算法在计算的不同部分选用不同精度，在保证精度与稳定性的同时提高性能。

低精度与混合精度的使用

何时使用低精度？

- 只需较低的计算精度
- 与其他来源的误差相平衡：
  模型降阶、近似（如低秩近似、离散化）
- 具有自我修正机制（如迭代改进）✓

⇝ 混合精度算法在计算的不同部分选用不同精度，在保证精度与稳定性的同时提高性能。

低精度与混合精度的使用

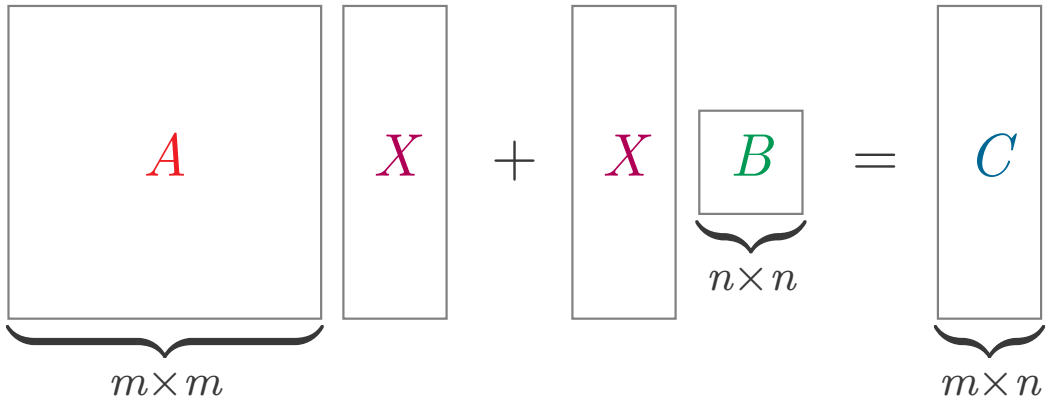
何时使用低精度？

- 只需较低的计算精度
- 与其他来源的误差相平衡：
  模型降阶、近似（如低秩近似、离散化）
- 具有自我修正机制（如迭代改进）✓

$\rightsquigarrow$ 混合精度算法在计算的不同部分选用不同精度，在保证精度与稳定性的同时提高性能。

报告提纲

Sylvester 矩阵方程



应用：分块对角化 • 矩阵函数扰动理论 • 系统平衡 • 控制 • 模型降阶等。

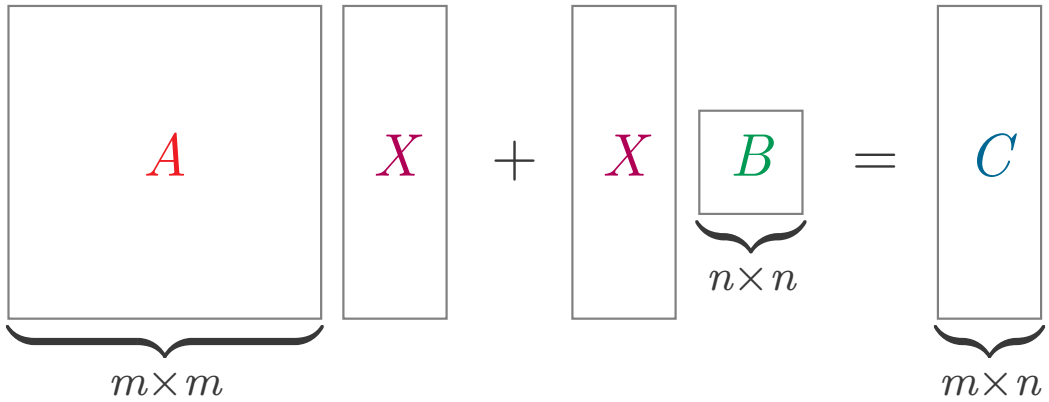
问题设置：

- 中等规模的稠密系数矩阵 $\rightsquigarrow$ 例如，$\max(m, n) \lesssim 10^4$
- 不要求具有特定结构

$\rightsquigarrow$ 首选方法：Bartels–Stewart 算法 (**Bartels & Stewart, '72**).

目标：结合两种精度求解 X：高精度（工作精度）+ 低精度

Sylvester 矩阵方程



应用：分块对角化 • 矩阵函数扰动理论 • 系统平衡 • 控制 • 模型降阶等。

问题设置：

- 中等规模的稠密系数矩阵 $\rightsquigarrow$ 例如，$\max(m, n) \lesssim 10^4$
- 不要求具有特定结构

$\rightsquigarrow$ 首选方法：Bartels–Stewart 算法 (**Bartels & Stewart, '72**).

目标：结合两种精度求解 X：高精度（工作精度）+ 低精度

Bartels–Stewart 算法

$$A\,X + X\,B = C$$

1. 计算 Schur 分解：

$$A = U_A\,T_A\,U_A^*, \qquad B = U_B\,T_B\,U_B^*$$

2. 通过代入法求解

$$T_A\,Y + Y\,T_B = \widetilde{C}, \quad \widetilde{C} = U_A^* C U_B$$

3. 返回 $X := U_A\,Y\,U_B^*$

Bartels–Stewart 算法

$$AX + XB = C$$

1. 计算 Schur 分解：

$$A = U_A T_A U_A^*, \quad B = U_B T_B U_B^*$$

2. 通过代入法求解

$$T_A Y + Y T_B = \widetilde{C}, \quad \widetilde{C} = U_A^* C U_B$$

3. 返回 $X := U_A Y U_B^*$

Bartels–Stewart 算法

$$AX + XB = C$$

1. 计算 Schur 分解：

$$A = U_A T_A U_A^*, \quad B = U_B T_B U_B^*$$

2. 通过代入法求解

$$T_A Y + Y T_B = \widetilde{C}, \quad \widetilde{C} = U_A^* C U_B$$

3. 返回 $X := U_A Y U_B^*$

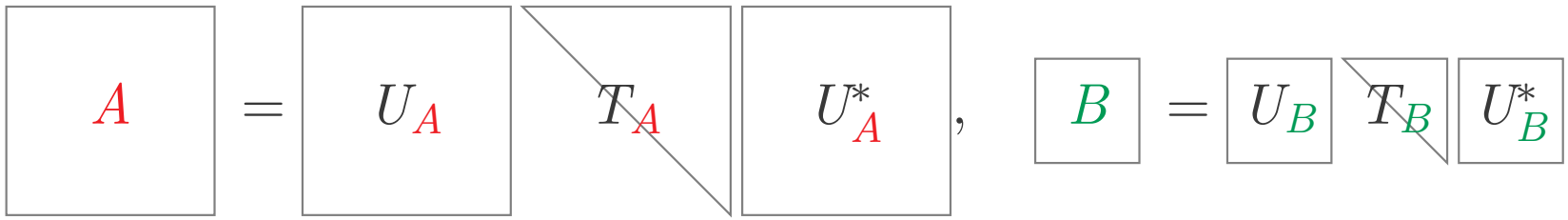
Bartels–Stewart 算法

$$A\,X + X\,B = C$$

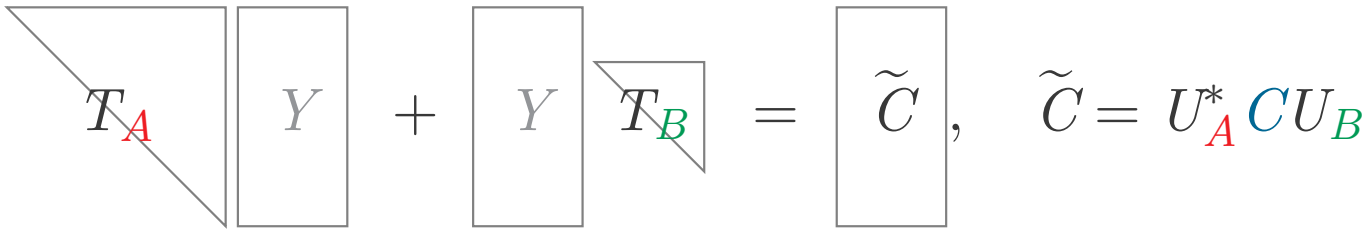
$$\Updownarrow$$

$$U_A^*\,U_A\,T_A\,U_A^*\,X\,U_B + U_A^*\,X\,U_B\,T_B\,U_B^*\,U_B = U_A^*\,C\,U_B$$

1. 计算 Schur 分解：



2. 通过代入法求解



3. 返回 $X := U_A\,Y\,U_B^*$

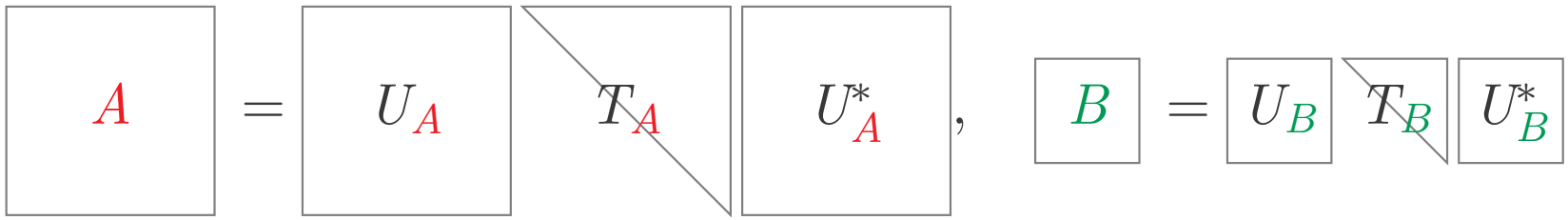
Bartels–Stewart 算法

$$A\,X + X\,B = C$$

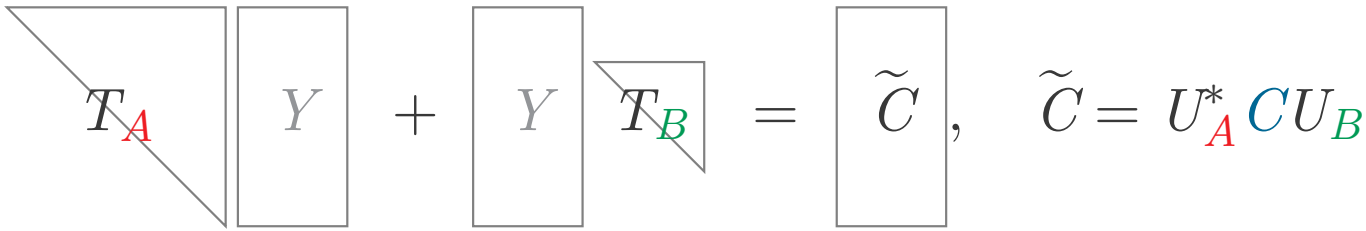
$$\Updownarrow$$

$$U_A^* U_A\, T_A\, U_A^* X U_B + U_A^* X U_B\, T_B\, U_B^* U_B = U_A^*\, C\, U_B$$

1. 计算 Schur 分解：



2. 通过代入法求解



3. 返回 $X := U_A\, Y\, U_B^*$

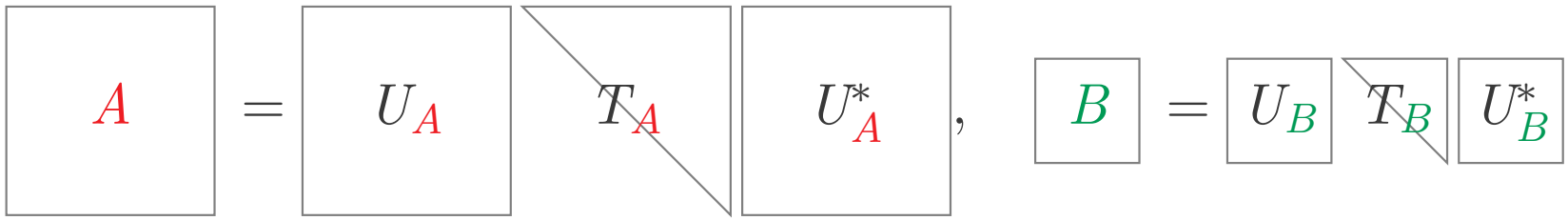
Bartels–Stewart 算法

$$A\,X + X\,B = C$$

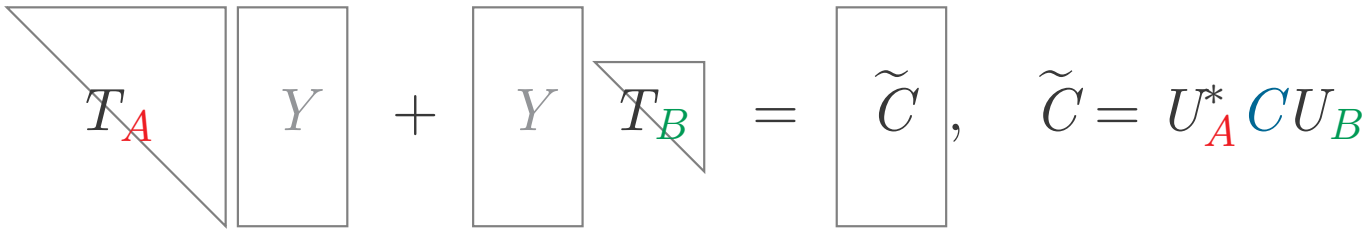
$$\Updownarrow$$

$$U_A^* U_A\, T_A\, U_A^* X U_B + U_A^* X U_B\, T_B\, U_B^* U_B = U_A^*\, C\, U_B$$

1. 计算 Schur 分解：



2. 通过代入法求解



3. 返回 $X := U_A\, Y\, U_B^*$

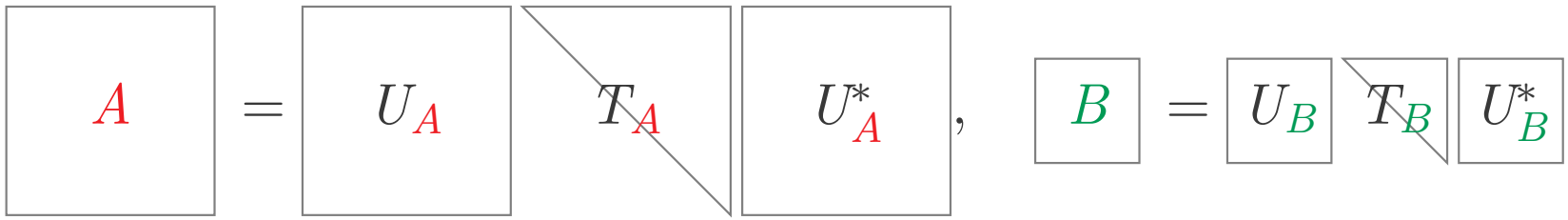
Bartels–Stewart 算法

$$A\,X + X\,B = C$$

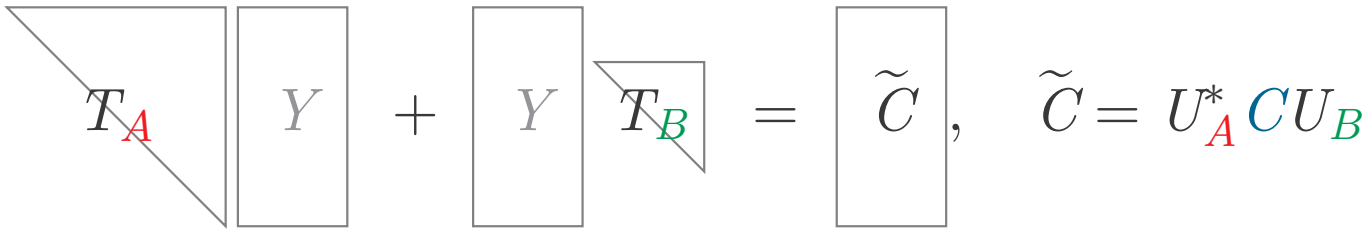
$$\Updownarrow$$

$$T_A\,Y + Y\,T_B = U_A^*\,C\,U_B, \quad Y = U_A^*\,X\,U_B$$

1. 计算 Schur 分解：



2. 通过代入法求解



3. 返回 $X := U_A\,Y\,U_B^*$

构建混合精度算法

1. 计算 Schur 分解 $A =: U_A T_A U_A^*$ ▷ $25m^3$
2. 计算 Schur 分解 $B =: U_B T_B U_B^*$ ▷ $25n^3$
3. 计算 $\widetilde{C} := U_A^* C U_B$ ▷ $2mn(m+n)$
4. 求解 $T_A Y + Y T_B = \widetilde{C}$，得到 Y ▷ $mn(m+n)$
5. 返回 $X := U_A Y U_B^*$ ▷ $2mn(m+n)$

观察

- 步骤 1 和 2 采用低精度
- 使用低精度 Schur 因子 $\rightsquigarrow$ 需要修改步骤 3–5

$\rightsquigarrow$ 我们的思路

1. 已有低精度三角因子 $\rightsquigarrow$ 用迭代改进（IR）求解与步骤 4 中方程“邻近”的方程：其系数矩阵为低精度三角因子加上小扰动。
2. 对低精度酉因子重新正交归一化或显式求逆

构建混合精度算法

1. 计算 Schur 分解 $A =: U_A T_A U_A^*$ ▷ $25m^3$
2. 计算 Schur 分解 $B =: U_B T_B U_B^*$ ▷ $25n^3$
3. 计算 $\widetilde{C} := U_A^* C U_B$ ▷ $2mn(m+n)$
4. 求解 $T_A Y + Y T_B = \widetilde{C}$，得到 Y ▷ $mn(m+n)$
5. 返回 $X := U_A Y U_B^*$ ▷ $2mn(m+n)$

观察

- 步骤 1 和 2 采用低精度
- 使用低精度 Schur 因子 $\rightsquigarrow$ 需要修改步骤 3–5

$\rightsquigarrow$ 我们的思路

1. 已有低精度三角因子 $\rightsquigarrow$ 用迭代改进（IR）求解与步骤 4 中方程“邻近”的方程：其系数矩阵为低精度三角因子加上小扰动。
2. 对低精度酉因子重新正交归一化或显式求逆

构建混合精度算法

1. 计算 Schur 分解 $A =: U_A T_A U_A^*$ ▷ $25m^3$
2. 计算 Schur 分解 $B =: U_B T_B U_B^*$ ▷ $25n^3$
3. 计算 $\widetilde{C} := U_A^* C U_B$ ▷ $2mn(m+n)$
4. 求解 $T_A Y + Y T_B = \widetilde{C}$，得到 Y ▷ $mn(m+n)$
5. 返回 $X := U_A Y U_B^*$ ▷ $2mn(m+n)$

观察

- 步骤 1 和 2 采用低精度
- 使用低精度 Schur 因子 $\rightsquigarrow$ 需要修改步骤 3–5

$\rightsquigarrow$ 我们的思路

1. 已有低精度三角因子 $\rightsquigarrow$ 用迭代改进（IR）求解与步骤 4 中方程“邻近”的方程：其系数矩阵为低精度三角因子加上小扰动。
2. 对低精度酉因子重新正交归一化或显式求逆

扰动三角方程的迭代改进

目标：求解 $\underbrace{(T_A+\Delta_A)}_{T_A\text{ 低精度}}Y+Y\underbrace{(T_B+\Delta_B)}_{T_B\text{ 低精度}}=\underbrace{U_A^*CU_B}_{U_A,\,U_B\text{ 高精度下保持酉性}}=:D$

---

1 $k\leftarrow 0$

2 **当** $\|\Delta_Y\|>\varepsilon$ **执行**

3 $\quad R_k\leftarrow D-(T_A+\Delta_A)Y_k+Y_k(T_B+\Delta_B)$

4 $\quad$ 用 Bartels–Stewart 算法求解 $T_A\Delta_Y+\Delta_Y T_B=R_k$，得到 Δ_Y

5 $\quad Y_{k+1}\leftarrow Y_k+\Delta_Y$

6 $\quad k\leftarrow k+1$

7 **结束**

8 **返回** Y_k

计算成本：$3kmn(m+n)$ 次浮点运算

- **收敛的充分条件**：$\|\Delta_A\|_2+\|\Delta_B\|_2<\mathrm{sep}_F(T_A,-T_B)$

扰动三角方程的迭代改进

目标：求解 $\underbrace{(T_A + \Delta_A)}_{T_A \text{ 低精度}} Y + Y \underbrace{(T_B + \Delta_B)}_{T_B \text{ 低精度}} = \underbrace{U_A^* C U_B}_{U_A,\, U_B \text{ 高精度下保持酉性}} =: D$

---

1 $k \leftarrow 0$

2 **当** $\|\Delta_Y\| > \varepsilon$ **执行**

3 $\quad R_k \leftarrow D - (T_A + \Delta_A) Y_k + Y_k (T_B + \Delta_B)$

4 $\quad$ 用 Bartels–Stewart 算法求解 $T_A \Delta_Y + \Delta_Y T_B = R_k$，得到 Δ_Y

5 $\quad Y_{k+1} \leftarrow Y_k + \Delta_Y$

6 $\quad k \leftarrow k + 1$

7 **结束**

8 **返回** Y_k

计算成本：$3kmn(m+n)$ 次浮点运算

- 收敛的充分条件：$\|\Delta_A\|_2 + \|\Delta_B\|_2 < \operatorname{sep}_F(T_A, -T_B)$

扰动三角方程的迭代改进

目标：求解 $\underbrace{(T_A + \Delta_A)}_{T_A \text{ 低精度}} Y + Y \underbrace{(T_B + \Delta_B)}_{T_B \text{ 低精度}} = \underbrace{U_A^* C U_B}_{U_A,\, U_B \text{ 高精度下保持酉性}} =: D$

---

1 $k \leftarrow 0$

2 **当** $\|\Delta_Y\| > \varepsilon$ **执行**

3 $\quad R_k \leftarrow D - (T_A + \Delta_A) Y_k + Y_k (T_B + \Delta_B)$

4 $\quad$ 用 Bartels–Stewart 算法求解 $T_A \Delta_Y + \Delta_Y T_B = R_k$，得到 Δ_Y

5 $\quad Y_{k+1} \leftarrow Y_k + \Delta_Y$

6 $\quad k \leftarrow k + 1$

7 **结束**

8 **返回** Y_k

计算成本：$3kmn(m+n)$ 次浮点运算

- **收敛的充分条件**：$\|\Delta_A\|_2 + \|\Delta_B\|_2 < \operatorname{sep}_F(T_A, -T_B)$

混合精度算法

对低精度酉因子重新正交归一化

1 计算 Schur 分解 $A = U_A T_A U_A^*$ ▷ 采用低精度 u_ℓ

2 计算 Schur 分解 $B = U_B T_B U_B^*$

3 计算 QR 分解 $U_A = Q_A R_A$, $U_B = Q_B R_B$ ▷ 重新正交归一化

4 $\Delta_A \leftarrow Q_A^* A Q_A - T_A$

5 $\Delta_B \leftarrow Q_B^* B Q_B - T_B$

6 $D \leftarrow Q_A^* C Q_B$

7 求解 $T_A Y + Y T_B = D$，得到 Y

8 **当** $\|\Delta_Y\| > \varepsilon$ **执行**

9 $\quad R \leftarrow D - (T_A + \Delta_A) Y + Y(T_B + \Delta_B)$

10 $\quad$ 用 Bartels–Stewart 算法求解 $T_A \Delta_Y + \Delta_Y T_B = R$，得到 Δ_Y

11 $\quad Y \leftarrow Y + \Delta_Y$

12 **结束**

13 $X \leftarrow Q_A Y Q_B^*$

混合精度算法

对低精度酉因子求逆

1 计算 Schur 分解 $A = U_A T_A U_A^*$ ▷ 采用低精度 u_ℓ

2 计算 Schur 分解 $B = U_B T_B U_B^*$

3 计算 U_A^* 和 U_B 的 LU 分解

4 $\Delta_A \leftarrow U_A^* A U_A^{-*} - T_A$

5 $\Delta_B \leftarrow U_B^{-1} B U_B - T_B$

6 $D \leftarrow U_A^* C U_B$

7 求解 $T_A Y + Y T_B = D$，得到 Y

8 **当** $\|\Delta_Y\| > \varepsilon$ **执行**

9 $\quad R \leftarrow D - (T_A + \Delta_A) Y + Y(T_B + \Delta_B)$

10 $\quad$ 用 Bartels–Stewart 算法求解 $T_A \Delta_Y + \Delta_Y T_B = R$，得到 Δ_Y

11 $\quad Y \leftarrow Y + \Delta_Y$

12 **结束**

13 $X \leftarrow U_A^{-*} Y U_B^{-1}$

何时值得使用低精度？

混合精度算法的计算成本

- 重新正交归一化
  - $25(m^3+n^3)+mn(m+n)$ 低精度浮点运算
  - $6(m^3+n^3)+(4+3k)mn(m+n)$ 高精度浮点运算
- 求逆
  - $25(m^3+n^3)+mn(m+n)$ 低精度浮点运算
  - $4\frac{2}{3}(m^3+n^3)+(4+3k)mn(m+n)$ 高精度浮点运算

Bartels–Stewart 算法的计算成本

- $25(m^3+n^3)+5mn(m+n)$ 高精度浮点运算

这些计算成本应如何比较？

何时值得使用低精度？

混合精度算法的计算成本

- 重新正交归一化
  - $25(m^3+n^3)+mn(m+n)$ 低精度浮点运算
  - $6(m^3+n^3)+(4+3k)mn(m+n)$ 高精度浮点运算
- 求逆
  - $25(m^3+n^3)+mn(m+n)$ 低精度浮点运算
  - $4\frac{2}{3}(m^3+n^3)+(4+3k)mn(m+n)$ 高精度浮点运算

Bartels–Stewart 算法的计算成本

- $25(m^3+n^3)+5mn(m+n)$ 高精度浮点运算

这些计算成本应如何比较？

混合精度的计算成本

计算成本模型

$$\rho = \frac{\text{低精度单次浮点运算平均时间}}{\text{高精度单次浮点运算平均时间}}$$

$\rho = 0$ 低精度浮点运算的成本可忽略

$\rho = 1$ 低、高精度浮点运算的成本相同

$\rho > 1$ 模拟的低精度运算

$\boxed{0 \leq \rho \ll 1}$ 在新型硬件（如 GPU）上成立

混合精度的计算成本

计算成本模型

$$\rho = \frac{\text{低精度单次浮点运算平均时间}}{\text{高精度单次浮点运算平均时间}}$$

$\rho = 0$ 低精度浮点运算的成本可忽略

$\rho = 1$ 低、高精度浮点运算的成本相同

$\rho > 1$ 模拟的低精度运算

$0 \leq \rho \ll 1$ 在新型硬件（如 GPU）上成立

混合精度的计算成本

计算成本模型

$$\rho = \frac{\text{低精度单次浮点运算平均时间}}{\text{高精度单次浮点运算平均时间}}$$

$\rho = 0$　低精度浮点运算的成本可忽略

$\rho = 1$　低、高精度浮点运算的成本相同

$\rho > 1$　模拟的低精度运算

$\boxed{0 \leq \rho \ll 1}$ 在新型硬件（如 GPU）上成立

混合精度的计算成本

归一化计算成本（以高精度运算为单位）

$$C_N = C_h + \rho \cdot C_\ell$$

- C_h: 高精度浮点运算次数
- C_ℓ: 低精度浮点运算次数

C_ℓ 中的每次浮点运算均按 ρ 加权。

可承受多少次迭代？

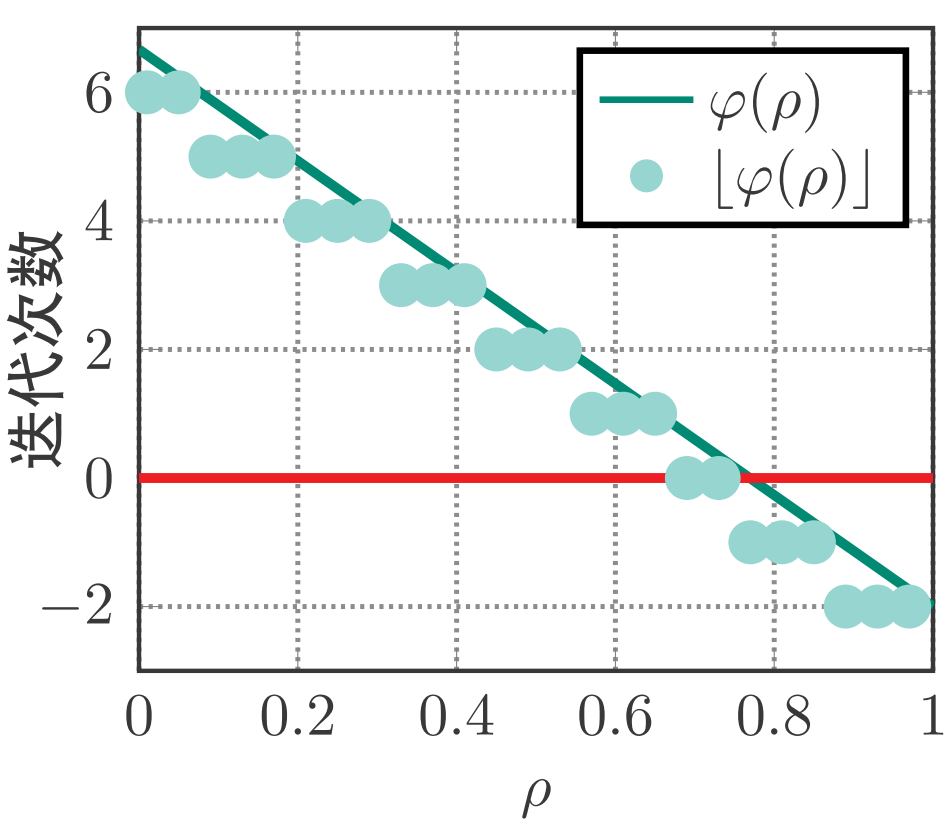
最大迭代次数

$$\lfloor \varphi(\rho) \rfloor, \qquad \varphi(\rho) = \frac{C_{BS}(m,n) - C_N^{ni}(m,n)}{C_N^{it}(m,n)}$$

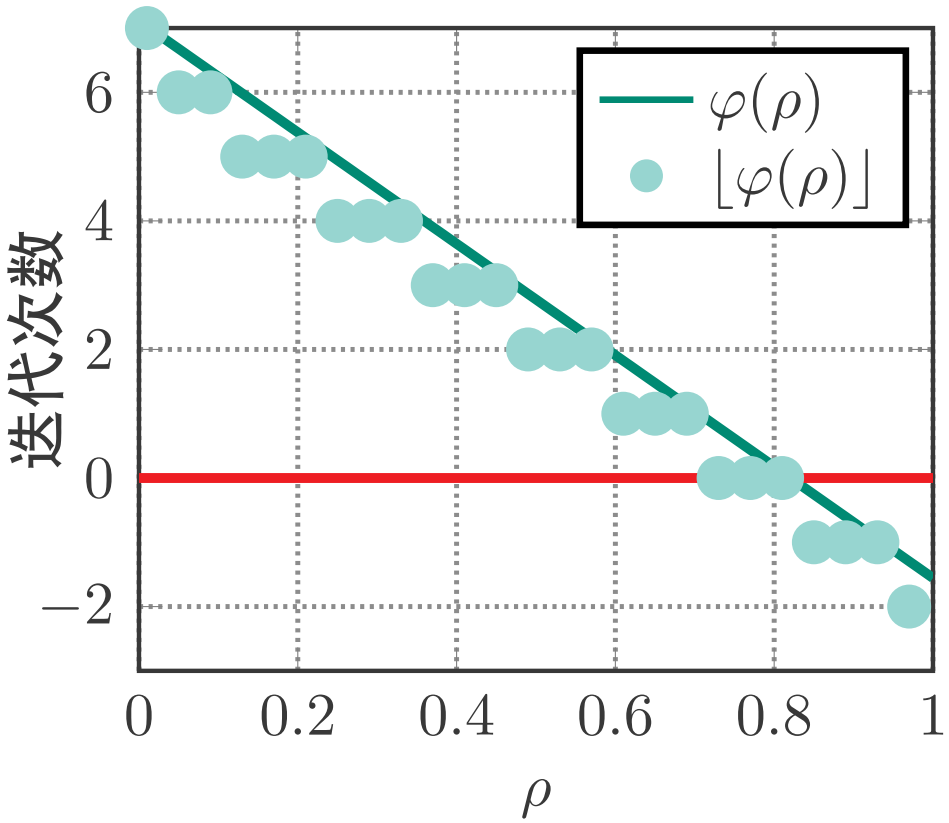
其中

- $C_{BS}(m,n)$: 高精度 Bartels–Stewart 算法的成本
- $C_N^{ni}(m,n)$: 新算法非迭代部分的归一化成本
- $C_N^{it}(m,n)$: 改进过程一次迭代的归一化成本

当 $m = n$ 时，至多迭代 6 次



(a) 基于重新正交归一化。



(b) 基于求逆。

- $\rho =$ 低精度单次浮点运算平均时间 / 高精度单次浮点运算平均时间

实验设置

算法

- `sylv`: MATLAB 内置函数 `sylvester`
- `mp_orth`: **重新正交归一化算法**的 MATLAB 实现
- `mp_inv`: **求逆算法**的 MATLAB 实现

迭代改进容差: 1e-12$\cdot \max(m, n)$

测试集

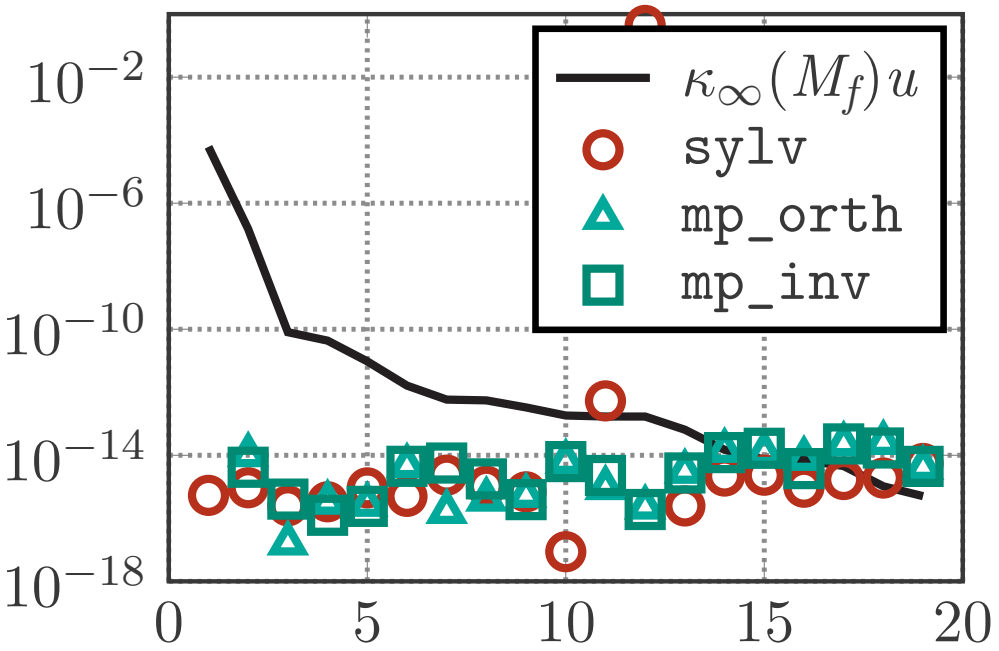
- 文献中的 19 个 Sylvester 方程，规模为 31–1,668

精度

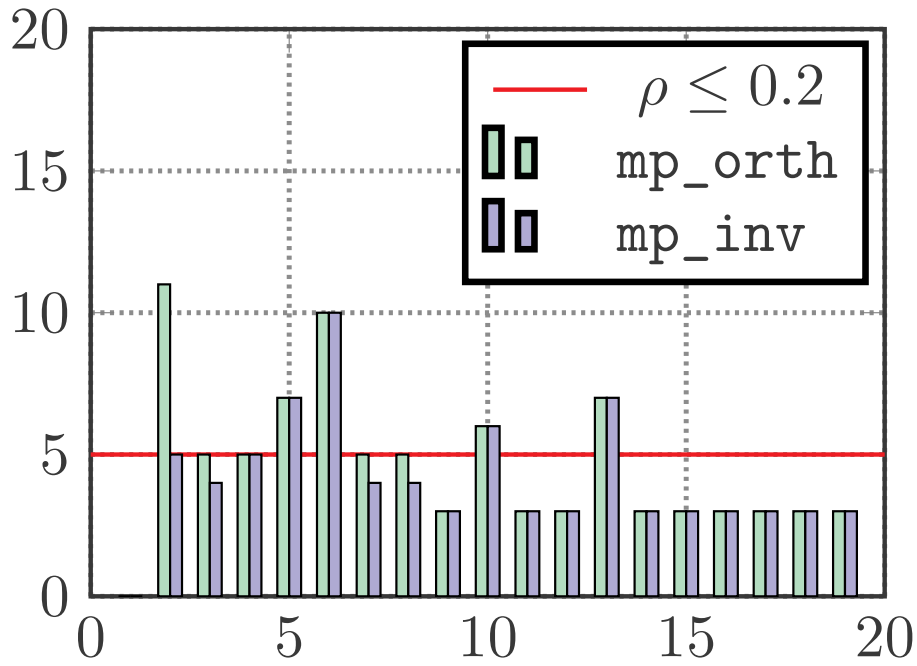
- TensorFloat-32：由 CPFloat **模拟** **(Fasi & Mikaitis, '23)**
- 自定义 24 位格式：有效数为 16 位，数值范围与 TensorFloat-32 相同
- binary64（高精度）

相对残差与迭代次数

低精度 = TensorFloat-32



(c) 相对残差。

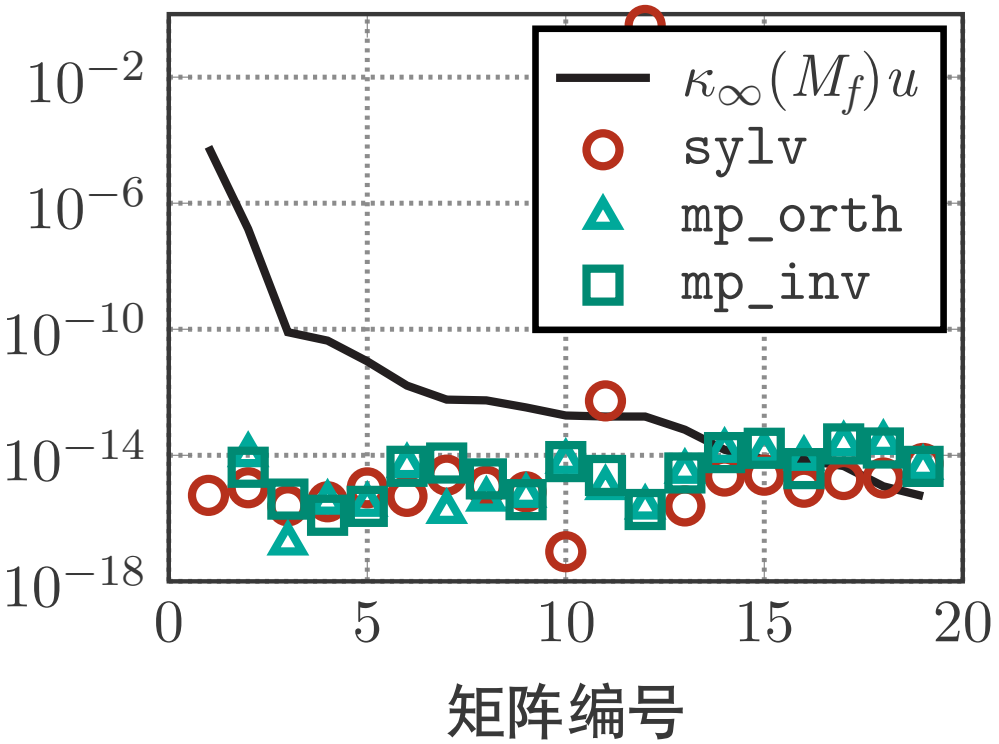


(d) 迭代次数。

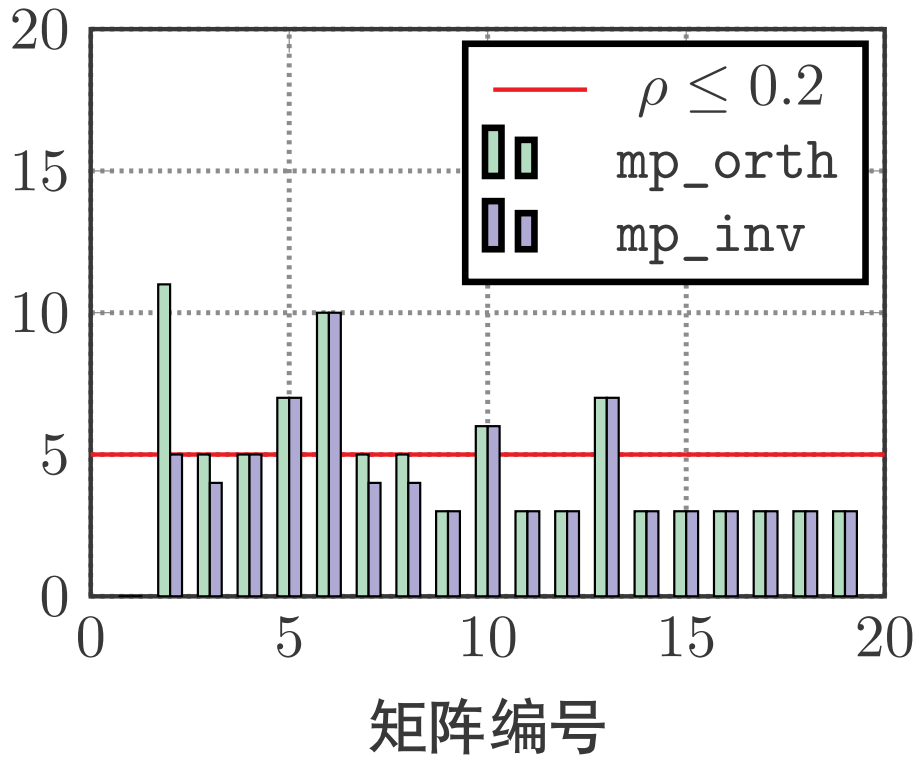
- 采用 tf32/fp64 且 $\rho \leq 0.2$ 时，多数情况下快于 Bartels–Stewart 算法
- GB200 TC 上 tf32 加速 $31.3\times$（90 PFLOPS/2880 TFLOPS）(NVIDIA, '25)

相对残差与迭代次数

低精度 = TensorFloat-32



(e) 相对残差。

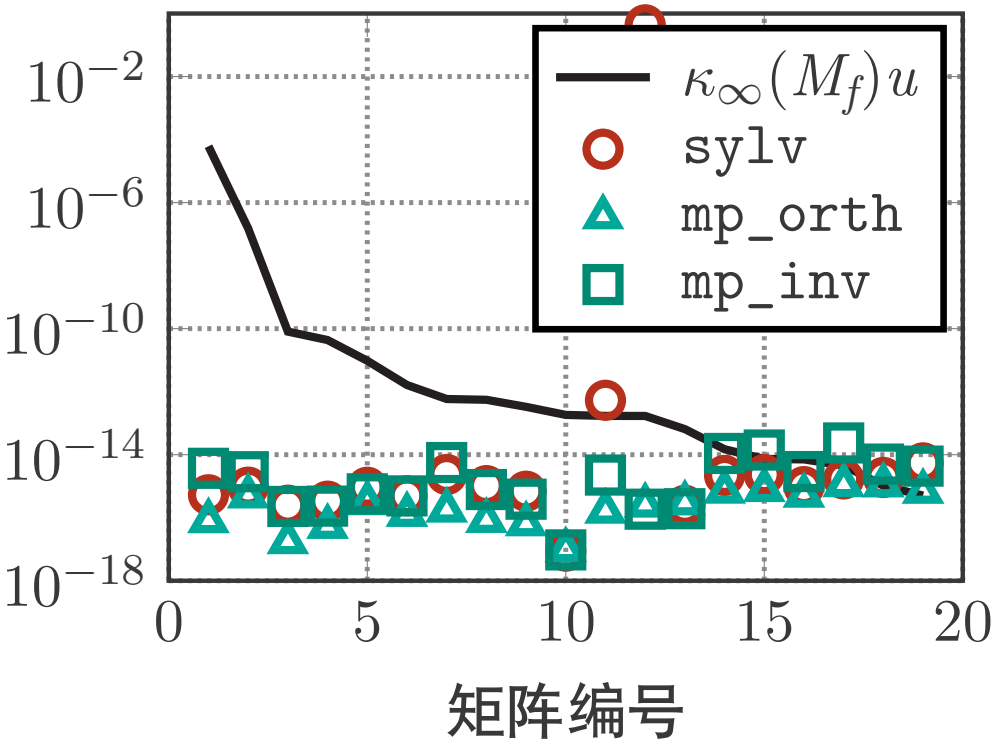


(f) 迭代次数。

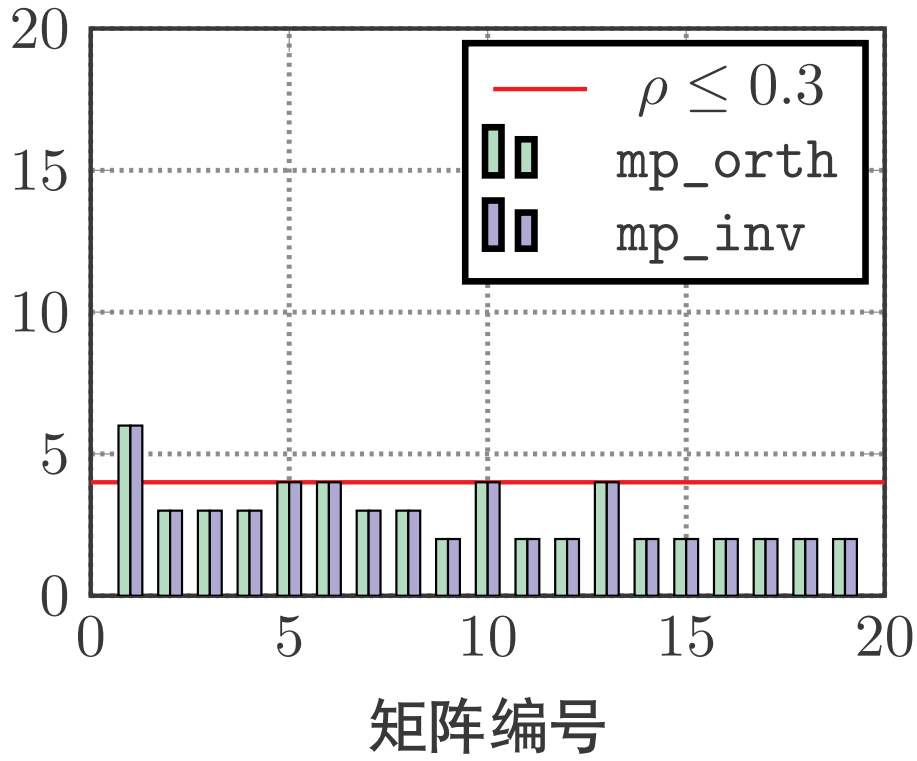
- 采用 tf32/fp64 且 $\rho \leq 0.2$ 时，多数情况下快于 Bartels–Stewart 算法
- GB200 TC 上 tf32 加速 $31.3\times$（90 PFLOPS/2880 TFLOPS）**(NVIDIA, '25)**

相对残差与迭代次数

低精度 = 自定义 24 位格式



(g) 相对残差。

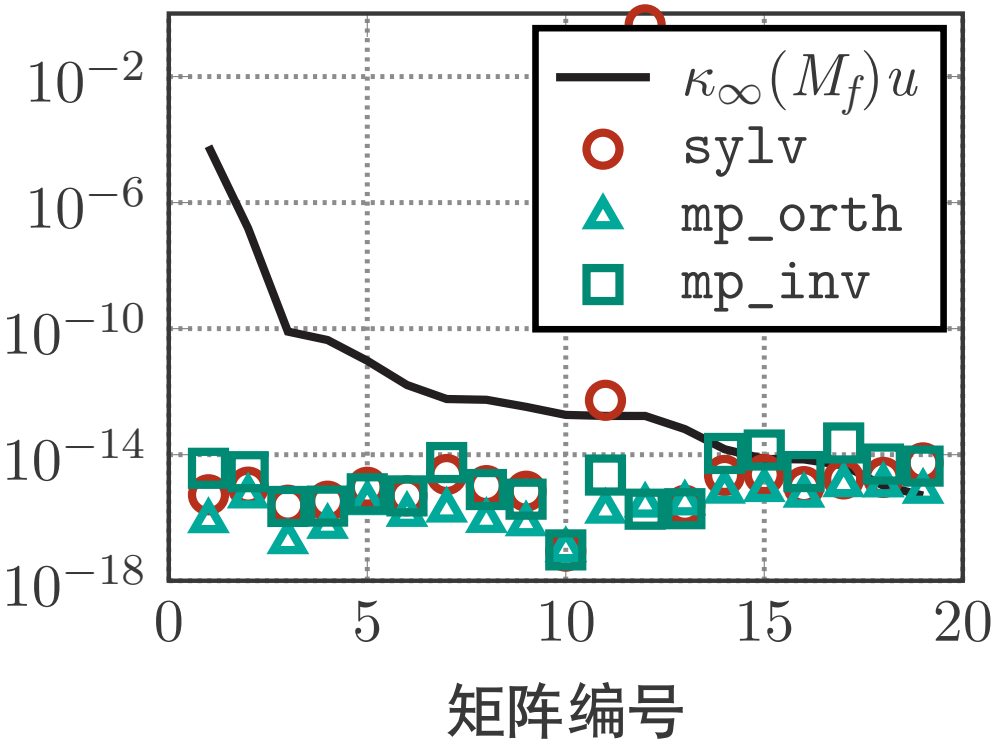


(h) 迭代次数。

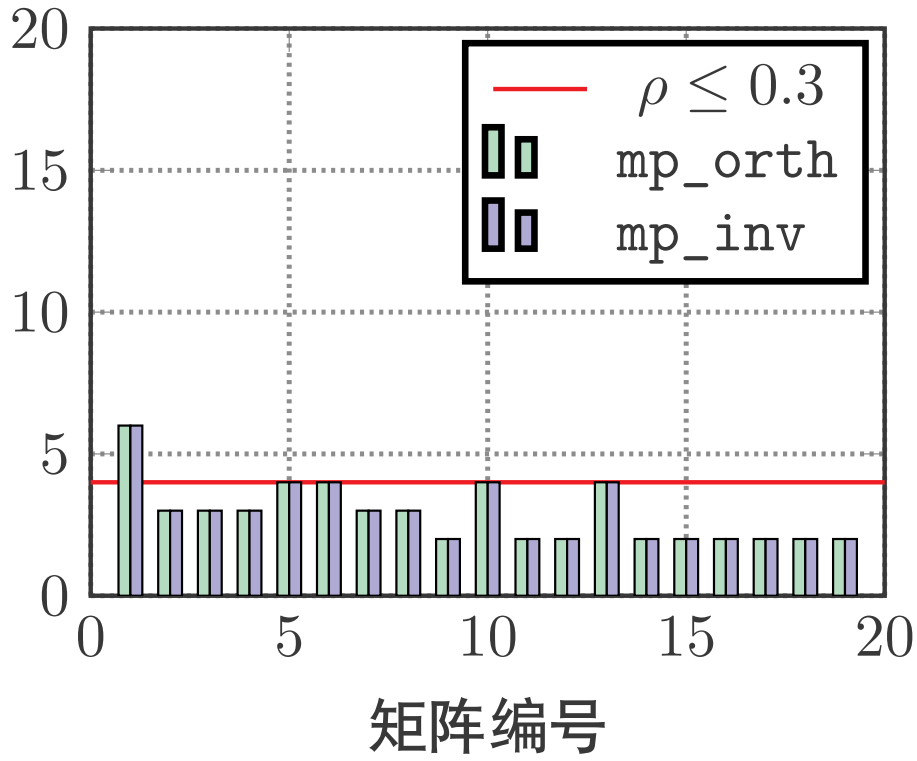
- 新算法在所有测试问题上均收敛
- 采用 24 位格式/fp64 且 $\rho \leq 0.3$ 时，多数情况下快于 Bartels–Stewart 算法

相对残差与迭代次数

低精度 = 自定义 24 位格式



(i) 相对残差。



(j) 迭代次数。

- 新算法在所有测试问题上均收敛
- 采用 24 位格式/fp64 且 $\rho \leq 0.3$ 时，多数情况下快于 Bartels–Stewart 算法

报告提纲

1. 混合精度简介
2. Sylvester 矩阵方程
3. **低秩 Lyapunov 矩阵方程**
4. 总结

Lyapunov 矩阵方程

低秩（代数）Lyapunov 方程

$$\underbrace{A}_{n\times n} X + X A^T = -\underbrace{L L^T}_{W=LL^T} \quad \text{或} \quad -\underbrace{L S L^T}_{W=LSL^T}$$

$$L \in \mathbb{R}^{n\times m}, \qquad S = S^T \in \mathbb{R}^{m\times m}, \qquad m \ll n.$$

应用于控制理论、系统平衡和模型降阶等。

- $n \sim 10^4$ 或更大
- $\operatorname{Re}\lambda_i(A) < 0$, W 为半正定矩阵
- X 为半正定矩阵，且具有较小的数值秩：

$$\|X - X_\varepsilon\| \le u\|X\|, \quad \operatorname{rank}(X_\varepsilon) \ll n$$

$\rightsquigarrow$ 寻求 $X = ZZ^T$，其中因子 $Z \approx$ 低秩矩阵 (Komaroff, '88).

Lyapunov 矩阵方程

低秩（代数）Lyapunov 方程

$$\underbrace{A}_{n\times n}X + X\underbrace{A^T}_{n\times n} = -\underbrace{LL^T}_{W=LL^T} \quad \text{或} \quad -\underbrace{LSL^T}_{W=LSL^T}$$

$$L \in \mathbb{R}^{n\times m}, \qquad S = S^T \in \mathbb{R}^{m\times m}, \qquad m \ll n.$$

应用于控制理论、系统平衡和模型降阶等。

- $n \sim 10^4$ 或更大
- $\operatorname{Re}\lambda_i(A) < 0$，$W$ 为半正定矩阵
- X 为半正定矩阵，且具有较小的数值秩：

$$\|X - X_\varepsilon\| \le u\|X\|, \quad \operatorname{rank}(X_\varepsilon) \ll n$$

$\rightsquigarrow$ 寻求 $X = ZZ^T$，其中因子 $Z \approx$ 低秩矩阵 (Komaroff, '88).

线性方程组的迭代改进

求解 $Ax = b$ 的混合精度框架

对 $Ax = b$，使用三种精度 $u_r \leq u \leq u_s$： **(Carson & Higham, '18)**

- u, 工作精度
- u_s, 求解修正方程
- u_r, 残差计算

---

1 以精度 u_s 求解 $Ax = b$，以精度 u 存储 x

2 **重复**

3 　　以精度 u_r 计算$r = b - Ax$，并将 r 舍入至精度 u_s

4 　　以精度 u_s 求解$Ad = r$，以精度 u 存储 d

5 　　以精度 u 更新 $x \leftarrow x + d$。

6 **直到** 收敛

- 后向误差可达 $O(u)$，取决于 $\kappa(A)$

推广至低秩 Lyapunov 方程

- 已有工作：用两种精度的迭代改进（IR）求解 $AX + XA^T + LL^T = 0$ **(Benner et al., '11)**.
  - 在 CPU–GPU 混合平台上的 fp32–fp64 实现
  - $X = ZZ^T$ 的 **Cholesky 型**解因子 Z
  - 未给出舍入误差分析
- 我们的贡献：
  - 推广至 **LDL^T 型**因子形式 $X = ZYZ^T$ $\rightsquigarrow$ 允许解的迭代矩阵为不定矩阵
  - 针对两种形式开展混合精度误差分析

推广至低秩 Lyapunov 方程

- 已有工作：用两种精度的迭代改进（IR）求解 $AX + XA^T + LL^T = 0$ **(Benner et al., '11)**.
  - 在 CPU–GPU 混合平台上的 fp32–fp64 实现
  - $X = ZZ^T$ 的 **Cholesky 型**解因子 Z
  - 未给出舍入误差分析
- 我们的贡献：
  - 推广至 **LDL^T 型**因子形式 $X = ZYZ^T$ $\rightsquigarrow$ 允许解的迭代矩阵为不定矩阵
  - 针对两种形式开展混合精度误差分析

低秩 Lyapunov 方程的混合精度迭代改进

LDL^T 型形式

计算 $AX + XA^T + LSL^T = 0$ 的解 $X = ZYZ^T$。

---

1 以精度 u_s 求解 $AX + XA^T + LSL^T = 0$，以精度 u 存储 Z 和 Y

2 **重复**

3 计算$\mathcal{R}(Z, Y) := AZYZ^T + ZYZ^TA^T + LSL^T = L_\Delta S_\Delta L_\Delta^T$ （采用精度 u_r），并将 L_Δ 和 S_Δ 舍入至精度 u_s

4 以精度 u_s 求解$AX_\Delta + X_\Delta A^T + L_\Delta S_\Delta L_\Delta^T = 0$，并以精度 u 存储 Z_Δ 和 Y_Δ（其中 $X_\Delta = Z_\Delta Y_\Delta Z_\Delta^T$）

5 以精度 u 求与 $X \leftarrow X + X_\Delta$ 最近的半正定矩阵的因子 Z 和 Y。

- 右端项非半正定；计算 Y、Z 时不显式构造 $n \times n$ 矩阵

6 **直到** 收敛

- 高于 u_s 的精度下仅需 $O(n^2)$ 次浮点运算：残差计算与解更新。

低秩 Lyapunov 方程的混合精度迭代改进

LDL^T 型形式

计算 $AX + XA^T + LSL^T = 0$ 的解 $X = ZYZ^T$。

---

1 以精度 u_s 求解 $AX + XA^T + LSL^T = 0$，以精度 u 存储 Z 和 Y

2 **重复**

3 　计算$\mathcal{R}(Z, Y) := AZYZ^T + ZYZ^TA^T + LSL^T = L_\Delta S_\Delta L_\Delta^T$ （采用精度 u_r），并将 L_Δ 和 S_Δ 舍入至精度 u_s

4 　以精度 u_s 求解$AX_\Delta + X_\Delta A^T + L_\Delta S_\Delta L_\Delta^T = 0$，并以精度 u 存储 Z_Δ 和 Y_Δ（其中 $X_\Delta = Z_\Delta Y_\Delta Z_\Delta^T$）

5 　以精度 u 求与 $X \leftarrow X + X_\Delta$ 最近的半正定矩阵的因子 Z 和 Y。

- 右端项非半正定；计算 Y、Z 时不显式构造 $n \times n$ 矩阵

6 **直到** 收敛

- 高于 u_s 的精度下仅需 $O(n^2)$ 次浮点运算：残差计算与解更新。

低秩 Lyapunov 方程的混合精度迭代改进

LDL^T 型形式

计算 $AX + XA^T + LSL^T = 0$ 的解 $X = ZYZ^T$。

---

1 以精度 u_s 求解 $AX + XA^T + LSL^T = 0$，以精度 u 存储 Z 和 Y

2 **重复**

3 　计算$\mathcal{R}(Z, Y) := AZYZ^T + ZYZ^TA^T + LSL^T = L_\Delta S_\Delta L_\Delta^T$（采用精度 u_r），并将 L_Δ 和 S_Δ 舍入至精度 u_s

4 　以精度 u_s 求解$AX_\Delta + X_\Delta A^T + L_\Delta S_\Delta L_\Delta^T = 0$，并以精度 u 存储 Z_Δ 和 Y_Δ（其中 $X_\Delta = Z_\Delta Y_\Delta Z_\Delta^T$）

5 　以精度 u 求与 $X \leftarrow X + X_\Delta$ 最近的半正定矩阵的因子 Z 和 Y。

- 右端项非半正定；计算 Y、Z 时不显式构造 $n \times n$ 矩阵

6 **直到** 收敛

- 高于 u_s 的精度下仅需 $O(n^2)$ 次浮点运算：残差计算与解更新。

低秩 Lyapunov 方程的混合精度迭代改进

LDL^T 型形式

计算 $AX + XA^T + LSL^T = 0$ 的解 $X = ZYZ^T$。

---

1 以精度 u_s 求解 $AX + XA^T + LSL^T = 0$，以精度 u 存储 Z 和 Y

2 **重复**

3 　计算$\mathcal{R}(Z, Y) := AZYZ^T + ZYZ^TA^T + LSL^T = L_\Delta S_\Delta L_\Delta^T$ （采用精度 u_r），并将 L_Δ 和 S_Δ 舍入至精度 u_s

4 　以精度 u_s 求解$AX_\Delta + X_\Delta A^T + L_\Delta S_\Delta L_\Delta^T = 0$，并以精度 u 存储 Z_Δ 和 Y_Δ（其中 $X_\Delta = Z_\Delta Y_\Delta Z_\Delta^T$）

5 　以精度 u 求与 $X \leftarrow X + X_\Delta$ 最近的半正定矩阵的因子 Z 和 Y。

- 右端项非半正定；计算 Y、Z 时不显式构造 $n \times n$ 矩阵

6 **直到** 收敛

- 高于 u_s 的精度下仅需 $O(n^2)$ 次浮点运算：残差计算与解更新。

LDL^T 型残差分解

1. 将 $\mathcal{R}(Z,\,Y) = AZYZ^T + ZYZ^TA^T + LSL^T$ 改写为

$$\mathcal{R}(Z,\,Y) = \begin{bmatrix} Z & AZ & L \end{bmatrix} \begin{bmatrix} 0 & Y & 0 \\ Y & 0 & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} Z^T \\ Z^TA^T \\ L^T \end{bmatrix}$$

$$=: FNF^T,\ F \text{ 为高瘦矩阵 } (Z \text{ 和 } L \text{ 亦如此}).$$

2. 计算薄 QR 分解 $F = UT$，构造 $H = TNT^T$，再计算谱分解 $H = Q\Lambda Q^T$。

3. 进行特征值截断，舍去满足 $|\lambda_i| < \eta_r$ 的项：

$$\mathcal{R}(Z,\,Y) \approx UQ_t\Lambda_tQ_t^TU^T =: L_\Delta S_\Delta L_\Delta^T,$$

其中

$$L_\Delta = UQ_t, \qquad S_\Delta = \Lambda_t.$$

LDL^T 型残差分解

1. 将 $\mathcal{R}(Z, Y) = AZYZ^T + ZYZ^TA^T + LSL^T$ 改写为

$$\mathcal{R}(Z, Y) = \begin{bmatrix} Z & AZ & L \end{bmatrix} \begin{bmatrix} 0 & Y & 0 \\ Y & 0 & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} Z^T \\ Z^TA^T \\ L^T \end{bmatrix}$$

$$=: FNF^T,\ F \text{ 为高瘦矩阵 } (Z \text{ 和 } L \text{ 亦如此}).$$

2. 计算薄 QR 分解 $F = UT$，构造 $H = TNT^T$，再计算谱分解 $H = Q\Lambda Q^T$。

3. 进行特征值截断，舍去满足 $|\lambda_i| < \eta_r$ 的项：

$$\mathcal{R}(Z, Y) \approx UQ_t\Lambda_tQ_t^TU^T =: L_\Delta S_\Delta L_\Delta^T,$$

其中

$$L_\Delta = UQ_t, \qquad S_\Delta = \Lambda_t.$$

LDL^T 型残差分解

1. 将 $\mathcal{R}(Z,\,Y) = AZYZ^T + ZYZ^TA^T + LSL^T$ 改写为

$$\mathcal{R}(Z,\,Y) = \begin{bmatrix} Z & AZ & L \end{bmatrix} \begin{bmatrix} 0 & Y & 0 \\ Y & 0 & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} Z^T \\ Z^TA^T \\ L^T \end{bmatrix}$$

$$=: FNF^T,\ F \text{ 为高瘦矩阵 } (Z \text{ 和 } L \text{ 亦如此}).$$

2. 计算薄 QR 分解 $F = UT$，构造 $H = TNT^T$，再计算谱分解 $H = Q\Lambda Q^T$。

3. 进行特征值截断，舍去满足 $|\lambda_i| < \eta_r$ 的项：

$$\mathcal{R}(Z,\,Y) \approx UQ_t\Lambda_tQ_t^TU^T =: L_\Delta S_\Delta L_\Delta^T,$$

其中

$$L_\Delta = UQ_t, \qquad S_\Delta = \Lambda_t.$$

LDL^T 型解因子更新

利用 Z_Δ 和 Y_Δ 更新 Z 和 Y。

1. 构造

$$G := \begin{bmatrix} Z & Z_\Delta \end{bmatrix},\ G \text{ 为高瘦矩阵},\ \text{且 } \Upsilon = \begin{bmatrix} Y & \\ & Y_\Delta \end{bmatrix}.$$

2. 计算薄 QR 分解 $G = V\Gamma$，构造 $K = \Gamma\Upsilon\Gamma^T$，再计算谱分解 $K = \Theta\Sigma\Theta^T$。

3. 进行特征值截断，舍去满足 $\sigma_i < \eta_s$ 的项：

$$Z_{\text{new}} = V\Theta_t, \quad Y_{\text{new}} = \Sigma_t,$$

使得

$$X_{\text{new}} \approx V\Theta_t\Sigma_t\Theta_t^T V^T = Z_{\text{new}}\, Y_{new} Z_{\text{new}}^T.$$

LDL^T 型解因子更新

利用 Z_Δ 和 Y_Δ 更新 Z 和 Y。

1. 构造

$$G := \begin{bmatrix} Z & Z_\Delta \end{bmatrix},\ G \text{ 为高瘦矩阵},\ \text{且 } \Upsilon = \begin{bmatrix} Y & \\ & Y_\Delta \end{bmatrix}.$$

2. 计算薄 QR 分解 $G = V\Gamma$，构造 $K = \Gamma\Upsilon\Gamma^T$，再计算谱分解 $K = \Theta\Sigma\Theta^T$。

3. 进行特征值截断，舍去满足 $\sigma_i < \eta_s$ 的项：

$$Z_{\text{new}} = V\Theta_t, \quad Y_{\text{new}} = \Sigma_t,$$

使得

$$X_{\text{new}} \approx V\Theta_t\Sigma_t\Theta_t^T V^T = Z_{\text{new}}\, Y_{new} Z_{\text{new}}^T.$$

LDL^T 型解因子更新

利用 Z_Δ 和 Y_Δ 更新 Z 和 Y。

1. 构造

$$G := \begin{bmatrix} Z & Z_\Delta \end{bmatrix},\ G \text{ 为高瘦矩阵},\ \text{且 } \Upsilon = \begin{bmatrix} Y & \\ & Y_\Delta \end{bmatrix}.$$

2. 计算薄 QR 分解 $G = V\Gamma$，构造 $K = \Gamma\Upsilon\Gamma^T$，再计算谱分解 $K = \Theta\Sigma\Theta^T$。

3. 进行特征值截断，舍去满足 $\sigma_i < \eta_s$ 的项：

$$Z_{\text{new}} = V\Theta_t, \quad Y_{\text{new}} = \Sigma_t,$$

使得

$$X_{\text{new}} \approx V\Theta_t\Sigma_t\Theta_t^T V^T = Z_{\text{new}}\, Y_{new} Z_{\text{new}}^T.$$

两种精度下的收敛界

表：采用稳定求解器时的条件数界与极限残差。

u_s	$u_r = u$	条件数上界 $\kappa_F(\mathcal{L})$	极限残差
bf16	fp32	10^3	fp32
fp16		10^4	
fp32		10^8	
bf16	fp64	10^3	fp64
fp16		10^4	
fp32		10^8	
fp64		10^{16}	

矩阵符号函数的 Newton 迭代

作为混合精度迭代改进的求解器

- **矩阵符号函数的 Newton 迭代 作为 LDL^T 型求解器 (Roberts, '80)**:

$$AX + XA^T + W = 0, \quad W = LSL^T, \quad X = ZYZ^T.$$

  - 实现简单且稳健
  - 具有良好的并行性
  - 大量使用三级 BLAS 运算
- 其他求解器也适用，只需返回 **Cholesky/LDL^T 型**解因子

矩阵符号函数的 Newton 迭代

对解因子进行迭代

对 $\mu_{k-1} > 0$，进行迭代 (Larin & Aliev, '93; Benner & Quintana-Ortí, '99)

$$
\begin{aligned}
A_k &= \frac{1}{2}\left(\mu_{k-1} A_{k-1} + \mu_{k-1}^{-1} A_{k-1}^{-1}\right), && A_0 = A, \\
Z_k &= \begin{bmatrix} Z_{k-1} & A_{k-1}^{-1} Z_{k-1} \end{bmatrix}, && Z_0 = L, \\
Y_k &= \frac{1}{2}\begin{bmatrix} \mu_{k-1} Y_{k-1} & \\ & \mu_{k-1}^{-1} Y_{k-1} \end{bmatrix}, && Y_0 = S.
\end{aligned}
$$

收敛时，

$$
\frac{1}{2} Z_k Y_k Z_k^T \longrightarrow X, \qquad Z = Z_k, \quad Y = \frac{1}{2} Y_k,
$$

且

$$
A_k \longrightarrow -I.
$$

矩阵符号函数的 Newton 迭代

对解因子进行迭代

对 $\mu_{k-1} > 0$，进行迭代 (Larin & Aliev, '93; Benner & Quintana-Ortí, '99)

$$
\begin{aligned}
A_k &= \frac{1}{2}\left(\mu_{k-1} A_{k-1} + \mu_{k-1}^{-1} A_{k-1}^{-1}\right), && A_0 = A, \\
Z_k &= \begin{bmatrix} Z_{k-1} & A_{k-1}^{-1} Z_{k-1} \end{bmatrix}, && Z_0 = L, \\
Y_k &= \frac{1}{2}\begin{bmatrix} \mu_{k-1} Y_{k-1} & \\ & \mu_{k-1}^{-1} Y_{k-1} \end{bmatrix}, && Y_0 = S.
\end{aligned}
$$

收敛时，

$$
\frac{1}{2} Z_k Y_k Z_k^T \longrightarrow X, \qquad Z = Z_k, \quad Y = \frac{1}{2} Y_k,
$$

且

$$
A_k \longrightarrow -I.
$$

矩阵符号函数的 Newton 迭代

秩截断

- 每次迭代中，Z_k 和 Y_k 的列数**加倍**，导致**存储量指数增长**：

$$
\begin{aligned}
A_k &= \frac{1}{2}\left(\mu_{k-1} A_{k-1} + \mu_{k-1}^{-1} A_{k-1}^{-1}\right), && A_0 = A, \\
Z_k &= \begin{bmatrix} Z_{k-1} & A_{k-1}^{-1} Z_{k-1} \end{bmatrix}, && Z_0 = L, \\
Y_k &= \begin{bmatrix} \frac{\mu_{k-1}}{2} Y_{k-1} & \\ & \frac{1}{2\mu_{k-1}} Y_{k-1} \end{bmatrix}, && Y_0 = S.
\end{aligned}
$$

$\rightsquigarrow$ 解决方法：对 Z_k 和 Y_k 进行秩截断 (Benner & Quintana-Ortí, '99)：

子程序 $[Z,\ Y] = \textsc{RankTruncLdlt}(Z,\ Y)$

1 计算薄 QR 分解 $Z = QR$。

2 计算谱分解 $RYR^T = \widetilde{V}\widetilde{\Lambda}\widetilde{V}^T$。

3 $\Lambda = \operatorname{diag}(\widetilde{\lambda}_i),\ i \in I_\lambda := \{i \mid \widetilde{\lambda}_i > u\|\widetilde{\Lambda}\|_1\}$ 且 $V = \widetilde{V}(:, I_\lambda)$

4 $Z = QV,\ Y = \Lambda$

矩阵符号函数的 Newton 迭代

秩截断

- 每次迭代中，Z_k 和 Y_k 的列数**加倍**，导致**存储量指数增长**：

$$A_k = \frac{1}{2}\left(\mu_{k-1} A_{k-1} + \mu_{k-1}^{-1} A_{k-1}^{-1}\right), \qquad A_0 = A,$$
$$Z_k = \begin{bmatrix} Z_{k-1} & A_{k-1}^{-1} Z_{k-1} \end{bmatrix}, \qquad Z_0 = L,$$
$$Y_k = \begin{bmatrix} \frac{\mu_{k-1}}{2} Y_{k-1} & \\ & \frac{1}{2\mu_{k-1}} Y_{k-1} \end{bmatrix}, \qquad Y_0 = S.$$

$\rightsquigarrow$ **解决方法**：对 Z_k 和 Y_k 进行秩截断 **(Benner & Quintana-Ortí, '99)**：

子程序 $[Z,\ Y] = \textsc{RankTruncLdlt}(Z,\ Y)$

1 计算薄 QR 分解 $Z = QR$。

2 计算谱分解 $RYR^T = \widetilde{V}\widetilde{\Lambda}\widetilde{V}^T$。

3 $\Lambda = \operatorname{diag}(\widetilde{\lambda}_i),\ i \in I_\lambda := \{i \mid \widetilde{\lambda}_i > u\|\widetilde{\Lambda}\|_1\}$ 且 $V = \widetilde{V}(:, I_\lambda)$

4 $Z = QV,\ Y = \Lambda$

矩阵符号函数的 Newton 迭代

完整算法

算法：求解 Lyapunov 方程的 LDL^T 型矩阵符号函数 Newton 迭代 $AX + XA^T + W = 0,\ W = LSL^T$.

$Z_0 = L,\ Y_0 = S,\ A_0 = A$

对 $k \leftarrow 1$ **至** $k_{\max}$ **执行**

  $A_k = \frac{1}{2}(\mu_{k-1} A_{k-1} + \mu_{k-1}^{-1} A_{k-1}^{-1})$ [*] 其中 μ_{k-1} 为缩放因子。

  $Z_k = \begin{bmatrix} Z_{k-1} & A_{k-1}^{-1} Z_{k-1} \end{bmatrix}$

  $Y_k = \frac{1}{2}\,\mathrm{blkdiag}(\mu_{k-1}\, Y_{k-1},\ \mu_{k-1}^{-1}\, Y_{k-1})$

  若 $\mathrm{size}(Z_k, 2) > \rho n$ **则** $[Z_k,\ Y_k] = \textsc{RankTruncLdlt}(Z_k,\ Y_k)$

  若 $\|A_k + I_n\|_1 \le \tau$ **则**

    再迭代两次后终止。

$Y = Y_k/2$

- 参数：收敛容差：$\tau = 10\sqrt{nu}$, 秩截断容差：$\rho \ll 1$
- [*] 每次迭代的主要计算成本：$2n^3$

Lyapunov 方程的迭代改进

计算成本分析

每个改进（外层）步骤中：

- 包含三个主要步骤
- $O(n^2)$ 次浮点运算用于残差分解和解因子更新
- 假设精度 u_r 或 u 下的一次浮点运算，成本不超过精度 u_s 下的 n 次浮点运算
- 主要成本（求解步骤）：k 次 Newton（内层）迭代共需 $2kn^3$ 次浮点运算

- 主要成本：精度 u_s 下 $2(k_0 + k_1 + \cdots + k_i)n^3$ 次浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 成本权衡：降低求解精度 u_s $\rightsquigarrow$ Newton（内层）迭代收敛变慢，k_i 增大。

Lyapunov 方程的迭代改进

计算成本分析

每个改进（外层）步骤中：

- 包含三个主要步骤
- $O(n^2)$ 次浮点运算用于残差分解和解因子更新
- 假设精度 u_r 或 u 下的一次浮点运算，成本不超过精度 u_s 下的 n 次浮点运算
- 主要成本（求解步骤）：k 次 Newton（内层）迭代共需 $2kn^3$ 次浮点运算

- 主要成本：精度 u_s 下 $2(k_0 + k_1 + \cdots + k_i)n^3$ 次浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 成本权衡：降低求解精度 $u_s \rightsquigarrow$ Newton（内层）迭代收敛变慢，k_i 增大。

Lyapunov 方程的迭代改进

计算成本分析

每个改进（外层）步骤中：

- 包含三个主要步骤
- $O(n^2)$ 次浮点运算用于残差分解和解因子更新
- 假设精度 u_r 或 u 下的一次浮点运算，成本不超过精度 u_s 下的 n 次浮点运算
- 主要成本（求解步骤）：k 次 Newton（内层）迭代共需 $2kn^3$ 次浮点运算

- 主要成本：精度 u_s 下 $2(k_0 + k_1 + \cdots + k_i)n^3$ 次浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 成本权衡：降低求解精度 u_s $\rightsquigarrow$ Newton（内层）迭代收敛变慢，k_i 增大。

Lyapunov 方程的迭代改进

另一种成本模型

- 事实：每个改进（外层）步骤都生成相同的序列：

$$A_1^{-1}, A_2^{-1}, \ldots, \quad A_0 = A.$$

⇝ **思路**：在不同改进迭代之间复用 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$

- 主要成本：~~$2\sum_{\ell=0}^{i} k_\ell n^3$~~ $2\max_{0\le\ell\le i} k_\ell n^3$ 次精度 u_s 下的浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 以额外存储换取更低的计算成本：当 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$ 可容纳于高速存储器时，尤其值得考虑

比较计算成本时关注的量：

$$k^\sigma =: \sum_{\ell=0}^{i} k_\ell \quad \text{或} \quad k^{\max} =: \max_{0\le\ell\le i} k_\ell.$$

比较采用不同求解器精度的混合精度迭代改进成本：

$$k^\sigma_{\text{low}} / k^\sigma_{\text{high}} \quad \text{或} \quad k^{\max}_{\text{low}} / k^{\max}_{\text{high}}$$

Lyapunov 方程的迭代改进

另一种成本模型

- 事实：每个改进（外层）步骤都生成相同的序列：

$$A_1^{-1}, A_2^{-1}, \ldots, \quad A_0 = A.$$

$\rightsquigarrow$ **思路**：在不同改进迭代之间复用 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$

- 主要成本：~~$2\sum_{\ell=0}^{i} k_\ell n^3$~~ $2\max_{0\le\ell\le i} k_\ell n^3$ 次精度 u_s 下的浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 以额外存储换取更低的计算成本：当 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$ 可容纳于高速存储器时，尤其值得考虑

比较计算成本时关注的量：

$$k^\sigma =: \sum_{\ell=0}^{i} k_\ell \quad \text{或} \quad k^{\max} =: \max_{0\le\ell\le i} k_\ell.$$

比较采用不同求解器精度的混合精度迭代改进成本：

$$k^\sigma_{\text{low}} / k^\sigma_{\text{high}} \quad \text{或} \quad k^{\max}_{\text{low}} / k^{\max}_{\text{high}}$$

Lyapunov 方程的迭代改进

另一种成本模型

- 事实：每个改进（外层）步骤都生成相同的序列：

$$A_1^{-1}, A_2^{-1}, \ldots, \quad A_0 = A.$$

$\rightsquigarrow$ **思路**：在不同改进迭代之间复用 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$

- 主要成本：~~$2\sum_{\ell=0}^{i} k_\ell n^3$~~ $2\max_{0\le\ell\le i} k_\ell n^3$ 次精度 u_s 下的浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 以额外存储换取更低的计算成本：当 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$ 可容纳于高速存储器时，尤其值得考虑

比较计算成本时关注的量：

$$k^\sigma =: \sum_{\ell=0}^{i} k_\ell \quad \text{或} \quad k^{\max} =: \max_{0\le\ell\le i} k_\ell.$$

比较采用不同求解器精度的混合精度迭代改进成本：

$$k^\sigma_{\text{low}} / k^\sigma_{\text{high}} \quad \text{或} \quad k^{\max}_{\text{low}} / k^{\max}_{\text{high}}$$

Lyapunov 方程的迭代改进

另一种成本模型

- 事实：每个改进（外层）步骤都生成相同的序列：

$$A_1^{-1}, A_2^{-1}, \ldots, \quad A_0 = A.$$

$\rightsquigarrow$ **思路**：在不同改进迭代之间复用 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$

- 主要成本：~~$2\sum_{\ell=0}^{i} k_\ell n^3$~~ $2\max_{0\le\ell\le i} k_\ell n^3$ 次精度 u_s 下的浮点运算。k_ℓ 为第 ℓ 个改进步骤中的 Newton（内层）迭代次数。
- 以额外存储换取更低的计算成本：当 $\widehat{A}_1^{-1}, \widehat{A}_2^{-1}, \ldots$ 可容纳于高速存储器时，尤其值得考虑

比较计算成本时关注的量：

$$k^\sigma =: \sum_{\ell=0}^{i} k_\ell \quad \text{或} \quad k^{\max} =: \max_{0\le\ell\le i} k_\ell.$$

比较采用不同求解器精度的混合精度迭代改进成本：

$$k^\sigma_{\text{low}} / k^\sigma_{\text{high}} \quad \text{或} \quad k^{\max}_{\text{low}} / k^{\max}_{\text{high}}$$

数值实验

实验设置

- 在 MATLAB 中使用 bf16、fp32、fp64，bf16 由 chop 模拟 (**Higham & Pranesh, '19**)
- 以 fp64 计算以下指标：

$$\mathrm{Res}(\widehat{Z},\widehat{Y}) = \frac{\|A\widehat{Z}\widehat{Y}\widehat{Z}^T + \widehat{Z}\widehat{Y}\widehat{Z}^T A^T + W\|_F}{\|W\|_F + 2\|\widehat{Z}\widehat{Y}\widehat{Z}^T\|_F\|A\|_F}.$$

参数

- 混合精度迭代改进的收敛容差 $= nu$
- 改进（外层）步数上限 $= 50$
- Newton（内层）迭代次数上限 $= 50$

运行时间模型

$$\rho = \frac{\text{一次 bf16 运算的平均时间}}{\text{高精度单次浮点运算平均时间}} = \frac{t_{\mathrm{bf16}}}{t_{\mathrm{high}}}$$

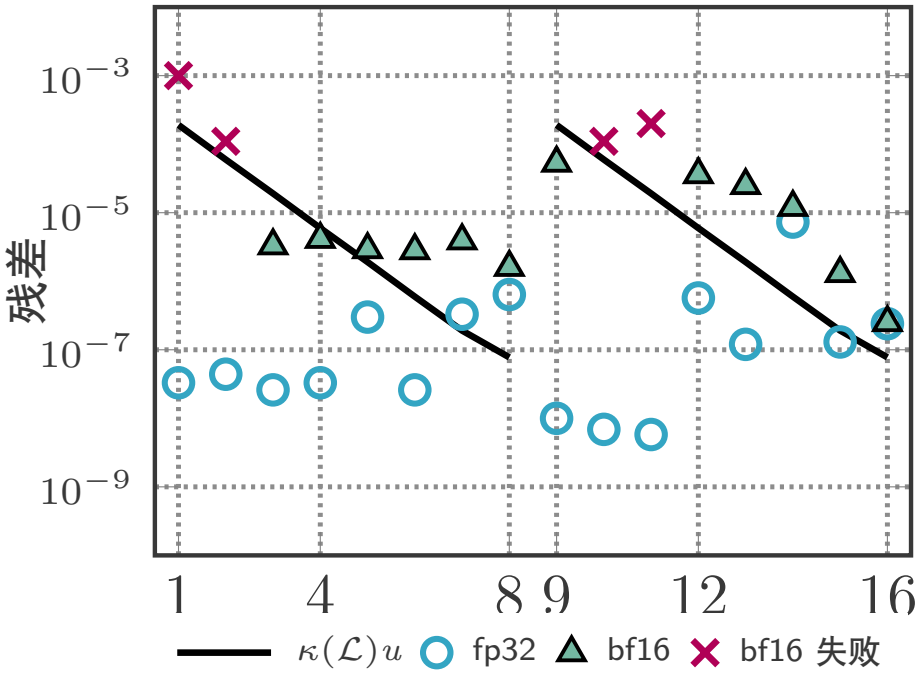
bf16 的主要耗时 $T_{\mathrm{bf16}} = 2n^3 k_{\mathrm{bf16}} \cdot t_{\mathrm{bf16}}$

高精度的主要耗时 $T_{\mathrm{high}} = 2n^3 k_{\mathrm{high}} \cdot t_{\mathrm{high}}$

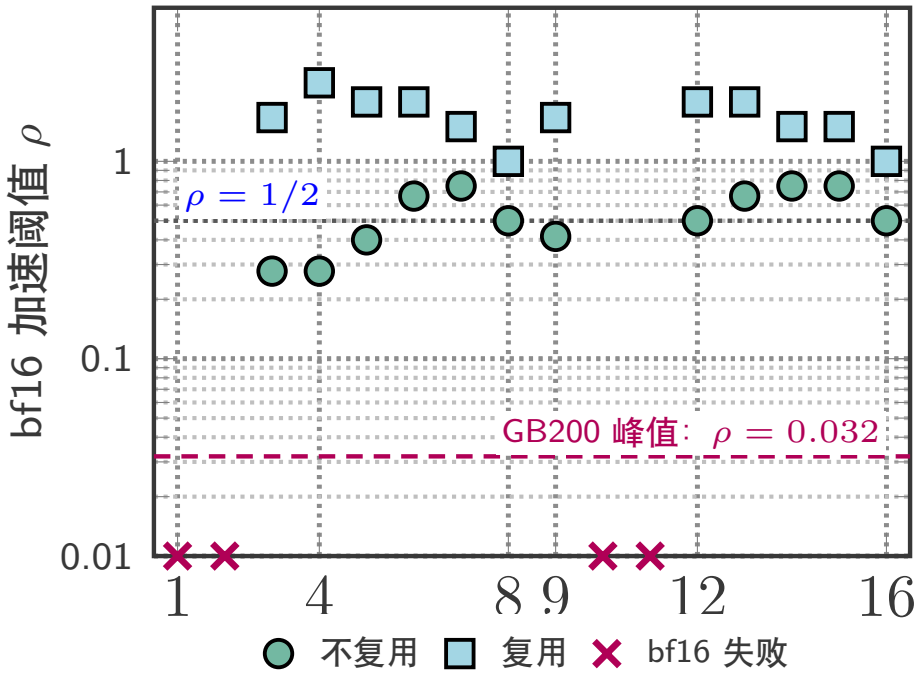
$$T_{\mathrm{bf16}} < T_{\mathrm{high}} \iff \frac{k_{\mathrm{bf16}}}{k_{\mathrm{high}}} < \frac{1}{\rho}, \quad 0 \le \rho \ll 1$$

伪随机矩阵，高精度 = fp32

横轴：问题编号



ρ 低于标记值时 bf16 方案更快



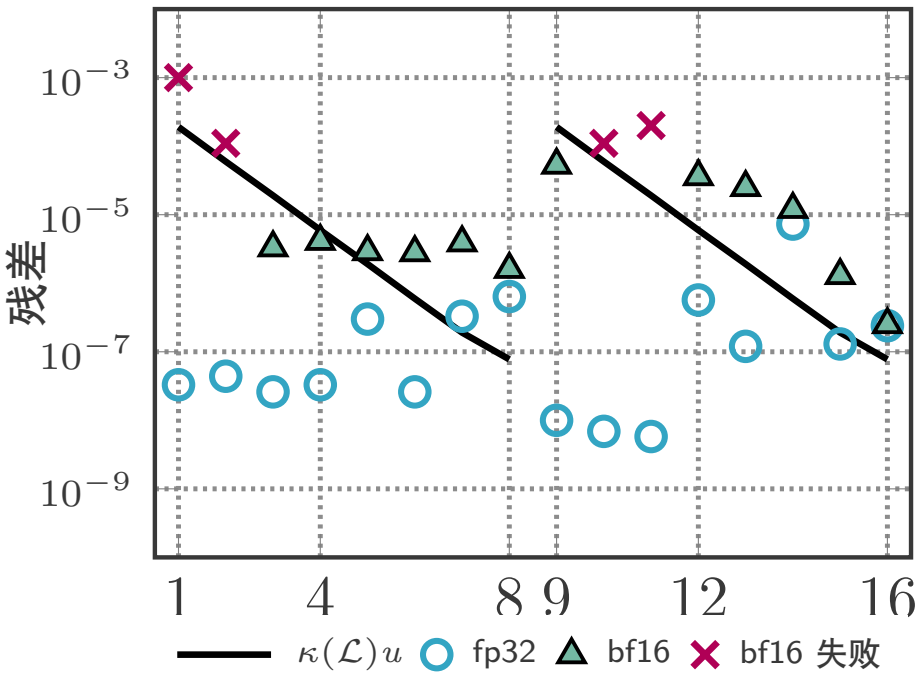
图：问题 1–8 的 $n = 100$，问题 9–16 的 $n = 1000$。

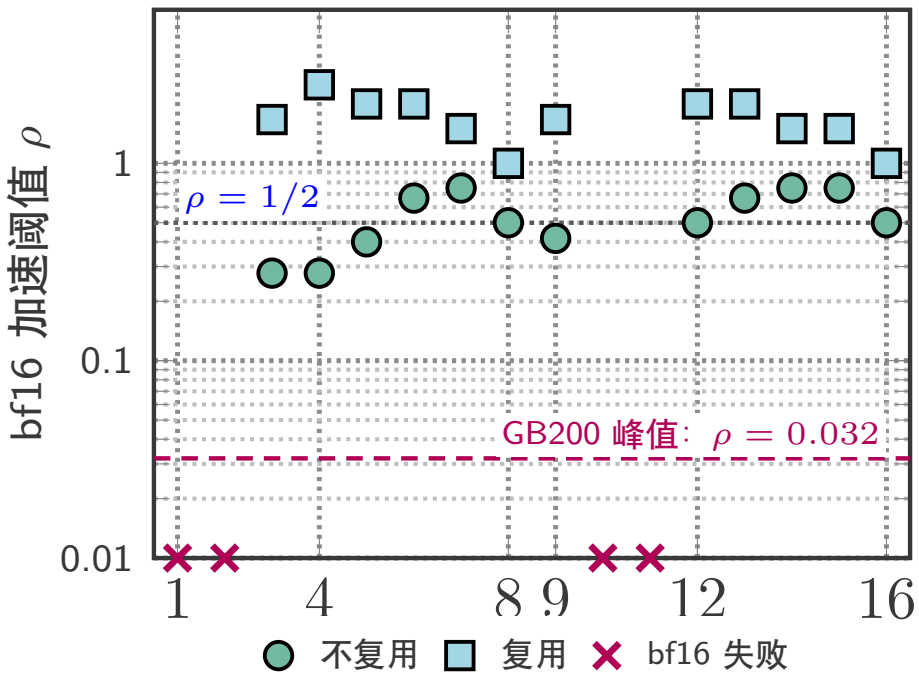
- 对良态问题，bf16/fp32 组合收敛
- GB200 参考线基于稠密运算速率：bf16 TC / 原生 fp32 (**NVIDIA, '25**)

伪随机矩阵，高精度 = fp32

横轴：问题编号

ρ 低于标记值时 bf16 方案更快



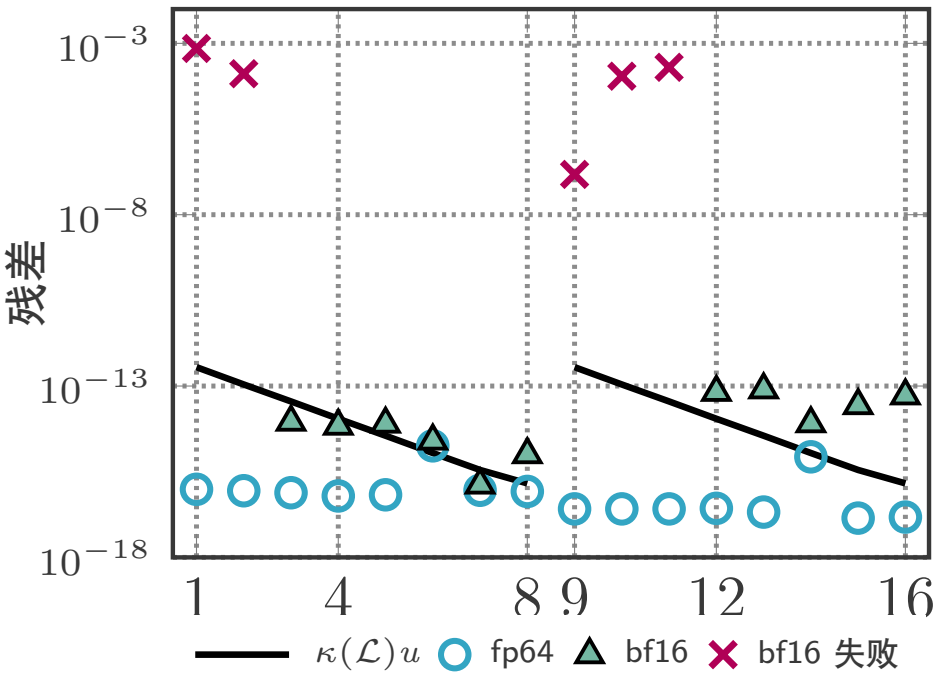


图：问题 1–8 的 $n = 100$，问题 9–16 的 $n = 1000$。

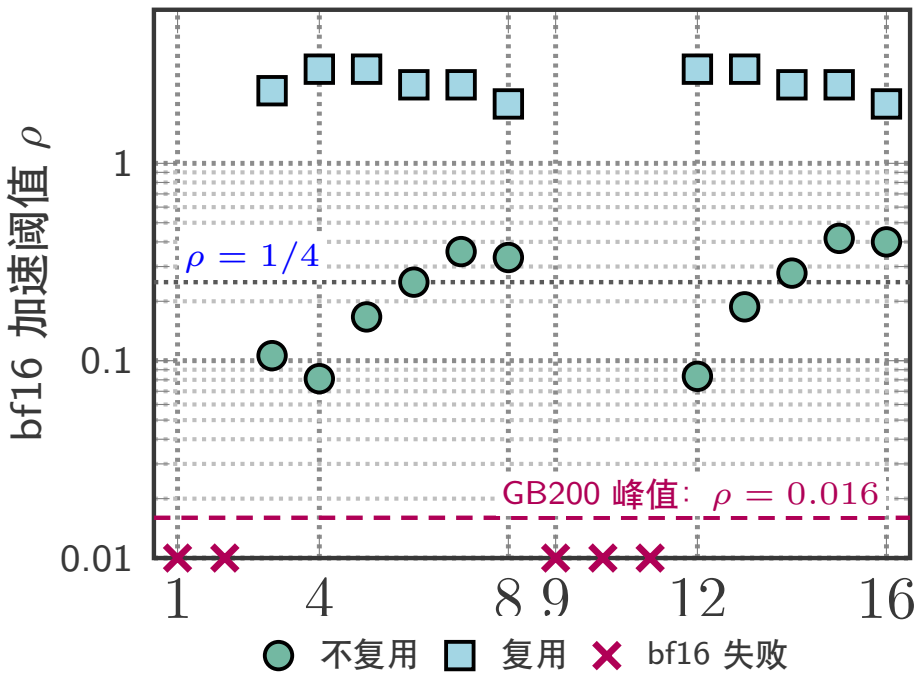
- 对良态问题，bf16/fp32 组合收敛
- GB200 参考线基于稠密运算速率：bf16 TC / 原生 fp32 **(NVIDIA, '25)**

伪随机矩阵，高精度 = fp64

横轴：问题编号



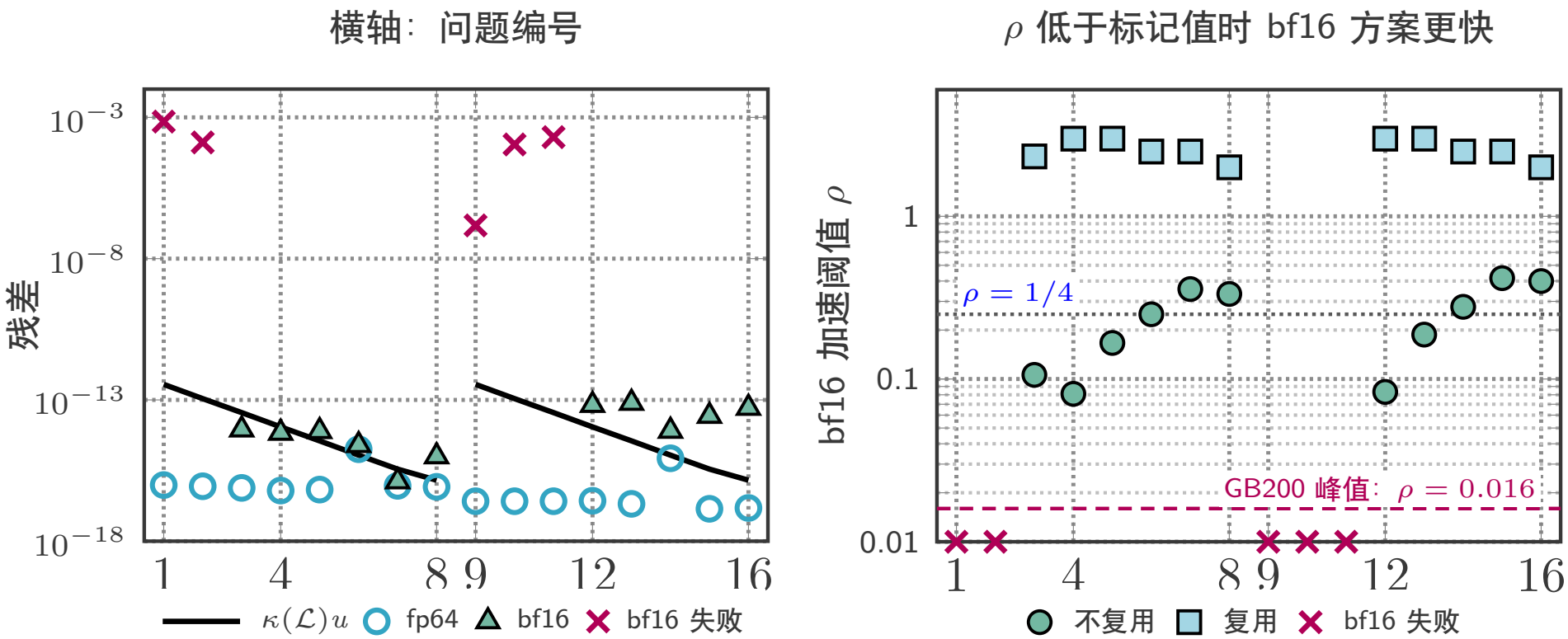
ρ 低于标记值时 bf16 方案更快



图：问题 1–8 的 $n = 100$，问题 9–16 的 $n = 1000$。

- 对良态问题，bf16/fp64 组合收敛
- GB200 参考线基于稠密运算速率：bf16 TC / fp64 TC (**NVIDIA, '25**)

伪随机矩阵，高精度 = fp64



图：问题 1–8 的 $n = 100$，问题 9–16 的 $n = 1000$。

- 对良态问题，bf16/fp64 组合收敛
- GB200 参考线基于稠密运算速率：bf16 TC / fp64 TC (**NVIDIA, '25**)

报告提纲

1. 混合精度简介
2. Sylvester 矩阵方程
3. 低秩 Lyapunov 矩阵方程
4. **总结**

总结

- 面向两类线性矩阵方程的混合精度算法

Sylvester 矩阵方程:

- 基于 Schur 分解 $\rightsquigarrow$ 中等规模稠密问题的首选方法
- 通过迭代改进获得三角方程的高精度解
- 利用低精度酉矩阵易于求逆或重新正交化的特点

低秩 Lyapunov 方程:

- Cholesky/LDL^T 型混合精度迭代改进框架
- 重点考虑以矩阵符号函数的 Newton 迭代为求解器
- bf16 等较低精度在良态问题上表现最佳

- 已完成但本报告未展开: 收敛性与舍入误差分析

下一步？

- 高性能混合精度实现（低精度 Schur 分解？）
- 在混合精度迭代改进内部采用低精度 ADI/Krylov 求解器？

更多细节请参阅预印本:

arxiv.org/abs/2503.03456 和 arxiv.org/abs/2510.02126

报告幻灯片: xiaobo-liu.github.io/talks

感谢聆听！

总结

- 面向两类线性矩阵方程的混合精度算法

Sylvester 矩阵方程:

- 基于 Schur 分解 $\rightsquigarrow$ 中等规模稠密问题的首选方法
- 通过迭代改进获得三角方程的高精度解
- 利用低精度酉矩阵易于求逆或重新正交化的特点

低秩 Lyapunov 方程:

- Cholesky/LDL^T 型混合精度迭代改进框架
- 重点考虑以矩阵符号函数的 Newton 迭代为求解器
- bf16 等较低精度在良态问题上表现最佳

- 已完成但本报告未展开: 收敛性与舍入误差分析

下一步？

- 高性能混合精度实现（低精度 Schur 分解？）
- 在混合精度迭代改进内部采用低精度 ADI/Krylov 求解器？

更多细节请参阅预印本:

arxiv.org/abs/2503.03456 和 arxiv.org/abs/2510.02126

报告幻灯片: xiaobo-liu.github.io/talks

感谢聆听!

总结

- 面向两类线性矩阵方程的混合精度算法

Sylvester 矩阵方程:

- 基于 Schur 分解 $\rightsquigarrow$ 中等规模稠密问题的首选方法
- 通过迭代改进获得三角方程的高精度解
- 利用低精度酉矩阵易于求逆或重新正交化的特点

低秩 Lyapunov 方程:

- Cholesky/LDL^T 型混合精度迭代改进框架
- 重点考虑以矩阵符号函数的 Newton 迭代为求解器
- bf16 等较低精度在良态问题上表现最佳

- 已完成但本报告未展开: 收敛性与舍入误差分析

下一步 ?

- 高性能混合精度实现 (低精度 Schur 分解 ?)
- 在混合精度迭代改进内部采用低精度 ADI/Krylov 求解器 ?

更多细节请参阅预印本:

arxiv.org/abs/2503.03456 和 arxiv.org/abs/2510.02126

报告幻灯片: xiaobo-liu.github.io/talks

感谢聆听!

总结

- 面向两类线性矩阵方程的混合精度算法

Sylvester 矩阵方程:

- 基于 Schur 分解 $\rightsquigarrow$ 中等规模稠密问题的首选方法
- 通过迭代改进获得三角方程的高精度解
- 利用低精度酉矩阵易于求逆或重新正交化的特点

低秩 Lyapunov 方程:

- Cholesky/LDL^T 型混合精度迭代改进框架
- 重点考虑以矩阵符号函数的 Newton 迭代为求解器
- bf16 等较低精度在良态问题上表现最佳

- 已完成但本报告未展开: 收敛性与舍入误差分析

下一步 ?

- 高性能混合精度实现 (低精度 Schur 分解 ?)
- 在混合精度迭代改进内部采用低精度 ADI/Krylov 求解器 ?

更多细节请参阅预印本:

arxiv.org/abs/2503.03456 和 arxiv.org/abs/2510.02126

报告幻灯片: xiaobo-liu.github.io/talks

感谢聆听!

Bibliography

- R. H. Bartels and G. W. Stewart.
  Algorithm 432: Solution of the Matrix Equation $AX + XB = C$.
  Comm. ACM, 15(9):820–826, 1972.
- P. Benner, P. Ezzatti, D. Kressner, E. S. Quintana-Ortí, and A. Remón.
  A mixed-precision algorithm for the solution of Lyapunov equations on hybrid CPU–GPU platforms.
  Parallel Comput., 37:439–450, 2011.
- P. Benner and E. S. Quintana-Ortí.
  Solving stable generalized Lyapunov equations with the matrix sign function.
  Numer. Algorithms, 20:75–100, 1999.
- E. Carson and N. J. Higham.
  Accelerating the solution of linear systems by iterative refinement in three precisions.
  SIAM J. Sci. Comput., 40(2):A817–A847, 2018.
- M. Fasi and M. Mikaitis.
  CPFloat: A C Library for Simulating Low-precision Arithmetic.
  ACM Trans. Math. Software, 49(2):1–32, 2023.
- N. J. Higham and S. Pranesh.
  Simulating low precision floating-point arithmetic.
  SIAM J. Sci. Comput., 41(5):C585–C602, 2019.
- N. Komaroff.
  Simultaneous eigenvalue lower bounds for the Lyapunov matrix equation.
  IEEE Trans. Automat. Control, 33(1):126–128, 1988.
- V. B. Larin and F. A. Aliev.
  Construction of square root factor for solution of the Lyapunov matrix equation.
  Syst. Control Lett., 20(2):109–112, 1993.
- NVIDIA Corporation.
  NVIDIA Blackwell Architecture Technical Brief.
  Tech. report, 2025.
- J. D. Roberts.
  Linear model reduction and solution of the algebraic Riccati equation by use of the sign function.
  Int. J. Control, 32(4):677–687, 1980.