MANCHESTER 1824

The University of Manchester

M\crNA
Manchester Numerical Analysis

Iterative Refinement in Three Precisions for Fast and Accurate Solution of Ill-Conditioned Sparse Linear Systems

Nick Higham
School of Mathematics
The University of Manchester

`http://www.maths.manchester.ac.uk/~higham/`

Joint work with Erin Carson (NYU)

SIAM Conference on Parallel Processing for Scientific Computing, Waseda University, Tokyo, March 7-10, 2018

IEEE Standard 754-1985 and 2008 Revision

Type	Size	Range	$u = 2^{-t}$
half	16 bits	$10^{\pm 5}$	$2^{-11} \approx 4.9 \times 10^{-4}$
single	32 bits	$10^{\pm 38}$	$2^{-24} \approx 6.0 \times 10^{-8}$
double	64 bits	$10^{\pm 308}$	$2^{-53} \approx 1.1 \times 10^{-16}$
quadruple	128 bits	$10^{\pm 4932}$	$2^{-113} \approx 9.6 \times 10^{-35}$

- Arithmetic ops $(+, -, *, /, \sqrt{})$ performed *as if* first calculated to infinite precision, then rounded.
- Default: round to nearest, round to even in case of tie.
- Half precision is a *storage format only*.

ARM NEON

ARM Developer ▾ Products Markets Technologies Support Se

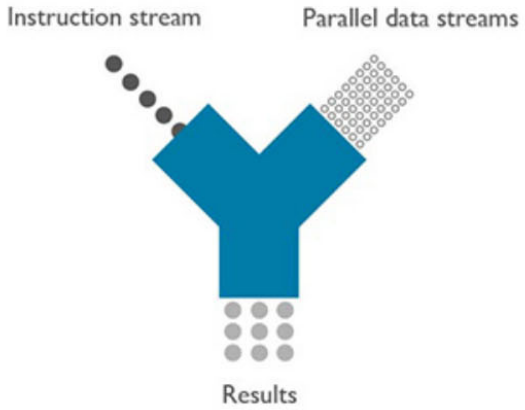
Overview big.LITTLE NEON DSP ▾ DynamIQ Graphics Technologies ▾ TrustZone ▾ Compute Library Floating Point

Overview

The NEON technology is a packed SIMD architecture. NEON registers are considered as vectors of elements of the same data type. Multiple data types are supported by the technology. The following table describes data types as supported by the architecture version.

	ARMv7-A/R	ARMv8-A/R	ARMv8-A
		AArch32	AArch64
Floating-point	32-bit	16-bit*/32-bit	16-bit*/32-bit/64-bit
Integer	8-bit/16-bit/32-bit	8-bit/16-bit/32-bit/64-bit	8-bit/16-bit/32-bit/64-bit

SIMD Architecture

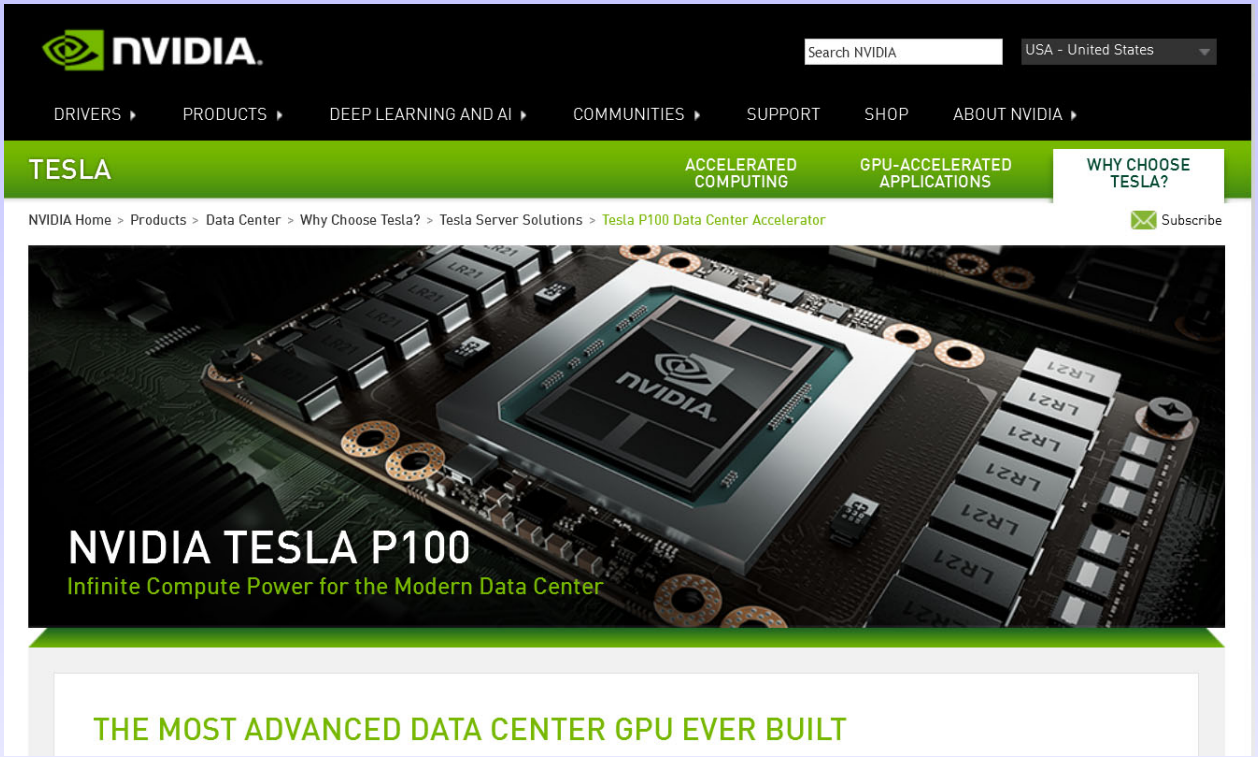


The NEON instructions perform the same operations in all lanes of the vectors. The number of operations performed depends on the data types. NEON instructions allow up to:

- 16x8-bit, 8x16-bit, 4x32-bit, 2x64-bit integer operations
- 8x16-bit*, 4x32-bit, 2x64-bit** floating-point operations

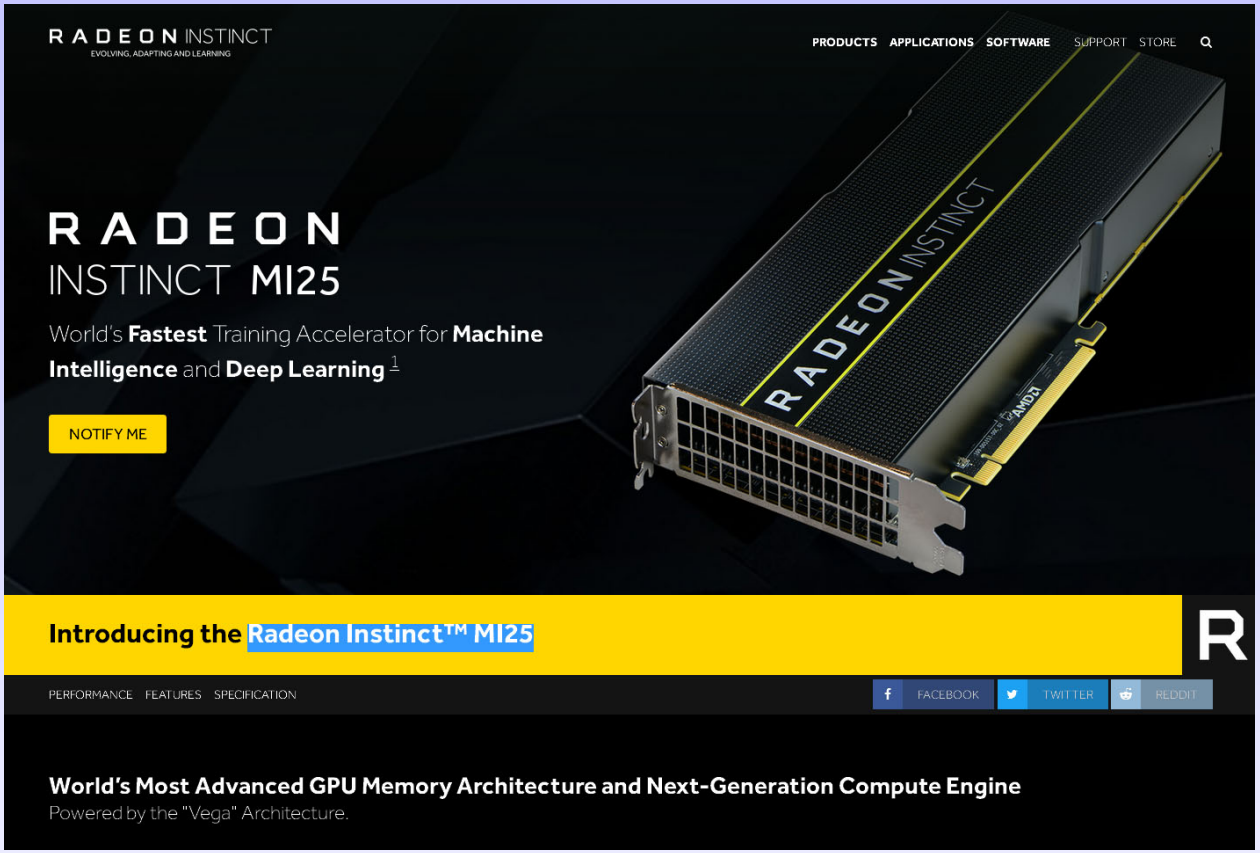
The implementation on NEON technology can also support issue of multiple instructions in parallel.

NVIDIA Tesla P100 (2016)



"The Tesla P100 is the world's first accelerator built for deep learning, and has native hardware ISA support for **FP16** arithmetic, delivering over 21 TeraFLOPS of **FP16** processing power."

AMD Radeon Instinct MI25 GPU (2017)



"24.6 TFLOPS FP16 or 12.3 TFLOPS FP32 peak GPU compute performance on a single board . . . Up to 82 GFLOPS/watt FP16 or 41 GFLOPS/watt FP32 peak GPU compute performance"

Accelerating the Solution of $Ax = b$

$A \in \mathbb{R}^{n\times n}$ nonsingular.

Standard method for solving $Ax = b$: factorize $A = LU$, solve $LUx = b$, all at working precision.

Can we solve $Ax = b$ **faster** and/or **more accurately** by exploiting multiprecision arithmetic?

Iterative Refinement for $Ax = b$ (classic)

Solve $Ax_0 = b$ by LU factorization in **double precision**.

- $r = b - Ax_0$ **quad precision**
- Solve $Ad = r$ **double precision**
- $x_1 = x_0 + d$ **double precision**

($x_0 \leftarrow x_1$ and iterate as necessary.)

- Programmed in **J. H. Wilkinson**, *Progress Report on the Automatic Computing Engine* (1948).
- Popular up to 1970s, exploiting cheap accumulation of inner products.

Iterative Refinement (1970s, 1980s)

Solve $Ax_0 = b$ by LU factorization.

- $r = b - Ax_0$
- Solve $Ad = r$
- $x_1 = x_0 + d$

Everything in **double precision**.

- **Skeel** (1980).
- **Jankowski & Woźniakowski** (1977) for a general solver.

Iterative Refinement (2000s)

Solve $Ax_0 = b$ by LU factorization in **single precision**.

- $r = b - Ax_0$ **double precision**
- Solve $Ad = r$ **single precision**
- $x_1 = x_0 + d$ **double precision**

- **Dongarra et al.** (2006).
- Motivated by single precision being at least twice as fast as double.

Iterative Refinement in Three Precisions

A, b given in **precision u**.

Solve $Ax_0 = b$ by LU factorization in **precision u_f**.

- $r = b - Ax_0$ **precision u_r**
- Solve $Ad = r$ **precision u_f**
- $x_1 = \mathrm{fl}(x_0 + d)$ **precision u**

- Three previous usages are special cases.
- Choose precisions from half, single, double, quadruple subject to $u_r \leq u \leq u_f$.
- **Can we compute *more accurate* solutions *faster*?**

Existing Rounding Error Analysis

- **Wilkinson** (1963): fixed-point arithmetic.
- **Moler** (1967): floating-point arithmetic.
- **Higham** (1997, 2002): more general analysis for arbitrary solver.
- **Dongarra et al.** (2006): lower precision LU.

At most two precisions and require $\kappa(A)u < 1$.

New Analysis

- Applies to any solver.
- Covers b'err and f'err. Focus mainly on f'err.
- Allows $\kappa(A)u \lesssim 1$.

New Analysis

Assume computed solution to $Ad_i = r_i$ has normwise rel b'err $O(u_f)$ and satisfies

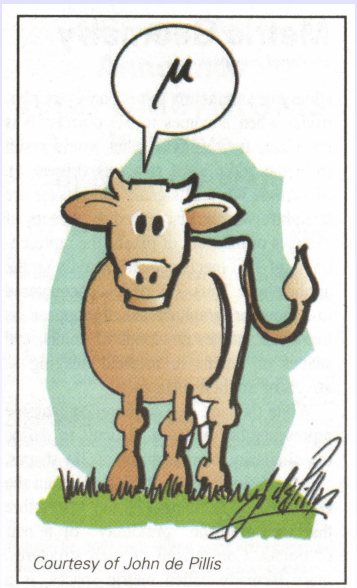
$$\frac{\|d_i - \widehat{d}_i\|_\infty}{\|d_i\|} \leq u_f \theta_i < 1.$$

Define μ_i by

$$\|A(x - \widehat{x}_i)\|_\infty = \mu_i \|A\|_\infty \|x - \widehat{x}_i\|_\infty,$$

and note that

$$\kappa_\infty(A)^{-1} \leq \mu_i \leq 1.$$



Condition Numbers

$|A| = (|a_{ij}|)$.

$$\operatorname{cond}(A, x) = \frac{\| \, |A^{-1}||A||x| \, \|_\infty}{\|x\|_\infty},$$

$$\operatorname{cond}(A) = \operatorname{cond}(A, e) = \| \, |A^{-1}||A| \, \|_\infty,$$

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.$$

$$1 \le \operatorname{cond}(A, x) \le \operatorname{cond}(A) \le \kappa_\infty(A).$$

Convergence Result

Theorem (Carson & H, 2017)

For IR in precisions $u_r \le u \le u_f$ if

$$\phi_i = 2u_f \min\bigl(\mathrm{cond}(A), \kappa_\infty(A)\mu_i\bigr) + u_f\theta_i$$

is sufficiently less than 1*, the forward error is reduced on the ith iteration by a factor $\approx \phi_i$ until an iterate $\widehat{x}$ satisfies*

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim 4nu_r\mathrm{cond}(A,x) + u.$$

Analogous standard bound would have

- $\mu_i = 1$,
- $u_f\theta_i = \kappa(A)u_f$.

Backward Error Result

Theorem (Carson & H, 2017)

For IR in precisions $u_r \le u \le u_f$, if

$$\phi = (c_1 \kappa_\infty(A) + c_2) u_f$$

is sufficiently less than 1 *then the residual is reduced on each iteration by a factor approximately ϕ until*

$$\|b - A\widehat{x}_i\|_\infty \lesssim nu(\|b\|_\infty + \|A\|_\infty \|\widehat{x}_i\|_\infty).$$

Precision Combinations

H = half, S = single, D = double, Q = quad. "u_f u u_r":

Traditional:

SSD
DDQ
HHS
HHD
HHQ
SSQ

1970s/1980s:

SSS
DDD
HHH
QQQ

2000s:

SDD
HSS
DQQ
HDD
HQQ
SQQ

3 precisions:

HSD
HSQ
HDQ
SDQ

Results for LU Factorization (1)

u_f	u	u_r	$\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
H	S	S	10^4	S	S	$\mathrm{cond}(A,x)\cdot$ S
H	S	D	10^4	S	S	S
H	D	D	10^4	D	D	$\mathrm{cond}(A,x)\cdot$ D
H	D	Q	10^4	D	D	D
S	S	S	10^8	S	S	$\mathrm{cond}(A,x)\cdot$ S
S	S	D	10^8	S	S	S
S	D	D	10^8	D	D	$\mathrm{cond}(A,x)\cdot$ D
S	D	Q	10^8	D	D	D

Results (2): HSD vs. SSD

u_f	u	u_r	$\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
H	S	S	10^4	S	S	$\mathrm{cond}(A,x)\cdot$ S
H	S	D	10^4	S	S	S
H	D	D	10^4	D	D	$\mathrm{cond}(A,x)\cdot$ D
H	D	Q	10^4	D	D	D
S	S	S	10^8	S	S	$\mathrm{cond}(A,x)\cdot$ S
S	S	D	10^8	S	S	S
S	D	D	10^8	D	D	$\mathrm{cond}(A,x)\cdot$ D
S	D	Q	10^8	D	D	D

Results (2): HSD vs. SSD

u_f	u	u_r	$\kappa_\infty(A)$	Backward error norm	Backward error comp	Forward error
H	S	S	10^4	S	S	$\mathrm{cond}(A,x)\cdot$ S
H	S	D	10^4	S	S	S
H	D	D	10^4	D	D	$\mathrm{cond}(A,x)\cdot$ D
H	D	Q	10^4	D	D	D
S	S	S	10^8	S	S	$\mathrm{cond}(A,x)\cdot$ S
S	S	D	10^8	S	S	S
S	D	D	10^8	D	D	$\mathrm{cond}(A,x)\cdot$ D
S	D	Q	10^8	D	D	D

Can we get the benefit of "HSD" while allowing a larger range of $\kappa_\infty(A)$?

Extending the Range of Applicability

Recall that the convergence condition is

$$\phi_i = 2u_f \min\bigl(\mathrm{cond}(A), \kappa_\infty(A)\mu_i\bigr) + u_f\,\theta_i \ll 1.$$

- We need **both terms** to be smaller than $\kappa_\infty(A)u_f$.

Recall that

$$\frac{\|d_i - \widehat{d}_i\|_\infty}{\|d_i\|} \leq u_f\,\theta_i,$$

$$\mu_i \|A\|_\infty \|x - x_i\|_\infty = \|A(x - x_i)\|_\infty = \|b - Ax_i\|_\infty = \|r_i\|_\infty.$$

Bounding μ_i

For a stable solver, in the early stages we expect

$$\frac{\|r_i\|}{\|A\|\|x_i\|} \approx u \ll \frac{\|x - x_i\|}{\|x\|},$$

or equivalently $\mu_i \ll 1$. But **close to convergence**

$$\frac{\|r_i\|}{\|A\|\|x_i\|} \approx u \approx \frac{\|x - x_i\|}{\|x\|} \quad \text{or} \quad \mu_i \approx 1.$$

Conclude

$\mu_i \ll 1$ initially and $\mu_i \to 1$ as the iteration converges.

Bounding θ_i

- $u_f\,\theta_i$ bounds rel error in solution of $Ad_i = r_i$.
- We need $u_f\,\theta_i \ll 1$.

Standard solvers cannot achieve this for very ill conditioned A!

Empirically observed by **Rump (1990)** that if $\widehat{L}$ and $\widehat{U}$ are computed LU factors of A from GEPP then

$$\kappa(\widehat{L}^{-1}A\widehat{U}^{-1}) \approx 1 + \kappa(A)u,$$

even for $\kappa(A) \gg u^{-1}$.

Preconditioned GMRES

To compute the updates d_i we apply **GMRES** to

$$\widetilde{A}d_i \equiv \widehat{U}^{-1}\widehat{L}^{-1}Ad_i = \widehat{U}^{-1}\widehat{L}^{-1}r_i.$$

- $\widetilde{A}$ is applied in twice the working precision.
- $\kappa(\widetilde{A}) \ll \kappa(A)$ typically.
- Rounding error analysis shows we get an accurate $\widehat{d}_i$ even for numerically singular A.
- Call the overall alg **GMRES-IR**.

- GMRES cgce rate not directly related to $\kappa(\widetilde{A})$.

Cf. **Kobayashi & Ogita** (2015), who explicitly form $\widetilde{A}$.

Benefits of GMRES-IR

Recall $H = 10^{-4}$, $S = 10^{-8}$, $D = 10^{-16}$, $Q = 10^{-34}$.

	u_f	u	u_r	$\kappa_\infty(A)$	Backward error nrm	cmp	F'error
LU	H	D	Q	10^4	D	D	D
LU	S	D	Q	10^8	D	D	D

Benefits of GMRES-IR

Recall $H = 10^{-4}$, $S = 10^{-8}$, $D = 10^{-16}$, $Q = 10^{-34}$.

	u_f	u	u_r	$\kappa_\infty(A)$	Backward error nrm	cmp	F'error
LU	H	D	Q	10^4	D	D	D
LU	S	D	Q	10^8	D	D	D
GMRES-IR	H	D	Q	10^{12}	D	D	D
GMRES-IR	S	D	Q	10^{16}	D	D	D

Benefits of GMRES-IR

Recall $H = 10^{-4}$, $S = 10^{-8}$, $D = 10^{-16}$, $Q = 10^{-34}$.

	u_f	u	u_r	$\kappa_\infty(A)$	Backward error nrm	cmp	F'error
LU	H	D	Q	10^4	D	D	D
LU	S	D	Q	10^8	D	D	D
GMRES-IR	H	D	Q	10^{12}	D	D	D
GMRES-IR	S	D	Q	10^{16}	D	D	D

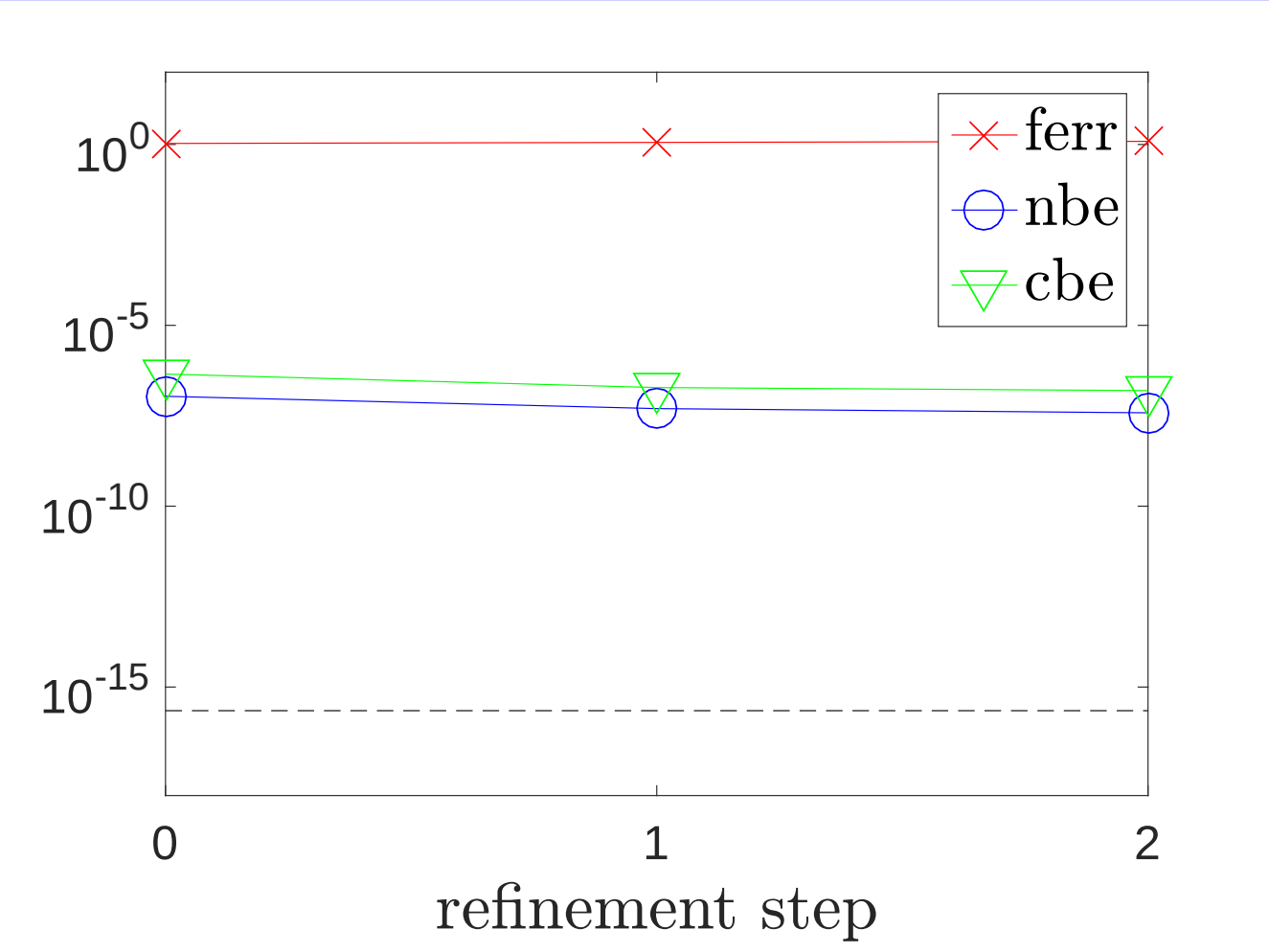
How many GMRES iterations are required?

Some tests with 100×100 matrices ...

Test 1: LU-IR, $(u_f, u, u_r) = (S, D, D)$

$\kappa_\infty(A) \approx 10^{10}$, $\sigma_i = \alpha^i$

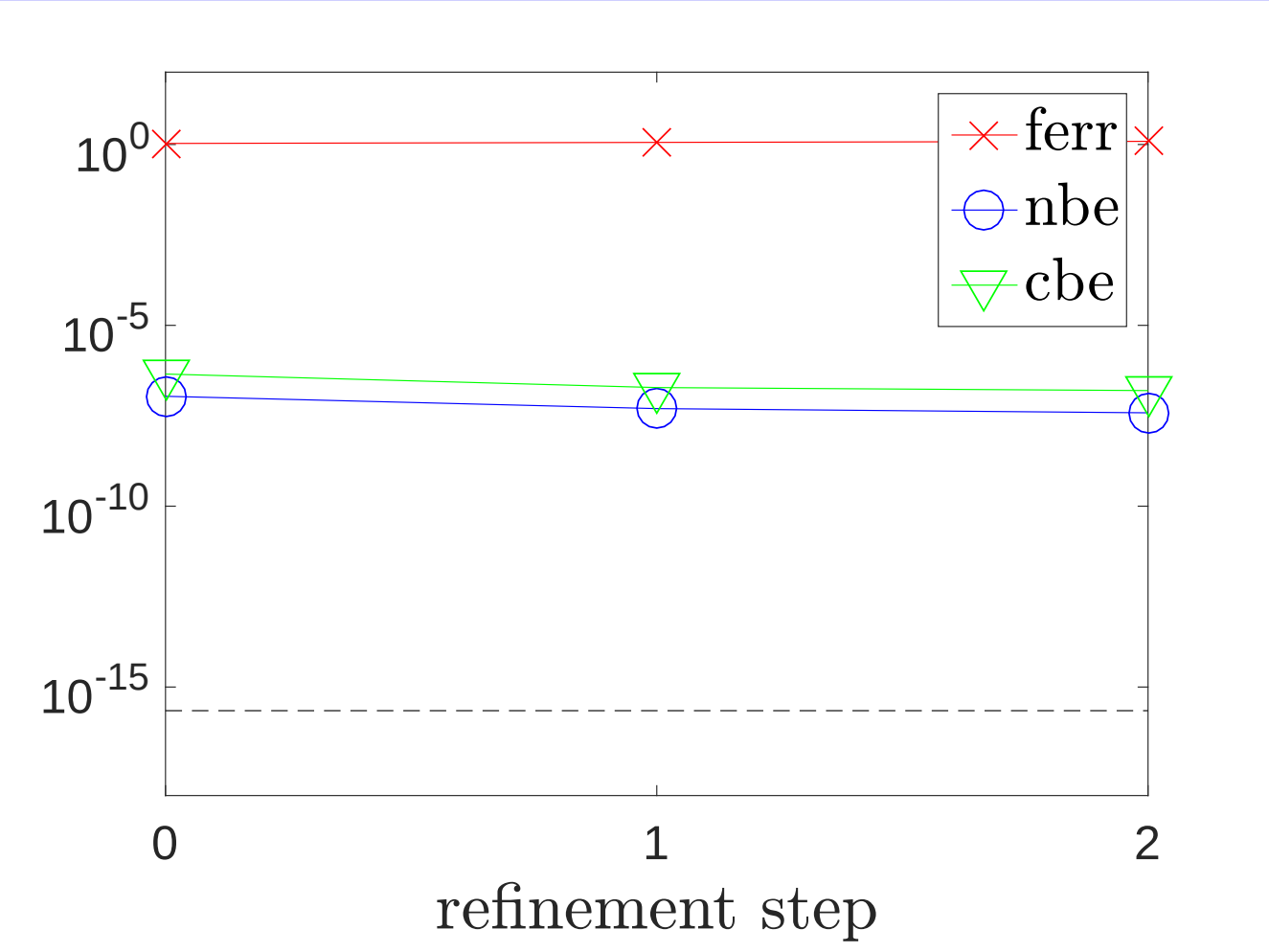
Divergence!



Test 1: LU-IR, $(u_f, u, u_r) = (S, D, Q)$

$\kappa_\infty(A) \approx 10^{10}$, $\sigma_i = \alpha^i$

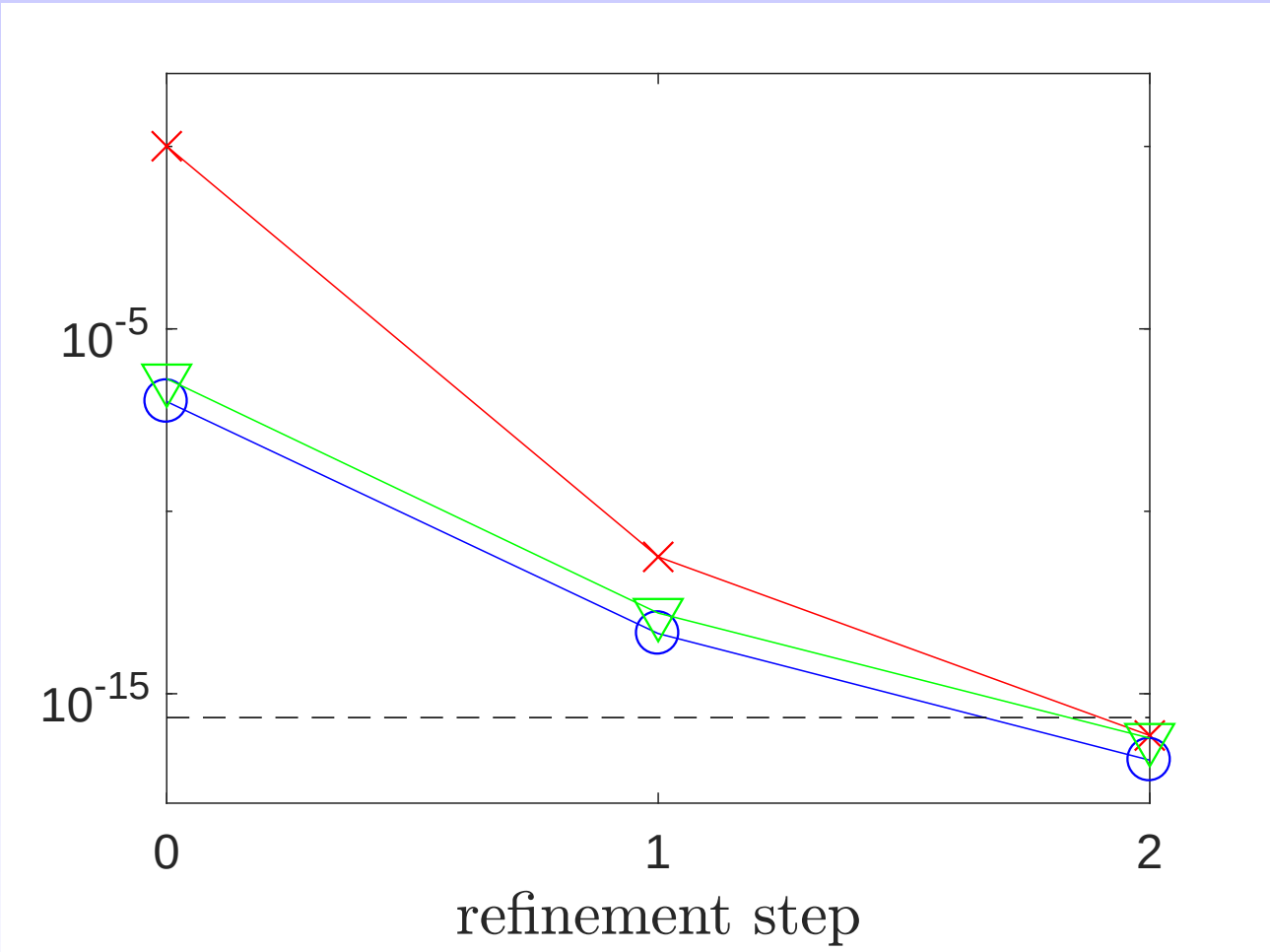
Divergence!



Test 1: LU-IR, $(u_f, u, u_r) = (S, D, Q)$

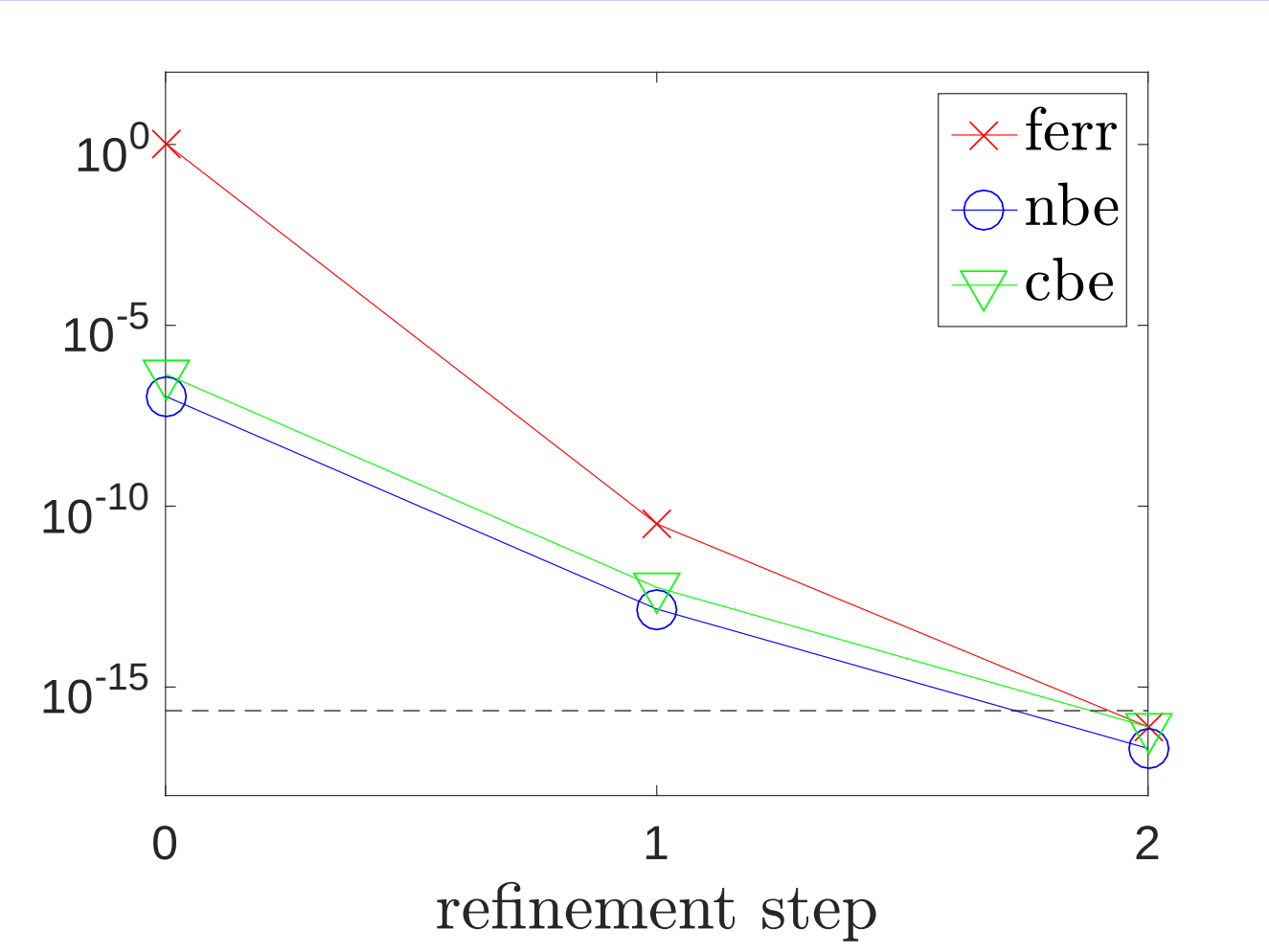
$\kappa_\infty(A) \approx 10^4$, $\sigma_i = \alpha^i$

Convergence



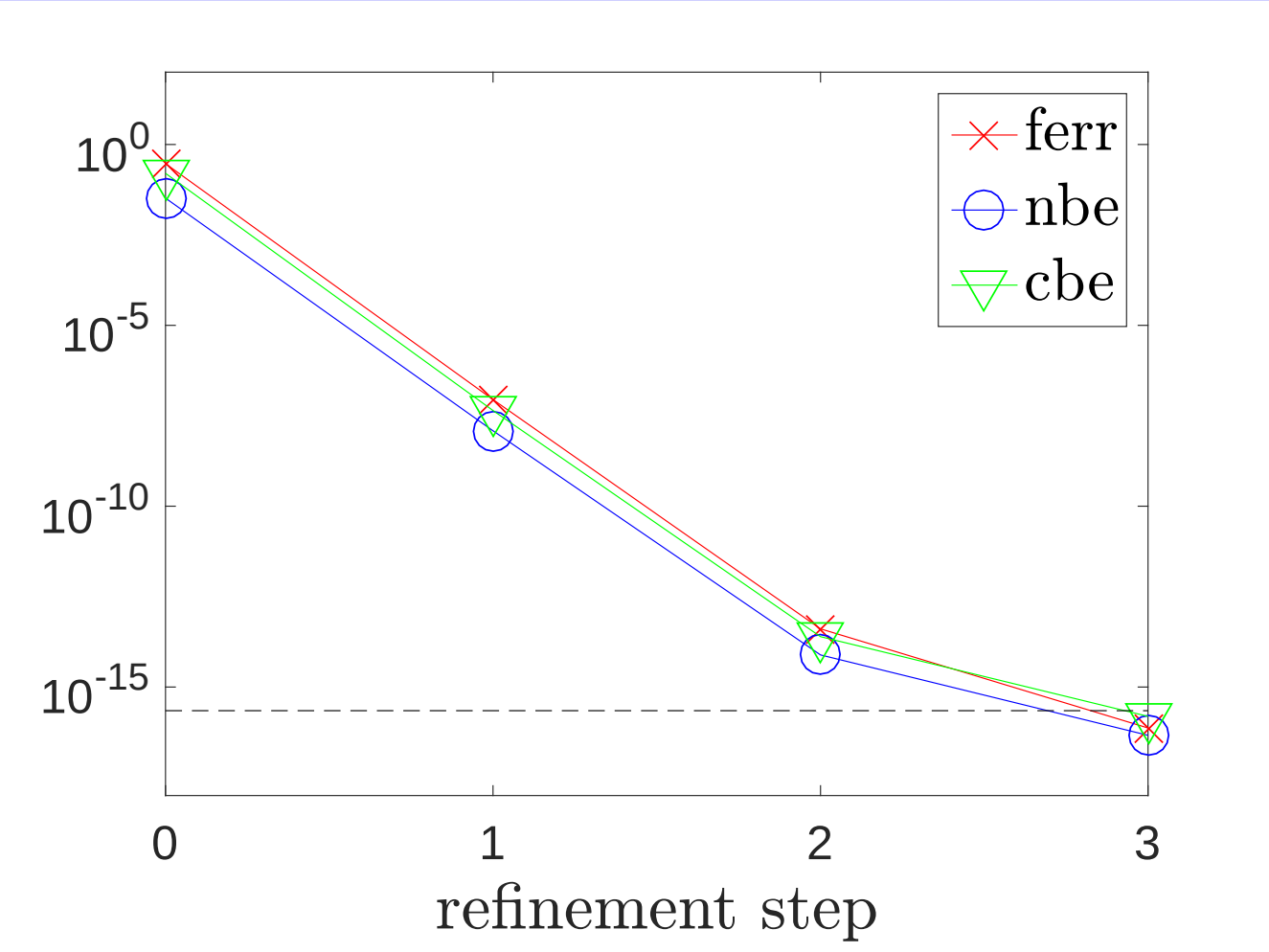
Test 1: GMRES-IR, $(u_f, u, u_r) = (S, D, Q)$

$\kappa_\infty(A) \approx 10^{10}$, $\sigma_i = \alpha^i$, GMRES its (2,3) **Convergence**



Test 2: GMRES-IR, $(u_f, u, u_r) = (H, D, Q)$

$\kappa_\infty(A) \approx 10^2$, 1 small σ_i, GMRES its (8,8,8) **Convergence**

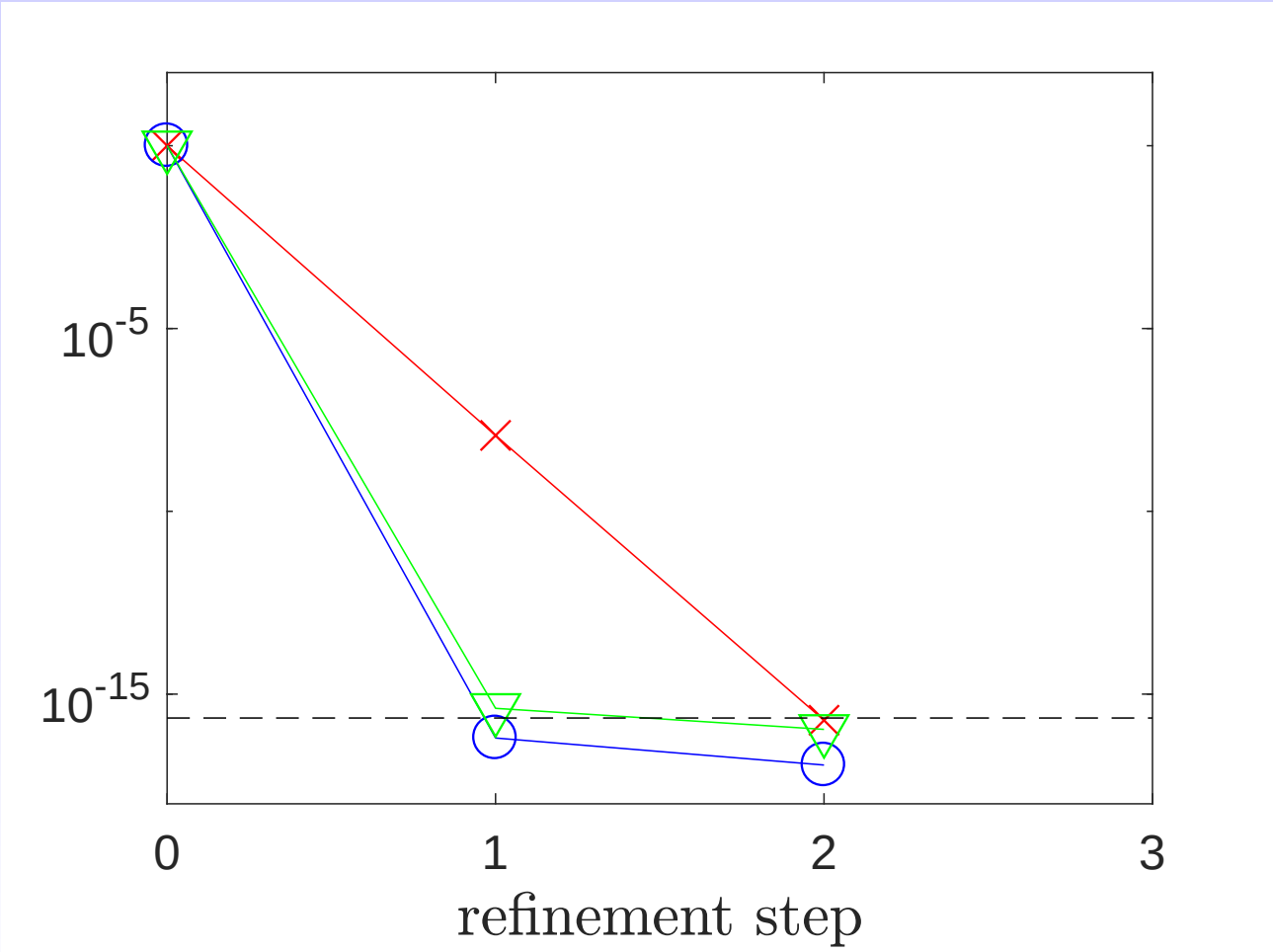


Test 3: GMRES-IR, $(u_f, u, u_r) = (H, D, Q)$

$\kappa_\infty(A) \approx 10^{12}$, $\sigma_i = \alpha^i$, GMRES (100,100)

Take $x_0 = 0$ because of overflow!

Convergence



Conclusions

- Both **low and high precision** floating-point arithmetic becoming more prevalent, in hardware and software.
- Showed that using **three precisions** in iter ref brings major benefits.
  - LU in lower precision $\Rightarrow$ twice as fast as trad. IR.
  - GMRES-IR cges when trad. IR doesn't thanks to preconditioned GMRES solved of $Ad_i = r_i$.
- **GMRES-IR** handles $\kappa_\infty(A) \approx u^{-1}$.
- Dongarra et al. (2017) have achieved 1.8x speedups on GPUs.

Erin Carson & H, **A New Analysis of Iterative Refinement and its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems**, *SIAM J. Sci. Comput.*, 39(6):A2834–A2856, 2017.

Erin Carson & H, **Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions**, MIMS EPrint 2017.24, The University of Manchester, July 2017; to appear in *SIAM J. Sci. Comput.*

References I

E. Carson and N. J. Higham.
Accelerating the solution of linear systems by iterative refinement in three precisions.
MIMS EPrint 2017.24, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, July 2017.
30 pp.
Revised January 2018. To appear in SIAM J. Sci. Comput.

References II

E. Carson and N. J. Higham.
A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems.
SIAM J. Sci. Comput., 39(6):A2834–A2856, 2017.

A. Haidar, P. Wu, S. Tomov, and J. Dongarra.
Investigating half precision arithmetic to accelerate dense linear system solvers.
In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA '17, pages 10:1–10:8, Nov. 2017.

References III

N. J. Higham.
Iterative refinement for linear systems and LAPACK.
IMA J. Numer. Anal., 17(4):495–509, 1997.

N. J. Higham.
Accuracy and Stability of Numerical Algorithms.
Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.
ISBN 0-89871-521-0.
xxx+680 pp.

Y. Kobayashi and T. Ogita.
A fast and efficient algorithm for solving ill-conditioned linear systems.
JSIAM Lett., 7:1–4, 2015.

References IV

J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra.
Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems).
In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, Nov. 2006.