MANCHESTER 1824

The University of Manchester

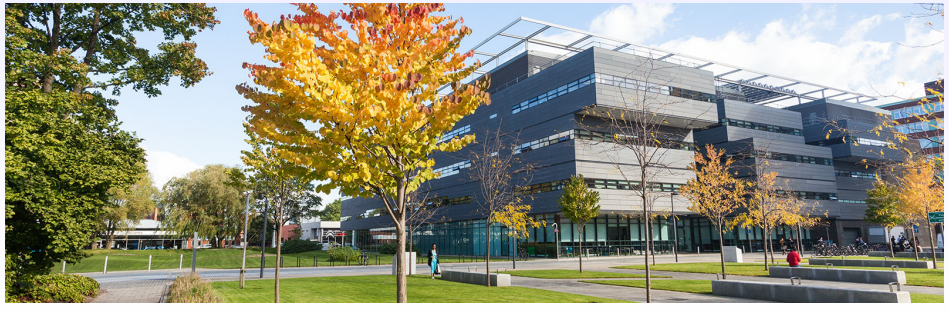
M\cr NA
Manchester Numerical Analysis

The Rise of Multiprecision Computations

Nick Higham
School of Mathematics
The University of Manchester

`http://www.ma.man.ac.uk/~higham`
`@nhigham, nickhigham.wordpress.com`

24th IEEE Symposium on Computer Arithmetic, London, July 24–26, 2017



Outline

Multiprecision arithmetic: floating point arithmetic supporting multiple, possibly arbitrary, precisions.

- Applications of & support for low precision.
- Applications of & support for high precision.
- How to exploit different precisions to achieve faster algs with higher accuracy.
- Focus on **iterative refinement** for $Ax = b$.

Download this talk from
http://bit.ly/higham-arith24

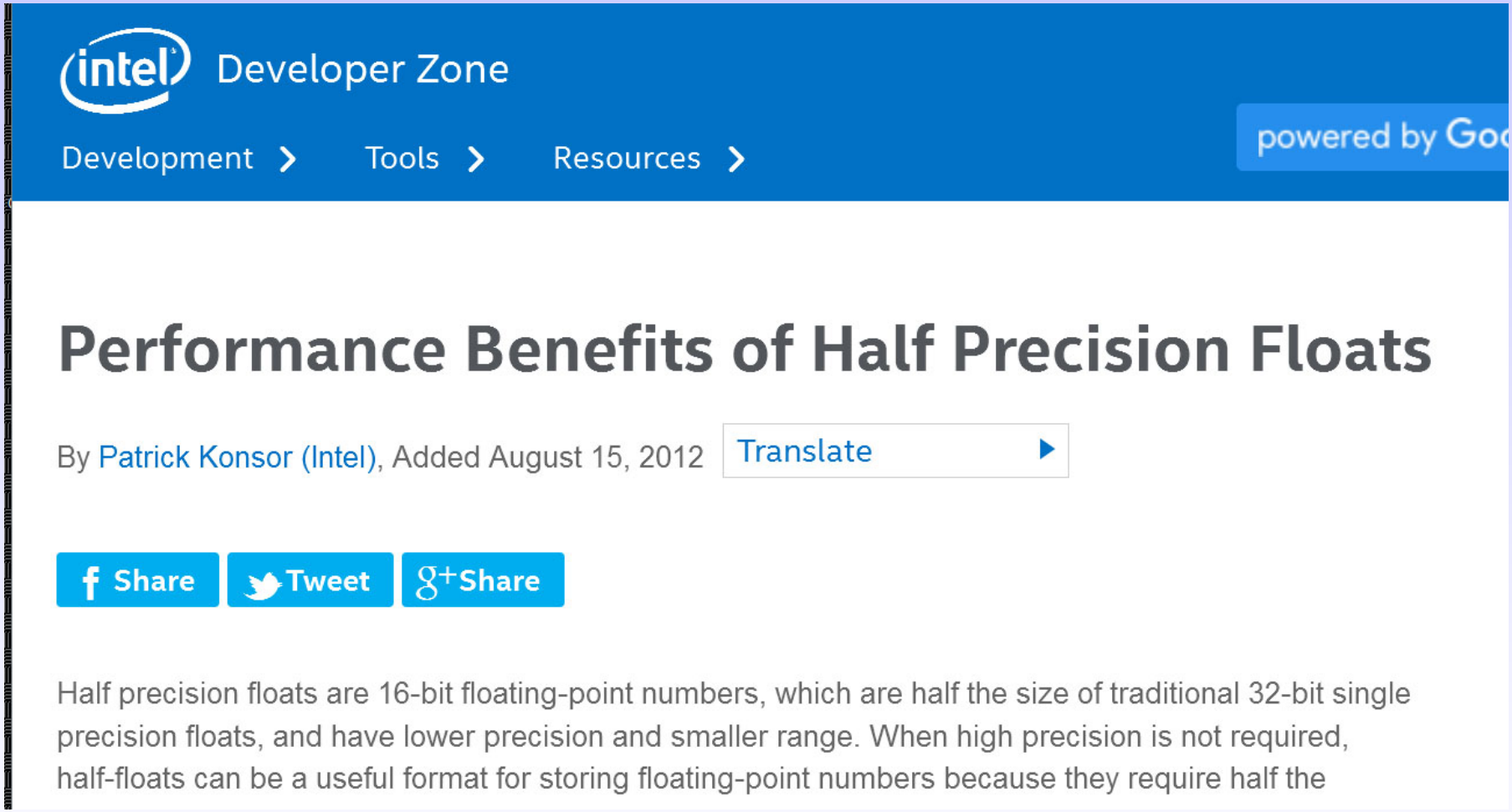
IEEE Standard 754-1985 and 2008 Revision

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	$10^{\pm 4932}$	$2^{-113} \approx 9.6 \times 10^{-35}$

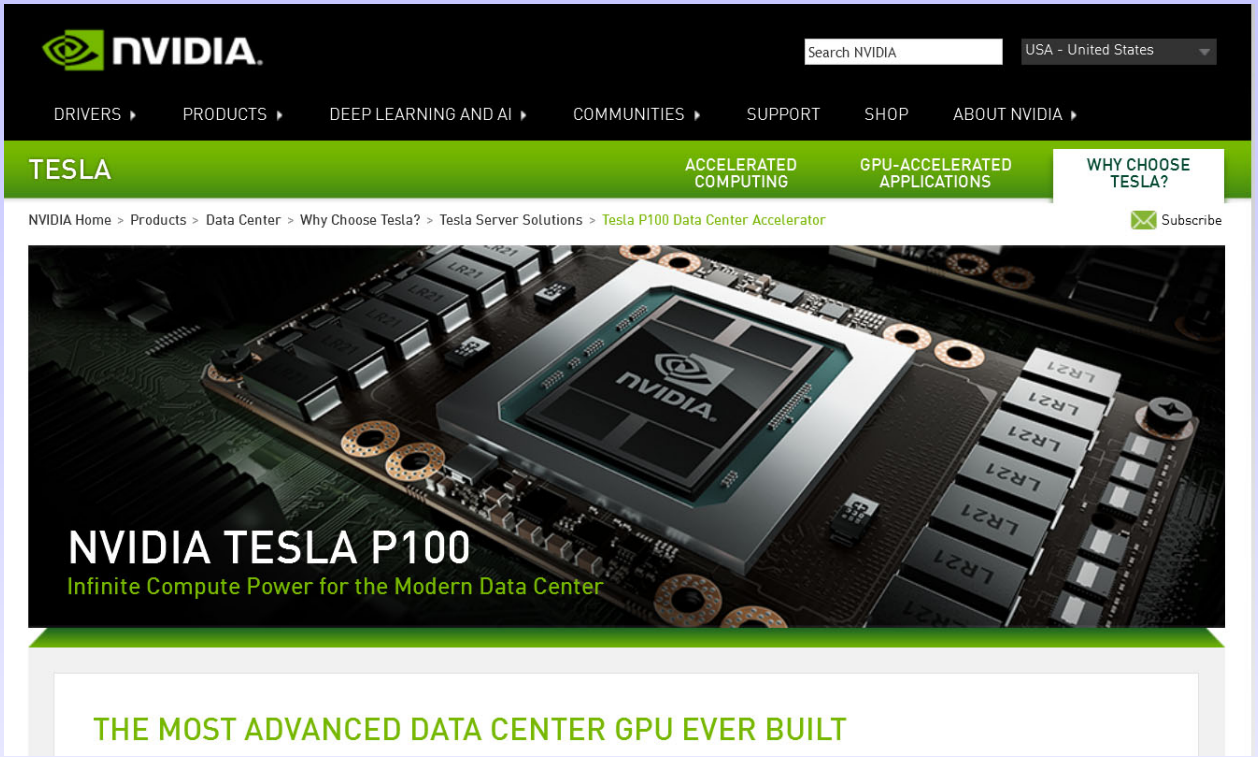
- Arithmetic ops $(+, -, *, /, \sqrt{})$ performed *as if* first calculated to infinite precision, then rounded.
- Default: round to nearest, round to even in case of tie.
- Half precision is a *storage format only*.

Intel Core Family (3rd gen., 2012)

Ivy Bridge supports half precision for storage.

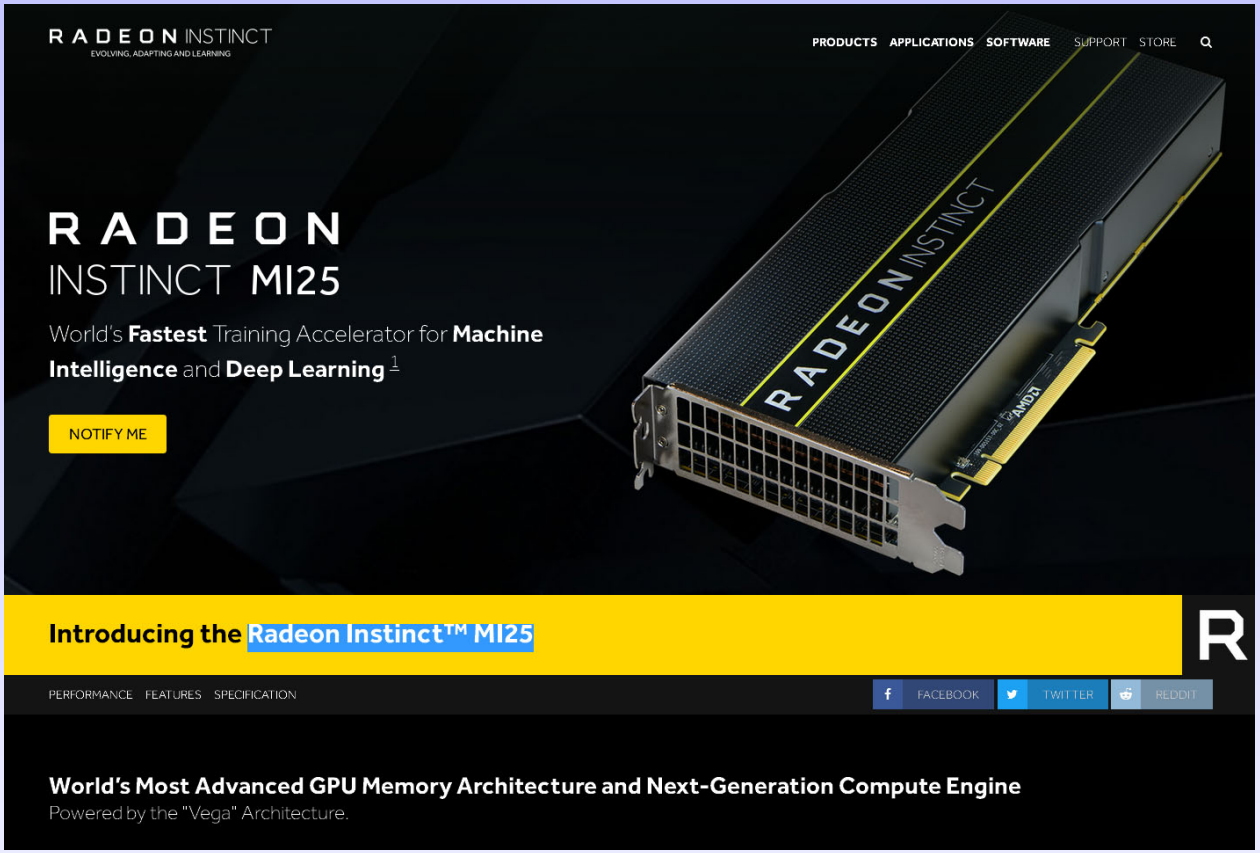


NVIDIA Tesla P100 (2016)



"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"

Japan to Build 130 Petaflop ABCI Supercomputer

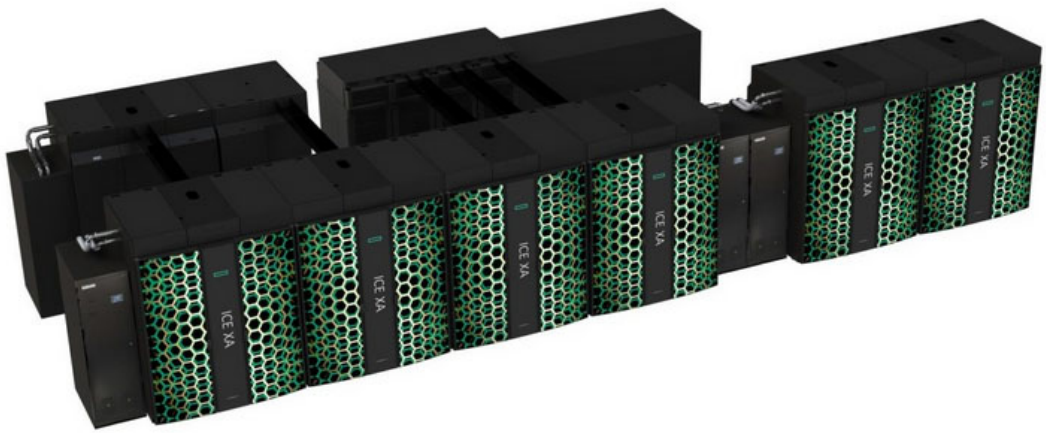
November 25, 2016 by Rich Brueckner

Today Japan announced plans to build a 130 Petaflop (half or single precision) supercomputer for deployment in 2017. And while such a machine would not surpass the current #1 93 Petaflop Sunway TaihuLight supercomputer in China, it would certainly propel Japan to the top of an all new category of supercomputing leadership.

TSUBAME 3.0 (HPC Wire, Feb 16, 2017)

In a press event Friday afternoon local time in Japan, Tokyo Institute of Technology (Tokyo Tech) announced its plans for the TSUBAME3.0 supercomputer, which will be Japan's "fastest AI supercomputer," when it comes online this summer (2017). Projections are that it will deliver 12.2 double-precision petaflops and 64.3 half-precision (peak specs).

Nvidia was the first vendor to publicly share the news in the US. We know that Nvidia will be supplying Pascal P100 GPUs, but the big surprise here is the system vendor. The Nvidia blog did not specifically mention HPE or SGI but it did include this photo with a caption referencing it as TSUBAME3.0:

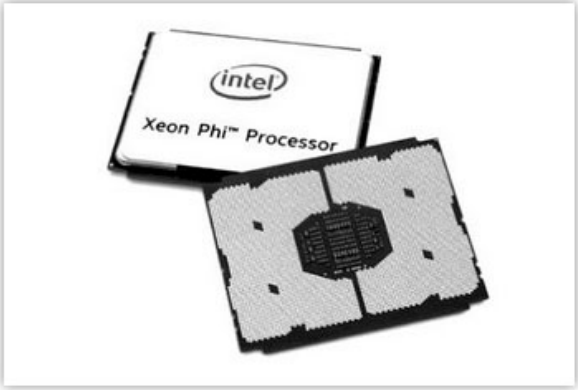


TSUBAME3.0 – *click to expand* (Source: Nvidia)

"The Knights Mill will get at least a 2-4X speedup for deep learning workloads thanks to new instructions that provide optimizations for *single*, *half* and *quarter-precision* . . . Knights Mill uses different instruction sets to *improve lower-precision performance* at the expense of the double-precision performance."

WHY INTEL IS TWEAKING XEON PHI FOR DEEP LEARNING

August 22, 2016 Timothy Prickett Morgan



If there is anything that chip giant Intel has learned over the past two decades as it has gradually climbed to dominance in processing in the datacenter, it is ironically that one size most definitely does not fit all. Quite the opposite, and increasingly so.

As the tight co-design of hardware and software continues in all parts of the IT industry, we can expect fine-grained customization for very precise – and lucrative – workloads, like data analytics and machine learning, just to name two of the hottest areas today.

"for machine learning as well as for certain image processing and signal processing applications, *more data at lower precision actually yields better results with certain algorithms than a smaller amount of more precise data*."

Google Tensorflow Processor

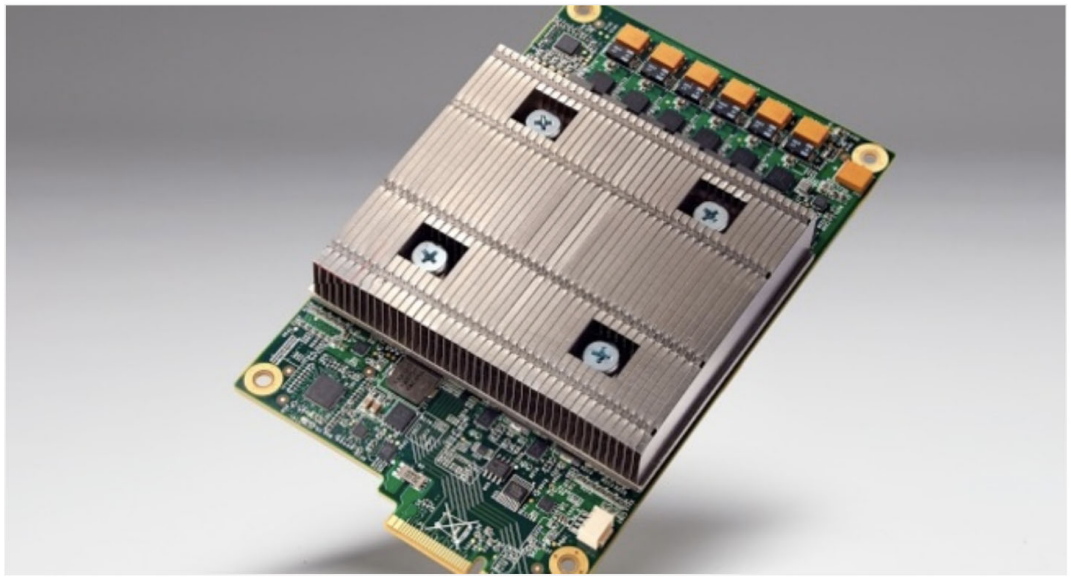
The Register®
Biting the hand that feeds IT

DATA CENTER SOFTWARE SECURITY TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH

Emergent Tech ▸ Artificial Intelligence

Revealed: Blueprints to Google's AI FPU aka the Tensor Processing Unit

PCIe-connected super-calculator trounces outdated competition



"The TPU is special-purpose hardware designed to accelerate the inference phase in a neural network, in part through quantizing 32-bit floating point computations into **lower-precision 8-bit arithmetic**."

Machine Learning

Courbariaux, Benji & David (2015)

> We find that very low precision is sufficient not just for running trained networks but also for training them.

- We are solving the wrong problem anyway (Scheinberg, 2016), so don't need an accurate solution.
- Low precision provides regularization.
- See Jorge Nocedal's plenary talk **Stochastic Gradient Methods for Machine Learning** at SIAM CSE 2017.

Climate Modelling

- **T. Palmer**, *More reliable forecasts with less precise computations: a fast-track route to cloud-resolved weather and climate simulators?*, Phil. Trans. R. Soc. A, 2014:

  > Is there merit in representing variables at sufficiently high wavenumbers using half or even quarter precision floating-point numbers?

- **T. Palmer**, *Build imprecise supercomputers*, Nature, 2015.

Need for Higher Precision

- He and Ding, **Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Parallel Applications**, 2001.
- Bailey, Barrio & Borwein, **High-Precision Computation: Mathematical Physics & Dynamics**, 2012.
- Khanna, **High-Precision Numerical Simulations on a CUDA GPU: Kerr Black Hole Tails**, 2013.
- Beliakov and Matiyasevich, **A Parallel Algorithm for Calculation of Determinants and Minors Using Arbitrary Precision Arithmetic**, 2016.
- Ma and Saunders, **Solving Multiscale Linear Programs Using the Simplex Method in Quadruple Precision**, 2015.

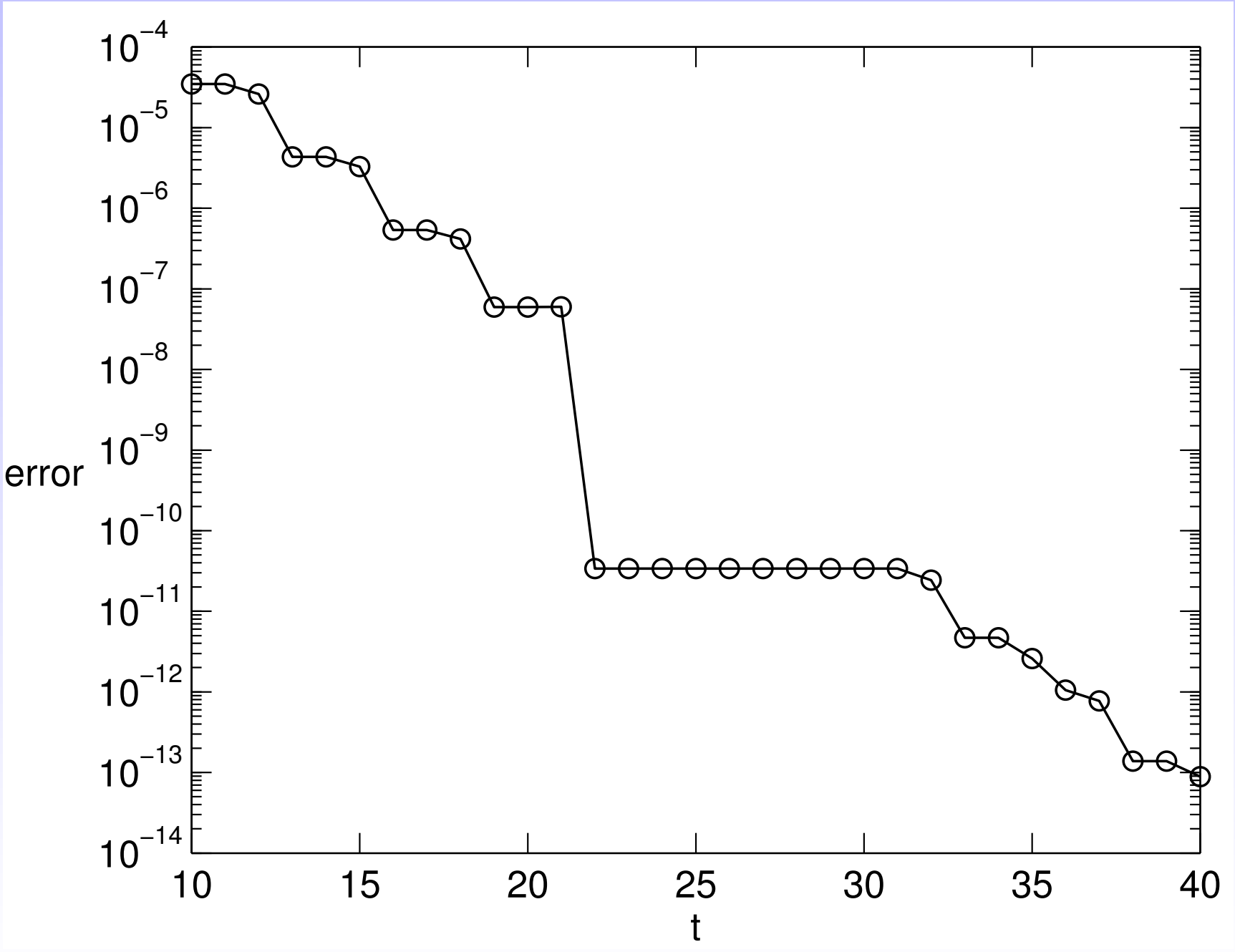
Increasing the Precision

Myth

Increasing the precision at which a computation is performed increases the accuracy of the answer.

Consider the evaluation in precision $u = 2^{-t}$ of

$$y = x + a\sin(bx), \quad x = 1/7, \quad a = 10^{-8}, \quad b = 2^{24}.$$


error
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
10
15
20
25
30
35
40
t

IBM z13 Mainframe Systems


IBM Systems MAGAZINE

AIX IBM i LINUX ON POWER POWER

Mainframe BUSINESS STRATEGY ADMINISTRATOR TRENDS STORAGE

MAINFRAME > ADMINISTRATOR > PERFORMANCE

Technical Corner

New IBM z13 Chip Improves Throughput and Analytics Performance

May 2015 | by Brian Curran, Eric Schwarz, C. Kevin Shum

z13 processor (2015) has **quadruple precision** in the vector & floating point unit.

Lichtenau, Carlough & Mueller (2016):

"designed to maximize performance for **quad precision** floating-point operations that are occurring with increased frequency on Business Analytics workloads . . .

on commercial products like ILOG and SPSS, replacing double precision operations with **quad-precision** operations in critical routines yield 18% faster convergence due to reduced rounding error.

Availability of Multiprecision in Software

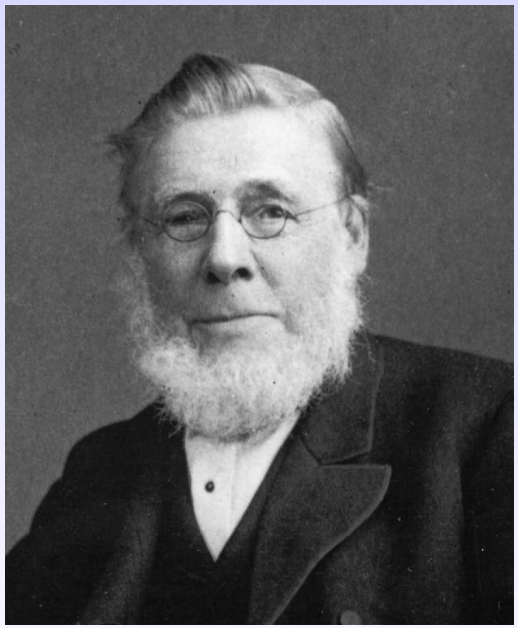
- **Maple**, Mathematica, PARI/GP, **Sage**.
- MATLAB: Symbolic Math Toolbox, **Multiprecision Computing Toolbox** (Advanpix).
- Julia: **BigFloat**.
- Mpmath and SymPy for Python.
- GNU MP Library.
- **GNU MPFR Library**.
- (Quad only): some C, Fortran compilers.

Gone, but not forgotten:

- Numerical Turing: **Hull et al.**, 1985.

Note on Log Tables

Name	Year	Range	Decimal places
R. de Prony	1801	$1 - 10,000$	19
Edward Sang	1875	$1 - 20,000$	28

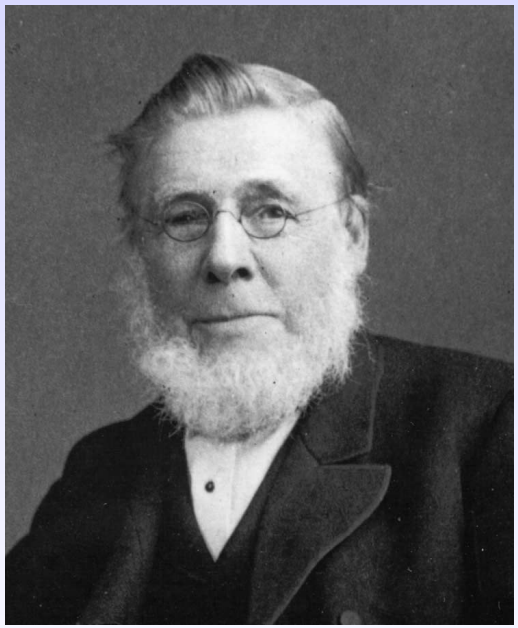


Age 82

Edward Sang (1805–1890).
Born in Kirkcaldy.
Teacher of maths and actuary in Edinburgh.

Note on Log Tables

Name	Year	Range	Decimal places
R. de Prony	1801	$1 - 10,000$	19
Edward Sang	1875	$1 - 20,000$	28



Age 82

Edward Sang (1805–1890).
Born in Kirkcaldy.
Teacher of maths and actuary in Edinburgh.

It's better to be approximately right than precisely wrong.

Going to Higher Precision

If we have quadruple or higher precision, how can we modify existing algorithms to exploit it?

Going to Higher Precision

If we have quadruple or higher precision, how can we modify existing algorithms to exploit it?

To what extent are existing algs precision-independent?

- Newton-type algs: just decrease `tol`?
- How little higher precision can we get away with?
- Gradually increase precision through the iterations?
- Or decrease precision through the iterations for Krylov methods?

Matrix Functions

(Inverse) scaling and squaring-type algorithms for e^A, $\log A$, $\cos A$, A^t use Padé approximants.

- Padé degree and algorithm parameters chosen to achieve double precision accuracy, $u = 2^{-53}$.
- Change u and the algorithm logic needs changing!
- **H & Fasi**, 2017: **Multiprecision Algorithms for Computing the Matrix Logarithm**.

- *Open questions*, even for scalar elementary functions?

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** 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**
- $x_1 = \mathrm{fl}(x_0 + d)$ **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 $Ad = r$
- $x_1 = \mathrm{fl}(x_0 + d)$

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**
- $x_1 = \mathrm{fl}(x_0 + d)$ **double precision**

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

Iterative Refinement in Three Precisions

Joint work with **Erin Carson** (NYU).

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.
- **Langou et al.** (2006): lower precision LU.

All the above require support **at most two precisions** and require $\kappa(A)u < 1$.

New Analysis

Assume computed solution to $Ad_i = r_i$ satisfies

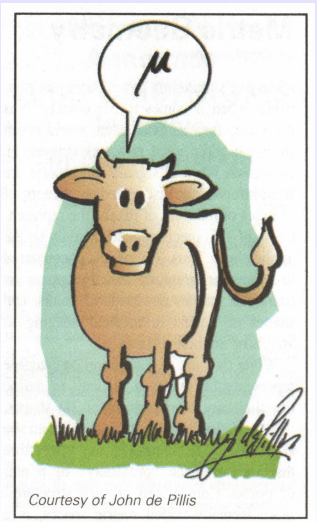
$$\frac{\|d_i - \widehat{d}_i\|_\infty}{\|d_i\|} \leq u_f \theta_i < 1.$$

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

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

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

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

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

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

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 $\widehat{x}$ is produced for which*

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

Analogous standard bound would have

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

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 (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**

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

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

$$\underbrace{\widehat{U}^{-1}\widehat{L}^{-1}A}_{\widetilde{A}}\, d_i = \widehat{U}^{-1}\widehat{L}^{-1} r_i.$$

- $\widetilde{A}$ 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**.

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	S	D	D	10^8	D	D	$\mathrm{cond}(A, x) \cdot D$
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	S	D	D	10^8	D	D	$\mathrm{cond}(A,x)\cdot D$
LU	S	D	Q	10^8	D	D	D
GMRES-IR	S	D	Q	10^{16}	D	D	D
GMRES-IR	H	D	Q	10^{12}	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	S	D	D	10^8	D	D	$\mathrm{cond}(A,x)\cdot D$
LU	S	D	Q	10^8	D	D	D
GMRES-IR	S	D	Q	10^{16}	D	D	D
GMRES-IR	H	D	Q	10^{12}	D	D	D

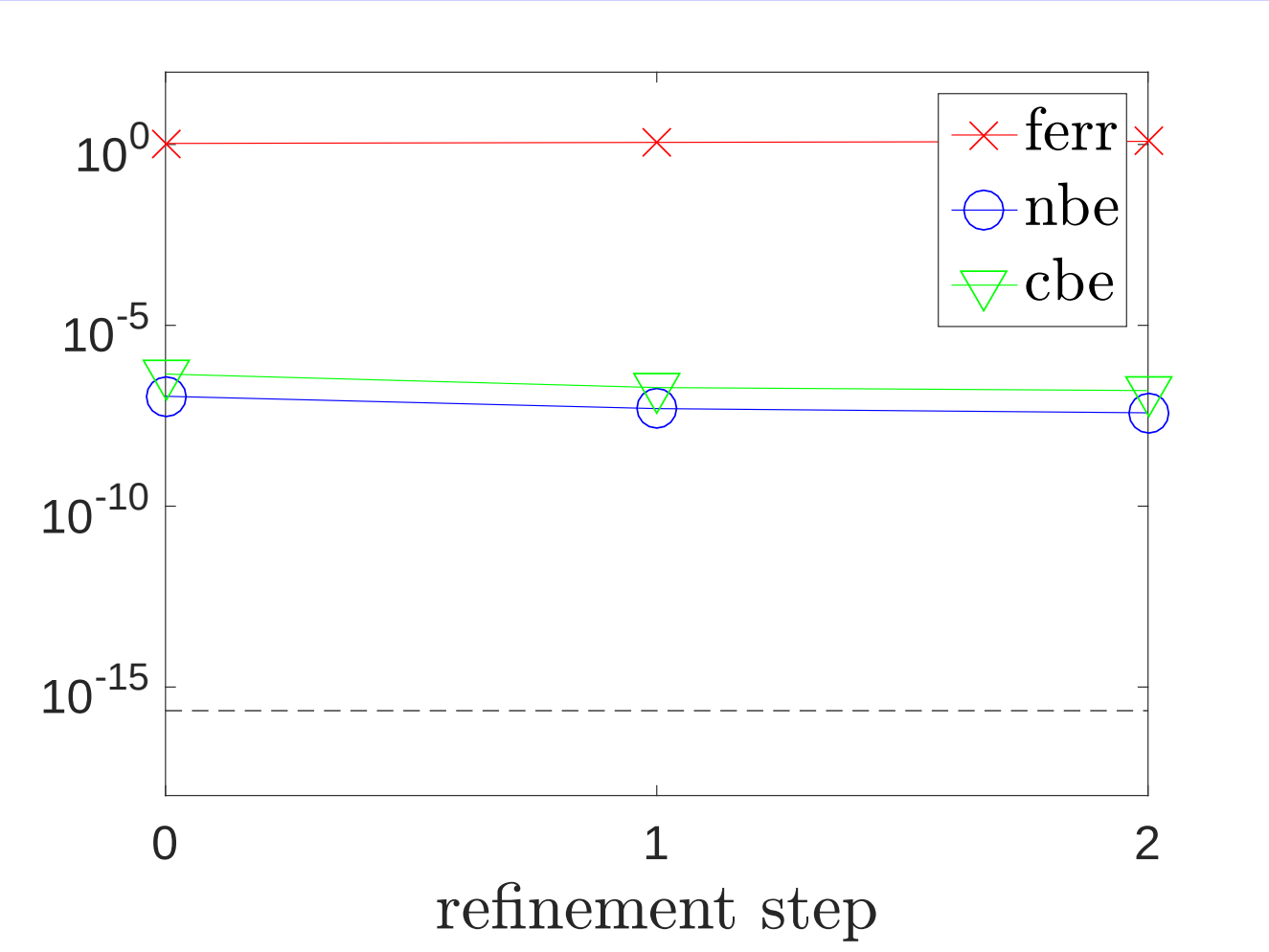
Do we have a bounded number of GMRES iterations?

Some tests with 100×100 matrices ...

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

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

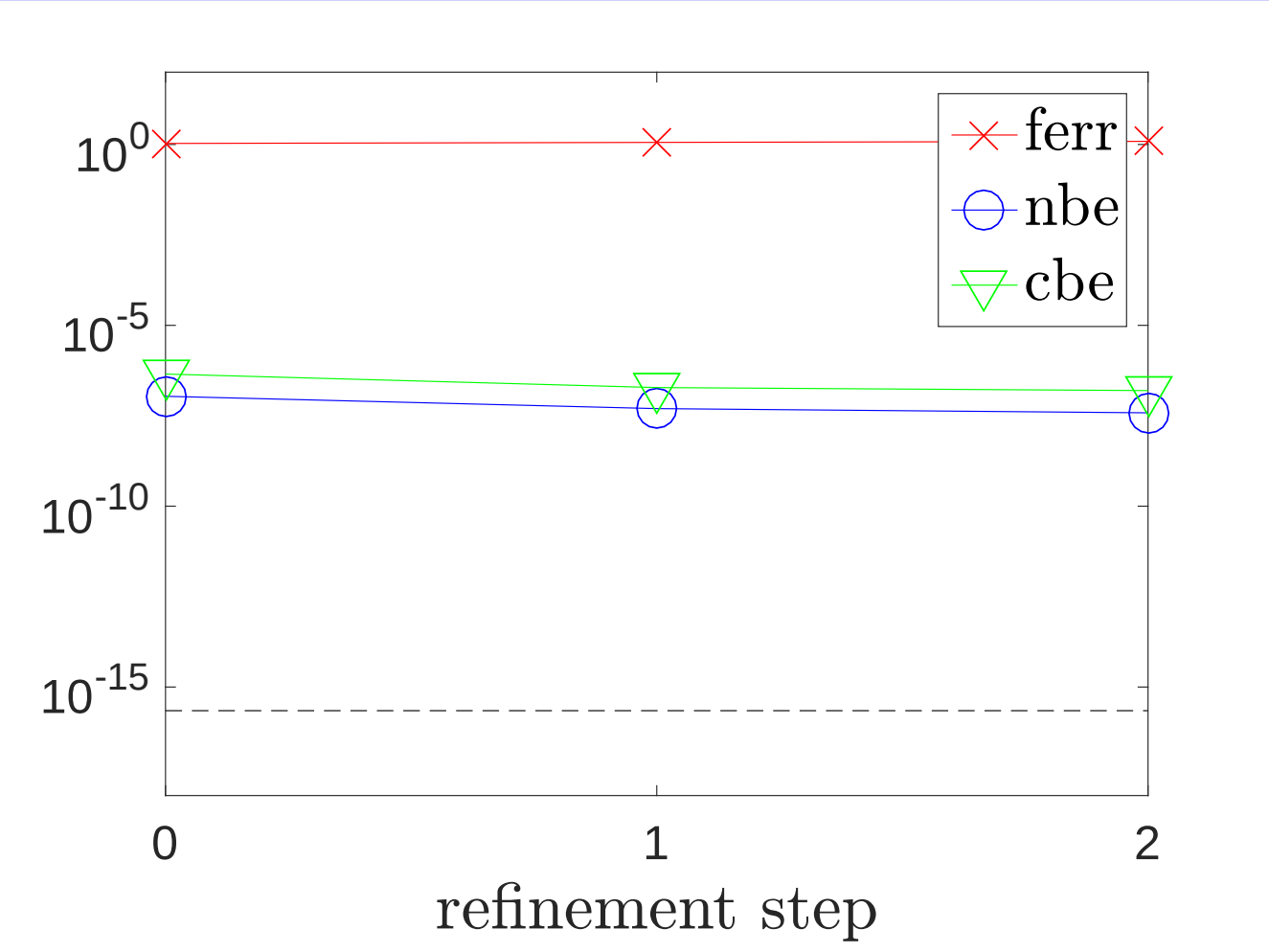
Divergence



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

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

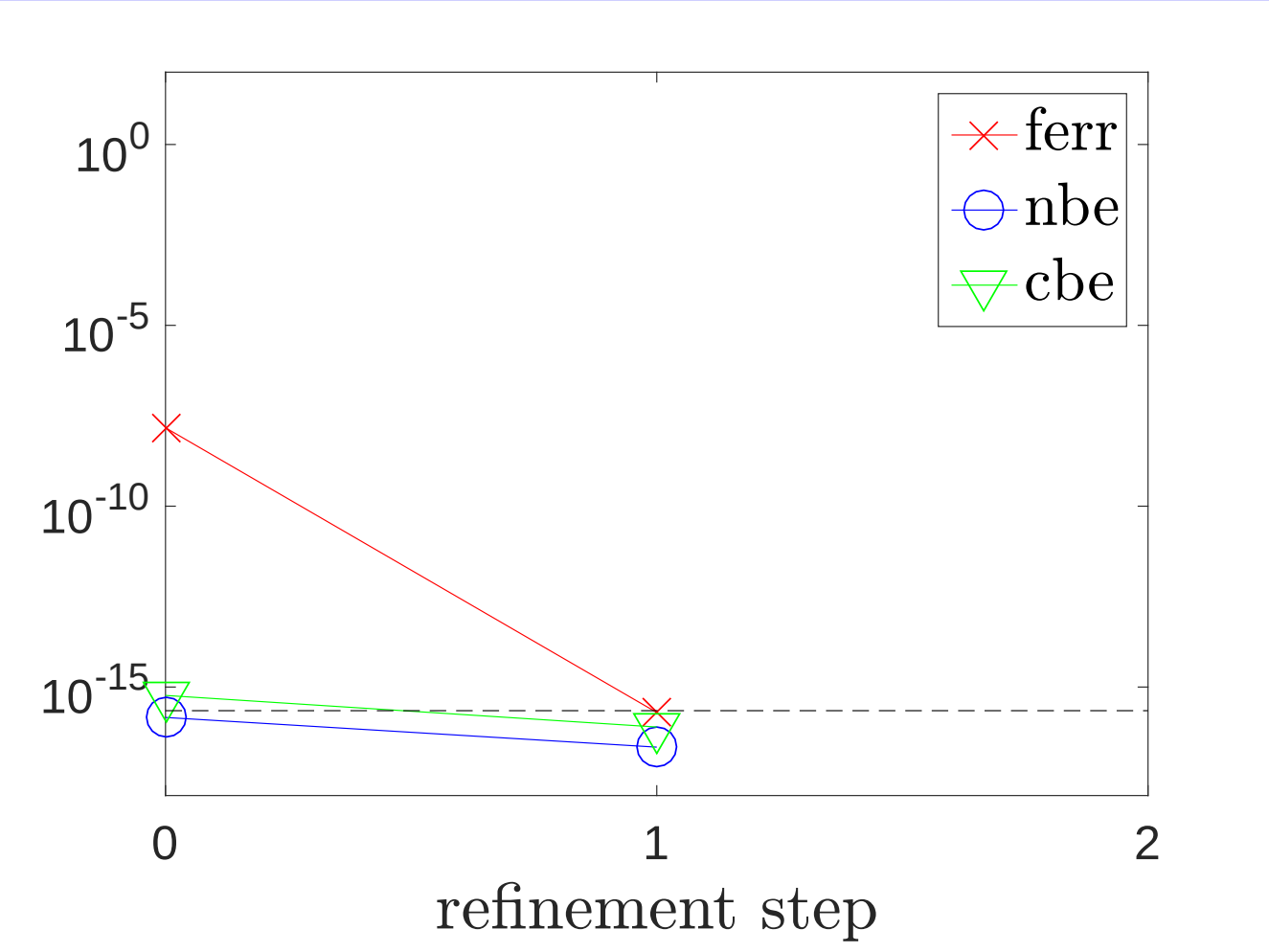
Divergence



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

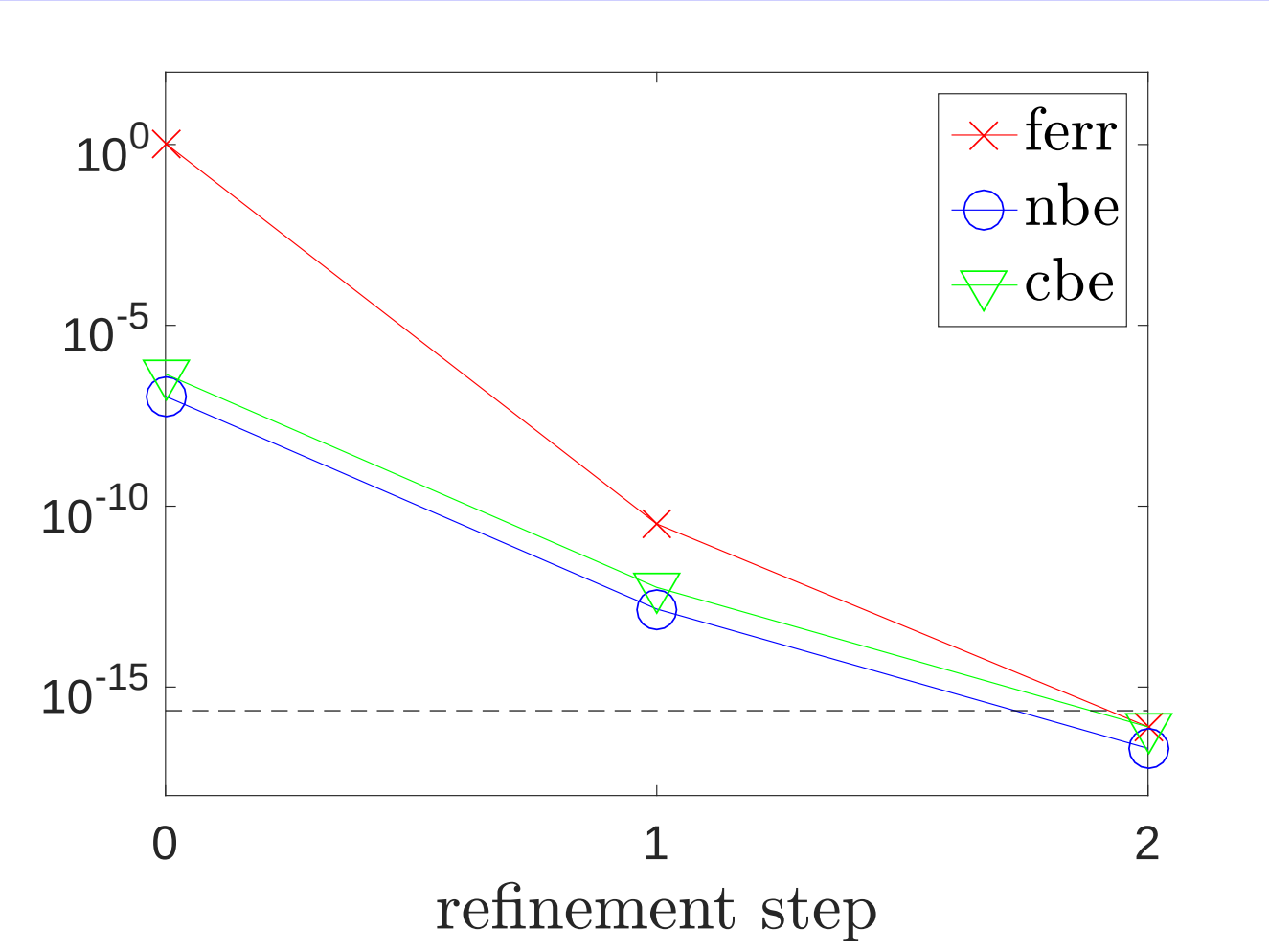
$\kappa_\infty(A) \approx 10^{10}$, $\sigma_i = \alpha^i$

Convergence



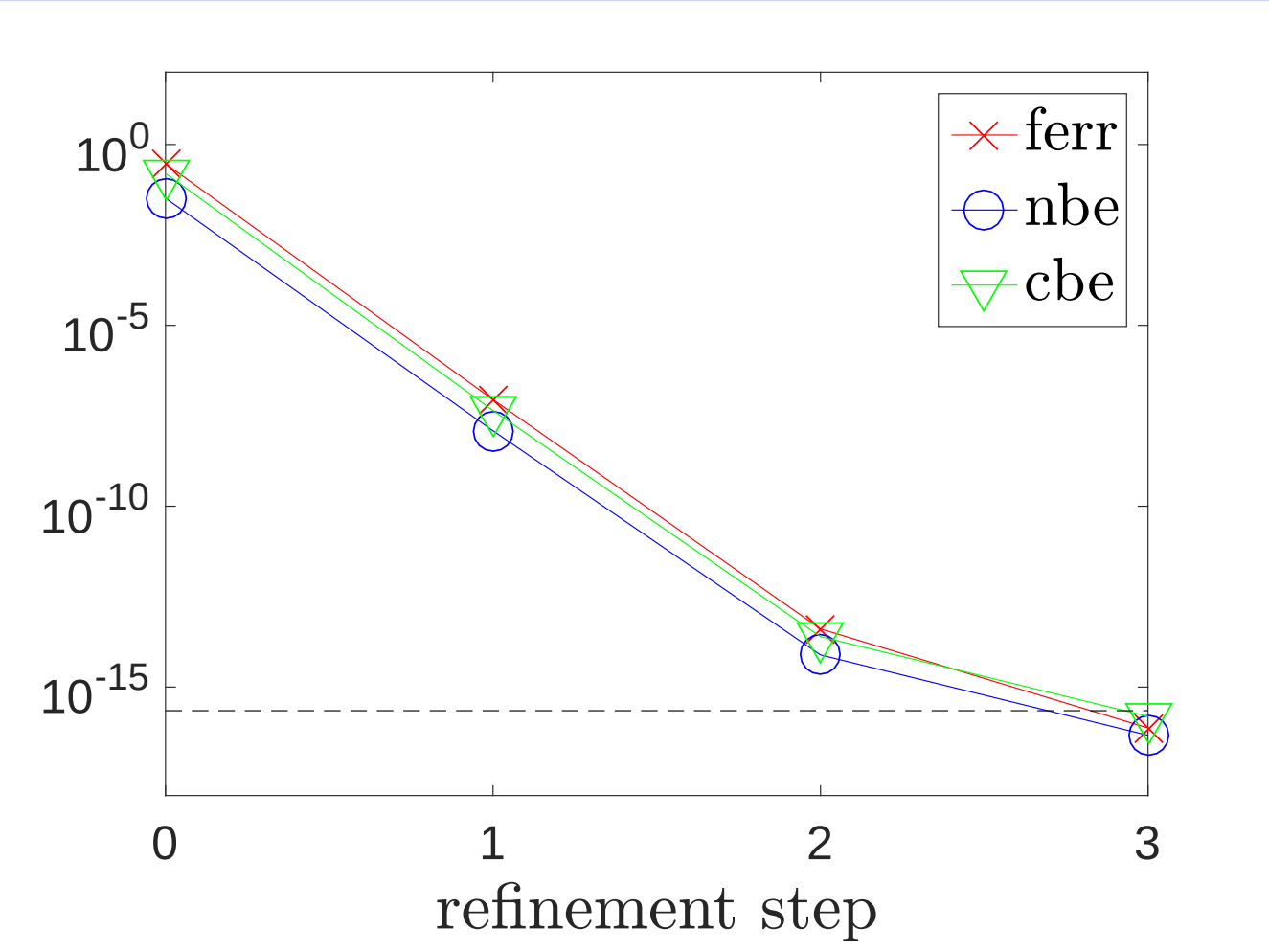
Test 1: GMRES-IR, $(u_f, u, u_r) = (S, D, Q)$

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

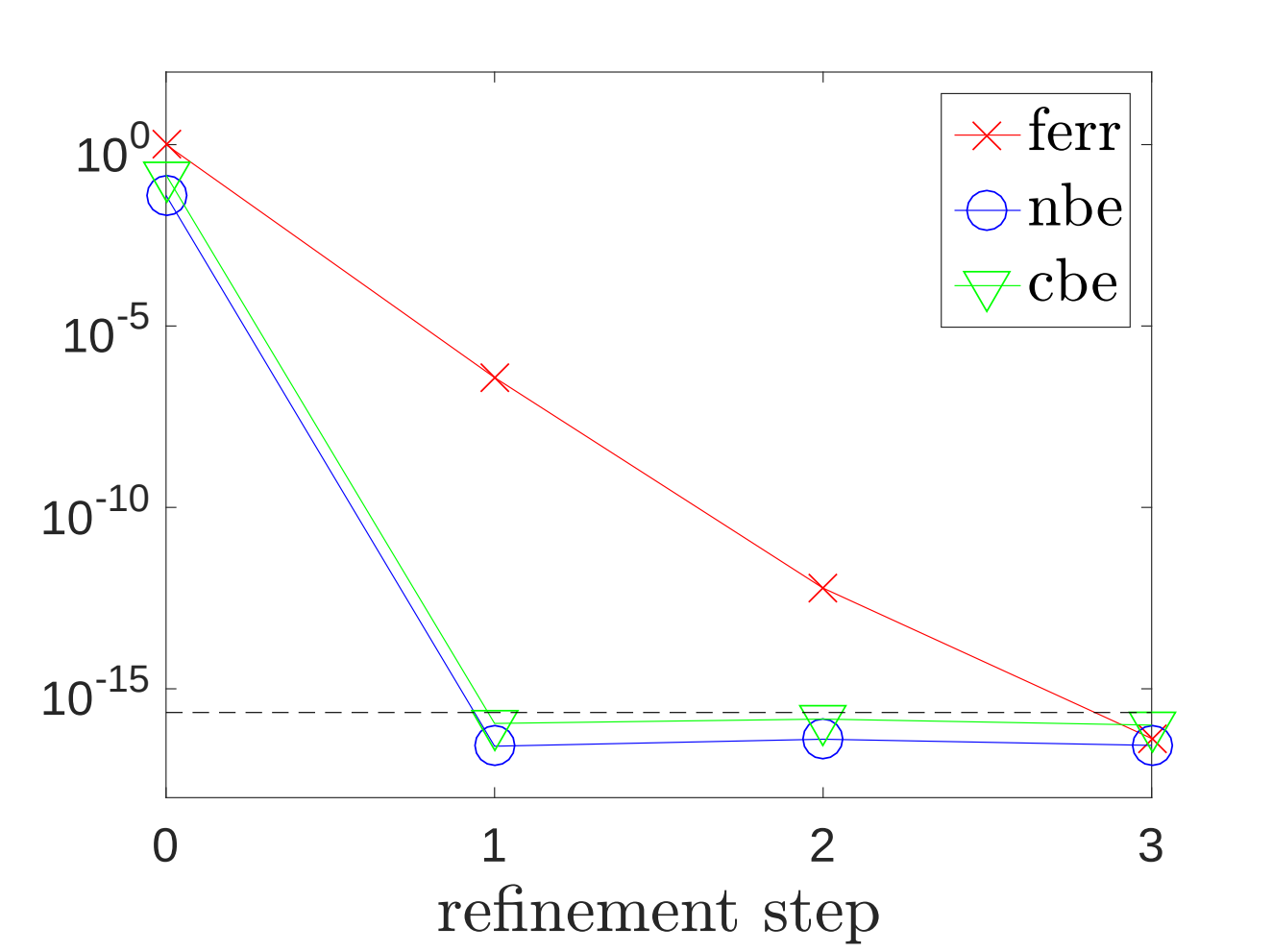


Test 3: GMRES-IR, $(u_f, u, u_r) = (H, D, Q)$

$\kappa_\infty(A) \approx 10^{12}$, 1 small σ_i, GMRES (17,19,19)

$x_0 = 0$ here due to overflow!

Convergence



Conclusions

- Both **low and high precision** floating-point arithmetic becoming more prevalent, in hardware and software.
- Need **better understanding** of behaviour of algs in low precision arithmetic. *Lower energy usage* is a driver.
- Judicious use of **a little high precision** can bring major benefits.
- Showed that using **three precisions** in iter ref brings major benefits and permits **faster** and **more accurate** solution of $Ax = b$.
- **GMRES-IR** handles $\kappa_\infty(A) \approx u^{-1}$. Further work: tune cgce tol, alternative preconditioners etc.

Erin Carson and & H (2017), **A New Analysis of Iterative Refinement and its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems**, MIMS EPrint 2017.12, The University of Manchester, March 2017; under revision for *SIAM J. Sci. Comput.*

Erin Carson and & H (2017), **Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions**, MIMS EPrint 2017.24, The University of Manchester, July 2017.

References I

D. H. Bailey, R. Barrio, and J. M. Borwein.
High-precision computation: Mathematical physics and dynamics.
Appl. Math. Comput., 218(20):10106–10121, 2012.

G. Beliakov and Y. Matiyasevich.
A parallel algorithm for calculation of determinants and minors using arbitrary precision arithmetic.
BIT, 56(1):33–50, 2015.

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.
MIMS EPrint 2017.12, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Mar. 2017.
23 pp.
Revised July 2017. Submitted to SIAM J. Sci. Comput.

M. Courbariaux, Y. Bengio, and J.-P. David.
Training deep neural networks with low precision multiplications, 2015.
ArXiv preprint 1412.7024v5.

References III


A. D. D. Craik.
The logarithmic tables of Edward Sang and his daughters.
Historia Mathematica, 30(1):47–84, 2003.


Y. He and C. H. Q. Ding.
Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications.
J. Supercomputing, 18(3):259–277, 2001.


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

References IV

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.

G. Khanna.
High-precision numerical simulations on a CUDA GPU: Kerr black hole tails.
J. Sci. Comput., 56(2):366–380, 2013.

References V

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.

C. Lichtenau, S. Carlough, and S. M. Mueller.
Quad precision floating point on the IBM z13.
In *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, pages 87–94, July 2016.

References VI

D. Ma and M. Saunders.
Solving multiscale linear programs using the simplex method in quadruple precision.
In M. Al-Baali, L. Grandinetti, and A. Purnama, editors, *Numerical Analysis and Optimization*, number 134 in Springer Proceedings in Mathematics and Statistics, pages 223–235. Springer-Verlag, Berlin, 2015.

K. Scheinberg.
Evolution of randomness in optimization methods for supervised machine learning.
SIAG/OPT Views and News, 24(1):1–8, 2016.