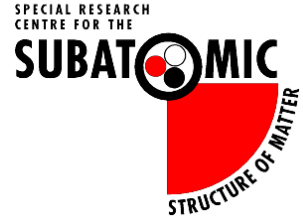
Adventures in Single Precision on the GTX 580

Waseem Kamleh

University of Adelaide

Lattice 2012, Cairns, Australia



Desert Country

- Australia is famous for its vast inland desert.

Desert Country

COUNTRY	Top 20	Top 100	Top 200	Top 500
USA	17	53	89	151
UK	3	10	19	37
Germany	0	6	14	39
Japan	0	5	9	23
Canada	0	4	8	22
Australia	0	4	7	19

Table: Top 6 countries, as ranked by the Academic Ranking of World Universities (ARWU) in 2011. Also know as the Shanghai Ranking.

`http://www.shanghairanking.com/`

Desert Country

Country	Top500	Total Cores	R_{max}	R_{peak}
Japan	15	379274	5380.2	7669.0
USA	26	488037	4128.1	5341.6
Germany	8	260920	3986.1	4666.4
UK	7	228000	2299.4	2899.3
Canada	6	90576	710.7	935.7
Australia	2	21536	213.6	247.4

Table: Selected statistics from the Academic segment of the June 2012 Supercomputing Top 500. Data are shown for the Top 6 ARWU countries only. R_{max} and R_{peak} are in Tflops.

`http://www.top500.org/`

Desert Country

COUNTRY	TOP 500	TOTAL R_{max}	ARWU 500	500/500
Japan	15	5380.2	23	233.9
Germany	8	3986.1	39	102.2
UK	7	2299.4	37	62.1
Canada	6	710.7	22	32.3
USA	26	4128.1	151	27.3
Australia	2	213.6	19	11.2

Table: Data of interest for the selected countries. Listed are the number of Top 500 entries, the combined Teraflops under the Academic segment, the number of ARWU 500 entries and our proposed 500/500 measure of Academic HPC resources.

500/500 Scores by Country

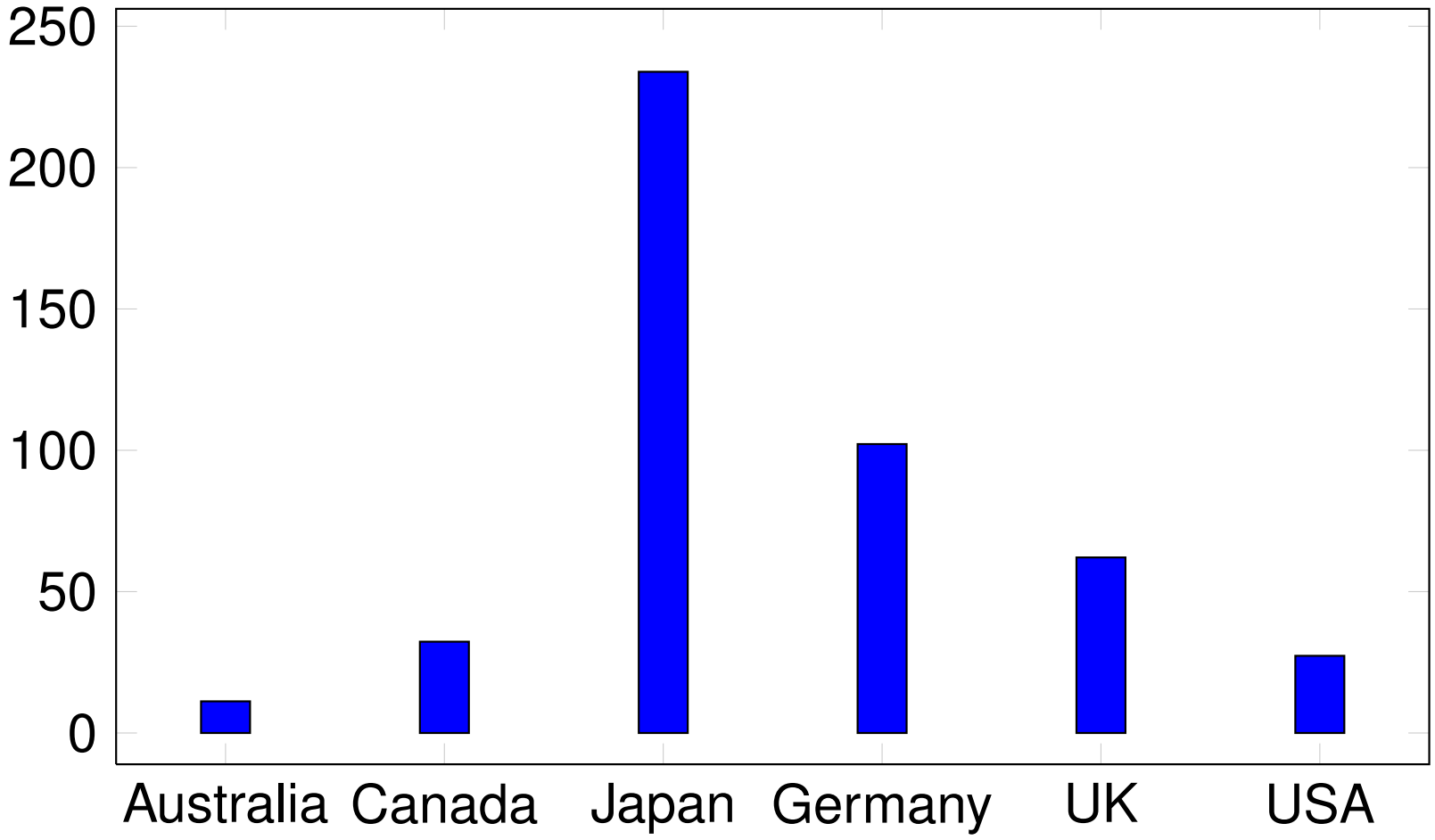


Figure: The 500/500 scores for the selected countries.

Where the bloody hell are you?

Where the bloody hell are you?

Where the bloody hell are you?

Where the bloody hell are you?

GPGPU Computing

- General purpose computing on Graphics Processing Units.
- 2006: Egri et al. *Lattice QCD as a Video Game* http://arxiv.org/abs/hep-lat/0611022
- NVIDIA CUDA.
- Cost-effective alternative to CPU clusters.
- Tesla vs GTX?

GPGPU Computing

- General purpose computing on Graphics Processing Units.
- 2006: Egri et al. *Lattice QCD as a Video Game* http://arxiv.org/abs/hep-lat/0611022
- NVIDIA CUDA.
- Cost-effective alternative to CPU clusters.
- Tesla vs GTX?

GPGPU Computing

- General purpose computing on Graphics Processing Units.
- 2006: Egri et al. *Lattice QCD as a Video Game* http://arxiv.org/abs/hep-lat/0611022
- NVIDIA CUDA.
- Cost-effective alternative to CPU clusters.
- Tesla vs GTX?

GPGPU Computing

- General purpose computing on Graphics Processing Units.
- 2006: Egri et al. *Lattice QCD as a Video Game* http://arxiv.org/abs/hep-lat/0611022
- NVIDIA CUDA.
- Cost-effective alternative to CPU clusters.
- Tesla vs GTX?

GPGPU Computing

- General purpose computing on Graphics Processing Units.
- 2006: Egri et al. *Lattice QCD as a Video Game* http://arxiv.org/abs/hep-lat/0611022
- NVIDIA CUDA.
- Cost-effective alternative to CPU clusters.
- Tesla vs GTX?

NVIDIA Tesla

SELECT THE RIGHT TESLA GPU

Features	Tesla K10	Tesla M2090	Tesla M2075	Tesla M2070-Q
Number and Type of GPU	2 Kepler GK104s	1 Fermi GPU	1 Fermi GPU	1 Fermi GPU
GPU Computing Applications	Seismic processing, signal and image processing, video analytics	Seismic processing, CFD, CAE, Financial computing, Computational chemistry and Physics, Data analytics, Satellite imaging, Weather modeling		
Visualization Applications	N/A	N/A	N/A	CAD, CAM, CAE pre/post processing Remote desktop
Peak double precision floating point performance	190 Gigaflops (95 Gflops per GPU)	665 Gigaflops	515 Gigaflops	515 Gigaflops
Peak single precision floating point performance	4577 Gigaflops (2288 Gflops per GPU)	1331 Gigaflops	1030 Gigaflops	1030 Gigaflops
Memory bandwidth (ECC off)	320 GB/sec (160 GB/sec per GPU	177 GBytes/sec	150 GBytes/sec	150 GBytes/sec
Memory size (GDDR5)	8GB (4 GB per GPU)	6 GigaBytes	6 GigaBytes	6 GigaBytes
CUDA cores	3072 (1536 per GPU)	512	448	448

* Note: With ECC on, 12.5% of the GPU memory is used for ECC bits. For example, 6 GB total memory yields 5.25 GB of user available memory with ECC on.

Figure: http://www.nvidia.com/object/tesla-servers.html

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - $1 \times$ Tesla
  - $4 \times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - 1 $\times$ Tesla
  - 4 $\times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - $1 \times$ Tesla
  - $4 \times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - $1 \times$ Tesla
  - $4 \times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - 1 $\times$ Tesla
  - 4 $\times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - 1 $\times$ Tesla
  - 4 $\times$ GTX
- How badly do we need double precision?

Tesla vs GeForce

- GTX 580 peaks at 1581 flops in single precision.
- GTX 680 peaks at 3090 flops in single precision.
- GTX card costs 20%-25% of a Tesla card.
- Single compute node, similar cost:
  - 1 $\times$ Tesla
  - 4 $\times$ GTX
- How badly do we need double precision?

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity

    $\Longrightarrow$ double precision.
  - Can't easily "check" solution

    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$

    $\Longrightarrow$ only need single precision.
  - Can check solution easily.

    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity

    $\Longrightarrow$ double precision.
  - Can't easily "check" solution

    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$

    $\Longrightarrow$ only need single precision.
  - Can check solution easily.

    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

Single vs Double

- Gauge field generation:
  - Need to preserve unitarity
    $\Longrightarrow$ double precision.
  - Can't easily "check" solution
    $\Longrightarrow$ needs ECC memory.
- Quark propagator calculation:
  - Solution tolerance $\sim 10^{-5}$
    $\Longrightarrow$ only need single precision.
  - Can check solution easily.
    $\Longrightarrow$ don't need ECC.
- Strategy:
  - Obtain gauge fields from the ILDG.
  - Focus computational efforts on quark propagators.

PACS-CS Simulation Details

PACS-CS Collaboration: S. Aoki, et al., Phys. Rev. **D79** (2009) 034503.

- Lattice volume: $32^3 \times 64$
- Non-perturbative $\mathcal{O}(a)$-improved Wilson quark action
- Iwasaki gauge action
- $2 + 1$ flavour dynamical-fermion QCD
- $\beta = 1.9$ providing $a = 0.0907$ fm
- $K_{ud} = \{$ 0.13700, 0.13727, 0.13754, 0.13770, 0.13781 $\}$
- $K_s = 0.13640$
- Lightest pion mass is 156 MeV.
- Five ensembles of 350 configurations.
- 750 sources for lightest mass.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

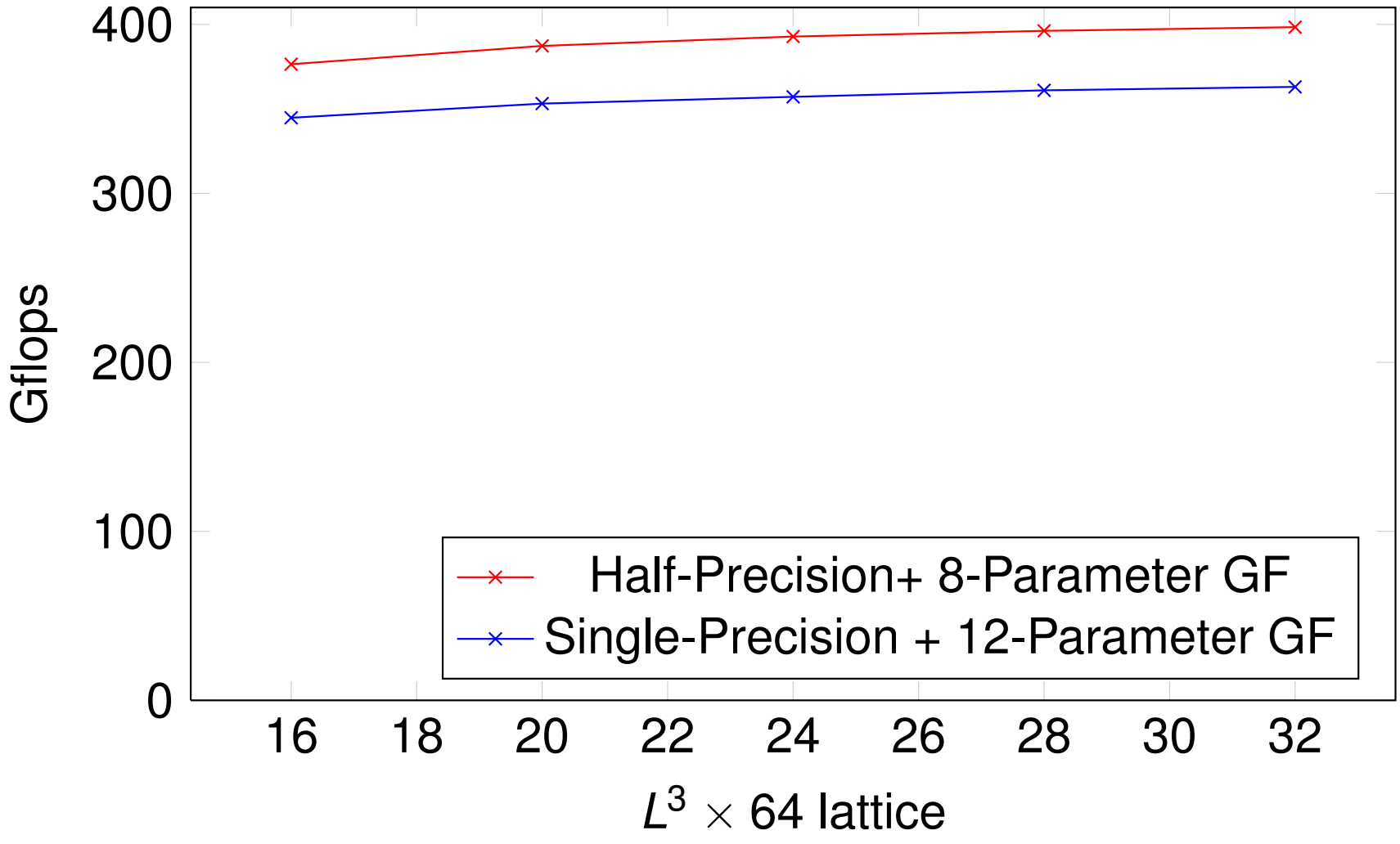
GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

GPU Matrix Kernel

- M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs*, Comput.Phys.Commun. 181 (2010)
  - 8 parameter gauge field reconstruction.
  - 12 parameter gauge field reconstruction.
  - Half-precision (FP16).
  - Fixed-precision.
  - Dirac basis: Chiral vs non-relativistic.
  - Temporal gauge fixing.
- In-house CUDA fermion matrix code benchmarks.

Clover Matrix Kernel Benchmarks



Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

Solver Choice

- PACS-CS configurations:
  - Large lattice sizes.
  - Light quark mass.
- Mixed precision:
  - Large memory footprint.
  - Half-precision too unstable.
- Minimal performance loss by using single precision.
- Solver is bandwidth-limited anyway.
- Single precision:
  - Reduced memory footprint.
  - BiCGStab is unstable.
  - Flying restarts can help a little bit.
  - Try a minimum residual solver: CGNE.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE

- Within tolerance δ_{tol}, want solution to

$$D\mathbf{x} = \mathbf{b}.$$

- Instead solve normal equation (NE):

$$D^\dagger D\mathbf{x} = D^\dagger \mathbf{b}.$$

- In double precision, NE convergence usually $\Rightarrow$ solution.
- In single precision, NE converges well before we have obtained desired solution.
- Trick: When CGNE converges with tolerance δ_{ne}, check if we have solution to within δ_{tol}.
- If not, adjust δ_{ne} and restart CGNE with current solution.

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2 \quad (\delta_{\mathrm{ne}} := \delta_{\mathrm{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\mathrm{ne}} := \mathbf{r}_{\mathrm{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\mathrm{ne}}| < \delta_{\mathrm{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\mathrm{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\mathrm{ne}} := \tau \times \delta_{\mathrm{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2 \quad (\delta_{\mathrm{ne}} := \delta_{\mathrm{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\mathrm{ne}} := \mathbf{r}_{\mathrm{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\mathrm{ne}}| < \delta_{\mathrm{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\mathrm{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\mathrm{ne}} := \tau \times \delta_{\mathrm{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2 \quad (\delta_{\mathrm{ne}} := \delta_{\mathrm{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\mathrm{ne}} := \mathbf{r}_{\mathrm{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\mathrm{ne}}| < \delta_{\mathrm{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\mathrm{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\mathrm{ne}} := \tau \times \delta_{\mathrm{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2 \quad (\delta_{\mathrm{ne}} := \delta_{\mathrm{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\mathrm{ne}} := \mathbf{r}_{\mathrm{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\mathrm{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\mathrm{ne}}| < \delta_{\mathrm{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\mathrm{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\mathrm{ne}} := \tau \times \delta_{\mathrm{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\mathrm{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\text{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\text{ne}}|^2 \quad (\delta_{\text{ne}} := \delta_{\text{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\text{ne}} := \mathbf{r}_{\text{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\text{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\text{ne}}| < \delta_{\text{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\text{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\text{ne}} := \tau \times \delta_{\text{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\text{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\text{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\text{ne}}|^2 \quad (\delta_{\text{ne}} := \delta_{\text{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\text{ne}} := \mathbf{r}_{\text{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\text{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\text{ne}}| < \delta_{\text{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\text{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\text{ne}} := \tau \times \delta_{\text{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\text{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\text{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\text{ne}}|^2 \quad (\delta_{\text{ne}} := \delta_{\text{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\text{ne}} := \mathbf{r}_{\text{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\text{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\text{ne}}| < \delta_{\text{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\text{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\text{ne}} := \tau \times \delta_{\text{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\text{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\text{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\text{ne}}|^2 \quad (\delta_{\text{ne}} := \delta_{\text{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\text{ne}} := \mathbf{r}_{\text{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\text{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\text{ne}}| < \delta_{\text{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\text{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\text{ne}} := \tau \times \delta_{\text{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\text{ne}} - \theta\mathbf{y}$ and continue (go to 2).

CGNE with Restarts

1. Set $\mathbf{y} := \mathbf{r}_{\text{ne}} := D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}, \quad \rho := |\mathbf{r}_{\text{ne}}|^2 \quad (\delta_{\text{ne}} := \delta_{\text{tol}})$.
2. Set $\beta := \langle \mathbf{y}, D^\dagger D\mathbf{y} \rangle, \quad \omega := \rho/\beta$.
3. Set $\mathbf{x} := \mathbf{x} + \omega\mathbf{y}, \quad \mathbf{r}_{\text{ne}} := \mathbf{r}_{\text{ne}} - \omega D^\dagger D\mathbf{y}$.
4. Set $\rho' := \rho, \quad \rho := |\mathbf{r}_{\text{ne}}|^2, \quad \theta := -\rho/\rho'$.
5. If $\sqrt{\rho} = |\mathbf{r}_{\text{ne}}| < \delta_{\text{ne}}$ then:
   - If true error $\epsilon := |D\mathbf{x} - \mathbf{b}| < \delta_{\text{tol}}$, we are finished.
   - Otherwise, set $\epsilon' := |D^\dagger D\mathbf{x} - D^\dagger \mathbf{b}|$ and update

$$\delta_{\text{ne}} := \tau \times \delta_{\text{tol}} \times \frac{\epsilon'}{\epsilon},$$

     then restart CGNE (go to 1).
   - $\tau \sim 0.9$ controls restart frequency.

   else:
   - Set $\mathbf{y} := \mathbf{r}_{\text{ne}} - \theta\mathbf{y}$ and continue (go to 2).

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

In-house CUDA quark propagator code.

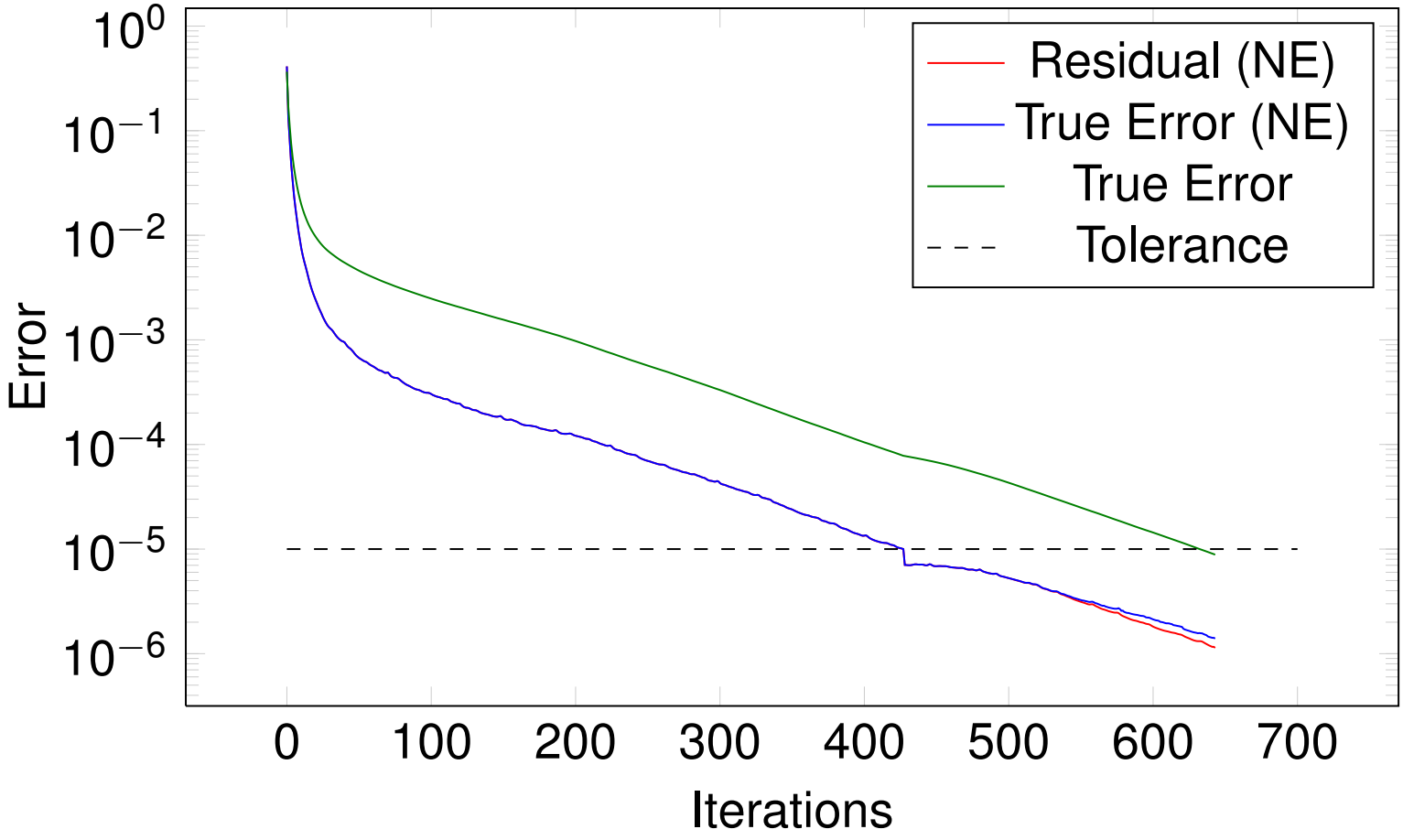
- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

Production Code

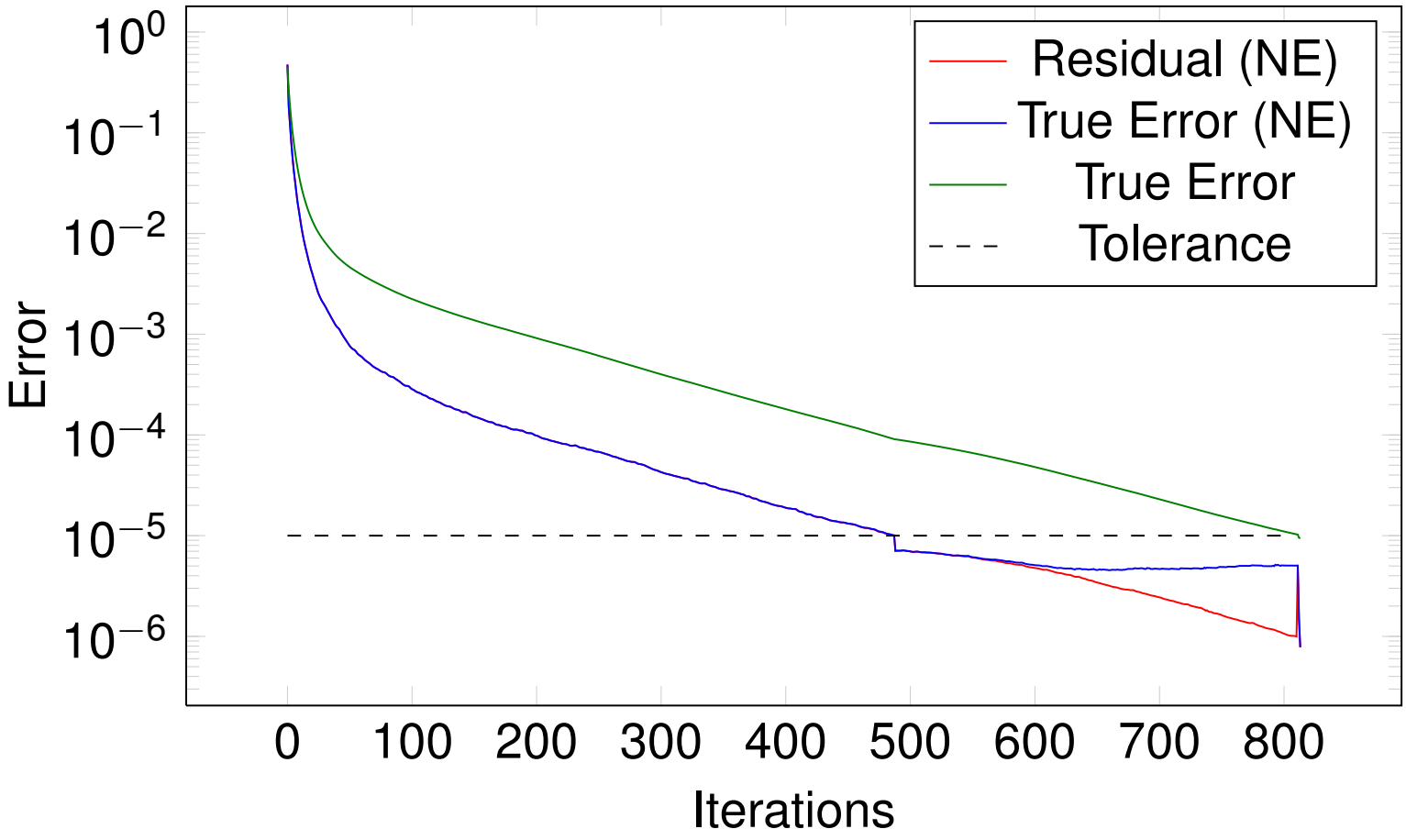
In-house CUDA quark propagator code.

- 8 parameter gauge field reconstruction. ✗
- 12 parameter gauge field reconstruction. ✓
- Half-precision (FP16). ✗
- Fixed precision. ✗
- Dirac basis: Chiral ✓ vs non-relativistic. ✗
- Temporal gauge fixing. ✓
  - + Fixed boundary conditions optimisation. ✓
- Reconstruct uniform background U(1) field. ✓
- CGNE with restarts. ✓
- Linear systems solver sustains above 200 Gflops.

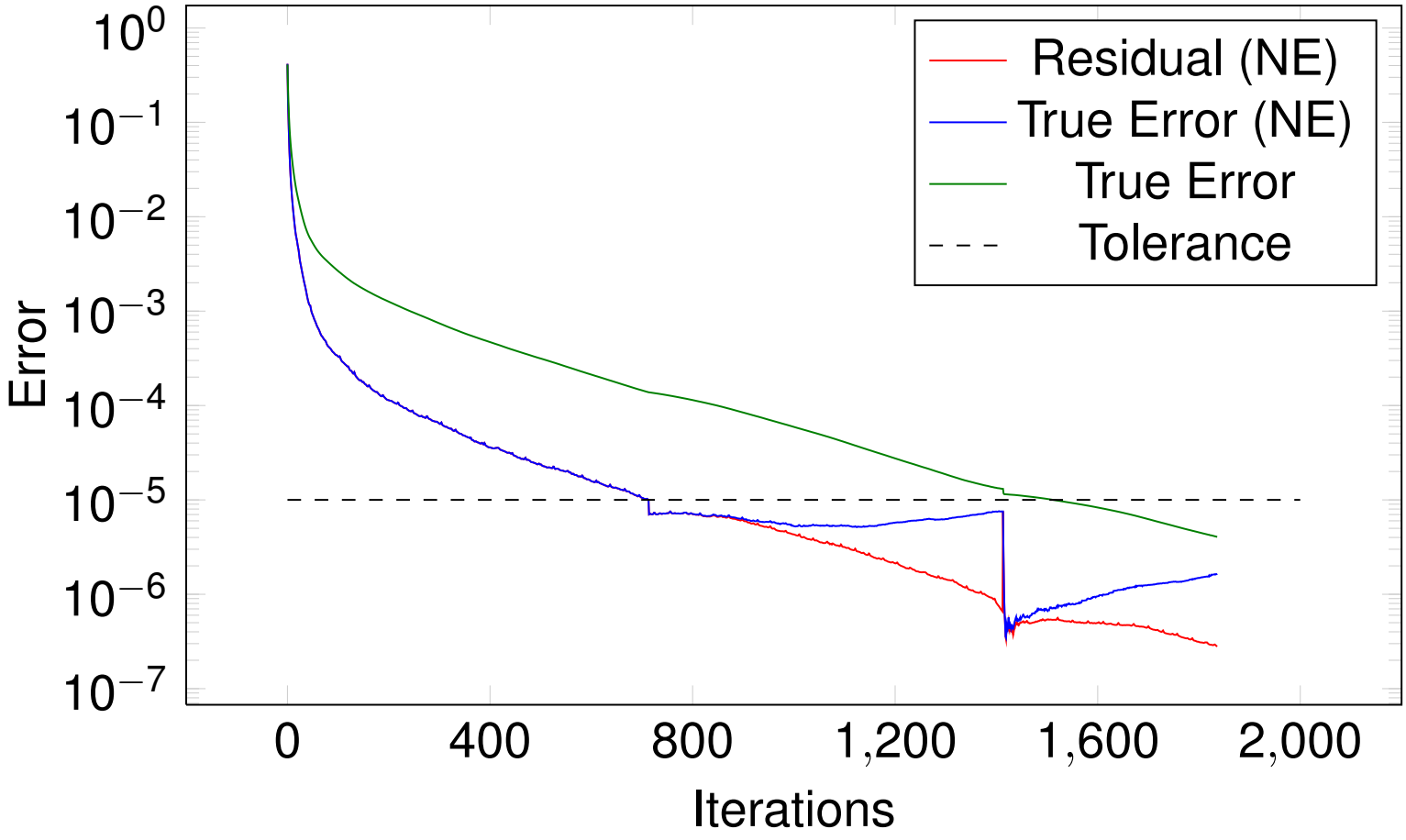
Solver Performance, $\kappa = 0.13700$



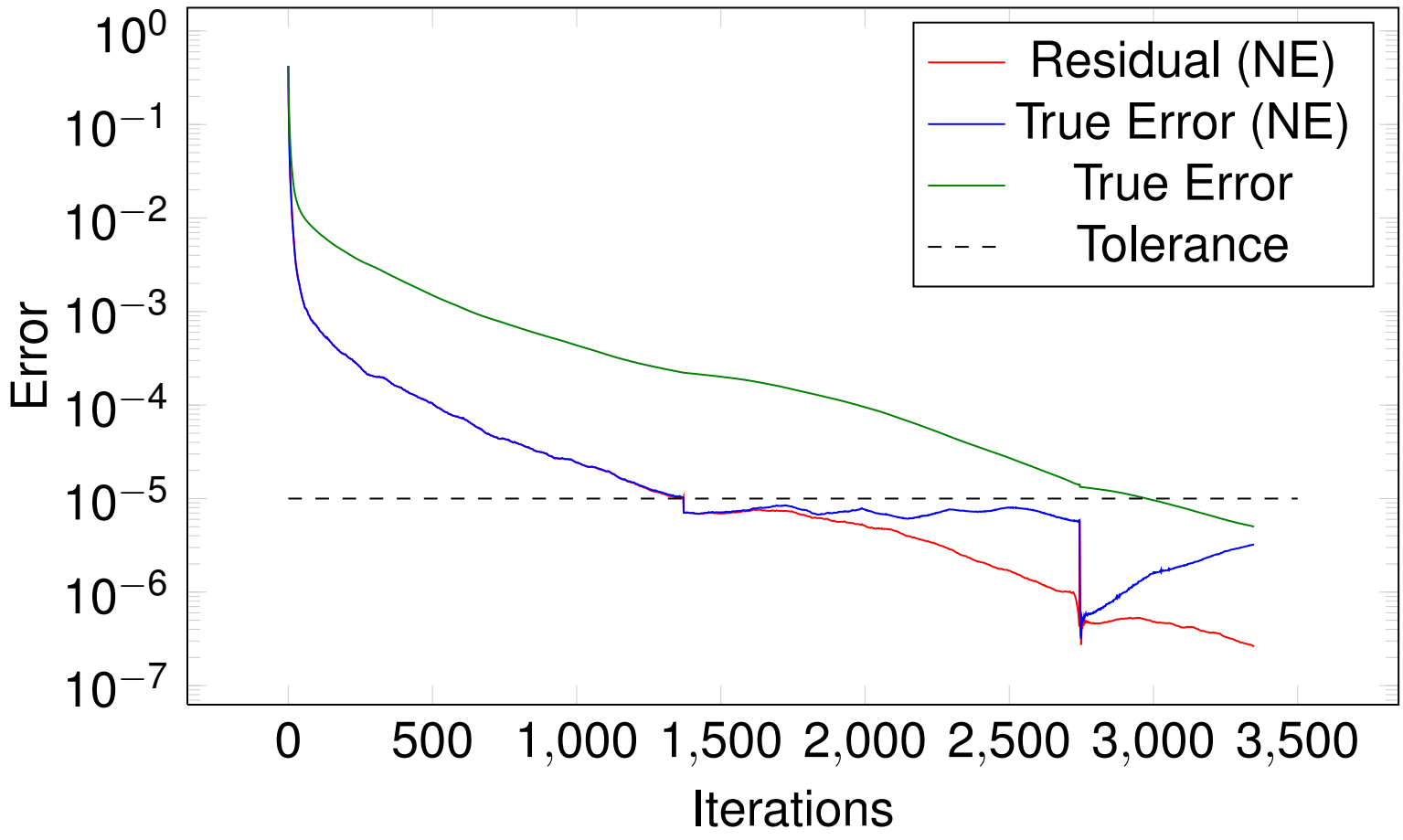
Solver Performance, $\kappa = 0.13727$



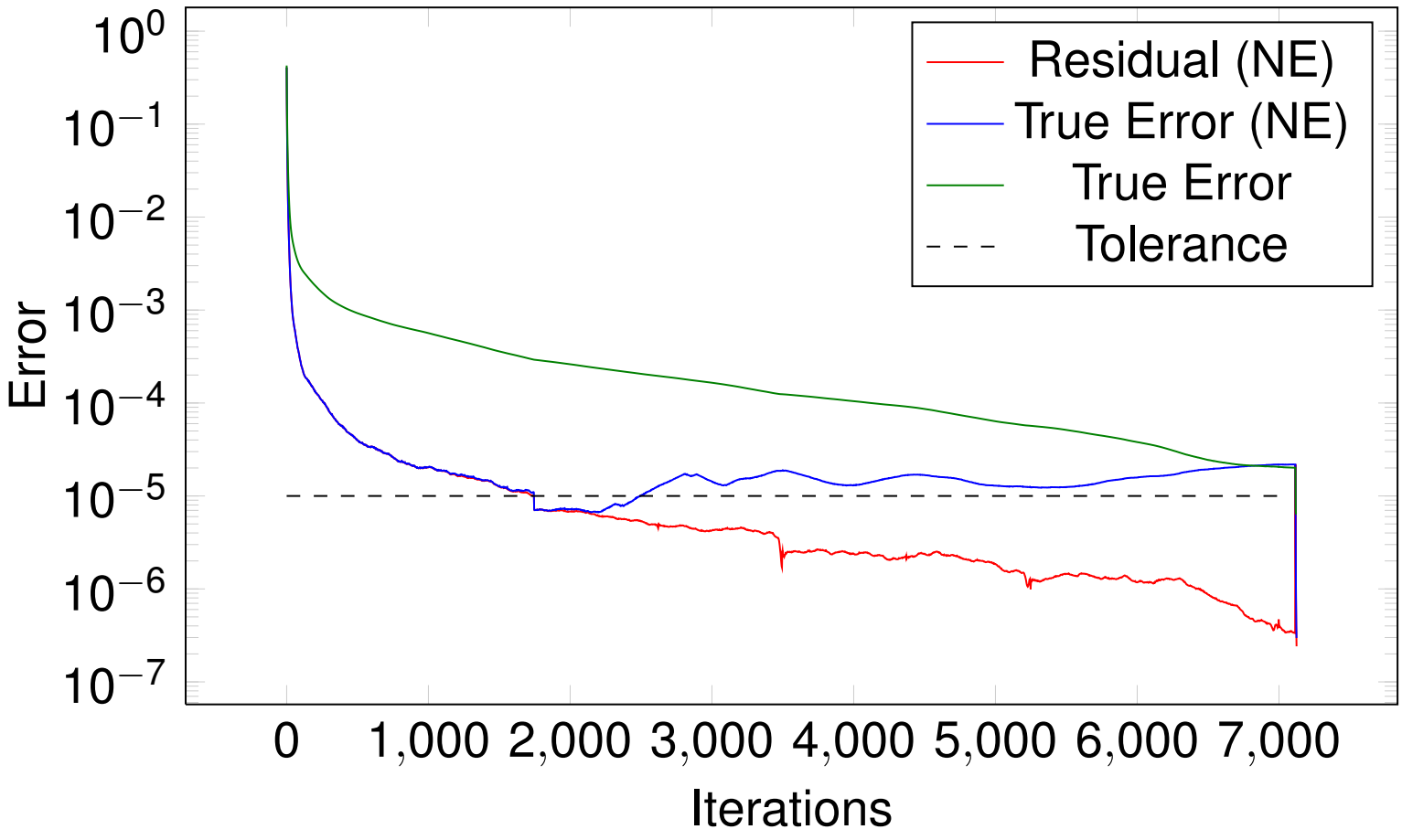
Solver Performance, $\kappa = 0.13754$



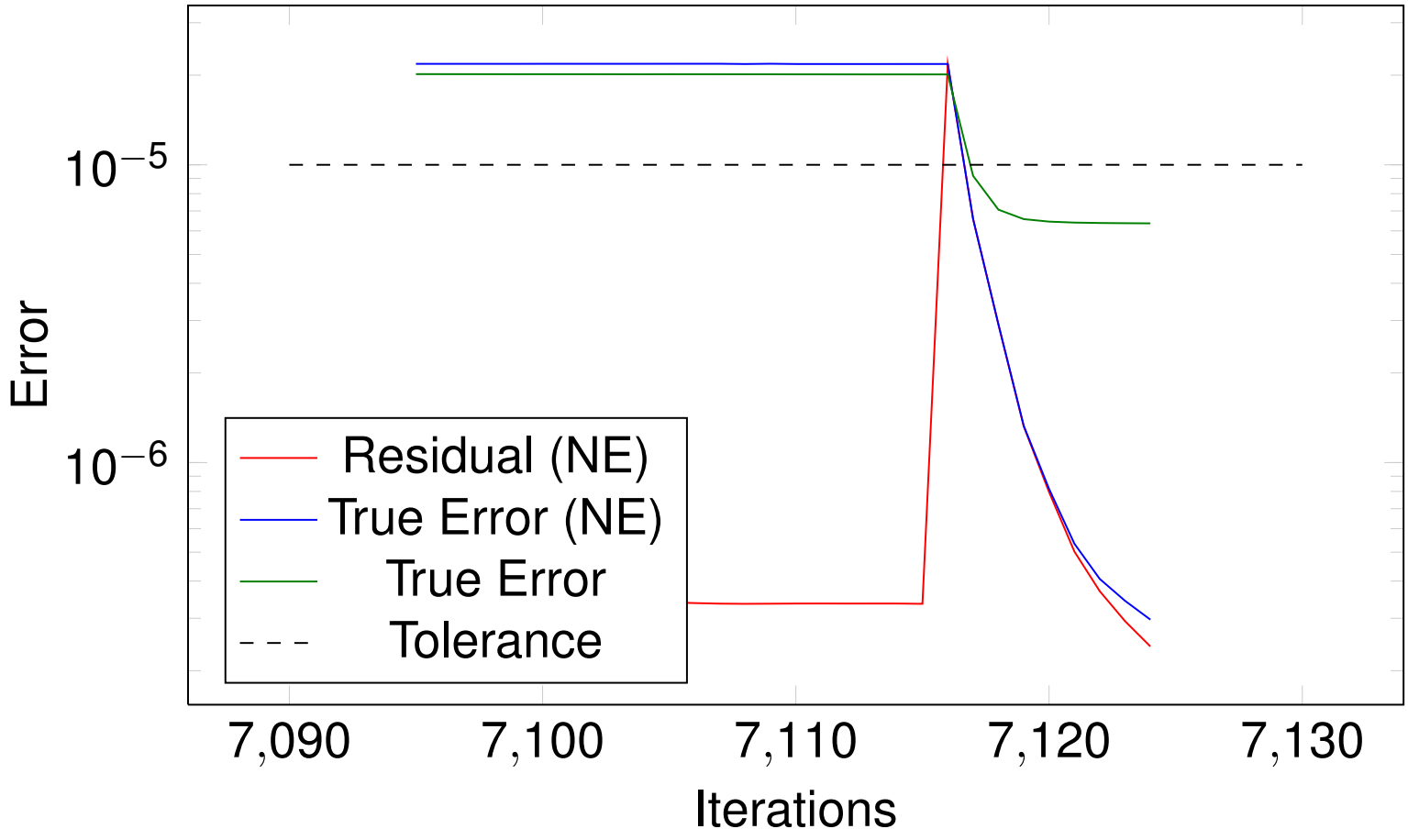
Solver Performance, $\kappa = 0.13770$



Solver Performance, $\kappa = 0.13781$



Solver Performance, $\kappa = 0.13781$



Coming Soon

	Cores	*Peak (SP)*	*Peak (DP)*
GTX 580	512 (Fermi)	1581 Gflops	166 Gflops
Tesla M2090	512 (Fermi)	1331 Gflops	665 Gflops
GTX 680	1536 (Kepler)	3090 Gflops	95 Gflops
Tesla K20	??? (Kepler)	???	$>$ 1 Tflop
Xeon Phi	$>$ 50 (MIC)	???	$>$ 1 Tflop

- Xeon Phi (codenamed Knight's Corner).
- 1 Tflop double precision.
- x86-compatible architecture.
- Reduces code-porting overhead.
- Cost?

Coming Soon

	Cores	*Peak (SP)*	*Peak (DP)*
GTX 580	512 (Fermi)	1581 Gflops	166 Gflops
Tesla M2090	512 (Fermi)	1331 Gflops	665 Gflops
GTX 680	1536 (Kepler)	3090 Gflops	95 Gflops
Tesla K20	??? (Kepler)	???	$>$ 1 Tflop
Xeon Phi	$>$ 50 (MIC)	???	$>$ 1 Tflop

- Xeon Phi (codenamed Knight's Corner).
- 1 Tflop double precision.
- x86-compatible architecture.
- Reduces code-porting overhead.
- Cost?

Coming Soon

	Cores	*Peak (SP)*	*Peak (DP)*
GTX 580	512 (Fermi)	1581 Gflops	166 Gflops
Tesla M2090	512 (Fermi)	1331 Gflops	665 Gflops
GTX 680	1536 (Kepler)	3090 Gflops	95 Gflops
Tesla K20	??? (Kepler)	???	$>$ 1 Tflop
Xeon Phi	$>$ 50 (MIC)	???	$>$ 1 Tflop

- Xeon Phi (codenamed Knight's Corner).
- 1 Tflop double precision.
- x86-compatible architecture.
- Reduces code-porting overhead.
- Cost?

Coming Soon

	Cores	*Peak (SP)*	*Peak (DP)*
GTX 580	512 (Fermi)	1581 Gflops	166 Gflops
Tesla M2090	512 (Fermi)	1331 Gflops	665 Gflops
GTX 680	1536 (Kepler)	3090 Gflops	95 Gflops
Tesla K20	??? (Kepler)	???	$>$ 1 Tflop
Xeon Phi	$>$ 50 (MIC)	???	$>$ 1 Tflop

- Xeon Phi (codenamed Knight's Corner).
- 1 Tflop double precision.
- x86-compatible architecture.
- Reduces code-porting overhead.
- Cost?

Coming Soon

	Cores	*Peak (SP)*	*Peak (DP)*
GTX 580	512 (Fermi)	1581 Gflops	166 Gflops
Tesla M2090	512 (Fermi)	1331 Gflops	665 Gflops
GTX 680	1536 (Kepler)	3090 Gflops	95 Gflops
Tesla K20	??? (Kepler)	???	$>$ 1 Tflop
Xeon Phi	$>$ 50 (MIC)	???	$>$ 1 Tflop

- Xeon Phi (codenamed Knight's Corner).
- 1 Tflop double precision.
- x86-compatible architecture.
- Reduces code-porting overhead.
- Cost?

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.

Summary

- Australian academic supercomputing resources are sparse.
- GeForce cards are a cost-effective alternative for quark propagator calculations.
- Single-precision is enough to solve for the inverse fermion matrix.
- For stability of convergence, a minimal residual solver is best.
- CGNE with restarts works for light quark masses and large lattices.
- Use of accelerators/co-processors is likely to increase in the short term.